

이것이 자바다

3판

최신 버전 반영
JAVA 21

This is
JAVA



교육 현장에서 가장 많이 쓰이는 JAVA 프로그래밍 기본서

신용권, 임경균 지음

무료
특별판

이것이 자바다

3판

무료
특별판



『이것이 자바다(3판)』의

선택/심화 학습을 위해 필요한 내용을 담았습니다.

Appendix 01 | 데이터베이스 입출력(MySQL용)

* Appendix 01은 본책 889쪽 'Chapter 20. 데이터베이스 입출력'의 MySQL용 버전이므로 본책의 세부 목차를 따라 표기합니다.

20.1 JDBC 개요	2
20.2 DBMS 설치	4
20.3 Client Tool 설치	11
20.4 DB 구성	13
20.5 DB 연결	16
20.6 데이터 저장	21
20.7 데이터 수정	28
20.8 데이터 삭제	31
20.9 데이터 읽기	33
20.10 트랜잭션 처리	45
20.11 게시판 구현	50

Appendix 02 | Java UI – Swing

01 Swing 소개	72
02 이벤트 디스패칭 스레드	75
03 Swing 컨테이너	77
04 컴포넌트 배치	92
05 이벤트 처리	111
06 버튼 컴포넌트	122
07 텍스트 컴포넌트	136

08 리스트 컴포넌트	150
09 테이블 컴포넌트	159
10 트리 컴포넌트	179
11 메뉴 컴포넌트	191
12 툴바 컴포넌트	200
13 다이얼로그	205
14 2D 그래픽스	224
15 Swing 과제	244

Appendix 03 | Java UI – JavaFX

01 JavaFX 개요	248
02 JavaFX 프로젝트 생성 및 실행	253
03 JavaFX 레이아웃	261
04 JavaFX 컨테이너	276
05 JavaFX 이벤트 처리	294
06 JavaFX 속성 감시와 바인딩	301
07 JavaFX 컨트롤	312
08 JavaFX 메뉴바와 툴바	355
09 JavaFX 다이얼로그	362
10 JavaFX CSS 스타일	376
11 JavaFX 스레드 UI 변경	403
12 장면 이동과 애니메이션	411
13 JavaFX 과제	427

Appendix 04 | NIO 기반 입출력 및 네트워킹

01 NIO 소개	428
02 파일과 디렉토리	430
03 버퍼	439
04 파일 입출력	460
05 파일 비동기 입출력	467
06 TCP 네트워크 입출력	476
07 TCP 비동기 네트워크 입출력	492
08 UDP 네트워크 입출력	510
09 NIO 과제	515

01

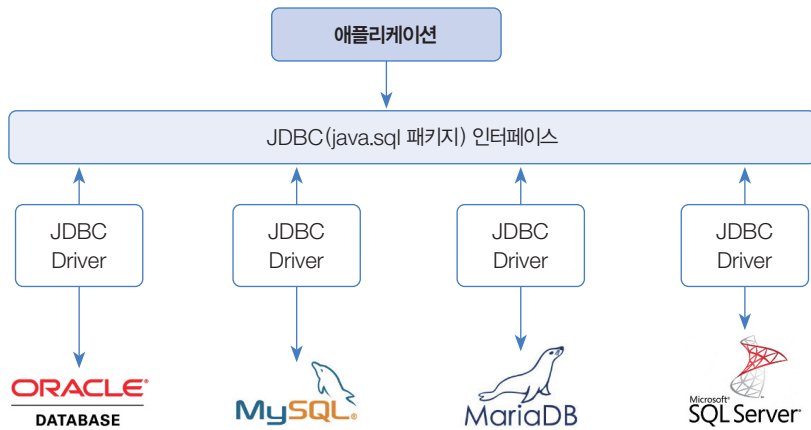
▶ 데이터베이스 입출력 (MySQL용)

- 20.1 JDBC 개요
- 20.2 DBMS 설치
- 20.3 Client Tool 설치
- 20.4 DB 구성
- 20.5 DB 연결
- 20.6 데이터 저장
- 20.7 데이터 수정
- 20.8 데이터 삭제
- 20.9 데이터 읽기
- 20.10 트랜잭션 처리
- 20.11 게시판 구현

* Appendix 01은 본책 889쪽 'Chapter 20. 데이터베이스 입출력'의 MySQL용 버전이므로 본책의 세부 목차를 따라 표기합니다.

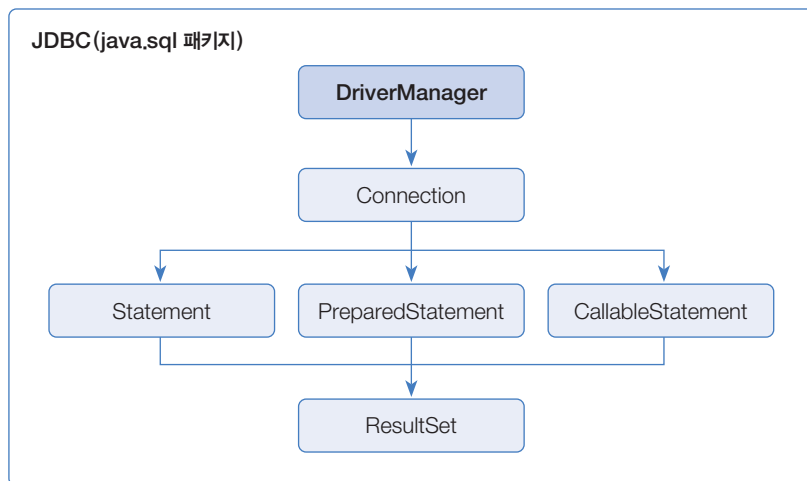
20.1 JDBC 개요

자바는 데이터베이스(DB)와 연결해서 데이터 입출력 작업을 할 수 있도록 JDBC^{Java Database Connectivity} 라이브러리(java.sql 패키지)를 제공한다. JDBC는 데이터베이스 관리시스템(DBMS)의 종류와 상관없이 동일하게 사용할 수 있는 클래스와 인터페이스로 구성되어 있다.



JDBC 인터페이스를 통해 실제로 DB와 작업하는 것은 JDBC Driver이다. JDBC Driver는 JDBC 인터페이스를 구현한 것으로, DBMS마다 별도로 다운로드받아 사용해야 한다.

JDBC에 포함되어 있는 클래스와 인터페이스들의 연관 관계는 다음과 같다.



DriverManager

DriverManager 클래스는 JDBC Driver를 관리하며 DB와 연결해서 Connection 구현 객체를 생성한다.

Connection

Connection 인터페이스는 Statement, PreparedStatement, CallableStatement 구현 객체를 생성하며, 트랜잭션(Transaction) 처리 및 DB 연결을 끊을 때 사용한다.

Statement

Statement 인터페이스는 SQL의 DDL(Data Definition Language)과 DML(Data Manipulation Language)을 실행할 때 사용한다. 주로 변경되지 않는 정적 SQL 문을 실행할 때 사용한다.

PreparedStatement

PreparedStatement는 Statement와 동일하게 SQL의 DDL, DML 문을 실행할 때 사용한다. 차이점은 매개변수화된 SQL 문을 사용할 수 있기 때문에 편리성과 보안성이 좋다. 그래서 Statement 보다는 PreparedStatement를 주로 사용한다.

CallableStatement

CallableStatement는 DB에 저장되어 있는 프로시저(procedure)와 함수(function)를 호출할 때 사용한다.

ResultSet

ResultSet은 DB에서 가져온 데이터를 읽을 때 사용한다.

여기서 잠깐



DBMS별 학습 내용 선택

다음 절부터는 DBMS를 설치하고 JDBC를 사용해서 데이터베이스 연동 프로그램을 어떻게 작성하는지를 학습한다. 여기에서 문제는 DBMS별로 설치 방법과 SQL 문이 다르기 때문에 연동 프로그램 소스가 달라진다는 것이다.

학습용 DBMS로 Oracle을 사용한다면 이 책으로 계속해서 학습하고, MySQL을 사용한다면 부록으로 제공되는 데이터베이스 입출력(MySQL용) PDF로 대체해서 학습하길 바란다.

- Oracle 학습 환경일 경우: 책 본문으로 학습
- MySQL 학습 환경일 경우: 부록으로 제공되는 PDF로 학습

20.2 DBMS 설치

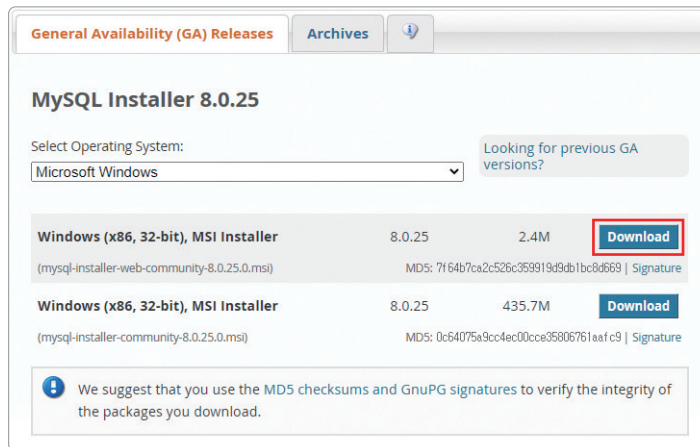
DBMS 마다 조금씩 다른 SQL을 사용하기 때문에 부록에서는 교육 과정에서 많이 사용되는 MySQL을 기준으로 설명한다.

MySQL 설치

윈도우 운영체제에서 MySQL을 설치해 보자.

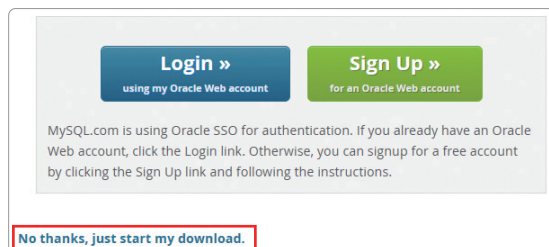
01 MySQL 설치 파일은 다음 URL에 접속해서 다운로드할 수 있다.

<https://dev.mysql.com/downloads/installer>

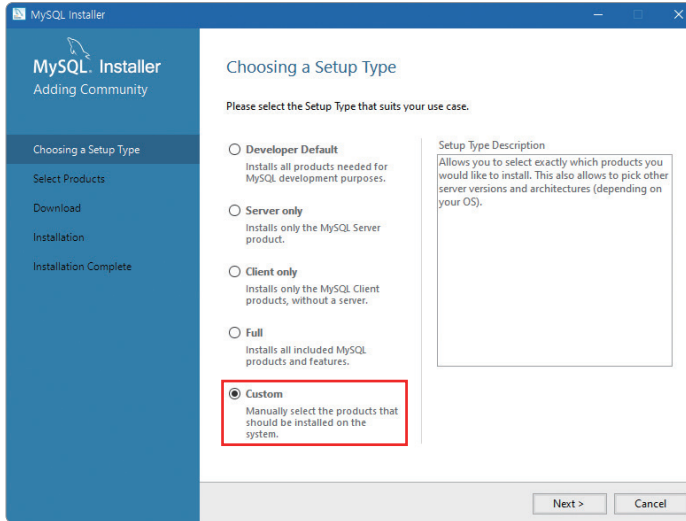


NOTE ▶ MySQL 버전(8.0.x)은 다운로드하는 시점에 따라 달라질 수 있다.

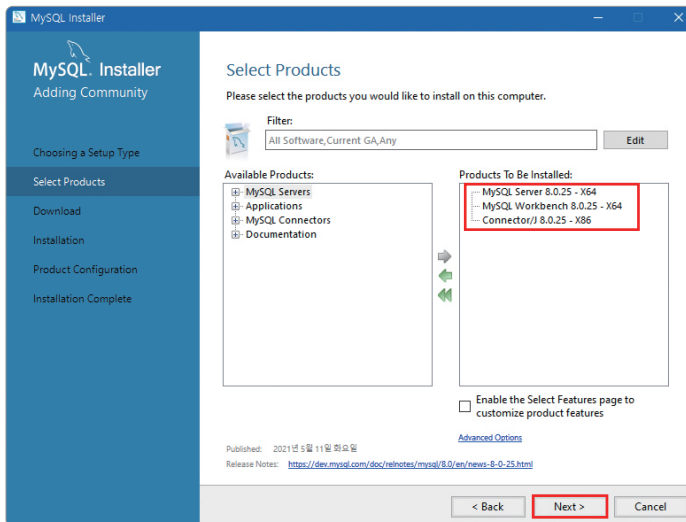
02 [Download] 버튼을 눌러 mysql-installer-web-community-8.0.25.0.msi 파일을 다운로드한다. 다운로드하기 전에 로그인 창이 뜨는데, 로그인할 필요 없이 하단에 'No thanks, just start my download' 링크를 클릭하면 된다.



03 다운로드한 파일을 실행하면 다음과 같이 MySQL Installer 설치 화면이 나타난다. Choosing a Setup Type 단계에서 Installer가 제공하는 SW를 선택적으로 설치하기 위해 'Custom'을 선택한 후 [Next] 버튼을 클릭한다.

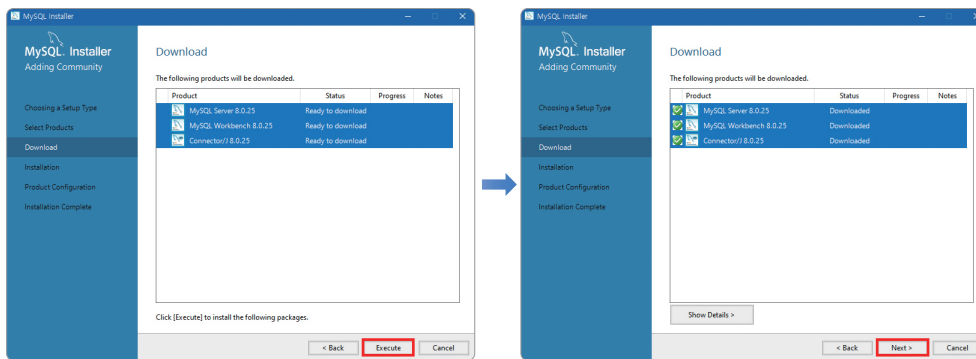


04 Select Products 단계에서 Available Products 목록의 'MySQL Server 8.0.25', 'MySQL Workbench 8.0.25', 'Connector/J 8.0.25'를 선택하고 화살표를 눌러 오른쪽의 Products To Be Installed 목록으로 추가시키고 [Next] 버튼을 클릭한다.

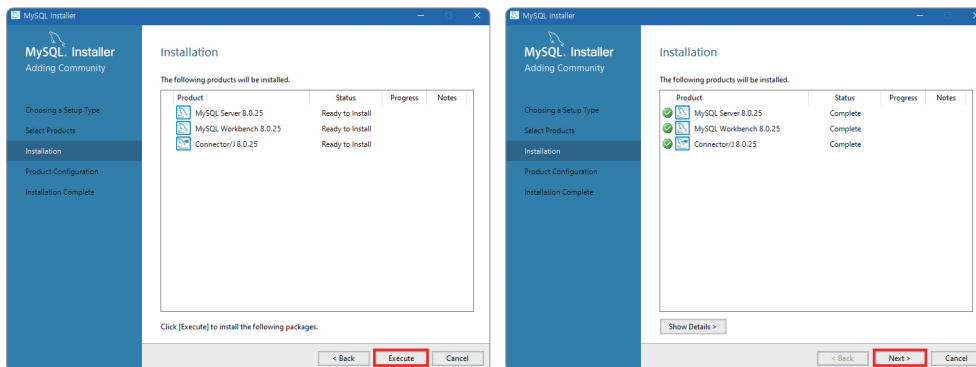


Products	설명
MySQL Server 8.0.25 – X64	DBMS
MySQL Workbench 8.0.25 – X64	DB 관리 및 개발을 위한 Client Tool
Connector/J 8.0.25 – X86	JDBC Driver

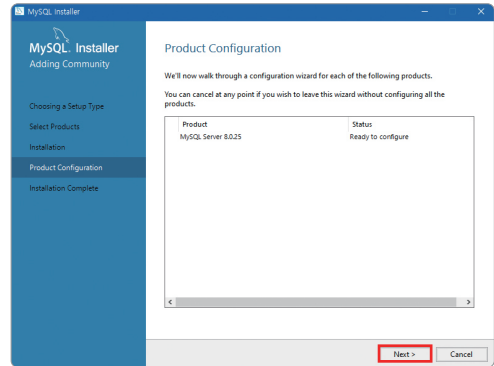
05 선택한 Product를 다운로드하기 위해 [Execute] 버튼을 클릭한다. 다운로드가 완료되면 [Next] 버튼을 클릭한다.



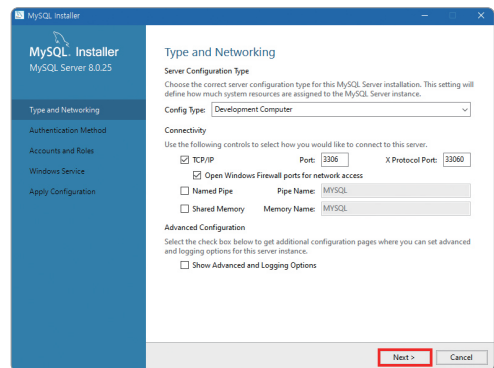
06 Installation 단계에서 설치를 위해 [Execute] 버튼을 클릭한다. 설치가 완료되면 [Next] 버튼을 클릭한다.



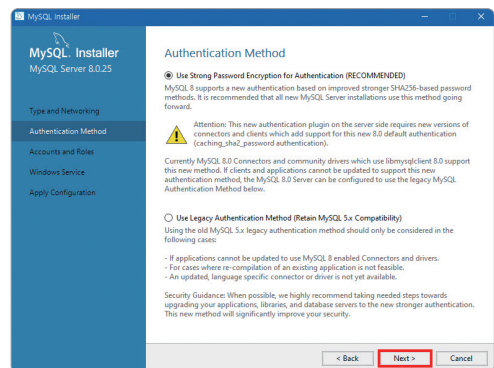
07 DBMS 설정을 위해 Production Configuration 단계에서 [Next] 버튼을 클릭한다.



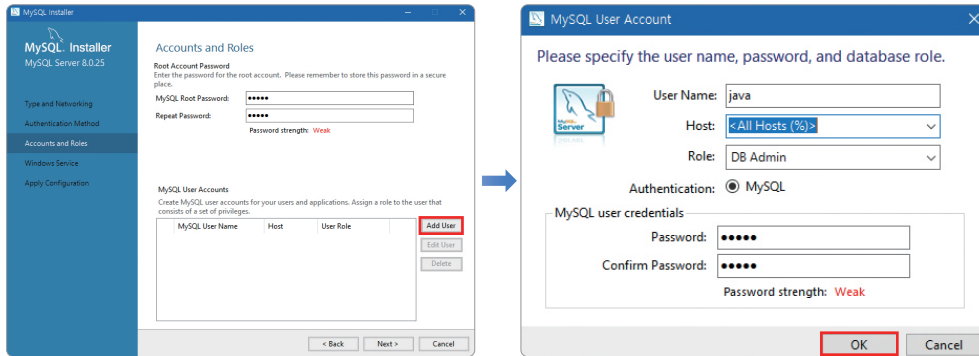
08 Type and Networking 단계에서는 MySQL 서버 구성 타입 및 네트워크 접속 환경을 설정한다. 개발용 PC에 설치하므로 Config Type으로 'Development Computer'를 선택하고, Connectivity에서는 'TCP/IP'의 체크박스에 체크하고 Port는 '3306'으로 설정한다. 나머지는 모두 기본 설정 그대로 두고 [Next] 버튼을 클릭한다.



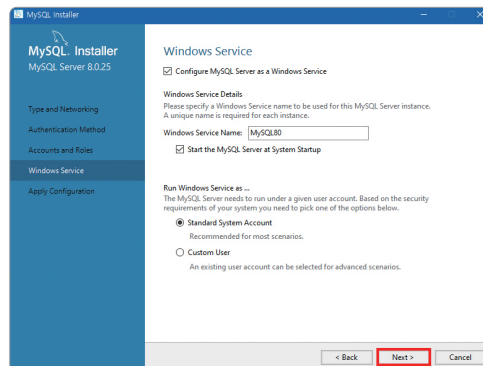
09 Authentication Method 단계에서는 기본으로 체크된 내용을 그대로 두고 [Next] 버튼을 클릭한다.



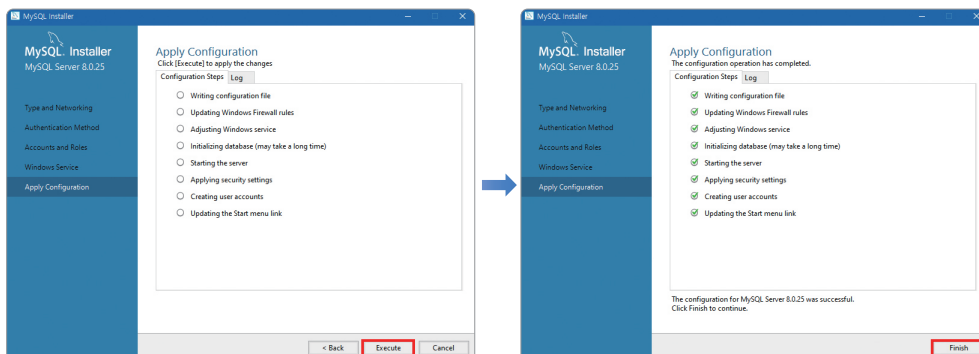
10 Accounts and Roles 단계에서는 관리자 계정인 Root의 비밀번호로 'mysql'을 입력한다. 그리고 사용자 계정을 추가하기 위해 [Add User] 버튼을 클릭하고, User Name에는 'java'를, Password는 'mysql'을 입력해준다. [OK] 버튼을 클릭하고 [Next] 버튼을 클릭한다.



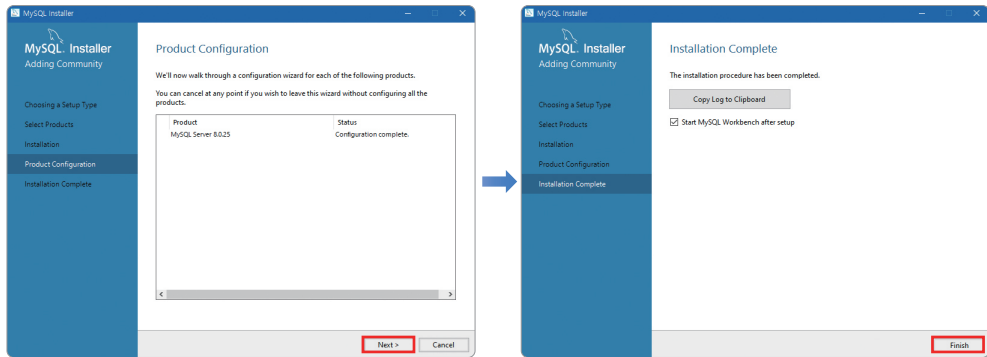
11 Windows Service 단계에서는 윈도우 부팅 시 자동 실행될 수 있도록 기본 내용을 그대로 두고 [Next] 버튼을 클릭한다.



12 Apply Configuration 단계에서는 설치 시 설정해야 할 목록을 보여 준다. [Execute] 버튼을 클릭하고 설정이 끝나면 [Finish] 버튼을 클릭한다.



13 Product Configuration에서 [Next] 버튼을 클릭하고 Install Complete에서 [Finish] 버튼을 클릭하면 설치 과정을 모두 마치게 된다.

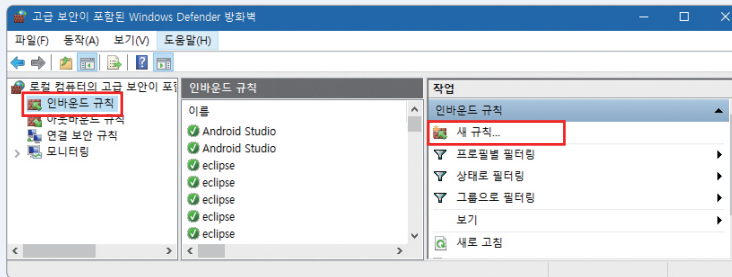


여기서 잠깐

수동으로 3306 포트 개방하기

MySQL은 설치 과정에서 원격으로 접속할 수 있도록 3306 포트를 자동으로 개방 설정하기 때문에 지금부터 하는 작업은 불필요할 수도 있다. 하지만 자동으로 원격 접속이 안 된다면 다음과 같이 수동으로 3306 포트를 개방해야 한다.

01 윈도우 시작 버튼을 클릭한 후 [설정] – [업데이트 및 보안] – [Windows 보안]으로 들어가 [Windows 보안 열기] 버튼을 클릭한다. Windows 보안 대화상자에서 [방화벽 및 네트워크 보안] – [고급 설정]을 클릭하면 [고급 보안이 포함된 Windows Defender 방화벽] 대화상자가 나온다. 여기에서 [인바운드 규칙] – [새 규칙]을 선택한다.



02 새 인바운드 규칙 마법사가 다음과 같이 실행되면 '포트'를 선택하고 [다음] 버튼을 클릭한다.

만들려는 규칙 종류는 무엇입니까?

☐ 프로그램(P)
프로그램의 연결을 제어하는 규칙

☒ 포트(O)
TCP 또는 UDP 포트의 연결을 제어하는 규칙

☐ 미리 정의됨(E):
@FirewallAPI.dll, -80200
Windows 환경의 연결을 제어하는 규칙

☐ 사용자 지정(C)
사용자 지정 규칙

03 'TCP'를 선택하고, 특정 로컬 포트 입력란에 '3306'을 입력한 후 [다음] 버튼을 클릭한다.

이 규칙은 TCP에 적용될까요, UDP에 적용될까요?

☒ TCP(T)
☐ UDP(U)

이 규칙은 모든 로컬 포트에 적용될까요, 특정 로컬 포트에만 적용될까요?

☐ 모든 로컬 포트(A)
☒ 특정 로컬 포트(S): 3306
예: 80, 443, 5000-5010

04 '연결 허용' 선택을 그대로 두고 [다음] 버튼을 클릭한다.

지정된 조건과 연결이 일치할 경우 어떤 작업을 수행해야 할까요?

☒ 연결 허용(A)
IPsec으로 보호되는 연결과 보호되지 않은 연결이 포함됩니다.

☐ 보안 연결만 허용(C)
IPsec을 사용하여 인증된 연결만 포함됩니다. 연결 보안 규칙 노드의 IPsec 속성 및 규칙 설정을 사용하여 연결이 보호됩니다.

☐ 사용자 지정(Z):...

☐ 연결 차단(K)

05 규칙이 적용되는 시기는 모두 체크된 상태로 두고 [다음] 버튼을 클릭한다.

이 규칙이 적용되는 시기는 언제입니까?

☒ 도메인(D)
컴퓨터가 회사 도메인에 연결된 경우 적용됩니다.

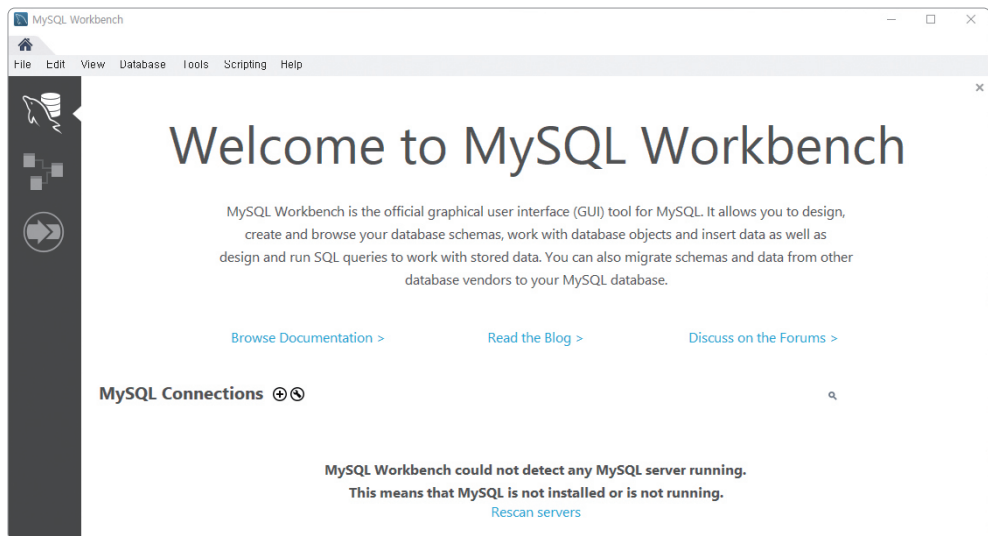
☒ 개인(P)
컴퓨터가 개인 네트워크 위치(가정 또는 직장)에 연결된 경우 적용됩니다.

☒ 공용(U)
컴퓨터가 공용 네트워크 위치에 연결된 경우 적용됩니다.

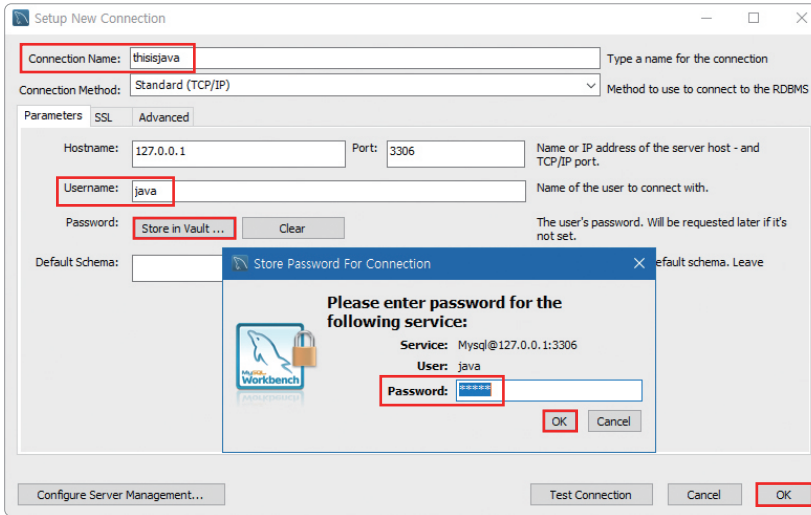
06 규칙 이름 입력란에 'MySQL'이라고 입력하고, [마침] 버튼을 클릭해 닫는다.

20.3 Client Tool 설치

MySQL Workbench는 DB 개발 및 관리에 사용되는 Client Tool이다. MySQL 설치 과정에서 같이 설치되었기 때문에 추가로 설치할 필요가 없다. 설치가 완료되면 자동으로 실행되지만, 시작 메뉴에서 [MySQL] - [MySQL Workbench x.x CE]를 선택해서 실행해도 된다.



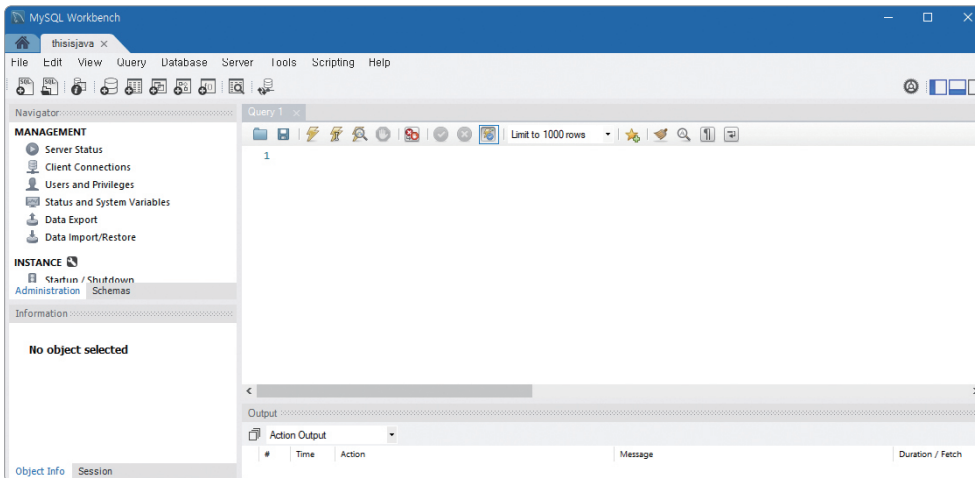
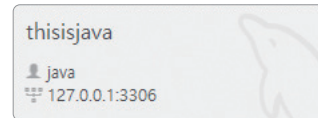
MySQL 설치 과정에서 java 사용자 계정을 추가했기 때문에 사용자 계정을 생성하는 과정은 생략하고 여기에서는 접속 방법만 알아보자. Workbench에서 MySQL Connections 바로 옆의 (+) 아이콘을 클릭한다. Setup New Connection 화면에서 다음과 같이 입력한 후 [OK] 버튼을 클릭한다.



- **Connection Name**: 'thisisjava' 입력
- **Username**: 'java' 입력
- **Password**: [Store in Vault] 버튼 클릭
- **Store Password For Connection 대화상자의 Password**: 'mysql' 입력 – [OK] 버튼 클릭

오른쪽 화면과 같이 연결 정보 박스가 생성되면 마우스로 클릭한다.


다음은 성공적으로 접속된 화면이다.

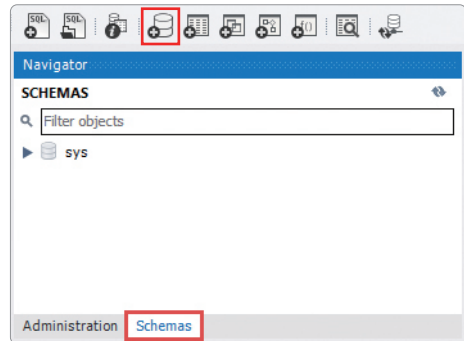


20.4 DB 구성

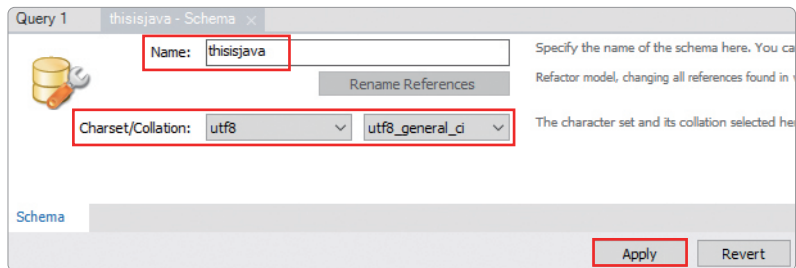
이제 Workbench에서 학습용 DB와 Table을 생성해 보자.

DB 생성

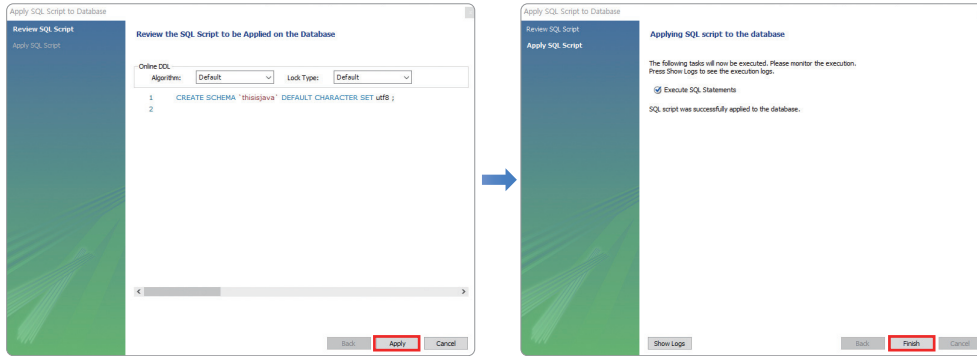
01 MySQL에서는 DB를 스키마^{Schema}라고 부르므로, Navigator 뷰 하단의 [Schemas] 탭을 선택해서 Schema 리스트로 전환한 다음 DB 생성을 위해 툴바에서  아이콘을 클릭한다.



02 Name 입력란에 'thisisjava'라고 입력하고, Charset/Collation에서 드롭다운 버튼을 클릭해 각각 'utf8'과 'utf8_general_ci'로 변경한 다음 [Apply] 버튼을 클릭한다.



03 Review SQL Script 단계에서 [Apply] 버튼을, Apply SQL Script 단계에서 [Finish] 버튼을 차례대로 클릭한다.



04 성공적으로 DB가 생성되면 Navigator 뷰에 thisisjava가 다음과 같이 생성된다. thisisjava를 선택하고 마우스 오른쪽 버튼을 클릭하여 [Set as Default Schema]를 선택해 java 계정의 기본Default 스키마로 설정한다.

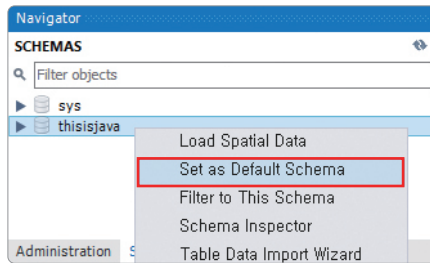
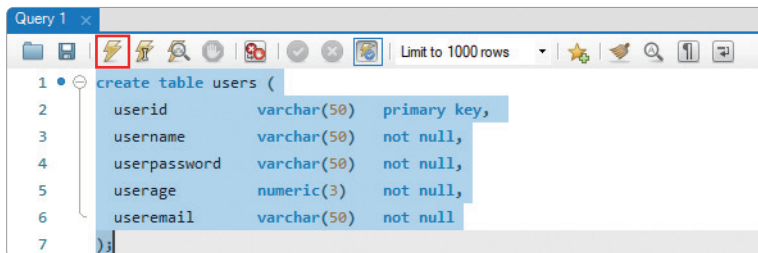


Table 생성

01 사용자 정보가 저장될 users 테이블을 생성하기 위해 예제소스 sql/mysql/users.sql 파일의 텍스트 내용을 복사해 Workbench의 Query 편집기에 붙여 넣는다. 그리고 마우스로 모든 코드를 선택한 다음 실행을 위해 번개 모양 아이콘(⚡)을 클릭한다.



02 게시물 정보가 저장될 boards 테이블을 생성하기 위해 예제소스 sql/mysql/boards.sql 파일의 텍스트 내용을 복사해 Workbench의 Query 편집기에 붙여 넣는다. 그리고 마우스로 모든 코드를 선택한 다음 실행을 위해 번개 모양 아이콘(⚡)을 클릭한다.

```

1 create table boards (
2     bno          int          primary key auto_increment,
3     btitle       varchar(100) not null,
4     bcontent     longtext     not null,
5     bwriter      varchar(50)  not null,
6     bdate        datetime     default now(),
7     bfilename    varchar(50)  null,
8     bfiledata    longblob     null
9 );

```

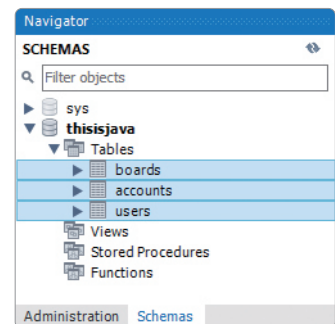
03 계좌 정보가 저장될 accounts 테이블을 생성하기 위해 예제소스 sql/mysql/accounts.sql 파일의 텍스트 내용을 복사해 Workbench의 Query 편집기에 붙여 넣는다. 그리고 create 문에서 insert 문까지 모든 코드를 마우스로 선택한 다음 실행을 위해 번개 모양 아이콘(⚡)을 클릭한다.

```

1 create table accounts (
2     ano          varchar(20)  primary key,
3     owner        varchar(20) not null,
4     balance      numeric      not null
5 );
6
7 insert into accounts (ano, owner, balance)
8 values ('111-111-1111', '하여름', 1000000);
9
10 insert into accounts (ano, owner, balance)
11 values ('222-222-2222', '한겨울', 0);

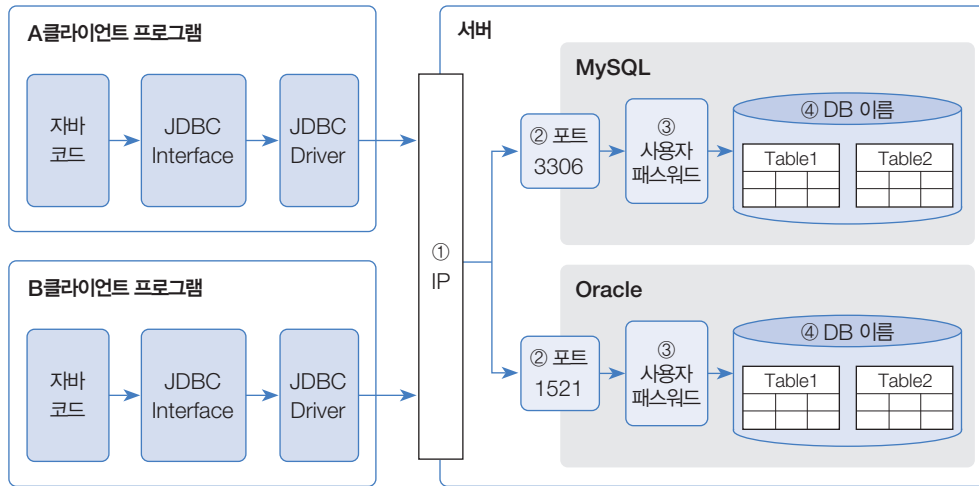
```

04 3개의 테이블이 모두 성공적으로 생성되었다면 SCHEMAS 뷰에서 Tables를 확장했을 때 다음과 같이 테이블 이름들이 보여야 한다. 만약 보이지 않는다면 우측 상단의 아이콘(🔄)을 클릭해서 목록을 갱신시킨다.



20.5 DB 연결

클라이언트 프로그램에서 DB와 연결하려면 해당 DBMS의 JDBC Driver가 필요하다. 또한 연결에 필요한 다음 네 가지 정보가 있어야 한다.



- ① DBMS가 설치된 컴퓨터의 IP 주소
- ② DBMS가 허용하는 포트(Port) 번호
- ③ 사용자(DB 계정) 및 비밀번호
- ④ 사용하고자 하는 DB 이름

IP 주소는 컴퓨터를 찾아가기 위해, Port 번호는 DBMS로 연결하기 위해 필요하다. DBMS는 여러 개의 DB를 관리하므로 실제로 사용할 DB 이름이 필요하며, 어떤 사용자인지 인증받기 위한 계정 및 비밀번호가 필요하다.

JDBC Driver 설치

20.2절에서 로컬 PC에 MySQL을 설치했다면 다음 경로에서 JDBC Driver 파일을 찾을 수 있다.


```
C:\Program Files (x86)\MySQL\Connector J 8.0\mysql-connector-java-8.0.25.jar
```

만약 MySQL을 설치하지 않고 원격 MySQL을 사용한다면 JDBC Driver만 별도로 다음 URL에서 다운로드할 수 있다.

<https://mvnrepository.com/artifact/mysql/mysql-connector-java>

위 사이트에서는 MySQL 버전별로 JDBC Driver를 제공하는데, MySQL 8.0과 호환되는 가장 마지막 버전인 8.0.x를 받아 보자. 다음 그림과 같이 버전 링크를 클릭한다.

Home » mysql » mysql-connector-java

 **MySQL Connector/J**
JDBC Type 4 driver for MySQL

License	GPL 2.0
Categories	MySQL Drivers
Tags	mysql database connector driver
Used By	5,301 artifacts


Central (84) Jahia (1) Redhat GA (4) Redhat EA (1) ICM (10) EBIPublic (6)

Version	Repository	Usages	Date
8.0.25	Central	24	May, 2021
8.0.24	Central	44	Apr, 2021

NOTE ▶ MySQL 버전(8.0.x)은 다운로드하는 시점에 따라 달라질 수 있다.

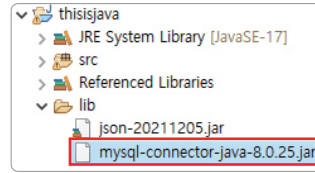
JDBC Driver 라이브러리 파일(jar) 파일을 받기 위해 Files에 있는 jar 링크를 클릭한다.

Home » mysql » mysql-connector-java » 8.0.25

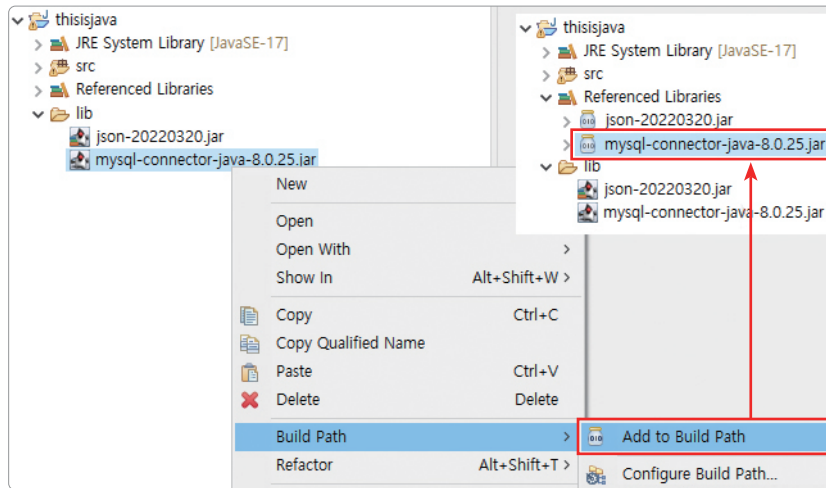
 **MySQL Connector/J » 8.0.25**
JDBC Type 4 driver for MySQL

License	GPL 2.0
Categories	MySQL Drivers
Organization	Oracle Corporation
HomePage	http://dev.mysql.com/doc/connector-j/en/
Date	(May 10, 2021)
Files	jar (2.3 MB) View All
Repositories	Central
Used By	5,301 artifacts

로컬 PC에서 찾았거나 URL에서 내려받은 mysql-connector-java-8.0.x.jar 파일을 thisisjava 프로젝트의 lib 폴더에 복사한다. lib 폴더가 없으면 thisisjava 프로젝트를 마우스 오른쪽 버튼으로 클릭한 후 [New] - [Folder]를 선택해서 생성한다.



그리고 JAR 파일 안에 있는 클래스를 사용하기 위해 JAR 파일을 선택 후 마우스 오른쪽 버튼을 클릭해서 다음과 같이 Build Path에 추가한다.

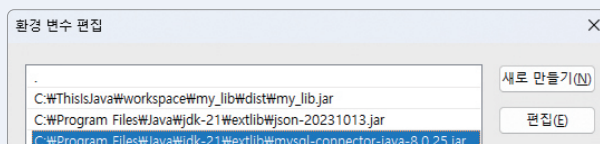


여기서 잠깐

⚙ 환경 변수 CLASSPATH에 JAR 파일 경로 추가하기

명령 프롬프트에서 클라이언트 프로그램을 실행하려면 환경 변수 CLASSPATH에 JDBC Driver JAR 파일 경로를 추가해야 한다. 다음 순서대로 CLASSPATH에 경로를 추가해 보자.

1. C:\Program Files\Java\jdk-21 안에 외부 라이브러리가 저장될 extlib 디렉토리를 생성한다.
2. C:\Program Files\Java\jdk-21\extlib 디렉토리 안에 mysql-connector-java-8.0.25.jar 파일을 저장한다.
3. 환경 변수 CLASSPATH에 다음과 같이 JAR 파일 경로를 추가해 준다.
(주의 첫 줄에는 마침표(.))가 반드시 있어야 함)



DB 연결

클라이언트 프로그램을 DB와 연결하기 위해 가장 먼저 해야 할 작업은 JDBC Driver를 메모리로 로딩하는 것이다. `Class.forName()` 메소드는 문자열로 주어진 JDBC Driver 클래스를 Build Path에서 찾고, 메모리로 로딩한다.

```
Class.forName("com.mysql.cj.jdbc.Driver");
```

이 과정에서 JDBC Driver 클래스의 static 블록이 실행되면서 DriverManager에 JDBC Driver 객체를 등록하게 된다. 만약 Build Path에서 JDBC Driver 클래스를 찾지 못하면 `ClassNotFoundException`이 발생하므로 예외 처리를 해야 한다.

DriverManager에 JDBC Driver가 등록되면 `getConnection()` 메소드로 DB와 연결을 할 수 있다.

```
Connection conn = DriverManager.getConnection("연결 문자열", "사용자", "비밀번호");
```

첫 번째 매개값은 연결 문자열인데, DBMS마다 다른 형식을 가지고 있다. 다음은 MySQL의 연결 문자열을 보여 준다.

`jdbc:mysql://localhost:3306/thisisjava`

↑ ↑ ↑
IP 주소 포트 DB명

localhost는 로컬에 설치된 MySQL에 연결하겠다는 의미이다. 원격 MySQL에 연결하려면 IP 주소로 기술해야 한다. 3306은 Port 번호, thisisjava는 DB명이다.

연결이 성공하면 `getConnection()` 메소드는 `Connection` 객체를 리턴한다. 만약 연결이 실패하면 `SQLException`이 발생하므로 예외 처리를 해야 한다.

다음은 20.4에서 설치한 thisisjava DB에 연결하는 방법을 보여 준다.

>>> ConnectionExample.java

```
1  package ch20.mysql.sec05;
2
3  import java.sql.Connection;
4  import java.sql.DriverManager;
5  import java.sql.SQLException;
6
7  public class ConnectionExample {
8      public static void main(String[] args) {
9          Connection conn = null;
10         try {
11             //JDBC Driver 등록
12             Class.forName("com.mysql.cj.jdbc.Driver");
13
14             //연결하기
15             conn = DriverManager.getConnection(
16                 "jdbc:mysql://localhost:3306/thisisjava",
17                 "java",
18                 "mysql"
19             );
20
21             System.out.println("연결 성공");
22         } catch (ClassNotFoundException e) {
23             e.printStackTrace();
24         } catch (SQLException e) {
25             e.printStackTrace();
26         } finally {
27             if(conn != null) {
28                 try {
29                     //연결 끊기
30                     conn.close();
31                     System.out.println("연결 끊기");
32                 } catch (SQLException e) {}
33             }
34         }
35     }
36 }
```

```

연결 성공
연결 끊기

```

연결 성공했던 DB를 끊을 때에는 Connection 객체의 `close()` 메소드를 호출한다. 이 메소드는 `SQLException`이 발생할 수 있으므로 예외 처리가 필요하다.

20.6 데이터 저장

이번 절에서는 JDBC를 이용해서 INSERT 문을 실행하는 방법을 알아보자. users 테이블에 새로운 사용자 정보를 저장하는 INSERT 문은 다음과 같다.

```

INSERT INTO users (userid, username, userpassword, userage, useremail)
VALUES ('winter', '한겨울', '12345', 25, 'winter@mycompany.com')

```

값을 ?(물음표)로 대체한 매개변수화된 INSERT 문으로 변경하면 다음과 같다.

```

INSERT INTO users (userid, username, userpassword, userage, useremail)
VALUES (?, ?, ?, ?, ?)

```

그리고 INSERT 문을 String 타입 변수 sql에 문자열로 대입한다.

```

String sql = new StringBuilder()
    .append("INSERT INTO users (userid, username, userpassword, userage, useremail) ")
    .append("VALUES (?, ?, ?, ?, ?)")
    .toString();

```

또는

```

String sql = "" +
    "INSERT INTO users (userid, username, userpassword, userage, useremail) " +
    "VALUES (?, ?, ?, ?, ?)";

```

매개변수화된 SQL 문을 실행하려면 PreparedStatement가 필요하다. 다음과 같이 Connection의 prepareStatement() 메소드로부터 PreparedStatement를 얻는다.

```
PreparedStatement pstmt = conn.prepareStatement(sql);
```

그리고 ?에 들어갈 값을 지정해주는데, ?는 순서에 따라 1번부터 번호가 부여된다. 값의 타입에 따라 Setter 메소드를 선택한 후 첫 번째에는 ? 순번, 두 번째에는 값을 지정해 준다.

```
pstmt.setString(1, "winter");  
pstmt.setString(2, "한겨울");  
pstmt.setString(3, "12345");  
pstmt.setInt(4, 25);  
pstmt.setString(5, "winter@mycompany.com");
```

값을 지정한 후 executeUpdate() 메소드를 호출하면 SQL문이 실행되면서 users 테이블에 1개의 행이 저장된다. executeUpdate() 메소드가 리턴하는 값은 저장된 행 수인데, 정상적으로 실행 되었을 경우 1을 리턴한다.

```
int rows = pstmt.executeUpdate();
```

PreparedStatement를 더 이상 사용하지 않을 경우에는 close() 메소드를 호출해서 PreparedStatement가 사용했던 메모리를 해제시킨다.

```
pstmt.close();
```

다음 예제는 users 테이블에 사용자 정보를 저장하는 전체 코드를 보여 준다.

>> UserInsertExample.java

```
1  package ch20.mysql.sec06
2
3  import java.sql.Connection;
4  import java.sql.DriverManager;
5  import java.sql.PreparedStatement;
6  import java.sql.SQLException;
7
8  public class UserInsertExample {
9      public static void main(String[] args) {
10         Connection conn = null;
11         try {
12             //JDBC Driver 등록
13             Class.forName("com.mysql.cj.jdbc.Driver");
14
15             //연결하기
16             conn = DriverManager.getConnection(
17                 "jdbc:mysql://localhost:3306/thisisjava",
18                 "java",
19                 "mysql"
20             );
21
22             //매개변수화된 SQL 문 작성
23             String sql = "" +
24                 "INSERT INTO users (userid, username, userpassword, userage, " +
25                 "useremail) " +
26                 "VALUES (?, ?, ?, ?, ?)";
27
28             //PreparedStatement 얻기 및 값 지정
29             PreparedStatement pstmt = conn.prepareStatement(sql);
30             pstmt.setString(1, "winter");
31             pstmt.setString(2, "한겨울");
32             pstmt.setString(3, "12345");
33             pstmt.setInt(4, 25);
34             pstmt.setString(5, "winter@mycompany.com");
35
36             //SQL 문 실행
37             int rows = pstmt.executeUpdate();
38             System.out.println("저장된 행 수: " + rows);
39         } catch (Exception e) {
40             e.printStackTrace();
41         } finally {
42             if (conn != null) {
43                 conn.close();
44             }
45         }
46     }
47 }
```



```

38
39         //PreparedStatement 닫기
40         pstmt.close();
41     } catch (ClassNotFoundException e) {
42         e.printStackTrace();
43     } catch (SQLException e) {
44         e.printStackTrace();
45     } finally {
46         if(conn != null) {
47             try {
48                 //연결 끊기
49                 conn.close();
50             } catch (SQLException e) {}
51         }
52     }
53 }
54 }

```

실행 결과

저장된 행 수: 1

이번에는 boards 테이블에 게시물 정보를 저장해 보자. 새로운 게시물 정보를 저장하는 INSERT 문은 다음과 같다. bno는 자동 증가 컬럼이므로 생략되고, now()는 현재 시간이다.

```

INSERT INTO boards (btitle, bcontent, bwriter, bdate, bfilename, bfiledata)
VALUES ('눈 오는 날', '함박눈이 내려요', 'winter', now(), 'snow.jpg', binaryData)

```

now()를 제외하고 나머지는 ?로 대체한 매개변수화된 INSERT 문으로 만들고, String 타입 변수 sql에 저장한다.

```

String sql = "" +
    "INSERT INTO boards (btitle, bcontent, bwriter, bdate, bfilename, bfiledata) " +
    "VALUES (?, ?, ?, now(), ?, ?)";

```

매개변수화된 INSERT 문을 실행하기 위해 다음과 같이 `prepareStatement()` 메소드로부터 `PreparedStatement`를 얻는데, 이전과는 다르게 두 번째 매개값이 있다.

```
PreparedStatement pstmt = conn.prepareStatement(sql,  
Statement.RETURN_GENERATED_KEYS);
```

두 번째 매개값은 INSERT 문이 실행된 후 가져올 키 값으로, 자동 증가된 `bno` 값을 가져온다. SQL 문이 실행되기 전까지는 `bno` 값을 모르기 때문에 SQL 문이 실행된 후에 `bno` 컬럼에 실제로 저장된 값을 얻어보는 것이다.

이제 ?에 해당하는 값을 지정한다. `bfiledata` 컬럼은 바이너리 타입(blob)이므로 ?에 값을 지정하려면 `setBinaryStream()`, `setBlob()`, `setBytes()` 메소드 중 하나를 이용해야 한다. 다음은 `setBlob`을 이용해서 바이트 입력 스트림을 제공한 것이다.

```
pstmt.setString(1, "눈 오는 날");  
pstmt.setString(2, "함박눈이 내려요.");  
pstmt.setString(3, "winter");  
pstmt.setString(4, "snow.jpg");  
pstmt.setBlob(5, new FileInputStream("src/ch20/mysql/sec06/snow.jpg"));
```

INSERT 문을 실행하고 저장된 `bno` 값을 얻는 방법은 다음과 같다. 게시물 정보가 저장되었을 경우(rows가 1일 경우) `getGeneratedKeys()` 메소드로 `ResultSet`을 얻고, `getInt()` 메소드로 `bno`를 얻는다. `ResultSet`에 대해서는 20.9절에서 자세히 설명한다.

```
int rows = pstmt.executeUpdate();           //SQL 문 실행  
if(rows == 1) {  
    ResultSet rs = pstmt.getGeneratedKeys(); //new String[] { "bno" }에 기술된 컬럼  
                                           값을 가져옴  
    if(rs.next()) {                         //값이 있다면  
        int bno = rs.getInt(1); //new String[] { "bno" }의 첫 번째 항목 bno 컬럼 값을 읽음  
    }  
    rs.close();                             //ResultSet이 사용했던 메모리 해제  
}
```

다음은 boards 테이블에 게시물 정보를 저장하는 전체 코드이다.

>>> BoardInsertExample.java

```
1  package ch20.mysql.sec06;
2
3  import java.io.FileInputStream;
4  import java.sql.Connection;
5  import java.sql.DriverManager;
6  import java.sql.PreparedStatement;
7  import java.sql.ResultSet;
8  import java.sql.SQLException;
9  import java.sql.Statement;
10
11 public class BoardInsertExample {
12     public static void main(String[] args) {
13         Connection conn = null;
14         try {
15             //JDBC Driver 등록
16             Class.forName("com.mysql.cj.jdbc.Driver");
17
18             //연결하기
19             conn = DriverManager.getConnection(
20                 "jdbc:mysql://localhost:3306/thisisjava",
21                 "java",
22                 "mysql"
23             );
24
25             //매개변수화된 SQL 문 작성
26             String sql = "" +
27                 "INSERT INTO boards (btitle, bcontent, bwriter, bdate, bfilename,
28                     bfiledata) " +
29                 "VALUES (?, ?, ?, now(), ?, ?)";
30
31             //PreparedStatement 얻기 및 값 지정
32             PreparedStatement pstmt = conn.prepareStatement(
33                 sql, Statement.RETURN_GENERATED_KEYS);
34             pstmt.setString(1, "눈 오는 날");
35             pstmt.setString(2, "함박눈이 내려요.");
36             pstmt.setString(3, "winter");
```

```

36      pstmt.setString(4, "snow.jpg");
37      pstmt.setBlob(5, new FileInputStream("src/ch20/mysql/sec06/snow.jpg"));
38
39      //SQL 문 실행
40      int rows = pstmt.executeUpdate();
41      System.out.println("저장된 행 수: " + rows);
42
43      //bno 값 얻기
44      if(rows == 1) {
45          ResultSet rs = pstmt.getGeneratedKeys();
46          if(rs.next()) {
47              int bno = rs.getInt(1);
48              System.out.println("저장된 bno: " + bno);
49          }
50          rs.close();
51      }
52
53      //PreparedStatement 닫기
54      pstmt.close();
55      } catch (Exception e) {
56          e.printStackTrace();
57      } finally {
58          if(conn != null) {
59              try {
60                  //연결 끊기
61                  conn.close();
62              } catch (SQLException e) {}
63          }
64      }
65  }
66  }

```

실행 결과

저장된 행 수: 1

저장된 bno: 1 (실행한 횟수에 따라 bno 값은 다를 수 있음)

20.7 데이터 수정

이번 절에서는 JDBC를 이용해서 UPDATE 문을 실행하는 방법을 알아보자. boards 테이블에 저장된 게시물 중에서 bno가 3인 게시물의 btitle, bcontent, bfilename, bfiledata를 변경하는 SQL 문은 다음과 같다.

```
UPDATE boards SET
  btitle='눈사람',
  bcontent='눈으로 만든 사람;',
  bfilename='snowman.jpg',
  bfiledata=binaryData
WHERE bno=3
```

값을 ?로 대체한 매개변수화된 UPDATE 문으로 변경한다.

```
UPDATE boards SET
  btitle=?,
  bcontent=?,
  bfilename=?,
  bfiledata=?
WHERE bno=?
```

String 타입 변수 sql에 매개변수화된 UPDATE 문을 저장한다.

```
String sql = new StringBuilder()
    .append("UPDATE boards SET ")
    .append("btitle=?, ")
    .append("bcontent=?, ")
    .append("bfilename=?, ")
    .append("bfiledata=? ")
    .append("WHERE bno=?")
    .toString();
```

매개변수화된 UPDATE 문을 실행하기 위해 다음과 같이 preparedStatement() 메소드로부터 PreparedStatement를 얻고, ?에 해당하는 값을 지정한다.

```

PreparedStatement pstmt = conn.prepareStatement(sql);
pstmt.setString(1, "눈사람");
pstmt.setString(2, "눈으로 만든 사람");
pstmt.setString(3, "snowman.jpg");
pstmt.setBlob(4, new FileInputStream("src/ch20/mysql/sec07/snowman.jpg"));
pstmt.setInt(5, 3);

```

값을 모두 지정하였다면 UPDATE 문을 실행하기 위해 executeUpdate() 메소드를 호출한다. 성공적으로 실행되면 수정된 행의 수가 리턴된다. 만약 0이 리턴되면 조건에 맞는 행이 없어 수정된 내용이 없음을 의미한다.

```
int rows = pstmt.executeUpdate();
```

다음은 boards 테이블에 저장된 게시물 정보를 수정하는 전체 코드이다. 39라인의 게시물 번호 3은 여러분의 boards 테이블에 저장된 번호로 알맞게 수정해야 한다.

» BoardUpdateExample.java

```

1  package ch20.mysql.sec07;
2
3  import java.io.FileInputStream;
4  import java.sql.Connection;
5  import java.sql.DriverManager;
6  import java.sql.PreparedStatement;
7  import java.sql.SQLException;
8
9  public class BoardUpdateExample {
10     public static void main(String[] args) {
11         Connection conn = null;
12         try {
13             //JDBC Driver 등록
14             Class.forName("com.mysql.cj.jdbc.Driver");
15
16             //연결하기
17             conn = DriverManager.getConnection(

```

```

18         "jdbc:mysql://localhost:3306/thisisjava",
19         "java",
20         "mysql"
21     );
22
23     //매개변수화된 SQL 문 작성
24     String sql = new StringBuilder()
25         .append("UPDATE boards SET ")
26         .append("btitle=?, ")
27         .append("bcontent=?, ")
28         .append("bfilename=?, ")
29         .append("bfiledata=? ")
30         .append("WHERE bno=?")
31         .toString();
32
33     //PreparedStatement 얻기 및 값 지정
34     PreparedStatement pstmt = conn.prepareStatement(sql);
35     pstmt.setString(1, "눈사람");
36     pstmt.setString(2, "눈으로 만든 사람");
37     pstmt.setString(3, "snowman.jpg");
38     pstmt.setBlob(4, new FileInputStream("src/ch20/mysql/sec07/snowman.
39         jpg"));
40     pstmt.setInt(5, 3); //boards 테이블에 있는 게시물 번호(bno) 지정
41
42     //SQL 문 실행
43     int rows = pstmt.executeUpdate();
44     System.out.println("수정된 행 수: " + rows);
45
46     //PreparedStatement 닫기
47     pstmt.close();
48     } catch (Exception e) {
49         e.printStackTrace();
50     } finally {
51         if(conn != null) {
52             try {
53                 //연결 끊기
54                 conn.close();
55             } catch (SQLException e) {}
56         }
57     }

```

```
57     }  
58 }
```

실행 결과

수정된 행 수: 1

20.8 데이터 삭제

이번 절에서는 JDBC를 이용해서 DELETE 문을 실행하는 방법을 알아보자. boards 테이블에서 bwriter가 winter인 모든 게시물을 삭제하는 DELETE 문은 다음과 같다.

```
DELETE FROM boards WHERE bwriter='winter'
```

조건절의 값을 ?로 대체한 매개변수화된 DELETE 문으로 변경한다.

```
DELETE FROM boards WHERE bwriter=?
```

매개변수화된 DELETE 문을 String 타입 변수 sql에 대입한다.

```
String sql = "DELETE FROM boards WHERE bwriter=?";
```

매개변수화된 DELETE 문을 실행하기 위해 preparedStatement() 메소드로부터 PreparedStatement를 얻고 ?에 값을 지정한 후, executeUpdate로 SQL 문을 실행한다. 리턴 값은 삭제된 행 수이다.

```
String sql = "DELETE FROM boards WHERE bwriter=?";  
PreparedStatement pstmt = conn.prepareStatement(sql);  
pstmt.setString(1, "winter");  
int rows = pstmt.executeUpdate();
```


다음은 boards 테이블에 저장된 게시물 정보를 삭제하는 전체 코드이다.

>>> BoardDeleteExample.java

```
1  package ch20.mysql.sec08;
2
3  import java.sql.Connection;
4  import java.sql.DriverManager;
5  import java.sql.PreparedStatement;
6  import java.sql.SQLException;
7
8  public class BoardDeleteExample {
9      public static void main(String[] args) {
10         Connection conn = null;
11         try {
12             //JDBC Driver 등록
13             Class.forName("com.mysql.cj.jdbc.Driver");
14
15             //연결하기
16             conn = DriverManager.getConnection(
17                 "jdbc:mysql://localhost:3306/thisisjava",
18                 "java",
19                 "mysql"
20             );
21
22             //매개변수화된 SQL 문 작성
23             String sql = "DELETE FROM boards WHERE bwriter=?";
24
25             //PreparedStatement 얻기 및 값 지정
26             PreparedStatement pstmt = conn.prepareStatement(sql);
27             pstmt.setString(1, "winter");
28
29             //SQL 문 실행
30             int rows = pstmt.executeUpdate();
31             System.out.println("삭제된 행 수: " + rows);
32
33             //PreparedStatement 닫기
34             pstmt.close();
35         } catch (Exception e) {
36             e.printStackTrace();
37         } finally {
```

```

38         if(conn != null) {
39             try {
40                 //연결 끊기
41                 conn.close();
42             } catch (SQLException e) {}
43         }
44     }
45 }
46 }

```

실행 결과

삭제된 행 수: 10 (winter가 작성한 게시물 개수에 따라 삭제된 행 수는 다를 수 있음)

20.9 데이터 읽기

PreparedStatement를 생성할 때 SQL 문이 INSERT, UPDATE, DELETE일 경우에는 executeUpdate() 메소드를 호출하지만, 데이터를 가져오는 SELECT 문일 경우에는 executeQuery() 메소드를 호출해야 한다. executeQuery() 메소드는 가져온 데이터를 ResultSet에 저장하고 리턴한다.

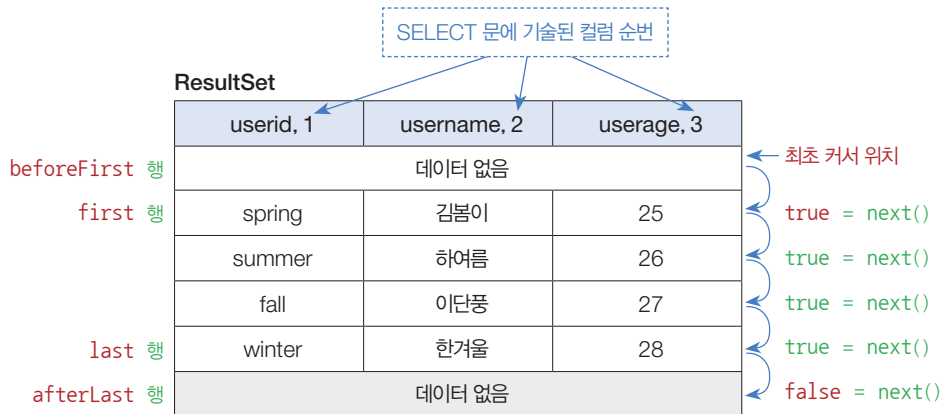
```
ResultSet rs = pstmt.executeQuery();
```

ResultSet 구조

ResultSet은 SELECT 문에 기술된 컬럼으로 구성된 행^{row}의 집합이다. 예를 들어 다음 SELECT 문은 userid, username, userage 컬럼으로 구성된 ResultSet을 리턴한다.

```
SELECT userid, username, userage FROM users
```

위의 SELECT 문이 가져온 데이터 행이 4개라면 ResultSet의 내부 구조는 다음과 같다.



ResultSet의 특징은 커서(cursor)가 있는 행의 데이터만 읽을 수 있다는 것이다. 여기서 커서는 행을 가리키는 포인터를 말한다. ResultSet은 실제 가져온 데이터 행의 앞과 뒤에 beforeFirst 행과 afterLast 행이 붙는데, 최초 커서는 beforeFirst를 가리킨다. 따라서 첫 번째 데이터 행인 first 행을 읽으려면 커서를 이동시켜야 한다. 이때 next() 메소드를 사용한다.

```
boolean result = rs.next();
```

next() 메소드는 커서를 다음 행으로 이동시키는데, 이동한 행에 데이터가 있으면 true를, 없으면 false를 리턴한다. 앞의 그림을 보면 last 행까지는 true를 리턴하고 afterLast 행으로 이동하면 false를 리턴하는 것을 볼 수 있다.

만약 SELECT 문으로 가져온 데이터 행이 없다면 beforeFirst 행과 afterLast 행이 붙어 있기 때문에 첫 번째 next() 결과는 false가 된다. 다음은 SELECT 문으로 가져온 행의 수에 따라서 커서를 이동시키는 코드이다.

1개의 데이터 행만 가져올 경우

```
ResultSet rs = pstmt.executeQuery();
if(rs.next()) {
    //첫 번째 데이터 행 처리
} else {
    //afterLast 행으로 이동했을 경우
}
```

n개의 데이터 행을 가져올 경우

```
ResultSet rs = pstmt.executeQuery();
while(rs.next()) {
    //last 행까지 이동하면서 데이터 행 처리
}
//afterLast 행으로 이동했을 경우
```

1개의 데이터 행만 가져올 경우에는 if 조건식에서 next() 메소드를 1번 호출한다. true일 경우(첫 번째 데이터 행이 있을 경우)와 false일 경우(afterLast 행으로 이동했을 경우)에 따라서 적절한 처리를 해야 한다. 주로 SELECT 문이 기본키(primary key)를 조건으로 데이터를 가져오는 경우에 해당한다. n개의 데이터 행을 가져올 경우에는 while 문을 이용해서 next() 메소드를 반복 호출해 true가 리턴될 동안(last 행까지 이동할 때까지) 데이터 행을 처리하고, false가 리턴되면(afterLast 행으로 이동할 때) 반복을 종료시킨다.

SELECT 문에 따라 ResultSet에는 많은 데이터 행이 저장될 수 있기 때문에 ResultSet을 더 이상 사용하지 않는다면 close() 메소드를 호출해서 ResultSet이 사용한 메모리를 해제하는 것이 좋다.

```
rs.close();
```

데이터 행 읽기

커서가 있는 데이터 행에서 각 컬럼의 값은 Getter 메소드로 읽을 수 있다. 컬럼의 데이터 타입에 따라서 getXxx() 메소드가 사용되며, 매개값으로 컬럼의 이름 또는 컬럼 순번을 줄 수 있다.

ResultSet에서 컬럼 순번은 1부터 시작하기 때문에 userid = 1, username = 2, userage = 3이 된다.

컬럼 이름으로 읽기

```
String userId =  
    rs.getString("userid");  
String userName =  
    rs.getString("username");  
int userAge = rs.getInt("userage");
```

컬럼 순번으로 읽기

```
String userId = rs.getString(1);  
String userName = rs.getString(2);  
int userAge = rs.getInt(3);
```

만약 SELECT 문에 연산식이나 함수 호출이 포함되어 있다면 컬럼 이름 대신에 컬럼 순번으로 읽어야 한다. 예를 들어 다음과 같은 SELECT 문에서 userage - 1 연산식이 사용되면 컬럼 순번으로만 읽을 수 있다. userage - 1은 컬럼명이 아니기 때문이다.

```
SELECT userid, userage - 1
FROM users
```

```
String userId =
    rs.getString("userid");
int userAge = rs.getInt(2);
```

NOTE ▶ (userage - 1) as userage와 같이 별명(alias)이 있다면 별명이 컬럼 이름이 된다.

사용자 정보 읽기

users 테이블에서 userid가 winter인 사용자의 정보를 가져와 출력해 보자. 먼저 users 테이블의 한 개의 행(사용자)을 저장할 User 클래스를 작성한다. 컬럼 개수와 타입에 맞게 필드를 선언하고, 롬복 @Data 어노테이션을 이용해서 Getter, Setter, toString() 메소드를 자동 생성시킨다.

»» User.java

```
1 package ch20.mysql.sec09.exam01;
2
3 import lombok.Data;
4
5 @Data //Constructor, Getter, Setter, hashCode(), equals(), toString() 자동 생성
6 public class User {
7     private String userId;
8     private String userName;
9     private String userPassword;
10    private int userAge;
11    private String userEmail;
12 }
```

userid가 winter인 사용자 정보를 가져오는 SELECT 문은 다음과 같다.

```
SELECT userid, username, userpassword, userage, useremail
FROM users
WHERE userid='winter';
```

조건절의 값을 ?로 대체한 매개변수화된 SQL 문을 String 타입 변수 sql에 대입한다.

```
String sql = "" +
    "SELECT userid, username, userpassword, userage, useremail " +
    "FROM users " +
    "WHERE userid=?";
```

매개변수화된 SELECT 문을 실행하기 위해 `prepareStatement()` 메소드로부터 `PreparedStatement`를 얻고, ?에 값을 지정한다.

```
PreparedStatement pstmt = conn.prepareStatement(sql);
pstmt.setString(1, "winter");
```

`executeQuery()` 메소드로 SELECT 문을 실행해서 `ResultSet`을 얻는다. `userid`는 기본키(primary key)이므로 조건에 맞는 행은 1개이거나, 0개이므로 if 문을 이용해서 `next()` 메소드가 true를 리턴할 경우 데이터 행을 `User` 객체에 저장하고 출력한다.

```
ResultSet rs = pstmt.executeQuery();
if(rs.next()) { //1개의 데이터 행을 가져왔을 경우
    User user = new User();
    user.setUserId(rs.getString("userid"));
    user.setUserName(rs.getString("username"));
    user.setUserPassword(rs.getString("userpassword"));
    user.setUserAge(rs.getInt(4)); //컬럼 순번을 이용해서 컬럼 지정
    user.setUserEmail(rs.getString(5)); //컬럼 순번을 이용해서 컬럼 지정
    System.out.println(user);
} else { //데이터 행을 가져오지 않았을 경우
    System.out.println("사용자 아이디가 존재하지 않음");
}
```

`System.out.println(user)`는 롬복이 생성한 `User`의 `toString()` 메소드를 호출해서 받은 리턴값을 출력한다. 다음은 `users` 테이블에서 `userid`가 `winter`인 사용자 정보를 가져오는 전체 코드를 보여 준다.

>>> UserSelectExample.java

```
1  package ch20.mysql.sec09.exam01;
2
3  import java.sql.Connection;
4  import java.sql.DriverManager;
5  import java.sql.PreparedStatement;
6  import java.sql.ResultSet;
7  import java.sql.SQLException;
8
9  public class UserSelectExample {
10     public static void main(String[] args) {
11         Connection conn = null;
12         try {
13             //JDBC Driver 등록
14             Class.forName("com.mysql.cj.jdbc.Driver");
15
16             //연결하기
17             conn = DriverManager.getConnection(
18                 "jdbc:mysql://localhost:3306/thisisjava",
19                 "java",
20                 "mysql"
21             );
22
23             //매개변수화된 SQL 문 작성
24             String sql = "" +
25                 "SELECT userid, username, userpassword, userage, useremail " +
26                 "FROM users " +
27                 "WHERE userid=?";
28
29             //PreparedStatement 얻기 및 값 지정
30             PreparedStatement pstmt = conn.prepareStatement(sql);
31             pstmt.setString(1, "winter");
32
33             //SQL 문 실행 후, ResultSet을 통해 데이터 읽기
34             ResultSet rs = pstmt.executeQuery();
35             if(rs.next()) { //1개의 데이터 행을 가져왔을 경우
36                 User user = new User();
37                 user.setUserId(rs.getString("userid"));
38                 user.setUserName(rs.getString("username"));
```

```

39         user.setUserPassword(rs.getString("userpassword"));
40         user.setUserAge(rs.getInt(4));           //컬럼 순번을 이용
41         user.setUserEmail(rs.getString(5));     //컬럼 순번을 이용
42         System.out.println(user);
43     } else { //데이터 행을 가져오지 않았을 경우
44         System.out.println("사용자 아이디가 존재하지 않음");
45     }
46     rs.close();
47
48     //PreparedStatement 닫기
49     pstmt.close();
50 } catch (Exception e) {
51     e.printStackTrace();
52 } finally {
53     if(conn != null) {
54         try {
55             //연결 끊기
56             conn.close();
57         } catch (SQLException e) {}
58     }
59 }
60 }
61 }

```

실행 결과

```
User(userId=winter, userName=한겨울, userPassword=12345, userAge=25,
userEmail=winter@mycompany.com)
```

게시물 정보 읽기

이번에는 boards 테이블에서 bwriter가 winter인 게시물의 정보를 가져와보자. 먼저 20.6의 BoardInsertExample 예제를 이용해서 다음과 같이 boards 테이블에 bwriter를 winter로 하는 게시물을 2개 이상 저장해둔다.

Result Grid							
		Filter Rows:		Edit:		Export/Import:	
	bno	btitle	bcontent	bwriter	bdate	bfilename	bfiledata
▶	6	봄의 정원	정원의 꽃이 이쁘네요	winter	2022-04-02 08:17:38	spring.jpg	BLOB
	7	눈오는 날	활짝눈이 내려요	winter	2022-04-02 08:18:36	snow.jpg	BLOB
	8	크리스마스	메리 크리스마스~	winter	2022-04-02 08:19:33	chrismas.jpg	BLOB
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

NOTE ▶ bno의 값은 다를 수 있다.

먼저 boards 테이블의 1개 행(게시물)을 저장할 Board 클래스를 작성한다. 컬럼 개수와 타입에 맞게 필드를 선언하고, 롬복 @Data 어노테이션을 이용해서 Getter, Setter, toString() 메소드를 자동 생성한다.

>>> Board.java

```

1  package ch20.mysql.sec09.exam02;
2
3  import java.sql.Blob;
4  import java.util.Date;
5  import lombok.Data;
6
7  @Data //Constructor, Getter, Setter, hashCode(), equals(), toString() 자동 생성
8  public class Board {
9      private int bno;
10     private String btitle;
11     private String bcontent;
12     private String bwriter;
13     private Date bdate;
14     private String bfilename;
15     private Blob bfiledata;
16 }

```

bwriter가 winter인 게시물 정보를 가져오는 SELECT 문은 다음과 같다.

```

SELECT bno, btitle, bcontent, bwriter, bdate, bfilename, bfiledata
FROM boards
WHERE bwriter='winter';

```

조건절의 값을 ?로 대체한 매개변수화된 SELECT 문을 String 타입 변수 sql에 대입한다.

```
String sql = "" +
    "SELECT bno, btitle, bcontent, bwriter, bdate, bfilename, bfiledata " +
    "FROM boards " +
    "WHERE bwriter=?";
```

매개변수화된 SELECT 문을 실행하기 위해 다음과 같이 `prepareStatement()` 메소드로부터 `PreparedStatement`를 얻고, ?에 값을 지정한다.

```
PreparedStatement pstmt = conn.prepareStatement(sql);
pstmt.setString(1, "winter");
```

`executeQuery()` 메소드로 SELECT 문을 실행해서 `ResultSet`을 얻는다. 조건에 맞는 행은 n개이므로 while 문을 이용해서 `next()` 메소드가 false를 리턴할 때까지 반복해서 데이터 행을 Board 객체에 저장하고 출력한다.

```
ResultSet rs = pstmt.executeQuery();
while(rs.next()) {
    //데이터 행을 읽고 Board 객체에 저장
    Board board = new Board();
    board.setBno(rs.getInt("bno"));
    board.setBtitle(rs.getString("btitle"));
    board.setBcontent(rs.getString("bcontent"));
    board.setBwriter(rs.getString("bwriter"));
    board.setBdate(rs.getDate("bdate"));
    board.setBfilename(rs.getString("bfilename"));
    board.setBfiledata(rs.getBlob("bfiledata"));

    //콘솔에 출력
    System.out.println(board);
}
```

`System.out.println(board)`는 롬복이 생성한 Board의 `toString()` 메소드를 호출해서 받은 리턴값을 출력한다. Board의 `bfiledata`는 Blob 객체이므로 콘솔에 출력하면 `com.mysql.cj.jdbc.Blob@65b104b9`와 같이 의미 없는 타입 정보만 출력된다.

Blob 객체에 저장된 바이너리 데이터를 얻기 위해서는 다음과 같이 입력 스트림 또는 배열을 얻어 내야 한다.

```
Blob blob = board.getBfiledata();
InputStream is =
    blob.getBinaryStream();
```

```
Blob blob = board.getBfiledata();
byte[] bytes = blob.getBytes(0,
    blob.length());
```

다음은 Blob 객체에서 InputStream을 얻고, 읽은 바이트를 파일로 저장하는 방법을 보여 준다.

```
InputStream is = blob.getBinaryStream();
OutputStream os = new FileOutputStream("C:/Temp/" + board.getBfilename());
is.transferTo(os);
os.flush();
os.close();
is.close();
```

다음은 boards 테이블에서 bwriter가 winter인 게시물 정보를 가져오는 전체 코드이다.

>>> BoardSelectExample.java

```
1  package ch20.mysql.sec09.exam02;
2
3  import java.io.FileOutputStream;
4  import java.io.InputStream;
5  import java.io.OutputStream;
6  import java.sql.Blob;
7  import java.sql.Connection;
8  import java.sql.DriverManager;
9  import java.sql.PreparedStatement;
10 import java.sql.ResultSet;
11 import java.sql.SQLException;
12
13 public class BoardSelectExample {
14     public static void main(String[] args) {
15         Connection conn = null;
16         try {
```

```

17      //JDBC Driver 등록
18      Class.forName("com.mysql.cj.jdbc.Driver");
19
20      //연결하기
21      conn = DriverManager.getConnection(
22          "jdbc:mysql://localhost:3306/thisisjava",
23          "java",
24          "mysql"
25      );
26
27      //매개변수화된 SQL 문 작성
28      String sql = "" +
29          "SELECT bno, btitle, bcontent, bwriter, bdate, bfilename, bfiledata " +
30          "FROM boards " +
31          "WHERE bwriter=?";
32
33      //PreparedStatement 얻기 및 값 지정
34      PreparedStatement pstmt = conn.prepareStatement(sql);
35      pstmt.setString(1, "winter");
36
37      //SQL 문 실행 후, ResultSet을 통해 데이터 읽기
38      ResultSet rs = pstmt.executeQuery();
39      while(rs.next()) {
40          //데이터 행을 읽고 Board 객체 생성
41          Board board = new Board();
42          board.setBno(rs.getInt("bno"));
43          board.setBtitle(rs.getString("btitle"));
44          board.setBcontent(rs.getString("bcontent"));
45          board.setBwriter(rs.getString("bwriter"));
46          board.setBdate(rs.getDate("bdate"));
47          board.setBfilename(rs.getString("bfilename"));
48          board.setBfiledata(rs.getBlob("bfiledata"));
49
50          //콘솔에 출력
51          System.out.println(board);
52
53          //파일로 저장
54          Blob blob = board.getBfiledata();
55          if(blob != null) {
56              InputStream is = blob.getBinaryStream();

```

```

57         OutputStream os = new FileOutputStream("C:/Temp/" +
            board.getBfilename());
58         is.transferTo(os);
59         os.flush();
60         os.close();
61         is.close();
62     }
63 }
64 rs.close();
65
66 //PreparedStatement 닫기
67 pstmt.close();
68 } catch (Exception e) {
69     e.printStackTrace();
70 } finally {
71     if(conn != null) {
72         try {
73             //연결 끊기
74             conn.close();
75         } catch (SQLException e) {}
76     }
77 }
78 }
79 }

```

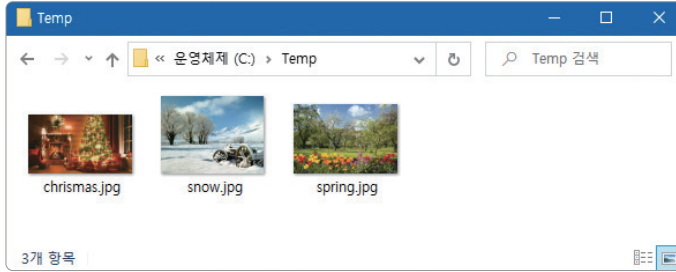
실행 결과

Board(bno=14, btitle=봄의 정원, bcontent=정원의 꽃이 예쁘네요., bwriter=winter, bdate=2022-01-25, bfilename=spring.jpg, bfiledata= com.mysql.cj.jdbc.Blob@5f354bcf)

Board(bno=12, btitle=눈 오는 날, bcontent=함박눈이 내려요., bwriter=winter, bdate=2022-01-25, bfilename=snow.jpg, bfiledata= com.mysql.cj.jdbc.Blob@146dfe6)

Board(bno=13, btitle=크리스마스, bcontent=메리 크리스마스~, bwriter=winter, bdate=2022-01-25, bfilename=christmas.jpg, bfiledata= com.mysql.cj.jdbc.Blob@4716be8b)

bfiledata 컬럼의 그림 데이터는 다음과 같이 bfilename 컬럼 값을 파일명으로 해서 C:\Temp 디렉토리에 저장된다.



20.10 트랜잭션 처리

트랜잭션(transaction)은 기능 처리의 최소 단위를 말한다. 하나의 기능은 여러 가지 소작업들로 구성될 수 있다. 최소 단위란 것은 이 소작업들을 분리할 수 없으며, 전체를 하나로 본다는 개념이다. 트랜잭션은 소작업들이 모두 성공하거나 모두 실패해야 한다.

예를 들어 계좌 이체는 출금 작업과 입금 작업으로 구성된 트랜잭션이다. 출금과 입금 작업 중 하나만 성공할 수 없으며, 모두 성공하거나 모두 실패되어야 한다.

계좌 이체(트랜잭션)

출금 작업

입금 작업

계좌 이체는 DB 입장에서 보면 두 개의 계좌 금액을 수정하는 작업이다. 출금 계좌에서 금액을 감소시키고, 입금 계좌에서 금액을 증가시킨다. 따라서 다음과 같이 두 개의 UPDATE 문이 필요하다. 두 UPDATE 문은 모두 성공하거나 모두 실패해야 하며, 하나만 성공할 수 없다.

계좌 이체(트랜잭션)

//출금 작업

```
UPDATE accounts SET balance=balance-이체금액 WHERE ano=출금계좌번호
```

//입금 작업

```
UPDATE accounts SET balance=balance+이체금액 WHERE ano=입금계좌번호
```

DB는 트랜잭션을 처리하기 위해 커밋(commit)과 롤백(rollback)을 제공한다. 커밋은 내부 작업을 모두 성공 처리하고, 롤백은 실행 전으로 돌아간다는 의미에서 모두 실패 처리한다.

JDBC에서는 INSERT, UPDATE, DELETE 문을 실행할 때마다 자동 커밋이 일어난다. 이 기능은 계좌 이체와 같이 두 가지 UPDATE 문을 실행할 때 문제가 된다. 출금 작업이 성공되면 바로 커밋이 되기 때문에 입금 작업의 성공 여부와 상관없이 출금 작업만 별도 처리된다.

따라서 JDBC에서 트랜잭션을 코드로 제어하려면 자동 커밋 기능을 꺼야 한다. 자동 커밋 설정 여부는 Connection의 setAutoCommit() 메소드로 할 수 있다. 다음 코드는 자동 커밋 기능을 끈다.

```
conn.setAutoCommit(false);
```

자동 커밋 기능이 꺼지면, 다음과 같은 코드로 커밋과 롤백을 제어할 수 있다.

```
conn.commit();    //커밋하기
conn.rollback();   //롤백하기
```

트랜잭션을 위한 일반적인 코드 작성 패턴은 다음과 같다.

```
Connection conn = null;
try {
    //트랜잭션 시작 -----
    //자동 커밋 기능 끄기
    conn.setAutoCommit(false);
    //소작업 처리
    ...
    //소작업 처리
    ...
    //커밋 -> 모두 성공 처리
    conn.commit();
    //트랜잭션 종료 -----
} catch (Exception e) {
    try {
        //롤백 -> 모두 실패 처리
        conn.rollback();
    } catch (SQLException e1) {}
}
```

```

    } finally {
        if(conn != null) {
            try {
                //원래대로 자동 커밋 기능 켜기
                conn.setAutoCommit(true);
                //연결 끊기
                conn.close();
            } catch (SQLException e) {}
        }
    }
}

```

다음 예제는 accounts 테이블에서 111-111-1111 계좌에서 222-222-2222 계좌로 10000원을 이체하기 위해 트랜잭션 처리를 한다.

>>> TransactionExample.java

```

1  package ch20.mysql.sec10;
2
3  import java.sql.Connection;
4  import java.sql.DriverManager;
5  import java.sql.PreparedStatement;
6  import java.sql.SQLException;
7
8  public class TransactionExample {
9      public static void main(String[] args) {
10         Connection conn = null;
11         try {
12             //JDBC Driver 등록
13             Class.forName("com.mysql.cj.jdbc.Driver");
14
15             //연결하기
16             conn = DriverManager.getConnection(
17                 "jdbc:mysql://localhost:3306/thisisjava",
18                 "java",
19                 "mysql"
20             );
21
22             //트랜잭션 시작 -----

```



```

23     //자동 커밋 기능 끄기
24     conn.setAutoCommit(false);
25
26     //출금 작업
27     String sql1 = "UPDATE accounts SET balance=balance-? WHERE ano=?";
28     PreparedStatement pstmt1 = conn.prepareStatement(sql1);
29     pstmt1.setInt(1, 10000);
30     pstmt1.setString(2, "111-111-1111");
31     int rows1 = pstmt1.executeUpdate();
32     if(rows1 == 0) throw new Exception("출금되지 않았음");
33     pstmt1.close();
34
35     //입금 작업
36     String sql2 = "UPDATE accounts SET balance=balance+? WHERE ano=?";
37     PreparedStatement pstmt2 = conn.prepareStatement(sql2);
38     pstmt2.setInt(1, 10000);
39     pstmt2.setString(2, "222-222-2222");
40     int rows2 = pstmt2.executeUpdate();
41     if(rows2 == 0) throw new Exception("입금되지 않았음");
42     pstmt2.close();
43
44     //수동 커밋 -> 모두 성공 처리
45     conn.commit();
46     System.out.println("계좌 이체 성공");
47     //트랜잭션 종료 -----
48 } catch (Exception e) {
49     try {
50         //수동 롤백 -> 모두 실패 처리
51         conn.rollback();
52     } catch (SQLException e1) {}
53     System.out.println("계좌 이체 실패");
54     e.printStackTrace();
55 } finally {
56     if(conn != null) {
57         try {
58             //원래대로 자동 커밋 기능 켜기
59             conn.setAutoCommit(true);
60             //연결 끊기
61             conn.close();
62         } catch (SQLException e) {}

```

```

63         }
64     }
65 }
66 }

```

실행 결과

계좌 이체 성공

39라인에서 입금 계좌를 '333-333-3333'처럼 다르게 주면 rows2가 0이 되므로 41라인에서 예외가 발생하고, 예외 처리 코드 51라인에서 롤백된다. 롤백이 되면 출금도 실패 처리되므로 출금 계좌와 입금 계좌의 금액은 변동되지 않는다.

>>> TransactionExample.java

```

...     ...
39         pstmt2.setString(2, "333-333-3333");
...     ...

```

실행 결과

계좌 이체 실패

```

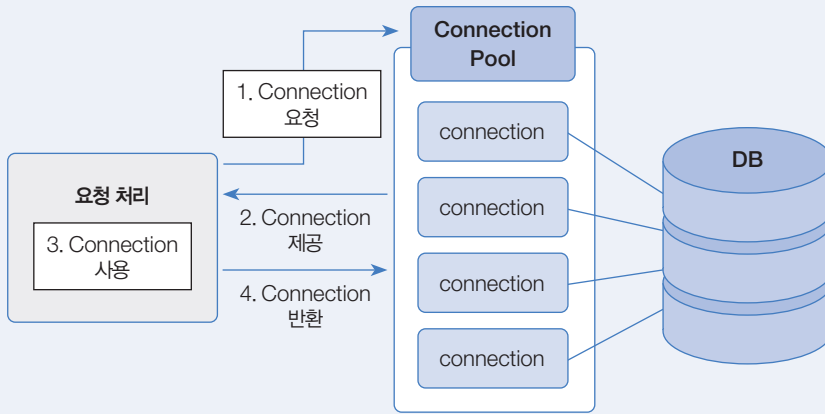
java.lang.Exception: 입금되지 않았음
    at ch20.mysql.sec10.TransactionExample.main(TransactionExample.java:41)

```

트랜잭션을 처리한 이후에는 원래대로 자동 커밋 기능을 켜줘야 한다. 앞의 예제는 더 이상 Connection을 사용하지 않기 때문에 상관은 없지만, Connection을 다른 기능 처리를 위해 계속 사용해야 한다면 setAutoCommit(true) 코드로 자동 커밋 기능을 켜줘야 한다. 특히 커넥션 풀 Connection Pool을 사용할 때 주의해야 할 부분이다.

☆ 커넥션 풀

다수의 클라이언트의 요청을 처리하는 서버 프로그램은 대부분 커넥션 풀(Connection Pool)을 사용한다. 커넥션 풀은 일정량의 Connection을 미리 생성시켜놓고, 서버에서 클라이언트의 요청을 처리할 때 Connection을 제공해주고 다시 반환받는 역할을 수행한다.



커넥션 풀을 사용하면 생성된 Connection을 재사용할 수 있기 때문에 DB 연결 시간을 줄일 수 있고, 전체 Connection 수를 관리할 수도 있다. 따라서 이것은 불특정 다수의 클라이언트 요청을 처리하는 서버 프로그램에서는 필수 기능 중 하나이다.

20.11 게시판 구현

지금까지 학습한 JDBC를 활용해서 명령 프롬프트(윈도우) 또는 터미널(맥OS)에서 실행되는 게시판을 구현해 보자. 게시판은 기본적인 CRUD(Create, Read, Update, Delete) 기능이 포함되어 있어 가장 좋은 실습 주제 중 하나이다.

메인 메뉴

BoardExample 클래스를 실행하면 다음과 같은 게시물 목록과 함께 메뉴가 나오도록 작성해 보자.

[게시물 목록]

no	writer	date	title
1	winter	2022-01-27	게시판에 오신 것을 환영합니다.
2	winter	2022-01-27	올 겨울은 많이 춥습니다.

메인 메뉴: 1.Create | 2.Read | 3.Clear | 4.Exit

메뉴 선택:

먼저 `list()`, `mainMenu()`, `main()` 메소드를 다음과 같이 작성한다. `main()` 메소드는 `BoardExample` 객체를 생성하고 `list()` 메소드를 호출한다. `list()` 메소드는 게시물 목록을 출력하고 `mainMenu()` 메소드를 호출한다.

>>> BoardExample1.java

```

1  package ch20.mysql.sec11;
2
3  public class BoardExample1 {
4      //Field
5
6      //Constructor
7
8      //Method
9      public void list() {
10         System.out.println();
11         System.out.println("[게시물 목록]");
12         System.out.println("-----");
13         System.out.printf("%-6s%-12s%-16s%-40s\n", "no", "writer", "date", "title");
14         System.out.println("-----");
15         System.out.printf("%-6s%-12s%-16s%-40s \n",
16             "1", "winter", "2022.01.27", "게시판에 오신 것을 환영합니다.");
17         System.out.printf("%-6s%-12s%-16s%-40s \n",
18             "2", "winter", "2022.01.27", "올 겨울은 많이 춥습니다.");
19         mainMenu();
20     }

```

```

21
22     public void mainMenu() {
23         System.out.println();
24         System.out.println("-----");
25         System.out.println("메인 메뉴: 1.Create | 2.Read | 3.Clear | 4.Exit");
26         System.out.print("메뉴 선택: ");
27         System.out.println();
28     }
29
30     public static void main(String[] args) {
31         BoardExample1 boardExample = new BoardExample1();
32         boardExample.list();
33     }
34 }

```

메인 메뉴 선택 기능

메인 메뉴 1부터 4 중 하나를 선택하면 다음과 같이 해당 메소드가 실행되도록 작성해 보자.

실행 결과

[게시물 목록]

no	writer	date	title
1	winter	2022.01.27	게시판에 오신 것을 환영합니다.
2	winter	2022.01.27	올 겨울은 많이 춥습니다.

메인 메뉴: 1.Create | 2.Read | 3.Clear | 4.Exit

메뉴 선택: 1

*** create() 메소드 실행됨

[게시물 목록]

no	writer	date	title
1	winter	2022.01.27	게시판에 오신 것을 환영합니다.

2 winter 2022.01.27 올 겨울은 많이 춥습니다.

메인 메뉴: 1.Create | 2.Read | 3.Clear | 4.Exit

메뉴 선택: 2

*** read() 메소드 실행됨

[게시물 목록]

no	writer	date	title
1	winter	2022.01.27	게시판에 오신 것을 환영합니다.
2	winter	2022.01.27	올 겨울은 많이 춥습니다.

메인 메뉴: 1.Create | 2.Read | 3.Clear | 4.Exit

메뉴 선택: 3

*** clear() 메소드 실행됨

[게시물 목록]

no	writer	date	title
1	winter	2022.01.27	게시판에 오신 것을 환영합니다.
2	winter	2022.01.27	올 겨울은 많이 춥습니다.

메인 메뉴: 1.Create | 2.Read | 3.Clear | 4.Exit

메뉴 선택: 4

키보드 입력을 받기 위해 Scanner 필드를 추가하고(7라인), mainMenu() 메소드에서 키보드 입력을 받기 위해 nextLine() 메소드를 호출한다(30라인). 그리고 메뉴 선택 번호에 따라 해당 메소드를 호출한다(33~38라인).

>>> BoardExample2.java

```
1  package ch20.mysql.sec11;
2
3  import java.util.Scanner;
4
5  public class BoardExample2 {
6      //Field
7      private Scanner scanner = new Scanner(System.in);
8
9      //Constructor
10
11     //Method
12     public void list() {
13         System.out.println();
14         System.out.println("[게시물 목록]");
15         System.out.println("-----");
16         System.out.printf("%-6s%-12s%-16s%-40s\n", "no", "writer", "date", "title");
17         System.out.println("-----");
18         System.out.printf("%-6s%-12s%-16s%-40s\n",
19             "1", "winter", "2022.01.27", "게시판에 오신 것을 환영합니다.");
20         System.out.printf("%-6s%-12s%-16s%-40s\n",
21             "2", "winter", "2022.01.27", "올 겨울은 많이 춥습니다.");
22         mainMenu();
23     }
24
25     public void mainMenu() {
26         System.out.println();
27         System.out.println("-----");
28         System.out.println("메인 메뉴: 1.Create | 2.Read | 3.Clear | 4.Exit");
29         System.out.print("메뉴 선택: ");
30         String menuNo = scanner.nextLine();
31         System.out.println();
32
33         switch(menuNo) {
34             case "1" -> create();
35             case "2" -> read();
36             case "3" -> clear();
37             case "4" -> exit();
38         }
```

```

39     }
40
41     public void create() {
42         System.out.println("*** create() 메소드 실행됨");
43         list();
44     }
45
46     public void read() {
47         System.out.println("*** read() 메소드 실행됨");
48         list();
49     }
50
51     public void clear() {
52         System.out.println("*** clear() 메소드 실행됨");
53         list();
54     }
55
56     public void exit() {
57         System.exit(0);
58     }
59
60     public static void main(String[] args) {
61         BoardExample2 boardExample = new BoardExample2();
62         boardExample.list();
63     }
64 }

```

Board 클래스 작성

boards 테이블의 한 개의 행(게시물)을 저장할 Board 클래스를 작성한다. 컬럼 개수와 타입에 맞게 필드를 선언하고, 롬복 @Data 어노테이션을 이용해서 Getter, Setter, toString() 메소드를 자동 생성시킨다.

>>> Board.java

```
1 package ch20.mysql.sec11;
2
3 import java.util.Date;
4 import lombok.Data;
5
6 @Data
7 public class Board {
8     private int bno;
9     private String btitle;
10    private String bcontent;
11    private String bwriter;
12    private Date bdate;
13 }
```

boards 테이블에는 bfilename과 bfiledata 컬럼이 있지만, 이번 게시판 구현에서는 첨부 파일은 제외할 것이므로 필드로 선언하지 않는다.

게시물 목록 기능

boards 테이블에서 모든 게시물 정보들을 가져온 다음 게시물 목록으로 출력시켜 보자.

실행 결과

[게시물 목록]

no	writer	date	title
3	winter	2022-01-27	봄의 정원
2	winter	2022-01-27	크리스마스
1	winter	2022-01-27	눈 오는 날

메인 메뉴: 1.Create | 2.Read | 3.Clear | 4.Exit
메뉴 선택:

DB 연결이 필요하므로 Connection 필드를 추가하고(15라인), 생성자에서 DB 연결을 한다(18~33라인). 그리고 boards 테이블에서 게시물 정보들을 가져와서 게시물 목록으로 출력하도록 list() 메소드를 수정한다(36~74라인).

» BoardExample3.java

```
1  package ch20.mysql.sec11;
2
3  import java.sql.Connection;
4  import java.sql.DriverManager;
5  import java.sql.PreparedStatement;
6  import java.sql.ResultSet;
7  import java.sql.SQLException;
8  import java.util.Scanner;
9
10 import ch20.mysql.sec11.Board;
11
12 public class BoardExample3 {
13     //Field
14     private Scanner scanner = new Scanner(System.in);
15     private Connection conn;
16
17     //Constructor
18     public BoardExample3() {
19         try {
20             //JDBC Driver 등록
21             Class.forName("com.mysql.cj.jdbc.Driver");
22
23             //연결하기
24             conn = DriverManager.getConnection(
25                 "jdbc:mysql://localhost:3306/thisisjava",
26                 "java",
27                 "mysql"
28             );
29         } catch (Exception e) {
30             e.printStackTrace();
31             exit();
32         }
33     }
```

```

34
35 //Method
36 public void list() {
37     //타이틀 및 컬럼명 출력
38     System.out.println();
39     System.out.println("[게시물 목록]");
40     System.out.println("-----");
41     System.out.printf("%-6s%-12s%-16s%-40s\n", "no", "writer", "date", "title");
42     System.out.println("-----");
43
44     //boards 테이블에서 게시물 정보를 가져와서 출력하기
45     try {
46         String sql = "" +
47             "SELECT bno, btitle, bcontent, bwriter, bdate " +
48             "FROM boards " +
49             "ORDER BY bno DESC";
50         PreparedStatement pstmt = conn.prepareStatement(sql);
51         ResultSet rs = pstmt.executeQuery();
52         while(rs.next()) {
53             Board board = new Board();
54             board.setBno(rs.getInt("bno"));
55             board.setBtitle(rs.getString("btitle"));
56             board.setBcontent(rs.getString("bcontent"));
57             board.setBwriter(rs.getString("bwriter"));
58             board.setBdate(rs.getDate("bdate"));
59             System.out.printf("%-6s%-12s%-16s%-40s \n",
60                 board.getBno(),
61                 board.getBwriter(),
62                 board.getBdate(),
63                 board.getBtitle());
64         }
65         rs.close();
66         pstmt.close();
67     } catch(SQLException e) {
68         e.printStackTrace();
69         exit();
70     }
71
72     //메인 메뉴 출력
73     mainMenu();

```

```

74     }
75
76     //이하 동일
...     ...

```

게시물 생성 기능

메인 메뉴에서 '1.Create'를 선택하면 새로운 게시물의 제목, 내용, 작성자를 키보드로 입력받고, 보조 메뉴에서 '1.Ok'를 선택하면 boards 테이블에 새로운 게시물이 저장되도록 해보자.

실행 결과

메인 메뉴: 1.Create | 2.Read | 3.Clear | 4.Exit
메뉴 선택: 1

[새 게시물 입력]

제목: 여름에 가장 시원할 때

내용: 에어컨이 나오는 강의실에서 자바 공부할 때입니다^^.

작성자: summer

보조 메뉴: 1.Ok | 2.Cancel

메뉴 선택: 1

[게시물 목록]

no	writer	date	title
4	summer	2022-01-27	여름에 가장 시원할 때
3	winter	2022-01-27	봄의 정원
2	winter	2022-01-27	크리스마스
1	winter	2022-01-27	눈 오는 날

메인 메뉴: 1.Create | 2.Read | 3.Clear | 4.Exit

메뉴 선택:

메인 메뉴에서 '1.Create'를 선택했을 때 호출되는 create() 메소드를 다음과 같이 수정한다.

>>> BoardExample4.java

```
1    //이상 동일
...    ...
91
92    public void create() {
93        //입력 받기
94        Board board = new Board();
95        System.out.println("[새 게시물 입력]");
96        System.out.print("제목: ");
97        board.setBtitle(scanner.nextLine());
98        System.out.print("내용: ");
99        board.setBcontent(scanner.nextLine());
100       System.out.print("작성자: ");
101       board.setBwriter(scanner.nextLine());
102
103       //보조 메뉴 출력
104       System.out.println("-----");
105       System.out.println("보조 메뉴: 1.Ok | 2.Cancel");
106       System.out.print("메뉴 선택: ");
107       String menuNo = scanner.nextLine();
108       if(menuNo.equals("1")) {
109           //boards 테이블에 게시물 정보 저장
110           try {
111               String sql = "" +
112                   "INSERT INTO boards (btitle, bcontent, bwriter, bdate) " +
113                   "VALUES (?, ?, ?, now());";
114               PreparedStatement pstmt = conn.prepareStatement(sql);
115               pstmt.setString(1, board.getBtitle());
116               pstmt.setString(2, board.getBcontent());
117               pstmt.setString(3, board.getBwriter());
118               pstmt.executeUpdate();
119               pstmt.close();
120           } catch (Exception e) {
121               e.printStackTrace();
122               exit();
123           }
124       }
```

```

125
126         //게시물 목록 출력
127         list();
128     }
129
130     //이하 동일
...     ...

```

게시물 읽기 기능

메인 메뉴에서 '2.Read'를 선택했을 때 게시물의 번호를 키보드로 입력받고, boards 테이블에 있는 해당 게시물을 가져와서 출력해 보자.

실행 결과

[게시물 목록]

no	writer	date	title
4	summer	2022-01-27	여름에 가장 시원할 때
3	winter	2022-01-27	봄의 정원
2	winter	2022-01-27	크리스마스
1	winter	2022-01-27	눈 오는 날

메인 메뉴: 1.Create | 2.Read | 3.Clear | 4.Exit

메뉴 선택: 2

[게시물 읽기]

bno: 4

#####

번호: 4

제목: 여름에 가장 시원할 때

내용: 에이컨이 나오는 강의실에서 자바 공부할 때입니다^^.

작성자: summer

날짜: 2022-01-27

#####

메인 메뉴에서 '2.Read'를 선택했을 때 호출되는 read() 메소드를 다음과 같이 수정한다.

>>> BoardExample5.java

```
1      //이상 동일
...      ...

129
130      public void read() {
131          //입력받기
132          System.out.println("[게시물 읽기]");
133          System.out.print("bno: ");
134          int bno = Integer.parseInt(scanner.nextLine());
135
136          //boards 테이블에서 해당 게시물을 가져와 출력
137          try {
138              String sql = "" +
139                  "SELECT bno, btitle, bcontent, bwriter, bdate " +
140                  "FROM boards " +
141                  "WHERE bno=?";
142              PreparedStatement pstmt = conn.prepareStatement(sql);
143              pstmt.setInt(1, bno);
144              ResultSet rs = pstmt.executeQuery();
145              if(rs.next()) {
146                  Board board = new Board();
147                  board.setBno(rs.getInt("bno"));
148                  board.setBtitle(rs.getString("btitle"));
149                  board.setBcontent(rs.getString("bcontent"));
150                  board.setBwriter(rs.getString("bwriter"));
151                  board.setBdate(rs.getDate("bdate"));
152                  System.out.println("#####");
153                  System.out.println("번호: " + board.getBno());
154                  System.out.println("제목: " + board.getBtitle());
155                  System.out.println("내용: " + board.getBcontent());
156                  System.out.println("작성자: " + board.getBwriter());
157                  System.out.println("날짜: " + board.getBdate());
158                  System.out.println("#####");
159              }
160              rs.close();
161              pstmt.close();
162          } catch (Exception e) {
```

```

163         e.printStackTrace();
164         exit();
165     }
166
167     //게시물 목록 출력
168     list();
169 }
170
171 //이하 동일
...

```

게시물 수정 기능

게시물 읽기에서 보조 메뉴를 추가하고, 보조 메뉴에서 '1.Update'를 선택했을 때 제목, 내용, 작성자의 수정 내용을 입력할 수 있도록 해보자. 그리고 보조 메뉴에서 '1.Ok'를 선택했을 때 boards 테이블의 해당 게시물을 수정하도록 해보자.

실행 결과

[게시물 읽기]

bno: 3

#####

번호: 3

제목: 봄의 정원

내용: 정원의 꽃이 예쁘네요.

작성자: winter

날짜: 2022-01-27

보조 메뉴: 1.Update | 2.Delete | 3.List

메뉴 선택: 1

[수정 내용 입력]

제목: 봄이 왔어요.

내용: 들에는 꽃들이 만발했네요.

작성자: spring

보조 메뉴: 1.Ok | 2.Cancel

메뉴 선택: 1

[게시물 목록]

no	writer	date	title
4	summer	2022-01-27	여름에 가장 시원할 때
3	spring	2022-01-27	봄이 왔어요.
2	winter	2022-01-27	크리스마스
1	winter	2022-01-27	눈 오는 날

메인 메뉴: 1.Create | 2.Read | 3.Clear | 4.Exit

메뉴 선택:

read() 메소드에서 보조 메뉴 '1.Update | 2.Delete | 3.List'를 추가하고, 보조 메뉴에서 '1.Update'를 선택하면 update() 메소드가, '2.Delete'를 선택하면 delete() 메소드가 호출되도록 한다. update() 메소드는 매개값으로 받은 Board 객체를 수정해서 boards 테이블의 게시물 정보를 수정하도록 한다.

>>> BoardExample6.java

```
1    //이상 동일
...
129
130    public void read() {
...
152        System.out.println("#####");
153        System.out.println("번호: " + board.getBno());
154        System.out.println("제목: " + board.getBtitle());
155        System.out.println("내용: " + board.getBcontent());
156        System.out.println("작성자: " + board.getBwriter());
157        System.out.println("날짜: " + board.getBdate());
158        //보조 메뉴 출력
159        System.out.println("-----");
160        System.out.println("보조 메뉴: 1.Update | 2.Delete | 3.List");
161        System.out.print("메뉴 선택: ");
162        String menuNo = scanner.nextLine();
163        System.out.println();
164
165        if(menuNo.equals("1")) {
```

```

166         update(board);
167     } else if(menuNo.equals("2")) {
168         delete(board);
169     }
170     ...
180 }
181
182 public void update(Board board) {
183     //수정 내용 입력받기
184     System.out.println("[수정 내용 입력]");
185     System.out.print("제목: ");
186     board.setBtitle(scanner.nextLine());
187     System.out.print("내용: ");
188     board.setBcontent(scanner.nextLine());
189     System.out.print("작성자: ");
190     board.setBwriter(scanner.nextLine());
191
192     //보조 메뉴 출력
193     System.out.println("-----");
194     System.out.println("보조 메뉴: 1.0k | 2.Cancel");
195     System.out.print("메뉴 선택: ");
196     String menuNo = scanner.nextLine();
197     if(menuNo.equals("1")) {
198         //boards 테이블에서 게시물 정보 수정
199         try {
200             String sql = "" +
201                 "UPDATE boards SET btitle=?, bcontent=?, bwriter=? " +
202                 "WHERE bno=?";
203             PreparedStatement pstmt = conn.prepareStatement(sql);
204             pstmt.setString(1, board.getBtitle());
205             pstmt.setString(2, board.getBcontent());
206             pstmt.setString(3, board.getBwriter());
207             pstmt.setInt(4, board.getBno());
208             pstmt.executeUpdate();
209             pstmt.close();
210         } catch (Exception e) {
211             e.printStackTrace();
212             exit();
213         }
214     }

```

```

215
216         //게시물 목록 출력
217         list();
218     }
219
220     //이하 동일
...     ...

```

게시물 삭제 기능

게시물 읽기의 보조 메뉴에서 '2.Delete'를 선택했을 때 boards 테이블에서 해당 게시물을 삭제하도록 해보자.

실행 결과

[게시물 목록]

no	writer	date	title
4	summer	2022-01-27	여름에 가장 시원할 때
3	spring	2022-01-27	봄이 왔어요.
2	winter	2022-01-27	크리스마스
1	winter	2022-01-27	눈 오는 날

메인 메뉴: 1.Create | 2.Read | 3.Clear | 4.Exit

메뉴 선택: 2

[게시물 읽기]

bno: 4

#####

번호: 4

제목: 여름에 가장 시원할 때

내용: 에이컨이 나오는 강의실에서 자바 공부할 때입니다^^.

작성자: summer

날짜: 2022-01-27

보조 메뉴: 1.Update | 2.Delete | 3.List

메뉴 선택: 2

[게시물 목록]

no	writer	date	title
3	spring	2022-01-27	봄이 왔어요.
2	winter	2022-01-27	크리스마스
1	winter	2022-01-27	눈 오는 날

메인 메뉴: 1.Create | 2.Read | 3.Clear | 4.Exit

메뉴 선택:

게시물 수정 기능을 구현할 때 보조 메뉴에서 '2.Delete'를 선택했을 때 `delete()` 메소드가 호출되도록 하였다. `delete()` 메소드를 다음과 같이 수정해 매개값으로 받은 `Board` 객체에서 `bno`를 얻어 `boards` 테이블에서 해당 게시물을 삭제하도록 한다.

>>> BoardExample7.java

```
1      //이상 동일
...      ...

219
220      public void delete(Board board) {
221          //boards 테이블에 게시물 정보 삭제
222          try {
223              String sql = "DELETE FROM boards WHERE bno=?";
224              PreparedStatement pstmt = conn.prepareStatement(sql);
225              pstmt.setInt(1, board.getBno());
226              pstmt.executeUpdate();
227              pstmt.close();
228          } catch (Exception e) {
229              e.printStackTrace();
230              exit();
231          }
232
233          //게시물 목록 출력
234          list();
235      }
236
237      //이하 동일
...      ...
```

게시물 전체 삭제 기능

메인 메뉴에서 '3.Clear'를 선택하고 보조 메뉴에서 '1.Ok'를 선택했을 때 boards 테이블의 전체 게시물 정보를 삭제하도록 해보자.

실행 결과

[게시물 목록]

no	writer	date	title
3	spring	2022-01-27	봄이 왔어요.
2	winter	2022-01-27	크리스마스
1	winter	2022-01-27	눈 오는 날

메인 메뉴: 1.Create | 2.Read | 3.Clear | 4.Exit

메뉴 선택: 3

[게시물 전체 삭제]

보조 메뉴: 1.Ok | 2.Cancel

메뉴 선택: 1

[게시물 목록]

no	writer	date	title
----	--------	------	-------

메인 메뉴: 1.Create | 2.Read | 3.Clear | 4.Exit

메뉴 선택:

메인 메뉴에서 '3.Clear'를 선택했을 때 호출되는 clear() 메소드를 다음과 같이 수정한다.

>>> BoardExample8.java

```
1    //이상 동일
...    ...
236
237    public void clear() {
```

```

238     System.out.println("[게시물 전체 삭제]");
239     System.out.println("-----");
240     System.out.println("보조 메뉴: 1.0k | 2.Cancel");
241     System.out.print("메뉴 선택: ");
242     String menuNo = scanner.nextLine();
243     if(menuNo.equals("1")) {
244         //boards 테이블에 게시물 정보 전체 삭제
245         try {
246             String sql = "TRUNCATE TABLE boards";
247             PreparedStatement pstmt = conn.prepareStatement(sql);
248             pstmt.executeUpdate();
249             pstmt.close();
250         } catch (Exception e) {
251             e.printStackTrace();
252             exit();
253         }
254     }
255
256     //게시물 목록 출력
257     list();
258 }
259
260 //이하 동일
...

```

종료 기능

메인 메뉴에서 '4.Exit'를 선택하면 Connection을 닫고 프로그램을 종료시켜 보자.

실행 결과

[게시물 목록]

```

-----
no      writer      date      title
-----

```

메인 메뉴: 1.Create | 2.Read | 3.Clear | 4.Exit

메뉴 선택: 4

** 게시판 종료 **

메인 메뉴에서 '4.Exit'를 선택했을 때 호출되는 `exit()` 메소드를 다음과 같이 수정한다.

>>> BoardExample9.java

```
1      //이상 동일
...
259
260      public void exit() {
261          if(conn != null) {
262              try {
263                  conn.close();
264              } catch (SQLException e) {
265              }
266          }
267          System.out.println("** 게시판 종료 **");
268          System.exit(0);
269      }
270
271      //이하 동일
...
...
```

NOTE ▶ 전체 코드는 예제소스 파일을 참조한다.

게시판 구현은 여기에서 마무리를 짓고, 사용자 가입 기능 및 로그인 기능은 배운 내용을 토대로 상상력을 발휘해서 구현해보길 바란다.

Appendix

02

▶ Java UI – Swing

- 01 Swing 소개
- 02 이벤트 디스패칭 스레드
- 03 Swing 컨테이너
- 04 컴포넌트 배치
- 05 이벤트 처리
- 06 버튼 컴포넌트
- 07 텍스트 컴포넌트
- 08 리스트 컴포넌트
- 09 테이블 컴포넌트
- 10 트리 컴포넌트
- 11 메뉴 컴포넌트
- 12 툴바 컴포넌트
- 13 다이얼로그
- 14 2D 그래픽스
- 15 Swing 과제

01 Swing 소개

UI User Interface 프로그램은 윈도우, 메뉴, 버튼, 라디오, 리스트 등 시각적인 컴포넌트를 제공해서 사용자와 상호작용하도록 돕는다. 자바는 이러한 UI 프로그램을 개발할 수 있도록 JDK에서 JFC^{Java Foundation Classes}를 제공한다.

JFC는 UI 프로그램을 만들기 위한 클래스들의 모음으로, AWT^{Abstract Window Toolkit}와 Swing (스윙)을 제공하고 있다. AWT는 java.awt 패키지로, Swing은 javax.swing 패키지로 사용 가능하다.

AWT는 운영체제가 가지고 있는 컴포넌트를 그대로 이용하고, Swing은 자바에서 직접 컴포넌트를 만든다는 점이 다르다. 따라서 AWT는 여러 운영체제들이 공통적으로 가지고 있는 컴포넌트만 사용하므로 컴포넌트 수가 제한적이지만, Swing은 자바에서 직접 제공하는 컴포넌트이기 때문에 종류가 매우 다양하다. Swing의 단점은 자바가 직접 컴포넌트를 생성하기 때문에 AWT에 비해 CPU와 메모리를 상대적으로 많이 사용한다는 것이다.

다음은 AWT로 작성한 간단한 윈도우 프로그램이다. import되는 내용을 보면 모두 java.awt 패키지를 사용하고 있는 것을 볼 수 있다.

>>> App.java

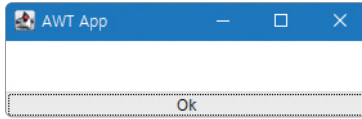
```
1  package sec01.exam01_awt;
2
3  import java.awt.BorderLayout;
4  import java.awt.Button;
5  import java.awt.Frame;
6  import java.awt.event.WindowAdapter;
7  import java.awt.event.WindowEvent;
8
9  //Frame 상속
10 public class App extends Frame {
11     public App() {
12         //제목 설정
13         setTitle("AWT App");
14         //윈도우 크기 설정
15         setSize(300, 100);
16         //Button 추가
17         add(new Button("Ok"), BorderLayout.SOUTH);
```

```

18      //윈도우 종료 버튼을 클릭하면 프로세스 종료
19      addWindowListener(new WindowAdapter() {
20          @Override
21          public void windowClosing(WindowEvent e) {
22              System.exit(0);
23          }
24      });
25  }
26
27  public static void main(String[] args) {
28      //윈도우 생성
29      App app = new App();
30      //윈도우를 보여줌
31      app.setVisible(true);
32  }
33  }

```

실행 결과



다음은 Swing으로 작성한 간단한 윈도우 프로그램이다. Swing은 AWT가 제공하는 것을 공유하기 때문에 import되는 내용을 보면 java.awt 패키지와 javax.swing 패키지를 사용하고 있는 것을 볼 수 있다.

>>> App.java

```

1  package sec01.exam02_swing;
2
3  import java.awt.BorderLayout;
4  import java.awt.event.WindowAdapter;
5  import java.awt.event.WindowEvent;
6  import javax.swing.JButton;
7  import javax.swing.JFrame;

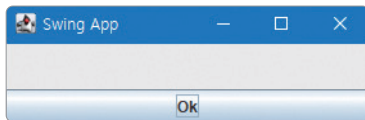
```

```

8
9 //JFrame 상속
10 public class App extends JFrame {
11     public App() {
12         //제목 설정
13         setTitle("Swing App");
14         //윈도우 크기 설정
15         setSize(300, 100);
16         //JButton 추가
17         getContentPane().add(new JButton("Ok"), BorderLayout.SOUTH);
18         //윈도우 종료 버튼을 클릭하면 프로세스 종료
19         addWindowListener(new WindowAdapter() {
20             @Override
21             public void windowClosing(WindowEvent e) {
22                 System.exit(0);
23             }
24         });
25     }
26
27     public static void main(String[] args) {
28         //윈도우 생성
29         App app = new App();
30         //윈도우를 보여줌
31         app.setVisible(true);
32     }
33 }

```

실행 결과

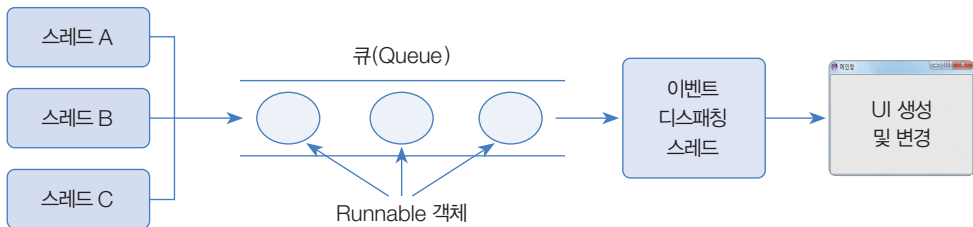


윈도우를 만들기 위해 AWT는 Frame을 상속했고, Swing은 JFrame을 상속했다. 그리고 버튼은 AWT는 Button을, Swing은 JButton을 사용하고 있다. Swing의 UI 관련 클래스는 AWT의 클래스에 J를 붙였기 때문에 쉽게 구분할 수 있다.

02 이벤트 디스패칭 스레드

Swing은 스레드에 안전하지 않기 때문에 작업 스레드들이 동시에 접근해서 UI를 변경하게 되면 문제가 발생할 수 있다. 그래서 Swing은 이벤트 디스패칭 스레드(event-dispatching thread)에 의해 순차적으로 UI 변경 작업을 진행하도록 설계되어 있다.

작업 스레드에서 UI를 생성하거나 변경하는 작업이 필요할 때에는 작업해야 할 내용을 Runnable 객체로 생성한 뒤, 큐(queue)에 저장해 놓는다. 그러면 이벤트 디스패칭 스레드(Event Dispatching Thread)가 순차적으로 큐에 있는 Runnable 객체를 꺼내어 실행하면서 UI를 생성하거나 변경시킨다.



작업 스레드는 SwingUtilities 클래스의 invokeLater() 메소드를 이용해서 큐(queue)에 Runnable 객체를 저장한다. 메소드 이름이 invokeLater 인 이유는 큐에 먼저 저장된 Runnable을 처리하고 나중에 처리한다는 의미이다. 다음은 invokeLater() 메소드를 호출하는 방법을 보여 준다.

>>> App.java

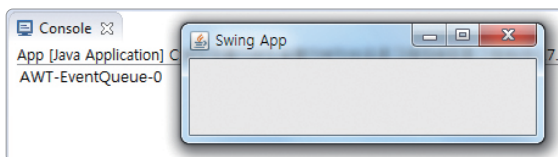
```
1 package sec02.exam01_invokeLater;
2
3 import java.awt.event.WindowAdapter;
4 import java.awt.event.WindowEvent;
5 import javax.swing.JFrame;
6 import javax.swing.SwingUtilities;
7
8 //JFrame 상속
9 public class App extends JFrame {
10     public App() {
11         //제목 설정
12         setTitle("Swing App");
13         //윈도우 크기 설정
```

```

14     setSize(300, 100);
15     //윈도우 종료 버튼을 클릭하면 프로세스 종료
16     addWindowListener(new WindowAdapter() {
17         @Override
18         public void windowClosing(WindowEvent e) {
19             System.exit(0);
20         }
21     });
22 }
23
24 public static void main(String[] args) {
25     //이벤트 큐에 Runnable 넣기
26     SwingUtilities.invokeLater(new Runnable() {
27         public void run() {
28             //윈도우 생성
29             App app = new App();
30             //윈도우를 보여줌
31             app.setVisible(true);
32             System.out.println(Thread.currentThread().getName());
33         }
34     });
35 }
36 }

```

실행 결과



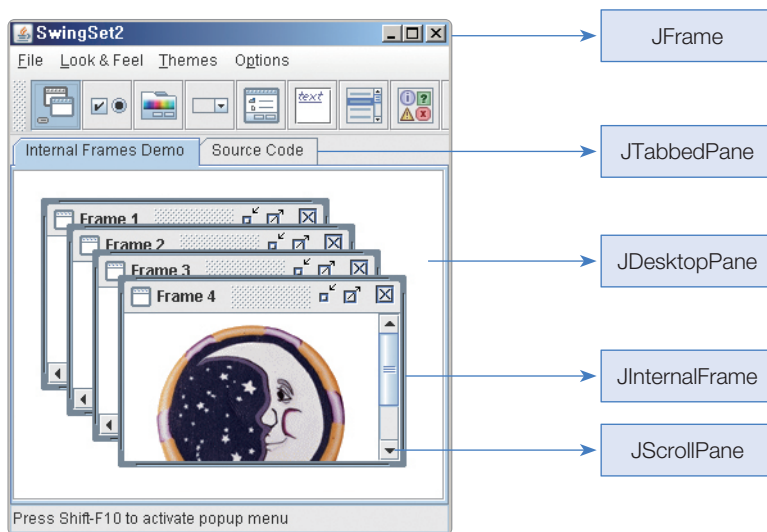
Runnable 객체를 실행하는 실제 스레드가 무엇인지 32라인에서 스레드의 이름을 출력하게 했다. 출력 내용을 보면 AWT-EventQueue-0이 출력되었는데, 이것이 이벤트 디스패칭 스레드의 이름이다.

03 Swing 컨테이너

윈도우 창과 같이 버튼, 라디오, 체크박스, 텍스트 필드 등의 컴포넌트를 배치할 수 있는 클래스를 컨테이너라고 한다. Swing은 기능에 따라 컨테이너 클래스를 다음과 같이 제공한다.

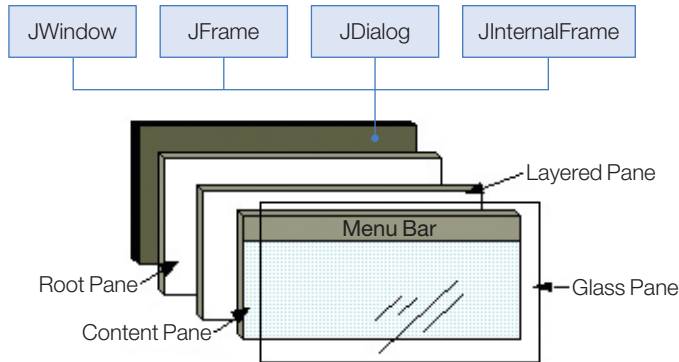
컨테이너 클래스	용도
JDesktopPane	내부 윈도우를 여러 개 포함할 수 있는 MDI 프로그램을 만들 때 사용
JDialog	다이얼로그 윈도우를 만들 때 사용
JFrame	작업 표시줄, 메뉴가 제공되는 윈도우를 만들 때 사용
JInternalFrame	JDesktopPane에 포함되는 내부 윈도우를 만들 때 사용
JPanel	컴포넌트들을 배치할 때 사용
JScrollPane	수직 또는 수평 스크롤이 필요할 때 사용
JSplitPane	수직 또는 수평으로 보여주는 크기를 조절할 때 사용
JTabbedPane	여러 가지 탭을 제공할 때 사용
JWindow	작업 표시줄, 메뉴가 없는 윈도우를 만들 때 사용

JWindow, JFrame, JDialog는 완전한 윈도우 창 형태를 갖고 있는 최상위 레벨 컨테이너^{TopLevel Containers}이다. 이들을 제외한 나머지는 최상위 레벨 컨테이너의 내부에서 사용되는 보조 컨테이너들이다.



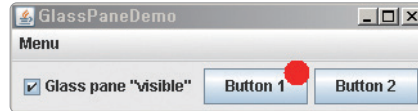
Swing 컨테이너 구조

JWindow, JFrame, JDialog, JInternalFrame 컨테이너는 다른 컨테이너와 달리 기본 판^{Root Pane} 위에 다음과 같이 여러 겹의 판^{Pane}으로 구성되어 있다.



1) GlassPane

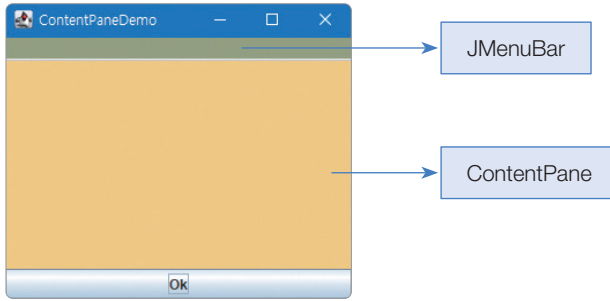
GlassPane은 다른 패널 위에 존재하면서 기본적으로 숨겨져 있는 투명한 판이다. GlassPane에 대한 이해를 돕기 위해 책과 함께 제공되는 소스에서 GlassPaneDemo.java를 찾아 이클립스에서 실행해 보자.



GlassPaneDemo 대화상자에서 Glass pane “visible”의 체크박스에 체크하지 않은 상태에서는 다른버튼을 클릭할 수 있지만 체크하면 GlassPane이 최상판이 되어 바로 밑에 있는 버튼과 메뉴들을 사용할 수 없다. 대신 GlassPane에 그려지는 동그란 빨간 점만 볼 수 있다. 마치 유리로 덮여있는 진열대에서 유리 속의 물건을 만질 수 없는 이치와 같다.

2) JMenuBar와 ContentPane

JMenuBar는 메뉴가 포함되는 판이고, 그 아래쪽에 있는 ContentPane은 버튼과 같은 UI 컴포넌트가 배치되는 판이다. ContentPane과 JMenuBar에 대한 이해를 돕기 위해 예제 소스에서 ContentPaneDemo.java를 찾아 이클립스에서 실행해 보자.



JMenuBar는 JFrame의 `setJMenuBar()` 메소드로 추가할 수 있다. 하지만 버튼과 같은 UI 컴포넌트는 JWindow, JFrame, JDialog, JInternalFrame에 직접 배치할 수 없고 `getContentPane()` 메소드를 이용해서 ContentPane을 얻은 후에 배치해야 한다.

다음 코드는 JFrame에서 JMenuBar를 추가하고, ContentPane 남쪽에 JButton을 배치한다.

```
JFrame jFrame = new JFrame();

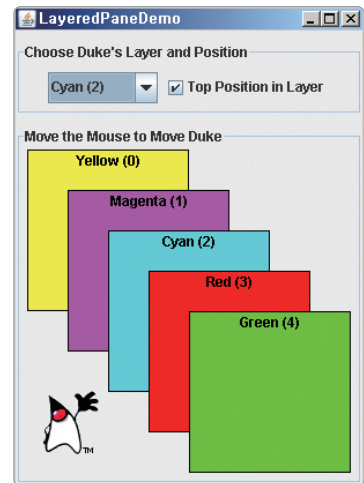
//MenuBar 추가
jFrame.setJMenuBar(new JMenuBar());

//컴포넌트 추가
jFrame.getContentPane().add(new JButton("확인"), BorderLayout.SOUTH);
```

3) LayeredPane

여러 컴포넌트들이 겹쳐질 때 각 컴포넌트의 상하 위치를 결정한다. LayeredPane에 대한 이해를 돕기 위해 예제 소스에서 LayeredPaneDemo.java를 찾아 이클립스에서 실행해 보자.

손 흔드는 듀크^{duke}에 마우스를 올려 놓으면 듀크가 마우스를 따라 움직인다. 듀크를 Yellow, Magenta, Cyan 영역에 갖다 놓으면 잘 보이지만, Red, Green 영역에 갖다 놓으면 가려진다. 듀크가 Cyan 레이어 위에 위치하기 때문이다. 드롭다운 리스트에서 듀크의 위치를 변경해 보면 레이어 개념을 알 수 있을 것이다.



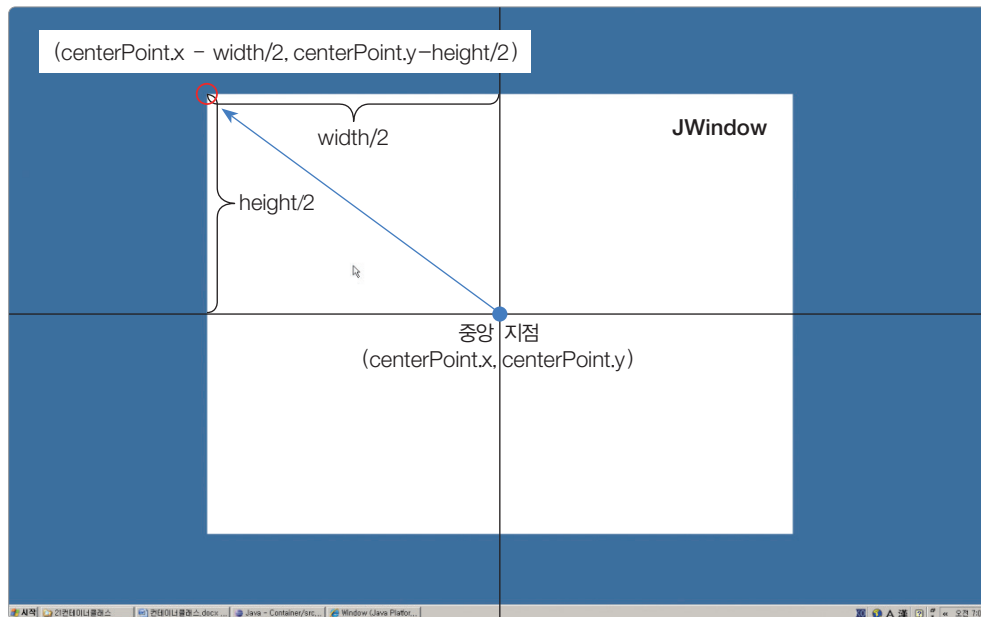
JWindow

JWindow는 윈도우 경계선, 제목 표시줄, 메뉴바가 모두 없는 윈도우를 만드는 컨테이너로, 컴포넌트만 배치할 수 있는 평면 공간만을 갖는다. 게임 애플리케이션처럼 제목 표시줄이 없는 윈도우를 만들 때 주로 이용된다.

새로운 개발자 정의 윈도우는 다음과 같이 JWindow를 상속해서 만들 수 있다.

```
public class JWindowExample extends JWindow {  
    public JWindowExample() {  
        this.setSize(450, 300);  
    }  
}
```

윈도우는 반드시 폭width과 높이height가 있어야 하기 때문에 생성자에서 setSize() 메소드로 폭과 높이를 주면 된다. 윈도우를 화면 중앙에 띄우기 위해서는 화면의 중앙 지점을 얻어서 윈도우의 좌측 상단 모서리의 좌표를 계산해야 한다.



다음은 JWindow를 화면 중앙에 띄우기 위해 필요한 코드들이다.

```
GraphicsEnvironment ge = GraphicsEnvironment.getLocalGraphicsEnvironment();
Point centerPoint = ge.getCenterPoint();
int leftTopX = centerPoint.x - getWidth()/2;
int leftTopY = centerPoint.y - getHeight()/2;
this.setLocation(leftTopX, leftTopY);
```

java.awt.GraphicsEnvironment는 그래픽 환경에 대한 정보를 가지고 있는 객체이다. 이 객체는 정적^{static} 메소드인 `getLocalGraphicsEnvironment()`를 호출해서 얻을 수 있다.

GraphicsEnvironment의 `getContentPoint()` 메소드는 화면 중앙 지점의 X좌표와 Y좌표를 가지고 있는 Point 객체를 리턴한다. 이렇게 얻은 화면 중앙 좌표와 윈도우 폭, 높이로 JWindow의 좌측 상단 모서리 좌표를 계산할 수 있다. 그런 다음 JWindow의 `setLocation()` 메소드로 좌측 상단 모서리 좌표를 설정해주면 된다.

JWindow를 화면에 띄우려면 `setVisible(true)` 메소드를 호출하면 된다.

```
JWindowExample jWindow = new JWindowExample();
jWindow.setVisible(true);
```

반대로 `setVisible(false)`을 호출하면 JWindow가 화면에서 사라지는데, 이것은 JWindow가 화면에서 완전히 제거되는 것이 아니라 단지 숨겨질 뿐이다. 다시 `setVisible(true)`을 호출하면 언제든 나타난다. 만약 JWindow를 화면에서 완전히 제거하고 싶다면 `dispose()` 메소드를 호출하면 된다.

```
jWindow.dispose();
```

다음 예제는 JWindow를 이용해서 게임을 시작할 때 보여줄 로고 윈도우를 만든다.

>>> JWindowExample.java

```
1  package sec03.exam02_jwindow;
2
3  import java.awt.BorderLayout;
4  import java.awt.GraphicsEnvironment;
5  import java.awt.Point;
6  import java.awt.event.MouseAdapter;
7  import java.awt.event.MouseEvent;
8  import javax.swing.ImageIcon;
9  import javax.swing.JLabel;
10 import javax.swing.JWindow;
11 import javax.swing.SwingUtilities;
12
13 public class JWindowExample extends JWindow {
14     public JWindowExample() {
15         //JWindow의 크기
16         this.setSize(600, 350);
17
18         //JWindow를 화면 중앙으로 띄우기
19         GraphicsEnvironment ge = GraphicsEnvironment.getLocalGraphicsEnvironment();
20         Point centerPoint = ge.getCenterPoint();
21         int leftTopX = centerPoint.x - this.getWidth()/2;
22         int leftTopY = centerPoint.y - this.getHeight()/2;
23         this.setLocation(leftTopX, leftTopY);
24
25         //JWindow에 이미지가 포함된 JLabel 추가
26         JLabel label = new JLabel();
27         label.setIcon(new ImageIcon(getClass().getResource("game.png")));
28         getContentPane().add(label, BorderLayout.CENTER);
29
30         //마우스 클릭 이벤트 처리
31         this.addMouseListener(new MouseAdapter() {
32             @Override
33             public void mouseClicked(MouseEvent e) {
34                 dispose();
35             }
36         });
37     }
38 }
```

```

39     public static void main(String[] args) {
40         SwingUtilities.invokeLater(new Runnable() {
41             public void run() {
42                 JWindowExample jWindow = new JWindowExample();
43                 jWindow.setVisible(true);
44             }
45         });
46     }
47 }

```

실행 결과



JWindow에 이미지를 넣기 위해 26~27 라인에서 JLabel 컴포넌트를 활용하였다. JLabel은 글자 및 이미지를 포함할 수 있는 컴포넌트인데, setIcon() 메소드로 ImageIcon 객체를 매개값으로 주면 이미지를 나타낼 수 있다.

ImageIcon 생성자는 이미지 파일의 URL 객체를 매개값으로 받는데, JWindowExample.class 와 동일한 폴더에 있는 'game.png' 파일에 대한 URL 객체를 얻기 위해 다음 코드를 사용하였다.

```

getClass().getResource("game.png")

```

28라인은 JLabel을 JWindow의 중앙에 배치시킨다. 컴포넌트 배치는 4절에서 설명한다. 31~36 라인은 마우스로 JWindow를 클릭했을 때 JWindow를 제거하기 위해 이벤트를 처리한 것이다. 이벤트 처리는 5절에서 설명한다.

JFrame

JFrame은 JWindow와는 달리 윈도우 경계선, 제목 표시줄, 메뉴바가 있는 윈도우를 만드는 컨테이너 클래스이다. 새로운 개발자 정의 윈도우를 만들기 위해서는 다음과 같이 JFrame을 상속해서 만들 수 있다.

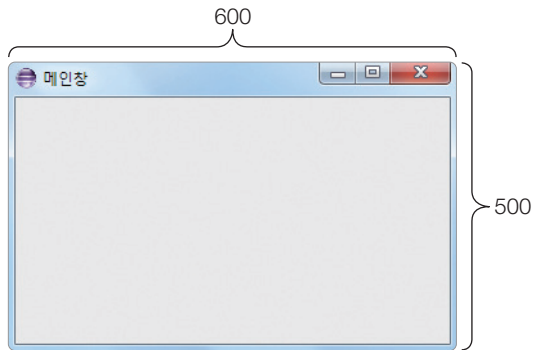
```
public class JFrameExample extends JFrame {
    public JFrameExample() {
        //제목 표시줄
        this.setIconImage(
            new ImageIcon(getClass().getResource("icon.png")).getImage());
        this.setTitle("창제목");
        //JFrame 크기
        this.setSize(600, 500);
        //종료 버튼의 기본 기능
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

JFrame의 제목 표시줄은 아이콘, 제목, 크기 조절용 버튼, 종료 버튼으로 구성된다.

아이콘은 `setIconImage()` 메소드로 설정하면 되는데, `ImageIcon` 객체의 `getImage()` 메소드로 `Image` 객체를 얻어 매개값으로 설정하면 된다.

창 제목은 `setTitle()` 메소드로 설정할 수

있다. 종료 버튼의 기본 기능은 JFrame을 단순히 숨기기만 하고 프로세스를 종료하지 않는다. 프로세스를 완전히 종료하려면 `setDefaultCloseOperation()` 메소드로 종료 버튼의 기본 기능을 변경해야 한다. `setDefaultCloseOperation()` 메소드에 지정할 수 있는 종료 버튼의 기능별 매개값은 다음 네 가지 종류가 있다.



기능별 상수	설명
WindowConstants.DO_NOTHING_ON_CLOSE	아무 것도 하지 않음
WindowConstants.HIDE_ON_CLOSE	화면에서 JFrame 숨김 (기본)
WindowConstants.DISPOSE_ON_CLOSE	화면에서 JFrame 완전히 제거, 다른 JFrame이 없다면 윈도우 프로세스를 종료
JFrame.EXIT_ON_CLOSE	윈도우 프로세스를 종료

창 크기는 `setSize()`로 폭과 높이값을 주고, JFrame을 화면의 중앙 부분에 위치시키는 방법은 JWindow에서 설명한 것과 동일하다. 화면 중앙 좌표와 윈도우 폭, 높이로 JFrame의 좌측 상단 모서리 좌표를 구한 다음, `setLocation()` 메소드로 설정해주면 된다.

JFrame를 화면에 띄우려면 `setVisible(true)` 메소드를 다음과 같이 호출하면 된다.

```
JFrameExample jFrame = new JFrameExample ();
jFrame.setVisible(true);
```

반대로 `setVisible(false)`를 호출하면 JFrame이 화면에서 사라지는데, 이것은 JFrame이 화면에서 완전히 제거되는 것이 아니라 숨겨질 뿐이다. 다시 `setVisible(true)`를 호출하면 언제든지 나타난다. 만약 JFrame를 화면에서 완전히 제거하고 싶다면 `dispose()` 메소드를 호출하면 된다.

```
jFrame.dispose();
```

다음 예제는 JFrame으로 제목 표시줄이 있는 윈도우를 띄우고 종료하는 방법을 보여 준다.

>>> JFrameExample.java

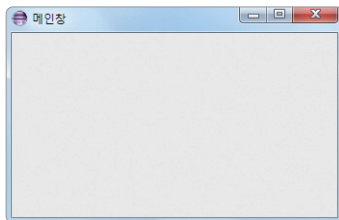
```
1  package sec03.exam03_jframe;
2
3  import java.awt.GraphicsEnvironment;
4  import java.awt.Point;
5  import javax.swing.ImageIcon;
6  import javax.swing.JFrame;
7  import javax.swing.SwingUtilities;
```

```

8
9 public class JFrameExample extends JFrame {
10     public JFrameExample() {
11         //JWindow의 크기
12         this.setSize(600, 500);
13
14         //제목 표시줄 내용
15         this.setIconImage(new ImageIcon(getClass().getResource("icon.png")).
            getImage());
16         this.setTitle("메인창");
17
18         //종료 버튼의 기본 기능
19         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
20
21         //JWindow를 화면 중앙으로 띄우기
22         GraphicsEnvironment ge = GraphicsEnvironment.getLocalGraphicsEnvironment();
23         Point centerPoint = ge.getCenterPoint();
24         int leftTopX = centerPoint.x - this.getWidth()/2;
25         int leftTopY = centerPoint.y - this.getHeight()/2;
26         this.setLocation(leftTopX, leftTopY);
27     }
28
29     public static void main(String[] args) {
30         SwingUtilities.invokeLater(new Runnable() {
31             public void run() {
32                 JFrameExample jFrame = new JFrameExample();
33                 jFrame.setVisible(true);
34             }
35         });
36     }
37 }

```

실행 결과



JTabbedPane

JTabbedPane은 탭^{tab}별로 다른 내용을 보여주기 위해 사용되는 컨테이너이다. JTabbedPane은 독립적인 윈도우 모양을 갖고 있지 않기 때문에 JWindow, JFrame, JDialog 등과 같은 최상위 레벨 컨테이너에 배치된다.

오른쪽 화면은 JTabbedPane의 구현 모습이다. 탭의 위치는 상단, 하단, 왼쪽, 오른쪽에 위치시킬 수 있다. JTabbedPane을 생성하려면 다음과 같이 기본 생성자를 호출하면 된다.



```
JTabbedPane jTabbedPane = new JTabbedPane();
```

탭의 위치를 설정하려면 `setTabPlacement()` 메소드로 탭의 위치 상수를 다음과 같이 지정한다.

```
jTabbedPane.setTabPlacement(  
    JTabbedPane.TOP | JTabbedPane.BOTTOM | JTabbedPane.LEFT | JTabbedPane.RIGHT  
);
```

JTabbedPane에 탭을 추가하려면 `addTab()` 메소드를 이용한다. `addTab()` 메소드는 탭의 이름과 탭안에 배치될 컴포넌트를 매개값으로 받는데, 컴포넌트는 주로 JPanel을 객체를 사용한다.

```
jTabbedPane.addTab("TapName1", jPanel1);  
jTabbedPane.addTab("TapName2", jPanel2);
```

그리고 각각의 JPanel 안에는 해당 탭에서 보여줄 컴포넌트를 배치하면 된다. 이렇게 생성된 JTabbedPane은 다른 컴포넌트와 마찬가지로 ContentPane에 배치된다. 다음은 JFrame의 중앙에 JTabbedPane을 배치하는 코드이다.

```
jFrame.getContentPane().add("Center", jTabbedPane);
```


다음 예제는 두 개의 탭을 추가한 다음 각 탭을 클릭하면 두 개의 JPanel이 이미지를 교체해 보여 준다.

>> JTabbedPaneExample.java

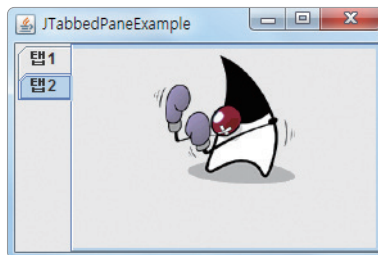
```
1  package sec03.exam04_jtabbedpane;
2
3  import java.awt.BorderLayout;
4
5  import javax.swing.ImageIcon;
6  import javax.swing.JFrame;
7  import javax.swing.JLabel;
8  import javax.swing.JPanel;
9  import javax.swing.JTabbedPane;
10 import javax.swing.SwingUtilities;
11
12 public class JTabbedPaneExample extends JFrame {
13     private JTabbedPane jTabbedPane;
14     private JPanel tab1Panel;
15     private JPanel tab2Panel;
16
17     //메인 윈도우 설정
18     public JTabbedPaneExample() {
19         this.setTitle("JTabbedPaneExample");
20         this.setSize(300, 200);
21         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
22         this.getContentPane().add(getJTabbedPane(), BorderLayout.CENTER);
23     }
24
25     //JTabbedPane 생성 및 Tab 추가
26     private JTabbedPane getJTabbedPane() {
27         if (jTabbedPane == null) {
28             jTabbedPane = new JTabbedPane();
29             jTabbedPane.setTabPlacement(JTabbedPane.LEFT);
30             jTabbedPane.addTab("탭1", getTab1Panel());
31             jTabbedPane.addTab("탭2", getTab2Panel());
32         }
33         return jTabbedPane;
34     }
35
36     //Tab1에 추가된 JPanel 생성
```

```

37     private JPanel getTab1Panel() {
38         if(tab1Panel == null) {
39             tab1Panel = new JPanel();
40             JLabel jLabel = new JLabel();
41             jLabel.setIcon(new ImageIcon(getClass().getResource("duke1.gif")));
42             tab1Panel.add(jLabel);
43         }
44         return tab1Panel;
45     }
46
47     //Tab2에 추가될 JPanel 생성
48     private JPanel getTab2Panel() {
49         if(tab2Panel == null) {
50             tab2Panel = new JPanel();
51             JLabel jLabel = new JLabel();
52             jLabel.setIcon(new ImageIcon(getClass().getResource("duke2.gif")));
53             tab2Panel.add(jLabel);
54         }
55         return tab2Panel;
56     }
57
58     public static void main(String[] args) {
59         SwingUtilities.invokeLater(new Runnable() {
60             public void run() {
61                 JTabbedPaneExample jFrame = new JTabbedPaneExample();
62                 jFrame.setVisible(true);
63             }
64         });
65     }
66 }

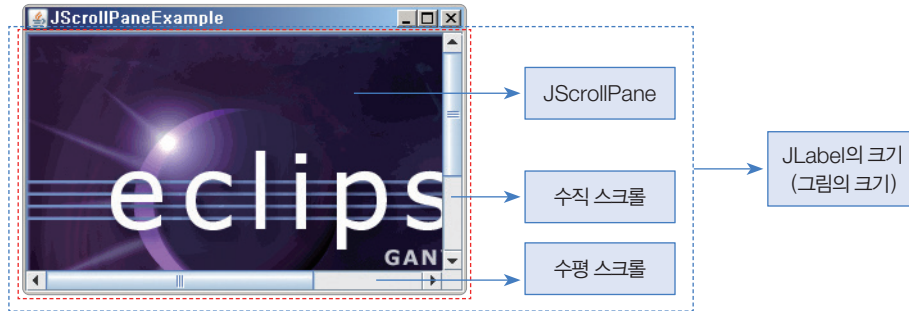
```

실행 결과



JScrollPane

JScrollPane은 포함된 컴포넌트의 크기가 JScrollPane 자신보다 큰 경우 수평 또는 수직 스크롤 바를 이용해서 볼 수 있게 해준다. JScrollPane은 다른 컨테이너와는 달리 단 하나의 컴포넌트만을 포함시킬 수 있다.



위 그림을 보면 JScrollPane은 JFrame의 중앙에 위치하고 있다. JScrollPane안에는 그림을 포함하고 있는 JLabel 배치되어 있는데, 그림의 크기가 JScrollPane보다 크기 때문에 수직 및 수평 스크롤이 생긴다.

스크롤이 필요한 컴포넌트에는 큰 내용을 포함하고 있는 JLabel, JTextArea, JList, JTable, JTree 등이 있다. 이 컴포넌트에 스크롤을 적용시키려면 다음과 같이 JScrollPane 생성자에 컴포넌트를 매개값으로 주면 된다. 이렇게 생성된 JScrollPane은 컴포넌트가 배치될 수 있는 곳이라면 어디든지 배치가 가능하다.

```
JScrollPane scrollJList = new JScrollPane(jLabel);
JScrollPane scrollJTextArea = new JScrollPane(jTextArea);
JScrollPane scrollJList = new JScrollPane(jList);
JScrollPane scrollJTable = new JScrollPane(jTable);
```

다음 예제에서는 JLabel에 큰 이미지를 넣고, JScrollPane에 JLabel을 추가시켰다. 그리고 JFrame 중앙에 JScrollPane을 배치시켰는데, JFrame의 사이즈가 이미지보다 작기 때문에 스크롤이 자동 생성된다.

>>> JScrollPaneExample.java

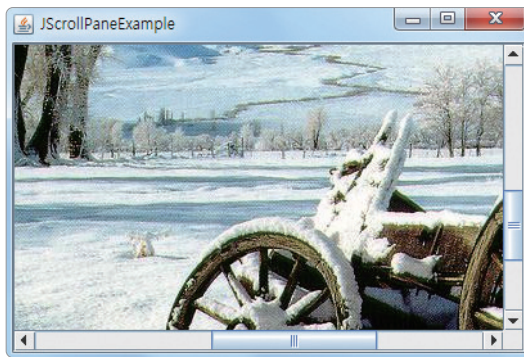
```
1  package sec03.exam05_jscrollpane;
2
3  import java.awt.BorderLayout;
4  import javax.swing.ImageIcon;
5  import javax.swing.JFrame;
6  import javax.swing.JLabel;
7  import javax.swing.JScrollPane;
8  import javax.swing.SwingUtilities;
9
10 public class JScrollPaneExample extends JFrame {
11     private JScrollPane scrollImage;
12     private JLabel lblImage;
13
14     //메인 윈도우 설정
15     public JScrollPaneExample() {
16         this.setTitle("JScrollPaneExample");
17         this.setSize(350, 230);
18         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
19         //JFrame 중앙에 JScrollPane 추가
20         this.getContentPane().add(getScrollImage(), BorderLayout.CENTER);
21     }
22
23     //JScrollPane 생성
24     private JScrollPane getScrollImage() {
25         if (scrollImage == null) {
26             scrollImage = new JScrollPane(getLblImage());
27         }
28         return scrollImage;
29     }
30
31     //JLabel 생성
32     public JLabel getLblImage() {
33         if(lblImage == null) {
34             lblImage = new JLabel();
35             lblImage.setIcon(new ImageIcon(getClass().getResource("snow.jpg")));
36         }
37         return lblImage;
38     }
```

```

39
40     public static void main(String[] args) {
41         SwingUtilities.invokeLater(new Runnable() {
42             public void run() {
43                 JScrollPaneExample jFrame = new JScrollPaneExample();
44                 jFrame.setVisible(true);
45             }
46         });
47     }
48 }

```

실행 결과



04 컴포넌트 배치

컨테이너에는 UI 컴포넌트들이 배치된다. 대표적인 컴포넌트에는 버튼, 체크박스, 라디오 버튼, 콤포, 리스트 등이 있다. 컨테이너는 기본적으로 배치 관리자로 컴포넌트를 배치한다.

배치 관리자는 좌표값으로 컴포넌트를 배치하지 않고, 컨테이너를 몇 개의 구획으로 나누어 하나의 구획에 하나의 컴포넌트를 배치해준다. 배치 관리자로 배치하게 되면 컨테이너의 크기가 사용자에게 의해 변경되더라도 컴포넌트의 크기가 비율적으로 늘거나 줄게되어 배치 모양이 그대로 유지된다는 장점이 있다.

크기가 고정된 컨테이너일 경우, 세밀한 배치를 위해서 좌표값으로 컴포넌트를 배치할 수도 있다.

이 경우 컨테이너의 좌측 상단 모서리를 (0,0)으로 보고, x축과 y축 좌표로 컴포넌트의 위치를 정해서 배치한다.

Layout Manager

컨테이너가 컴포넌트를 배치할 때에는 배치 관리자(Layout Manager)가 무엇이냐에 따라 달라진다. JWindow, JFrame, JDialog는 기본적으로 BorderLayout 배치 관리자를 사용하고, JPanel은 FlowLayout을 사용한다. 자바는 java.awt 패키지에서 다음과 같은 배치 관리자를 제공한다.

배치 관리자	설명
BorderLayout	동 · 서 · 남 · 북 · 중앙으로 컴포넌트를 배치
CardLayout	여러 장의 카드에 컴포넌트를 각각 배치
FlowLayout	왼쪽에서 오른쪽으로 컴포넌트를 배치
GridLayout	바둑판과 같은 격자에 컴포넌트를 배치
GridBagLayout	바둑판과 같은 격자에 컴포넌트를 배치하지만 격자 간 병합 가능

컨테이너의 기본 배치 관리자 대신 다른 것을 사용하고 싶다면 `setLayout()` 메소드로 변경할 수 있다. `setLayout()`의 매개변수 데이터 타입은 `LayoutManager` 인터페이스인데, 모든 배치 관리자의 인스턴스가 올 수 있다. 다음은 JFrame의 배치 관리자를 변경하는 방법을 보여 준다.

```
jFrame.getContentPane().setLayout( LayoutManager layoutManager );
```

```
new BorderLayout()
```

```
new FlowLayout()
```

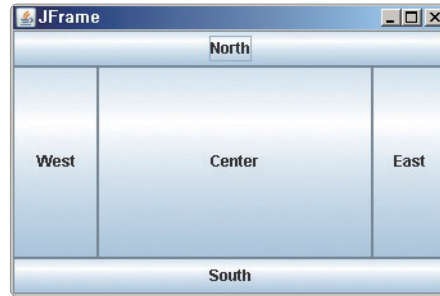
```
new GridLayout(2,3)
```

컨테이너의 배치 관리자 설정은 컴포넌트를 배치하기 전에 변경하는 것이 좋다. 컨테이너에서 사용하는 배치 관리자를 얻고 싶다면 컨테이너의 `getLayout()` 메소드를 호출하면 된다. `getLayout()`의 리턴타입은 `LayoutManager` 인터페이스이므로 다음과 같이 타입 변환을 해야 한다.

```
BorderLayout borderLayout = (BorderLayout) jFrame.getContentPane().getLayout();
```

BorderLayout

BorderLayout 배치 관리자는 컨테이너를 중앙·동·서·남·북으로 구획 짓고, 각 구획에 하나의 컴포넌트 또는 컨테이너를 배치한다. 일반적으로 각 구획에는 JPanel 컨테이너가 배치되어 복잡한 형태의 UI를 만들어낸다. 다음 그림은 JFrame의 각 구획에 JButton 컴포넌트를 배치한 것이다.



BorderLayout을 기본적으로 사용하는 컨테이너는 JWindow, JFrame, JDialog 등이 있다. BorderLayout이 적용된 컨테이너에 컴포넌트를 배치할 때에는 다음과 같이 ContentPane을 얻고 add() 메소드를 사용해야 한다.

```
jFrame.getContentPane().add(컴포넌트, BorderLayout.CENTER);
jFrame.getContentPane().add(컴포넌트, BorderLayout.EAST);
jFrame.getContentPane().add(컴포넌트, BorderLayout.WEST);
jFrame.getContentPane().add(컴포넌트, BorderLayout.SOUTH);
jFrame.getContentPane().add(컴포넌트, BorderLayout.NORTH);
```

첫 번째 매개값에는 배치할 컴포넌트 객체가 오고, 두 번째 매개값에는 어떤 구획에 배치할 것인지 지정하는 BorderLayout의 상수가 온다. 만약 동·서·남·북 중에서 컴포넌트가 배치되지 않은 구획이 있다면 중앙에 배치된 컴포넌트가 해당 구획까지 확장된다.

다음 예제는 중앙, 북쪽, 남쪽에만 컴포넌트를 배치하고, 동쪽과 서쪽은 배치하지 않았다. 그래서 중앙에 배치된 컴포넌트가 동쪽과 서쪽으로 확장되었다.

>>> BorderLayoutExample.java

```
1 package sec04.exam01_borderlayout;
2
3 import java.awt.BorderLayout;
4 import java.awt.Color;
5 import javax.swing.JButton;
6 import javax.swing.JFrame;
```

```

7  import javax.swing.JTextArea;
8  import javax.swing.JTextField;
9  import javax.swing.SwingUtilities;
10
11 public class BorderLayoutExample extends JFrame {
12     private JTextField txtNorth;
13     private JTextArea txtCenter;
14     private JButton btnSouth;
15
16     //메인 윈도우 설정
17     public BorderLayoutExample() {
18         this.setTitle("BorderLayoutExample");
19         this.setSize(300, 200);
20         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
21
22         //북쪽, 중앙, 남쪽에 컴포넌트 배치
23         this.getContentPane().add(getTxtNorth(), BorderLayout.NORTH);
24         this.getContentPane().add(getTxtCenter(), BorderLayout.CENTER);
25         this.getContentPane().add(getBtnSouth(), BorderLayout.SOUTH);
26     }
27
28     //JTextField 생성
29     private JTextField getTxtNorth() {
30         if (txtNorth == null) {
31             txtNorth = new JTextField();
32             txtNorth.setText("북쪽 컴포넌트");
33             txtNorth.setBackground(Color.YELLOW);
34         }
35         return txtNorth;
36     }
37
38     //JTextArea 생성
39     private JTextArea getTxtCenter() {
40         if (txtCenter == null) {
41             txtCenter = new JTextArea();
42             txtCenter.append("중앙 컴포넌트\n");
43             txtCenter.append("동쪽 컴포넌트가 없으니 동쪽으로 확장\n");
44             txtCenter.append("서쪽 컴포넌트가 없으니 서쪽으로 확장\n");
45         }
46         return txtCenter;

```

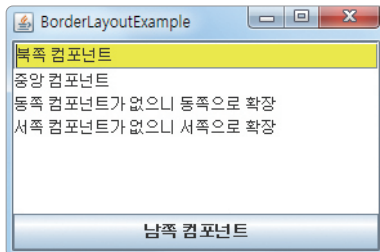


```

47     }
48
49     //JButton 생성
50     private JButton getBtnSouth() {
51         if (btnSouth == null) {
52             btnSouth = new JButton();
53             btnSouth.setText("남쪽 컴포넌트");
54         }
55         return btnSouth;
56     }
57
58     public static void main(String[] args) {
59         SwingUtilities.invokeLater(new Runnable() {
60             public void run() {
61                 BorderLayoutExample jFrame = new BorderLayoutExample();
62                 jFrame.setVisible(true);
63             }
64         });
65     }
66 }

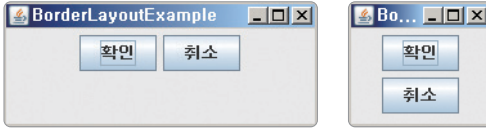
```

실행 결과



FlowLayout

FlowLayout 배치 관리자는 이미 배치된 컴포넌트의 오른쪽 옆에 새로운 컴포넌트를 배치한다. 오른쪽에 배치할 공간이 부족하면 하단에 배치하기 때문에 사용자에게 의해 컨테이너의 폭width이 변경되면 컴포넌트의 배치 위치가 변경될 수 있다.



FlowLayout이 적용된 컨테이너에 컴포넌트를 배치할 때는 컴포넌트만 매개변수로 갖는 add() 메소드를 사용한다. 예를 들어 JFrame이 FlowLayout을 사용하여 컴포넌트를 배치한다면, 다음과 같은 add() 메소드로 컴포넌트를 배치한다.

```
jFrame.getContentPane().setLayout(new FlowLayout());
jFrame.getContentPane().add(컴포넌트);
```

다음 예제는 JFrame의 배치 관리자로 FlowLayout을 적용하고 버튼 두 개를 배치한 것이다.

>>> FlowLayoutExample.java

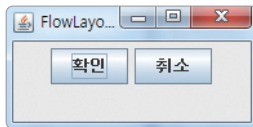
```
1  package sec04.exam02_flowlayout;
2
3  import java.awt.FlowLayout;
4  import javax.swing.JButton;
5  import javax.swing.JFrame;
6  import javax.swing.SwingUtilities;
7
8  public class FlowLayoutExample extends JFrame {
9      private JButton btnOk;
10     private JButton btnCancel;
11
12     //메인 윈도우 설정
13     public FlowLayoutExample() {
14         this.setTitle("FlowLayoutExample");
15         this.setSize(300, 100);
16         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
17
18         //FlowLayout으로 변경하고 두 개의 버튼 추가
19         this.setLayout(new FlowLayout());
20         this.getContentPane().add(getBtnOk());
21         this.getContentPane().add(getBtnCancel());
```

```

22     }
23
24     //Ok 버튼 생성
25     private JButton getBtnOk() {
26         if(btnOk == null) {
27             btnOk = new JButton();
28             btnOk.setText("확인");
29         }
30         return btnOk;
31     }
32
33     //Cancel 버튼 생성
34     private JButton getBtnCancel() {
35         if(btnCancel == null) {
36             btnCancel = new JButton();
37             btnCancel.setText("취소");
38         }
39         return btnCancel;
40     }
41
42     public static void main(String[] args) {
43         SwingUtilities.invokeLater(new Runnable() {
44             public void run() {
45                 FlowLayoutExample jFrame = new FlowLayoutExample();
46                 jFrame.setVisible(true);
47             }
48         });
49     }
50 }

```

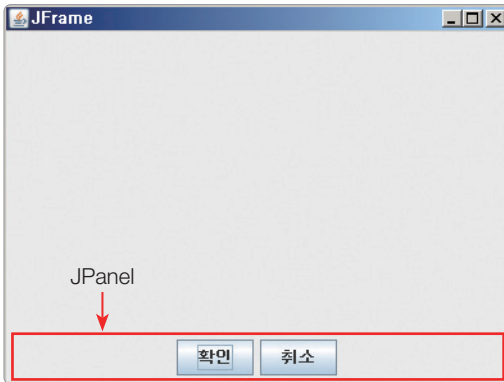
실행 결과



FlowLayout을 기본적으로 사용하는 컨테이너에는 JPanel이 있다. JPanel은 JWindow, JFrame, JDialog처럼 하나의 윈도우 창을 만드는 최상위 레벨 컨테이너가 아니라, 컨테이너 속에서

컴포넌트의 배치를 위해 사용되는 투명한 보조 컨테이너이다.

다음은 JFrame 남쪽에 JPanel을 추가하고, JPanel 내부에 JButton 두 개를 배치한 것이다.



JPanel은 컴포넌트가 배치될 수 있는 어떤 곳이라도 배치가 가능하다. 심지어 JPanel에 또다른 JPanel을 배치하는 것도 가능하다. JPanel은 기본적으로 FlowLayout을 사용하지만, 다음과 같이 `setLayout()` 메소드로 배치 관리자를 변경할 수도 있다.

```
JPanel jPanel = new JPanel();  
jPanel.setLayout(new BorderLayout());
```

JPanel은 JWindow, JFrame, JDialog처럼 Root Pane을 이용해서 컴포넌트를 관리하지 않기 때문에 `ContentPane`이 없이 JPanel의 `add()` 메소드로 컴포넌트를 배치하면 된다.

```
FlowLayout일 경우:    jPanel.add(컴포넌트);  
BorderLayout일 경우:  jPanel.add(컴포넌트, BorderLayout.CENTER);
```

JPanel을 사용하지 않고 JFrame, JDialog에서 복잡한 형태로 컴포넌트를 배치할 수 없기 때문에 거의 필수적으로 JPanel이 사용된다. 다음 예제는 JFrame의 남쪽에 JPanel을 배치하고 확인 및 취소 버튼을 JPanel에 배치한다.

>>> JPanelExample.java

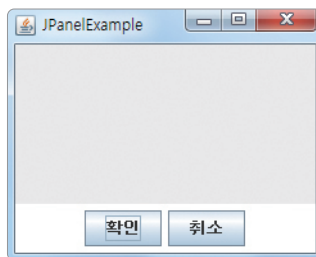
```
1  package sec04.exam03_jpanel;
2
3  import java.awt.BorderLayout;
4  import java.awt.Color;
5  import javax.swing.JButton;
6  import javax.swing.JFrame;
7  import javax.swing.JPanel;
8  import javax.swing.SwingUtilities;
9
10 public class JPanelExample extends JFrame {
11     private JPanel panelSouth;
12     private JButton btnOk;
13     private JButton btnCancel;
14
15     //메인 윈도우 설정
16     public JPanelExample() {
17         this.setTitle("JPanelExample");
18         this.setSize(250, 200);
19         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
20         //남쪽에 JPanel 추가
21         this.getContentPane().add(getPanelSouth(), BorderLayout.SOUTH);
22     }
23
24     //JPanel 생성
25     public JPanel getPanelSouth() {
26         if (panelSouth == null) {
27             panelSouth = new JPanel();
28             panelSouth.setBackground(Color.WHITE);
29             panelSouth.add(getBtnOk());
30             panelSouth.add(getBtnCancel());
31         }
32         return panelSouth;
33     }
34
35     //Ok 버튼 생성
36     public JButton getBtnOk() {
37         if (btnOk == null) {
38             btnOk = new JButton();
```

```

39         btnOk.setText("확인");
40     }
41     return btnOk;
42 }
43
44 //Cancel 버튼 생성
45 public JButton getBtnCancel() {
46     if (btnCancel == null) {
47         btnCancel = new JButton();
48         btnCancel.setText("취소");
49     }
50     return btnCancel;
51 }
52
53 public static void main(String[] args) {
54     SwingUtilities.invokeLater(new Runnable() {
55         public void run() {
56             JPanelExample jFrame = new JPanelExample();
57             jFrame.setVisible(true);
58         }
59     });
60 }
61 }

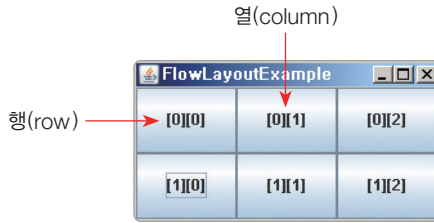
```

실행 결과



GridLayout

GridLayout 배치 관리자는 컨테이너를 행^{row}과 열^{column}로 구성된 테이블 모양으로 구획 짓고, 각 구획에 하나의 컴포넌트를 배치한다.



행과 열의 수는 GridLayout 객체를 생성할 때 생성자의 매개값으로 주거나, 객체 생성 후 `setRows()`, `setColumns()` 메소드로 지정할 수도 있다. 다음 코드는 생성자에서 행과 열의 수를 지정하여 GridLayout을 생성한 뒤 JFrame의 배치 관리자로 설정한다.

```
jFrame.getContentPane().setLayout(new GridLayout(행수, 열수));
```

GridLayout을 배치 관리자로 사용하는 컨테이너가 컴포넌트를 배치할 때는 FlowLayout과 마찬가지로 컴포넌트만 매개변수로 갖는 `add()` 메소드를 사용한다.

```
jFrame.getContentPane().add(컴포넌트);
```

컴포넌트 배치 순서는 첫 번째 행의 첫 번째 열부터 배치되고, 행의 마지막 열까지 배치가 끝나면 다음 행의 첫 번째 열부터 다시 차례대로 배치된다.

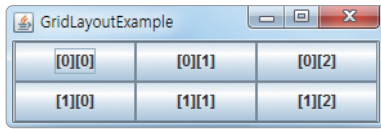
>>> GridLayoutExample.java

```
1 package sec04.exam04_gridlayout;
2
3 import java.awt.GridLayout;
4 import javax.swing.JButton;
5 import javax.swing.JFrame;
6 import javax.swing.SwingUtilities;
7
8 public class GridLayoutExample extends JFrame {
9     private JButton[][] btn;
10
11     //메인 윈도우 설정
```

```

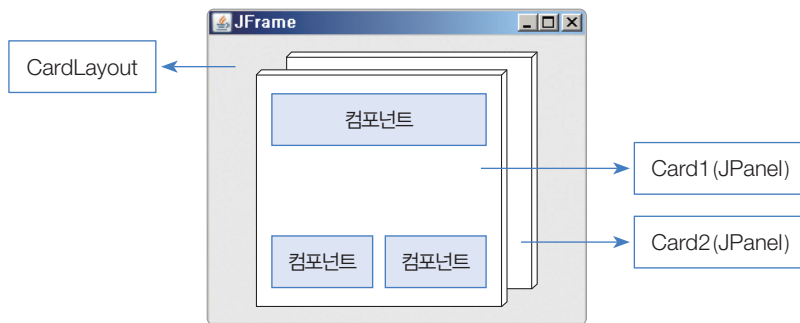
12     public GridLayoutExample() {
13         setTitle("GridLayoutExample");
14         setSize(300, 100);
15         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
16
17         //GridLayout으로 변경하고 버튼 추가
18         setLayout(new GridLayout(2,3));
19         for(int r = 0; r<2; r++) {
20             for(int c=0; c<3; c++) {
21                 getContentPane().add(getBtn()[r][c]);
22             }
23         }
24     }
25
26     //버튼 배열 생성
27     public JButton[][] getBtn() {
28         if(btn == null) {
29             btn = new JButton[2][3];
30             for(int r = 0; r<2; r++) {
31                 for(int c=0; c<3; c++) {
32                     btn[r][c] = new JButton();
33                     btn[r][c].setText("[ "+r+" ][ " + c + " ]");
34                 }
35             }
36         }
37         return btn;
38     }
39
40     public static void main(String[] args) {
41         SwingUtilities.invokeLater(new Runnable() {
42             public void run() {
43                 GridLayoutExample jFrame = new GridLayoutExample();
44                 jFrame.setVisible(true);
45             }
46         });
47     }
48 }

```

CardLayout

CardLayout 배치 관리자는 이름에서도 알 수 있듯이 여러 장의 카드를 포개 놓고 한 번에 하나의 카드를 보여주는 역할을 한다. 이때 카드는 하나의 JPanel로 구성된다.



CardLayout이 적용된 컨테이너에 카드 하나를 추가할 때에는 카드의 이름과 JPanel을 추가할 수 있는 `add()` 메소드를 사용한다. 그리고 Card의 내용은 JPanel 안에 배치하면 된다.

```
jFrame.getContentPane().add("Card1", jPanel1);
jFrame.getContentPane().add("Card2", jPanel2);
```

여러 개의 카드를 추가하더라도 제일 먼저 추가한 카드만 보인다. 다른 카드는 아래에 겹쳐져 있어 볼 수 없는데, 이 카드를 보이게 하려면 CardLayout의 `first()`, `last()`, `next()`, `show()` 메소드를 호출하면 된다.

CardLayout 메소드	설명
first(Container container)	첫 번째 배치한 카드를 보이게 한다.
last(Container container)	마지막에 배치한 카드를 보이게 한다.
next(Container container)	현재 카드 다음에 배치한 카드를 보이게 한다.
show(Container container, String name)	지정된 이름의 카드를 보이게 한다.

메소드의 첫 번째 매개값인 Container는 CardLayout을 사용하는 컨테이너이고, show() 메소드의 두 번째 매개값은 보여줄 카드 이름이다. 다음 예제는 3개의 색깔 카드를 JFrame에 추가하고 1초 간격으로 카드를 변경해서 보여 준다.

>>> CardLayoutExample.java

```

1  package sec04.exam05_cardlayout;
2
3  import java.awt.CardLayout;
4  import java.awt.Color;
5  import javax.swing.JFrame;
6  import javax.swing.JPanel;
7  import javax.swing.SwingUtilities;
8
9  public class CardLayoutExample extends JFrame {
10     private JPanel redCard, greenCard, blueCard;
11
12     //메인 윈도우 설정
13     public CardLayoutExample() {
14         this.setTitle("CardLayoutExample");
15         this.setSize(250, 400);
16         this.setResizable(false);
17         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
18
19         //CardLayout으로 변경하고 3개의 카드 추가
20         this.getContentPane().setLayout(new CardLayout());
21         this.getContentPane().add("RedCard", getRedCard());
22         this.getContentPane().add("GreenCard", getGreenCard());
23         this.getContentPane().add("BlueCard", getBlueCard());
24     }
25

```

```

26     //RedCard에 해당하는 JPanel 생성
27     public JPanel getRedCard() {
28         if (redCard == null) {
29             redCard = new JPanel();
30             redCard.setBackground(Color.RED);
31         }
32         return redCard;
33     }
34
35     //GreenCard에 해당하는 JPanel 생성
36     public JPanel getGreenCard() {
37         if (greenCard == null) {
38             greenCard = new JPanel();
39             greenCard.setBackground(Color.GREEN);
40         }
41         return greenCard;
42     }
43
44     //BlueCard에 해당하는 JPanel 생성
45     public JPanel getBlueCard() {
46         if (blueCard == null) {
47             blueCard = new JPanel();
48             blueCard.setBackground(Color.BLUE);
49         }
50         return blueCard;
51     }
52
53     public static void main(String[] args) {
54         SwingUtilities.invokeLater(new Runnable() {
55             public void run() {
56                 final CardLayoutExample jFrame = new CardLayoutExample();
57                 jFrame.setVisible(true);
58                 //반복 스레드 생성
59                 Thread thread = new Thread() {
60                     @Override
61                     public void run() {
62                         for (int i = 0; i < 10; i++) {
63                             try {
64                                 //2초간 일시정지
65                                 Thread.sleep(1000);

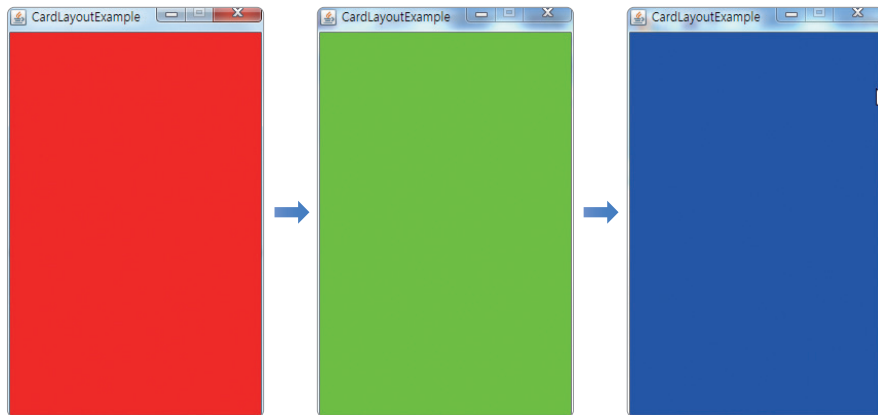
```

```

66         } catch (InterruptedException e) {
67         }
68         //이벤트 큐에 Runnable 객체 넣기
69         SwingUtilities.invokeLater(new Runnable() {
70             @Override
71             public void run() {
72                 //CardLayout을 얻어 다음 카드 보여주기
73                 CardLayout cardLayout =
74                     (CardLayout) jFrame.getContentPane().getLayout();
75                 cardLayout.next(jFrame.getContentPane());
76             }
77         });
78     }
79 };
80 //반복 스레드 시작
81 thread.start();
82 }
83 });
84 }
85 }

```

실행 결과



NullLayout

NullLayout은 컨테이너의 `setLayout()` 메소드에 배치 관리자 대신 매개값을 `null`로 설정한 것을 말한다. 이것은 어떠한 배치 관리자도 사용하지 않고 좌표값으로 컴포넌트를 배치함을 뜻한다.

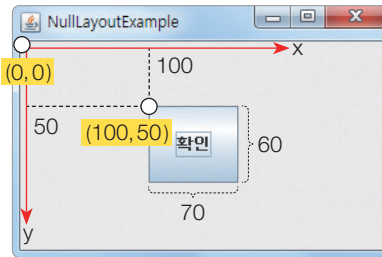
```
jFrame.getContentPane().setLayout(null);
```

컨테이너가 컴포넌트를 배치할 때 좌표값을 주는 것이 아니라, 컴포넌트쪽에서 컨테이너의 어떤 위치에 배치될 것인지 `setBounds()` 메소드로 좌표값을 설정해야 한다.

```
setBounds(int x, int y, int width, int height);
```

x, y 매개값은 픽셀 단위의 좌표값인데, 컨테이너의 좌측 상단이 (0, 0)이고 우측이 x축, 하단이 y축이다. x의 최대값은 컨테이너의 폭 `width`이고, y의 최대값은 컨테이너의 높이 `height`이다. `width` 매개값은 컴포넌트의 폭을 말하고, `height` 매개값은 컴포넌트의 높이를 말한다.

다음 예제는 JButton을 다음 그림과 같이 배치한다.



>>> NullLayoutExample.java

```
1 package sec04.exam06_nulllayout;
2
3 import javax.swing.JButton;
4 import javax.swing.JFrame;
5 import javax.swing.SwingUtilities;
6
7 public class NullLayoutExample extends JFrame {
8     private JButton btnOk;
9
10    //메인 윈도우 설정
11    public NullLayoutExample() {
12        this.setTitle("NullLayoutExample");
```

```

13     this.setSize(300, 200);
14     this.setResizable(false);
15     this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
16
17     //NullLayout 설정과 버튼 추가
18     this.getContentPane().setLayout(null);
19     this.getContentPane().add(getBtnOk());
20 }
21
22 //버튼 생성
23 public JButton getBtnOk() {
24     if(btnOk == null) {
25         btnOk = new JButton();
26         btnOk.setText("확인");
27         //버튼이 위치할 좌표값과 폭과 높이 설정
28         btnOk.setBounds(100, 50, 70, 60);
29     }
30     return btnOk;
31 }
32
33 public static void main(String[] args) {
34     SwingUtilities.invokeLater(new Runnable() {
35         public void run() {
36             NullLayoutExample jFrame = new NullLayoutExample();
37             jFrame.setVisible(true);
38         }
39     });
40 }
41 }

```

Pack

JWindow, JFrame, JDialog와 같이 java.awt.Window를 상속받는 최상위 레벨 컨테이너는 pack()이라는 메소드를 사용해서 내부의 컴포넌트의 크기에 맞게 컨테이너의 크기를 자동으로 조절할 수 있다.

컴포넌트에는 PreferredSize라는 속성이 있는데, 이것은 컴포넌트의 기본 배치 크기를 말한다. 컨

테이너의 `pack()` 메소드가 호출되면 내부 컴포넌트의 `getPreferredSize()`를 호출해서 컴포넌트의 기본 배치 크기를 알아낸 뒤, 컨테이너의 크기를 계산한다.

컨테이너의 `setSize()` 메소드는 직접 컨테이너의 폭과 높이를 설정하지만, `pack()` 메소드는 내부 컴포넌트의 크기에 따라 컨테이너의 크기가 결정된다. 따라서 `pack()` 메소드를 호출하는 시점은 컨테이너에 컴포넌트들이 모두 배치가 끝난 시점이어야 한다.

다음 예제는 `JFrame`에 두 개의 `JButton`을 추가하고 `pack()` 메소드를 호출했다. `JFrame`의 크기는 두 버튼의 크기에 최대한 맞추게 된다.

>>> `PackExample.java`

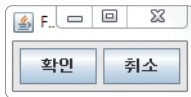
```
1  package sec04.exam07_pack;
2
3  import java.awt.FlowLayout;
4  import javax.swing.JButton;
5  import javax.swing.JFrame;
6  import javax.swing.SwingUtilities;
7
8  public class PackExample extends JFrame {
9      private JButton btnOk;
10     private JButton btnCancel;
11
12     //메인 윈도우 설정
13     public PackExample() {
14         this.setTitle("FlowLayoutExample");
15         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
16
17         //FlowLayout으로 변경하고 버튼 추가
18         this.setLayout(new FlowLayout());
19         this.getContentPane().add(getBtnOk());
20         this.getContentPane().add(getBtnCancel());
21
22         //pack() 메소드 호출
23         this.pack();
24     }
25
26     //Ok 버튼 생성
27     private JButton getBtnOk() {
```

```

28     if(btnOk == null) {
29         btnOk = new JButton();
30         btnOk.setText("확인");
31     }
32     return btnOk;
33 }
34
35 //Cancel 버튼 생성
36 private JButton getBtnCancel() {
37     if(btnCancel == null) {
38         btnCancel = new JButton();
39         btnCancel.setText("취소");
40     }
41     return btnCancel;
42 }
43
44 public static void main(String[] args) {
45     SwingUtilities.invokeLater(new Runnable() {
46         public void run() {
47             PackExample jFrame = new PackExample();
48             jFrame.setVisible(true);
49         }
50     });
51 }
52 }

```

실행 결과

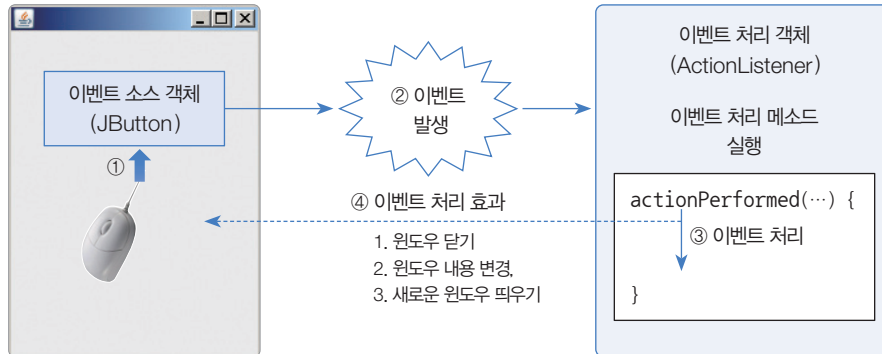


05 이벤트 처리

UI 프로그램은 사용자와 상호작용을 하면서 코드를 실행한다. 사용자가 UI의 컴포넌트를 사용하는 순간 이벤트(event)가 발생하고, 프로그램은 이벤트를 처리하기 위해 코드를 실행한다.

자바는 이벤트 소스 객체(컨테이너, 컴포넌트)와 이벤트 처리 객체(리스너`Listener`)를 분리하는 위임형`Delegation` 방식을 사용한다. 위임형 방식이란 이벤트 소스에서 이벤트가 발생하면 직접 처리하지 않고 이벤트 소스에 추가된 리스너에게 이벤트를 위임하는 방식이다.

예를 들어 사용자가 `JButton`(이벤트 소스 객체)를 클릭하면 액션 이벤트`ActionEvent`가 발생하고, `JButton`에 추가된 `ActionListener` 객체(이벤트 처리 객체)가 액션 이벤트를 처리한다.



이벤트 처리 객체인 리스너는 컴포넌트에서 이벤트가 발생하면 이벤트 처리 메소드를 실행시킨다. 이벤트 처리 메소드는 현재 윈도우를 닫거나, 윈도우 내용을 변경하거나, 새로운 윈도우 또는 다이얼로그를 띄우기도 한다.

컴포넌트는 하나의 이벤트만 발생하는 것이 아니라 동시에 여러 개의 이벤트가 발생하기도 한다. 예를 들어 `JButton`을 마우스로 클릭하면 액션 이벤트`ActionEvent`와 함께 마우스 이벤트`MouseEvent`도 발생한다. 액션 이벤트는 마우스로 클릭하거나 `Enter` 키를 눌러 사용하는 컴포넌트에서 주로 발생하고, 마우스 이벤트는 대부분의 컨테이너 또는 컴포넌트에서 발생한다.

컴포넌트에서 발생하는 모든 이벤트를 처리하기 위해서는 이벤트별로 리스너가 추가되어야 한다. `JButton`에서 발생하는 액션 이벤트와 마우스 이벤트를 동시에 처리하기 위해서는 액션 리스너와 마우스 리스너가 모두 필요하다.

하지만, 동시에 발생하는 이벤트가 많다고 하더라도 모두 처리할 필요가 없다. 처리하고 싶은 관심 이벤트에 대해서만 리스너를 추가하면 된다. 다음 표는 컨테이너 및 컴포넌트에서 발생할 수 있는 대표적인 이벤트와 이벤트 처리 리스너를 정리한 표이다.

이벤트 소스	발생 이벤트	발생 원인	리스너
JFrame	WindowEvent	 중 하나를 클릭했을 때	WindowListener
JDialog	WindowEvent	 중 하나를 클릭했을 때	WindowListener
JTextField	ActionEvent	 키를 눌렀을 때	ActionListener
JButton	ActionEvent	클릭했을 때	ActionListener
JRadioButton	ActionEvent	클릭했을 때	ActionListener
JCheckBox	ActionEvent	클릭했을 때	ActionListener
JMenuItem	ActionEvent	선택했을 때	ActionListener
JComboBox	ActionEvent	다른 항목을 선택했을 때	ActionListener
JList	ListSelectionEvent	다른 항목을 선택했을 때	ListSelectionListener

이벤트 소스에 리스너를 추가하려면 addXXXListener() 메소드를 사용한다. XXX는 이벤트명인데, 예를 들어 WindowEvent, ActionEvent, ListSelectionEvent를 처리하기 위한 리스너를 추가하는 메소드는 다음과 같다.

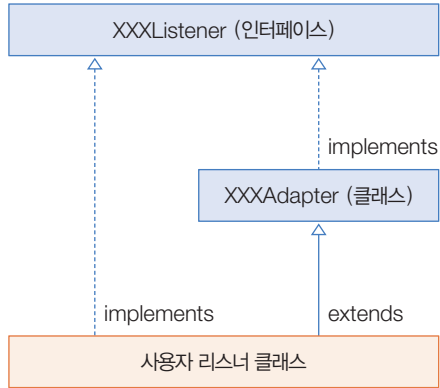
```
jFrame.addWindowListener(WindowListener listener);
jButton.addActionListener(ActionListener listener);
jList.addListSelectionListener(ListSelectionListener listener);
```

자바 API 도큐먼트를 보면 컴포넌트에서 어떤 이벤트들이 발생하는지 알 수 있다. 컴포넌트의 메소드 목록을 보면 addXXXListener() 메소드를 많이 볼 수 있다. 이것은 해당 컴포넌트가 XXXEvent가 발생할 수 있으니 XXXListener를 등록할 수 있다는 말이다.

대부분의 컴포넌트들은 java.awt.Component를 상속하는데, Component 클래스는 addKeyListener()와 addMouseListener() 메소드를 가지고 있다. 따라서 KeyEvent와 MouseEvent가 발생할 수 있고, 이들을 처리하기 위해 KeyListener와 MouseListener를 등록할 수 있다.

Listener와 Adapter

컴포넌트에 리스너를 추가하기 위해서는 리스너 클래스를 먼저 작성해야 한다. 리스너 클래스를 생성하는 방법은 리스너 인터페이스를 구현하는 방법과 어댑터 클래스를 상속하는 방법이 있다.



리스너 인터페이스를 구현하는 방법

리스너 인터페이스를 구현하려면 리스너 인터페이스에 정의되어 있는 이벤트 처리 메소드를 모두 재정의해야 한다.

예를 들어 WindowListener에는 windowClosing() 메소드를 포함하여 7개의 메소드가 정의되어 있다. 이 메소드들은 WindowEvent가 발생했을 때 사용자의 행위에 따라 개별적으로 실행된다. 윈도우 상단 우측 닫기(×) 버튼을 클릭하면 windowClosing() 메소드가 실행되고, 최소화(_) 버튼을 클릭하면 windowIconified() 메소드가 실행된다. 닫기(×) 버튼에서 발생하는 WindowEvent만 처리하고 싶어도 windowClosing() 및 나머지 6개를 다음과 같이 모두 재정의해야 한다.

```

class MyWindowListener implements WindowListener {
    public void windowActivated(WindowEvent e) {}
    public void windowClosed(WindowEvent e) {}
    public void windowClosing(WindowEvent e) {
        //닫기(×) 버튼을 클릭했을 때 처리 방법 코딩
    }
    public void windowDeactivated(WindowEvent e) {}
    public void windowDeiconified(WindowEvent e) {}
    public void windowIconified(WindowEvent e) {}
    public void windowOpened(WindowEvent e) {}
}
  
```

리스너 어댑터를 상속하는 방법

리스너 어댑터를 상속하면 관심 있는 이벤트 처리 메소드만 재정의할 수 있기 때문에 리스너 인터페이스를 구현하는 방법보다 좀 더 효율적이다.

예를 들어 WindowAdapter 클래스를 상속할 경우 windowClosing() 메소드만 재정의하면 된다. 나머지 6개의 메소드는 WindowAdapter에서 내용이 없는 채로 이미 구현되어 있기 때문이다.

```
class MyWindowListener extends WindowAdapter {  
    public void windowClosing(WindowEvent e) {  
        //닫기(x) 버튼을 클릭했을 때 처리 방법 코딩  
    }  
}
```

리스너 인터페이스에 대응되는 어댑터 클래스가 모두 존재하는 것은 아니다. 리스너 인터페이스에 2개 이상의 이벤트 처리 메소드가 정의되어 있을 경우에만 어댑터 클래스가 제공된다. ActionEvent를 처리하는 ActionListener일 경우 actionPerformed() 메소드 한 개만 정의되어 있기 때문에 ActionListener는 제공되지 않는다. 다음 표는 리스너 인터페이스와 대응되는 어댑터 클래스를 나타낸 것이다.

리스너 인터페이스	어댑터
java.awt.event.WindowListener	java.awt.event.WindowAdapter
java.awt.event.MouseListener	java.awt.event.MouseAdapter
java.awt.event.KeyListener	java.awt.event.KeyAdapter
java.awt.event.ActionListener	없음
javax.swing.event.ListSelectionListener	없음

다음 예제는 JFrame의 제목 표시줄의 닫기(×) 버튼과 하단에 있는 btnClose 버튼 중 하나를 클릭하면 프로그램이 종료되도록 한다. 제목 표시줄의 닫기 버튼은 WindowAdapter를, btnClose는 ActionListener를 이용해서 리스너 클래스를 작성하였다.

>>> ClosableExample1.java

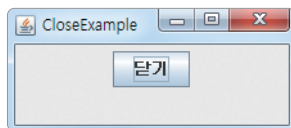
```
1  package sec05.exam01_windowadpater;
2
3  import java.awt.FlowLayout;
4  import java.awt.event.ActionEvent;
5  import java.awt.event.ActionListener;
6  import java.awt.event.WindowAdapter;
7  import java.awt.event.WindowEvent;
8  import javax.swing.JButton;
9  import javax.swing.JFrame;
10 import javax.swing.SwingUtilities;
11
12 public class ClosableExample1 extends JFrame {
13     private JButton btnClose;
14
15     //메인 윈도우 설정
16     public ClosableExample1() {
17         this.setTitle("CloseExample");
18         this.setSize(300, 100);
19
20         this.setLayout(new FlowLayout());
21         this.getContentPane().add(getBtnClose());
22
23         //WindowListener 추가
24         this.addWindowListener(new MyWindowAdapter());
25     }
26
27     //닫기 버튼 생성
28     private JButton getBtnClose() {
29         if(btnClose == null) {
30             btnClose = new JButton();
31             btnClose.setText("닫기");
32
33             //ActionListener 추가
34             btnClose.addActionListener(new MyActionListener());
35         }
36         return btnClose;
37     }
38
39     public static void main(String[] args) {
```

```

40     SwingUtilities.invokeLater(new Runnable() {
41         public void run() {
42             ClosableExample1 jFrame = new ClosableExample1();
43             jFrame.setVisible(true);
44         }
45     });
46 }
47 }
48
49 //WindowAdapter 클래스를 상속해서 WindowListener 클래스 작성
50 class MyWindowAdapter extends WindowAdapter {
51     @Override
52     public void windowClosing(WindowEvent e) {
53         System.exit(0);
54     }
55 }
56
57 //ActionListener를 구현해서 ActionListener 클래스 작성
58 class MyActionListener implements ActionListener {
59     @Override
60     public void actionPerformed(ActionEvent e) {
61         System.exit(0);
62     }
63 }

```

실행 결과



Anonymous Listener

이벤트를 처리할 때 리스너 클래스를 외부 클래스로 선언하게 되면 컨테이너의 필드와 메소드에 접근하는 것이 불편하다. 그래서 리스너는 일반적으로 익명Anonymous 객체로 작성한다. 다음 예제는 이전 예제를 수정한 것으로, 익명 객체를 사용해서 이벤트를 처리한다.

>>> ClosableExample2.java

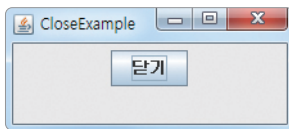
```
1  package sec05.exam01_windowadpater;
2
3  import java.awt.FlowLayout;
4  import java.awt.event.ActionEvent;
5  import java.awt.event.ActionListener;
6  import java.awt.event.WindowAdapter;
7  import java.awt.event.WindowEvent;
8  import javax.swing.JButton;
9  import javax.swing.JFrame;
10 import javax.swing.SwingUtilities;
11
12 public class ClosableExample2 extends JFrame {
13     private JButton btnClose;
14
15     //메인 윈도우 설정
16     public ClosableExample2() {
17         this.setTitle("CloseExample");
18         this.setSize(300, 100);
19
20         this.setLayout(new FlowLayout());
21         this.getContentPane().add(getBtnClose());
22
23         //익명 WindowListener 객체 추가
24         this.addWindowListener(new WindowAdapter() {
25             @Override
26             public void windowClosing(WindowEvent e) {
27                 System.exit(0);
28             }
29         });
30     }
31
32     private JButton getBtnClose() {
33         if(btnClose == null) {
34             btnClose = new JButton();
35             btnClose.setText("닫기");
36
37             //익명 ActionListener 객체 추가
```

```

38         btnClose.addActionListener(new ActionListener() {
39             @Override
40             public void actionPerformed(ActionEvent e) {
41                 System.exit(0);
42             }
43         });
44     }
45     return btnClose;
46 }
47
48 public static void main(String[] args) {
49     SwingUtilities.invokeLater(new Runnable() {
50         public void run() {
51             ClosableExample2 jFrame = new ClosableExample2();
52             jFrame.setVisible(true);
53         }
54     });
55 }
56 }

```

실행 결과



만약 복수 개의 컴포넌트에서 동일한 리스너를 사용해서 이벤트를 처리하고 싶다면 리스너를 필드로 선언한다. 이 경우에는 리스너에서 어떤 컴포넌트에서 이벤트가 발생되었는지 코드로 구분해야 한다.

다음 예제는 두 개의 JButton에서 발생하는 ActionEvent를 하나의 ActionListener 익명 객체로 처리하는 방법을 보여 준다. 어떤 컴포넌트에서 ActionEvent가 발생되었는지 구분하기 위해 ActionEvent의 getSource() 메소드를 이용하였다.

>>> ActionListenerExample.java

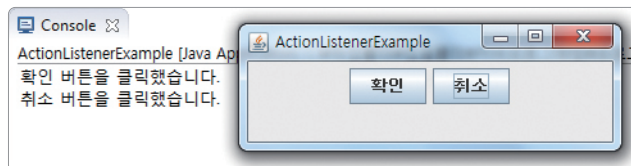
```
1  package sec05.exam02_actionlistener;
2
3  import java.awt.FlowLayout;
4  import java.awt.event.ActionEvent;
5  import java.awt.event.ActionListener;
6  import javax.swing.JButton;
7  import javax.swing.JFrame;
8  import javax.swing.SwingUtilities;
9
10 public class ActionListenerExample extends JFrame {
11     private JButton btnOk;
12     private JButton btnCancel;
13
14     //메인 윈도우 설정
15     public ActionListenerExample() {
16         this.setTitle("ActionListenerExample");
17         this.setSize(300, 100);
18         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
19
20         this.setLayout(new FlowLayout());
21         this.getContentPane().add(getBtnOk());
22         this.getContentPane().add(getBtnCancel());
23     }
24
25     //ActionListener 타입의 필드 선언 및 익명 객체로 초기화
26     private ActionListener actionListener = new ActionListener() {
27         @Override
28         public void actionPerformed(ActionEvent e) {
29             //ActionEvent가 발생한 컴포넌트 구분
30             if(e.getSource() == btnOk) {
31                 System.out.println("확인 버튼을 클릭했습니다.");
32             } else if(e.getSource() == btnCancel) {
33                 System.out.println("취소 버튼을 클릭했습니다.");
34             }
35         }
36     };
37
38     //Ok 버튼 생성
```

```

39     private JButton getBtnOk() {
40         if(btnOk == null) {
41             btnOk = new JButton();
42             btnOk.setText("확인");
43             //actionListener 필드 대입
44             btnOk.addActionListener(actionListener);
45         }
46         return btnOk;
47     }
48
49     //Cancel 버튼 생성
50     private JButton getBtnCancel() {
51         if(btnCancel == null) {
52             btnCancel = new JButton();
53             btnCancel.setText("취소");
54             //actionListener 필드 대입
55             btnCancel.addActionListener(actionListener);
56         }
57         return btnCancel;
58     }
59
60     public static void main(String[] args) {
61         SwingUtilities.invokeLater(new Runnable() {
62             public void run() {
63                 ActionListenerExample jFrame = new ActionListenerExample();
64                 jFrame.setVisible(true);
65             }
66         });
67     }
68 }

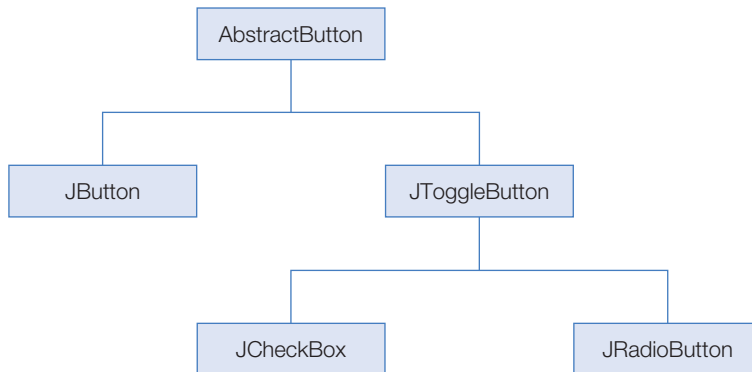
```

실행 결과



06 버튼 컴포넌트

버튼 컴포넌트는 `AbstractButton`을 상속받은 하위 클래스들을 말한다. 버튼 컴포넌트에는 `JButton`, `JToggleButton`, `JRadioButton`, `JCheckBox`가 있는데, 모두 사용자가 마우스로 클릭하여 사용할 수 있도록 되어 있다. 다음은 버튼 컴포넌트의 상속 관계를 보여 준다.



버튼 컴포넌트를 마우스로 클릭하면 모두 `ActionEvent`가 발생한다. 그래서 `addActionListener()` 메소드로 `ActionListener` 객체를 등록하여 이벤트를 처리할 수 있다.

JButton

`JButton`은 이미지와 텍스트로 구성된 일반적인 버튼을 만들 때 사용한다. `JButton`의 `setText()` 메소드는 버튼의 텍스트를 설정하고, `setIcon()` 메소드는 버튼의 이미지를 설정한다.

```
JButton jButton = new JButton();
jButton.setText("새문서");
jButton.setIcon( new ImageIcon( getClass().getResource("new.gif") ) );
```

다음 예제는 텍스트, 이미지, 텍스트+이미지 버튼을 생성한다. 버튼의 이벤트 처리는 파일 열기 대화상자를 보여주도록 했다. 이 대화상자를 활용하는 방법은 13절에서 설명한다.

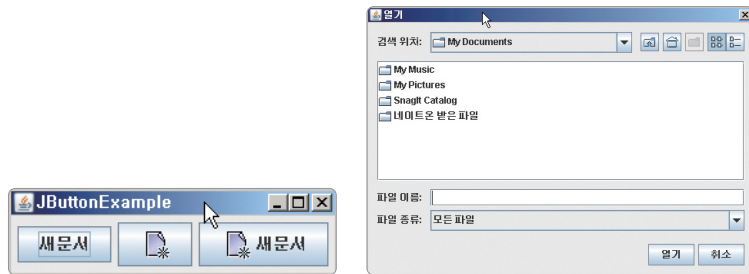
» JButtonExample.java

```
1  package sec06.exam01_jbutton;
2
3  import java.awt.FlowLayout;
4  import java.awt.event.ActionEvent;
5  import java.awt.event.ActionListener;
6  import javax.swing.ImageIcon;
7  import javax.swing.JButton;
8  import javax.swing.JFileChooser;
9  import javax.swing.JFrame;
10 import javax.swing.SwingUtilities;
11
12 public class JButtonExample extends JFrame {
13     private JButton btn1, btn2, btn3;
14
15     //메인 윈도우 설정
16     public JButtonExample() {
17         this.setTitle("JButtonExample");
18         this.setSize(300, 100);
19         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
20         this.getContentPane().setLayout(new FlowLayout());
21         this.getContentPane().add(getBtn1());
22         this.getContentPane().add(getBtn2());
23         this.getContentPane().add(getBtn3());
24     }
25
26     //글자만 있는 버튼 생성
27     public JButton getBtn1() {
28         if(btn1 == null) {
29             btn1 = new JButton();
30             btn1.setText("새문서");
31             btn1.addActionListener(new ActionListener() {
32                 public void actionPerformed(ActionEvent e) {
33                     JFileChooser jFileChooser = new JFileChooser();
34                     jFileChooser.showOpenDialog(JButtonExample.this);
35                 }
36             });
37         }
38         return btn1;
39     }
```

```

40
41 //아이콘만 있는 버튼 생성
42 public JButton getBtn2() {
43     if(btn2 == null) {
44         btn2 = new JButton();
45         btn2.setIcon(new ImageIcon(getClass().getResource("new.gif")));
46         btn2.addActionListener(new ActionListener() {
47             public void actionPerformed(ActionEvent e) {
48                 JFileChooser jFileChooser = new JFileChooser();
49                 jFileChooser.showOpenDialog(JButtonExample.this);
50             }
51         });
52     }
53     return btn2;
54 }
55
56 //아이콘과 글자가 있는 버튼 생성
57 public JButton getBtn3() {
58     if(btn3 == null) {
59         btn3 = new JButton();
60         btn3.setText("새문서");
61         btn3.setIcon(new ImageIcon(getClass().getResource("new.gif")));
62         btn3.addActionListener(new ActionListener() {
63             public void actionPerformed(ActionEvent e) {
64                 JFileChooser jFileChooser = new JFileChooser();
65                 jFileChooser.showOpenDialog(JButtonExample.this);
66             }
67         });
68     }
69     return btn3;
70 }
71
72 public static void main(String[] args) {
73     SwingUtilities.invokeLater(new Runnable() {
74         public void run() {
75             JButtonExample jFrame = new JButtonExample();
76             jFrame.setVisible(true);
77         }
78     });
79 }
80 }

```



JToggleButton

JToggleButton은 선택된 상태와 그렇지 않은 두 가지 상태를 가지는 버튼이다. 생성 방법은 JButton과 유사해서 텍스트와 이미지를 설정할 수 있다.

```
JToggleButton jToggleButton = new JToggleButton();
jToggleButton.setText("확인");
jToggleButton.setIcon(new ImageIcon( getClass().getResource("ok.gif") ));
```

JToggleButton은 선택된 상태를 가지는 버튼이기 때문에 ActionListener보다는 다음과 같이 ItemListener로 이벤트를 처리하는 것이 좋다. ItemEvent의 getStateChange() 메소드는 JToggleButton이 선택되었을 경우 ItemEvent.SELECTED 상수값을 리턴한다.

```
jToggleButton.addItemListener(new ItemListener() {
    @Override
    public void itemStateChanged(ItemEvent e) {
        if(e.getStateChange() == ItemEvent.SELECTED) {
            //선택된 상태
        } else {
            //해제된 상태
        }
    }
});
```

JToggleButton은 단독으로도 사용 가능하지만, JToggleButton 두 개를 ButtonGroup에 포함 시키면 두 버튼을 배타적으로 선택할 수 있다. ButtonGroup 내부에서는 하나의 버튼만이 선택된 상태로 존재할 수 있기 때문이다.

```
ButtonGroup buttonGroup = new ButtonGroup();
buttonGroup.add( jToggleButton1 );
buttonGroup.add( jToggleButton2 );
```

다음 예제는 [On] [Off] 토글 버튼과, 배타적으로 선택할 수 있는 [Start]와 [Stop] 버튼을 생성한다. 그리고 [Start]와 [Stop] 버튼을 클릭하면 메시지 다이얼로그를 보여 준다. 다이얼로그에 대한 내용은 13절에서 자세히 설명한다.

>>> JToggleButtonExample.java

```
1  package sec06.exam02_jtogglebutton;
2
3  import java.awt.GridLayout;
4  import java.awt.event.ActionEvent;
5  import java.awt.event.ActionListener;
6  import java.awt.event.ItemEvent;
7  import java.awt.event.ItemListener;
8  import javax.swing.ButtonGroup;
9  import javax.swing.ImageIcon;
10 import javax.swing.JFrame;
11 import javax.swing.JOptionPane;
12 import javax.swing.JPanel;
13 import javax.swing.JToggleButton;
14 import javax.swing.SwingUtilities;
15 import javax.swing.border.TitledBorder;
16
17 public class JToggleButtonExample extends JFrame {
18     private JPanel pFirst;
19     private JPanel pSecond;
20     private JToggleButton tbOnOff;
21     private JToggleButton tbStart;
22     private JToggleButton tbStop;
23 }
```

```

24     //메인 윈도우 설정
25     public JToggleButtonExample() {
26         this.setTitle("JToggleButtonExample");
27         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
28         this.getContentPane().setLayout(new GridLayout(2, 1));
29         this.getContentPane().add(getPFirst());
30         this.getContentPane().add(getPSecond());
31         this.pack();
32     }
33
34     public JPanel getPFirst() {
35         if(pFirst == null) {
36             pFirst = new JPanel();
37             pFirst.add(getTbOnOff());
38         }
39         return pFirst;
40     }
41
42     public JPanel getPSecond() {
43         if(pSecond == null) {
44             pSecond = new JPanel();
45             pSecond.setBorder(new TitledBorder("원하는 기능은?"));
46             pSecond.add(getTbStart());
47             pSecond.add(getTbStop());
48
49             //배타적 선택을 위한 ButtonGroup 생성 및 토글 버튼 추가
50             ButtonGroup buttonGroup = new ButtonGroup();
51             buttonGroup.add(getTbStart());
52             buttonGroup.add(getTbStop());
53         }
54         return pSecond;
55     }
56
57     //On/Off 토글 버튼 생성
58     public JToggleButton getTbOnOff() {
59         if(tbOnOff == null) {
60             tbOnOff = new JToggleButton();
61             tbOnOff.setText("On");
62             tbOnOff.addItemListener(new ItemListener() {
63                 @Override

```



```

64         public void itemStateChanged(ItemEvent e) {
65             if(e.getStateChange() == ItemEvent.SELECTED) {
66                 getTbOnOff().setText("Off");
67             } else {
68                 getTbOnOff().setText("On");
69             }
70         }
71     });
72 }
73     return tbOnOff;
74 }
75
76 //Start 토글 버튼 생성
77 public JToggleButton getTbStart() {
78     if(tbStart == null) {
79         tbStart = new JToggleButton();
80         tbStart.setText("Start");
81         tbStart.setIcon(new ImageIcon(getClass().getResource("start.gif")));
82         tbStart.addActionListener(new ActionListener() {
83             public void actionPerformed(ActionEvent e) {
84                 JOptionPane.showMessageDialog(JToggleButtonExample.this, "Start");
85             }
86         });
87     }
88     return tbStart;
89 }
90
91 //Stop 토글 버튼 생성
92 public JToggleButton getTbStop() {
93     if(tbStop == null) {
94         tbStop = new JToggleButton();
95         tbStop.setText("Stop");
96         tbStop.setIcon(new ImageIcon(getClass().getResource("stop.gif")));
97         tbStop.addActionListener(new ActionListener() {
98             public void actionPerformed(ActionEvent e) {
99                 JOptionPane.showMessageDialog(JToggleButtonExample.this, "Stop");
100             }
101         });
102     }
103     return tbStop;

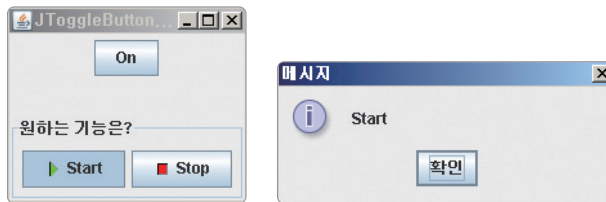
```

```

104     }
105
106     public static void main(String[] args) {
107         SwingUtilities.invokeLater(new Runnable() {
108             public void run() {
109                 JToggleButtonExample jFrame = new JToggleButtonExample();
110                 jFrame.setVisible(true);
111             }
112         });
113     }
114 }

```

실행 결과



JRadioButton

JRadioButton은 JToggleButton의 하위 클래스로, 둥근 모양의 선택과 텍스트를 함께 보여주는 버튼이다. JRadioButton은 동일한 ButtonGroup에 포함되어 한 번에 하나의 JRadioButton만 선택된 상태를 가진다. 다음 코드는 남자와 여자 둘 중 하나를 선택할 수 있다.

```

JRadioButton jRadioButton1 = new JRadioButton();
jRadioButton1.setText("남자");

JRadioButton jRadioButton2 = new JRadioButton();
jRadioButton2.setText("여자");

ButtonGroup buttonGroup = new ButtonGroup();
buttonGroup.add(jRadioButton1);
buttonGroup.add(jRadioButton2);

```

JRadioButton은 마우스로 클릭했을 때 `ActionEvent`가 발생하므로 `ActionListener`로 이벤트를 처리할 수 있다. 다음 예제는 두 개의 JRadioButton 중에 하나를 선택하면 JLabel의 이미지가 변경되도록 하였다.

>> JRadioButtonExample.java

```
1  package sec06.exam03_jradiobutton;
2
3  import java.awt.BorderLayout;
4  import java.awt.GridLayout;
5  import java.awt.event.ActionEvent;
6  import java.awt.event.ActionListener;
7  import javax.swing.ButtonGroup;
8  import javax.swing.ImageIcon;
9  import javax.swing.JFrame;
10 import javax.swing.JLabel;
11 import javax.swing.JPanel;
12 import javax.swing.JRadioButton;
13 import javax.swing.SwingUtilities;
14
15 public class JRadioButtonExample extends JFrame {
16     private JPanel radioPanel;
17     private JRadioButton rbBird;
18     private JRadioButton rbCat;
19     private JLabel lblPicture;
20
21     //메인 윈도우 설정
22     public JRadioButtonExample() {
23         setTitle("JRadioButtonExample");
24         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
25         this.getContentPane().add(getRadioPanel(), BorderLayout.WEST);
26         this.getContentPane().add(getLblPicture(), BorderLayout.CENTER);
27         pack();
28     }
29
30     //JRadioButton이 배치된 JPanel 생성
31     public JPanel getRadioPanel() {
32         if (radioPanel == null) {
33             radioPanel = new JPanel();
```

```

34
35     radioPanel.setLayout(new GridLayout(2, 1));
36     radioPanel.add(getRbBird());
37     radioPanel.add(getRbCat());
38
39     //배타적 선택을 위해 ButtonGroup에 두 개의 JRadioButton 추가
40     ButtonGroup group = new ButtonGroup();
41     group.add(getRbBird());
42     group.add(getRbCat());
43 }
44 return radioPanel;
45 }
46
47 //JRadioButton 생성
48 public JRadioButton getRbBird() {
49     if (rbBird == null) {
50         rbBird = new JRadioButton();
51         rbBird.setText("Bird");
52         rbBird.setSelected(true); //기본적으로 선택되도록 설정
53         rbBird.addActionListener(new ActionListener() {
54             public void actionPerformed(ActionEvent e) {
55                 getLblPicture().setIcon(new ImageIcon(getClass().getResource(
56                     "Bird.gif")));
57             }
58         });
59     }
60     return rbBird;
61 }
62
63 //JRadioButton 생성
64 public JRadioButton getRbCat() {
65     if (rbCat == null) {
66         rbCat = new JRadioButton();
67         rbCat.setText("Cat");
68         rbCat.addActionListener(new ActionListener() {
69             public void actionPerformed(ActionEvent e) {
70                 getLblPicture().setIcon(new ImageIcon(getClass().getResource(
71                     "Cat.gif")));
72             }
73         });
74     }
75     return rbCat;
76 }

```

```

72     }
73     return rbCat;
74 }
75
76 //이미지를 보여줄 JLabel 생성
77 public JLabel getLblPicture() {
78     if (lblPicture == null) {
79         lblPicture = new JLabel();
80         lblPicture.setIcon(new ImageIcon(getClass().getResource("Bird.gif")));
81     }
82     return lblPicture;
83 }
84
85 public static void main(String[] args) {
86     SwingUtilities.invokeLater(new Runnable() {
87         public void run() {
88             JRadioButtonExample jFrame = new JRadioButtonExample();
89             jFrame.setVisible(true);
90         }
91     });
92 }
93 }

```

실행 결과



JCheckBox

JCheckBox는 사각형의 체크박스와 텍스트를 함께 보여주는 컴포넌트이다. 이 컴포넌트도 JToggleButton을 상속하며 체크와 언체크의 두 가지 상태를 갖는다. JRadioButton과 차이점은 개별적으로 선택할 수 있다는 것이다. 다음 코드는 축구와 농구를 개별 선택할 수 있도록 한다.

```
JCheckBox jCheckBox1 = new JCheckBox();
jCheckBox1.setText("축구");
```

```
JCheckBox jCheckBox2 = new JCheckBox();
jCheckBox2.setText("농구");
```

JCheckBox도 마우스로 클릭했을 때 `ActionEvent`가 발생하므로 `ActionListener`로 이벤트를 처리할 수 있다. JCheckBox가 선택되었는지 확인하는 방법은 `isSelected()` 메소드의 리턴값을 확인하면 된다. `true`가 리턴되면 선택이 된 것이다.

다음 예제는 JCheckBox 체크 상태에 따라 JLabel의 이미지를 변경한다.

» JCheckBoxExample.java

```
1  package sec06.exam04_jcheckbox;
2
3  import java.awt.BorderLayout;
4  import java.awt.GridLayout;
5  import java.awt.event.ActionEvent;
6  import java.awt.event.ActionListener;
7  import javax.swing.ImageIcon;
8  import javax.swing.JCheckBox;
9  import javax.swing.JFrame;
10 import javax.swing.JLabel;
11 import javax.swing.JPanel;
12 import javax.swing.SwingUtilities;
13
14 public class JCheckBoxExample extends JFrame {
15     private JPanel pWest;
16     private JCheckBox cbGlasses;
17     private JCheckBox cbHair;
18     private JLabel lblPicture;
19
20     //메인 윈도우 설정
21     public JCheckBoxExample() {
22         this.setTitle("JCheckBoxExample");
23         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
24         this.getContentPane().add(getPWest(), BorderLayout.WEST);
```

```

25     this.getContentPane().add(getLblPicture(), BorderLayout.CENTER);
26     this.pack();
27 }
28
29 //서쪽에 부착할 JPanel 생성
30 public JPanel getPWest() {
31     if (pWest == null) {
32         pWest = new JPanel(new GridLayout(2, 1));
33         // JCheckBox 추가
34         pWest.add(getCbGlasses());
35         pWest.add(getCbHair());
36     }
37     return pWest;
38 }
39
40 //JCheckBox 생성
41 public JCheckBox getCbGlasses() {
42     if (cbGlasses == null) {
43         cbGlasses = new JCheckBox();
44         cbGlasses.setText("Glasses");
45         cbGlasses.addActionListener(actionListener);
46     }
47     return cbGlasses;
48 }
49
50 //JCheckBox 생성
51 public JCheckBox getCbHair() {
52     if (cbHair == null) {
53         cbHair = new JCheckBox();
54         cbHair.setText("Hair");
55         cbHair.addActionListener(actionListener);
56     }
57     return cbHair;
58 }
59
60 //이미지를 보여줄 JLabel 생성
61 public JLabel getLblPicture() {
62     if (lblPicture == null) {
63         lblPicture = new JLabel();
64         lblPicture.setIcon(new ImageIcon(getClass().getResource("geek.gif")));

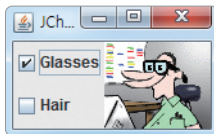
```

```

65     }
66     return lblPicture;
67 }
68
69 //JCheckBox 이벤트 처리 리스너를 위한 필드 선언
70 private ActionListener actionListener = new ActionListener() {
71     @Override
72     public void actionPerformed(ActionEvent e) {
73         if (cbGlasses.isSelected() && cbHair.isSelected()) {
74             lblPicture.setIcon(new ImageIcon(getClass().getResource
75                 ("geek-glasses-hair.gif")));
76         } else if (cbGlasses.isSelected()) {
77             lblPicture.setIcon(new ImageIcon(getClass().getResource
78                 ("geek-glasses.gif")));
79         } else if (cbHair.isSelected()) {
80             lblPicture.setIcon(new ImageIcon(getClass().getResource
81                 ("geek-hair.gif")));
82         } else {
83             lblPicture.setIcon(new ImageIcon(getClass().getResource("geek.gif")));
84         }
85     }
86 };
87
88 public static void main(String[] args) {
89     SwingUtilities.invokeLater(new Runnable() {
90         public void run() {
91             JCheckBoxExample jFrame = new JCheckBoxExample();
92             jFrame.setVisible(true);
93         }
94     });
95 }

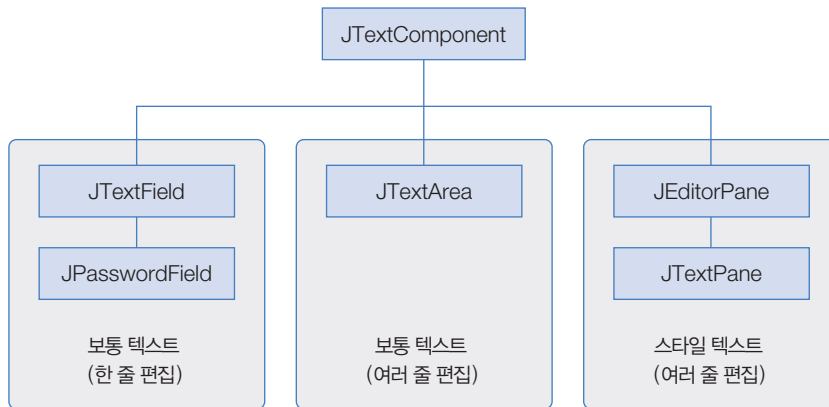
```

실행 결과



07 텍스트 컴포넌트

텍스트 컴포넌트는 텍스트를 나타내거나 편집할 수 있는 컴포넌트를 말한다. 텍스트 컴포넌트에는 JLabel, JTextField, JPasswordField, JTextArea, JEditorPane, JTextPane 등이 있다. 이 중에서 JLabel만 텍스트를 편집할 수 없고, 나머지는 텍스트를 편집할 수 있다. 편집 가능한 텍스트 컴포넌트는 모두 JTextComponent를 상속받아 각 컴포넌트의 특징에 맞게 설계되었다.



JTextField와 JPasswordField는 단일 라인의 텍스트를 편집할 수 있고, JTextArea, JEditorPane, JTextPane은 멀티 라인 편집을 지원한다.

JLabel

JLabel은 편집할 수 없는 한 줄의 간단한 텍스트와 정적인 이미지를 보여주는 컴포넌트이다. JLabel에 텍스트와 이미지를 설정하는 방법은 다음과 같다.

```
JLabel jLabel = new JLabel();
jLabel.setText("텍스트");
jLabel.setIcon(new ImageIcon( getClass().getResource("이미지파일"));
```

텍스트와 이미지의 배치는 정렬^{alignment}과 위치^{position} 그리고 간격^{gap}으로 조절할 수 있다. 정렬은 JLabel 전체 내용물의 위치를 의미하고, 위치는 이미지와 텍스트 사이의 상대적인 위치를 의미한다. 그리고 간격은 텍스트와 이미지의 간격이다.

```
//JLabel 영역에서의 내용물(텍스트+이미지)의 위치
setHorizontalAlignment( JLabel.LEFT | JLabel.CENTER | JLabel.RIGHT );
setVerticalAlignment( JLabel.TOP | JLabel.CENTER | JLabel.BOTTOM );

//텍스트와 이미지의 상대적인 위치
setHorizontalTextPosition( JLabel.LEFT | JLabel.CENTER | JLabel.RIGHT );
setVerticalTextPosition( JLabel.TOP | JLabel.CENTER | JLabel.BOTTOM );

//텍스트와 이미지 사이의 간격
setIconTextGap( iconTextGap );
```

JLabel의 경계선은 기본적으로 없기 때문에 경계선의 모양을 주고 싶다면 `setBorder()` 메소드를 사용할 수 있다.

```
setBorder(Border border);
```

매개값은 Border 인터페이스 구현 객체인데, Border 구현 클래스는 `javax.swing.border` 패키지에 포함되어 있다. 예를 들어 조각칼로 판 모양의 경계를 사용하고 싶다면 다음과 같이 `EtchedBorder`를 지정하면 된다.

```
setBorder(new EtchedBorder());
```

JLabel은 마우스로 클릭할 수 없고, 키보드로 편집할 수도 없기 때문에, 특별한 이벤트가 발생하지 않는다.

>>> JLabelExample.java

```
1 package sec07.exam01_jlabel;
2
3 import java.awt.GridLayout;
4 import javax.swing.ImageIcon;
5 import javax.swing.JFrame;
6 import javax.swing.JLabel;
```

```

7  import javax.swing.SwingUtilities;
8  import javax.swing.border.EtchedBorder;
9
10 public class JLabelExample extends JFrame {
11     private JLabel jLabel1, jLabel2, jLabel3, jLabel4;
12
13     //메인 윈도우 설정
14     public JLabelExample() {
15         this.setTitle("JLabelExample");
16         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
17         this.getContentPane().setLayout(new GridLayout(4, 1));
18
19         this.getContentPane().add(getJLabel1());
20         this.getContentPane().add(getJLabel2());
21         this.getContentPane().add(getJLabel3());
22         this.getContentPane().add(getJLabel4());
23         this.setSize(200, 300);
24     }
25
26     //JLabel 생성: 텍스트 좌측 정렬, EtchedBorder 적용
27     public JLabel getJLabel1() {
28         if (jLabel1 == null) {
29             jLabel1 = new JLabel();
30             jLabel1.setText("JLabel1");
31             jLabel1.setHorizontalAlignment(JLabel.LEFT);
32             jLabel1.setBorder(new EtchedBorder());
33         }
34         return jLabel1;
35     }
36
37     //JLabel 생성: 이미지 추가, 내용물 중앙 정렬
38     public JLabel getJLabel2() {
39         if (jLabel2 == null) {
40             jLabel2 = new JLabel();
41             jLabel2.setText("JLabel2");
42             jLabel2.setIcon(new ImageIcon(getClass().getResource("user.gif")));
43             jLabel2.setHorizontalAlignment(JLabel.CENTER);
44             jLabel2.setBorder(new EtchedBorder());
45         }
46         return jLabel2;

```

```

47     }
48
49     //JLabel 생성: 이미지 왼쪽에 텍스트가 오도록 설정
50     public JLabel getJLabel3() {
51         if (jLabel3 == null) {
52             jLabel3 = new JLabel();
53             jLabel3.setText("JLabel3");
54             jLabel3.setIcon(new ImageIcon(getClass().getResource("user.gif")));
55             jLabel3.setHorizontalAlignment(JLabel.CENTER);
56             jLabel3.setHorizontalTextPosition(JLabel.LEFT);
57             jLabel3.setBorder(new EtchedBorder());
58         }
59         return jLabel3;
60     }
61
62     //JLabel 생성: 이미지와 텍스트 사이의 간격 설정
63     public JLabel getJLabel4() {
64         if (jLabel4 == null) {
65             jLabel4 = new JLabel();
66             jLabel4.setText("JLabel4");
67             jLabel4.setIcon(new ImageIcon(getClass().getResource("user.gif")));
68             jLabel4.setHorizontalAlignment(JLabel.CENTER);
69             jLabel4.setIconTextGap(20);
70             jLabel4.setBorder(new EtchedBorder());
71         }
72         return jLabel4;
73     }
74
75     public static void main(String[] args) {
76         SwingUtilities.invokeLater(new Runnable() {
77             public void run() {
78                 JLabelExample jFrame = new JLabelExample();
79                 jFrame.setVisible(true);
80             }
81         });
82     }
83 }

```



JTextField와 JPasswordField

JTextField와 JPasswordField는 단일 라인의 텍스트 입력란을 제공하는 컴포넌트이다. 차이점은 JPasswordField는 사용자의 입력을 다른 사람이 볼 수 없도록 숨긴다는 것이다.

```
JTextField jTextField = new JTextField();
JPasswordField jPasswordField = new JPasswordField();
```

사용자가 입력한 텍스트는 JTextField일 경우 `getText()` 메소드로, JPasswordField일 경우에는 `getPassword()` 메소드로 얻을 수 있다. `getText()`는 String 타입으로 리턴하지만, `getPassword()`는 `char[]` 배열로 리턴하므로 String 타입으로 변환할 필요가 있다.

```
String inputData = jTextField.getText();
String inputData = new String( jPasswordField.getPassword() );
```

JTextField와 JPasswordField는 두 가지 주요 이벤트가 발생한다. 키보드로 문자를 입력할 때마다 `KeyEvent`가 발생하고, `[Enter]` 키를 입력하면 `ActionEvent`가 발생한다. 사용자가 입력하는 각 문자마다 처리할 내용이 있다면 `KeyEvent`를 처리하는 것이 좋고, `[Enter]` 키를 누르기 전까지 입력된 모든 문자들을 한꺼번에 처리하려면 `ActionEvent`를 처리하는 것이 좋다.

KeyEvent를 처리하기 위해 KeyListener 객체를 등록하는 코드는 다음과 같다. KeyListener의 keyPressed() 메소드는 키보드를 누를 때마다 실행된다.

```
jTextField.addKeyListener( new KeyAdapter() {
    public void keyPressed(KeyEvent e) {
        char keyCar = e.getKeyChar(); //입력된 문자 얻기
        int keyCode = e.getKeyCode(); //입력된 키코드 얻기
        ...
    }
} );
```

KeyEvent의 getKeyChar() 메소드는 입력된 키문자를 리턴하고, getKeyCode() 메소드는 키코드를 리턴한다. 키코드는 KeyEvent 클래스의 상수로 선언되어 있기 때문에 키보드에서 어떤 키가 입력되었는지 확인하려면 상수와 비교하면 된다. 예를 들어 다음은 [F1], [F2], [F3] 키가 입력되었는지 확인한다.

```
if(e.getKeyCode() == KeyEvent.VK_F1) { ... }
if(e.getKeyCode() == KeyEvent.VK_F2) { ... }
if(e.getKeyCode() == KeyEvent.VK_F3) { ... }
```

입력한 키가 유효한 키코드 범위에 속하는 지 검사할 수도 있다. 예를 들어 다음은 알파벳을 입력했는지 확인하는 코드이다.

```
if( (e.getKeyCode() >= KeyEvent.VK_A) && (e.getKeyCode() <= KeyEvent.VK_Z) ) { ... }
```

다음 예제는 아이디 입력 내용이 알파벳인지 검사하고, [Enter] 키를 누르면 입력한 패스워드를 보여준다.

>>> JTextFieldJPasswordFieldExample.java

```
1 package sec07.exam02_jtextfield_jpasswordfield;
2
3 import java.awt.GridLayout;
```

```

4  import java.awt.event.ActionEvent;
5  import java.awt.event.ActionListener;
6  import java.awt.event.KeyAdapter;
7  import java.awt.event.KeyEvent;
8  import javax.swing.JFrame;
9  import javax.swing.JLabel;
10 import javax.swing.JOptionPane;
11 import javax.swing.JPasswordField;
12 import javax.swing.JTextField;
13 import javax.swing.SwingUtilities;
14
15 public class JTextFieldJPasswordFieldExample extends JFrame {
16     private JTextField txtId;
17     private JPasswordField txtPassword;
18
19     //메인 윈도우 설정
20     public JTextFieldJPasswordFieldExample() {
21         this.setTitle("JTextField & JPasswordField");
22         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
23         this.getContentPane().setLayout(new GridLayout(2, 2));
24         this.getContentPane().add(new JLabel("아이디", JLabel.CENTER));
25         this.getContentPane().add(getTxtId());
26         this.getContentPane().add(new JLabel("패스워드", JLabel.CENTER));
27         this.getContentPane().add(getTxtPassword());
28         this.setSize(200, 100);
29     }
30
31     //JTextField 생성
32     public JTextField getTxtId() {
33         if (txtId == null) {
34             txtId = new JTextField();
35             txtId.addKeyListener(new KeyAdapter() {
36                 public void keyPressed(KeyEvent e) {
37                     if (e.getKeyCode() >= KeyEvent.VK_A && e.getKeyCode() <=
38                         KeyEvent.VK_Z) {
39                         JOptionPane.showMessageDialog(
40                             JTextFieldJPasswordFieldExample.this, "알파벳 이군요");
41                     } else {
42                         JOptionPane.showMessageDialog(
43                             JTextFieldJPasswordFieldExample.this, "알파벳이 아니군요");

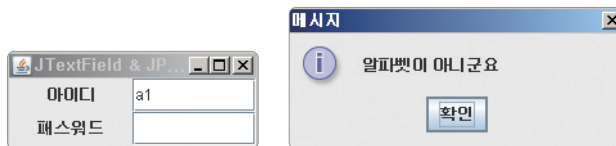
```

```

43         }
44     }
45     });
46 }
47     return txtId;
48 }
49
50 //JPasswordField 생성
51 public JPasswordField getTxtPassword() {
52     if (txtPassword == null) {
53         txtPassword = new JPasswordField();
54         txtPassword.addActionListener(new ActionListener() {
55             public void actionPerformed(ActionEvent e) {
56                 String password = new String(txtPassword.getPassword());
57                 JOptionPane.showMessageDialog(
58                     JTextFieldJPasswordFieldExample.this, "입력한 패스워드: " +
59                     password);
59             }
60         });
61     }
62     return txtPassword;
63 }
64
65 public static void main(String[] args) {
66     SwingUtilities.invokeLater(new Runnable() {
67         public void run() {
68             JTextFieldJPasswordFieldExample jFrame = new
69                 JTextFieldJPasswordFieldExample();
70             jFrame.setVisible(true);
71         }
72     });
73 }
74 }

```

실행 결과



JTextArea

JTextArea는 멀티 라인의 텍스트를 편집할 수 있는 컴포넌트이다. JTextArea는 자체적으로 스크롤을 제공하지 않으므로 JScrollPane에 추가해서 사용된다.

```
JTextArea jTextArea = new JTextArea();
JScrollPane jScrollPane = new JScrollPane( jTextArea );
```

JTextArea에서 키보드로 텍스트를 편집할 경우에는 스크롤이 따라 움직이지만, 프로그램에 의해 편집될 경우에는 스크롤이 따라 움직이지 않는다. 이럴 때 다음 코드를 추가하면 스크롤이 자동으로 내용에 맞게 움직이게 된다.

```
jTextArea.setCaretPosition( jTextArea.getText().length() );
```

다음 예제는 채팅창을 흉내내어 입력한 내용을 전송하면 JTextArea에 출력하도록 했다.

>>> JTextAreaExample.java

```
1  package sec07.exam03_jtextarea;
2
3  import java.awt.BorderLayout;
4  import java.awt.event.ActionEvent;
5  import java.awt.event.ActionListener;
6  import javax.swing.JButton;
7  import javax.swing.JFrame;
8  import javax.swing.JPanel;
9  import javax.swing.JScrollPane;
10 import javax.swing.JTextArea;
11 import javax.swing.JTextField;
12 import javax.swing.SwingUtilities;
13
14 public class JTextAreaExample extends JFrame {
15     private JTextArea txtDisplay;
16     private JPanel pSouth;
17     private JTextField txtInput;
```

```

18     private JButton btnSend;
19
20     //메인 윈도우 설정
21     public JTextAreaExample() {
22         this.setTitle("JTextAreaExample");
23         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
24         this.getContentPane().add(new JScrollPane(getTxtDisplay()),
25                                     BorderLayout.CENTER);
26         this.getContentPane().add(getPSouth(), BorderLayout.SOUTH);
27         this.setSize(300, 200);
28     }
29
30     //JTextArea 생성
31     public JTextArea getTxtDisplay() {
32         if (txtDisplay == null) {
33             txtDisplay = new JTextArea();
34             txtDisplay.setEditable(false);
35         }
36         return txtDisplay;
37     }
38
39     //남쪽에 부착할 JPanel 생성
40     public JPanel getPSouth() {
41         if (pSouth == null) {
42             pSouth = new JPanel();
43             pSouth.setLayout(new BorderLayout());
44             pSouth.add(getTxtInput(), BorderLayout.CENTER);
45             pSouth.add(btnSend, BorderLayout.EAST);
46         }
47         return pSouth;
48     }
49
50     //JTextField 생성
51     public JTextField getTxtInput() {
52         if (txtInput == null) {
53             txtInput = new JTextField();
54         }
55         return txtInput;
56     }

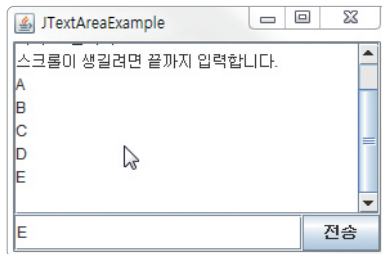
```

```

57     //JButton 생성
58     public JButton getBtnSend() {
59         if (btnSend == null) {
60             btnSend = new JButton();
61             btnSend.setText("전송");
62             btnSend.addActionListener(new ActionListener() {
63                 public void actionPerformed(ActionEvent e) {
64                     getTxtDisplay().append(getTxtInput().getText() + "\n");
65                     getTxtInput().setText("");
66                 }
67             });
68         }
69         return btnSend;
70     }
71
72     public static void main(String[] args) {
73         SwingUtilities.invokeLater(new Runnable() {
74             public void run() {
75                 JTextAreaExample jFrame = new JTextAreaExample();
76                 jFrame.setVisible(true);
77             }
78         });
79     }
80 }

```

실행 결과



JEditorPane

JEditorPane은 다양한 타입의 문서를 보여주거나 편집이 가능한 멀티 라인의 텍스트 컴포넌트이다. 기본적으로 단순 텍스트(text/plain), HTML(text/html) 타입의 문서를 지원한다. 다음 코드는 JEditorPane으로 HTML 파일의 내용을 보는 방법을 보여 준다.

```
JEditorPane jEditorPane = new JEditorPane();
try {
    jEditorPane.setPage(HTML파일URL);
} catch(Exception e) {}
jEditorPane.setEditable(false);
```

HTML 문서를 표시할 경우에는 setEditable(false)를 호출해서 편집할 수 없도록 하고, 사용자가 링크를 클릭할 경우 HyperlinkListener를 다음과 같이 등록해서 HyperlinkEvent 이벤트를 처리할 수 있다.

```
jEditorPane.addHyperlinkListener(new HyperlinkListener() {
    public void hyperlinkUpdate(HyperlinkEvent e) {
        if(e.getEventType() == HyperlinkEvent.EventType.ACTIVATED) {
            try {
                jEditorPane.setPage(e.getURL());
            } catch(Exception e) {}
        }
    }
});
```

HyperlinkEvent는 마우스를 링크 위로 가져갔을 경우와 링크 바깥으로 가져갔을 경우, 그리고 링크를 클릭했을 경우 모두 발생한다. 그래서 HyperlinkEvent의 getEventType() 메소드는 어떤 이벤트가 발생했느냐를 구별해주는 HyperlinkEvent.EventType형의 상수를 리턴한다.

마우스를 링크 위로 가져갔을 경우에는 ENTERED를 리턴하고, 마우스를 링크 바깥으로 가져갔을 경우에는 EXITED를 리턴한다. 그리고 링크를 클릭했을 경우에는 ACTIVATED를 리턴한다. ENTERED와 EXITED일 경우에는 링크의 색깔을 바꾸도록 처리하면 좋고, ACTIVATED는 실제로 링크된 문서를 불러와 보여주도록 처리하면 된다.

ACTIVATED가 리턴되었을 때 HyperlinkEvent의 `getURL()`을 호출하면 링크된 문서의 URL을 얻을 수 있다. 리턴된 URL을 JEditorPane의 `setPage()` 메소드의 매개변수로 주면 JEditorPane은 링크된 문서를 표시한다.

아쉬운 점은 JEditorPane은 HTML 3.2 태그만 지원하고, HTML 4.0이나, XHTML, HTML5는 지원하지 않는다.

>>> JEditorPaneExample.java

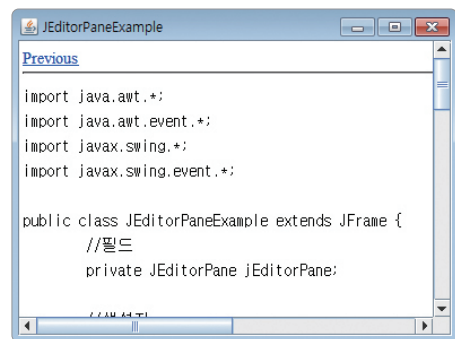
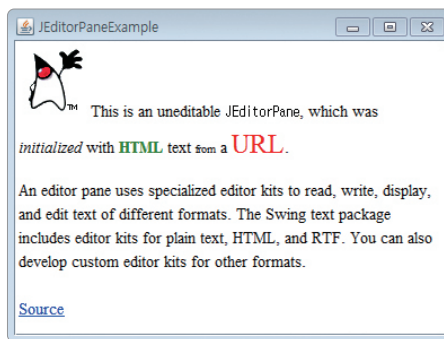
```
1  package sec07.exam04_jeditpane;
2
3  import java.awt.BorderLayout;
4  import java.io.IOException;
5  import javax.swing.JEditorPane;
6  import javax.swing.JFrame;
7  import javax.swing.JScrollPane;
8  import javax.swing.SwingUtilities;
9  import javax.swing.event.HyperlinkEvent;
10 import javax.swing.event.HyperlinkListener;
11
12 public class JEditorPaneExample extends JFrame {
13     private JEditorPane jEditorPane;
14
15     //메인 윈도우 설정
16     public JEditorPaneExample() {
17         this.setTitle("JEditorPaneExample");
18         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
19         this.getContentPane().add(new JScrollPane(getJEditorPane()),
20                                   BorderLayout.CENTER);
21         this.setSize(400, 300);
22     }
23
24     //JEditorPane 생성
25     public JEditorPane getJEditorPane() {
26         if(jEditorPane == null) {
27             jEditorPane = new JEditorPane();
28             try {
29                 jEditorPane.setPage(getClass().getResource("jeditorpane.html"));
30             } catch (Exception e) {}
31         }
32     }
33 }
```

```

30     jEditorPane.setEditable(false);
31     jEditorPane.addHyperlinkListener(new HyperlinkListener() {
32         public void hyperlinkUpdate(HyperlinkEvent e) {
33             if(e.getEventType() == HyperlinkEvent.EventType.ACTIVATED) {
34                 try {
35                     jEditorPane.setPage(e.getURL());
36                 } catch(IOException e2) {}
37             }
38         }
39     });
40 }
41 return jEditorPane;
42 }
43
44 public static void main(String[] args) {
45     SwingUtilities.invokeLater(new Runnable() {
46         public void run() {
47             JEditorPaneExample jFrame = new JEditorPaneExample();
48             jFrame.setVisible(true);
49         }
50     });
51 }
52 }

```

실행 결과



08 리스트 컴포넌트

리스트 컴포넌트는 목록 중에서 항목을 선택할 수 있는 컴포넌트를 말한다. 리스트 컴포넌트에는 JList, JComboBox가 있다. JList는 목록에서 하나 이상의 항목을 선택할 수 있고, JComboBox는 버튼을 클릭했을 때 목록이 보이고, 그 중 하나를 선택할 수 있다.

JList

JList는 목록에서 하나 이상의 항목을 선택할 수 있는 컴포넌트이다. JList의 항목은 다음과 같이 배열로 주거나,

```
String[] stringItems = { "one", "two", "tree", "four" };  
JList jList = new JList(stringItems);
```

다음과 같이 Vector 객체로 줄 수 있다.

```
Vector items = new Vector();  
items.add("one");  
items.add("two");  
items.add("tree");  
items.add("four");  
JList jList = new JList( items );
```

JList의 기본적인 항목 컴포넌트는 JLabel로 구성되어 있기 때문에 텍스트 이외에도 이미지 표현도 가능하다.

```
Vector items = new Vector();  
items.add( new ImageIcon( getClass().getResource("fruit1.gif") ) );  
items.add( new ImageIcon( getClass().getResource("fruit2.gif") ) );  
JList jList = new JList( items );
```

JList는 자동으로 스크롤 바가 생성되지 않으므로 JScrollPane에 추가해서 사용할 수 있다.

```
JScrollPane jScrollPane = new JScrollPane( jList );
```

사용자가 JList의 항목을 선택하면 ListSelectionEvent가 발생하기 때문에 ListSelectionListener를 등록하면 이벤트를 처리할 수 있다.

```
jList.addListSelectionListener(new ListSelectionListener() {  
    public void valueChanged(ListSelectionEvent e) {  
        if(e.getValueIsAdjusting() == false) {  
            //이벤트 처리 코드  
        }  
    }  
});
```

JList는 선택 항목이 변경되면 ListSelectionEvent를 두 번 발생시킨다. 이것을 해결하기 위해 ListSelectionEvent는 getValueIsAdjusting() 메소드를 제공하고 있다. 첫 번째 이벤트가 발생했을 때는 false를 리턴하고, 두 번째 이벤트가 연이어 발생하면 true를 리턴한다. 따라서 getValueIsAdjusting()가 false를 반환할 때만 이벤트 처리 코드를 실행하도록 해야 한다.

JList에서 선택된 항목을 알아내는 방법은 두 가지가 있다. getSelectedIndex() 메소드는 선택된 항목의 인덱스를 리턴하고, getSelectedValue()는 선택된 항목의 값이 리턴된다. 어떤 정보가 이벤트 처리 시 유리한지를 판단하고 사용하면 된다.

```
//선택된 항목의 인덱스 얻기  
int selectedIndex = jList.getSelectedIndex();  
int[] selectedIndices = jList.getSelectedIndices();  
  
//선택된 항목의 텍스트 얻기  
String selectedItem = (String) jList.getSelectedValue();  
String[] selectedItems = (String[]) jList.getSelectedValues();  
  
//선택된 항목의 이미지 얻기  
ImageIcon selectedItem = (ImageIcon) jList.getSelectedValue();  
ImageIcon[] selectedItems = (ImageIcon[]) jList.getSelectedValues();
```

만약 [Shift] 또는 [Ctrl] 키를 이용해서 두 개 이상의 항목을 선택했다면 선택된 항목의 인덱스와 값들을 배열로 리턴하는 메소드를 사용하면 된다.

다음 예제는 좌측 목록에서 항목을 선택하면 해당 항목의 이미지를 오른쪽에 보여 준다.

>>> JListExample.java

```
1  package sec08.exam01_jlist;
2
3  import java.awt.BorderLayout;
4  import java.awt.Color;
5  import java.awt.GridLayout;
6  import java.util.Vector;
7  import javax.swing.ImageIcon;
8  import javax.swing.JFrame;
9  import javax.swing.JLabel;
10 import javax.swing.JList;
11 import javax.swing.JPanel;
12 import javax.swing.JScrollPane;
13 import javax.swing.SwingUtilities;
14 import javax.swing.event.ListSelectionEvent;
15 import javax.swing.event.ListSelectionListener;
16
17 public class JListExample extends JFrame {
18     private JPanel pWest;
19     private JList listString;
20     private JList listImage;
21     private JLabel jLabel;
22
23     //메인 윈도우 설정
24     public JListExample() {
25         this.setTitle("JListExample");
26         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
27         this.setSize(250, 200);
28         this.getContentPane().setBackground(Color.WHITE);
29         this.getContentPane().add(getPWest(), BorderLayout.WEST);
30         this.getContentPane().add(getJLabel(), BorderLayout.CENTER);
31     }
32
33     //좌측 목록 JPanel 생성
34     public JPanel getPWest() {
35         if (pWest == null) {
36             pWest = new JPanel();
37             pWest.setLayout(new GridLayout(2, 1));
38             pWest.add(new JScrollPane(getListString()));
```

```

39         pWest.add(new JScrollPane(getListImage()));
40     }
41     return pWest;
42 }
43
44 //텍스트 목록을 갖는 JList 생성
45 public JList getListString() {
46     if (listString == null) {
47         String[] items = {
48             "Cantaloupe", "Grapefruit", "Grapes", "Kiwi", "Peach",
49             "pineapple", "strawberry", "tomato", "watermelon"
50         };
51         listString = new JList(items);
52         listString.addListSelectionListener(new ListSelectionListener() {
53             public void valueChanged(ListSelectionEvent e) {
54                 if (!e.getValueIsAdjusting()) {
55                     int selectedIndex = listString.getSelectedIndex();
56                     ImageIcon image = new ImageIcon(
57                         getClass().getResource("fruit" + (selectedIndex + 1) + ".jpg"));
58                     getJLabel().setIcon(image);
59                 }
60             }
61         });
62     }
63     return listString;
64 }
65
66 //이미지 목록을 갖는 JList 생성
67 public JList getListImage() {
68     if (listImage == null) {
69         Vector items = new Vector();
70         for (int i = 1; i < 10; i++) {
71             ImageIcon image = new ImageIcon(
72                 getClass().getResource("fruit" + i + ".jpg"));
73             items.addElement(image);
74         }
75         listImage = new JList(items);
76         listImage.addListSelectionListener(new ListSelectionListener() {
77             public void valueChanged(ListSelectionEvent e) {
78                 if (!e.getValueIsAdjusting()) {

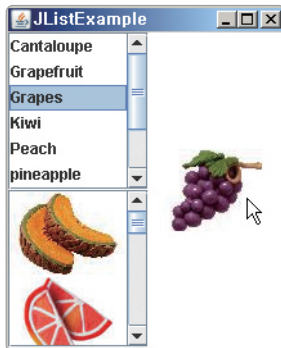
```

```

79         ImageIcon image = (ImageIcon) listImage.getSelectedValue();
80         getJLabel().setIcon(image);
81     }
82 }
83 });
84 }
85 return listImage;
86 }
87
88 //선택된 항목의 이미지를 보여주는 JLabel 생성
89 public JLabel getJLabel() {
90     if (jLabel == null) {
91         jLabel = new JLabel();
92         jLabel.setHorizontalAlignment(JLabel.CENTER);
93     }
94     return jLabel;
95 }
96
97 public static void main(String[] args) {
98     SwingUtilities.invokeLater(new Runnable() {
99         public void run() {
100             JListExample jFrame = new JListExample();
101             jFrame.setVisible(true);
102         }
103     });
104 }
105 }

```

실행 결과



JComboBox

JComboBox는 버튼을 클릭하면 항목이 나열되고 그 중 하나를 선택할 수 있다. JComboBox의 항목은 다음과 같이 배열로 주거나,

```
String[] items = { "one", "two", "tree", "four" }  
JComboBox jComboBox = new JComboBox( items );
```

다음과 같이 Vector 객체로 줄 수 있다.

```
Vector items = new Vector();  
items.add("one");  
items.add("two");  
items.add("tree");  
items.add("four");  
JComboBox jComboBox = new JComboBox( items );
```

JComboBox의 항목은 텍스트와 이미지 모두 가능하다. 이미지 항목은 ImageIcon 객체를 배열 또는 Vector 항목으로 만들면 된다.

```
Vector items = new Vector();  
items.add( new ImageIcon( getClass().getResource("fruit1.gif") ) );  
items.add( new ImageIcon( getClass().getResource("fruit2.gif") ) );  
JComboBox jComboBox = new JComboBox( items );
```

JComboBox에서도 `getSelectedIndex()` 메소드는 선택된 항목의 인덱스를 리턴하고, `getSelectedItem()`은 선택된 항목의 값을 리턴한다.

```
//선택된 항목의 인덱스 얻기  
int selectedIndex = jComboBox.getSelectedIndex();  
  
//선택된 항목의 텍스트 얻기  
String selectedItem = (String) jComboBox.getSelectedItem();  
  
//선택된 항목의 이미지 얻기  
ImageIcon selectedItem = (ImageIcon) jComboBox.getSelectedItem();
```

다음 예제는 상단의 JComboBox에서 항목을 선택하면, 해당 이미지를 아래에 보여 준다.

>> JComboBoxExample.java

```
1  package sec08.exam02_jcombobox;
2
3  import java.awt.BorderLayout;
4  import java.awt.Color;
5  import java.awt.event.ActionEvent;
6  import java.awt.event.ActionListener;
7  import java.util.Vector;
8  import javax.swing.ImageIcon;
9  import javax.swing.JComboBox;
10 import javax.swing.JFrame;
11 import javax.swing.JLabel;
12 import javax.swing.JPanel;
13 import javax.swing.SwingUtilities;
14
15 public class JComboBoxExample extends JFrame {
16     private JPanel pNorth;
17     private JComboBox comboString;
18     private JComboBox comboImage;
19     private JLabel jLabel;
20
21     //메인 윈도우 설정
22     public JComboBoxExample() {
23         this.setTitle("JComboBoxExample");
24         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
25         this.getContentPane().setBackground(Color.WHITE);
26         this.getContentPane().add(getPNorth(), BorderLayout.NORTH);
27         this.getContentPane().add(jLabel(), BorderLayout.CENTER);
28         this.setSize(250, 200);
29     }
30
31     //상단 목록 JPanel 생성
32     public JPanel getPNorth() {
33         if (pNorth == null) {
34             pNorth = new JPanel();
35             pNorth.add(getComboString());
36             pNorth.add(getComboImage());
```

```

37     }
38     return pNorth;
39 }
40
41 //텍스트 목록을 갖는 JComboBox 생성
42 public JComboBox getComboString() {
43     if (comboString == null) {
44         String[] arrString = {
45             "Cantaloupe", "Grapefruit", "Grapes", "Kiwi", "Peach",
46             "pineapple", "strawberry", "tomato", "watermelon" };
47         comboString = new JComboBox(arrString);
48         comboString.setBackground(Color.WHITE);
49         comboString.addActionListener(new ActionListener() {
50             public void actionPerformed(ActionEvent e) {
51                 int selectedIndex = comboString.getSelectedIndex();
52                 ImageIcon image = new ImageIcon(
53                     getClass().getResource("fruit" + (selectedIndex + 1) + ".jpg"));
54                 getJLabel().setIcon(image);
55             }
56         });
57     }
58     return comboString;
59 }
60
61 //이미지 목록을 갖는 JComboBox 생성
62 public JComboBox getComboImage() {
63     if (comboImage == null) {
64         Vector vImage = new Vector();
65         for (int i = 1; i < 10; i++) {
66             ImageIcon image = new ImageIcon(
67                 getClass().getResource("fruit" + i + ".jpg"));
68             vImage.add(image);
69         }
70         comboImage = new JComboBox(vImage);
71         comboImage.setBackground(Color.WHITE);
72         comboImage.addActionListener(new ActionListener() {
73             public void actionPerformed(ActionEvent e) {
74                 ImageIcon image = (ImageIcon) comboImage.getSelectedItem();
75                 getJLabel().setIcon(image);
76             }
77         });
78     }
79 }

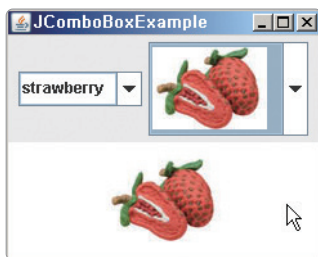
```

```

77         });
78     }
79     return comboImage;
80 }
81
82 //선택된 항목의 이미지를 보여주는 JLabel 생성
83 public JLabel getJLabel() {
84     if (jLabel == null) {
85         jLabel = new JLabel();
86         jLabel.setHorizontalAlignment(JLabel.CENTER);
87     }
88     return jLabel;
89 }
90
91 public static void main(String[] args) {
92     SwingUtilities.invokeLater(new Runnable() {
93         public void run() {
94             JComboBoxExample jFrame = new JComboBoxExample();
95             jFrame.setVisible(true);
96         }
97     });
98 }
99 }

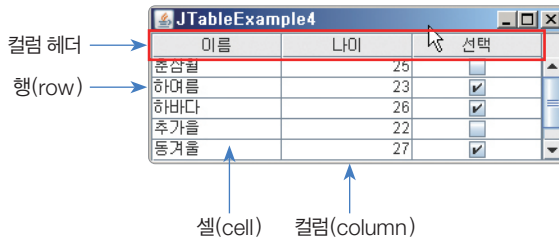
```

실행 결과



09 테이블 컴포넌트

JTable은 테이블 형식의 데이터를 표시하고 편집할 수 있는 컴포넌트이다. JTable은 다른 컴포넌트에 비해서 다소 복잡한 구조로 이루어져 있다.



테이블은 컬럼(column)과 행(row)으로 구성되어 있고, 컬럼과 행이 만나는 곳이 셀(cell)이다. 셀은 실제 데이터가 표시되는 곳이다. 하나의 컬럼을 구성하는 셀들은 동일한 데이터 타입을 가져야 한다. 위의 테이블에서 이름 컬럼의 데이터는 모두 텍스트, 나이 컬럼의 데이터는 모두 숫자로 구성되어 있는 것을 볼 수 있다.

테이블 생성

간단한 JTable 객체를 만들기 위해서는 먼저 컬럼 이름을 포함하고 있는 1차원 String 배열과 셀의 데이터인 2차원 Object 배열을 생성해야 한다. 그리고 이들을 JTable 생성자를 호출할 때 매개 값으로 넘겨준다.

```
String[] columnNames = { "이름", "나이" };
Object[][] rowData = {
    { "춘삼월", 25 },
    { "하여름", 23 },
    { "하바다", 26 }
};
JTable jTable = new JTable(rowData, columnNames);
```

2차원 Object 배열을 생성할 때 주의할 점은 하나의 컬럼을 구성하는 셀의 데이터들은 모두 같은 타입이어야 한다. 앞의 코드를 보면 이름 컬럼의 데이터는 모두 String 객체, 나이는 모두 Integer 객체로 주었다.

각 컬럼의 폭을 변경하고 싶다면 주어진 컬럼 이름에 해당하는 `TableColumn`을 `getColumn()` 메소드로 얻고, `TableColumn`의 `setPreferredWidth()` 메소드로 폭을 변경하면 된다.

```
TableColumn tableColumn = jTable.getColumn("컬럼 이름");
tableColumn.setPreferredWidth(폭길이);
```

`JTable`은 자체적으로 스크롤을 지원하지 않으므로 `JScrollPane`에 추가해서 사용할 수 있다.

```
JScrollPane jScrollPane = new JScrollPane( jTable);
jFrame.getContentPane().add(jScrollPane, BorderLayout.CENTER);
```

다음 예제는 이름과 나이 컬럼으로 구성된 테이블을 생성하는 방법을 보여 준다.

>>> `JTableExample.java`

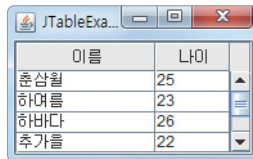
```
1  package sec09.exam01_jtable;
2
3  import java.awt.BorderLayout;
4  import javax.swing.JFrame;
5  import javax.swing.JScrollPane;
6  import javax.swing.JTable;
7  import javax.swing.SwingUtilities;
8
9  public class JTableExample extends JFrame {
10     private JTable jTable;
11
12     public JTableExample() {
13         this.setTitle("JTableExample1");
14         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
15         this.getContentPane().add(new JScrollPane(getJTable()),
16                                     BorderLayout.CENTER);
17         this.setSize(200, 125);
18     }
19
20     public JTable getJTable() {
21         if(jTable == null) {
```

```

21     String[] columnNames = { "이름", "나이" };
22     Object[][] rowData = {
23         { "춘삼월", 25 },
24         { "하여름", 23 },
25         { "하바다", 26 },
26         { "추가울", 22 },
27         { "동겨울", 27 },
28         { "동장군", 15 }
29     };
30     jTable = new JTable(rowData, columnNames);
31     jTable.getColumnModel().setPreferredWidth(100);
32     jTable.getColumnModel().setPreferredWidth(50);
33 }
34 return jTable;
35 }
36
37 public static void main(String[] args) {
38     SwingUtilities.invokeLater(new Runnable() {
39         public void run() {
40             JTableExample jFrame = new JTableExample();
41             jFrame.setVisible(true);
42         }
43     });
44 }
45 }

```

실행 결과



테이블 모델

JTable은 TableModel 객체를 사용해서 컬럼과 셀의 데이터를 관리한다. JTable을 생성할 때 컬럼 이름인 1차원 String 배열과 셀의 데이터인 2차원 Object 배열을 생성자로 넘겨주면, TableModel 이 내부적으로 생성되고 이 데이터들을 관리하게 된다.

JTable은 내부적으로 생성된 TableModel을 프로그램에서 사용할 수 있도록 getModel() 메소드를 제공한다.

```
TableModel tableModel = jTable.getModel();
```

TableModel에는 다음과 같이 데이터에 대한 여러 가지 정보를 제공하는 메소드들이 있다.

메소드		용도
int	getColumnCount()	총 컬럼의 수 얻기
String	getColumnName(int columnIndex)	컬럼의 이름 얻기
int	getRowCount()	총 행의 수 얻기
Object	getValueAt(int rowIndex, int columnIndex)	셀의 데이터 얻기
void	setValueAt(Object aValue, int rowIndex, int columnIndex)	셀의 데이터 변경
void	setDataVector(Object[][] rows, Object[] columnNames) setDataVector(Vector rows, Vector columnNames);	전체 테이블 데이터를 주어진 매개값으로 대체

>>> JTableExample.java

```

1  package sec09.exam02_tablemodel;
2
3  import java.awt.BorderLayout;
4  import java.awt.event.ActionEvent;
5  import java.awt.event.ActionListener;
6  import javax.swing.JButton;
7  import javax.swing.JFrame;
8  import javax.swing.JScrollPane;
9  import javax.swing.JTable;
10 import javax.swing.SwingUtilities;
11 import javax.swing.table.TableModel;
12
13 public class JTableExample extends JFrame {
14     private JTable jTable;
15     private JButton btnInfo;
16
17     public JTableExample() {

```

```

18     this.setTitle("JTableExample");
19     this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
20     this.getContentPane().add(new JScrollPane(getJTable()),
        BorderLayout.CENTER);
21     this.getContentPane().add(getBtnInfo(), BorderLayout.SOUTH);
22     this.setSize(200, 200);
23 }
24
25 public JTable getJTable() {
26     if(jTable == null) {
27         String[] columnNames = { "이름", "나이" };
28         Object[][] rowData = {
29             { "춘삼월", 25 },
30             { "하여름", 23 }
31         };
32         jTable = new JTable(rowData, columnNames);
33     }
34     return jTable;
35 }
36
37 public JButton getBtnInfo() {
38     if(btnInfo == null) {
39         btnInfo = new JButton();
40         btnInfo.setText("테이블 정보 출력");
41         btnInfo.addActionListener(new ActionListener() {
42             public void actionPerformed(ActionEvent e) {
43                 //테이블 모델 얻기
44                 TableModel tableModel = getJTable().getModel();
45                 //전체 컬럼 수 얻기
46                 int columnCount = tableModel.getColumnCount();
47                 //전체 행수 얻기
48                 int rowCount = tableModel.getRowCount();
49                 //컬럼 이름 출력
50                 for(int i=0; i<columnCount; i++) {
51                     String columnName = tableModel.getColumnName(i);
52                     System.out.print(columnName + "\t\t");
53                 }
54                 System.out.println();
55                 System.out.println("-----");
56                 //행의 데이터 출력

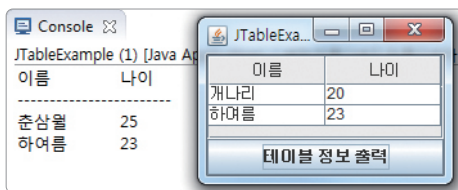
```

```

57         for(int i = 0; i<rowCount; i++) {
58             String column0 = (String) tableModel.getValueAt(i, 0);
59             Integer column1 = (Integer) tableModel.getValueAt(i, 1);
60             System.out.println(column0 + "\t\t" + column1);
61         }
62         //1행 셀 데이터 변경
63         tableModel.setValueAt("개나리", 0, 0);
64         tableModel.setValueAt(20, 0, 1);
65     }
66     });
67 }
68 return btnInfo;
69 }
70
71 public static void main(String[] args) {
72     SwingUtilities.invokeLater(new Runnable() {
73         public void run() {
74             JTableExample jFrame = new JTableExample();
75             jFrame.setVisible(true);
76         }
77     });
78 }
79 }

```

실행 결과



셀 표현 변경

컬럼 헤더와 셀 텍스트 내용을 수평 정렬하거나 셀 내용으로 다양한 컴포넌트를 넣고 싶다면 새로운 셀 렌더러를 정의해야 한다. 셀 렌더러는 넣고 싶은 컴포넌트를 상속하고 TableCellRenderer 인터페이스를 구현해서 만든다.

```
public class MyTableCellRenderer extends 컴포넌트 implements TableCellRenderer {
    public Component getTableCellRendererComponent(...) {
        //컴포넌트를 초기화하고 반환하는 코드
        return this;
    }
};
```

getTableCellRendererComponent() 메소드는 상속받은 컴포넌트를 매개값으로 초기화하고 리턴해야 한다. getTableCellRendererComponent() 메소드의 매개변수는 다음과 같다.

매개변수	값 또는 참조
table	셀을 포함하고 있는 JTable
value	셀에 표시될 데이터
isSelected	셀을 포함하고 있는 행이 선택 되었는지 여부
hasFocus	셀에 포커스가 있는지 여부
row	셀을 포함하고 있는 행의 순번
column	셀을 포함하고 있는 컬럼의 순번

새로운 셀 렌더러가 정의되었다면 TableColumn의 setHeaderRenderer() 또는 setCellRenderer() 메소드로 헤더와 셀 렌더러를 변경하면 된다.

```
TableColumn tc = jTable.getColumn("컬럼 이름");
TableCellRenderer tc = new new MyTableCellRenderer();
//헤더 모양 변경
tableColumn.setHeaderRenderer(tc);
//셀 모양 변경
tableColumn.setCellRenderer(tc);
```

다음 예제는 첫 번째 컬럼에 텍스트를 중앙 정렬한 JLabel을 넣었고, 두 번째 컬럼에 이미지와 텍스트가 포함된 JLabel을, 세 번째 컬럼에는 JCheckBox를 넣었다. 그리고 마우스로 행을 선택하면 노란색 배경으로 변경된다.

>>> JTableExample.java

```
1  package sec09.exam03_cellrenderer;
2
3  import java.awt.BorderLayout;
4  import java.awt.Color;
5  import java.awt.Component;
6  import java.awt.Font;
7  import javax.swing.ImageIcon;
8  import javax.swing.JCheckBox;
9  import javax.swing.JFrame;
10 import javax.swing.JLabel;
11 import javax.swing.JScrollPane;
12 import javax.swing.JTable;
13 import javax.swing.SwingUtilities;
14 import javax.swing.table.TableCellRenderer;
15 import javax.swing.table.TableColumn;
16
17 public class JTableExample extends JFrame {
18     private JTable jTable;
19
20     //메인 윈도우 설정
21     public JTableExample() {
22         this.setTitle("JTableExample");
23         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
24         this.getContentPane().add(new JScrollPane(getJTable()),
25                                     BorderLayout.CENTER);
26         this.setSize(200, 125);
27     }
28
29     //JTable 생성
30     public JTable getJTable() {
31         if (jTable == null) {
32             String[] columnNames = { "이름", "나이", "선택" };
33             Object[][] rowData = {
34                 { "춘삼월", 25, false },
35                 { "하여름", 26, true },
36                 { "추가울", 22, false },
37                 { "동겨울", 27, true }
38             };
39         }
40     }
41 }
```

```

38
39     jTable = new JTable(rowData, columnNames);
40
41     //각 컬럼의 셀 렌더러 변경
42     TableColumn tcName = jTable.getColumn("이름");
43     tcName.setCellRenderer(new NameTableCellRenderer());
44     TableColumn tcAge = jTable.getColumn("나이");
45     tcAge.setCellRenderer(new AgeTableCellRenderer());
46     TableColumn tcSelect = jTable.getColumn("선택");
47     tcSelect.setCellRenderer(new SelectTableCellRenderer());
48 }
49 return jTable;
50 }
51
52 //이름 컬럼의 셀 렌더러 정의
53 public class NameTableCellRenderer extends JLabel implements
    TableCellRenderer {
54     public Component getTableCellRendererComponent(
55         JTable table, Object value, boolean isSelected, boolean hasFocus,
56         int row, int column) {
57         setText(value.toString());
58         setFont(new Font(null, Font.PLAIN, 12));
59         setHorizontalAlignment(JLabel.CENTER);
60         setOpaque(true); //JLabel의 배경을 불투명하게 설정
61         if (isSelected) {
62             setBackground(Color.YELLOW);
63         } else {
64             setBackground(Color.WHITE);
65         }
66         return this;
67     }
68 }
69
70 //나이 컬럼의 셀 렌더러 정의
71 public class AgeTableCellRenderer extends JLabel implements
    TableCellRenderer {
72     public Component getTableCellRendererComponent(
73         JTable table, Object value, boolean isSelected, boolean hasFocus,
74         int row, int column) {
75         int age = ((Integer) value).intValue();

```



```

76         if (age <= 25) {
77             setIcon(new ImageIcon(getClass().getResource("key.gif")));
78         } else {
79             setIcon(new ImageIcon(getClass().getResource("start.gif")));
80         }
81         setText(value.toString());
82         setFont(new Font(null, Font.PLAIN, 12));
83         setHorizontalAlignment(JLabel.CENTER);
84         setOpaque(true); //JLabel의 배경을 불투명하게 설정
85         if (isSelected) {
86             setBackground(Color.YELLOW);
87         } else {
88             setBackground(Color.WHITE);
89         }
90         return this;
91     }
92 }
93
94 //선택 컬럼의 셀 렌더러 정의
95 public class SelectTableCellRenderer extends JCheckBox
96         implements TableCellRenderer {
97     public Component getTableCellRendererComponent(
98         JTable table, Object value, boolean isSelected, boolean hasFocus,
99         int row, int column) {
100         Boolean boolWrapper = (Boolean) value;
101         setSelected(boolWrapper.booleanValue());
102         setHorizontalAlignment(CENTER);
103         if (isSelected) {
104             setBackground(Color.YELLOW);
105         } else {
106             setBackground(Color.WHITE);
107         }
108         return this;
109     }
110 }
111
112 public static void main(String[] args) {
113     SwingUtilities.invokeLater(new Runnable() {
114         public void run() {
115             JTableExample jFrame = new JTableExample();

```

```

116         JFrame.setVisible(true);
117     }
118 });
119 }
120 }

```

실행 결과



이름	나이	선택	
홍삼철	25	<input type="checkbox"/>	
하머름	26	<input checked="" type="checkbox"/>	
추가름	22	<input type="checkbox"/>	
동겨름	27	<input checked="" type="checkbox"/>	

60라인과 84라인의 `setOpaque(true)`는 JLabel의 배경을 투명에서 불투명으로 변경한다. 테이블의 행을 선택했을 때 JLabel의 노란색 배경색이 보이도록 하기 위해서이다.

이벤트 처리

JTable은 MouseEvent가 발생하므로 MouseListener를 등록해서 이벤트를 처리할 수 있다. MouseEvent에서 `getSelectedColumn()`과 `getSelectedRow()` 메소드를 이용하면 어떤 컬럼과 행에서 클릭되었는지 알아낼 수 있다.

```

jTable.addMouseListener(new MouseAdapter() {
    public void mouseClicked(MouseEvent e) {
        int rowIndex = jTable.getSelectedRow();
        int columnIndex = jTable.getSelectedColumn();
        if(rowIndex != -1 || columnIndex != -1) {
            //이벤트 처리 코드
        }
    }
});

```

MouseEvent가 발생했지만 클릭된 컬럼과 행이 없다면 `getSelectedColumn()`과 `getSelectedRow()`는 모두 -1을 리턴한다. 따라서 -1이 아닐 때 이벤트 처리 코드를 실행하면 된다.

다음 예제는 사용자가 행을 선택하면 행의 데이터를 읽고 이름과 나이를 JTextField에 표시한다.

>>> JTableExample.java

```
1  package sec09.exam04_eventhandling;
2
3  import java.awt.BorderLayout;
4  import java.awt.GridLayout;
5  import java.awt.event.MouseAdapter;
6  import java.awt.event.MouseEvent;
7  import javax.swing.JFrame;
8  import javax.swing.JLabel;
9  import javax.swing.JPanel;
10 import javax.swing.JScrollPane;
11 import javax.swing.JTable;
12 import javax.swing.JTextField;
13 import javax.swing.SwingUtilities;
14 import javax.swing.table.TableModel;
15
16 public class JTableExample extends JFrame {
17     private JTable jTable;
18     private JPanel pSouth;
19     private JTextField txtName;
20     private JTextField txtAge;
21     private Object[][] rowData;
22
23     //메인 윈도우 설정
24     public JTableExample() {
25         this.setTitle("JTableExample");
26         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
27         this.getContentPane().add(new JScrollPane(getJTable()),
28             BorderLayout.CENTER);
29         this.getContentPane().add(getPSouth(), BorderLayout.SOUTH);
30         this.setSize(200, 230);
31     }
32
33     //JTable 생성
34     public JTable getJTable() {
35         if(jTable == null) {
36             String[] columnNames = new String[] {"이름", "나이" };
```

```

36         rowData = new Object[][] {
37             { "춘삼월", 25 },
38             { "하여름", 23 },
39             { "추가울", 22 },
40             { "동겨울", 27 }
41         };
42         jTable = new JTable(rowData, columnNames);
43
44         //MouseEvent 처리
45         jTable.addMouseListener(new MouseAdapter() {
46             public void mouseClicked(MouseEvent e) {
47                 int rowIndex = jTable.getSelectedRow();
48                 if(rowIndex != -1) {
49                     //클릭된 행의 셀값 읽기
50                     TableModel tableModel = jTable.getModel();
51                     String name = (String) tableModel.getValueAt(rowIndex, 0);
52                     Integer age = (Integer) tableModel.getValueAt(rowIndex, 1);
53                     //읽은 값을 아래쪽에 있는 JPanel에서 보여주기
54                     getTxtName().setText(name);
55                     getTxtAge().setText(String.valueOf(age.intValue()));
56                 }
57             }
58         });
59     }
60     return jTable;
61 }
62
63 //하단에 위치할 JPanel 생성
64 public JPanel getPSouth() {
65     if(pSouth == null) {
66         pSouth = new JPanel();
67         pSouth.setLayout(new GridLayout(4,2));
68         pSouth.add(new JLabel("[선택한 행 정보]"));
69         pSouth.add(new JLabel(""));
70         pSouth.add(new JLabel("이름", JLabel.CENTER));
71         pSouth.add(getTxtName());
72         pSouth.add(new JLabel("나이", JLabel.CENTER));
73         pSouth.add(getTxtAge());
74     }
75     return pSouth;

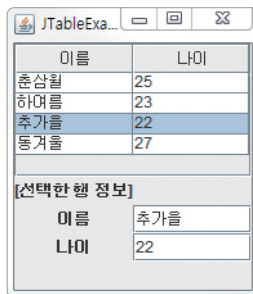
```

```

76     }
77
78     public JTextField getTxtName() {
79         if(txtName == null) {
80             txtName = new JTextField();
81         }
82         return txtName;
83     }
84
85     public JTextField getTxtAge() {
86         if(txtAge == null) {
87             txtAge = new JTextField();
88         }
89         return txtAge;
90     }
91
92     public static void main(String[] args) {
93         SwingUtilities.invokeLater(new Runnable() {
94             public void run() {
95                 JTableExample jFrame = new JTableExample();
96                 jFrame.setVisible(true);
97             }
98         });
99     }
100 }

```

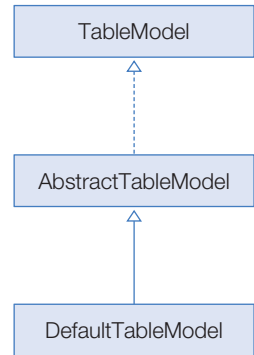
실행 결과



행 추가, 수정, 삭제

TableModel 타입은 JTable이 어떤 생성자로 만들어졌는지에 따라 다르다. JTable(Object[][] rowData, String[] columnNames) 생성자로 만들 경우 AbstractTableModel을 상속해서 만들어지고, 매개변수가 없는 JTable() 생성자로 만들었다면 내부적으로 DefaultTableModel을 상속해서 만든다.

AbstractTableModel과 DefaultTableModel의 차이점은 DefaultTableModel은 행을 추가, 삽입, 삭제할 수 있는 다음 메소드들이 추가적으로 제공된다는 점이다.



메소드		용도
void	addRow(Object[] rowData) addRow(Vector rowData)	맨 뒤에 행 추가
void	insertRow(int rowIndex, Object[] rowData) insertRow(int rowIndex, Vector rowData)	주어진 인덱스에 행 삽입
void	removeRow(int rowIndex)	주어진 인덱스의 행 삭제

이 메소드들을 이용하려면 JTable() 생성자로 JTable을 생성하고, getModel() 메소드가 리턴하는 TableModel을 DefaultTableModel로 변환해야 한다.

```
JTable jTable = new JTable();
DefaultTableModel tableModel = (DefaultTableModel) jTable.getModel();
//맨 뒤에 행 추가
tableModel.addRow(rowData);
//0 인덱스에 행 삽입
tableModel.insertRow(0, rowData);
//2 인덱스의 행 삭제
tableModel.removeRow(2);
```

DefaultTableModel은 행의 데이터를 Vector로 관리한다. 각 행은 Vector로 생성되고, 행 Vector들은 다시 전체 행을 관리하는 Vector에 저장된다.

```
Vector(전체 행 관리)
|-- Vector(0 인덱스 행 데이터)
|-- Vector(1 인덱스 행 데이터)
|-- ...
```

DefaultTableModel에서 전체 행을 관리하는 Vector는 `getDataVector()` 메소드로 얻을 수 있다.

```
Vector<Vector> rows = tableModel.getDataVector();
```

그리고 각 행의 Vector는 행 인덱스를 주고 다음과 같이 얻을 수 있다.

```
Vector row = rows.elementAt(행인덱스);
```

행 Vector의 내부 요소를 변경하기 위해서는 Vector의 `set()` 메소드를 이용하면 되는데, 첫 번째 매개값은 컬럼의 인덱스 번호이고, 두 번째 매개값은 컬럼의 값이다.

```
row.set(0, changeValue);    //0번 컬럼의 값을 changeValue로 수정
row.set(1, changeValue);    //1번 컬럼의 값을 changeValue로 수정
```

행 Vector를 이용해서 행의 데이터를 수정하였다면 JTable을 다시 렌더링할 수 있도록 DefaultTableModel의 `fireTableDataChanged()` 메소드를 호출해야 한다.

```
tableModel.fireTableDataChanged();
```

다음 예제는 사용자가 입력한 내용을 JTable의 행으로 넣는다. 그리고 선택된 행의 데이터를 수정하고 삭제할 수 있다.

>> JTableExample.java

```
1  package sec09.exam05_row_add_update_delete;
2
3  import java.awt.BorderLayout;
4  import java.awt.GridLayout;
5  import java.awt.event.ActionEvent;
6  import java.awt.event.ActionListener;
7  import java.awt.event.MouseAdapter;
8  import java.awt.event.MouseEvent;
9  import java.util.Vector;
10 import javax.swing.JButton;
11 import javax.swing.JFrame;
12 import javax.swing.JLabel;
13 import javax.swing.JPanel;
14 import javax.swing.JScrollPane;
15 import javax.swing.JTable;
16 import javax.swing.JTextField;
17 import javax.swing.SwingUtilities;
18 import javax.swing.table.DefaultTableModel;
19
20 public class JTableExample extends JFrame {
21     private JTable jTable;
22     private JPanel pSouth;
23     private JTextField txtName, txtAge;
24     private JButton btnInsert, btnUpdate, btnDelete;
25
26     //메인 윈도우 설정
27     public JTableExample() {
28         this.setTitle("JTableExample");
29         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
30         this.getContentPane().add(new JScrollPane(getJTable()),
31                                     BorderLayout.CENTER);
32         this.getContentPane().add(getPSouth(), BorderLayout.SOUTH);
33         this.setSize(250, 250);
34     }
35
36     //JTable 생성
37     public JTable getJTable() {
38         if (jTable == null) {
```



```

38     jTable = new JTable();
39     final DefaultTableModel tableModel = (DefaultTableModel)
        jTable.getModel();
40     tableModel.addColumn("이름");
41     tableModel.addColumn("나이");
42     //행을 선택했을 때 이벤트 처리
43     jTable.addMouseListener(new MouseAdapter() {
44         public void mouseClicked(MouseEvent e) {
45             int rowIndex = jTable.getSelectedRow();
46             if (rowIndex != -1) {
47                 String name = (String) tableModel.getValueAt(rowIndex, 0);
48                 String age = (String) tableModel.getValueAt(rowIndex, 1);
49                 txtName.setText(name);
50                 txtAge.setText(age.toString());
51             }
52         }
53     });
54 }
55 return jTable;
56 }
57
58 //사용자 입력 JPanel 생성
59 public JPanel getPSouth() {
60     if (pSouth == null) {
61         pSouth = new JPanel();
62
63         pSouth.setLayout(new GridLayout(3, 1));
64
65         JPanel pNameInput = new JPanel();
66         pNameInput.setLayout(new GridLayout(1, 2));
67         pNameInput.add(new JLabel("이름", JLabel.CENTER));
68         pNameInput.add(getTxtName());
69         pSouth.add(pNameInput);
70
71         JPanel pAgeInput = new JPanel();
72         pAgeInput.setLayout(new GridLayout(1, 2));
73         pAgeInput.add(new JLabel("나이", JLabel.CENTER));
74         pAgeInput.add(getTxtAge());
75         pSouth.add(pAgeInput);
76

```

```

77         JPanel pButton = new JPanel();
78         pButton.add(getBtnInsert());
79         pButton.add(getBtnUpdate());
80         pButton.add(getBtnDelete());
81         pSouth.add(pButton);
82     }
83     return pSouth;
84 }
85
86 public JTextField getTxtName() {
87     if (txtName == null) {
88         txtName = new JTextField();
89     }
90     return txtName;
91 }
92
93 public JTextField getTxtAge() {
94     if (txtAge == null) {
95         txtAge = new JTextField();
96     }
97     return txtAge;
98 }
99
100 //행 삽입 버튼 생성
101 public JButton getBtnInsert() {
102     if (btnInsert == null) {
103         btnInsert = new JButton();
104         btnInsert.setText("추가");
105         btnInsert.addActionListener(new ActionListener() {
106             public void actionPerformed(ActionEvent e) {
107                 Object[] rowData = { txtName.getText(), txtAge.getText() };
108                 DefaultTableModel tableModel = (DefaultTableModel)
109                     getJTable().getModel();
110                 tableModel.addRow(rowData);
111                 txtName.setText("");
112                 txtAge.setText("");
113             }
114         });
115     return btnInsert;

```

```

116     }
117
118     //행 수정 버튼 생성
119     public JButton getBtnUpdate() {
120         if (btnUpdate == null) {
121             btnUpdate = new JButton();
122             btnUpdate.setText("수정");
123             btnUpdate.addActionListener(new ActionListener() {
124                 public void actionPerformed(ActionEvent e) {
125                     DefaultTableModel tableModel = (DefaultTableModel)
126                         getJTable().getModel();
127                     Vector<Vector> rows = tableModel.getDataVector();
128                     Vector row = rows.elementAt(jTable.getSelectedRow());
129                     row.set(0, txtName.getText());
130                     row.set(1, txtAge.getText());
131                     tableModel.fireTableDataChanged();
132                     txtName.setText("");
133                     txtAge.setText("");
134                 }
135             });
136             return btnUpdate;
137         }
138
139         //행 삭제 버튼 생성
140         public JButton getBtnDelete() {
141             if (btnDelete == null) {
142                 btnDelete = new JButton();
143                 btnDelete.setText("삭제");
144                 btnDelete.addActionListener(new ActionListener() {
145                     public void actionPerformed(ActionEvent e) {
146                         int rowIndex = getJTable().getSelectedRow();
147                         if (rowIndex != -1) {
148                             DefaultTableModel tableModel = (DefaultTableModel)
149                                 getJTable().getModel();
150                             tableModel.removeRow(rowIndex);
151                             txtName.setText("");
152                             txtAge.setText("");
153                         }
154                     }
155                 });
156             }
157             return btnDelete;
158         }
159     }

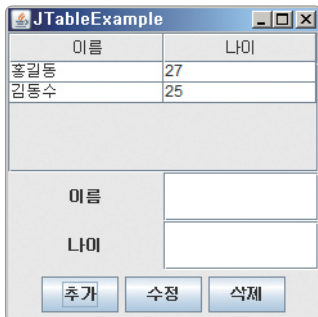
```

```

154         });
155     }
156     return btnDelete;
157 }
158
159 public static void main(String[] args) {
160     SwingUtilities.invokeLater(new Runnable() {
161         public void run() {
162             JTableExample jFrame = new JTableExample();
163             jFrame.setVisible(true);
164         }
165     });
166 }
167 }

```

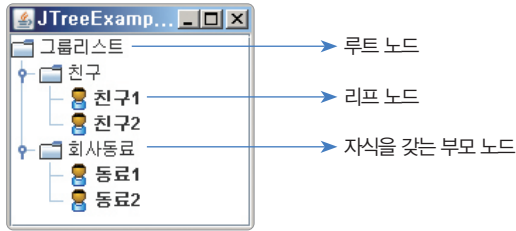
실행 결과



10 트리 컴포넌트

트리 컴포넌트는 계층적인 데이터를 표시하는 컴포넌트이다. Swing은 트리 컴포넌트로 JTree를 제공한다. JTree는 하나의 루트 노드root node 아래에 여러 개의 자식 노드child node를 가지며, 자식 노드는 또 다시 자식 노드를 가질 수 있다.

동일한 부모 노드parent node를 갖는 노드들을 묶어서 형제 노드sibling node라고 부르는데, 다음 그림에서 친구1과 친구2는 형제 노드이다. 자식 노드가 없는 마지막 노드는 잎사귀 노드라고 해서 리프leaf 노드라고 부른다.



트리 생성

JTree를 생성하려면 생성자의 매개값으로 루트 노드를 대입해야 하는데, 루트 노드는 DefaultMutableTreeNode로 생성한다. DefaultMutableTreeNode는 루트 노드뿐만 아니라 부모 노드, 리프 노드를 생성하는 데 사용된다.

다음은 DefaultMutableTreeNode로 생성된 계층적 데이터를 JTree를 생성할 때 매개값으로 제공해야 함을 보여 준다.

```
DefaultMutableTreeNode
|-- DefaultMutableTreeNode
|   |-- DefaultMutableTreeNode
|   |-- DefaultMutableTreeNode
|   |-- DefaultMutableTreeNode
JTree jTree = new JTree( );
```

노드의 표현 방법은 DefaultTreeCellRenderer가 결정한다. DefaultTreeCellRenderer는 JLabel의 하위 클래스로서 노드를 아이콘과 텍스트로 조합해서 보여 준다. 기본적으로 루트 노드나 부모 노드는 폴더 모양의 아이콘으로 표현하고, 리프 노드는 문서 모양의 아이콘으로 표현한다.

다음 예제는 기본적인 노드의 모양을 보여 준다.

>>> JTreeExample.java

```
1 package sec10.exam01_jtree;
2
3 import java.awt.BorderLayout;
4 import javax.swing.JFrame;
```

```

5  import javax.swing.JScrollPane;
6  import javax.swing.JTree;
7  import javax.swing.SwingUtilities;
8  import javax.swing.tree.DefaultMutableTreeNode;
9
10 public class JTreeExample extends JFrame {
11     private JTree jTree;
12
13     //메인 윈도우 설정
14     public JTreeExample() {
15         this.setTitle("JTreeExample");
16         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
17         this.getContentPane().add(new JScrollPane(getJTree()),
18                                     BorderLayout.CENTER);
19         this.setSize(200, 150);
20     }
21
22     //JTree 생성
23     public JTree getJTree() {
24         if (jTree == null) {
25             DefaultMutableTreeNode root = new DefaultMutableTreeNode("그룹리스트");
26
27             DefaultMutableTreeNode node1 = new DefaultMutableTreeNode("친구");
28             node1.add(new DefaultMutableTreeNode("친구1"));
29             node1.add(new DefaultMutableTreeNode("친구2"));
30             root.add(node1);
31
32             DefaultMutableTreeNode node2 = new DefaultMutableTreeNode("회사동료");
33             node2.add(new DefaultMutableTreeNode("동료1"));
34             node2.add(new DefaultMutableTreeNode("동료2"));
35             root.add(node2);
36
37             jTree = new JTree(root);
38         }
39         return jTree;
40     }
41
42     public static void main(String[] args) {
43         SwingUtilities.invokeLater(new Runnable() {
44             public void run() {

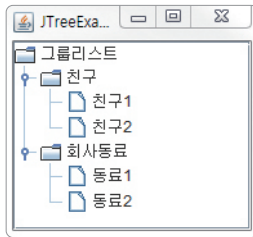
```

```

44         JTreeExample JFrame = new JTreeExample();
45         JFrame.setVisible(true);
46     }
47     });
48 }
49 }

```

실행 결과



노드 표현 변경

노드의 아이콘과 리프 노드의 표현을 다르게 변형하고 싶다면 새로운 `TreeCellRenderer`를 만들어 기본 렌더러인 `DefaultTreeCellRenderer`를 대체하면 된다. 새로운 `TreeCellRenderer`는 `TreeCellRenderer` 인터페이스를 구현해서 만든다.

```

public class MyTreeCellRenderer implements TreeCellRenderer {
    public Component getTreeCellRendererComponent(...) {
        //컴포넌트를 초기화하고 리턴하는 코드
    }
}

```

`getTreeCellRendererComponent()` 메소드는 주어진 매개값을 이용해서 노드를 표현할 컴포넌트를 리턴하는 역할을 한다. 매개변수는 총 6개로 다음과 같은 값을 가지고 있다.

매개변수	값 또는 참조
tree	노드를 포함하고 있는 JTree 참조
value	노드인 DefaultMutableTreeNode 객체
sel	노드가 선택되었는지 여부
expanded	부모 노드가 펼쳐졌는지 여부
leaf	리프 노드인지 여부
hasFocus	포커스를 가지고 있는지 여부

새로운 TreeCellRenderer가 정의되었다면 JTree의 setCellRenderer() 메소드로 기본 렌더러를 변경하면 된다.

```
jTree.setCellRenderer(new MyTreeCellRenderer());
```

다음 예제는 부모 노드의 아이콘을 변경하고 리프 노드를 아이콘+글자+아이콘으로 표현한다.

>>> JTreeExample.java

```

1  package sec10.exam02_cellrenderer;
2
3  import java.awt.BorderLayout;
4  import java.awt.Color;
5  import java.awt.Component;
6  import javax.swing.BorderFactory;
7  import javax.swing.ImageIcon;
8  import javax.swing.JFrame;
9  import javax.swing.JLabel;
10 import javax.swing.JPanel;
11 import javax.swing.JScrollPane;
12 import javax.swing.JTree;
13 import javax.swing.SwingUtilities;
14 import javax.swing.tree.DefaultMutableTreeNode;
15 import javax.swing.tree.TreeCellRenderer;
16
17 public class JTreeExample extends JFrame {

```



```

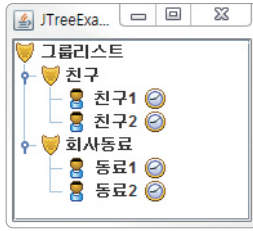
18     private JTree jTree;
19
20     //메인 윈도우 설정
21     public JTreeExample() {
22         this.setTitle("JTreeExample");
23         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
24         this.getContentPane().add(new JScrollPane(getJTree()),
25             BorderLayout.CENTER);
26         this.setSize(200, 150);
27     }
28
29     //JTree 생성
30     public JTree getJTree() {
31         if (jTree == null) {
32             DefaultMutableTreeNode root = new DefaultMutableTreeNode("그룹리스트");
33
34             DefaultMutableTreeNode node1 = new DefaultMutableTreeNode("친구");
35             node1.add(new DefaultMutableTreeNode("친구1"));
36             node1.add(new DefaultMutableTreeNode("친구2"));
37             root.add(node1);
38
39             DefaultMutableTreeNode node2 = new DefaultMutableTreeNode("회사동료");
40             node2.add(new DefaultMutableTreeNode("동료1"));
41             node2.add(new DefaultMutableTreeNode("동료2"));
42             root.add(node2);
43
44             jTree = new JTree(root);
45             //노드 표현 방법 변경
46             jTree.setCellRenderer(new MyTreeCellRenderer());
47         }
48         return jTree;
49     }
50
51     //TreeCellRender 정의
52     public class MyTreeCellRenderer implements TreeCellRenderer {
53         public Component getTreeCellRendererComponent(
54             JTree tree, Object value, boolean sel, boolean expanded,
55             boolean leaf, int row, boolean hasFocus) {
56             if (!leaf) {
57                 JLabel jLabel = new JLabel();

```

```

57         jLabel.setBorder(BorderFactory.createEmptyBorder(5, 0, 5, 0));
           //노드 간 상하 간격
58         jLabel.setIcon(new ImageIcon(getClass().getResource("parentnode.gif")));
59         jLabel.setText(value.toString());
60         return jLabel;
61     } else {
62         JPanel jPanel = new JPanel();
63         jPanel.setBackground(Color.WHITE);
64         jPanel.setLayout(new BorderLayout());
65         jPanel.setBorder(BorderFactory.createEmptyBorder(3, 0, 3, 0));
           //노드 간 상하 간격
66
67         JLabel lblWest = new JLabel(new ImageIcon(getClass().
           getResource("logon.gif")));
68         JLabel lblCenter = new JLabel(" " + value.toString() + " ");
69         JLabel lblEast = new JLabel(new ImageIcon(getClass().
           getResource("time.gif")));
70         jPanel.add(lblWest, BorderLayout.WEST);
71         jPanel.add(lblCenter, BorderLayout.CENTER);
72         jPanel.add(lblEast, BorderLayout.EAST);
73
74         if (sel) {
75             jPanel.setBackground(Color.ORANGE); //노드 선택 시 오렌지 배경색 설정
76         }
77         return jPanel;
78     }
79 }
80 }
81
82 public static void main(String[] args) {
83     SwingUtilities.invokeLater(new Runnable() {
84         public void run() {
85             JTreeExample jFrame = new JTreeExample();
86             jFrame.setVisible(true);
87         }
88     });
89 }
90 }

```



이벤트 처리

JTree에서 노드 선택이 변경되면 TreeSelectionEvent가 발생하기 때문에 TreeSelectionListener를 추가해서 이벤트를 처리할 수 있다. TreeSelectionListener의 valueChanged() 메소드는 노드 선택이 변경되면 호출된다. 다음은 TreeSelectionListener 구현 클래스를 작성하는 방법을 보여 준다.

```
public class MyTreeSelectionListener implements TreeSelectionListener {
    public void valueChanged(TreeSelectionEvent e) {
        //선택된 노드의 전체 경로 얻기
        TreePath treePath = e.getPath();
        //선택된 노드의 DefaultMutableTreeNode 얻기
        DefaultMutableTreeNode node =
            (DefaultMutableTreeNode) treePath.getLastPathComponent();
        //선택된 노드의 텍스트 얻기
        String userObject = (String) node.getUserObject();
        //처리 코드
        //~
    }
}
```

TreeSelectionEvent의 getPath()는 루트 노드에서부터 선택된 노드까지의 경로 정보를 가지고 있는 TreePath를 리턴한다. TreePath의 getLastPathComponent()는 선택된 노드를 Object 타입으로 리턴하는데, DefaultMutableTreeNode로 타입 변환할 수 있다.

선택된 노드의 문자열은 DefaultMutableTreeNode의 getUserObject() 메소드로 얻을 수 있

다. 만약 노드 선택 변경보다는 클릭과 더블 클릭에 더 관심이 있다면 다음과 같이 MouseEvent를 처리할 수도 있다.

```
public class MyMouseListener extends MouseAdapter {
    public void mousePressed(MouseEvent e) {
        //JTree 얻기
        JTree jTree = (JTree) e.getSource();
        //선택된 노드의 전체 경로 얻기
        TreePath treePath = jTree.getPathForLocation(e.getX(), e.getY());
        //선택된 노드가 있을 경우
        if(selRow != -1) {
            if(e.getClickCount() == 1) {
                //클릭했을 경우 실행할 코드
            } else if(e.getClickCount() == 2) {
                //더블 클릭했을 경우 실행할 코드
            }
        }
    }
}
```

JTree의 getPathForLocation() 메소드는 마우스로 클릭된 좌표를 가지고 선택된 노드의 전체 경로를 가진 TreePath를 리턴한다. 마우스를 클릭했느냐 더블 클릭했느냐는 MouseEvent의 getClickCount()의 리턴값으로 구분할 수 있으므로 if 문으로 실행할 코드를 선택할 수 있다.

다음 예제는 TreeSelectionListener를 추가해서 선택 노드의 텍스트를 메시지 다이얼로그로 나타내고, MouseListener를 추가해서 마우스 더블 클릭을 처리했다.

» JTreeExample.java

```
1 package sec10.exam03_eventhandling;
2
3 import java.awt.BorderLayout;
4 import java.awt.Color;
5 import java.awt.Component;
6 import java.awt.event.MouseAdapter;
7 import java.awt.event.MouseListener;
8 import javax.swing.BorderFactory;
```

```

9  import javax.swing.ImageIcon;
10 import javax.swing.JFrame;
11 import javax.swing.JLabel;
12 import javax.swing.JOptionPane;
13 import javax.swing.JPanel;
14 import javax.swing.JScrollPane;
15 import javax.swing.JTree;
16 import javax.swing.SwingUtilities;
17 import javax.swing.event.TreeSelectionEvent;
18 import javax.swing.event.TreeSelectionListener;
19 import javax.swing.tree.DefaultMutableTreeNode;
20 import javax.swing.tree.TreeCellRenderer;
21 import javax.swing.tree.TreePath;
22
23 public class JTreeExample extends JFrame {
24     private JTree jTree;
25
26     //메인 윈도우 설정
27     public JTreeExample() {
28         this.setTitle("JTreeExample");
29         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
30         this.getContentPane().add(new JScrollPane(getJTree()),
31                                     BorderLayout.CENTER);
32         this.setSize(200, 150);
33     }
34
35     //JTree 생성
36     public JTree getJTree() {
37         if (jTree == null) {
38             DefaultMutableTreeNode root = new DefaultMutableTreeNode("그룹리스트");
39
40             DefaultMutableTreeNode node1 = new DefaultMutableTreeNode("친구");
41             node1.add(new DefaultMutableTreeNode("친구1"));
42             node1.add(new DefaultMutableTreeNode("친구2"));
43             root.add(node1);
44
45             jTree = new JTree(root);
46             jTree.setCellRenderer(new MyTreeCellRenderer());
47
48             //이벤트 리스너 추가

```

```

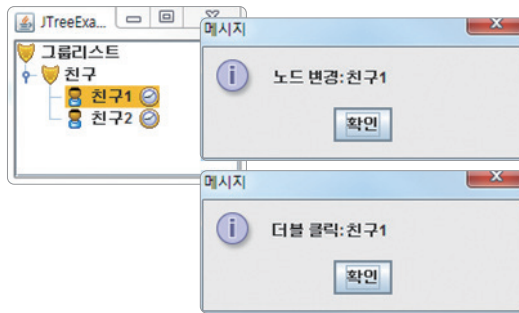
48         jTree.addTreeSelectionListener(treeSelectionListener);
49         jTree.addMouseListener(mouseListener);
50
51     }
52     return jTree;
53 }
54
55 //TreeSelectionListener 필드 선언
56 private TreeSelectionListener treeSelectionListener =
57     new TreeSelectionListener() {
58         @Override
59         public void valueChanged(TreeSelectionEvent e) {
60             TreePath treePath = e.getPath();
61             DefaultMutableTreeNode treeNode =
62                 (DefaultMutableTreeNode) treePath.getLastPathComponent();
63             String nodeText = (String) treeNode.getUserObject();
64             JOptionPane.showMessageDialog(JTreeExample.this, "노드 변경: " +
65                 nodeText);
66         }
67     };
68
69 //MouseListener 필드 선언
70 private MouseListener mouseListener = new MouseAdapter() {
71     public void mouseClicked(java.awt.event.MouseEvent e) {
72         //더블 클릭이 되었을 경우에만 실행
73         if (e.getClickCount() == 2) {
74             TreePath treePath = jTree.getPathForLocation(e.getX(), e.getY());
75             DefaultMutableTreeNode treeNode =
76                 (DefaultMutableTreeNode) treePath.getLastPathComponent();
77             String nodeText = (String) treeNode.getUserObject();
78             JOptionPane.showMessageDialog(JTreeExample.this, "더블 클릭: " +
79                 nodeText);
80         }
81     };
82 };
83
84 //TreeCellRenderer 정의
85 public class MyTreeCellRenderer implements TreeCellRenderer {
86     public Component getTreeCellRendererComponent(
87         JTree tree, Object value, boolean sel, boolean expanded,

```

```

85         boolean leaf, int row, boolean hasFocus) {
86     if (!leaf) {
87         JLabel jLabel = new JLabel();
88         jLabel.setBorder(BorderFactory.createEmptyBorder(5, 0, 5, 0));
89         jLabel.setIcon(new ImageIcon(getClass().getResource
90             ("parentnode.gif")));
91         jLabel.setText(value.toString());
92         return jLabel;
93     } else {
94         JPanel jPanel = new JPanel();
95         jPanel.setBackground(Color.WHITE);
96         jPanel.setLayout(new BorderLayout());
97         jPanel.setBorder(BorderFactory.createEmptyBorder(3, 0, 3, 0));
98
99         JLabel lblWest = new JLabel(new ImageIcon(getClass().
100             getResource("logon.gif")));
101         JLabel lblCenter = new JLabel(" " + value.toString() + " ");
102         JLabel lblEast = new JLabel(new ImageIcon(getClass().
103             getResource("time.gif")));
104
105         jPanel.add(lblWest, BorderLayout.WEST);
106         jPanel.add(lblCenter, BorderLayout.CENTER);
107         jPanel.add(lblEast, BorderLayout.EAST);
108
109         if (sel) {
110             jPanel.setBackground(Color.ORANGE);
111         }
112         return jPanel;
113     }
114 }
115
116 public static void main(String[] args) {
117     SwingUtilities.invokeLater(new Runnable() {
118         public void run() {
119             JTreeExample jFrame = new JTreeExample();
120             jFrame.setVisible(true);
121         }
122     });
123 }

```

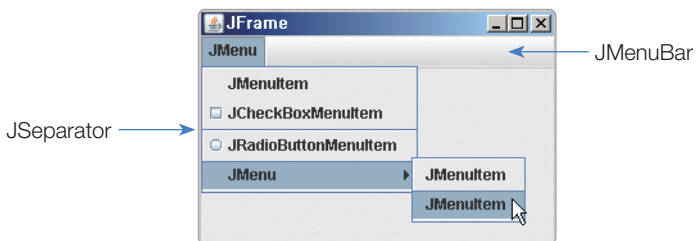


11 메뉴 컴포넌트

UI 프로그램에서 메뉴는 빠질 수 없는 구성 요소이다. Swing은 메뉴 생성을 위해 다음과 같은 컴포넌트를 javax.swing 패키지에서 제공한다.

컴포넌트	설명
JMenuBar	메뉴바 컴포넌트
JMenu	주 메뉴 및 자식 메뉴 아이템을 갖는 서브 메뉴 컴포넌트
JPopupMenu	팝업 메뉴 컴포넌트
JMenuItem	메뉴 아이템 컴포넌트
JCheckBoxMenuItem	JCheckBox로 선택할 수 있는 메뉴 아이템 컴포넌트
JRadioButtonMenuItem	JRadioButton으로 선택할 수 있는 메뉴 아이템 컴포넌트
JSeparator	메뉴를 수직 또는 수평으로 분리시키는 컴포넌트

각 컴포넌트의 사용 위치를 그림으로 표시하면 다음과 같다.



메뉴 생성

메뉴를 생성하기 위해서는 제일 먼저 JMenuBar를 생성하고 윈도우 컨테이너 상단에 배치해야 한다. JMenuBar를 배치할 수 있는 윈도우 컨테이너에는 JFrame, JDialog 등이 있다. 이 컨테이너는 JMenuBar를 상단에 배치하기 위해 setMenuBar() 메소드를 제공하고 있다.

```
JMenuBar jMenuBar = new JMenuBar();
jFrame.setJMenuBar(jMenuBar);
```

JMenuBar는 윈도우 상단에 메뉴를 수평으로 배치하는 역할만 하기 때문에 메뉴가 없을 경우에는 윈도우 상단에 보이지 않는다. JMenu로 생성된 메뉴를 JMenuBar에 다음과 같이 추가해야만 윈도우 상단에 나타난다.

```
JMenu jMenu = new JMenu("메뉴명");
jMenuBar.add(jMenu);
```

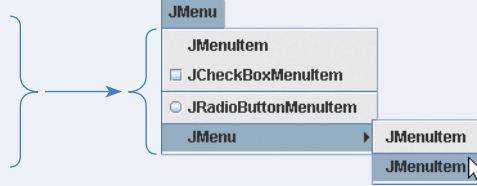
JMenuBar에 추가된 JMenu는 마우스로 클릭했을 때 보여줄 메뉴 아이템과 서브 메뉴를 가져야 한다. 메뉴 아이템은 JMenuItem, JCheckBoxMenuItem, JRadioButtonMenuItem이고, 서브 메뉴는 JMenu를 말한다. 이들은 JMenu의 add() 메소드에 의해 추가된다.

```
//메뉴 아이템 생성
JMenuItem menuItem1 = new JMenuItem("JMenuItem");
JMenuItem menuItem2 = new JCheckBoxMenuItem("JCheckBoxMenuItem");
JMenuItem menuItem3 = new JRadioButtonMenuItem("JRadioButtonMenuItem");

//서브 메뉴 생성
JMenu subMenu = new JMenu("JMenu");
JMenuItem subMenuItem1 = new JMenuItem("JMenuItem");
JMenuItem subMenuItem2 = new JMenuItem("JMenuItem");
jMenuSub.add(subMenuItem1);
jMenuSub.add(subMenuItem1);

//메뉴 아이템과 서브 메뉴 추가
```

```
jMenu.add(menuItem1);
jMenu.add(menuItem2);
jMenu.add(new JSeparator());
jMenu.add(menuItem3);
jMenu.add(subMenu);
```



메뉴 텍스트 앞에 아이콘을 넣고 싶다면 JMenuItem, JCheckBoxMenuItem, JRadioButtonMenuItem 생성자에 ImageIcon 객체를 다음과 같이 추가하면 된다.

```
JMenuItem mi = new JMenuItem("메뉴명",
    new ImageIcon(getClass().getResource("파일명")));
```

다음 예제는 메뉴바, 메뉴, 서브 메뉴, 메뉴 아이템을 생성하고, 추가하는 방법을 보여 준다.

>>> JMenuExample.java

```

1  package sec11.exam01_jmenu;
2
3  import javax.swing.ImageIcon;
4  import javax.swing.JCheckBoxMenuItem;
5  import javax.swing.JFrame;
6  import javax.swing.JMenu;
7  import javax.swing.JMenuBar;
8  import javax.swing.JMenuItem;
9  import javax.swing.JSeparator;
10 import javax.swing.SwingUtilities;
11
12 public class JMenuExample extends JFrame {
13     private JMenuBar jMenuBar;
14     private JMenu menuFile, menuNew, menuHelp;
15     private JMenuItem menuItemNewFile, menuItemNewFolder;
16     private JMenuItem menuItemOpen, menuItemSave, menuItemExit;
17
18     //메인 윈도우 설정
19     public JMenuExample() {
20         this.setTitle("JMenuExample");
```

```

21     this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
22     this.setJMenuBar(getJMenuBar());
23     this.setSize(300, 200);
24 }
25
26 //JMenuBar 생성
27 public JMenuBar getJMenuBar() {
28     if (jMenuBar == null) {
29         jMenuBar = new JMenuBar();
30         //메뉴 추가
31         jMenuBar.add(getMenuFile());
32         jMenuBar.add(getMenuHelp());
33     }
34     return jMenuBar;
35 }
36
37 //파일 메뉴 생성
38 public JMenu getMenuFile() {
39     if (menuFile == null) {
40         menuFile = new JMenu("파일");
41         //서브 메뉴와 메뉴 아이템 추가
42         menuFile.add(getMenuNew());
43         menuFile.add(getMenuItemOpen());
44         menuFile.add(getMenuItemSave());
45         menuFile.add(new JSeparator());
46         menuFile.add(getMenuItemExit());
47     }
48     return menuFile;
49 }
50
51 //도움말 메뉴 생성
52 public JMenu getMenuHelp() {
53     if (menuHelp == null) {
54         menuHelp = new JMenu("도움말");
55     }
56     return menuHelp;
57 }
58
59 //새로 만들기 서브 메뉴 생성
60 public JMenu getMenuNew() {

```

```

61     if (menuNew == null) {
62         menuNew = new JMenu("새로 만들기");
63         //메뉴 아이템 추가
64         menuNew.add(getMenuItemNewFile());
65         menuNew.add(getMenuItemNewFolder());
66     }
67     return menuNew;
68 }
69
70 //새 파일 메뉴 아이템 생성
71 public JMenuItem getMenuItemNewFile() {
72     if (menuItemNewFile == null) {
73         menuItemNewFile = new JMenuItem("새 파일");
74     }
75     return menuItemNewFile;
76 }
77
78 //새 폴더 메뉴 아이템 생성
79 public JMenuItem getMenuItemNewFolder() {
80     if (menuItemNewFolder == null) {
81         menuItemNewFolder = new JMenuItem("새 폴더");
82     }
83     return menuItemNewFolder;
84 }
85
86 //파일 열기 메뉴 아이템 생성
87 public JMenuItem getMenuItemOpen() {
88     if (menuItemOpen == null) {
89         menuItemOpen = new JMenuItem("파일 열기",
90             new ImageIcon(getClass().getResource("open.gif")));
91     }
92     return menuItemOpen;
93 }
94
95 //파일 저장 메뉴 아이템 생성
96 public JMenuItem getMenuItemSave() {
97     if (menuItemSave == null) {
98         menuItemSave = new JCheckBoxMenuItem("파일 저장");
99     }
100    return menuItemSave;

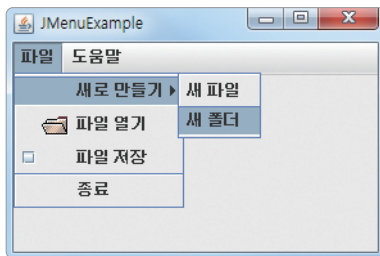
```

```

101     }
102
103     //종료 메뉴 아이템 생성
104     public JMenuItem getMenuExit() {
105         if (menuItemExit == null) {
106             menuItemExit = new JMenuItem("종료");
107         }
108         return menuItemExit;
109     }
110
111     public static void main(String[] args) {
112         SwingUtilities.invokeLater(new Runnable() {
113             public void run() {
114                 JMenuExample jFrame = new JMenuExample();
115                 jFrame.setVisible(true);
116             }
117         });
118     }
119 }

```

실행 결과



이벤트 처리

사용자가 마우스로 메뉴 아이템을 선택하면 `ActionEvent`가 발생하게 된다. 따라서 메뉴 아이템에 `ActionListener`를 등록하여 이벤트를 처리할 수 있다. 메뉴 아이템마다 별도의 `ActionListener`를 추가해도 되지만, 하나만 만들어서 모든 메뉴 아이템의 `ActionEvent`를 처리하는 것이 편리하다.

`ActionEvent`의 `getSource()`는 마우스로 선택한 `JMenuItem`을 리턴하고,

getActionCommand()는 JMenuItem의 텍스트를 리턴한다. 따라서 어떤 JMenuItem을 선택 했는지는 다음과 같이 2가지 방법으로 구분할 수 있다.

```
public class MenuActionListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        //선택된 JMenuItem 객체로 구분
        JMenuItem selected = e.getSource();
        if(selected == menuItem1) { //처리 코드 }
        else if(selected == menuItem2) { //처리 코드 }

        //선택된 JMenuItem의 텍스트로 구분
        String ac = e.getActionCommand();
        if(ac.equals("메뉴 문자열1")) { //처리 코드 }
        else if(ac.equals("메뉴 문자열2")) { //처리 코드 }
    }
}
```

JCheckBoxMenuItem과 JRadioButtonMenuItem의 경우 현재 선택되어 있는지, 아닌지에 따라 처리 코드 내용이 달라질 수 있기 때문에 선택 여부를 알 필요가 있고, 선택 및 해제가 코드로 가능해야 한다.

JCheckBoxMenuItem과 JRadioButtonMenuItem는 모두 isSelected() 메소드로 선택 여부를 알 수 있고, setSelected() 메소드로 선택 및 해제를 할 수 있다. JCheckBoxMenuItem은 추가로 getState()와 setState()를 제공하는데, 이들 메소드를 사용해도 된다.

```
jCheckBoxMenuItem.addActionListener( new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        if(jCheckBoxMenuItem.isSelected()) {
            //선택 상태일 때 실행할 코드
        } else {
            //해제 상태일 때 실행할 코드
        }
    }
}
```

다음 예제는 JCheckBoxMenuItem에서 선택과 해제 상태를 메시지 다이얼로그로 보여 준다.

>>> JMenuExample.java

```
1  package sec11.exam02_eventhandling;
2
3  import java.awt.event.ActionEvent;
4  import java.awt.event.ActionListener;
5  import javax.swing.JCheckBoxMenuItem;
6  import javax.swing.JFrame;
7  import javax.swing.JMenu;
8  import javax.swing.JMenuBar;
9  import javax.swing.JMenuItem;
10 import javax.swing.JOptionPane;
11 import javax.swing.SwingUtilities;
12
13 public class JMenuExample extends JFrame {
14     private JMenuBar jMenuBar;
15     private JMenu menuFile;
16     private JMenuItem menuItemSave, menuItemExit;
17
18     //메인 윈도우 설정
19     public JMenuExample() {
20         this.setTitle("JMenuExample");
21         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
22         this.setJMenuBar(getJMenuBar());
23         this.setSize(300, 200);
24     }
25
26     //JMenuBar 생성
27     public JMenuBar getJMenuBar() {
28         if (jMenuBar == null) {
29             jMenuBar = new JMenuBar();
30             jMenuBar.add(getMenuFile());
31         }
32         return jMenuBar;
33     }
34
35     //파일 메뉴 생성
36     public JMenu getMenuFile() {
37         if (menuFile == null) {
38             menuFile = new JMenu("파일");
```

```

39         menuFile.add(getMenuItemSave());
40         menuFile.add(getMenuItemExit());
41     }
42     return menuFile;
43 }
44
45 //파일 저장 메뉴 아이템 생성
46 public JMenuItem getMenuItemSave() {
47     if (menuItemSave == null) {
48         menuItemSave = new JCheckBoxMenuItem("파일 저장");
49         menuItemSave.addActionListener(actionListener);
50     }
51     return menuItemSave;
52 }
53
54 //종료 메뉴 아이템 생성
55 public JMenuItem getMenuItemExit() {
56     if (menuItemExit == null) {
57         menuItemExit = new JMenuItem("종료");
58         menuItemExit.addActionListener(actionListener);
59     }
60     return menuItemExit;
61 }
62
63 //메뉴 선택 이벤트를 처리하는 ActionListener 필드 선언
64 private ActionListener actionListener = new ActionListener() {
65     @Override
66     public void actionPerformed(ActionEvent e) {
67         if (e.getSource() == menuItemSave) {
68             if (menuItemSave.isSelected()) {
69                 JOptionPane.showMessageDialog(
70                     JMenuExample.this, "해제 상태 >> 체크 상태.");
71             } else {
72                 JOptionPane.showMessageDialog(
73                     JMenuExample.this, "체크 상태 >> 해제 상태.");
74             }
75         } else if (e.getSource() == menuItemExit) {
76             System.exit(0);
77         }
78     }

```

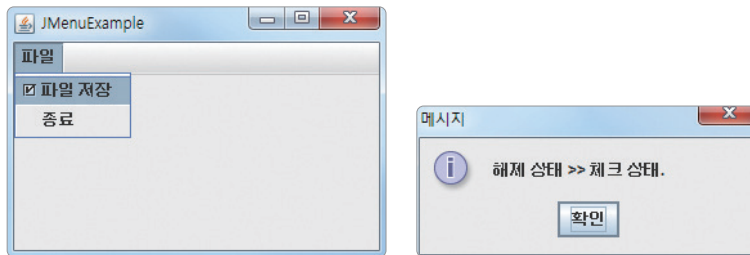


```

79     };
80
81     public static void main(String[] args) {
82         SwingUtilities.invokeLater(new Runnable() {
83             public void run() {
84                 JMenuExample jFrame = new JMenuExample();
85                 jFrame.setVisible(true);
86             }
87         });
88     }
89 }

```

실행 결과

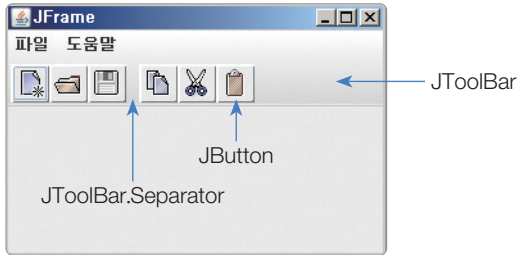


12 툴바 컴포넌트

툴바는 메뉴바 아래에 위치하고 주로 버튼들이 배치되는 컨테이너이다. 메뉴보다는 빠르게 원하는 기능을 마우스로 선택할 수 있기 때문에 사용자들은 메뉴보다는 툴바를 더 선호한다. Swing은 툴바 생성을 위해서 다음과 같은 컴포넌트를 제공하고 있다.

컴포넌트	설명
JToolBar	툴바 생성을 위한 컴포넌트
JToolBar.Separator	버튼들을 그룹 짓기 위해 공백을 제공하는 컴포넌트

각 컴포넌트의 위치를 그림으로 표시하면 다음과 같다.



최상위 레벨 컨테이너에서 JToolBar를 상단에 배치하는 방법은 JToolBar를 생성한 뒤 BorderLayout을 이용해서 북쪽에 배치하면 된다. JToolBar 내부에 컴포넌트가 추가되지 않으면 JToolBar가 북쪽에 위치하더라도 보이지 않는다.

```
JToolBar jToolBar = new JToolBar();
jFrame.getContentPane().add(jToolBar, BorderLayout.NORTH);
```

프로그램 실행 중에는 JToolBar의 요철 부분을 마우스로 끌어서 컨테이너의 동·서·남·북에 붙일 수 있다. 또한 JToolBar를 중앙에 끌어 놓으면 툴바 윈도우가 만들어진다. 이러한 JToolBar의 끌기 기능을 없애려면 setFloatable(false)를 호출하면 된다.

```
jToolBar.setFloatable(false)
```

JToolBar 내부 컴포넌트는 일반적으로 JButton으로 구성되나 다른 컴포넌트도 추가 가능하다. 다음은 JButton을 JToolBar에 추가하는 코드이다.

```
JButton jButton = new JButton();
jButton.setIcon(new ImageIcon(getClass().getResource("image.gif"))); //아이콘
jButton.setBorder(new SoftBevelBorder(SoftBevelBorder.RAISED)); //돌출된 모양
jButton.setToolTipText("새로만들기"); //마우스를 올려놓았을 때 나오는 풍선 도움말
jToolBar.add(jButton);
```

다음 예제는 툴바 버튼을 클릭하면 메시지 다이얼로그로 어떤 버튼이 클릭되었는지 알려준다.

>>> JToolBarExample.java

```
1  package sec12.exam01_jtoolbar;
2
3  import java.awt.BorderLayout;
4  import java.awt.event.ActionEvent;
5  import java.awt.event.ActionListener;
6  import javax.swing.ImageIcon;
7  import javax.swing.JButton;
8  import javax.swing.JFrame;
9  import javax.swing.JMenu;
10 import javax.swing.JMenuBar;
11 import javax.swing.JOptionPane;
12 import javax.swing.JToolBar;
13 import javax.swing.SwingUtilities;
14 import javax.swing.border.SoftBevelBorder;
15
16 public class JToolBarExample extends JFrame {
17     private JMenuBar jMenuBar;
18     private JToolBar jToolBar;
19     private JButton btnNew, btnSave, btnCopy, btnPaste;
20
21     //메인 윈도우 설정
22     public JToolBarExample() {
23         this.setTitle("JFrame");
24         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
25         this.setJMenuBar(getJMenuBar());
26
27         //JToolBar를 JFrame의 북쪽에 배치
28         this.getContentPane().add(getJToolBar(), BorderLayout.NORTH);
29
30         this.setSize(300, 200);
31     }
32
33     //JMenuBar 생성
34     public JMenuBar getJMenuBar() {
35         if (jMenuBar == null) {
36             jMenuBar = new JMenuBar();
37             jMenuBar.add(new JMenu("파일"));
38             jMenuBar.add(new JMenu("도움말"));
```

```

39     }
40     return jMenuBar;
41 }
42
43 //JToolBar 생성
44 public JToolBar getJToolBar() {
45     if (jToolBar == null) {
46         jToolBar = new JToolBar();
47         jToolBar.add(getBtnNew());
48         jToolBar.add(getBtnSave());
49         jToolBar.addSeparator();
50         jToolBar.add(getBtnCopy());
51         jToolBar.add(getBtnPaste());
52     }
53     return jToolBar;
54 }
55
56 //JToolBar의 New 버튼 생성
57 public JButton getBtnNew() {
58     if (btnNew == null) {
59         btnNew = new JButton();
60         btnNew.setIcon(new ImageIcon(getClass().getResource("new.gif")));
61         btnNew.setBorder(new SoftBevelBorder(SoftBevelBorder.RAISED));
62         btnNew.setToolTipText("새로만들기");
63         btnNew.addActionListener(actionListener);
64     }
65     return btnNew;
66 }
67
68 //JToolBar의 Save 버튼 생성
69 public JButton getBtnSave() {
70     if (btnSave == null) {
71         btnSave = new JButton();
72         btnSave.setIcon(new ImageIcon(getClass().getResource("save.gif")));
73         btnSave.setBorder(new SoftBevelBorder(SoftBevelBorder.RAISED));
74         btnSave.setToolTipText("저장");
75         btnSave.addActionListener(actionListener);
76     }
77     return btnSave;
78 }

```

```

79
80 //JToolBar의 Copy 버튼 생성
81 public JButton getBtnCopy() {
82     if (btnCopy == null) {
83         btnCopy = new JButton();
84         btnCopy.setIcon(new ImageIcon(getClass().getResource("copy.gif")));
85         btnCopy.setBorder(new SoftBevelBorder(SoftBevelBorder.RAISED));
86         btnCopy.setToolTipText("복사");
87         btnCopy.addActionListener(actionListener);
88     }
89     return btnCopy;
90 }
91
92 //JToolBar의 Paste 버튼 생성
93 public JButton getBtnPaste() {
94     if (btnPaste == null) {
95         btnPaste = new JButton();
96         btnPaste.setIcon(new ImageIcon(getClass().getResource("paste.gif")));
97         btnPaste.setBorder(new SoftBevelBorder(SoftBevelBorder.RAISED));
98         btnPaste.setToolTipText("붙여넣기");
99         btnPaste.addActionListener(actionListener);
100     }
101     return btnPaste;
102 }
103
104 //JToolBar의 버튼 이벤트 처리
105 private ActionListener actionListener = new ActionListener() {
106     @Override
107     public void actionPerformed(ActionEvent e) {
108         if (e.getSource() == btnNew) {
109             JOptionPane.showMessageDialog(JToolBarExample.this,
110                 "[새로만들기]를 클릭");
111         } else if (e.getSource() == btnSave) {
112             JOptionPane.showMessageDialog(JToolBarExample.this, "[저장]를 클릭");
113         } else if (e.getSource() == btnCopy) {
114             JOptionPane.showMessageDialog(JToolBarExample.this, "[복사]를 클릭");
115         } else if (e.getSource() == btnPaste) {
116             JOptionPane.showMessageDialog(JToolBarExample.this, "[붙여넣기]를 클릭");
117         }
118     }
119 }

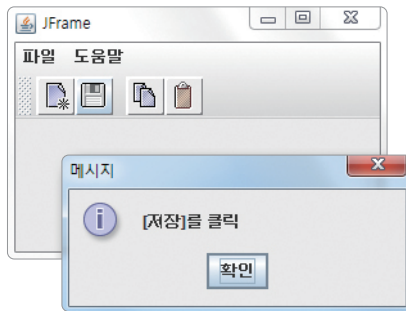
```

```

118     };
119
120     public static void main(String[] args) {
121         SwingUtilities.invokeLater(new Runnable() {
122             public void run() {
123                 JToolBarExample JFrame = new JToolBarExample();
124                 JFrame.setVisible(true);
125             }
126         });
127     }
128 }

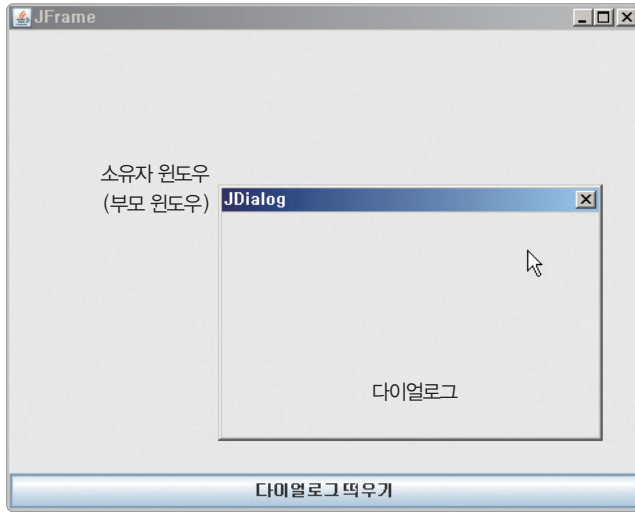
```

실행 결과



13 다이얼로그

다이얼로그(Dialog)는 주 윈도우에서 사용자의 선택 또는 입력을 위해서 띄우는 서브 윈도우이다. 다이얼로그를 띄우는 주 윈도우를 부모(parent) 윈도우 또는 소유자(owner) 윈도우라고 한다.



다이얼로그는 모달(modal)과 모달리스(modalless) 두 가지 종류가 있다. 모달 다이얼로그는 다이얼로그를 닫기 전까지 부모 윈도우를 사용할 수 없는 다이얼로그를 말한다. 모달리스 다이얼로그는 부모 윈도우를 계속 사용할 수 있는 다이얼로그를 말한다. 다이얼로그의 기본은 모달이다.

Swing은 사용자 정의 다이얼로그를 생성하기 위해서 JDialog를 제공한다. 또한 프로그램에서 자주 사용되는 표준화된 다이얼로그도 제공한다.

다이얼로그를 띄울 때는 소유자 윈도우가 반드시 필요하다. 소유자 윈도우는 JWindow, JFrame 등이 될 수 있고, JDialog에서 또 다른 JDialog를 생성할 수 있기 때문에 JDialog도 소유자 윈도우가 될 수 있다.

사용자 정의 다이얼로그

사용자 정의 다이얼로그를 만들기 위해서는 JDialog를 상속해야 한다. 생성자는 소유자 윈도우를 받도록 선언해야 하며 소유자 윈도우를 JDialog 생성자에게 넘겨주기 위해 super(owner)를 호출해야 한다.

```

public class MyDialog extends JDialog {
    public MyDialog(JFrame owner) {
        super(owner);
        setTitle("Title");           //다이얼로그 제목
        setSize(width, height);      //다이얼로그의 크기
        setModal(true);              //모달 다이얼로그(기본)
        setResizable(false);         //다이얼로그의 크기를 변경하지 않도록 설정

        //우측 상단 종료 버튼 이벤트 처리
        setDefaultCloseOperation(JDialog.DISPOSE_ON_CLOSE);

        //다이얼로그의 위치를 소유자 윈도우의 정중앙으로 설정
        setLocation(
            owner.getLocationOnScreen().x + (owner.getWidth()-getWidth())/2,
            owner.getLocationOnScreen().y + (owner.getHeight()-getHeight())/2
        );
    }
}

```

JDialog는 JFrame과 마찬가지로 우측 상단에 종료 버튼 [×]를 제공하는데, 기본 기능은 JDialog를 화면에서 숨긴다. JDialog를 완전히 제거하고 싶다면 setDefaultCloseOperation(JDialog.DISPOSE_ON_CLOSE)를 호출하면 된다. 종료 버튼에 적용할 수 있는 기능은 다음 세 가지이다.

기능별 상수	설명
WindowConstants.DO_NOTHING_ON_CLOSE	아무 것도 하지 않음
WindowConstants.HIDE_ON_CLOSE	화면에서 JDialog 숨김(기본)
WindowConstants.DISPOSE_ON_CLOSE	화면에서 JDialog 완전히 제거

보통 다이얼로그는 소유자 윈도우 중앙에 나타나므로 다이얼로그의 좌상단 좌표는 소유자 윈도우의 위치와 크기 정보를 이용해서 다음과 같이 구할 수 있다.

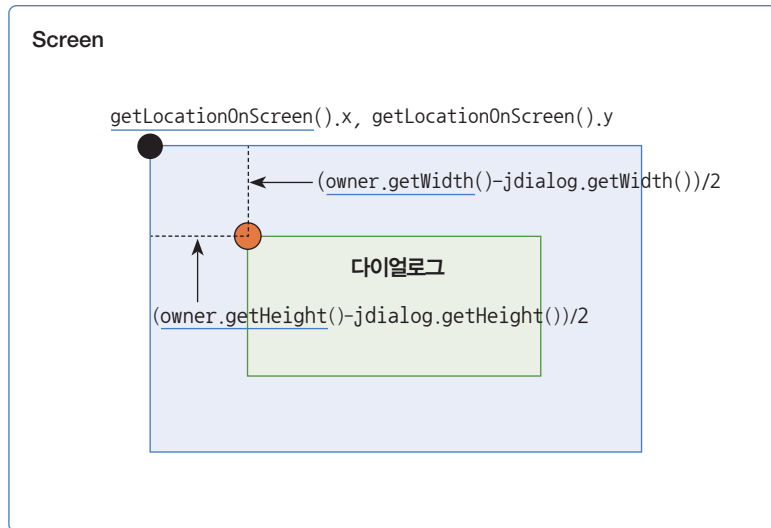
```

int left = owner.getLocationOnScreen().x +
            (owner.getWidth()-dialog.getWidth())/2;
int top = owner.getLocationOnScreen().y +
            (owner.getHeight()-dialog.getHeight())/2;

```


이렇게 구한 다이얼로그의 좌상단 좌표는 JDialog의 setLocation() 메소드를 호출할 때 매개값으로 전달된다.

```
setLocation( left, top );
```



사용자 정의 다이얼로그를 화면에 띄우려면 소유자 윈도우를 생성자 매개값으로 제공하고, 객체를 생성한 다음 setVisible(true) 메소드를 호출하면 된다.

```
MyDialog jDialog = new MyDialog(jFrame);  
jDialog.setVisible(true);
```

JDialog는 기본적으로 BorderLayout 배치 관리자를 사용하며, 언제든지 배치 관리자를 변경할 수 있다. 내부 컴포넌트는 JFrame과 동일하게 getContentPane()을 얻어 배치해야 한다.

다음 예제는 메인 윈도우에서 [다이얼로그 띄우기] 버튼을 클릭하면 모달 다이얼로그가 나타나고, [닫기] 버튼을 클릭하면 모달 다이얼로그가 사라지도록 한다.

>>> JDialogExample.java

```
1  package sec13.exam01_jdialog;
2
3  import java.awt.BorderLayout;
4  import java.awt.event.ActionEvent;
5  import java.awt.event.ActionListener;
6  import javax.swing.JButton;
7  import javax.swing.JDialog;
8  import javax.swing.JFrame;
9  import javax.swing.WindowConstants;
10
11 public class JDialogExample extends JDialog {
12     private JButton btnClose;
13
14     //다이얼로그 설정
15     public JDialogExample(JFrame owner) {
16         super(owner);
17         this.setTitle("JDialogExample");
18         this.setSize(300, 200);
19         this.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
20         this.setResizable(false);
21         this.setModal(true);
22         this.setLocation(
23             owner.getLocationOnScreen().x + (owner.getWidth() - this.getWidth()) / 2,
24             owner.getLocationOnScreen().y + (owner.getHeight() - this.getHeight()) / 2
25         );
26         this.getContentPane().add(getBtnClose(), BorderLayout.SOUTH);
27     }
28
29     //남쪽 버튼 생성
30     public JButton getBtnClose() {
31         if (btnClose == null) {
32             btnClose = new JButton();
33             btnClose.setText("닫기");
34             btnClose.addActionListener(new ActionListener() {
35                 @Override
36                 public void actionPerformed(ActionEvent e) {
37                     JDialogExample.this.dispose();
38                 }
39             });
39         }
40     }
41 }
```

```

39         });
40     }
41     return btnClose;
42 }
43 }

```

>>> JFrameExample.java

```

1  package sec13.exam01_jdialog;
2
3  import java.awt.BorderLayout;
4  import javax.swing.JButton;
5  import javax.swing.JFrame;
6  import javax.swing.SwingUtilities;
7
8  public class JFrameExample extends JFrame {
9      private JButton btnOpenDialog;
10
11      //메인 윈도우 설정
12      public JFrameExample() {
13          this.setTitle("JFrame");
14          this.setSize(500, 400);
15          this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
16          this.getContentPane().add(getBtnOpenDialog(), BorderLayout.SOUTH);
17      }
18
19      //남쪽 버튼 생성
20      private JButton getBtnOpenDialog() {
21          if (btnOpenDialog == null) {
22              btnOpenDialog = new JButton();
23              btnOpenDialog.setText("다이얼로그 띄우기");
24              btnOpenDialog.addActionListener(new java.awt.event.ActionListener() {
25                  public void actionPerformed(java.awt.event.ActionEvent e) {
26                      //다이얼로그 띄우기
27                      JDialogExample jDialog = new JDialogExample(JFrameExample.this);
28                      jDialog.setVisible(true);
29                  }
30              });

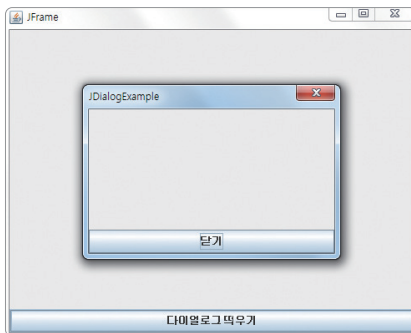
```

```

31     }
32     return btnOpenDialog;
33 }
34
35 public static void main(String[] args) {
36     SwingUtilities.invokeLater(new Runnable() {
37         public void run() {
38             JFrameExample jFrame = new JFrameExample();
39             jFrame.setVisible(true);
40         }
41     });
42 }
43 }

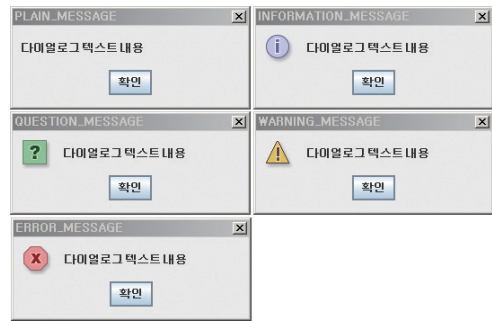
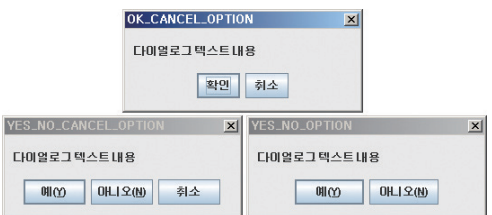

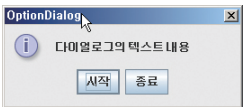
```

실행 결과



표준화된 다이얼로그

Swing은 자주 사용되는 표준화된 다이얼로그(MessageDialog, ConfirmDialog, InputDialog, OptionDialog)를 쉽게 생성하기 위해 JOptionPane 클래스를 제공한다. JOptionPane의 showXXXDialog() 메소드를 호출하면 XXX 다이얼로그가 생성된다.

메소드	다이얼로그 내용	다이얼로그 모양
showMessageDialog(...)	텍스트 메시지 [확인]	
showConfirmDialog(...)	텍스트 메시지 [확인] [취소] [예] [아니오] [취소] [예] [아니오]	
showInputDialog(...)	입력 컴포넌트 [확인] [취소]	
showOptionDialog(...)	텍스트 메시지 [사용자 정의 버튼]	

showXXXDialog() 메소드는 다음과 같은 매개변수를 가지는데, 이들 매개변수로 다이얼로그의 모양을 결정지을 수 있다.

매개변수	설명	값의 종류
Component parentComponent	소유자 윈도우 또는 null (소유자 윈도우의 중앙에 위치, null을 주면 스크린 중앙에 위치)	JWindow, JFrame, JDialog
Object message	텍스트 메시지	String
String title	다이얼로그 제목	String

매개변수	설명	값의 종류
<code>int</code> optionType	버튼 집합 결정 JOptionPane의 상수 이용	OK_CANCEL_OPTION YES_NO_OPTION YES_NO_CANCEL_OPTION
<code>int</code> messageType	표준 아이콘 설정 JOptionPane의 상수 이용	PLAIN_MESSAGE INFORMATION_MESSAGE QUESTION_MESSAGE WARNING_MESSAGE ERROR_MESSAGE
<code>Icon</code> icon	사용자 정의 아이콘 설정	ImageIcon
<code>Object[]</code> selectionValues	InputDialog에서 사용됨. 항목이 12개 미만이면 JComboBox가 사용됨. 항목이 12개 이상이면 JList가 사용됨.	String[] Icon[]
<code>Object</code> initialSelectionValue	InputDialog에서 초기 선택값	String, Icon
<code>Object[]</code> options	OptionDialog의 버튼들에 표시되는 텍스트 또는 이미지	String[] Icon[]
<code>Object</code> initialValue	초기 포커스를 갖는 버튼의 텍스트 또는 이미지	String Icon

showMessageDialog() 메소드를 제외하고 나머지 세 개의 showXXXDialog() 메소드는 모두 리턴값이 있다. 리턴값은 어떤 버튼이 클릭되었는지, 입력된 값이 무엇인지에 대한 정보를 담고 있다.

- ① ConfirmDialog는 어떤 버튼이 클릭되었는가가 중요하기 때문에 showConfirmDialog() 메소드는 클릭된 버튼에 해당하는 `int` 타입의 JOptionPane 상수를 리턴한다.

JOptionPane 상수	설명
JOptionPane.OK_OPTION	[확인] 버튼을 눌렀을 때
JOptionPane.CANCEL_OPTION	[취소] 버튼을 눌렀을 때
JOptionPane.YES_OPTION	[예] 버튼을 눌렀을 때
JOptionPane.NO_OPTION	[아니오] 버튼을 눌렀을 때
JOptionPane.CLOSED_OPTION	우측 상단의 [×] 버튼을 눌렀을 때

- ② InputDialog는 입력된 값이 중요하기 때문에 showInputDialog() 메소드는 입력 컴포넌트가 JTextField이면 String을 리턴하고, JList나 JComboBox라면 선택된 Object를 리턴한다.
- ③ OptionDialog는 어떤 버튼이 클릭되었는지가 중요하기 때문에 showOptionDialog() 메소드는 매개값으로 주어진 options 배열에서 클릭된 버튼의 인덱스를 리턴한다.

다음 예제는 표준 다이얼로그들을 생성하는 방법과 showXXXDialog() 메소드의 리턴값을 이용하는 방법을 보여 준다.

>> JOptionPaneExample.java

```
1  package sec13.exam02_joptionpane;
2
3  import java.awt.GridLayout;
4  import java.awt.event.ActionEvent;
5  import java.awt.event.ActionListener;
6  import javax.swing.JButton;
7  import javax.swing.JFrame;
8  import javax.swing.JOptionPane;
9  import javax.swing.SwingUtilities;
10
11 public class JOptionPaneExample extends JFrame {
12     private JButton btnMessage, btnConfirm;
13     private JButton btnInput, btnOption;
14
15     //메인 윈도우 설정
16     public JOptionPaneExample() {
17         this.setTitle("JOptionPaneExample");
18         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
19         this.getContentPane().setLayout(new GridLayout(4,1));
20         this.getContentPane().add(getBtnMessage());
21         this.getContentPane().add(getBtnConfirm());
22         this.getContentPane().add(getBtnInput());
23         this.getContentPane().add(getBtnOption());
24         this.setSize(500, 300);
25     }
26 }
```

```

27     //MessageDialog를 보여주는 버튼 생성
28     public JButton getBtnMessage() {
29         if(btnMessage == null) {
30             btnMessage = new JButton();
31             btnMessage.setText("MessageDialog");
32             btnMessage.addActionListener(new ActionListener() {
33                 public void actionPerformed(ActionEvent e) {
34                     //MessageDialog를 보여줌
35                     JOptionPane.showMessageDialog(
36                         JOptionPaneExample.this,
37                         "다이얼로그 텍스트 내용",
38                         "INFORMATION_MESSAGE",
39                         JOptionPane.INFORMATION_MESSAGE);
40                 }
41             });
42         }
43         return btnMessage;
44     }
45
46     //ConfirmDialog를 보여주는 버튼 생성
47     public JButton getBtnConfirm() {
48         if(btnConfirm == null) {
49             btnConfirm = new JButton();
50             btnConfirm.setText("ConfirmDialog");
51             btnConfirm.addActionListener(new ActionListener() {
52                 public void actionPerformed(ActionEvent e) {
53                     //ConfirmDialog를 보여줌
54                     int option = JOptionPane.showConfirmDialog(
55                         JOptionPaneExample.this,
56                         "다이얼로그 텍스트 내용",
57                         "OK_CANCEL_OPTION",
58                         JOptionPane.OK_CANCEL_OPTION,
59                         JOptionPane.PLAIN_MESSAGE,
60                         null);
61                     //클릭된 버튼 확인하기
62                     if(option == JOptionPane.OK_OPTION) {
63                         System.out.println("확인 버튼을 눌렀군요");
64                     } else if(option == JOptionPane.CANCEL_OPTION) {
65                         System.out.println("취소 버튼을 눌렀군요");
66                     } else if(option == JOptionPane.CLOSED_OPTION) {

```



```

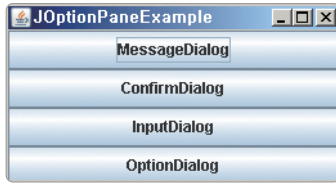
67         System.out.println("닫기 버튼을 눌렀군요");
68     }
69 }
70 });
71 }
72     return btnConfirm;
73 }
74
75 //InputDialog를 보여주는 버튼 생성
76 public JButton getBtnInput() {
77     if(btnInput == null) {
78         btnInput = new JButton();
79         btnInput.setText("InputDialog");
80         btnInput.addActionListener(new ActionListener() {
81             public void actionPerformed(ActionEvent e) {
82                 String input = null;
83
84                 //텍스트 입력을 받는 InputDialog를 보여줌
85                 input = JOptionPane.showInputDialog(
86                     JOptionPaneExample.this,
87                     "다이얼로그 텍스트 내용",
88                     "InputDialog",
89                     JOptionPane.INFORMATION_MESSAGE);
90                 //선택된 항목을 출력
91                 System.out.println("입력된 텍스트: " + input);
92
93                 //JComboBox로 항목을 선택받는 InputDialog를 보여줌
94                 input = (String) JOptionPane.showInputDialog(
95                     JOptionPaneExample.this,
96                     "다이얼로그 텍스트 내용",
97                     "InputDialog",
98                     JOptionPane.PLAIN_MESSAGE,
99                     null,
100                     new String[] {"Java", "JDBC", "JSP", "Spring"},
101                     "JDBC");
102                 //선택된 항목을 출력
103                 System.out.println("선택된 항목: " + input);
104             }
105         });
106     }

```

```

107         return btnInput;
108     }
109
110     //OptionDialog를 보여주는 버튼 생성
111     public JButton getBtnOption() {
112         if(btnOption == null) {
113             btnOption = new JButton();
114             btnOption.setText("OptionDialog");
115             btnOption.addActionListener(new ActionListener() {
116                 public void actionPerformed(ActionEvent e) {
117                     //OptionDialog를 보여줌
118                     int option = JOptionPane.showOptionDialog(
119                         JOptionPaneExample.this,
120                         "다이얼로그의 텍스트 내용",
121                         "OptionDialog",
122                         JOptionPane.YES_NO_OPTION,
123                         JOptionPane.INFORMATION_MESSAGE,
124                         null,
125                         new String[] {"시작", "종료"},
126                         "시작");
127                     //클릭된 버튼 확인하기
128                     if(option == 0) {
129                         System.out.println("시작 버튼을 눌렀군요");
130                     } else if(option == 1) {
131                         System.out.println("종료 버튼을 눌렀군요");
132                     }
133                 }
134             });
135         }
136         return btnOption;
137     }
138
139     public static void main(String[] args) {
140         SwingUtilities.invokeLater(new Runnable() {
141             public void run() {
142                 JOptionPaneExample jFrame = new JOptionPaneExample();
143                 jFrame.setVisible(true);
144             }
145         });
146     }
147 }

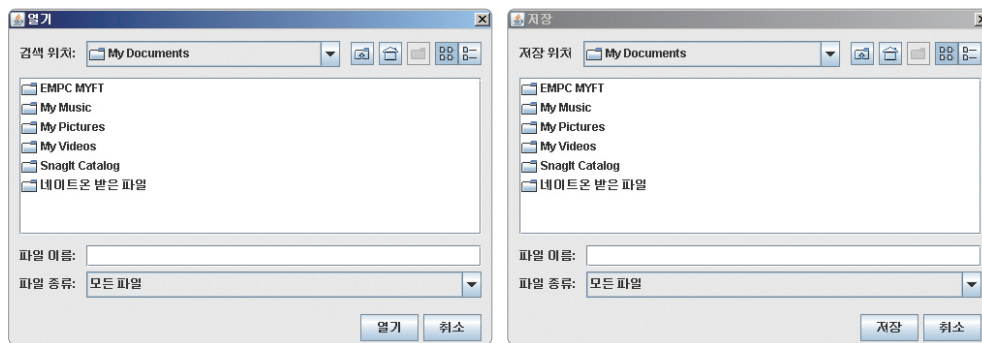
```



파일 다이얼로그

UI 프로그램에서 파일 다이얼로그는 사용자가 파일을 선택할 때 사용한다. Swing은 파일 다이얼로그를 위해 `JFileChooser`를 제공하는데, `showOpenDialog()`와 `showSaveDialog()` 메소드를 이용해서 열기와 저장용 파일 다이얼로그를 띄울 수 있다. `showOpenDialog()`와 `showSaveDialog()`의 매개값은 소유자 윈도우이다.

```
JFileChooser jFileChooser = new JFileChooser();
jFileChooser.showOpenDialog(jFrame); 또는 jFileChooser.showSaveDialog(jFrame);
```



파일 다이얼로그의 기본 디렉토리 위치는 사용자의 홈 디렉토리인데, 변경하고 싶다면 `setCurrentDirectory()` 메소드를 이용할 수 있다.

```
jFileChooser.setCurrentDirectory(new File("C:/Temp"));
```

파일 종류는 JComboBox로 선택할 수 있는데, 기본적으로 ‘모든 파일’ 항목만 있다. 파일 종류는 `FileNameExtensionFilter` 객체를 생성하고, `addChoosableFileFilter()` 메소드로 추가할 수 있다.

```
jFileChooser.addChoosableFileFilter(
    new FileNameExtensionFilter("텍스트 파일(*.txt)", "txt"));
```

`showOpenDialog()`와 `showSaveDialog()` 메소드는 모두 `int` 값을 리턴한다. 어떤 버튼이 눌러졌는지 구분할 용도로 사용되는데, 다음 상수값 중 하나를 리턴한다.

옵션 상수	설명
<code>APPROVE_OPTION</code>	[열기], [저장] 버튼을 클릭했을 때
<code>CANCEL_OPTION</code>	[취소], [x] 버튼을 클릭했을 때

`JFileChooser.APPROVE_OPTION`을 리턴했다면 사용자가 파일을 선택하고, [열기] 또는 [저장] 버튼을 클릭한 경우이다.

```
int option = jFileChooser.showOpenDialog(jFrame);
if(option == JFileChooser.APPROVE_OPTION) {
    //[열기] 버튼을 눌렀을 경우 실행할 코드
} else {
    //[취소], [닫기] 버튼을 눌렀을 경우 실행할 코드
}
```

사용자가 선택한 파일은 `JFileChooser`의 `getSelectedFile()` 메소드로 알 수 있는데, 리턴값은 파일 경로 정보가 있는 `java.io.File` 객체이다.

```
File file = jFileChooser.getSelectedFile();
System.out.println("선택한 파일 절대 경로: " + file.getAbsolutePath());
System.out.println("선택한 파일 이름: " + file.getName());
```

다음 예제는 파일 종류를 추가한 파일 다이얼로그를 보여주고, 어떤 버튼을 선택했는지에 따라 선택한 파일의 경로를 출력한다.

>>> JFileChooserExample.java

```
1  package sec13.exam03_jfilechooser;
2
3  import java.awt.GridLayout;
4  import java.awt.event.ActionEvent;
5  import java.awt.event.ActionListener;
6  import java.io.File;
7  import javax.swing.JButton;
8  import javax.swing.JFileChooser;
9  import javax.swing.JFrame;
10 import javax.swing.SwingUtilities;
11 import javax.swing.filechooser.FileNameExtensionFilter;
12
13 public class JFileChooserExample extends JFrame {
14     private JButton btnFileOpen, btnFileSave;
15
16     //메인 윈도우 설정
17     public JFileChooserExample() {
18         this.setTitle("JFileChooserExample");
19         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
20         this.getContentPane().setLayout(new GridLayout(2, 1));
21         this.getContentPane().add(getBtnFileOpen());
22         this.getContentPane().add(getBtnFileSave());
23         this.setSize(150, 100);
24     }
25
26     //열기 버튼 생성
27     public JButton getBtnFileOpen() {
28         if (btnFileOpen == null) {
29             btnFileOpen = new JButton();
30             btnFileOpen.setText("File Open");
31             btnFileOpen.addActionListener(new ActionListener() {
32                 public void actionPerformed(ActionEvent e) {
33                     //파일 열기 다이얼로그 보여주기
34                     JFileChooser jFileChooser = new JFileChooser();
35                     jFileChooser.addChoosableFileFilter(new FileNameExtensionFilter(
```

```

36         "그림파일(*.jpg, *.gif, *.bmp)", "jpg", "gif", "bmp"));
37     jFileChooser.addChoosableFileFilter(new FileNameExtensionFilter(
38         "텍스트 파일(*.txt)", "txt"));
39     int option = jFileChooser.showOpenDialog(JFileChooserExample.this);
40     //어떤 버튼을 클릭했는지 확인하기
41     if (option == JFileChooser.APPROVE_OPTION) {
42         File file = jFileChooser.getSelectedFile();
43         System.out.println("열어야 할 파일 절대경로: " +
44             file.getAbsolutePath());
45         System.out.println("열어야 할 파일 이름: " + file.getName());
46     } else if (option == JFileChooser.CANCEL_OPTION) {
47         System.out.println("취소 또는 닫기를 눌렀군요");
48     }
49     });
50 }
51 return btnFileOpen;
52 }
53
54 //저장 버튼 생성
55 public JButton getBtnFileSave() {
56     if (btnFileSave == null) {
57         btnFileSave = new JButton();
58         btnFileSave.setText("File Save");
59         btnFileSave.addActionListener(new ActionListener() {
60             public void actionPerformed(ActionEvent e) {
61                 //파일 저장 다이얼로그 보여주기
62                 JFileChooser jFileChooser = new JFileChooser();
63                 jFileChooser.addChoosableFileFilter(new FileNameExtensionFilter(
64                     "그림 파일(*.jpg, *.gif, *.bmp)", "jpg", "gif", "bmp"));
65                 jFileChooser.addChoosableFileFilter(new FileNameExtensionFilter(
66                     "텍스트 파일(*.txt)", "txt"));
67                 int option = jFileChooser.showSaveDialog(JFileChooserExample.this);
68                 //어떤 버튼을 클릭했는지 확인하기
69                 if (option == JFileChooser.APPROVE_OPTION) {
70                     File file = jFileChooser.getSelectedFile();
71                     System.out.println("저장할 파일: " + file.getAbsolutePath());
72                 } else if (option == JFileChooser.CANCEL_OPTION) {
73                     System.out.println("취소 또는 닫기를 눌렀군요");
74                 }

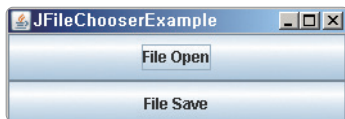
```

```

75         }
76     });
77 }
78     return btnFileSave;
79 }
80
81     public static void main(String[] args) {
82         SwingUtilities.invokeLater(new Runnable() {
83             public void run() {
84                 JFileChooserExample jFrame = new JFileChooserExample();
85                 jFrame.setVisible(true);
86             }
87         });
88     }
89 }

```

실행 결과



색상 다이얼로그

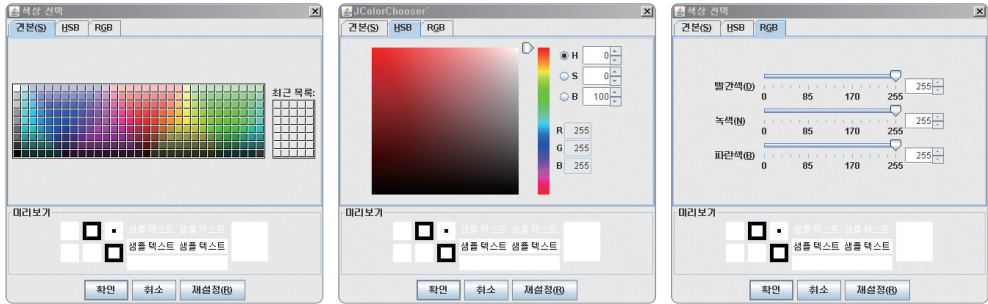
색상 다이얼로그를 이용하면 색상을 사용자가 선택할 수 있다. Swing은 색상 다이얼로그를 위해 `JColorChooser`를 제공하는데, `showDialog()` 메소드를 이용해서 색상 다이얼로그를 띄울 수 있다.

```

JColorChooser jColorChooser = new JColorChooser();
Color color = jColorChooser.showDialog(jFrame, "색상 선택", null);

```

첫 번째 매개값은 소유자 윈도우이고, 두 번째 매개값은 색상 다이얼로그의 제목이다. 세 번째 매개값은 미리보기 색상에 해당하는 `Color` 객체를 주면 되는데, 만약 `null`을 주면 미리보기 색상은 하얀색이 된다. `JColorChooser`는 색상을 선택할 수 있도록 여러 가지 탭을 제공하고 있다.



색상을 결정하고 나서 [확인] 버튼을 클릭하면 사용자가 선택한 색상은 Color 객체로 생성되고, showDialog() 메소드의 리턴값으로 나온다.

다음 예제는 색상 다이얼로그에서 선택한 색상을 버튼의 배경색으로 설정한다.

>> JColorChooserExample.java

```

1  package sec13.exam04_jcolorchooser;
2
3  import java.awt.Color;
4  import java.awt.GridLayout;
5  import java.awt.event.ActionEvent;
6  import java.awt.event.ActionListener;
7  import javax.swing.JButton;
8  import javax.swing.JColorChooser;
9  import javax.swing.JFrame;
10 import javax.swing.SwingUtilities;
11
12 public class JColorChooserExample extends JFrame {
13     private JButton btnColor;
14
15     //메인 윈도우 설정
16     public JColorChooserExample() {
17         this.setTitle("JColorChooserExample");
18         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
19         this.getContentPane().setLayout(new GridLayout(1, 1));
20         this.getContentPane().add(getBtnColor());
21         this.setSize(150, 60);
22     }
23

```



```

24     //버튼 생성
25     public JButton getBtnColor() {
26         if (btnColor == null) {
27             btnColor = new JButton();
28             btnColor.setText("JColorChooser");
29             btnColor.addActionListener(new ActionListener() {
30                 public void actionPerformed(ActionEvent e) {
31                     //색상 다이얼로그 보여주기
32                     Color color = JColorChooser.showDialog(
33                         JColorChooserExample.this, "색상 선택", Color.BLUE);
34                     //버튼의 배경색을 변경
35                     btnColor.setBackground(color);
36                 }
37             });
38         }
39         return btnColor;
40     }
41
42     public static void main(String[] args) {
43         SwingUtilities.invokeLater(new Runnable() {
44             public void run() {
45                 JColorChooserExample jFrame = new JColorChooserExample();
46                 jFrame.setVisible(true);
47             }
48         });
49     }
50 }

```

실행 결과



14 2D 그래픽스

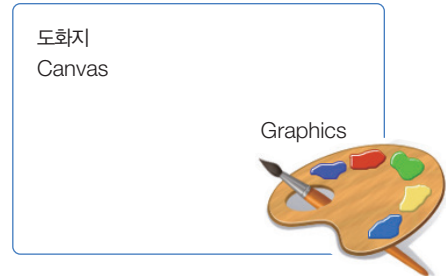
UI 프로그램에서 윈도우 창이나 버튼, 이미지 등은 모두 화면에서 드로잉^{drawing}된 2D 그래픽이다. Swing은 개발자가 코드로 2D 그래픽을 드로잉하도록 Java2D API를 제공한다.

Canvas와 Graphics

손으로 그림을 그리기 위해서는 도화지와 붓이 필요하듯이 Swing에서도 그런 역할을 하는 클래스가 있다. 도화지는 Canvas 클래스이고, 붓은 Graphics 클래스이다.

Canvas는 최초 드로잉 준비가 되면 paint() 메소드를 호출해서 Graphics로 드로잉을 한다. 그리고 다음과 같은 경우에 paint() 메소드를 다시 호출하여 Graphics로 재 드로잉을 한다.

- ① 도화지가 축소되었다가 다시 확대했을 때
- ② 도화지의 크기가 변경되었을 때
- ③ 도화지가 숨겨졌다가 다시 나타났을 때



다음 예제를 실행해 보면 윈도우 창이 축소되었다가 확대했을 때에 Canvas의 paint() 메소드가 다시 호출되어 글자를 다시 그리는 것을 볼 수 있다. 확인 방법은 콘솔창에서 “paint() 메소드 실행”이 계속 출력되는 것을 보면 된다.

» CanvasPaintExample.java

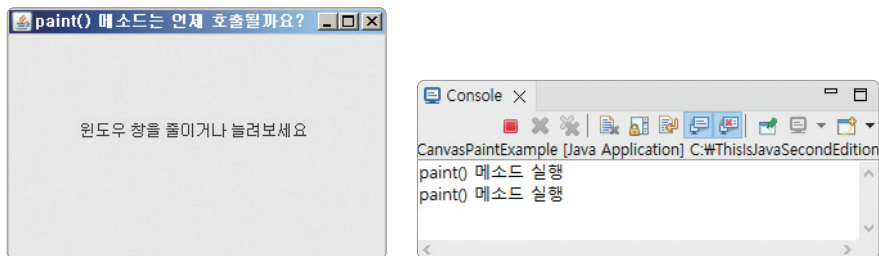
```
1  package sec14.exam01_paint;
2
3  import java.awt.BorderLayout;
4  import java.awt.Canvas;
5  import java.awt.Graphics;
6  import javax.swing.JFrame;
7  import javax.swing.SwingUtilities;
8
9  public class CanvasPaintExample extends JFrame {
10     //메인 윈도우 설정
11     public CanvasPaintExample() {
12         setTitle("paint() 메소드는 언제 호출될까요?");
13
14         //사용자 정의 Canvas 객체를 중앙에 배치
15         getContentPane().add(new MyCanvas(), BorderLayout.CENTER);
16         setSize(300, 200);
```

```

17     }
18
19     //사용자 정의 Canvas 클래스 선언
20     public class MyCanvas extends Canvas {
21         public void paint(Graphics g) {
22             g.drawString("윈도우 창을 줄이거나 늘려보세요", 50, 80);
23             System.out.println("paint() 메소드 실행");
24         }
25     }
26
27     public static void main(String[] args) {
28         SwingUtilities.invokeLater(new Runnable() {
29             public void run() {
30                 CanvasPaintExample jFrame = new CanvasPaintExample();
31                 jFrame.setVisible(true);
32             }
33         });
34     }
35 }

```

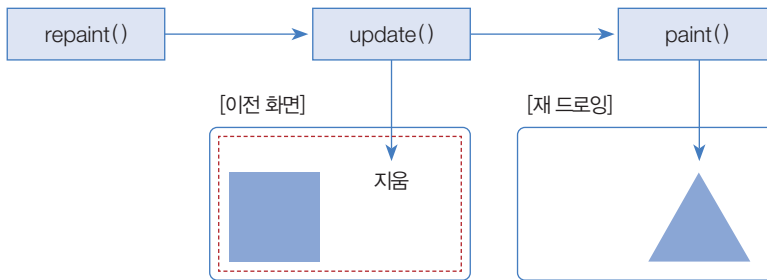
실행 결과



다시 그리기

Canvas에 드로잉하는 작업은 이벤트 디스패칭 스레드가 전담한다. 이벤트 디스패칭 스레드는 Canvas의 내용이 갱신될 필요가 있을 때 `paint()` 메소드를 다시 호출해서 재 드로잉을 한다. 하지만 개발자는 `paint()` 메소드를 직접 호출할 수 없고, 대신 `repaint()` 메소드를 호출할 수 있다.

Canvas의 repaint() 메소드가 호출되면 이벤트 디스패칭 스레드는 update() 메소드를 호출하고 다시 paint() 메소드를 호출해서 재 드로잉을 한다. update() 메소드의 기본 동작은 paint() 메소드를 호출하기 전에 이전 화면 내용을 지우는 역할을 한다.



만약 이전 내용을 유지하면서 재 드로잉을 하려면 update() 메소드를 재정의해서 이전 내용을 지우는 코드를 작성하지 않고 paint(g) 메소드만 호출하면 된다.

```

public void update(Graphics g) {
    paint(g);
}

public void paint(Graphics g) {
    //화면 드로잉
    ...
}
  
```

다음 예제는 Canvas 위에서 마우스를 누른 상태로 드래그하면 이전 경로에서부터 현재 경로까지 *를 유지하면서 드로잉한다. 만약 Canvas의 update() 메소드를 재정의하지 않으면 이전 화면을 지우는 update()의 기본 기능 때문에 이전 경로의 *는 모두 지워진다.

» RepaintExample.java

```

1 package sec14.exam02_repaint;
2
3 import java.awt.BorderLayout;
4 import java.awt.Canvas;
  
```

```

5  import java.awt.Graphics;
6  import java.awt.event.MouseEvent;
7  import java.awt.event.MouseMotionListener;
8  import javax.swing.JFrame;
9  import javax.swing.SwingUtilities;
10
11 public class RepaintExample extends JFrame {
12     //메인 윈도우 설정
13     public RepaintExample() {
14         setTitle("재 드로잉");
15
16         //Canvas를 중앙에 배치
17         getContentPane().add(new MyCanvas(), BorderLayout.CENTER);
18
19         setSize(500, 400);
20     }
21
22     //Canvas 클래스 선언
23     public class MyCanvas extends Canvas implements MouseMotionListener {
24         private int x;
25         private int y;
26
27         public MyCanvas() {
28             //MouseMotionListener 추가
29             addMouseMotionListener(this);
30         }
31
32         //Canvas의 update() 재정의
33         @Override
34         public void update(Graphics g) {
35             paint(g);
36         }
37
38         //Canvas의 paint() 재정의
39         @Override
40         public void paint(Graphics g) {
41             g.drawString("x, y", x, y);
42         }
43
44         //MouseMotionListener의 mouseDragged() 재정의

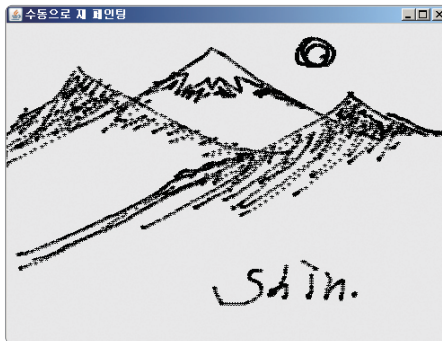
```

```

45     //마우스 버튼을 누르고 움직일 때 호출
46     @Override
47     public void mouseDragged(MouseEvent e) {
48         x = e.getX();
49         y = e.getY();
50
51         //Canvas의 repaint() 호출
52         repaint();
53     }
54
55     //MouseMotionListener의 mouseMoved() 재정의
56     //마우스 버튼을 누르지 않고 움직일 때 호출
57     @Override
58     public void mouseMoved(MouseEvent e) {
59         System.out.println("aaa");
60     }
61 }
62
63 public static void main(String[] args) {
64     SwingUtilities.invokeLater(new Runnable() {
65         public void run() {
66             RepaintExample JFrame = new RepaintExample();
67             JFrame.setVisible(true);
68         }
69     });
70 }
71 }

```

실행 결과



Color와 Font

붓에 물감을 묻혀 그림을 그리듯이 Graphics의 setColor() 메소드로 Color를 설정하면 해당 색상으로 문자, 선, 도형이 그려진다. 예를 들어 빨간색으로 설정하려면 다음과 같이 한다.

```
g.setColor(Color.RED);
```

Color 클래스는 기본적으로 13가지의 색깔을 표현하는 Color 상수를 가지고 있다. 이들 상수를 이용하면 13가지의 색깔 객체를 쉽게 얻을 수 있다.

Color.BLACK	Color.WHITE	Color.RED
Color.GREEN	Color.BLUE	Color.GRAY
Color.DARK_GRAY	Color.LIGHT_GRAY	Color.CYAN
Color.MAGENTA	Color.ORANGE	Color.PINK
Color.YELLOW		

13가지 상수 이외의 다른 색상을 얻고 싶다면 R(Red), G(Green), B(Blue) 값을 이용해서 Color 객체를 직접 만들면 된다. RGB값은 각각 0~255 사이의 값을 주면 된다.

```
Color color = new Color(100, 200, 50);  
g.setColor(color);
```

Graphics의 drawString()으로 그려지는 글자의 기본 색상은 검정색이고, 사이즈는 12, 폰트의 종류는 Dialog이다. Graphics의 setFont() 메소드를 이용하면 폰트 속성을 변경할 수 있다. 매개값은 Font 객체인데, 다음과 같이 생성한다.

```
Font font = new Font(String name, int style, int size);
```

name은 폰트 이름으로 굴림체, 돋움체, 바탕체, Arial 등을 말한다. 폰트는 운영체제마다 차이가 있기 때문에 운영체제에서 제공되는 폰트의 이름을 알아내기 위해 다음 코드를 사용할 수 있다.

```

GraphicsEnvironment ge = GraphicsEnvironment.getLocalGraphicsEnvironment();
String[] fontNames = ge.getAvailableFontFamilyNames();
for(String fontName : fontNames) {
    System.out.println(fontName);
}

```

style은 폰트 스타일로 보통, 굵음, 기울임을 말한다. 폰트 스타일은 Font 상수를 사용하는데, 다음과 같이 세 가지 종류가 있다.

Font.PLAIN (보통 글자)	Font.BOLD (굵은 글자)	Font.ITALIC(기울인 글자)
--------------------	-------------------	---------------------

이들 상수는 | 연산자를 사용해서 조합할 수 있다. 다음은 돋움체이고, 굵고 기울어지게 15 사이즈로 Font 객체를 만들고 Graphics에 설정한다.

```

Font font = new Font("돋움체", Font.BOLD | Font.ITALIC, 15);
g.setFont(font);

```

다음 예제는 Color와 Font를 변경해서 글자를 드로잉하는 방법을 보여 준다.

» ColorFontExample.java

```

1  package sec14.exam03_color_font;
2
3  import java.awt.BorderLayout;
4  import java.awt.Canvas;
5  import java.awt.Color;
6  import java.awt.Font;
7  import java.awt.Graphics;
8  import javax.swing.JFrame;
9  import javax.swing.SwingUtilities;
10
11 public class ColorFontExample extends JFrame {
12     //메인 윈도우 설정
13     public ColorFontExample() {
14         setTitle("색상과 폰트");

```



```

15     getContentPane().add(new MyCanvas(), BorderLayout.CENTER);
16     setSize(300, 200);
17 }
18
19 //Canvas 클래스 선언
20 public class MyCanvas extends Canvas {
21     public void paint(Graphics g) {
22         //Color 변경
23         g.setColor(Color.BLUE);
24         //Font 변경
25         g.setFont(new Font("돋움체", Font.BOLD | Font.ITALIC, 30));
26         //글자 드로잉
27         g.drawString("Color and Font", 20, 100);
28     }
29 }
30
31 public static void main(String[] args) {
32     SwingUtilities.invokeLater(new Runnable() {
33         public void run() {
34             ColorFontExample jFrame = new ColorFontExample();
35             jFrame.setVisible(true);
36         }
37     });
38 }
39 }

```

실행 결과

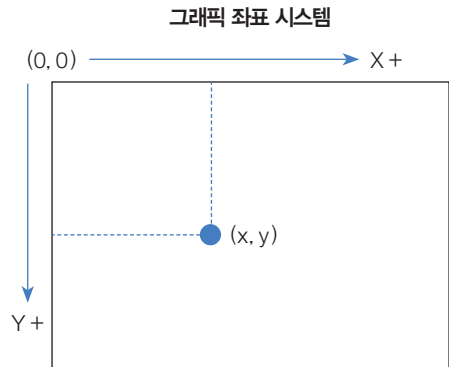


기본 도형 그리기

기본 도형에는 선, 원, 타원, 사각형, 호 등이 있다. Graphics는 이 기본 도형들을 그리기 위한 메소드를 다음과 같이 제공한다.

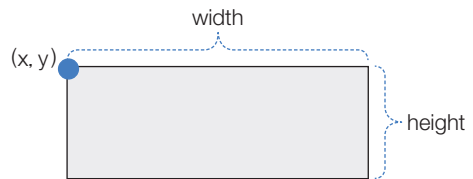
1) 선을 그리는 메소드

- `drawLine(int x1, int y1, int x2, int y2)`



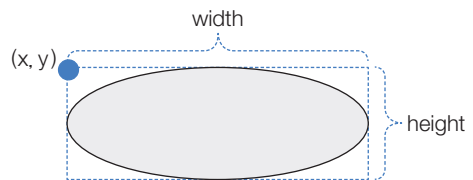
2) 사각형을 그리는 메소드

- `drawRect(int x, int y, int width, int height)`
- `fillRect(int x, int y, int width, int height)`



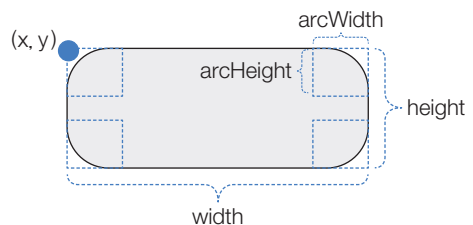
3) 원 또는 타원을 그리는 메소드

- `drawOval(int x, int y, int width, int height)`
- `fillOval(int x, int y, int width, int height)`



4) 모서리가 둥근 사각형을 그리는 메소드

- `drawRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight)`
- `fillRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight)`

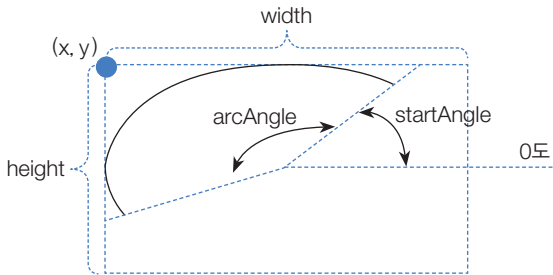


5) 입체 사각형을 그리는 메소드

- draw3DRect(int x, int y, int width, int height, boolean raised)
- fill3DRect(int x, int y, int width, int height, boolean raised)

6) 호를 그리는 메소드

- drawArc(int x, int y, int width, int height, int startAngle, int arcAngle)
- fillArc(int x, int y, int width, int height, int startAngle, int arcAngle)



7) 다각형을 그리는 메소드

- drawPolygon(int[] xPoints, int[] yPoints, int nPoints)
- fillPolygon(int[] xPoints, int[] yPoints, int nPoints)
- drawPolyline(int[] xPoints, int[] yPoints, int nPoints)

8) 영역을 지우는 메소드

- clearRect(int x, int y, int width, int height)

drawXXX()와 fillXXX()의 차이점은 선만 그리느냐 아니면 내부를 채우느냐이다.

다음 예제는 Graphics가 제공하는 다양한 메소드를 이용해서 Canvas에 그림을 그린다.

>>> ShapeExample.java

```
1 package sec14.exam04_shape;
2
3 import java.awt.BorderLayout;
4 import java.awt.Canvas;
```

```

5  import java.awt.Color;
6  import java.awt.Graphics;
7  import javax.swing.JFrame;
8  import javax.swing.SwingUtilities;
9
10 public class ShapeExample extends JFrame {
11     //메인 윈도우 설정
12     public ShapeExample() {
13         setTitle("도형 그리기");
14         getContentPane().add(new MyCanvas(), BorderLayout.CENTER);
15         setSize(400,300);
16     }
17
18     //Canvas 클래스 선언
19     public class MyCanvas extends Canvas {
20         public void paint(Graphics g) {
21             //원 그리기
22             g.drawOval(50,50, 50,50);
23             g.setColor(Color.GREEN);
24             g.fillOval(50,100, 50,50);
25
26             //선 그리기
27             g.setColor(Color.RED);
28             g.drawLine(50,100, 100,150);
29             g.drawLine(150,50, 50,150);
30
31             //사각형 그리기
32             g.setColor(Color.BLUE);
33             g.drawRoundRect(200, 50, 120, 80, 30, 30);
34             g.setColor(Color.YELLOW);
35             g.fillRoundRect(250, 100, 120, 80, 30, 30);
36
37             //다각형 그리기
38             g.setColor(Color.ORANGE);
39             g.fillPolygon(new int[]{50, 100, 150, 200}, new int[]{250, 200, 200,
40                                     250}, 4);
41
42             //호 그리기
43             g.setColor(Color.cyan);
44             g.fillArc(250, 200, 100, 100, 45, 120);

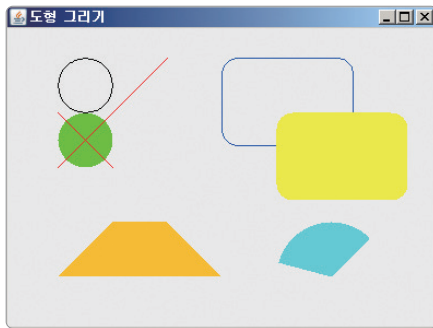
```

```

44     }
45 }
46
47 public static void main(String[] args) {
48     SwingUtilities.invokeLater(new Runnable() {
49         public void run() {
50             ShapeExample jFrame = new ShapeExample();
51             jFrame.setVisible(true);
52         }
53     });
54 }
55 }

```

실행 결과



안티 알리아싱

비트맵 그래픽의 방식은 작은 사각형(픽셀)을 최소 단위로 하기 때문에 이 픽셀들이 모여 만들어진 원, 곡선, 사선은 경계 부분에서 거친 계단 현상(알리어싱^{aliasing})이 나타난다. 이 문제를 해결하기 위해 안티-알리어싱^{anti-aliasing} 기능이 필요하다.

안티알리어싱은 배경색과 이미지 색상의 중간 색상을 단계적으로 채워 경계선을 부드럽게 만들어 주는 기능이다. 자바는 안티 알리어싱을 위해 Graphics2D의 setRenderingHint() 메소드를 제공한다.

paint() 메소드의 매개변수 타입은 Graphics이지만 Graphics2D 객체를 참조하고 있기 때문에 다음과 같이 타입 변환을 통해 Graphics2D 객체를 얻고 setRenderingHint() 메소드를 호출할 수 있다.

```
public void paint(Graphics g) {
    Graphics2D g2 = (Graphics2D) g;
    g2.setRenderingHint(
        RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);
    ...
}
```

Java 2D는 여러 가지 렌더링 알고리즘을 키와 값으로 구성된 맵Map 객체로 관리하고 있다. 안티 알리아싱 기능을 활성화할지 여부의 값은 RenderingHints.KEY_ANTIALIASING 키로 관리하고 있기 때문에 이 키의 값을 RenderingHints.VALUE_ANTIALIAS_ON으로 변경하면 된다.

다음 예제는 안티 알리아싱을 적용하지 않은 원과 적용한 원의 차이점을 보여 준다.

>>> AntiAliasingExample.java

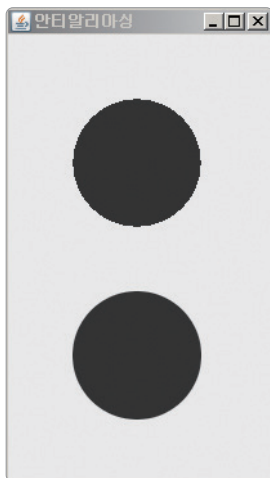
```
1  package sec14.exam05_antialiasing;
2
3  import java.awt.BorderLayout;
4  import java.awt.Canvas;
5  import java.awt.Graphics;
6  import java.awt.Graphics2D;
7  import java.awt.RenderingHints;
8  import javax.swing.JFrame;
9  import javax.swing.SwingUtilities;
10
11 public class AntiAliasingExample extends JFrame {
12     //메인 윈도우 설정
13     public AntiAliasingExample() {
14         setTitle("안티알리아싱");
15         getContentPane().add(new MyCanvas(), BorderLayout.CENTER);
16         setSize(200,350);
17     }
18
19     //Canvas 클래스 선언
```

```

20     public class MyCanvas extends Canvas {
21         public void paint(Graphics g) {
22             //안티알리어싱을 적용하지 않은 원
23             g.fillOval(50, 50, 100, 100);
24
25             //안티알리어싱을 적용한 원
26             Graphics2D g2 = (Graphics2D) g;
27             g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
28                                 RenderingHints.VALUE_ANTIALIAS_ON);
29             g.fillOval(50, 200, 100, 100);
30         }
31     }
32
33     public static void main(String[] args) {
34         SwingUtilities.invokeLater(new Runnable() {
35             public void run() {
36                 AntiAliasingExample jFrame = new AntiAliasingExample();
37                 jFrame.setVisible(true);
38             }
39         });
40     }

```

실행 결과



이미지 그리기

이미지를 드로잉하기 위해서는 이미지 파일을 메모리로 로딩해서 Image 객체를 얻어야 한다. Image 객체를 얻는 방법은 두 가지가 있는데, 첫 번째로는 Toolkit의 getImage() 메소드로 얻을 수 있다.

```
Toolkit toolkit = Toolkit.getDefaultToolkit();  
//파일 시스템의 이미지 경로를 가지고 있을 경우  
Image image1 = toolkit.getImage(String filename);  
//파일 시스템 또는 인터넷의 이미지 URL을 가지고 있을 경우  
Image image2 = toolkit.getImage(URL url);
```

두 번째 방법은 ImageIcon 객체를 생성해서 getImage() 메소드로 얻을 수 있다.

```
//파일 시스템의 이미지 경로를 가지고 있을 경우  
ImageIcon imageIcon1 = new ImageIcon(String filename);  
Image image1 = imageIcon1.getImage();  
  
//파일 시스템 또는 인터넷의 이미지 URL을 가지고 있을 경우  
ImageIcon imageIcon2 = new ImageIcon(URL url);  
Image image2 = imageIcon2.getImage();
```

URL을 생성하는 방법은 다음과 같다.

```
//파일 시스템의 이미지 경로를 가지고 있을 경우  
URL url = new URL("file:///C:\images\logo.gif");  
  
//클래스 파일과 동일한 디렉토리에 있는 이미지 파일일 경우  
URL url = getClass().getResource("logo.gif");  
  
//인터넷의 이미지 URL을 가지고 있을 경우  
URL url = new URL("http://www.naver.com/logo.gif");
```

두 가지 방법을 이용해서 Image 객체를 얻었다면 이미지를 Canvas에 드로잉할 수 있는데, 다음과 같이 Graphics의 drawImage() 메소드를 이용한다. img는 Image 객체이고, x와 y는 드로잉이 시작될 좌상단 좌표이다.


```
drawImage(Image img, int x, int y, ImageObserver observer)
```

observer는 이미지 로딩과 드로잉을 동시에 할 때 필요하다. 이미지를 로딩하는 도중에 주기적으로 ImageObserver의 `imageUpdate()` 메소드가 호출되는데, 이 메소드는 Canvas의 `repaint()` 메소드를 호출해서 그때까지 로딩된 이미지를 다시 그리도록 한다. Canvas는 기본적으로 ImageObserver 인터페이스를 구현하고 있기 때문에 this를 사용해서 observer의 매개값으로 줄 수 있다

다음 예제는 ImageExample 클래스와 동일한 위치에 있는 이미지 파일을 읽고, Canvas에 드로잉하는 방법을 보여 준다.

>>> ImageExample.java

```
1  package sec14.exam06_image;
2
3  import java.awt.BorderLayout;
4  import java.awt.Canvas;
5  import java.awt.Color;
6  import java.awt.Graphics;
7  import java.awt.Image;
8  import java.awt.Toolkit;
9  import javax.swing.ImageIcon;
10 import javax.swing.JFrame;
11 import javax.swing.SwingUtilities;
12
13 public class ImageExample extends JFrame {
14     //메인 윈도우 설정
15     public ImageExample() {
16         setTitle("이미지 그리기");
17         getContentPane().add(new MyCanvas(), BorderLayout.CENTER);
18         add(new MyCanvas(), BorderLayout.CENTER);
19         setSize(500,350);
20     }
21
22     //Canvas 클래스 선언
23     public class MyCanvas extends Canvas {
24         private Image imgSun, imgMoon;;
```

```

25
26     public MyCanvas() {
27         //배경색을 흰색으로 변경
28         setBackground(Color.WHITE);
29
30         //이미지를 로딩해서 읽고, Image 객체 얻기
31         Toolkit toolkit = Toolkit.getDefaultToolkit();
32         imgSun = toolkit.getImage(getClass().getResource("sun.gif"));
33         imgMoon = new ImageIcon(getClass().getResource("moon.gif")).getImage();
34     }
35
36     public void paint(Graphics g) {
37         //이미지 드로잉
38         g.drawImage(imgSun, 10, 10, this);
39         g.drawImage(imgMoon, 300, 20, this);
40     }
41 }
42
43 public static void main(String[] args) {
44     SwingUtilities.invokeLater(new Runnable() {
45         public void run() {
46             ImageExample jFrame = new ImageExample();
47             jFrame.setVisible(true);
48         }
49     });
50 }
51 }

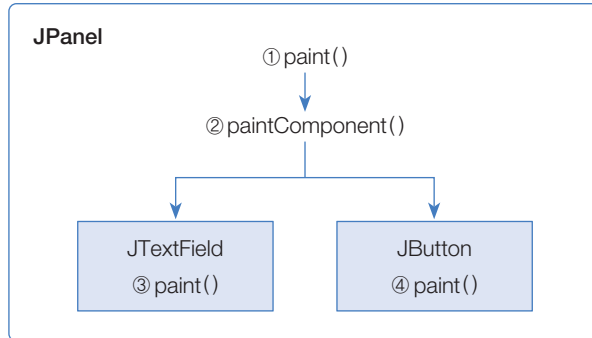
```

실행 결과



배경 이미지 넣기

JPanel에 배경 이미지를 넣는 방법을 알아보자. JPanel은 `paint()` 메소드를 실행할 때 자신의 `paintComponent()` 메소드를 먼저 호출하고, 자식 컴포넌트의 `paint()` 메소드를 나중에 호출하도록 되어 있다. 배경 그림은 모든 자식 컴포넌트보다 먼저 드로잉되어 자식 컴포넌트 밑에 깔려야 하므로 `paintComponent()`에서 배경 그림이 드로잉되어야 한다.



다음 예제는 JPanel에 배경 이미지를 드로잉해서 JTextField와 JButton가 배경 이미지 위에 배치 되도록 한다.

>>> BackgroundImageExample.java

```
1  package sec14.exam07_background_image;
2
3  import java.awt.BorderLayout;
4  import java.awt.Graphics;
5  import javax.swing.ImageIcon;
6  import javax.swing.JButton;
7  import javax.swing.JFrame;
8  import javax.swing.JPanel;
9  import javax.swing.JTextField;
10 import javax.swing.SwingUtilities;
11
12 public class BackgroundImageExample extends JFrame {
13     private JTextField txtId;
14     private JButton btnLogin;
```

```

15
16 //메인 윈도우 설정
17 public BackgroundImageExample() {
18     this.setTitle("배경 그림 넣기");
19     this.getContentPane().add(new JPanel(), BorderLayout.CENTER);
20     this.setSize(200, 270);
21     this.setResizable(false);
22     this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
23 }
24
25 //JPanel 클래스 선언
26 public class JPanel extends JPanel {
27     public JPanel() {
28         setLayout(null);
29         //JTextField와 JButton 부착
30         add(getTextField());
31         add(getButton());
32     }
33
34     @Override
35     public void paintComponent(Graphics g) {
36         //배경 그리기
37         ImageIcon icon = new ImageIcon(this.getClass().getResource("bg.jpg"));
38         g.drawImage(icon.getImage(), 0, 0, this);
39     }
40 }
41
42 //JTextField 생성
43 public JTextField getTextField() {
44     if(txtId == null) {
45         txtId = new JTextField();
46         txtId.setBounds(50, 50, 100, 30);
47     }
48     return txtId;
49 }
50
51 //JButton 생성
52 public JButton getButton() {
53     if(btnLogin == null) {
54         btnLogin = new JButton("버튼");

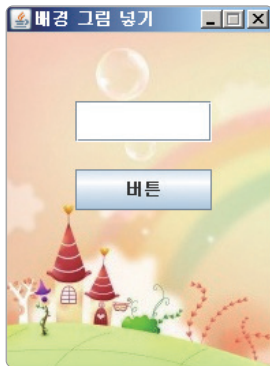
```

```

55         btnLogin.setBounds(50, 100, 100, 30);
56     }
57     return btnLogin;
58 }
59
60 public static void main(String[] args) {
61     SwingUtilities.invokeLater(new Runnable() {
62         public void run() {
63             BackgroundImageExample JFrame = new BackgroundImageExample();
64             JFrame.setVisible(true);
65         }
66     });
67 }
68 }

```

실행 결과



15 Swing 과제

지금까지 학습한 내용을 기반으로 몇 가지 과제를 수행해보자.

과제1 – 게시판

게시판 애플리케이션의 첫 메인 윈도우를 다음과 같이 JFrame을 사용해서 구성한다.

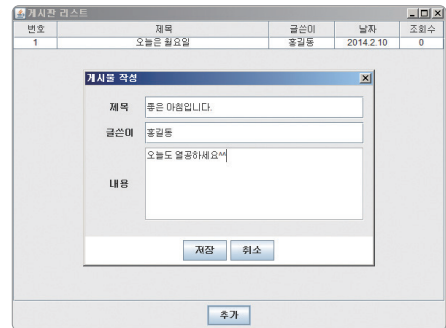
그리고 하단의 추가 버튼을 클릭하면 jTable에 새로운 행이 삽입될 수 있도록 BoardApp.java를 작성한다.



과제2 – 게시판

과제1에서 만든 BoardApp을 수정하는데, [추가] 버튼을 클릭하면 다음과 같이 게시물 입력 다이얼로그를 띄우도록 한다.

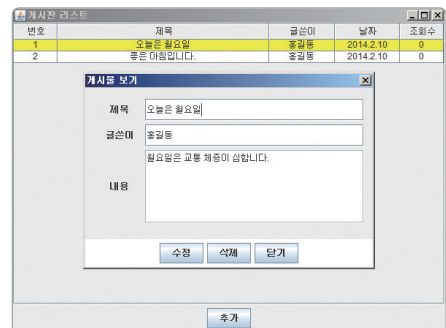
그리고 입력 다이얼로그에서 제목, 글쓴이, 내용을 입력하고 저장 버튼을 클릭하면 jTable에 행이 추가되도록 한다.



과제3 – 게시판

과제2에 이어서 jTable 행을 클릭하면 다음과 같이 게시물 뷰 다이얼로그를 띄운다.

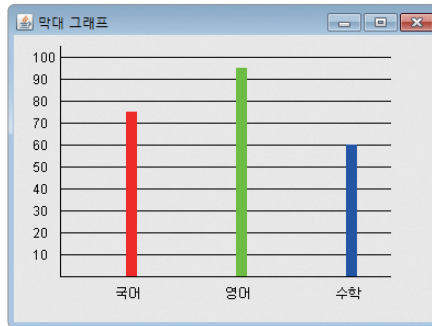
내용을 수정하고 [수정] 버튼을 클릭하면 해당 행의 데이터가 수정되게 하고, [삭제] 버튼을 클릭하면 해당 행이 삭제가 되도록 작성한다.



과제4 – 막대 그래프

국어, 영어, 수학 점수가 다음 표와 같다고 할 때, 오른쪽처럼 막대 그래프로 보여주는 애플리케이션을 작성한다.

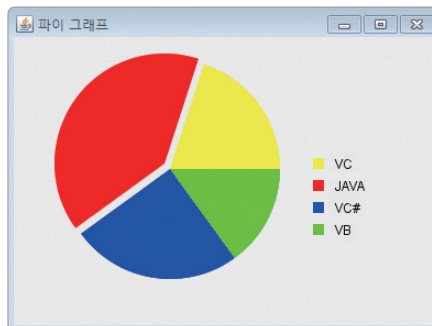
과목	점수
국어	75
영어	95
수학	60



과제5 – 파이 그래프

100명의 개발자에게 자신이 제일 좋아하는 언어를 조사하였더니 다음과 같은 결과가 나왔다. 오른쪽 쪽처럼 파이 그래프로 보여주는 애플리케이션을 작성한다.

언어	좋아하는 사람
VC	20
Java	40
C#	25
VB	15



Appendix

03

▶ Java UI – JavaFX

- 01 JavaFX 개요
- 02 JavaFX 프로젝트 생성 및 실행
- 03 JavaFX 레이아웃
- 04 JavaFX 컨테이너
- 05 JavaFX 이벤트 처리
- 06 JavaFX 속성 감시와 바인딩
- 07 JavaFx 컨트롤
- 08 JavaFX 메뉴바와 툴바
- 09 JavaFX 다이얼로그
- 10 JavaFX CSS 스타일
- 11 JavaFX 스레드 UI 변경
- 12 장면 이동과 애니메이션
- 13 JavaFX 과제

01 JavaFX 개요

JavaFX는 자바 기반 UI 애플리케이션(Rich Client Application)을 개발할 때 사용할 수 있는 그래픽과 미디어 모듈을 말한다. 자바의 UI 라이브러리의 변천 과정은 다음과 같다.

AWT

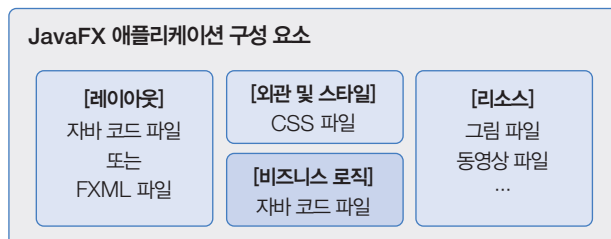
자바 언어가 탄생한 1995년에는 인터넷이 활성화되지 않았기 때문에 대부분의 클라이언트 애플리케이션은 운영체제가 제공하는 네이티브(native) UI 컴포넌트를 이용해서 개발되었다. 그래서 Java 1.0에 포함된 AWT(Abstract Window Toolkit)는 운영 체제가 제공하는 네이티브 UI 컴포넌트를 이용하는 자바 라이브러리였다. 그렇다 보니 자바 애플리케이션이 실행되는 운영체제에 따라 UI의 모양이 서로 달랐고, 종류도 제한적이었다.

Swing

AWT의 다음 주자로 Swing이 등장했다. Swing의 중심 아이디어는 운영체제가 제공하는 네이티브 UI 컴포넌트를 사용하지 말자는 것이었다. 즉 모든 운영체제상에서 동일한 UI를 갖도록, Swing 자신만의 UI(look and feel)를 갖도록 했다. 그렇다 보니 실행 성능이 느려지고, 메모리를 더 많이 사용하게 되었다. 시간이 흘러 네이티브 UI가 애니메이션과 다양한 시각적인 효과를 내면서 사용자는 Swing의 UI보다는 운영체제의 네이티브 UI를 더 선호하게 되었다. 그래서 Swing은 점점 네이티브 UI에 밀려나기 시작했다.

JavaFX

JavaFX는 Java 기반의 데스크톱, 모바일 및 임베디드 시스템을 위한 오픈 소스 차세대 UI 클라이언트 애플리케이션 플랫폼이다. JavaFX는 자바 코드와 분리해서 스타일 시트(CSS)를 작성할 수 있기 때문에 테마를 쉽게 변경할 수 있고, 안드로이드처럼 화면 레이아웃과 비즈니스 로직을 분리할 수 있기 때문에 Swing보다는 편리하게 디자인을 할 수 있다는 장점이 있다. 다음은 JavaFX 애플리케이션을 구성하는 파일 단위 요소를 보여준다.

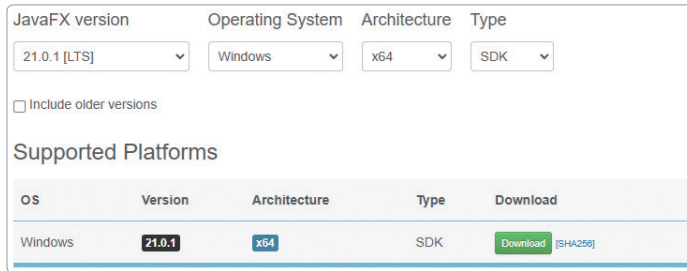


JavaFX 설치

JavaFX는 Java 7과 Java 8에서는 JDK에 포함되어 있지만, Java 11부터 JDK에 포함되지 않고 오픈 소스 모듈로 진화하였다. 따라서 JavaFX를 사용하려면 별도의 설치 작업이 필요하다. JavaFX 모듈 설치 파일은 다음 URL에서 다운로드할 수 있다.

<https://gluonhq.com/products/javafx>

장기 지원(Long Term Support, LTS) 버전을 받는 것이 좋기 때문에 21.0.1 [LTS] 버전을 선택한다. 운영체제와 CPU 아키텍처는 사용하는 PC에 따라 선택하고, 타입은 SDK를 선택한다.



JavaFX version: 21.0.1 [LTS]
Operating System: Windows
Architecture: x64
Type: SDK

☐ Include older versions

Supported Platforms

OS	Version	Architecture	Type	Download
Windows	21.0.1	x64	SDK	Download SHA256

NOTE ▶ JavaFX 버전은 다운로드하는 시점에 따라 달라질 수 있다.

윈도우 운영체제용 설치 파일은 압축 파일(zip) 형식으로 되어 있다. 압축을 해제하면 다음 폴더가 나온다.

`javafx-sdk-21.0.1`

이 폴더를 다음 경로로 이동시킨다.

`C:\ThisisJava\javafx-sdk-21.0.1`

`javafx-sdk-21.0.1` 폴더 내부를 보면 `lib` 폴더가 있다. 이 폴더에는 JavaFX에서 제공하는 모듈 파일(JAR)이 있다. 이들 모듈에 포함되어 있는 패키지명과 패키지 내부의 클래스 및 인터페이스를 보려면 API 도큐먼트가 필요하다. API 도큐먼트는 다음 URL에서 볼 수 있다. 학습할 때 언제든지 참고할 수 있도록 웹 브라우저 북마크(즐거찾기)에 추가해 두자.

<https://openjfx.io/javadoc/21>

Scene Builder 설치

씬 빌더(Scene Builder)는 드래그 앤 드롭 사용자 인터페이스 디자인 방식으로 JavaFX의 화면 레이아웃 FXML 파일을 자동 생성해주는 오픈 소스 도구이다. Scene Builder 설치 파일은 다음 URL에서 다운로드할 수 있다.

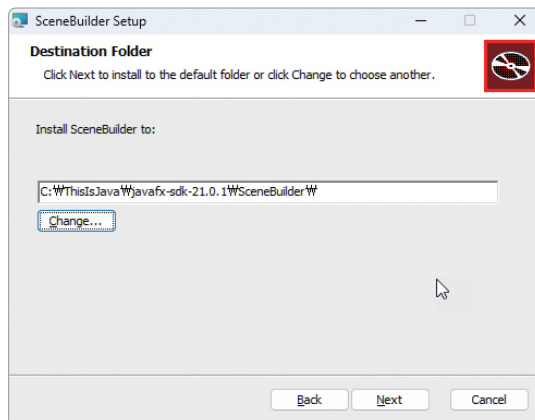
<https://gluonhq.com/products/scene-builder>

사용하는 PC의 운영체제에 따라 설치 파일을 다운로드한다.

Product	Platform	Download
Scene Builder	Windows Installer	Download
Scene Builder	Mac OS X dmg (Intel)	Download
Scene Builder	Mac OS X dmg (Apple Silicon)	Download
Scene Builder	Linux RPM	Download
Scene Builder	Linux Deb	Download
Scene Builder Kit info	Jar File	Download

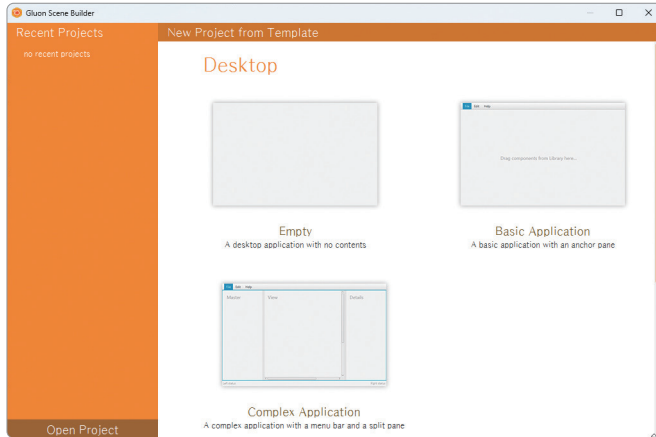
설치 과정에서 윈도우 운영체제에서의 설치 경로를 다음과 같이 변경해 준다.

C:\ThisIsJava\javafx-sdk-21.0.1\SceneBuilder

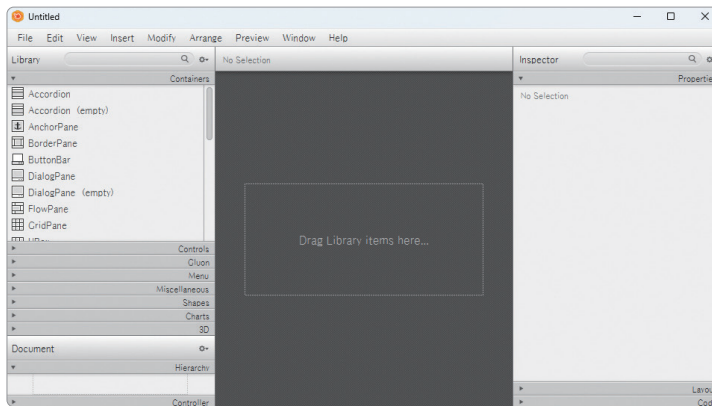


설치 완료 후 다음 파일을 더블 클릭하면 Scene Builder가 실행된다.

```
C:\ThisisJava\javafx-sdk-21.0.1\SceneBuilder\SceneBuilder.exe
```



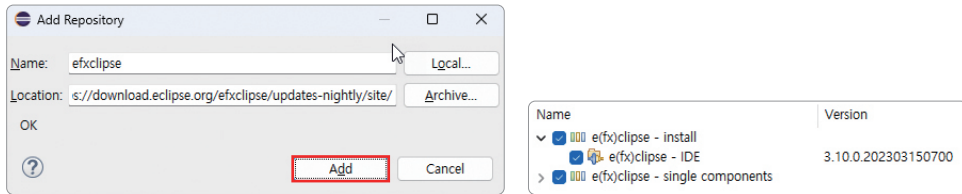
Empty 템플릿을 선택하면 다음과 같은 FXML 편집기가 실행된다. 편집기 실행을 확인하고 우측 상단에 있는 종료 버튼(ⓧ)을 클릭해서 편집기를 닫는다.



e(fx)clipse 설치

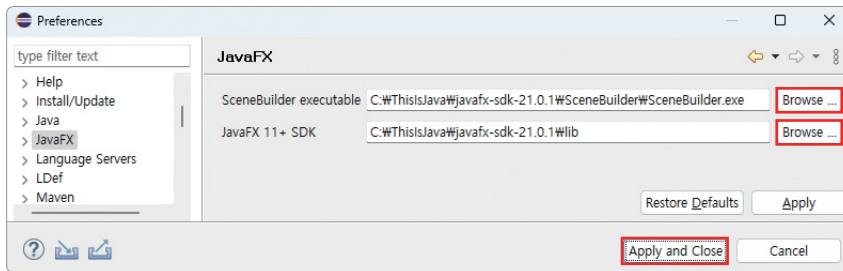
e(fx)clipse는 JavaFX 개발도구를 제공해 주는 이클립스 플러그인이다. 프로젝트 생성 및 자동 완성, FXML 파일을 Scene Builder로 열기 등의 기능을 제공하기 때문에 매우 편리하다.

이클립스 메뉴에서 [Help] - [Install New Software...]를 선택하고, [work with:]에 'https://download.eclipse.org/efxclipse/updates-nightly/site/'를 입력하고 [Add...] 버튼을 클릭하면 다음과 같은 [Add Repository] 대화상자를 확인할 수 있다. 대화상자의 [Name:]에 'efxclipse'라고 입력하고 [Add] 버튼을 클릭해 설치한다.



NOTE ▶ e(fx)clipse는 3.10 버전 이상을 설치해야 하지만, 현재는 아직 nightly 버전에 해당하므로 정식 버전이 출시 되면 [Eclipse Marketplace...]에서 설치할 수 있을 것이다.

이클립스 메뉴에서 [Window] - [Preferences]를 선택한다. 왼쪽 항목에서 JavaFX를 선택하면 다음과 같이 SceneBuilder 실행 파일 경로와 JavaFX SDK 경로를 입력하는 내용이 나온다.



입력란에서 [Browse] 버튼을 클릭해 다음 경로의 파일을 선택해준다.

SceneBuilder executable

C:\ThisisJava\javafx-sdk-21.0.1\SceneBuilder\SceneBuilder.exe

JavaFX 11 + SDK

C:\ThisisJava\javafx-sdk-12.0.1\lib

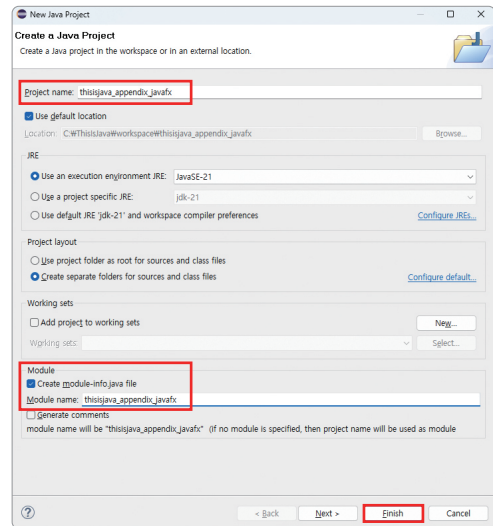
[Apply and Close] 버튼을 클릭해 적용한다.

02 JavaFX 프로젝트 생성 및 실행

JavaFX 애플리케이션을 위한 프로젝트는 Java Project 또는 e(fx)clipse가 제공하는 JavaFX Project 위자드를 이용해서 생성할 수 있다. 처음 학습할 때에는 Java Project로 생성하는 것이 좋고, JavaFX가 어느 정도 익숙해지면 JavaFX Project 위자드를 이용하는 것이 좋다.

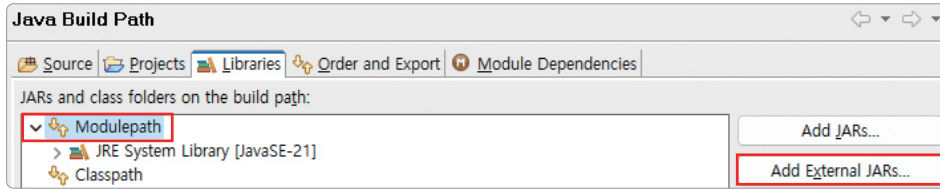
프로젝트 생성 및 설정

이클립스 메뉴에서 [File] - [New] - [Java Project]를 선택한다. [New Java Project] 대화상자의 [Project name]에 'thisisjava_appendix_javafx'를 입력한다. JavaFX 프로젝트는 JavaFX SDK가 제공하는 모듈을 참조해야 하고, 이들 모듈에 대한 의존 설정을 해야 하므로 모듈 기술자가 포함된 프로젝트로 생성해야 한다. 따라서 [Create module-info.java file] 체크박스에 체크한다. [Module name]은 [Project name]과 동일하게 'thisisjava_appendix_javafx'라고 입력하고 [Finish] 버튼을 클릭한다.



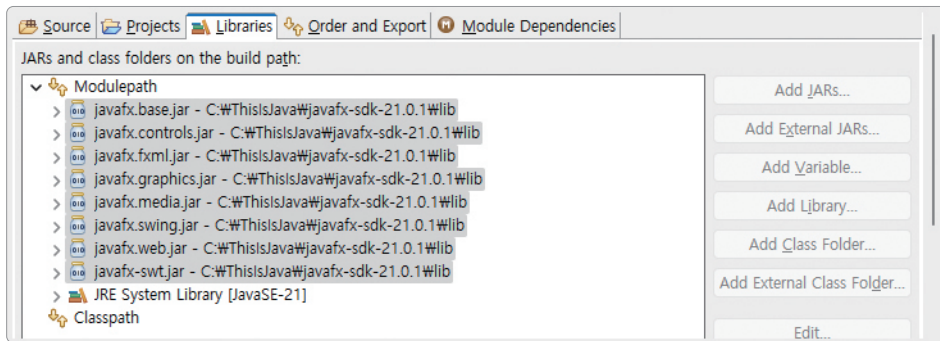
JavaFX 모듈 참조 및 의존성 설정

JavaFX 프로젝트는 JavaFX SDK가 제공하는 모듈을 참조해야 한다. thisisjava_appendix_javafx 프로젝트를 선택하고 마우스 오른쪽 버튼을 클릭하여 [Build Path] - [Configure Build Path] 메뉴를 선택한다. [Libraries] 탭에서 Modulepath를 선택하고, [Add External JARs] 버튼을 클릭한다.



다음 폴더에 있는 모든 모듈 파일(JAR)을 선택하고, 화살표를 눌러 확장한다.

C:\ThisisJava\javafx-sdk-21.0.1\lib



우리 책에서는 javafx.control, javafx.fxml, javafx.media 모듈을 사용하므로 모듈 기술자에서 이들 모듈에 대한 의존 설정을 해야 한다.

>>> module-info.java

```

1  open module thisisjava_appendix_javafx {
2      requires java.se;
3      requires javafx.controls;
4      requires javafx.fxml;
5      requires javafx.media;
6  }

```

open 키워드가 추가된 이유는 JavaFX 애플리케이션이 실행될 때 클래스 리플렉션을 수행하기 때문에 프로젝트 내부의 모든 클래스에 대해 리플렉션을 허용하기 위해서이다. java.se 집합 모듈은 JDK가 제공하는 모든 모듈을 사용하기 위해 추가하였다.

메인 클래스 작성

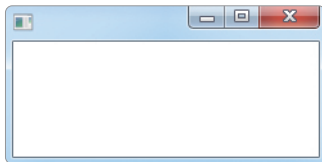
JavaFX 애플리케이션을 개발하려면 제일 먼저 메인 클래스를 작성해야 한다. 메인 클래스는 JavaFX 애플리케이션을 시작하는 관문이다. 메인 클래스는 추상 클래스인 `Application`을 상속받고, `start()` 메소드를 재정의해야 한다.

그리고 `main()` 메소드에서는 `Application`의 `launch()` 메소드를 호출해야 한다. `launch()` 메소드는 메인 클래스의 객체를 생성하고, 메인 윈도우를 생성한 다음 `start()` 메소드를 호출하는 역할을 한다.

>>> AppMain.java

```
1  package sec02.exam01_application_start;
2
3  import javafx.application.Application;
4  import javafx.stage.Stage;
5
6  public class AppMain extends Application {
7      @Override
8      public void start(Stage primaryStage) throws Exception {
9          primaryStage.show();    //윈도우 보여주기
10     }
11
12     public static void main(String[] args) {
13         launch(args);    //AppMain 객체 생성 및 메인 윈도우 생성
14     }
15 }
```

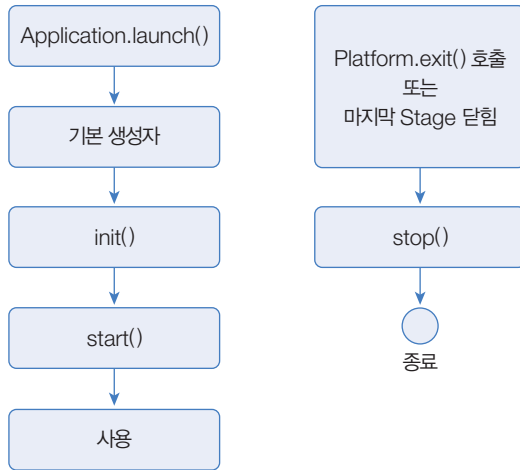
실행 결과



JavaFX는 윈도우를 무대(Stage)로 표현한다. `launch()`는 `start()` 메소드를 호출할 때 메인 윈도우를 `primaryStage`로 제공하는데, `start()`에서 `primaryStage.show()` 메소드를 호출함으로써 메인 윈도우가 보여진다.

JavaFX 라이프사이클

JavaFx 애플리케이션은 `Application.launch()` 메소드부터 시작해서 다음과 같은 순서로 진행된다.



`launch()` 메소드가 호출되면 메인 클래스의 기본 생성자를 호출해서 객체를 생성하고, 이어서 `init()` 메소드를 호출한다. `init()` 메소드는 메인 클래스의 실행 매개값을 얻어 애플리케이션이 이용할 수 있도록 해준다. `init()`이 끝나고 나면, `start()` 메소드를 호출해서 메인 윈도우를 실행시킨다.

JavaFX 애플리케이션이 종료되는 경우는 다음과 같이 세 가지가 있다.

- ① 마우스로 마지막 윈도우(Stage)의 우측 상단 닫기 버튼을 클릭했을 때
- ② 자바 코드로 마지막 윈도우(Stage)의 `close()` 메소드를 호출했을 때
- ③ 자바 코드로 `Platform.exit()` 또는 `System.exit(0)`을 호출했을 때

JavaFX 애플리케이션은 종료되기 직전에 `stop()` 메소드를 호출하는데, `stop()` 메소드는 사용한 자원을 정리(파일 닫기, 네트워크 끊기)할 기회를 준다. `init()`과 `stop()` 메소드는 옵션으로 필요한 경우에 재정의해서 사용하면 된다.

주목할 점은 라이프사이클의 각 단계에서 호출되는 메소드는 서로 다른 스레드상에서 실행된다는 것이다. JVM이 생성한 main 스레드가 `launch()`를 실행하면 다음과 같은 이름을 가진 2개의 스레드를 생성하고 시작시킨다.

- ① **JavaFX-Launcher**: `init()` 실행
- ② **JavaFX Application Thread**: 메인 클래스 기본 생성자, `start()` 및 `stop()` 실행

JavaFX 애플리케이션에 윈도우(Stage)를 비롯한 UI 생성 및 수정 작업 그리고 이벤트 처리 등은 모두 JavaFX Application Thread가 관장한다. 그 이유는 JavaFX API는 스레드에 안전하지 않아서 멀티 스레드가 동시에 UI를 생성하거나 수정하게 되면 문제가 발생하기 때문이다.

그래서 JavaFX Application Thread만 UI를 생성하거나 수정할 수 있도록 되어 있고, 다른 스레드가 UI에 접근하게 되면 예외가 발생한다. `init()` 메소드에서 UI를 생성하는 코드를 작성하면 안 되는데, 그 이유는 `init()` 메소드가 JavaFX-Launcher 스레드에서 실행되기 때문이다.

다음 예제는 기본 생성자, `init()`, `start()`, `stop()` 메소드가 어떤 스레드상에서 실행되는지 보여 준다.

>>> AppMain.java

```
1  package sec02.exam02_application_lifecycle;
2
3  import javafx.application.Application;
4  import javafx.stage.Stage;
5
6  public class AppMain extends Application {
7      public AppMain() {
8          System.out.println(Thread.currentThread().getName()+" : AppMain() 호출");
9      }
10
11     @Override
12     public void init() throws Exception {
13         System.out.println(Thread.currentThread().getName()+" : init() 호출");
14     }
15
16     @Override
17     public void start(Stage primaryStage) throws Exception {
18         System.out.println(Thread.currentThread().getName()+" : start() 호출");
19         primaryStage.show();
20     }
21
22     @Override
23     public void stop() throws Exception {
24         System.out.println(Thread.currentThread().getName()+" : stop() 호출");
25     }
26 }
```

```

27     public static void main(String[] args) {
28         System.out.println(Thread.currentThread().getName()+" : main() 호출");
29         launch(args);
30     }
31 }

```

실행 결과

```

main: main() 호출
JavaFX Application Thread: AppMain() 호출
JavaFX-Launcher: init() 호출
JavaFX Application Thread: start() 호출
JavaFX Application Thread: stop() 호출

```

메인 클래스 실행 매개값 얻기

init() 메소드의 역할은 메인 클래스의 실행 매개값을 얻어 애플리케이션의 초기값으로 이용할 수 있도록 하는 것이다. 메인 클래스를 실행할 때 실행 매개값을 다음과 같이 주었다고 가정해 보자.

```
java AppMain --ip=192.168.0.5 --port=50001
```

Application.launch()는 main() 메소드의 매개값을 그대로 넘겨 받는데, 이 매개값은 init() 메소드에서 다음과 같이 얻을 수 있다.

```

Parameters params = getParameters();
Map<String, String> map = params.getNamed();
String ip = map.get("ip");
String port = map.get("port");

```

Parameters의 getNamed() 메소드는 “ip”를 키로 해서 “192.168.0.5”를 저장하고, “port”를 키로 해서 “50001”을 저장하는 Map 컬렉션을 리턴한다.

무대와 장면

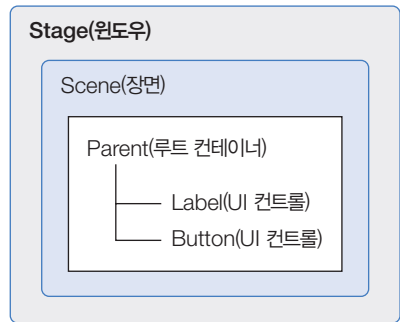
JavaFX는 윈도우를 무대^{Stage}로 표현한다. 무대는 한 번에 하나의 장면을 가질 수 있는데, JavaFX는 장면을 Scene으로 표현한다. 메인 윈도우는 start() 메소드의 primaryStage 매개값으로 전달되지만, 장면^{Scene}은 직접 생성해야 한다.

Scene을 생성하려면 UI의 루트 컨테이너인 javafx.scene.Parent가 필요하다.

```
Scene scene = new Scene(Parent root);
```

Parent는 추상 클래스이기 때문에 하위 컨테이너 클래스로 객체를 생성해서 제공해야 한다. 주로 javafx.scene.layout 패키지의 컨테이너들이 사용된다. 실제로 UI 컨트롤들이 추가되는 곳은 컨테이너이며, 컨테이너의 폭과 높이가 장면의 폭과 높이가 된다.

장면^{Scene}을 생성한 후에는 윈도우^{Stage}에 올려야 하는데, Stage의 setScene() 메소드를 사용한다. setScene() 메소드는 매개값으로 받은 Scene을 윈도우 내용으로 설정한다.



```
primaryStage.setScene(scene);
```

다음 예제는 Parent의 하위 클래스인 VBox 컨테이너를 이용해서 Scene을 생성하고 메인 윈도우(primaryStage)의 장면으로 설정했다. VBox에는 Label과 Button 컨트롤을 배치했다.

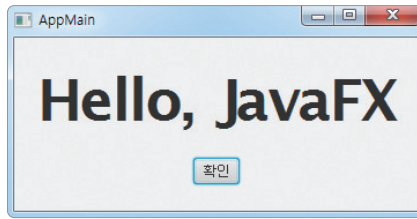
» AppMain.java

```
1 package sec02.exam03_stage_scene;
2
3 import javafx.application.Application;
4 import javafx.application.Platform;
5 import javafx.geometry.Pos;
6 import javafx.scene.Scene;
```

```

7  import javafx.scene.control.Button;
8  import javafx.scene.control.Label;
9  import javafx.scene.layout.VBox;
10 import javafx.scene.text.Font;
11 import javafx.stage.Stage;
12
13 public class AppMain extends Application {
14     @Override
15     public void start(Stage primaryStage) throws Exception {
16         VBox root = new VBox();           //Parent 하위 객체인 VBox 생성
17         root.setPrefWidth(350);           //VBox의 폭 설정
18         root.setPrefHeight(150);         //VBox의 높이 설정
19         root.setAlignment(Pos.CENTER);   //컨트롤을 중앙으로 정렬
20         root.setSpacing(20);             //컨트롤의 수직 간격
21
22         Label label = new Label();        //Label 컨트롤 생성
23         label.setText("Hello, JavaFX");   //text 속성 설정
24         label.setFont(new Font(50));      //font 속성 설정
25
26         Button button = new Button();     //Button 컨트롤 생성
27         button.setText("확인");           //text 속성 설정
28         button.setOnAction(event -> Platform.exit()); //클릭 이벤트 처리
29
30         root.getChildren().add(label);    //VBox에 Label 컨트롤 추가
31         root.getChildren().add(button);   //VBox에 Button 컨트롤 추가
32
33         Scene scene = new Scene(root);    //VBox를 루트 컨테이너로 해서 Scene 생성
34
35         primaryStage.setTitle("AppMain"); //윈도우의 제목 설정
36         primaryStage.setScene(scene);     //윈도우에 장면 설정
37         primaryStage.show();              //윈도우 보여주기
38     }
39
40     public static void main(String[] args) {
41         launch(args);
42     }
43 }

```



VBox는 수직 방향으로 컨트롤을 배치하는 컨테이너로 먼저 Label을 배치하고 그 아래에 Button을 배치했다. 28라인은 Button을 클릭했을 때 발생하는 ActionEvent를 처리한 것이다. Button을 클릭하면 Platform.exit()를 호출해서 애플리케이션을 종료하도록 했다.

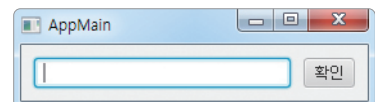
03 JavaFX 레이아웃

장면Scene에는 다양한 컨트롤이 포함되는데, 이들을 배치하는 것이 레이아웃Layout이다. 레이아웃을 작성하는 방법은 두 가지가 있다. 하나는 자바 코드로 작성하는 프로그램적 레이아웃이고, 다른 하나는 FXML로 작성하는 선언적 레이아웃이다.

프로그램적 레이아웃

프로그램적 레이아웃이란 자바 코드로 UI 컨트롤을 배치하는 것을 말한다. 자바 코드에 익숙한 개발자들이 선호하는 방식으로 컨트롤 배치, 스타일 지정, 이벤트 처리 등을 모두 자바 코드로 작성한다. 이 방법은 레이아웃이 복잡해지면 코드가 복잡해지고, 모든 것을 개발자가 직접 작성해야 하므로 디자이너와 협력해서 개발하는 것도 어렵다. 개발을 완료한 후 간단한 레이아웃 변경을 하더라도 자바 소스를 수정하고 재컴파일해야 한다는 단점도 있다.

다음 예제는 옆의 화면과 같은 레이아웃을 프로그램적 방법으로 작성한 것이다. 텍스트를 입력할 수 있는 TextField 컨트롤과 클릭할 수 있는 Button 컨트롤이 수평으로 나란히 배치되어 있기 때문에 루트 컨테이너로 HBox를 사용하였다.



>>> AppMain.java

```
1  package sec03.exam01_programmatical_layout;
2
3  import javafx.application.Application;
4  import javafx.stage.Stage;
5  import javafx.scene.layout.HBox;
6  import javafx.geometry.Insets;
7  import javafx.scene.control.TextField;
8  import javafx.scene.control.Button;
9  import javafx.scene.Scene;
10 import javafx.collections.ObservableList;
11
12 public class AppMain extends Application {
13     @Override
14     public void start(Stage primaryStage) throws Exception {
15         HBox hbox = new HBox();           //HBox 컨테이너 생성
16         hbox.setPadding(new Insets(10, 10, 10, 10)); //안쪽 여백 설정
17         hbox.setSpacing(10);              //컨트롤간의 수평 간격 설정
18
19         TextField textField = new TextField(); //TextField 컨트롤 생성
20         textField.setPrefWidth(200);         //TextField의 폭 설정
21
22         Button button = new Button();        //Button 컨트롤 생성
23         button.setText("확인");              //Button 글자 설정
24
25         ObservableList list = hbox.getChildren(); //HBox의 ObservableList 얻기
26         list.add(textField);                 //TextField 컨트롤 배치
27         list.add(button);                   //Button의 컨트롤 배치
28
29         Scene scene = new Scene(hbox);      //화면의 루트 컨테이너로 HBox 지정
30
31         primaryStage.setTitle("AppMain");   //윈도우 창 제목 설정
32         primaryStage.setScene(scene);       //윈도우 창에 화면 설정
33         primaryStage.show();               //윈도우 창 보여주기
34     }
35
36     public static void main(String[] args) {
37         launch(args);
38     }
39 }
```

FXML 레이아웃

FXML은 XML 기반의 마크업 언어로 JavaFX 애플리케이션의 UI 레이아웃을 자바 코드에서 분리해서 태그로 선언하는 방법을 제공한다. 이 방법은 안드로이드^{Android} 앱을 개발하는 방법과 유사한데, XML로 레이아웃을 작성하고, 이벤트 처리 및 애플리케이션 로직은 자바로 작성한다.

태그로 레이아웃을 정의하기 때문에 태그에 익숙한 디자이너와 협업이 쉽고, 개발 완료 후 레이아웃 변경 시 자바 소스를 수정할 필요 없이 FXML 태그만 수정하면 되므로 편리하다. 그리고 레이아웃이 비슷한 장면^{Scene}들간에 재사용이 가능하기 때문에 개발 기간을 단축시킬 수도 있다.

다음 예제는 프로그램적 레이아웃을 사용하는 이전 예제를 FXML 레이아웃으로 변경한 것이다. FXML 레이아웃은 root.fxml 파일로 다음과 같이 작성할 수 있다.

>>> root.fxml

```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <?import javafx.scene.layout.HBox?>
4  <?import javafx.geometry.Insets?>
5  <?import javafx.scene.control.*?>
6
7  <HBox xmlns:fx="http://javafx.com/fxml" > <!-- HBox 컨테이너 선언 -->
8    <padding> <!-- 안쪽 여백 설정 -->
9      <Insets top="10" right="10" bottom="10" left="10" />
10   </padding>
11   <spacing>10</spacing> <!-- 컨트롤간의 수평 간격 설정 -->
12
13   <children> <!-- 자식 컨트롤 추가 -->
14     <TextField> <!-- TextField 선언 -->
15       <prefWidth>200</prefWidth> <!-- TextField의 폭 설정 -->
16     </TextField>
17
18     <Button> <!-- Button 컨트롤 선언 -->
19       <text>확인</text> <!-- Button 글자 설정 -->
20     </Button>
21   </children>
22 </HBox>
```


루트 컨테이너인 HBox는 <HBox> 태그로 작성되고, fx 접두사에 대한 네임스페이스 선언 (xmlns:fx="http://javafx.com/fxml")이 추가되어 있는 것을 볼 수 있다. 추후 이것은 FXML 파일에 <fx:XXX> 형태의 태그 및 fx:xxx="값" 형태의 속성을 작성할 수 있다는 뜻이다.

HBox 컨테이너에 들어갈 TextField와 Button은 <children> 태그의 내용으로 각각 <TextField> 와 <Button> 태그로 작성된 것을 볼 수 있다. <padding>은 여백을 말하는데, 곧 이어서 학습한다. 다음은 자바 로직 부분을 작성한 메인 클래스이다.

>>> AppMain.java

```
1  package sec03.exam02_fxml_layout;
2
3  import javafx.application.Application;
4  import javafx.fxml.FXMLLoader;
5  import javafx.scene.Parent;
6  import javafx.scene.Scene;
7  import javafx.stage.Stage;
8
9  public class AppMain extends Application {
10     @Override
11     public void start(Stage primaryStage) throws Exception {
12         Parent root = FXMLLoader.load(getClass().getResource("root.fxml"));
13         Scene scene = new Scene(root);
14
15         primaryStage.setTitle("AppMain");
16         primaryStage.setScene(scene);
17         primaryStage.show();
18     }
19
20     public static void main(String[] args) {
21         launch(args);
22     }
23 }
```

메인 클래스는 start() 메소드에서 FXML 레이아웃 파일을 읽고 선언된 내용을 객체화해야 한다. 이것을 FXML 로딩이라고 한다. 이때 FXMLLoader의 load() 메소드가 사용된다. load() 메소드는 정적과 인스턴스 모두 있다.

다음은 정적 load() 메소드로 FXML 레이아웃 파일을 로딩하는 코드이다.

```
Parent root = FXMLLoader.load(getClass().getResource("root.fxml"));
```

load()의 매개값은 FXML 파일의 위치 정보가 있는 URL 객체이다. FXML 파일이 메인 클래스와 동일한 패키지에 있을 경우, getClass()로 메인 클래스 타입을 얻고 상대 경로를 이용해서 getResource()로 FXML 파일을 찾아 URL 객체를 얻을 수 있다.

다음은 인스턴스 load() 메소드로 FXML 레이아웃 파일을 로딩하는 코드이다.

```
FXMLLoader loader = new FXMLLoader(getClass().getResource("root.fxml"));
Parent root = (Parent)loader.load();
```

정적 또는 인스턴스 load() 메소드가 리턴하는 타입은 Parent 타입인데, 실제 객체는 FXML 파일에서 루트 태그로 선언된 컨테이너이다. 위 예제에서 FXML 파일의 루트 태그는 <HBox>이므로 다음과 같이 타입 변환이 가능하다.

```
HBox root = (HBox) FXMLLoader.load(getClass().getResource("root.fxml"));
```

FXML 파일을 로딩해서 Parent 객체를 얻었다면 이것을 가지고 다음과 같이 장면(Scene) 객체를 생성하면 된다.

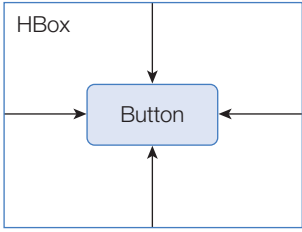
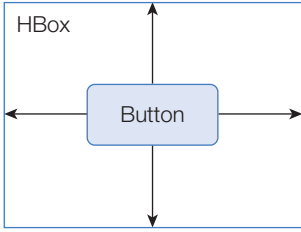
```
Scene scene = new Scene(root);
```

레이아웃 여백: 패딩과 마진

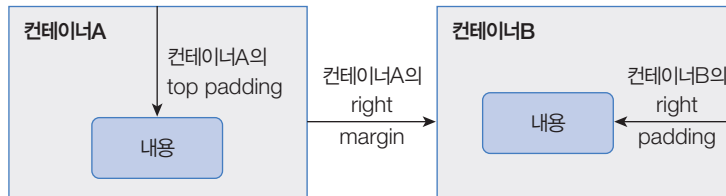
컨트롤을 보기 좋게 배치하기 위해서 여백이 거의 필수적으로 들어간다. 여백은 패딩(padding)과 마진(margin)이 있는데, 패딩은 안쪽 여백을 말하고 마진은 바깥 여백을 말한다.

패딩은 컨테이너의 setPadding() 메소드를 사용해서 설정하는 반면, 마진은 바깥 컨테이너의 setMargin() 메소드를 사용해야 한다. 예를 들어 Button이 HBox에 포함되어 있을 때 HBox에

서 패딩을 50으로 주는 것과 Button에서 마진을 50으로 주는 것은 동일한 결과를 얻지만 코드는 다르다.

구분	HBox의 패딩	Button의 마진
개념		
자바 코드	<pre>HBox hbox = new HBox(); hbox.setPadding(new Insets(50));</pre>	<pre>Button button = new Button(); HBox.setMargin(button, new Insets(50));</pre>
FXML 태그	<pre><HBox> <padding> <Insets topRightBottomLeft="50"/> </padding> </HBox></pre>	<pre><Button> <HBox.margin> <Insets topRightBottomLeft="50"/> </HBox.margin> </Button></pre>

마진과 패딩은 적용하는 위치에 따라 top, right, bottom, left로 구분된다.

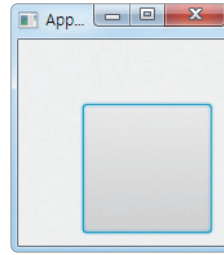


마진과 패딩값은 Insets 객체로 제공해야 하는데, 다음과 같이 생성한다. 생성자 매개값의 순서는 top을 시작으로 시계방향으로 나열되어 있어 쉽게 기억할 수 있을 것이다.

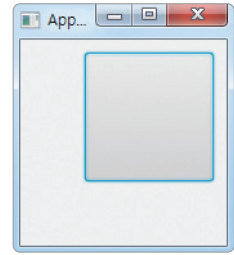
```
//top, right, bottom, left를 모두 동일한 값으로 설정할 때
new Insets(double topRightBottomLeft);

//top, right, bottom, left를 다른 값으로 설정할 때
new Insets(double top, double right, double bottom, double left)
```

다음 예제를 보면 14~17라인은 HBox의 안쪽 여백 패딩으로 top과 left는 50, right와 bottom은 10으로 설정했다. 그리고 20~23라인은 Button의 바깥 여백 마진으로 top과 right는 10, bottom과 left는 50으로 설정했다. 한쪽을 주석으로 처리해놓고 각각 실행해 보자.



HBox Padding



Button Margin

>>> AppMain.java

```

1  package sec03.exam03_margin_padding;
2
3  import javafx.application.Application;
4  import javafx.geometry.Insets;
5  import javafx.scene.Scene;
6  import javafx.scene.control.Button;
7  import javafx.scene.layout.HBox;
8  import javafx.stage.Stage;
9
10 public class AppMain extends Application {
11     @Override
12     public void start(Stage primaryStage) throws Exception {
13         //패딩 설정-----
14         /*HBox hbox = new HBox();
15         hbox.setPadding(new Insets(50, 10, 10, 50));
16         Button button = new Button();
17         button.setPrefSize(100, 100);*/
18
19         //마진 설정-----
20         HBox hbox = new HBox();
21         Button button = new Button();
22         button.setPrefSize(100, 100);
23         HBox.setMargin(button, new Insets(10, 10, 50, 50));
24
25         hbox.getChildren().add(button);
26
27         Scene scene = new Scene(hbox);
28
29         primaryStage.setTitle("AppMain");

```

```

30     primaryStage.setScene(scene);
31     primaryStage.show();
32 }
33
34 public static void main(String[] args) {
35     launch(args);
36 }
37 }

```

FXML 태그와 자바 코드 매핑

FXML로 선언된 태그는 자바 코드로 변환되어 실행되기 때문에 자바 코드와 매핑 관계가 존재한다. 이 매핑 관계만 잘 이해하면 JavaFX API 도큐먼트를 참조해서 FXML 태그를 쉽게 작성할 수 있다. 다음은 프로그램적 레이아웃 자바 코드와 FXML 레이아웃 태그를 매핑시킨 표이다.

프로그램적 레이아웃 자바 코드	FXML 레이아웃 태그
<pre> HBox hbox = new HBox(); hbox.setPadding(new Insets(10,10,10,10)); hbox.setSpacing(10); </pre>	<pre> <HBox xmlns:fx="http://javafx.com/fxml"> <padding> <Insets top="10" right="10" bottom="10" left="10"/> </padding> <spacing>10</spacing> </HBox> </pre>
<pre> TextField textField = new TextField(); textField.setPrefWidth(200); </pre>	<pre> <TextField> <prefWidth>200</prefWidth> </TextField> </pre>
<pre> Button button = new Button(); button.setText("확인"); </pre>	<pre> <Button > <text>확인</text> </Button> </pre>
<pre> ObservableList list = hbox.getChildren(); list.add(textField); list.add(button); </pre>	<pre> <children> <TextField>...</TextField> <Button >...</Button> </children> </pre>

FXML은 XML 기반의 마크업 언어이기 때문에 XML 작성 규칙을 잘 지켜서 작성해야 한다. FXML 태그를 무조건 외우기보다는 자바 코드와 매핑되는 FXML 작성 규칙을 이해하면 FXML을 빨리 익

힐 수 있다. 그럼 FXML 작성 규칙을 자세히 살펴보기로 하자.

패키지 선언

자바 코드의 패키지 선언과 매핑되는 FXML 태그는 `<?import?>`이다. 클래스 하나를 import하는 방법과 같은 패키지의 모든 클래스를 import하는 방법은 다음과 같다.

자바 코드	FXML 태그
<code>import javafx.scene.layout.HBox;</code>	<code><?import javafx.scene.layout.HBox?></code>
<code>import javafx.scene.control.*;</code>	<code><?import javafx.scene.control.*?></code>

`<?import?>` 태그를 작성하는 위치는 정해져 있다. XML 선언 태그인 `<?xml version="1.0" encoding="UTF-8"?>`과 루트 컨테이너 태그 사이이다.

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.layout.HBox?>
<?import javafx.scene.control.*?>

<루트컨테이너 xmlns:fx="http://javafx.com/fxml" >
    ...
</루트컨테이너>
```

FXML 태그의 이름은 하나의 JavaFX API 클래스 이름과 매핑되기 때문에 해당 클래스가 존재하는 패키지를 반드시 `<?import?>` 태그로 선언해야 한다. 그렇지 않으면 FXML을 로딩할 때 `not a valid type`이라는 메시지와 함께 `javafx.fxml.LoadException`이 발생한다.

태그 선언

FXML 태그는 `< 와 >` 사이에 태그 이름을 작성한 것인데, 반드시 시작 태그가 있으면 끝 태그도 있어야 한다. 그렇지 않으면 `javax.xml.stream.XMLStreamException` 예외가 발생한다.

```
<태그이름> ... </태그이름>
```

시작 태그와 끝 태그 사이에는 태그 내용이 작성되는데, 태그 내용이 없을 경우에는 다음과 같이 시작 태그 끝에 />를 붙이고 끝 태그를 생략할 수 있다.

```
<태그이름/>
```

태그 이름은 JavaFX의 클래스명이거나, Setter의 메소드명이 될 수 있다. 다음 표에서 Button 컨트롤을 자바 코드로 작성한 것과 FXML 태그로 작성한 것을 비교해보면 쉽게 이해가 될 것이다.

자바 코드	FXML
<pre>Button button = new Button(); button.setText("확인");</pre>	<pre><Button> <text>확인</text> </Button></pre>

속성 선언

FXML 태그는 다음과 같은 속성을 가질 수 있다. 속성값은 큰따옴표(") 또는 작은따옴표(')로 반드시 감싸야 한다. 그렇지 않으면 `javax.xml.stream.XMLStreamException` 예외가 발생한다.

```
<태그이름 속성명= "값" 속성명= <값> ... </태그이름>
```

속성명은 Setter 메소드명이 오는데, 모든 Setter가 사용될 수 있는 것은 아니고 기본 타입(boolean, byte, short, char, int, long, float, double)의 값을 세팅하거나, String(문자열)을 세팅하는 Setter만 쓸 수 있다. 예를 들어 Button의 글자를 설정할 때 `setText()` 메소드를 사용하는데, 매개값이 문자열이므로 다음과 같이 `text` 속성으로 작성할 수 있다.

자바 코드	FXML (Setter 태그)	FXML (Setter 속성)
<pre>Button button = new Button(); button.setText("확인");</pre>	<pre><Button > <text>확인</text> </Button></pre>	<pre><Button text="확인"/></pre>

객체 선언

Setter 메소드가 기본 타입과 String 타입이 아닌 다른 타입의 객체를 매개값으로 갖는다면 속성으로 작성할 수 없고, 태그로 작성해야 한다. 이때 매개값인 객체를 태그로 선언하는 방법을 알아보자.

객체 선언

1) <클래스 속성="값"/>

일반적으로 다음과 같이 클래스명으로 태그를 작성하면 new 연산자로 기본 생성자를 호출해서 객체가 생성된다.

```
<클래스>
```

만약 생성자에 매개변수가 있고, 매개변수에 `@NamedArg(javafx.beans.NamedArg)` 어노테이션이 적용되어 있다면 속성명이나 자식 태그로 작성할 수 있다.

```
<클래스 속성="값">
```

```
<클래스>
  <매개변수>값</매개변수>
</클래스>
```

예를 들어 HBox의 패딩을 설정할 때 `setPadding(Insets value)` 메소드를 사용하는데, Insets는 기본 생성자가 없고 `Insets(double topRightBottomLeft)` 또는 `Insets(double top, double right, double bottom, double left)`만 있다. 이 경우 Insets 객체를 FXML로 선언하면 다음과 같다.

자바 코드	FXML
<pre>HBox hbox = new HBox(); hbox.setPadding(new Insets(10,10,10,10));</pre>	<pre><HBox> <padding> <Insets top="10" right="10" bottom="10" left="10"/> </padding> </HBox></pre>

2) <클래스 fx:value="값"/>

클래스가 `valueOf(String)` 메소드를 제공해서 객체를 생성하는 경우가 있다. 예를 들어 `String`, `Integer`, `Double`, `Boolean` 클래스는 `valueOf(String)`을 호출해서 객체를 생성한다. 이 경우 다음과 같이 FXML 태그를 작성할 수 있다.

```
<클래스 fx:value= "값" />
```

예를 들어 `String`, `Integer`, `Double`, `Boolean` 객체를 FXML로 선언하면 다음과 같다.

자바 코드	FXML
<code>String.valueOf("Hello, World");</code> <code>Integer.valueOf("1");</code> <code>Double.valueOf("1.0");</code> <code>Boolean.valueOf("false");</code>	<code><String fx:value="Hello, World!"/></code> <code><Integer fx:value="1"/></code> <code><Double fx:value="1.0"/></code> <code><Boolean fx:value="false"/></code>

3) <클래스 fx:constant="상수"/>

클래스에 정의된 상수값을 얻고 싶을 경우에는 다음과 같이 FXML 태그를 작성할 수 있다.

```
<클래스 fx:constant= "상수" />
```

예를 들어 `Double.MAX_VALUE` 상수값을 `Button` 컨트롤의 `maxWidth` 속성값으로 설정할 경우 다음과 같이 FXML로 선언할 수 있다.

자바 코드	FXML
<code>Button button = new Button();</code> <code>button.setMaxWidth(</code> <code>Double.MAX_VALUE</code> <code>);</code>	<code><Button></code> <code><maxWidth></code> <code><Double fx:constant="MAX_VALUE"/></code> <code></maxWidth></code> <code></Button></code>

4) <클래스 fx:factory="정적메소드">

어떤 클래스는 new 연산자로 객체를 생성할 수 없고, 정적 메소드로 객체를 얻어야 하는 경우도 있다. 이 경우 다음과 같이 FXML 태그를 작성할 수 있다.

```
<클래스 fx:factory= "정적메소드">
```

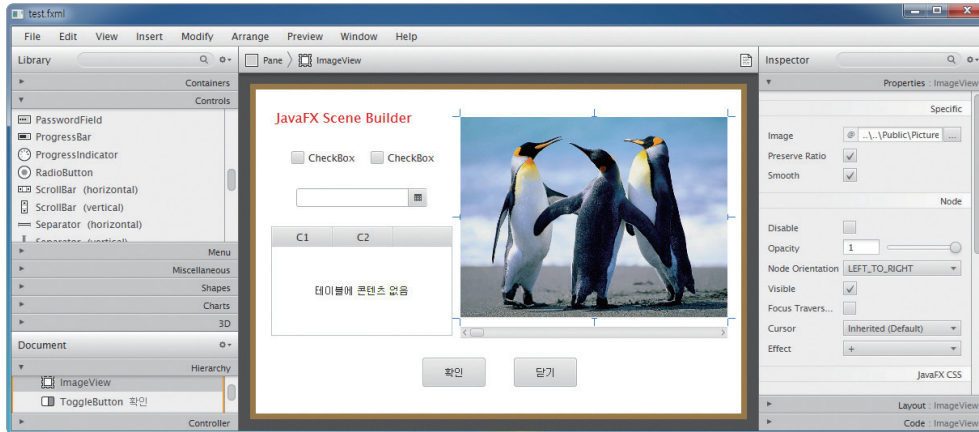
예를 들어 ComboBox의 setItems(ObservableList<T> value) 메소드는 ObservableList 인터페이스의 구현 객체를 매개값으로 가지는데, ObservableList 구현 객체는 FXCollections의 정적 메소드인 observableArrayList(E... items) 메소드로 얻을 수 있다. 그래서 다음과 같이 FXML을 작성해야 한다.

자바 코드	FXML
<pre>ComboBox combo = new ComboBox(); combo.setItems(FXCollections.observableArrayList("공개", "비공개"));</pre>	<pre><ComboBox> <items> <FXCollections fx:factory="observ ableArrayList"> <String fx:value="공개"/> <String fx:value="비공개"/> </FXCollections> </items> </ComboBox></pre>

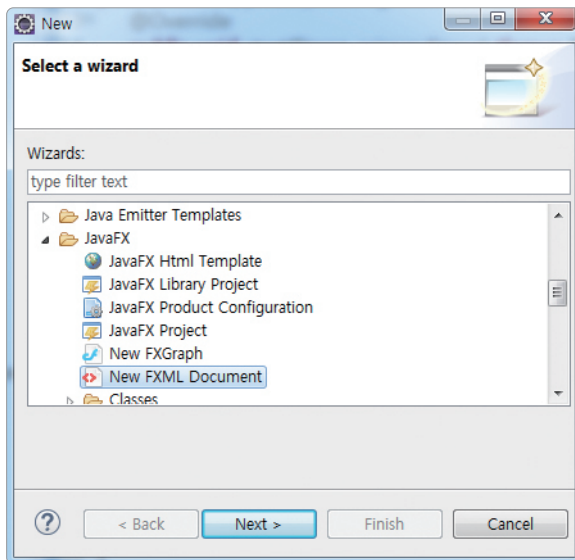
Scene Builder 사용

지금까지 FXML 태그를 작성하는 방법을 학습했지만, 레이아웃이 복잡해질수록 태그로 직접 작성한다는 것은 힘든 작업이다. 01절에서 설치한 Scene Builder를 이용해서 드래그 앤 드롭 방식으로 화면을 디자인하면 자동으로 FXML 파일이 생성되므로 매우 편리하다.

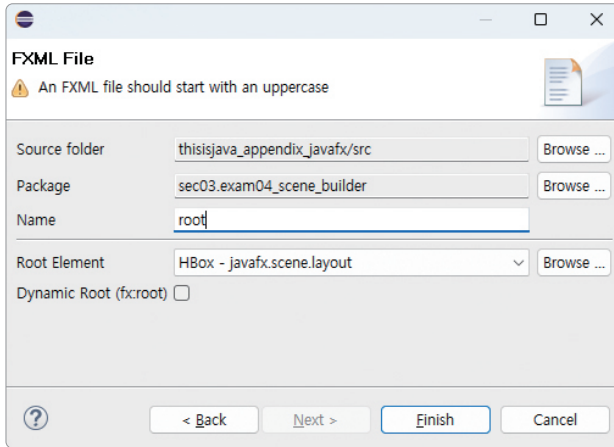
실제로 JavaFX 애플리케이션 개발은 전체 레이아웃 작업은 Scene Builder로 빠르게 진행하고, 간단한 수정 작업은 FXML 태그로 직접 작성하는 것이 일반적이다. 다음은 Scene Builder에서 FXML 레이아웃을 작성하는 모습을 보여준다.



이클립스에서 JavaFX Scene Builder를 실행하려면 먼저 FXML 파일을 생성해야 한다. Package Explorer 뷰에서 sec03.exam04_scene_builder 패키지를 생성하고, 마우스 오른쪽 버튼을 클릭한 후 [New] - [Other] - [JavaFX]로 들어가 [New FXML Document]를 선택한다.



FXML File 대화상자에서 Name 입력란에 생성할 파일명을 입력하고 Root Element에 루트 컨테이너가 될 클래스를 선택한 후 [Finish] 버튼을 클릭하면 FXML 파일이 생성된다.



다음은 Root Element를 HBox로 선택하고 생성한 FXML 파일 내용을 보여준다.

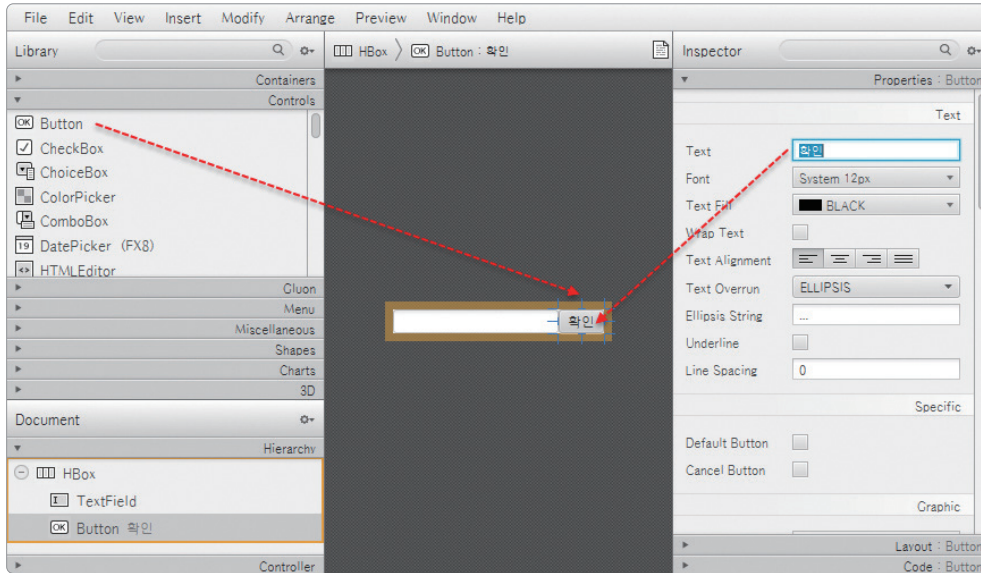
```
<?xml version="1.0" encoding="UTF-8"?>

<import javafx.scene.layout.HBox?>

<HBox xmlns:fx="http://javafx.com/fxml">
  <!-- TODO Add Nodes -->
</HBox>
```

xmlns:fx="http://javafx.com/fxml/1"로
되어있어도 상관없음, 1은 FXML 버전을 뜻함

생성된 FXML 파일을 Scene Builder로 편집하려면 이클립스 Package Explorer 뷰에서 FXML 파일을 선택하고 마우스 오른쪽 버튼을 클릭한 후 [Open with SceneBuilder]를 선택하면 된다. Scene Builder 화면의 왼쪽에 있는 [Containers]와 [Controls] 메뉴에서 원하는 항목을 드래그해 중앙의 디자인 영역에 드롭시키면 배치가 된다. 배치된 컨테이너와 컨트롤의 속성은 화면 오른쪽에 있는 [Properties]와 [Layout] 메뉴에서 설정할 수 있다.



레이아웃 디자인이 완성했다면 상단 메뉴에서 [File] - [Save]를 선택해서 FXML 파일로 저장한다.

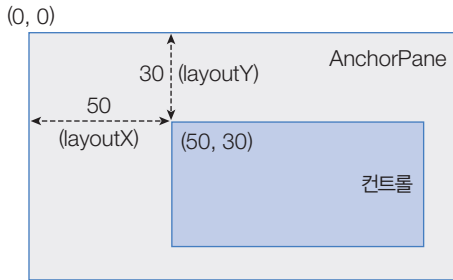
04 JavaFX 컨테이너

레이아웃을 작성할 때 컨트롤들을 쉽게 배치할 수 있도록 도와주는 클래스가 컨테이너이다. javafx.scene.layout 패키지에는 다양한 컨테이너 클래스들이 존재한다. 접미사가 Pane으로 끝나는 클래스는 모두 컨테이너이고, 그 이외에 Hbox, VBox가 있다.

컨테이너	설명
AnchorPane	컨트롤을 좌표를 이용해서 배치하는 레이아웃
BorderPane	위, 아래, 오른쪽, 왼쪽, 중앙에 컨트롤을 배치하는 레이아웃
FlowPane	행으로 배치하되 공간이 부족하면 새로운 행에 배치하는 레이아웃
GridPane	그리드로 배치하되 셀의 크기가 고정적이지 않은 레이아웃
StackPane	컨트롤을 겹쳐 배치하는 레이아웃
TilePane	그리드로 배치하되 고정된 셀의 크기를 갖는 레이아웃
HBox	수평으로 배치하는 레이아웃
VBox	수직으로 배치하는 레이아웃

AnchorPane 컨테이너

AnchorPane 컨테이너는 좌표를 이용하여 AnchorPane의 좌상단(0, 0)을 기준으로 컨트롤을 배치한다. 컨트롤 좌표는 좌상단(layoutX, layoutY) 값을 말하는데, (0, 0)으로부터 떨어진 거리를 의미한다.



AnchorPane에서 사용할 수 있는 주요 설정은 다음과 같다.

태그 및 속성	설명	적용
prefWidth	폭을 설정	AnchorPane
prefHeight	높이를 설정	AnchorPane
layoutX	컨트롤의 X 좌표	컨트롤
layoutY	컨트롤의 Y 좌표	컨트롤
<children>	컨트롤을 포함	AnchorPane

AnchorPane 컨테이너는 JavaFX Scene Builder를 사용해서 디자인하는 것이 좋다. 눈으로 거리를 확인해서 컨트롤을 드롭시킬 수 있기 때문이다. 프로그램적 레이아웃 방법은 직접 코드상에 좌표 값을 입력해야 하기 때문에 매우 불편하다.

다음 예제는 AnchorPane루트 컨테이너를 사용해서 로그인 레이아웃을 디자인한 것이다.

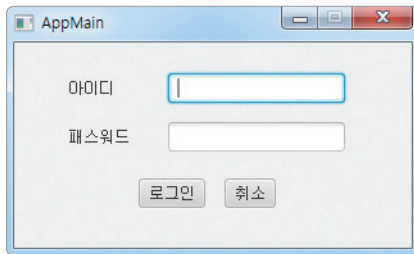
```
>>> root.fxml

1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <?import javafx.scene.layout.*?>
```

```

4  <?import javafx.scene.control.*?>
5
6  <AnchorPane xmlns:fx="http://javafx.com/fxml" prefHeight="150.0"
    prefWidth="300.0" >
7      <children>
8          <Label layoutX="42.0" layoutY="28.0" text="아이디" />
9          <Label layoutX="42.0" layoutY="66.0" text="패스워드" />
10         <TextField layoutX="120.0" layoutY="24.0" />
11         <PasswordField layoutX="120.0" layoutY="62.0" />
12         <Button layoutX="97.0" layoutY="106.0" text="로그인" />
13         <Button layoutX="164.0" layoutY="106.0" text="취소" />
14     </children>
15 </AnchorPane>

```



AnchorPane에 포함될 컨트롤은 <children> 태그의 내용으로 작성된다. AnchorPane을 사용해서 컨트롤을 좌표로 배치하면 윈도우 창이 줄거나 늘어날 경우 컨트롤의 재배치가 일어나지 않는다. 따라서 AnchorPane으로 배치할 경우에는 윈도우 창의 크기를 변경할 수 없도록 `primaryStage.setResizable(false)`;를 추가하는 것이 좋다.

>>> AppMain.java

```

1  package sec04.exam01_anchorpane;
2
3  import javafx.application.Application;
4  import javafx.fxml.FXMLLoader;
5  import javafx.scene.Parent;

```

```

6  import javafx.scene.Scene;
7  import javafx.stage.Stage;
8
9  public class AppMain extends Application {
10     @Override
11     public void start(Stage primaryStage) throws Exception {
12         Parent root = FXMLLoader.load(getClass().getResource("root.fxml"));
13         Scene scene = new Scene(root);
14
15         primaryStage.setTitle("AppMain");
16         primaryStage.setScene(scene);
17         primaryStage.show();
18         primaryStage.setResizable(false);
19     }
20
21     public static void main(String[] args) {
22         launch(args);
23     }
24 }

```

HBox와 VBox 컨테이너

HBox와 VBox는 수평과 수직으로 컨트롤을 배치하는 컨테이너이다. HBox와 VBox는 자식 컨트롤의 크기를 재조정하는데, HBox는 컨트롤의 높이를 확장하고, 컨트롤의 폭은 유지한다. VBox는 컨트롤의 폭을 확장하고 컨트롤의 높이는 유지한다.

단, 크기 조정이 가능한 컨트롤만 자동 확장되는데, 기본 Button의 경우는 크기 조정이 되지 않는다. 그 이유는 `maxWidth`와 `maxHeight`가 `-1.0`을 가지기 때문이다. 크기 조정이 가능하도록 하려면 다음과 같이 `maxWidth`와 `maxHeight`를 변경하면 된다.

```

<Button text="button">
  <maxWidthXDouble fx:constant="MAX_VALUE"/></maxWidth>
  <maxHeightXDouble fx:constant="MAX_VALUE"/></maxHeight>
</Button>

```


HBox에서 컨트롤의 높이를 확장하고 싶지 않다면 fillHeight 속성을 false로 설정하면 되고, VBox에서 컨트롤의 폭을 확장하고 싶지 않다면 fillWidth 속성을 false로 설정하면 된다. HBox와 VBox에서 사용할 수 있는 주요 설정은 다음과 같다.

태그 및 속성	설명	적용
prefWidth	폭을 설정	HBox, VBox
prefHeight	높이를 설정	HBox, VBox
alignment	컨트롤의 정렬을 설정	HBox, VBox
spacing	컨트롤의 간격을 설정	HBox, VBox
fillWidth	컨트롤의 폭 확장 여부 설정	VBox
fillHeight	컨트롤의 높이 확장 여부 설정	HBox
<children>	컨트롤을 포함	HBox, VBox
<HBox.hgrow> <Priority fx:constant="ALWAYS"/> </HBox.hgrow>	HBox의 남은 폭을 채움	컨트롤
<VBox.vgrow> <Priority fx:constant="ALWAYS"/> </VBox.vgrow>	VBox의 남은 높이를 채움	컨트롤

다음 예제는 VBox로 ImageView 컨트롤과 HBox 컨테이너를 수직으로 배치하고, HBox 안에는 두 개의 버튼을 수평으로 배치했다. [다음] 버튼은 HBox의 남은 폭을 채우도록 HBox의 hgrow 속성을 설정했다.

>>> root.fxml

```

1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <?import javafx.geometry.*?>
4  <?import javafx.scene.image.*?>
5  <?import javafx.scene.layout.*?>
6  <?import javafx.scene.control.*?>
7  <?import java.lang.*?>
8
9  <VBox xmlns:fx="http://javafx.com/fxml">

```

```

10     <padding>
11         <Insets bottom="10.0" left="10.0" right="10.0" top="10.0" />
12     </padding>
13
14     <children>
15         <ImageView fitWidth="200" preserveRatio="true" />
16         <image>
17             <Image url="@images/javafx.jpg" />
18         </image>
19     </ImageView>
20
21     <HBox alignment="CENTER" spacing="20.0">
22         <children>
23             <Button text="이전" />
24             <Button text="다음" />
25             <HBox.hgrow>Priority fx:constant="ALWAYS"/</HBox.hgrow>
26             <maxWidth>Double fx:constant="MAX_VALUE"/</maxWidth>
27         </children>
28     </HBox>
29     <VBox.margin>
30         <Insets top="10.0" />
31     </VBox.margin>
32 </HBox>
33 </children>
34 </VBox>

```

그림의 비율에 맞게
높이를 설정

현재 경로를 기준으로
상대 경로로 파일 지정

오른쪽 남은 공간을 버튼
이 모두 채우도록 설정

버튼의 폭을 자동으로
확장하기 위해 설정

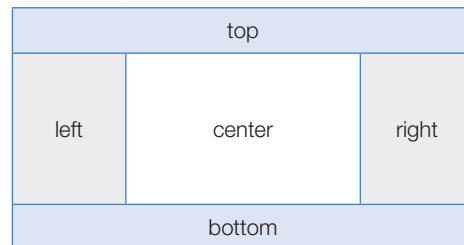


>>> AppMain.java

```
1  package sec04.exam02_hbox_vbox;
2
3  import javafx.application.Application;
4  import javafx.fxml.FXMLLoader;
5  import javafx.scene.Scene;
6  import javafx.scene.layout.VBox;
7  import javafx.stage.Stage;
8
9  public class AppMain extends Application {
10     @Override
11     public void start(Stage primaryStage) throws Exception {
12         VBox root = (VBox)FXMLLoader.load(
13             getClass().getResource("root.fxml")
14         );
15         Scene scene = new Scene(root);
16
17         primaryStage.setTitle("AppMain");
18         primaryStage.setScene(scene);
19         primaryStage.show();
20     }
21
22     public static void main(String[] args) {
23         launch(args);
24     }
25 }
```

BorderPane 컨테이너

BorderPane은 top, bottom, left, right, center 셀에 컨트롤을 배치하는 컨테이너이다. 컨트롤만 배치하는 것이 아니라 다른 컨테이너도 배치할 수 있기 때문에 다양한 레이아웃을 만들어 낼 수 있다. 주의할 점은 각 셀에는 하나의 컨트롤 또는 컨테이너만 배치할 수 있다.



다음은 BorderPane에서 사용할 수 있는 태그 및 속성들이다.

태그 및 속성	설명	적용
prefWidth	폭을 설정	BorderPane
prefHeight	높이를 설정	BorderPane
<top>	top에 배치될 컨트롤을 포함	BorderPane
<bottom>	bottom에 배치될 컨트롤을 포함	BorderPane
<right>	right에 배치될 컨트롤을 포함	BorderPane
<left>	left에 배치될 컨트롤을 포함	BorderPane
<center>	center에 배치될 컨트롤을 포함	BorderPane

BorderPane의 특징은 top, bottom, left, right에 컨트롤을 배치하지 않으면 center에 배치된 컨트롤이 top, bottom, left, right까지 확장된다.

다음은 BorderPane의 top에ToolBar를 배치하고 center에는 TextArea를, bottom에는 다시 BorderPane을 배치한 것이다.

>>> root.fxml

```

1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <?import javafx.scene.layout.*?>
4  <?import javafx.scene.control.*?>
5
6  <BorderPane xmlns:fx="http://javafx.com/fxml" prefHeight="200.0"
    prefWidth="300.0" >
7      <top>
8          <ToolBar>
9              <items>
10                 <Button text="Button" />
11                 <Button text="Button" />
12             </items>
13         </ToolBar>
14     </top>
15     <center>
16         <TextArea/>
17     </center>

```

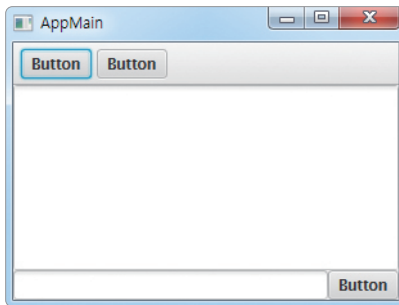
left, right까지 확장

```

18     <bottom>
19         <BorderPane>
20             <center>
21                 <TextField/>
22             </center>
23             <right>
24                 <Button text="Button"/>
25             </right>
26         </BorderPane>
27     </bottom>
28 </BorderPane>

```

top, bottom, left까지 확장



>>> AppMain.java

```

1  package sec04.exam03_borderpane;
2
3  import javafx.application.Application;
4  import javafx.fxml.FXMLLoader;
5  import javafx.scene.Parent;
6  import javafx.scene.Scene;
7  import javafx.stage.Stage;
8
9  public class AppMain extends Application {
10     @Override
11     public void start(Stage primaryStage) throws Exception {
12         Parent root = FXMLLoader.load(getClass().getResource("root.fxml"));

```

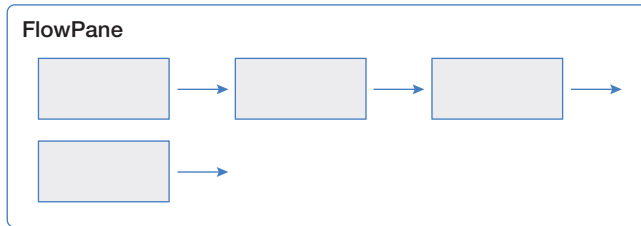
```

13     Scene scene = new Scene(root);
14
15     primaryStage.setTitle("AppMain");
16     primaryStage.setScene(scene);
17     primaryStage.show();
18 }
19
20 public static void main(String[] args) {
21     launch(args);
22 }
23 }

```

FlowPane 컨테이너

FlowPane은 행으로 컨트롤을 배치하되 공간이 부족하면 새로운 행에 배치하는 컨테이너이다.



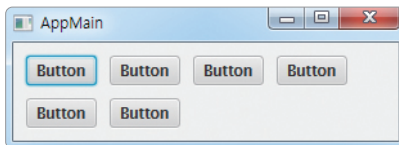
다음은 FlowPane에서 사용할 수 있는 태그와 속성들이다.

태그 및 속성	설명	적용
prefWidth	폭을 설정	FlowPane
prefHeight	높이를 설정	FlowPane
hgap	컨트롤의 수평 간격을 설정	FlowPane
vgap	컨트롤의 수직 간격을 설정	FlowPane
<children>	컨트롤을 포함	FlowPane

다음 예제는 FlowPane 컨테이너에 여섯 개의 Button을 배치한 것인데, 버튼 간의 수평 간격과 수직 간격을 주기 위해 hgap과 vgap을 10으로 설정하였다.

>>> root.fxml

```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <?import javafx.scene.layout.*?>
4  <?import javafx.geometry.*?>
5  <?import javafx.scene.control.*?>
6
7  <FlowPane xmlns:fx="http://javafx.com/fxml"
8    prefWidth="300.0" prefHeight="70.0" hgap="10.0" vgap="10.0" >
9    <padding>
10     <Insets bottom="10.0" left="10.0" right="10.0" top="10.0" />
11    </padding>
12
13    <children>
14      <Button text="Button" />
15      <Button text="Button" />
16      <Button text="Button" />
17      <Button text="Button" />
18      <Button text="Button" />
19      <Button text="Button" />
20    </children>
21  </FlowPane>
```



>>> AppMain.java

```
1  package sec04.exam04_flowpane;
2
3  import javafx.application.Application;
4  import javafx.fxml.FXMLLoader;
5  import javafx.scene.Parent;
```

```

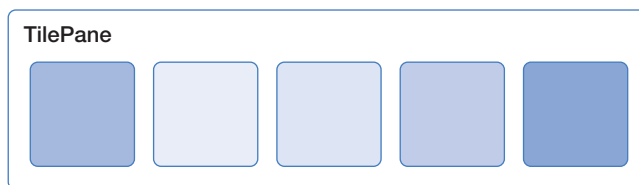
6  import javafx.scene.Scene;
7  import javafx.stage.Stage;
8
9  public class AppMain extends Application {
10     @Override
11     public void start(Stage primaryStage) throws Exception {
12         Parent root = FXMLLoader.load(getClass().getResource("root.fxml"));
13         Scene scene = new Scene(root);
14
15         primaryStage.setTitle("AppMain");
16         primaryStage.setScene(scene);
17         primaryStage.show();
18     }
19
20     public static void main(String[] args) {
21         launch(args);
22     }
23 }

```

FlowPane을 잘 이해하려면 윈도우 창을 늘렸다가 줄여 보면 되는데, 오른쪽에 배치될 공간이 부족할 경우에는 새로운 행에 컨트롤이 배치되는 것을 볼 수 있다.

TilePane 컨테이너

TilePane은 그리드로 컨트롤을 배치하되 고정된 셀(타일) 크기를 갖는 컨테이너이다. FlowPane과 마찬가지로 오른쪽에 컨트롤을 배치할 공간이 부족하면 새로운 행에 컨트롤을 배치한다.



다음은 TilePane에서 사용할 수 있는 태그와 속성들이다.

태그 및 속성	설명	적용
prefWidth	폭을 설정	TilePane
prefHeight	높이를 설정	TilePane
prefTileWidth	타일의 폭을 설정	TilePane
prefTileHeight	타일의 높이를 설정	TilePane
<children>	컨트롤을 포함	TilePane

다음은 여러 개의 ImageView를 TilePane에 배치한 예제이다. 셀의 크기를 100×100으로 지정하기 위해 prefTileHeight="100" prefTileWidth="100"으로 지정했다.

>> root.fxml

```

1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <?import javafx.scene.layout.*?>
4  <?import javafx.scene.image.*?>
5
6  <TilePane xmlns:fx="http://javafx.com/fxml" prefTileHeight="100"
   prefTileWidth="100" >
7    <children>
8      <ImageView>
9        <image><Image url="@images/fruit1.jpg" /></image>
10     </ImageView>
11     <ImageView>
12       <image><Image url="@images/fruit2.jpg" /></image>
13     </ImageView>
14     <ImageView>
15       <image><Image url="@images/fruit3.jpg" /></image>
16     </ImageView>
17     <ImageView>
18       <image><Image url="@images/fruit4.jpg" /></image>
19     </ImageView>
20     <ImageView>
21       <image><Image url="@images/fruit5.jpg" /></image>
22     </ImageView>
23   </children>
24 </TilePane>

```



>>> AppMain.java

```
1  package sec04.exam05_tilepane;
2
3  import javafx.application.Application;
4  import javafx.fxml.FXMLLoader;
5  import javafx.scene.Parent;
6  import javafx.scene.Scene;
7  import javafx.stage.Stage;
8
9  public class AppMain extends Application {
10     @Override
11     public void start(Stage primaryStage) throws Exception {
12         Parent root = FXMLLoader.load(getClass().getResource("root.fxml"));
13         Scene scene = new Scene(root);
14
15         primaryStage.setTitle("AppMain");
16         primaryStage.setScene(scene);
17         primaryStage.show();
18     }
19
20     public static void main(String[] args) {
21         launch(args);
22     }
23 }
```

GridPane 컨테이너

GridPane은 그리드로 컨트롤을 배치하되 셀의 크기가 고정적이지 않고 유동적인 컨테이너이다. 셀 병합이 가능하기 때문에 다양한 입력폼 화면을 만들 때 매우 유용하게 사용할 수 있다. 각 컨트롤은 자신이 배치될 행 인덱스와 컬럼 인덱스를 속성으로 가지며, 몇 개의 셀을 병합할 것인지도 지정할 수 있다.



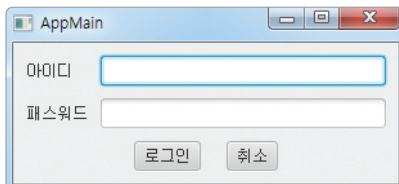
다음은 GridPane에 적용 가능한 속성들이다.

태그 및 속성	설명	적용
prefWidth	폭을 설정	GridPane
prefHeight	높이를 설정	GridPane
hgap	수평 컨트롤 간격을 설정	GridPane
vgap	수직 컨트롤 간격을 설정	GridPane
<children>	컨트롤을 포함	GridPane
GridPane.rowIndex	컨트롤이 위치하는 행 인덱스를 설정	컨트롤
GridPane.columnIndex	컨트롤이 위치하는 컬럼 인덱스를 설정	컨트롤
GridPane.rowSpan	행 병합 수를 설정	컨트롤
GridPane.columnSpan	컬럼 병합 수를 설정	컨트롤
GridPane.hgrow	수평 빈 공간 채우기를 설정	컨트롤
GridPane.vgrow	수직 빈 공간 채우기를 설정	컨트롤
GridPane.halignment	컨트롤의 수평 정렬을 설정	컨트롤
GridPane.valignment	컨트롤의 수직 정렬을 설정	컨트롤

다음은 로그인 화면을 GridPane으로 배치한 것이다.

>> root.fxml

```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <?import javafx.scene.layout.*?>
4  <?import javafx.geometry.*?>
5  <?import javafx.scene.control.*?>
6
7  <GridPane xmlns:fx="http://javafx.com/fxml"
8      prefWidth="300.0" hgap="10.0" vgap="10.0" >
9      <padding>
10         <Insets topRightBottomLeft="10.0"/>
11     </padding>
12     <children>
13         <Label text="아이디" GridPane.rowIndex="0" GridPane.columnIndex="0" />
14         <TextField GridPane.rowIndex="0" GridPane.columnIndex="1"
15             GridPane.hgrow="ALWAYS" />
16
17         <Label text="패스워드" GridPane.rowIndex="1" GridPane.columnIndex="0" />
18         <TextField GridPane.columnIndex="1" GridPane.rowIndex="1"
19             GridPane.hgrow="ALWAYS" />
20
21         <HBox GridPane.rowIndex="2" GridPane.columnIndex="0"
22             GridPane.columnSpan="2" GridPane.hgrow="ALWAYS"
23             alignment="CENTER" spacing="20.0" >
24             <children>
25                 <Button text="로그인" />
26                 <Button text="취소" />
27             </children>
28         </HBox>
29     </children>
30 </GridPane>
```

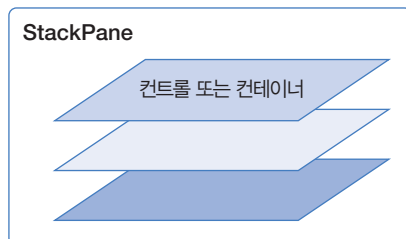


>>> AppMain.java

```
1  package sec04.exam06_gridpane;
2
3  import javafx.application.Application;
4  import javafx.fxml.FXMLLoader;
5  import javafx.scene.Parent;
6  import javafx.scene.Scene;
7  import javafx.stage.Stage;
8
9  public class AppMain extends Application {
10     @Override
11     public void start(Stage primaryStage) throws Exception {
12         Parent root = FXMLLoader.load(getClass().getResource("root.fxml"));
13         Scene scene = new Scene(root);
14
15         primaryStage.setTitle("AppMain");
16         primaryStage.setScene(scene);
17         primaryStage.show();
18     }
19
20     public static void main(String[] args) {
21         launch(args);
22     }
23 }
```

StackPane 컨테이너

StackPane은 컨트롤을 겹쳐 배치하는 컨테이너이다. 흔히 카드 레이아웃이라고 하는데, 그림과 같이 카드가 겹쳐 있는 것처럼 컨트롤도 겹쳐질 수 있다. 만약 위에 있는 컨트롤이 투명이라면 밑에 있는 컨트롤이 겹쳐 보이게 된다.



다음은 두 개의 ImageView를 StackPane에 겹치도록 배치한 예제이다. 하단 이미지는 설경이고 상단 이미지는 투명한 배경을 가지고 있는 듀크이다. 실행해보면 듀크와 설경이 하나의 이미지처럼 보이지만, 상하로 겹쳐 있다.

>> root.fxml

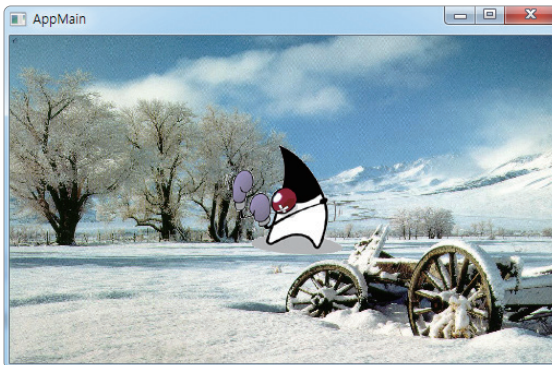
```

1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <?import javafx.scene.layout.*?>
4  <?import javafx.scene.image.*?>
5
6  <StackPane xmlns:fx="http://javafx.com/fxml">
7      <children>
8          <ImageView fitWidth="500" fitHeight="300">
9              <image>
10                 <Image url="@images/snow.jpg" />
11             </image>
12         </ImageView>
13         <ImageView preserveRatio="true">
14             <image>
15                 <Image url="@images/duke.gif" />
16             </image>
17         </ImageView>
18     </children>
19 </StackPane>

```

가로비와 세로비 상관없이
고정 길이로 설정

가로비와 세로비를 유지



>>> AppMain.java

```
1  package sec04.exam07_stackpane;
2
3  import javafx.application.Application;
4  import javafx.fxml.FXMLLoader;
5  import javafx.scene.Parent;
6  import javafx.scene.Scene;
7  import javafx.stage.Stage;
8
9  public class AppMain extends Application {
10     @Override
11     public void start(Stage primaryStage) throws Exception {
12         Parent root = FXMLLoader.load(getClass().getResource("root.fxml"));
13         Scene scene = new Scene(root);
14
15         primaryStage.setTitle("AppMain");
16         primaryStage.setScene(scene);
17         primaryStage.show();
18     }
19
20     public static void main(String[] args) {
21         launch(args);
22     }
23 }
```

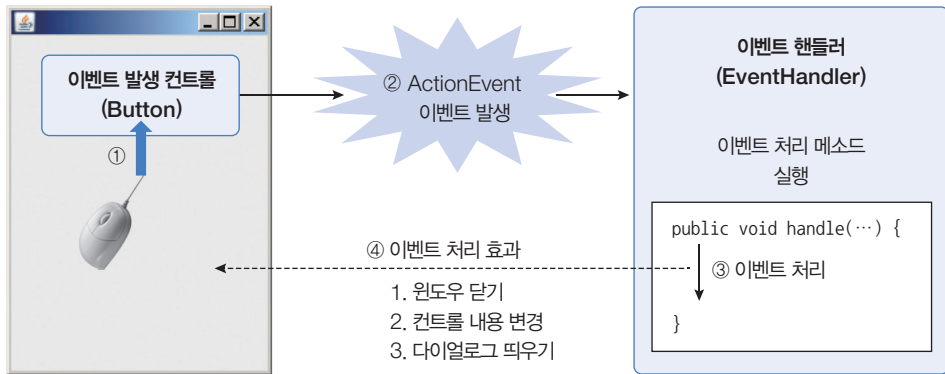
05 JavaFX 이벤트 처리

UI 애플리케이션은 사용자와 상호작용하면서 코드를 실행한다. 사용자가 UI 컨트롤을 사용하면 이벤트event가 발생하고, 프로그램은 이벤트를 처리하기 위해 코드를 실행한다.

이벤트 핸들러

JavaFX는 이벤트 발생 컨트롤과 이벤트 처리를 분리하기 위해 위임형Delegation 방식을 사용한다. 위임형 방식이란 컨트롤에서 이벤트가 발생하면 컨트롤이 직접 처리하지 않고 이벤트 핸들러에게 이벤트 처리를 위임하는 방식이다.

예를 들어 사용자가 Button을 클릭하면 ActionEvent가 발생하고, Button에 등록된 EventHandler가 ActionEvent를 처리한다.



EventHandler는 컨트롤에서 이벤트가 발생하면 자신의 handle() 메소드를 실행시킨다. handle() 메소드에는 윈도우 닫기, 컨트롤 내용 변경, 다이얼로그 띄우기 등의 코드를 작성할 수 있다.

EventHandler는 제네릭 타입이기 때문에 타입 파라미터는 발생한 이벤트의 타입이 된다. 예를 들어 ActionEvent를 처리하는 핸들러는 EventHandler<ActionEvent>가 되고, MouseEvent를 처리하는 핸들러는 EventHandler<MouseEvent>가 된다.

EventHandler가 컨트롤에서 발생한 이벤트를 처리하려면 먼저 컨트롤에 EventHandler를 등록해야 한다. 컨트롤은 발생하는 이벤트에 따라서 EventHandler를 등록하는 다양한 메소드가 있는데, 이 메소드들은 setOnXXX() 이름을 가지고 있다. XXX는 보통 이벤트 이름과 동일하다.

몇 가지 예를 들어보자. Button을 클릭할 때 발생하는 ActionEvent를 처리하는 EventHandler<ActionEvent>를 등록하려면 다음과 같이 setOnAction() 메소드를 사용한다.

```

Button button = new Button();
button.setOnAction(new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent event) { ... }
});
  
```

TableView의 행을 클릭할 때 발생하는 MouseEvent를 처리하는 EventHandler<MouseEvent>를 등록하려면 다음과 같이 setOnMouseClicked() 메소드를 사용한다.


```

TableView tableView = new TableView();
tableView.setOnMouseClicked(new EventHandler<MouseEvent>() {
    @Override
    public void handle(MouseEvent event) { ... }
});

```

윈도우^{Stage}의 우측 상단 닫기(×) 버튼을 클릭했을 때 발생하는 WindowEvent를 처리하는 EventHandler<WindowEvent>를 등록하려면 다음과 같이 setOnCloseRequest() 메소드를 사용한다.

```

stage.setOnCloseRequest(new EventHandler<WindowEvent>() {
    @Override
    public void handle(WindowEvent event) { ... }
});

```

EventHandler는 하나의 메소드를 가진 함수적 인터페이스이므로 람다식을 이용하면 보다 적은 코드로 EventHandler를 등록할 수 있다.

```

button.setOnAction( event->{ ... } );
tableView.setOnMouseClicked( event->{ ... } );
stage.setOnCloseRequest( event->{ ... } );

```

다음은 프로그램적 레이아웃을 작성하고 버튼의 ActionEvent를 처리한 것이다. 첫 번째 버튼은 직접 EventHandler 객체를 생성한 후 등록했고, 두 번째 버튼은 람다식을 이용해서 EventHandler를 등록했다.

>>> AppMain.java

```

1  package sec05.exam01_event_handler;
2
3  import javafx.application.Application;
4  import javafx.event.ActionEvent;
5  import javafx.event.EventHandler;

```

```

6  import javafx.geometry.Pos;
7  import javafx.scene.Scene;
8  import javafx.scene.control.Button;
9  import javafx.scene.layout.HBox;
10 import javafx.stage.Stage;
11
12 public class AppMain extends Application {
13     @Override
14     public void start(Stage primaryStage) throws Exception {
15         HBox root = new HBox();
16         root.setPrefSize(200, 50);
17         root.setAlignment(Pos.CENTER);
18         root.setSpacing(20);
19
20         Button btn1 = new Button("버튼1");
21         btn1.setOnAction(new EventHandler<ActionEvent>() {
22             @Override
23             public void handle(ActionEvent event) {
24                 System.out.println("버튼1 클릭");
25             }
26         });
27
28         Button btn2 = new Button("버튼2");
29         btn2.setOnAction(event -> System.out.println("버튼2 클릭"));
30
31         root.getChildren().addAll(btn1, btn2);
32         Scene scene = new Scene(root);
33
34         primaryStage.setTitle("AppMain");
35         primaryStage.setScene(scene);
36         primaryStage.setOnCloseRequest(event -> System.out.println("종료 클릭"));
37         primaryStage.show();
38     }
39
40     public static void main(String[] args) {
41         launch(args);
42     }
43 }

```

FXML 컨트롤러

프로그래믹 레이아웃은 레이아웃 코드와 이벤트 처리 코드를 모두 자바 코드로 작성해야 하므로 코드가 복잡해지고 유지보수도 힘들어지며, 디자이너와 협력해서 개발하는 것도 쉽지 않다. FXML 레이아웃은 FXML 파일당 별도의 컨트롤러(Controller)를 지정해서 이벤트를 처리할 수 있기 때문에 FXML 레이아웃과 이벤트 처리 코드를 완전히 분리할 수 있다.

1) fx:controller 속성과 컨트롤러 클래스

FXML 파일의 루트 태그에서 fx:controller 속성으로 컨트롤러를 지정하면 UI 컨트롤에서 발생하는 이벤트를 컨트롤러가 처리한다.

```
<루트컨테이너 xmlns:fx="http://javafx.com/fxml"
  fx:controller="package..RootController" >
  ...
</루트컨테이너>
```

컨트롤러는 다음과 같이 Initializable 인터페이스를 구현한 클래스로 작성하면 된다.

```
public class RootController implements Initializable {
    @Override
    public void initialize(URL location, ResourceBundle resources) { }
}
```

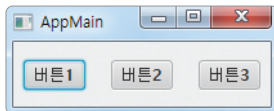
initialize() 메소드는 컨트롤러 객체가 생성되고 나서 호출되는데, 주로 UI 컨트롤의 초기화, 이벤트 핸들러 등록, 속성 감시 등의 코드가 작성된다.

2) fx:id 속성과 @FXML 컨트롤 주입

컨트롤러는 이벤트 핸들러를 등록하기 위해, 그리고 이벤트 처리 시 FXML 파일에 포함된 컨테이너 및 컨트롤의 참조가 필요하다. 이를 위해서 FXML 파일에 포함된 컨트롤들은 fx:id 속성을 가질 필요가 있다.

>> root.fxml

```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <?import javafx.scene.layout.*?>
4  <?import javafx.scene.control.*?>
5
6  <HBox xmlns:fx="http://javafx.com/fxml"
7      fx:controller="sec05.exam02_fxml_controller.RootController"
8      prefHeight="50.0" prefWidth="200.0"
9      alignment="CENTER" spacing="20.0" >
10     <children>
11         <Button fx:id="btn1" text="버튼1" />
12         <Button fx:id="btn2" text="버튼2" />
13         <Button fx:id="btn3" text="버튼3" />
14     </children>
15 </HBox>
```



fx:id 속성을 가진 컨트롤들은 컨트롤러의 @FXML 어노테이션이 적용된 필드에 자동 주입된다. 주의할 점은 fx:id 속성값과 필드명은 동일해야 한다.

```
public class RootController implements Initializable {
    @FXML private Button btn1;
    @FXML private Button btn2;
    @FXML private Button btn3;

    @Override
    public void initialize(URL location, ResourceBundle resources) { }
}
```

FXMLLoader가 FXML 파일을 로딩하면, 태그로 선언된 컨트롤들과 컨트롤러는 함께 객체로 생성된다. 그리고 나서 컨트롤러의 @FXML 어노테이션이 적용된 필드에 컨트롤 객체가 자동 주입된다. 주입이 완료되면 비로소 initialize() 메소드가 호출되기 때문에 initialize() 내부에서 필드를 안전하게 사용할 수 있다.

3) EventHandler 등록

컨트롤에서 발생하는 이벤트를 처리하려면 컨트롤러의 initialize() 메소드에서 EventHandler를 생성하고 등록해야 한다. 다음은 세 개의 Button에서 발생하는 ActionEvent를 처리하는 방법을 보여준다.

>>> RootController.java

```
1  package sec05.exam02_fxml_controller;
2
3  import java.net.URL;
4  import java.util.ResourceBundle;
5  import javafx.event.ActionEvent;
6  import javafx.event.EventHandler;
7  import javafx.fxml.FXML;
8  import javafx.fxml.Initializable;
9  import javafx.scene.control.Button;
10
11 public class RootController implements Initializable {
12     @FXML private Button btn1;
13     @FXML private Button btn2;
14     @FXML private Button btn3;
15
16     @Override
17     public void initialize(URL location, ResourceBundle resources) {
18         btn1.setOnAction(new EventHandler<ActionEvent>() {
19             @Override
20             public void handle(ActionEvent event) {
21                 handleBtn1Action(event);
22             }
23         });
24         btn2.setOnAction(event->handleBtn2Action(event));
25         btn3.setOnAction(event->handleBtn3Action(event));
```

직접 EventHandler
생성 후 등록

람다식 이용

```

26     }
27
28     public void handleBtn1Action(ActionEvent event) {
29         System.out.println("버튼1 클릭");
30     }
31     public void handleBtn2Action(ActionEvent event) {
32         System.out.println("버튼2 클릭");
33     }
34     public void handleBtn3Action(ActionEvent event) {
35         System.out.println("버튼3 클릭");
36     }
37 }

```

4) 이벤트 처리 메소드 매핑

컨트롤러에서 EventHandler를 생성하지 않고도 바로 이벤트 처리 메소드와 연결할 수 있는 방법이 있다. Button 컨트롤을 작성할 때 다음과 같이 onAction 속성값으로 "#메소드명"을 주면 내부적으로 EventHandler 객체가 생성되기 때문에 컨트롤러에서는 해당 메소드만 작성하면 된다.

FXML 파일

```
<Button fx:id="btn" text= "버튼" onAction= "#handleBtnAction"/>
```

Controller 클래스

```
public void handleBtnAction(ActionEvent event) { ... }
```

06 JavaFX 속성 감시와 바인딩

JavaFX는 컨트롤의 속성(property)을 감시하는 리스너를 설정할 수 있다. 예를 들어 Slider의 value 속성값을 감시하는 리스너를 설정해서 value 속성값이 변경되면 리스너가 다른 컨트롤러의 폰트나 이미지의 크기를 변경할 수 있다.

속성 감시

컨트롤의 모든 속성은 XXXProperty 객체로 생성된다. 그리고 Getter와 Setter 그리고 XXXProperty를 리턴하는 메소드로 구성된다. 예를 들어 TextField, TextArea 컨트롤의 text 속성은 StringProperty 객체로 생성되고 다음과 같은 메소드로 구성된다.

```
//StringProperty 타입의 text 속성 선언
private StringProperty text = new SimpleStringProperty();

//Getter와 Setter 선언
public void setText(String newValue) { text.set(newValue); }
public String getText() { return text.get(); }

//StringProperty 타입의 text 속성을 리턴하는 메소드
public StringProperty textProperty() { return text; }
```

StringProperty는 get()과 set() 메소드 이외에 리스너를 관리하는 textProperty() 메소드를 가지고 있다. text 속성을 감시하는 리스너는 textProperty()가 리턴하는 StringProperty에서 설정한다. javafx.beans.property 패키지에는 StringProperty 이외에도 다양한 XXXProperty 클래스가 존재한다.

다음은 TextField 컨트롤의 text 속성값을 감시하는 ChangeListener를 설정하는 코드이다.

```
StringProperty stringProperty = textField.textProperty();
stringProperty.addListener( new ChangeListener<String>() {
    @Override
    public void changed(ObservableValue<? extends String> observable,
                        String oldValue, String newValue) { ... }
} );
```

addListener() 메소드로 ChangeListener를 StringProperty 객체에 설정하면, text 속성이 변경되었을 때 ChangeListener의 changed() 메소드가 자동으로 실행된다. 속성의 이전 값은 oldValue에, 새로운 값은 newValue로 전달된다.

ChangeListener는 제네릭 타입인데, 타입 파라미터는 속성의 타입이 된다. 예를 들어 textProperty()가 리턴하는 StringProperty는 Property<String>을 구현하고 있기 때문에 타입

파라미터는 String이 된다. 따라서 oldValue와 newValue의 타입은 String이다.

다른 예를 보자. Slider의 value 속성에 리스너를 설정하려면 다음과 같이 작성하면 된다.

```
Slider slider = new Slider();
DoubleProperty doubleProperty = slider.valueProperty();
doubleProperty.addListener( new ChangeListener<Number>() {
    @Override
    public void changed(ObservableValue<? extends Number> observable,
        Number oldValue, Number newValue) { ... }
} );
```

valueProperty()가 리턴하는 DoubleProperty가 Property<Number>를 구현하고 있기 때문에 ChangeListener의 타입 파라미터는 Number가 된다. 따라서 oldValue와 newValue의 타입도 Number가 된다.

다음 예제는 Slider의 value 속성을 감시해서 value 속성값이 변경되면 Label의 폰트 크기를 변경하도록 리스너를 설정했다.

>>> root.fxml

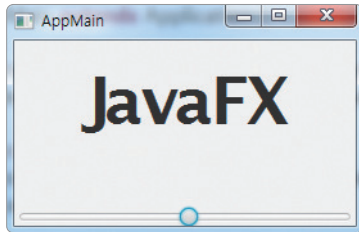
```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <?import javafx.scene.text.*?>
4  <?import javafx.scene.control.*?>
5  <?import java.lang.*?>
6  <?import javafx.scene.layout.*?>
7  <?import javafx.scene.layout.AnchorPane?>
8
9  <BorderPane xmlns:fx="http://javafx.com/fxml1"
10      fx:controller="sec06.exam01_property_listener.RootController"
11      prefHeight="250.0" prefWidth="350.0" >
12      <center>
13          <Label fx:id="label" text="JavaFX" >
14              <font>
15                  <Font size="0" />
16              </font>
```

Label의 기본 폰트 크기는 0


```

17     </Label>
18 </center>
19 <bottom>
20     <Slider fx:id="slider" />
21 </bottom>
22 </BorderPane>

```



>>> RootController.java

```

1  package sec06.exam01_property_listener;
2
3  import java.net.URL;
4  import java.util.ResourceBundle;
5  import javafx.beans.value.ChangeListener;
6  import javafx.beans.value.ObservableValue;
7  import javafx.fxml.FXML;
8  import javafx.fxml.Initializable;
9  import javafx.scene.control.Label;
10 import javafx.scene.control.Slider;
11 import javafx.scene.text.Font;
12
13 public class RootController implements Initializable {
14     @FXML private Slider slider;
15     @FXML private Label label;
16
17     @Override
18     public void initialize(URL location, ResourceBundle resources) {

```

```

19     slider.valueProperty().addListener(new ChangeListener<Number>() {
20         @Override
21         public void changed(ObservableValue<? extends Number> observable,
22                             Number oldValue, Number newValue) {
23             label.setFont( new Font( newValue.doubleValue() ) );
24         } Label의 폰트 변경      Label의 폰트 변경
25     });
26 }
27 }

```

value 속성 감시 리스너 등록

>>> AppMain.java

```

1  package sec06.exam01_property_listener;
2
3  import javafx.application.Application;
4  import javafx.fxml.FXMLLoader;
5  import javafx.scene.Parent;
6  import javafx.scene.Scene;
7  import javafx.stage.Stage;
8
9  public class AppMain extends Application {
10     @Override
11     public void start(Stage primaryStage) throws Exception {
12         Parent root = (Parent)FXMLLoader.load(getClass().getResource
13             ("root.fxml"));
14         Scene scene = new Scene(root);
15         primaryStage.setTitle("AppMain");
16         primaryStage.setScene(scene);
17         primaryStage.show();
18     }
19
20     public static void main(String[] args) {
21         launch(args);
22     }
23 }

```

속성 바인딩

컨트롤의 속성은 다른 컨트롤의 속성과 바인딩될 수 있다. 바인딩된 속성들은 하나가 변경되면 자동적으로 다른 하나도 변경된다. 예를 들어 두 개의 `TextArea` 컨트롤의 `text` 속성들을 바인딩하여 한 쪽 `text` 속성이 변경되면 다른 쪽 `text` 속성도 자동 변경된다.

속성을 바인딩하기 위해서는 컨트롤의 `xxxProperty()` 메소드가 리턴하는 `XXXProperty` 객체의 `bind()` 메소드를 이용하면 된다. 예를 들어 `textArea1`에서 입력된 내용이 `textArea2`에 자동으로 입력되도록 하려면 다음과 같이 작성하면 된다.

```
TextArea textArea1 = new TextArea();
TextArea textArea2 = new TextArea();
textArea2.textProperty().bind(textArea1.textProperty());
```

`bind()` 메소드는 단방향인데, `textArea1`에서 입력된 내용만 `textArea2`로 자동 입력되고 반대로 `textArea2`에서 입력된 내용은 `textArea1`로 자동 입력되지 않는다. 아예 `textArea2`는 입력조차 할 수 없다. 만약 양방향으로 바인딩하고 싶다면 `bind()` 메소드 대신 `bindBidirectional()` 메소드를 이용하거나 `Bindings.bindBidirectional()` 메소드를 이용하면 된다.

```
//양방향 바인딩 방법1
textArea2.textProperty().bindBidirectional(textArea1.textProperty());
//양방향 바인딩 방법2
Bindings.bindBidirectional(textArea1.textProperty(), textArea2.textProperty());
```

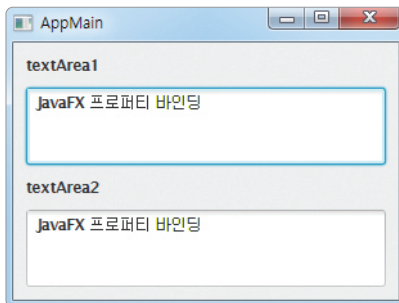
바인딩된 속성을 언바인드하려면 다음 메소드를 이용한다.

```
//단방향 해제
textArea2.textProperty().unbind();
//양방향 해제 방법1
textArea2.textProperty().unbindBidirectional(textArea1.textProperty());
//양방향 해제 방법2
Bindings.unbindBidirectional(textArea1.textProperty(), textArea2.textProperty());
```

다음 예제는 text 속성으로 두 개의 TextArea 컨트롤을 양방향으로 바인딩하였다.

>> root.fxml

```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <?import javafx.scene.layout.*?>
4  <?import javafx.geometry.*?>
5  <?import javafx.scene.control.*?>
6
7
8  <VBox xmlns:fx="http://javafx.com/fxml"
9      fx:controller="sec06.exam02_property_bind.RootController"
10     prefHeight="200.0" prefWidth="300.0" spacing="10.0" >
11     <padding>
12         <Insets bottom="10.0" left="10.0" right="10.0" top="10.0" />
13     </padding>
14     <children>
15         <Label text="textArea1" />
16         <TextArea fx:id="textArea1"/>
17         <Label text="textArea2" />
18         <TextArea fx:id="textArea2"/>
19     </children>
20 </VBox>
```



>>> RootController.java

```
1  package sec06.exam02_property_bind;
2
3  import java.net.URL;
4  import java.util.ResourceBundle;
5  import javafx.beans.binding.Bindings;
6  import javafx.fxml.FXML;
7  import javafx.fxml.Initializable;
8  import javafx.scene.control.TextArea;
9
10 public class RootController implements Initializable {
11     @FXML private TextArea textArea1;
12     @FXML private TextArea textArea2;
13
14     @Override
15     public void initialize(URL location, ResourceBundle resources) {
16         Bindings.bindBidirectional(textArea1.textProperty(),
17                                   textArea2.textProperty());
18     }
19 }
```

>>> AppMain.java

```
1  package sec06.exam02_property_bind;
2
3  import javafx.application.Application;
4  import javafx.fxml.FXMLLoader;
5  import javafx.scene.Parent;
6  import javafx.scene.Scene;
7  import javafx.stage.Stage;
8
9  public class AppMain extends Application {
10     @Override
11     public void start(Stage primaryStage) throws Exception {
12         Parent root = (Parent)FXMLLoader.load(getClass().getResource("root.fxml"));
13         Scene scene = new Scene(root);
14
15         primaryStage.setTitle("AppMain");
16     }
17 }
```

```

16     primaryStage.setScene(scene);
17     primaryStage.show();
18 }
19
20 public static void main(String[] args) {
21     launch(args);
22 }
23 }

```

Bindings 클래스

두 속성이 항상 동일한 값과 타입을 가질 수는 없다. 한쪽 속성값이 다른 쪽 속성값과 바인딩하기 위해서는 연산 작업이 필요할 수도 있다.

예를 들어 윈도우의 크기에 상관없이 항상 화면 정중앙에 원을 그린다고 가정해 보자. 루트 컨테이너 폭의 1/2이 원의 X좌표가 되고, 루트 컨테이너 높이의 1/2이 원의 Y좌표가 될 것이다. 따라서 루트 컨테이너의 폭과 높이를 원의 중심과 바인딩하기 위해서는 1/2이라는 연산이 필요하다.

이때 사용할 수 있는 것이 Bindings 클래스가 제공하는 정적 메소드들이다. Bindings의 정적 메소드는 속성을 연산하거나, 다른 타입으로 변환한 후 바인딩하는 기능을 제공한다. 다음은 Bindings 클래스가 제공하는 정적 메소드들을 설명한 표이다.

메소드	설명
add, subtract, multiply, divide	속성값을 덧셈, 뺄셈, 곱셈, 나눗셈 연산을 수행하고 바인딩함
max, min	속성값과 어떤 수를 비교해서 최대, 최소값을 얻고 바인딩함
greaterThan, greaterThanOrEqualTo	속성값이 어떤 값보다 크거나, 같거나 큰지를 조사해서 true/false로 변환하여 바인딩함
lessThan, lessThanOrEqualTo	속성값이 어떤 값보다 적거나, 같거나 적은지를 조사해서 true/false로 변환하여 바인딩함
equal, notEquals	속성값이 어떤 값과 같은지, 다른지를 조사해서 true/false로 변환하여 바인딩함
equalsIgnoreCase, notEqualsIgnoreCase	대소문자와 상관없이 속성값이 어떤 문자열과 같은지, 다른지를 조사해서 true/false로 변환하여 바인딩함
isEmpty, isEmpty	속성값이 비어있는지, 아닌지를 조사해서 true/false로 변환하여 바인딩함
isNull, isNotNull	속성값이 null 또는 not null인지를 조사해서 true/false로 변환하여 바인딩함

length	속성값이 문자열일 경우 문자 수를 얻어 바인딩함
size	속성 타입이 배열, List, Map, Set일 경우 요소 수를 얻어 바인딩함
and, or	속성값이 boolean일 경우, 논리곱, 논리합을 얻어 바인딩함
not	속성값이 boolean일 경우, 반대값으로 바인딩함
convert	속성값을 문자열로 변환해서 바인딩함
valueAt	속성이 List, Map일 경우 해당 인덱스 또는 키의 값을 얻어 바인딩함

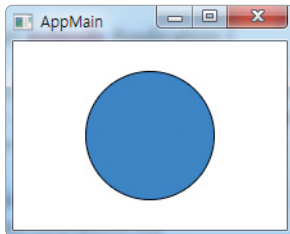
다음은 윈도우 창의 크기가 변경되더라도 항상 화면 정중앙에 원을 그리는 예제이다. 루트 컨테이너의 폭과 높이를 원의 중심과 바인딩하기 위해 1/2 연산을 해야 하므로 `Bindings.divide()` 메소드를 이용하였다.

>>> root.fxml

```

1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <?import javafx.scene.layout.*?>
4  <?import javafx.scene.shape.*?>
5
6  <AnchorPane xmlns:fx="http://javafx.com/fxml"
7              fx:id="root"
8              fx:controller="sec06.exam03_bindings.RootController"
9              prefHeight="200.0" prefWidth="300.0" >
10     <children>
11         <Circle fx:id="circle" fill="blue" radius="50.0" stroke="BLACK" />
12     </children>
13 </AnchorPane>

```



>>> RootController.java

```
1  package sec06.exam03_bindings;
2
3  import java.net.URL;
4  import java.util.ResourceBundle;
5  import javafx.beans.binding.Bindings;
6  import javafx.fxml.FXML;
7  import javafx.fxml.Initializable;
8  import javafx.scene.layout.AnchorPane;
9  import javafx.scene.shape.Circle;
10
11 public class RootController implements Initializable {
12     @FXML private AnchorPane root;
13     @FXML private Circle circle;
14
15     @Override
16     public void initialize(URL location, ResourceBundle resources) {
17         circle.centerXProperty().bind( Bindings.divide(root.widthProperty(), 2) );
18         circle.centerYProperty().bind( Bindings.divide(root.heightProperty(), 2) );
19     }
20 }
```

원도우 창의 width, height 속성에 1/2 연산후 Circle의 centerX, centerY 속성과 바인딩

>>> AppMain.java

```
1  package sec06.exam03_bindings;
2
3  import javafx.application.Application;
4  import javafx.fxml.FXMLLoader;
5  import javafx.scene.Parent;
6  import javafx.scene.Scene;
7  import javafx.stage.Stage;
8
9  public class AppMain extends Application {
10     @Override
11     public void start(Stage primaryStage) throws Exception {
12         Parent root = (Parent)FXMLLoader.load(getClass().getResource("root.fxml"));
13         Scene scene = new Scene(root);
14     }
```



```

15     primaryStage.setTitle("AppMain");
16     primaryStage.setScene(scene);
17     primaryStage.show();
18 }
19
20 public static void main(String[] args) {
21     launch(args);
22 }
23 }

```


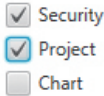
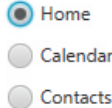

07 JavaFX 컨트롤

JavaFX는 다양한 UI 컨트롤을 제공하고 있다. 이번 절에서는 사용 빈도가 높은 버튼 컨트롤, 입력 컨트롤, 뷰 컨트롤, 미디어 컨트롤, 차트 컨트롤에 대해 살펴보자.

버튼 컨트롤

버튼 컨트롤은 마우스로 클릭할 수 있는 컨트롤로 `ButtonBase`를 상속하는 하위 컨트롤을 말한다.

`Button`, `CheckBox`, `RadioButton`, `ToggleButton`, `Hyperlink` 등이 있다.

Button	CheckBox	RadioButton	ToggleButton
			

기본 `Button`은 단순한 글자로 구성되지만 `setGraphic(ImageView)` 메소드로 아이콘을 넣을 수도 있다. 다음은 아이콘 버튼을 FXML로 작성하는 방법을 보여준다.

```

<Button text= "아이콘버튼">
  <graphic>
    <ImageView>

```

```

        <Image url= "@images/history_view.gif"/>
    </ImageView>
</graphic>
</Button>

```

CheckBox, RadioButton, ToggleButton 컨트롤은 선택과 미선택 두 가지 상태를 가질 수 있다. selected 속성의 값이 true이면 선택이고, false이면 미선택이다. 다음은 CheckBox 컨트롤을 FXML로 선언한 것이다. text 속성은 사용자에게 보여주는 문자열이고, userData 속성은 프로그램에서 처리하는 데이터이다.

```

<CheckBox text= "라벨1" userData= "값1"/>
<CheckBox text= "라벨2" userData= "값2" selected="true"/>

```

RadioButton, ToggleButton에는 toggleGroup 속성이 있는데, 이 속성의 값은 \$groupName 이다. \$groupName은 <ToggleGroup fx:id="\$groupName"/>의 fx:id를 참조한다. 같은 groupName을 참조하는 버튼들은 하나의 그룹으로 묶이며, 같은 그룹 내에서는 하나의 버튼만 선택된다.

```

<VBox>
    <fx:define>
        <ToggleGroup fx:id="group" />
    </fx:define>
    <children>
        <RadioButton ... toggleGroup="$group" />
        <RadioButton ... toggleGroup="$group" />
        <RadioButton ... toggleGroup="$group" />
    </children>
</VBox>

```

CheckBox, RadioButton, ToggleButton 컨트롤은 사용자가 클릭하면(ActionEvent)가 발생하기 때문에 EventHandler로 처리가 가능하고, onAction 속성을 작성해서 컨트롤러의 이벤트 처리 메소드로 연결할 수도 있다.

```
<CheckBox ... onAction= "#handleChkAction"/>
```

만약 같은 그룹 내에서 RadioButton 또는 ToggleButton의 선택 변경을 감시하고 싶다면 ToggleGroup의 selectedToggle 속성에 다음과 같이 감시자를 등록하면 된다.

```
groupName.selectedToggleProperty().addListener(new ChangeListener<Toggle>() {  
    @Override  
    public void changed(ObservableValue<? extends Toggle> observable,  
                        Toggle oldValue, Toggle newValue) { ... }  
});
```

선택이 변경되면 changed() 메소드가 실행되고 세 번째 매개값인 newValue에 마지막으로 선택된 버튼이 대입된다. 다음 예제는 CheckBox와 RadioButton의 이벤트 처리를 어떻게 하는지 보여준다.

>>> root.fxml

```
1  <?xml version="1.0" encoding="UTF-8"?>  
2  
3  <?import javafx.scene.layout.*?>  
4  <?import javafx.geometry.*?>  
5  <?import javafx.scene.control.*?>  
6  <?import javafx.scene.image.*?>  
7  
8  <BorderPane xmlns:fx="http://javafx.com/fxml"  
9      fx:controller="sec07.exam01_button.RootController" prefHeight="150.0"  
10     prefWidth="420.0">  
11     <padding>  
12         <Insets bottom="10.0" left="10.0" right="10.0" top="10.0" />  
13     </padding>  
14  
15     <center>  
16         <HBox alignment="CENTER" prefHeight="100.0" prefWidth="200.0"  
17             spacing="10">  
18             <children>  
19                 <VBox prefHeight="200.0" prefWidth="100.0" spacing="20.0"
```

```

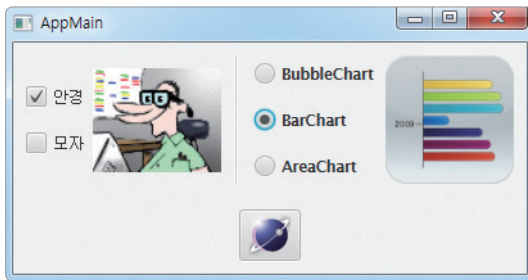
20         alignment="CENTER_LEFT">
21     <children>
22         <CheckBox fx:id="chk1" text="안경" onAction="#handleChkAction" />
23         <CheckBox fx:id="chk2" text="모자" onAction="#handleChkAction" />
24     </children>
25 </VBox>
26
27 <ImageView fx:id="checkImageView" fitWidth="100.0"
28     preserveRatio="true">
29     <image>
30         <Image url="@images/geek.gif" />
31     </image>
32 </ImageView>
33
34 <Separator orientation="VERTICAL" prefHeight="200.0" />
35
36 <VBox prefHeight="100" prefWidth="150" spacing="20.0"
37     alignment="CENTER_LEFT">
38     <fx:define> <ToggleGroup fx:id="group" /> </fx:define>
39     <children>
40         <RadioButton fx:id="rad1" text="BubbleChart"
41             userData="BubbleChart" toggleGroup="$group" />
42         <RadioButton fx:id="rad2" text="BarChart"
43             userData="BarChart" toggleGroup="$group" selected="true" />
44         <RadioButton fx:id="rad3" text="AreaChart"
45             userData="AreaChart" toggleGroup="$group" />
46     </children>
47 </VBox>
48
49 <ImageView fx:id="radioImageView" fitWidth="100.0"
50     preserveRatio="true">
51     <image>
52         <Image url="@images/BarChart.png" />
53     </image>
54 </ImageView>
55 </children>
56 </HBox>
57 </center>
58
59 <bottom>

```

```

60     <Button fx:id="btnExit" BorderPane.alignment="CENTER"
61         onAction="#handleBtnExitAction">
62         <graphic>
63             <ImageView <Image url="@images/exit.png" /> </ImageView>
64         </graphic>
65         <BorderPane.margin> <Insets top="20.0" /> </BorderPane.margin>
66     </Button>
67 </bottom>
68 </BorderPane>

```



>>> RootController.java

```

1  package sec07.exam01_button;
2
3  import java.net.URL;
4  import java.util.ResourceBundle;
5  import javafx.application.Platform;
6  import javafx.beans.value.ChangeListener;
7  import javafx.beans.value.ObservableValue;
8  import javafx.event.ActionEvent;
9  import javafx.fxml.FXML;
10 import javafx.fxml.Initializable;
11 import javafx.scene.control.Button;
12 import javafx.scene.control.CheckBox;
13 import javafx.scene.control.Toggle;
14 import javafx.scene.control.ToggleGroup;
15 import javafx.scene.image.Image;

```

```

16  import javafx.scene.image.ImageView;
17
18  public class RootController implements Initializable {
19      @FXML private CheckBox chk1;
20      @FXML private CheckBox chk2;
21      @FXML private ImageView checkImageView;
22      @FXML private ToggleGroup group;
23      @FXML private ImageView radioImageView;
24      @FXML private Button btnExit;
25
26      @Override
27      public void initialize(URL location, ResourceBundle resources) {
28          group.selectedToggleProperty().addListener(new
29              ChangeListener<Toggle>() {
30                  @Override
31                  public void changed(ObservableValue<? extends Toggle> observable,
32                      Toggle oldValue, Toggle newValue) {
33                      Image image = new Image(getClass().getResource(
34                          "images/" + newValue.getUserData().toString() + ".png").
35                          toString());
36                      radioImageView.setImage(image);
37                  }
38              });
39
40      public void handleChkAction(ActionEvent e) {
41          if(chk1.isSelected() && chk2.isSelected()) {
42              checkImageView.setImage(new Image(getClass().getResource(
43                  "images/geek-glasses-hair.gif").toString()));
44          } else if(chk1.isSelected()) {
45              checkImageView.setImage(new Image(getClass().getResource(
46                  "images/geek-glasses.gif").toString()));
47          } else if(chk2.isSelected()) {
48              checkImageView.setImage(new Image(getClass().getResource(
49                  "images/geek-hair.gif").toString()));
50          } else {
51              checkImageView.setImage(new Image(getClass().getResource(
52                  "images/geek.gif").toString()));
53          }
54      }
55  }

```

```

54
55     public void handleBtnExitAction(ActionEvent e) {
56         Platform.exit();
57     }
58 }

```

>>> AppMain.java

```

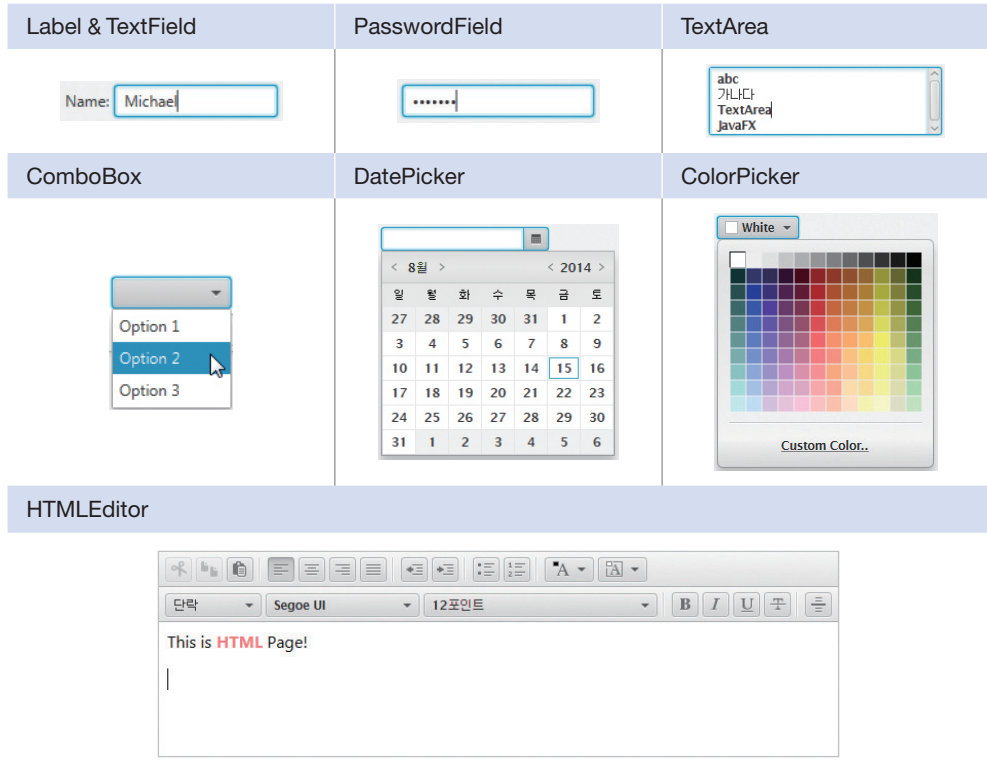
1  package sec07.exam01_button;
2
3  import javafx.application.Application;
4  import javafx.fxml.FXMLLoader;
5  import javafx.scene.Parent;
6  import javafx.scene.Scene;
7  import javafx.stage.Stage;
8
9  public class AppMain extends Application {
10     @Override
11     public void start(Stage primaryStage) throws Exception {
12         Parent root = (Parent)FXMLLoader.load(getClass().getResource("root.
            fxml"));
13         Scene scene = new Scene(root);
14
15         primaryStage.setTitle("AppMain");
16         primaryStage.setScene(scene);
17         primaryStage.show();
18     }
19
20     public static void main(String[] args) {
21         launch(args);
22     }
23 }

```

입력 컨트롤

입력 컨트롤에는 한 줄 입력을 위한 TextField, 다중 행 입력을 위한 TextArea, 패스워드 입력을 위한 PasswordField, 제한된 항목에서 선택하는 ComboBox가 있다. 또한 날짜를 선택할 수 있

는 DatePicker, 색상을 선택할 수 있는 ColorPicker, HTML을 입력하기 위한 HTMLEditor도 입력 컨트롤이라고 볼 수 있다. Label은 입력 컨트롤은 아니지만 입력 컨트롤의 제목을 표시할 때 사용된다.



다음은 입력 컨트롤을 FXML로 선언하는 방법을 보여준다.

```
<!-- Label 컨트롤 -->
<Label prefWidth="폭" prefHeight="높이" text="제목" />

<!-- TextField 컨트롤 -->
<TextField prefWidth="폭" prefHeight="높이" promptText="힌트문자열"/>

<!-- PasswordField 컨트롤 -->
<PasswordField prefWidth="폭" prefHeight="높이" promptText="힌트문자열"/>
```



```

<!-- ComboBox 컨트롤 -->
<ComboBox prefWidth="폭" prefHeight="높이" promptText="힌트문자열" >
    <items>
        <FXCollections fx:factory="observableArrayList">
            <String fx:value="공개"/>
            <String fx:value="비공개"/>
        </FXCollections>
    </items>
</ComboBox>

<!-- DatePicker 컨트롤 -->
<DatePicker prefWidth="폭" prefHeight="높이" promptText="힌트문자열" />

<!-- TextArea 컨트롤 -->
<TextArea prefWidth="폭" prefHeight="높이" promptText="힌트문자열"/>

```

prefWidth와 prefHeight는 각각 폭과 높이를 설정하고, promptText는 힌트 문자열로 컨트롤이 포커스를 얻게 되면 사라진다.

다음 예제는 입력 컨트롤로 구성된 폼을 제공한다. 제목은 TextField, 비밀번호는 PasswordField, 공개는 ComboBox, 게시종료는 DatePicker, 내용은 TextArea로 구성했다. [등록] 버튼을 클릭하면 모든 입력된 내용이 콘솔에 출력된다.

```
>>> root.fxml
```

```

1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <?import javafx.scene.text.*?>
4  <?import javafx.scene.shape.*?>
5  <?import javafx.scene.web.*?>
6  <?import javafx.geometry.*?>
7  <?import javafx.scene.image.*?>
8  <?import java.lang.*?>
9  <?import javafx.scene.control.*?>
10 <?import javafx.scene.layout.*?>
11 <?import javafx.collections.*?>
12

```

```

13 <AnchorPane xmlns:fx="http://javafx.com/fxml"
14     fx:controller="sec07.exam02_input.RootController"
15     prefHeight="380" prefWidth="485" >
16     <children>
17         <Label layoutX="22.0" layoutY="36.0" text="제목" />
18         <TextField fx:id="txtTitle" layoutX="84.0" layoutY="32.0"
19             prefHeight="23.0" prefWidth="375.0" />
20         <Label layoutX="22.0" layoutY="69.0" text="비밀번호" />
21         <PasswordField fx:id="txtPassword" layoutX="86.0" layoutY="65.0"
22             prefHeight="23.0" prefWidth="132.0" />
23         <Label layoutX="22.0" layoutY="104.0" text="공개" />
24         <ComboBox fx:id="comboPublic" layoutX="86.0" layoutY="100.0"
25             prefHeight="23.0" prefWidth="132.0" promptText="선택하세요" >
26             <items>
27                 <FXCollections fx:factory="observableArrayList">
28                     <String fx:value="공개"/>
29                     <String fx:value="비공개"/>
30                 </FXCollections>
31             </items>
32         </ComboBox>
33         <Label layoutX="240.0" layoutY="104.0" text="게시종료" />
34         <DatePicker fx:id="dateExit" layoutX="296.0" layoutY="100.0"
35             promptText="날짜를 선택하세요"/>
36         <Label layoutX="22.0" layoutY="135.0" text="내용" />
37         <TextArea fx:id="txtContent" layoutX="22.0" layoutY="154.0"
38             prefHeight="132.0" prefWidth="440.0"/>
39         <Separator layoutX="13.0" layoutY="320"
40             prefHeight="0.0" prefWidth="457.0" />
41         <Button fx:id="btnReg" layoutX="189.0" layoutY="340" text="등록"
42             onAction="#handleBtnRegAction"/>
43         <Button fx:id="btnCancel" layoutX="252.0" layoutY="340" text="취소"
44             onAction="#handleBtnCancelAction"/>
45     </children>
46 </AnchorPane>

```

AppMain

제목:

비밀번호:

공개: 게시종료:

내용:

[Java UI 변화]

AWT> Swing> JavaFX

>>> RootController.java

```

1  package sec07.exam02_input;
2
3  import java.net.URL;
4  import java.time.LocalDate;
5  import java.util.ResourceBundle;
6  import javafx.application.Platform;
7  import javafx.event.ActionEvent;
8  import javafx.fxml.FXML;
9  import javafx.fxml.Initializable;
10 import javafx.scene.control.ComboBox;
11 import javafx.scene.control.DatePicker;
12 import javafx.scene.control.PasswordField;
13 import javafx.scene.control.TextArea;
14 import javafx.scene.control.TextField;
15
16 public class RootController implements Initializable {
17     @FXML private TextField txtTitle;
18     @FXML private PasswordField txtPassword;
19     @FXML private ComboBox<String> comboPublic;
20     @FXML private DatePicker dateExit;

```

```

21     @FXML private TextArea txtContent;
22
23     @Override
24     public void initialize(URL location, ResourceBundle resources) {
25     }
26
27     public void handleBtnRegAction(ActionEvent e) {
28         String title = txtTitle.getText();
29         System.out.println("title: " + title);
30
31         String password = txtPassword.getText();
32         System.out.println("password: " + password);
33
34         String strPublic = comboPublic.getValue();
35         System.out.println("public: " + strPublic);
36
37         LocalDate localDate = dateExit.getValue();
38         if(localDate != null) {
39             System.out.println("dateExit: " + localDate.toString());
40         }
41
42         String content = txtContent.getText();
43         System.out.println("content: " + content);
44     }
45
46     public void handleBtnCancelAction(ActionEvent e) {
47         Platform.exit();
48     }
49 }

```

>>> AppMain.java

```

1     package sec07.exam02_input;
2
3     import javafx.application.Application;
4     import javafx.fxml.FXMLLoader;
5     import javafx.scene.Parent;
6     import javafx.scene.Scene;

```

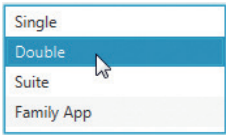
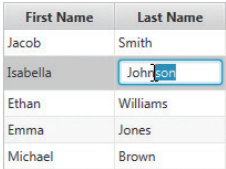

```

7  import javafx.stage.Stage;
8
9  public class AppMain extends Application {
10     @Override
11     public void start(Stage primaryStage) throws Exception {
12         Parent root = (Parent)FXMLLoader.load(getClass().getResource("root.
            fxml"));
13         Scene scene = new Scene(root);
14
15         primaryStage.setTitle("AppMain");
16         primaryStage.setScene(scene);
17         primaryStage.show();
18     }
19
20     public static void main(String[] args) {
21         launch(args);
22     }
23 }

```

뷰 컨트롤

뷰 컨트롤은 텍스트 또는 이미지 등을 보여주는데 목록 형태로 보여주는 ListView, 테이블 형태로 보여주는 TableView, 이미지를 보여주는 ImageView가 있다.

ListView	TableView	ImageView
		

1) ImageView 컨트롤

ImageView는 이미지를 보여주는 컨트롤이다. FXML로 선언하는 방법은 다음과 같다.

```
<ImageView fitWidth= "폭" fitHeight= "높이" preserveRatio="true"/>
```

fitWidth와 fitHeight는 ImageView의 폭과 높이를 지정한다. preserveRatio 속성은 이미지의 종횡비를 유지할 것인지를 지정한다. false를 주면 종횡비와 상관없이 fitWidth와 fitHeight에 맞게 ImageView 크기가 고정되고, true를 주면 이미지의 종횡비를 유지하기 위해 ImageView 크기가 조정된다.

ImageView에 보여줄 이미지는 두 가지 방법으로 설정할 수 있는데, 첫 번째 방법은 ImageView의 생성자 매개값으로 Image 객체를 설정하는 것이다.

```
<ImageView preserveRatio="true">
  <Image url= "@images/flower.png"/>
</ImageView>
```

두 번째 방법은 ImageView의 setImage() 메소드로 설정하는데, 역시 매개값은 Image 객체이다.

```
<ImageView fitWidth="200" fitHeight="150" preserveRatio="true">
  <image>
    <Image url= "@images/flower.png"/>
  </image>
</ImageView>
```

Image는 url 속성(생성자 매개변수)을 가지고 있는데, FXML 파일 위치에서 상대 경로로 "@이미지 경로"를 값으로 주면 된다.

2) ListView 컨트롤

ListView는 항목들을 목록으로 보여주는 컨트롤이다. FXML로 선언하는 방법은 다음과 같다.

```
<ListView prefWidth= "폭" prefHeight= "높이"/>
```

ListView에 항목을 추가하려면 `setItems(ObservableList<T> value)` 메소드를 사용한다. `ObservableList` 구현 객체는 `FXCollections.observableArrayList(E... items)` 정적 메소드로 생성할 수 있다.

```
listView.setItems(FXCollections.observableArrayList("Swing", "JavaFX"));
```

외부 데이터로 항목을 추가하는 경우가 많기 때문에 컨트롤러에서 자바 코드로 추가하는 것이 일반적이지만, 고정 항목일 경우에는 FXML 파일에서 다음과 같이 직접 선언해도 좋다.

```
<ListView fx:id="listView" prefHeight="100" prefWidth="100">
  <items>
    <FXCollections fx:factory="observableArrayList">
      <String fx:value="Swing"/>
      <String fx:value="JavaFX"/>
    </FXCollections>
  </items>
</ListView>
```

ListView에서 선택된 인덱스와 항목을 얻으려면 속성 감시를 이용할 수 있다. `getSelectionModel()` 메소드로 `MultipleSelectionModel`을 얻고 나서 `selectedIndexProperty` 또는 `selectedItemProperty`에 리스너를 설정하면 된다. `selectedIndexProperty`는 선택된 인덱스이고, `selectedItemProperty`는 선택된 항목이다. 다음은 `selectedItemProperty`에 리스너를 설정하는 코드이다.

```
listView.getSelectionModel().selectedItemProperty().addListener(
    new ChangeListener<String>() {
        @Override
        public void changed(ObservableValue<? extends String> observable,
                           String oldValue, String newValue) { ... }
    }
);
```

3) TableView 컨트롤

TableView는 컬럼으로 구성된 행의 데이터를 보여주는 컨트롤이다. TableView를 FXML로 선언하는 방법은 다음과 같다. <columns> 태그 안에 만들고자 하는 컬럼의 개수만큼 <TableColumn> 태그를 선언하면 된다.

```
<TableView prefHeight="100" prefWidth="300">
    <columns>
        <TableColumn prefWidth="150" text= "스마트폰"/>
        <TableColumn prefWidth="150" text= "이미지"/>
    </columns>
</TableView>
```

TableView에 행^{row}을 추가하려면 행의 데이터를 가지고 있는 모델^{model} 객체가 필요하다. 위 코드를 보면 스마트폰과 이미지 컬럼이 있는데, 이 두 값을 속성으로 갖는 모델 객체를 생성해서 행의 데이터로 제공해야 한다. 다음은 Phone 모델 클래스를 생성하는 방법을 보여준다.

>>> Phone.java

```
1  package sec07.exam03_view;
2
3  import javafx.beans.property.SimpleStringProperty;
4
5  public class Phone {
6      private SimpleStringProperty smartPhone;
7      private SimpleStringProperty image;
8
9      public Phone() {
10         this.smartPhone = new SimpleStringProperty();
11         this.image = new SimpleStringProperty();
12     }
13     public Phone(String smartPhone, String image) {
14         this.smartPhone = new SimpleStringProperty(smartPhone);
15         this.image = new SimpleStringProperty(image);
16     }
17
18     public String getSmartPhone() {
```



```

19         return smartPhone.get();
20     }
21     public void setSmartPhone(String smartPhone) {
22         this.smartPhone.set(smartPhone);
23     }
24     public String getImage() {
25         return image.get();
26     }
27     public void setImage(String image) {
28         this.image.set(image);
29     }
30 }

```

모델의 속성 타입은 컬럼 값의 데이터 타입에 따라서 `javafx.beans.property` 패키지의 `SimpleXXXProperty`를 사용하면 된다. 모델 클래스를 작성했다면 이제 모델 속성과 `TableColumn`을 연결시키는 코드를 작성해야 한다. `TableColumn`은 `TableView`의 `getColumns().get(index)`로 얻어내는데, 첫 번째 컬럼의 인덱스는 0이다.

`TableColumn`의 `setCellValueFactory()` 메소드는 `PropertyValueFactory` 매개값을 이용해서 모델 속성을 `TableColumn`으로 매핑한다. 다음은 첫 번째 `TableColumn`을 모델 클래스의 `smartPhone` 속성과 매핑시키는 코드이다.

```

TableColumn tcSmartPhone = tableView.getColumns().get(0);
tcSmartPhone.setCellValueFactory( new PropertyValueFactory("smartPhone") );

```

셀(cell) 내에서 정렬이 필요한 경우에는 다음과 같이 `TableColumn`의 `setStyle()` 메소드로 CSS를 적용하면 된다. CSS는 10절 JavaFX CSS 스타일에서 학습한다.

```

tcSmartPhone.setStyle("-fx-alignment: CENTER;");

```

`TableView`에 행들을 추가하기 위해서는 `ObservableList`에 모델 객체들을 저장하고, `ObservableList`를 매개값으로 해서 `TableView`의 `setItems()` 메소드를 호출하면 된다.

```
ObservableList phoneList = FXCollections.observableArrayList(
    new Phone("갤럭시S1", "phone01.png"),
    new Phone("갤럭시S2", "phone02.png"),
    new Phone("갤럭시S3", "phone03.png")
);
tableView.setItems( phoneList );
```

TableView에서 선택된 행의 인덱스와 모델 객체를 얻으려면 속성 감시를 이용할 수 있다. `getSelectionModel()` 메소드로 `TableViewSelectionModel`을 얻고 나서, `selectedIndexProperty` 또는 `selectedItemProperty`에 리스너를 설정하면 된다. `selectedIndexProperty`는 선택된 행의 인덱스이고, `selectedItemProperty`는 선택된 행의 모델 객체이다.

다음은 `selectedItemProperty`에 리스너를 설정하는 코드이다.

```
tableView.getSelectionModel().selectedItemProperty().addListener(
    new ChangeListener<Phone>() {
        @Override
        public void changed(ObservableValue<? extends Phone> observable,
                           Phone oldValue, Phone newValue) { ... }
    }
);
```

다음 예제는 `ListView`의 항목을 선택하면 같은 인덱스를 가지는 `TableView`의 행이 자동 선택되도록 한다. 그리고 `TableView`에서 행이 선택되면 이미지 컬럼값을 읽고 `ImageView`에 이미지를 보여준다. 하단의 [확인] 버튼을 클릭하면 `ListView`와 `TableView`에 선택된 정보를 콘솔에 출력한다.

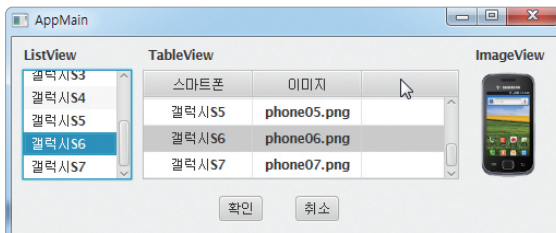
>> root.fxml

```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <?import javafx.scene.text.*?>
4  <?import javafx.scene.shape.*?>
5  <?import javafx.scene.web.*?>
6  <?import javafx.geometry.*?>
```

```

7  <?import javafx.scene.image.*?>
8  <?import java.lang.*?>
9  <?import javafx.scene.control.*?>
10 <?import javafx.scene.layout.*?>
11 <?import javafx.collections.*?>
12
13 <AnchorPane xmlns:fx="http://javafx.com/fxml"
14         fx:controller="sec07.exam03_view.RootController"
15         prefHeight="180.0" prefWidth="500.0" >
16     <children>
17         <Label layoutX="11.0" layoutY="9.0" text="ListView" />
18         <ListView fx:id="listView" layoutX="10.0" layoutY="30.0"
19             prefHeight="100.0" prefWidth="100.0" />
20         <Label layoutX="125.0" layoutY="9.0" text="TableView" />
21         <TableView fx:id="tableView" layoutX="120.0" layoutY="30.0"
22             prefHeight="100.0" prefWidth="290.0">
23             <columns>
24                 <TableColumn prefWidth="100.0" text="스마트폰" />
25                 <TableColumn prefWidth="100.0" text="이미지" />
26             </columns>
27         </TableView>
28         <Label layoutX="425.0" layoutY="9.0" text="ImageView" />
29         <ImageView fx:id="imageView" fitHeight="100.0" fitWidth="60.0"
30             layoutX="430.0" layoutY="30.0" preserveRatio="true" />
31         <Button layoutX="190.0" layoutY="145.0"
32             onAction="#handleBtnOkAction" text="확인" />
33         <Button layoutX="260.0" layoutY="145.0"
34             onAction="#handleBtnCancelAction" text="취소" />
35     </children>
36 </AnchorPane>

```



>> RootController.java

```
1  package sec07.exam03_view;
2
3  import java.net.URL;
4  import java.util.ResourceBundle;
5  import javafx.application.Platform;
6  import javafx.beans.value.ChangeListener;
7  import javafx.beans.value.ObservableValue;
8  import javafx.collections.FXCollections;
9  import javafx.collections.ObservableList;
10 import javafx.event.ActionEvent;
11 import javafx.fxml.FXML;
12 import javafx.fxml.Initializable;
13 import javafx.scene.control.ListView;
14 import javafx.scene.control.TableColumn;
15 import javafx.scene.control.TableView;
16 import javafx.scene.control.cell.PropertyValueFactory;
17 import javafx.scene.image.Image;
18 import javafx.scene.image.ImageView;
19
20 public class RootController implements Initializable {
21     @FXML private ListView<String> listView;
22     @FXML private TableView<Phone> tableView;
23     @FXML private ImageView imageView;
24
25     @Override
26     public void initialize(URL location, ResourceBundle resources) {
27         listView.setItems(FXCollections.observableArrayList(
28             "갤럭시S1", "갤럭시S2", "갤럭시S3", "갤럭시S4", "갤럭시S5", "갤럭시S6", "갤럭시S7"
29         ));
30         listView.getSelectionModel().selectedIndexProperty().addListener(
31             new ChangeListener<Number>() {
32                 @Override
33                 public void changed(ObservableValue<? extends Number> observable,
34                     Number oldValue, Number newValue) {
35                     tableView.getSelectionModel().select(newValue.intValue());
36                     tableView.scrollTo(newValue.intValue());    변경된 인덱스의 행 선택
37                 }                                             변경된 인덱스 위치로 스크롤시킴
38             });                                              selectedIndex 속성 감시

```

```

39
40     ObservableList<Phone> phoneList = FXCollections.observableArrayList(
41         new Phone("갤럭시S1", "phone01.png"),
42         new Phone("갤럭시S2", "phone02.png"),
43         new Phone("갤럭시S3", "phone03.png"),
44         new Phone("갤럭시S4", "phone04.png"),
45         new Phone("갤럭시S5", "phone05.png"),
46         new Phone("갤럭시S6", "phone06.png"),
47         new Phone("갤럭시S7", "phone07.png")
48     );
49
50     TableColumn tcSmartPhone = tableView.getColumns().get(0);
51     tcSmartPhone.setCellValueFactory(
52         new PropertyValueFactory("smartPhone")
53     );
54     tcSmartPhone.setStyle("-fx-alignment: CENTER;");
55
56     TableColumn tcImage = tableView.getColumns().get(1);
57     tcImage.setCellValueFactory(
58         new PropertyValueFactory("image")
59     );
60     tcImage.setStyle("-fx-alignment: CENTER;");
61
62     tableView.setItems(phoneList);
63
64     tableView.getSelectionModel().selectedItemProperty().addListener(
65         new ChangeListener<Phone>() {
66             @Override
67             public void changed(ObservableValue<? extends Phone> observable,
68                 Phone oldValue, Phone newValue) {
69                 if(newValue!=null) {
70                     imageView.setImage(new Image(getClass().getResource(
71                         "images/" + newValue.getImage().toString()));
72                 }
73             }
74         });
75     }
76
77     public void handleBtnOkAction(ActionEvent e) {

```

selectedItem 속성 감시

```

78     String item = listView.getSelectionModel().getSelectedItem();
79     System.out.println("ListView 스마트폰: " + item);
80
81     Phone phone = tableView.getSelectionModel().getSelectedItem();
82     System.out.println("TableView 스마트폰: " + phone.getSmartPhone());
83     System.out.println("TableView 이미지: " + phone.getImage());
84 }
85
86 public void handleBtnCancelAction(ActionEvent e) {
87     Platform.exit();
88 }
89 }

```

선택된 항목(행)의 데이터 얻기

>>> AppMain.java

```

1  package sec07.exam03_view;
2
3  import javafx.application.Application;
4  import javafx.fxml.FXMLLoader;
5  import javafx.scene.Parent;
6  import javafx.scene.Scene;
7  import javafx.stage.Stage;
8
9  public class AppMain extends Application {
10     @Override
11     public void start(Stage primaryStage) throws Exception {
12         Parent root = (Parent)FXMLLoader.load(getClass().getResource("root.
            fxml"));
13         Scene scene = new Scene(root);
14
15         primaryStage.setTitle("AppMain");
16         primaryStage.setScene(scene);
17         primaryStage.show();
18     }
19
20     public static void main(String[] args) {
21         launch(args);
22     }
23 }

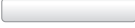


```

미디어 컨트롤

미디어와 관련된 컨트롤에는 비디오를 재생할 수 있는 MediaView 컨트롤, 볼륨 및 재생 위치 조절을 위한 Slider 컨트롤, 진행 상태를 보여주는 ProgressBar, ProgressIndicator 컨트롤 등이 있다.

MediaView



Slider	ProgressBar와 ProgressIndicator
	<p>progress: 0.0  0%</p> <p>progress: 0.6  60%</p> <p>progress: 1.0  Done</p>

1) MediaPlayer와 MediaView 컨트롤

MediaView 컨트롤은 비디오를 보여주는 용도로만 사용되기 때문에 특별한 UI를 가지고 있지 않다. FXML로 MediaView 컨트롤을 선언하는 방법은 다음과 같다.

```
<MediaView fitHeight="200.0" fitWidth="300.0" preserveRatio="false" />
```

fitWidth와 fitHeight는 MediaView의 폭과 높이를 지정한다. preserveRatio는 비디오의 종횡비를 유지할 것인지를 지정하는데, false는 비디오의 종횡비와 상관없이 fitWidth와 fitHeight에 맞게 MediaView의 크기가 고정되고, true는 비디오의 종횡비에 맞게 MediaView 크기가 조정된다.

MediaView 컨트롤은 비디오를 재생하는 기능이 없기 때문에 미디어를 재생하는 MediaPlayer가 필요하다. MediaPlayer는 비디오뿐만 아니라 오디오도 재생할 수 있는데, 미디어 파일 경로를 Media 객체 형태로 전달해서 다음과 같이 생성한다.

```
Media media = new Media("미디어 소스 경로");
MediaPlayer mediaPlayer = new MediaPlayer(media);
```

예를 들어 클래스 경로에 있는 비디오 파일을 재생하는 MediaPlayer는 다음과 같이 생성할 수 있다.

```
Media media = new Media(getClass().getResource("media/bigbuck.m4v").toString());
MediaPlayer mediaPlayer = new MediaPlayer(media);
```

미디어 소스가 비디오라면 MediaView의 setMediaPlayer() 메소드로 MediaPlayer 객체를 등록할 수 있다. 오디오 소스로부터 MediaPlayer가 생성되었다면 MediaView는 필요 없다.

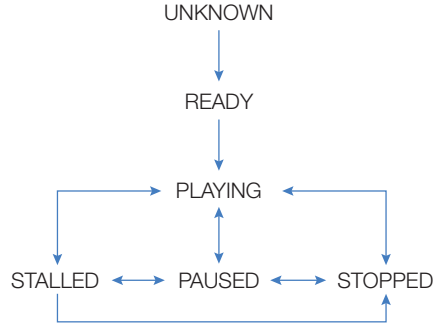
```
mediaView.setMediaPlayer(mediaPlayer);
```

MediaPlayer를 생성했다고 해서 바로 재생할 수는 없고, 재생할 상태(READY)가 될 때까지 기다려야 한다. 다음 표는 MediaPlayer가 가질 수 있는 상태와 상태를 변경하는 메소드가 무엇인지 보여준다.

다음 상태 현재 상태	READY	PAUSED	PLAYING	STALLED	STOPPED
UNKNOWN					
READY			setAutoplay(true) play()		
PAUSED			play()		stop()
PLAYING		pause()		buffering data	stop()
STALLED		pause()	data buffered		stop()
STOPPED		pause()	play()		

UNKNOWN은 MediaPlayer가 생성된 직후의 상태인데, 미디어를 재생할 준비가 되면 READY 상태로 자동 변경된다.

READY 상태에서 `setAutoPlay(true)` 또는 `play()`를 호출하면 PLAYING이 된다. PLAYING 상태에서 `pause()`를 호출하면 PAUSED 상태가 되고, `stop()`을 호출하면 STOPPED 상태가 된다.



만약 PLAYING 상태에서 재생 버퍼에 충분한 데이터가 없을 경우 STALLED 상태가 된다. 주로 네트워크상에서 미디어 소스를 받아 재생할 때 네트워크 속도가 느리면 STALLED 상태가 된다.

위 표의 상태들은 MediaPlayer.Status 열거 타입으로 모두 정의되어 있는데, 코드에서 MediaPlayer의 상태를 알고 싶다면 `getStatus()`의 리턴값을 확인하면 된다.

위 표에는 나와 있지 않지만 EndOfMedia 상태도 있다. EndOfMedia는 MediaPlayer가 미디어를 모두 재생했을 때의 상태를 말한다. EndOfMedia 상태에서 `play()`를 호출하면 다시 PLAYING 상태가 되는데, 처음 위치에서 자동 재생되지 않기 때문에 `seek()` 메소드로 재생 위치를 처음으로 되돌려놓고 `play()`를 호출해야 한다.

상태가 변경되면 자동 실행해야 할 코드들이 있을 수 있다. 이런 코드들은 Runnable의 `run()` 메소드에 작성하고, `setOnXXX()` 메소드로 등록하면 된다. 그러면 해당 상태가 되었을 때 Runnable의 `run()` 메소드가 자동 실행된다. 다음은 각 상태로 변경될 때 실행하는 Runnable을 설정하는 메소드이다.

상태	자동 실행 Runnable 설정 메소드	Runnable에 포함될 수 있는 코드
READY	<code>setOnReady(Runnable r)</code>	<ul style="list-style-type: none"> - 재생 시간을 표시하는 리스너 등록 - 재생 버튼 활성화
PLAYING	<code>setOnPlaying(Runnable r)</code>	<ul style="list-style-type: none"> - 멈춤 및 정지 버튼 활성화
PAUSED	<code>setOnPaused(Runnable r)</code>	<ul style="list-style-type: none"> - 재생 및 정지 버튼 활성화
STOPPED	<code>setOnStopped(Runnable r)</code>	<ul style="list-style-type: none"> - 재생 버튼 활성화
EndOfMedia	<code>setOnEndOfMedia(Runnable r)</code>	<ul style="list-style-type: none"> - 재생 버튼 활성화

다음은 `setOnReady()` 메소드를 작성하는 방법을 보여준다.

```

mediaPlayer.setOnReady(new Runnable() {
    @Override
    public void run() {
        //①재생 버튼을 활성화 코드 또는 setAutoPlay(true);

        //②재생 시간을 알 수 있도록 currentTime 속성 감시
        mediaPlayer.currentTimeProperty().addListener(new ChangeListener<Duration>()
        {
            @Override
            public void changed(ObservableValue<? extends Duration> observable,
                               Duration oldValue, Duration newValue) { ... }

        });
    }
});

```

보통 READY 상태로 변경될 경우에는 PLAYING 상태로 변경할 수 있도록 [재생 버튼]을 활성화하거나 자동 실행을 위해 setAutoPlay(true)를 호출한다. 그리고 재생 시간을 표시하기 위해 currentTime 속성을 감시하는 ChangeListener를 등록할 수 있다.

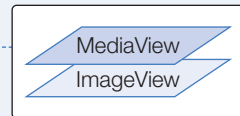
다음 예제는 비디오 파일과 오디오 파일을 재생하는 방법을 보여준다. RootController의 26라인과 27라인 중 하나를 주석처리하고 실행시켜 보자.

>>> root.fxml

```

1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <?import javafx.scene.layout.*?>
4  <?import javafx.scene.media.*?>
5  <?import javafx.scene.image.*?>
6  <?import javafx.scene.control.*?>
7
8  <AnchorPane xmlns:fx="http://javafx.com/fxml"
9      fx:controller="sec07.exam04_mediaview.RootController"
10     prefHeight="220.0" prefWidth="530.0" >
11     <children>
12         <StackPane layoutX="10.0" layoutY="10.0">
13             <children>
14                 <ImageView fx:id="imageView"

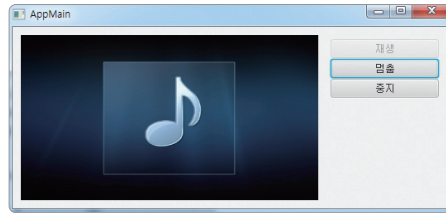
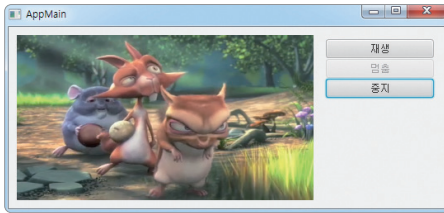
```



```

15         fitHeight="200.0" fitWidth="360.0" preserveRatio="false">
16         <image>Image url="@media/audio.png" /</image>
17     </ImageView>
18     <MediaView fx:id="mediaView"
19         fitHeight="200.0" fitWidth="360.0" preserveRatio="false" />
20 </children>
21 </StackPane>
22 <Button fx:id="btnPlay" layoutX="385.0" layoutY="15.0"
23     prefHeight="23.0" prefWidth="131.0" text="재생" />
24 <Button fx:id="btnPause" layoutX="385.0" layoutY="39.0"
25     prefHeight="23.0" prefWidth="131.0" text="멈춤" />
26 <Button fx:id="btnStop" layoutX="385.0" layoutY="63.0"
27     prefHeight="23.0" prefWidth="131.0" text="중지" />
28 </children>
29 </AnchorPane>

```



>>> RootController.java

```

1  package sec07.exam04_medialog;
2
3  import java.net.URL;
4  import java.util.ResourceBundle;
5  import javafx.fxml.FXML;
6  import javafx.fxml.Initializable;
7  import javafx.scene.control.Button;
8  import javafx.scene.image.ImageView;
9  import javafx.scene.media.Media;
10 import javafx.scene.media.MediaPlayer;
11 import javafx.scene.media.MediaView;

```

```

12
13 public class RootController implements Initializable {
14     @FXML private MediaView mediaView;
15     @FXML private ImageView imageView;
16     @FXML private Button btnPlay;
17     @FXML private Button btnPause;
18     @FXML private Button btnStop;
19
20     //재생 완료를 확인하는 플래그 필드
21     private boolean endOfMedia;
22
23     @Override
24     public void initialize(URL location, ResourceBundle resources) {
25         //미디어 객체 생성
26         //Media media = new Media(getClass().getResource("media/video.m4v").
27             toString());
28         Media media = new Media(getClass().getResource("media/audio.wav").
29             toString());
30
31         //미디어 플레이어 생성 및 미디어 뷰에 설정
32         MediaPlayer mediaPlayer = new MediaPlayer(media);
33         mediaView.setMediaPlayer(mediaPlayer);
34
35         //READY 상태가 될 때 실행할 Runnable 설정
36         mediaPlayer.setOnReady(new Runnable() {
37             @Override
38             public void run() {
39                 btnPlay.setDisable(false); btnPause.setDisable(true);
40                 btnStop.setDisable(true);
41                 if(mediaPlayer.isAutoPlay()) {mediaView.setVisible(false);}
42             }
43         });
44
45         //PLAYING 상태가 될 때 실행할 Runnable 설정
46         mediaPlayer.setOnPlaying(()->{
47             btnPlay.setDisable(true); btnPause.setDisable(false);
48             btnStop.setDisable(false);
49         });
50
51         //PAUSED 상태가 될 때 실행할 Runnable 설정

```

```

48     mediaPlayer.setOnPaused(()->{
49         btnPlay.setDisable(false); btnPause.setDisable(true);
50         btnStop.setDisable(false);
51     });
52     //EndOfMedia 상태가 될 때 실행할 Runnable 설정
53     mediaPlayer.setOnEndOfMedia(()->{
54         endOfMedia = true;
55         btnPlay.setDisable(false); btnPause.setDisable(true);
56         btnStop.setDisable(true);
57     });
58     //STOPPED 상태가 될 때 실행할 Runnable 설정
59     mediaPlayer.setOnStopped(()->{
60         mediaPlayer.seek(mediaPlayer.getStartTime());
61         btnPlay.setDisable(false); btnPause.setDisable(true);
62         btnStop.setDisable(true);
63     });
64     //버튼(ActionEvent) 처리
65     btnPlay.setOnAction(event->{
66         if(endOfMedia) {
67             mediaPlayer.stop(); //재생 중지
68             mediaPlayer.seek(mediaPlayer.getStartTime()); //재생 시간을 처음으로 돌림
69         }
70         mediaPlayer.play(); //재생하기
71         endOfMedia = false;
72     });
73     btnPause.setOnAction(event->mediaPlayer.pause()); //일시 정지
74     btnStop.setOnAction(event->mediaPlayer.stop()); //중지
75 }
76 }

```

>>> AppMain.java

```

1 package sec07.exam04_mediaview;
2
3 import javafx.application.Application;

```

```

4  import javafx.fxml.FXMLLoader;
5  import javafx.scene.Parent;
6  import javafx.scene.Scene;
7  import javafx.stage.Stage;
8
9  public class AppMain extends Application {
10     @Override
11     public void start(Stage primaryStage) throws Exception {
12         Parent root = (Parent)FXMLLoader.load(getClass().getResource("root.
            fxml"));
13         Scene scene = new Scene(root);
14
15         primaryStage.setTitle("AppMain");
16         primaryStage.setScene(scene);
17         primaryStage.show();
18     }
19
20     public static void main(String[] args) {
21         launch(args);
22     }
23 }

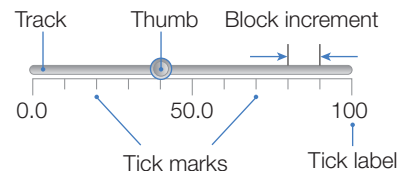
```

2) Slider 컨트롤

Slider 컨트롤은 Track과 Thumb로 구성되어 있다. Slider의 value 속성에는 현재 Thumb의 위치값이 저장되는데, 최소값은 0, 최대값은 100이다.

기본적으로 Tick marks와 Tick label이 숨겨져 있는

데, setShowTickMarks(true)와 setShowTickLabels(true)를 호출하면 볼 수 있다. Block increment 간격은 setBlockIncrement()로 설정할 수 있다. 다음은 FXML로 Slider 컨트롤을 선언하는 방법을 보여준다.



```

<Slider prefHeight= "폭" prefWidth= "높이" showTickLabels="true" showTickMarks=
    "true"/>

```

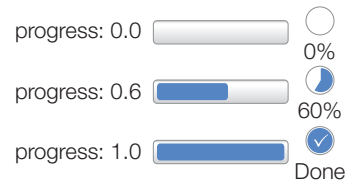
다음 코드는 MediaPlayer의 볼륨을 조정하기 위해 Slider 컨트롤을 사용하는 방법을 보여준다.

```
slider.valueProperty().addListener(new ChangeListener<Number>() {
    @Override
    public void changed(ObservableValue<? extends Number> observable,
        Number oldValue, Number newValue) {
        mediaPlayer.setVolume(sliderVolume.getValue() / 100.0);
    }
});
```

Slider의 value 속성을 감시하기 위해 ChangeListener를 등록하고 changed() 메소드에서 MediaPlayer의 setVolume() 메소드를 호출한다. MediaPlayer의 volume 속성은 0.0~1.0 값을 가지는데, Slider value 속성은 0.0~100.0 값을 가지므로 Slider의 value 속성값을 100.0으로 나누어서 MediaPlayer의 value 속성값으로 설정한다.

3) ProgressBar와 ProgressIndicator 컨트롤

오른쪽 그림과 같이 ProgressBar는 수평 막대 모양의 컨트롤이고, ProgressIndicator는 원형 모양의 컨트롤이다. 둘 다 작업의 진행 정도를 표시하는데, 미디어 재생 시간을 표시하거나 저장소의 사용량 및 네트워크 통신량을 표시할 때도 사용할 수 있다.



다음은 FXML로 선언하는 방법을 보여준다.

```
<ProgressBar prefHeight="15" prefWidth="100" progress="0.0" />
<ProgressIndicator prefHeight="47.0" prefWidth="31.0" progress="0.0" />
```

ProgressBar는 ProgressIndicator를 상속한 하위 컨트롤이기 때문에 사용하는 속성들이 동일하다. 이들 컨트롤의 progress 속성은 진행 정도를 설정하는데, 최소값은 0.0이고 최대값은 1.0이다. 다음은 진행 정도를 변경하는 코드이다.

```
progressBar.setProgress(0.0~1.0);
progressIndicator.setProgress(0.0~1.0);
```

ProgressBar는 ProgressIndicator로 MediaPlayer의 재생 시간을 나타내기 위해서는 현재 재생 시간을 전체 재생 시간으로 나눈값을 progress 속성값으로 설정하면 된다.

```
double currentSeconds = mediaPlayer.getCurrentTime().toSeconds();
double totalSeconds = mediaPlayer.getTotalDuration().toSeconds();
double progress = currentSeconds / totalSeconds;
progressBar.setProgress(progress);
progressIndicator.setProgress(progress);
```

주의할 점은 마지막 재생 시간과 전체 재생 시간이 정확히 일치하지 않기 때문에 마지막 값이 1.0이 되지 않을 수도 있다. 그래서 MediaPlayer가 EndOfMedia 상태가 되었을 때 progress 속성을 강제로 1.0으로 설정하는 것이 좋다.

```
mediaPlayer.setOnEndOfMedia(()->{
    progressBar.setProgress(1.0);
    progressIndicator.setProgress(1.0);
});
```

다음은 이전 예제에서 재생 시간을 표시하도록 ProgressBar와 ProgressIndicator 컨트롤을 추가 하고, 볼륨을 조정하기 위해 Slider 컨트롤을 추가한 것이다.

>> root.fxml

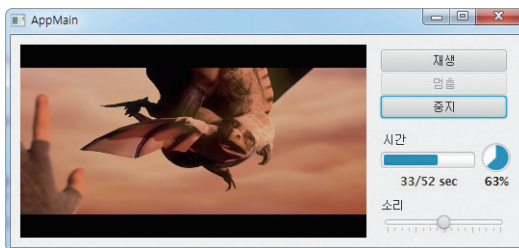
```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <?import javafx.scene.layout.*?>
4  <?import javafx.scene.media.*?>
5  <?import javafx.scene.image.*?>
6  <?import javafx.scene.control.*?>
7
8  <AnchorPane xmlns:fx="http://javafx.com/fxml"
9      fx:controller="sec07.exam05_slider_progressbar.RootController"
10     prefHeight="220.0" prefWidth="530.0" >
11     <children>
12         <StackPane layoutX="10.0" layoutY="10.0">
```



```

13     <children>
14         <ImageView fx:id="imageView"
15             fitHeight="200.0" fitWidth="360.0" preserveRatio="false">
16             <image><Image url="@media/audio.png" /></image>
17         </ImageView>
18         <MediaView fx:id="mediaView"
19             fitHeight="200.0" fitWidth="360.0" preserveRatio="false" />
20     </children>
21 </StackPane>
22 <Button fx:id="btnPlay" layoutX="385.0" layoutY="15.0"
23     prefHeight="23.0" prefWidth="131.0" text="재생" />
24 <Button fx:id="btnPause" layoutX="385.0" layoutY="39.0"
25     prefHeight="23.0" prefWidth="131.0" text="멈춤" />
26 <Button fx:id="btnStop" layoutX="385.0" layoutY="63.0"
27     prefHeight="23.0" prefWidth="131.0" text="중지" />
28
29 <Label layoutX="387.0" layoutY="101.0" text="시간" />
30 <ProgressBar fx:id="progressBar" layoutX="385.0" layoutY="121.0"
31     prefHeight="18.0" prefWidth="98.0" progress="0.0" />
32 <ProgressIndicator fx:id="progressIndicator" layoutX="489.0"
33     layoutY="112.0"
34     prefHeight="47.0" prefWidth="31.0" progress="0.0" />
35 <Label fx:id="labelTime" alignment="CENTER" layoutX="386.0"
36     layoutY="142.0"
37     prefHeight="18.0" prefWidth="98.0" text="0/0 sec" />
38
39 <Label layoutX="385.0" layoutY="169.0" text="소리" />
40 <Slider fx:id="sliderVolume" layoutX="385.0" layoutY="187.0"
41     prefHeight="14.0" prefWidth="131.0" showTickMarks="true" />
42 </children>
43 </AnchorPane>

```



>> RootController.java

```
1  package sec07.exam05_slider_progressbar;
2
3  import java.net.URL;
4  import java.util.ResourceBundle;
5  import javafx.beans.value.ChangeListener;
6  import javafx.beans.value.ObservableValue;
7  import javafx.fxml.FXML;
8  import javafx.fxml.Initializable;
9  import javafx.scene.control.Button;
10 import javafx.scene.control.Label;
11 import javafx.scene.control.ProgressBar;
12 import javafx.scene.control.ProgressIndicator;
13 import javafx.scene.control.Slider;
14 import javafx.scene.image.ImageView;
15 import javafx.scene.media.Media;
16 import javafx.scene.media.MediaPlayer;
17 import javafx.scene.media.MediaView;
18 import javafx.util.Duration;
19
20 public class RootController implements Initializable {
21     @FXML private MediaView mediaView;
22     @FXML private ImageView imageView;
23     @FXML private Button btnPlay;
24     @FXML private Button btnPause;
25     @FXML private Button btnStop;
26
27     @FXML private Label labelTime;
28     @FXML private Slider sliderVolume;
29     @FXML private ProgressBar progressBar;
30     @FXML private ProgressIndicator progressIndicator;
31
32     private boolean endOfMedia;
33
34     @Override
35     public void initialize(URL location, ResourceBundle resources) {
36         //미디어 객체 생성
37         Media media = new Media(getClass().getResource("media/video.mp4").
            toString());
```

```

38      //Media media = new Media(getClass().getResource("media/audio.wav").
        toString());
39
40      //미디어 플레이어 생성 및 미디어 뷰에 설정
41      MediaPlayer mediaPlayer = new MediaPlayer(media);
42      mediaView.setMediaPlayer(mediaPlayer);
43
44      //해당 상태가 되면 실행할 Runnable 설정
45      mediaPlayer.setOnReady(new Runnable() {
46          @Override
47          public void run() {
48              mediaPlayer.currentTimeProperty().addListener(new
49                  ChangeListener<Duration>() {
50                      @Override
51                      public void changed(ObservableValue<? extends Duration>
52                          observable,
53                          Duration oldValue, Duration newValue) {
54                          double progress = mediaPlayer.getCurrentTime().toSeconds() /
55                              mediaPlayer.getTotalDuration().toSeconds();
56                          progressBar.setProgress(progress);
57                          progressIndicator.setProgress(progress);
58                          labelTime.setText(
59                              (int)mediaPlayer.getCurrentTime().toSeconds()+" / "+
60                              (int)mediaPlayer.getTotalDuration().toSeconds()+" sec");
61                      }
62                  });
63
64              btnPlay.setDisable(false); btnPause.setDisable(true);
65              btnStop.setDisable(true);
66              if(mediaPlayer.isAutoPlay()) {mediaView.setVisible(false);}
67          }
68      });
69
70      mediaPlayer.setOnPlaying(()->{
71          btnPlay.setDisable(true); btnPause.setDisable(false);
72          btnStop.setDisable(false);
73      });
74
75      mediaPlayer.setOnPaused(()->{
76          btnPlay.setDisable(false); btnPause.setDisable(true);
77          btnStop.setDisable(false);
78      });

```

```

72     });
73     mediaPlayer.setOnEndOfMedia(()->{
74         progressBar.setProgress(1.0);
75         progressIndicator.setProgress(1.0);
76         endOfMedia = true;
77         btnPlay.setDisable(false); btnPause.setDisable(true);
78         btnStop.setDisable(true);
79     });
80     mediaPlayer.setOnStopped(()->{
81         btnPlay.setDisable(false); btnPause.setDisable(true);
82         btnStop.setDisable(true);
83     });
84     //볼륨 설정
85     sliderVolume.valueProperty().addListener(new ChangeListener<Number>() {
86         @Override
87         public void changed(ObservableValue<? extends Number> observable,
88             Number oldValue, Number newValue) {
89             mediaPlayer.setVolume(sliderVolume.getValue() / 100.0);
90         }
91     });
92     //Slider의 value 초기값 설정
93     sliderVolume.setValue(50.0);
94
95     //버튼 이벤트 처리
96     btnPlay.setOnAction(event->{
97         if(endOfMedia) {
98             mediaPlayer.stop();
99             mediaPlayer.seek(mediaPlayer.getStartTime());
100         }
101         mediaPlayer.play();
102         endOfMedia = false;
103     });
104     btnPause.setOnAction(event->mediaPlayer.pause());
105     btnStop.setOnAction(event->mediaPlayer.stop());
106 }
107 }

```

미디어 재생이 완료되었을 때
progress 속성값을 1.0으로 설정

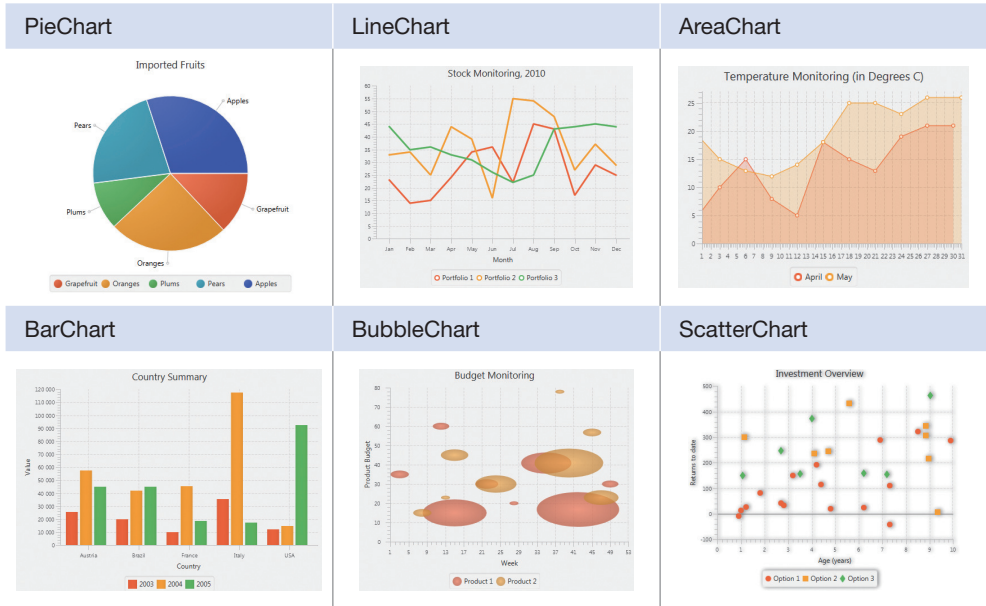
Slider의 value 속성 감시

>>> AppMain.java

```
1  package sec07.exam05_slider_progressbar;
2
3  import javafx.application.Application;
4  import javafx.fxml.FXMLLoader;
5  import javafx.scene.Parent;
6  import javafx.scene.Scene;
7  import javafx.stage.Stage;
8
9  public class AppMain extends Application {
10     @Override
11     public void start(Stage primaryStage) throws Exception {
12         Parent root = (Parent)FXMLLoader.load(getClass().getResource("root.
            fxml"));
13         Scene scene = new Scene(root);
14
15         primaryStage.setTitle("AppMain");
16         primaryStage.setScene(scene);
17         primaryStage.show();
18     }
19
20     public static void main(String[] args) {
21         launch(args);
22     }
23 }
```

차트 컨트롤

JavaFX에는 다음과 같이 다양한 차트를 생성하는 컨트롤이 제공된다. 이 차트의 컨트롤들은 `javafx.scene.chart` 패키지에 포함되어 있다.



FXML로 차트 컨트롤을 배치하는 방법은 매우 쉬운데, PieChart일 경우 다음과 같이 작성하면 된다. PieChart는 X축과 Y축이 없으므로 축으로 정의할 필요가 없다.

```
<PieChart/>
```

LineChart, AreaChart, BarChart일 경우 X축과 Y축이 필요하므로 축 정의가 필요하다. 다음은 BarChart를 선언하는 방법을 보여준다.

```
<BarChart>
  <xAxis>
    <CategoryAxis side="BOTTOM" />
  </xAxis>
  <yAxis>
    <NumberAxis side="LEFT" />
  </yAxis>
</BarChart>
```

〈xAxis〉와 〈yAxis〉는 각각 X축, Y축을 말한다. X축은 위쪽 또는 아래쪽이 있고, Y축은 왼쪽 또는 오른쪽이 있기 때문에 눈금이 나타날 위치를 지정해야 한다. 〈CategoryAxis side="BOTTOM"/〉은 분류 눈금을 아래쪽 축에 나타낸다. 〈NumberAxis side="LEFT"/〉는 숫자 눈금을 왼쪽 축에 나타낸다.

차트의 데이터는 데이터베이스나 네트워크에서 전달받아 사용하는 것이 일반적이므로 컨트롤러에서 Java 코드로 이들 데이터를 얻고 나서 차트 컨트롤에 추가한다.

1) 축이 필요 없는 차트 데이터

X축과 Y축이 필요 없는 PieChart에 데이터를 제공하는 방법은 다음과 같다. 데이터를 PieChart, Date 객체로 생성하고, 이것을 ObservableList에 저장한 다음 PieChart의 setData() 메소드로 전달하면 된다.

```
pieChart.setData(FXCollections.observableArrayList(
    new PieChart.Data("AWT", 10),
    new PieChart.Data("Swing", 30),
    new PieChart.Data("SWT", 25),
    new PieChart.Data("JavaFX", 35)
));
```

2) 축이 필요한 차트 데이터

X축과 Y축이 필요한 LineChart, AreaChart, BarChart에 데이터를 추가하는 방법은 모두 동일하다. 다음은 BarChart에 데이터를 추가하는 방법을 보여준다.

```
//시리즈 생성
XYChart.Series series = new XYChart.Series();
//시리즈 이름 설정
series.setName("남자");
//시리즈 데이터 세팅
series.setData(FXCollections.observableArrayList(
    new XYChart.Data("2015", 70),
    new XYChart.Data("2016", 40),
    new XYChart.Data("2017", 50),
```

```

        new XYChart.Data("2018", 30)
    ));
    //차트에 시리즈 추가
    barChart.getData().add( series );

```

XYChart.Series는 하나의 그래프에 대한 데이터이다. 여러 개의 그래프를 겹쳐 보이게 하려면 XYChart.Series를 그래프의 수에 맞게 생성해야 한다. 데이터는 XYChart.Data 객체로 생성하고, 이것을 ObservableList에 저장한 다음 XYChart.Series의 setData()으로 저장한다.

XYChart.Series가 완성되면 차트의 getData() 메소드로 ObservableList를 얻고 여기에 XYChart.Series를 추가하면 그래프가 생성된다. 다음 예제는 PieChart, BarChart, AreaChart를 생성하는 방법을 보여준다.

>>> root.fxml

```

1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <?import javafx.scene.chart.*?>
4  <?import java.lang.*?>
5  <?import javafx.geometry.*?>
6  <?import javafx.scene.layout.*?>
7  <?import javafx.scene.control.*?>
8
9  <HBox xmlns:fx="http://javafx.com/fxml"
10      fx:controller="sec07.exam06_chart.RootController"
11      prefHeight="250" prefWidth="850" >
12      <children>
13          <PieChart fx:id="pieChart" />
14          <BarChart fx:id="barChart">
15              <xAxis>
16                  <CategoryAxis side="BOTTOM" />
17              </xAxis>
18              <yAxis>
19                  <NumberAxis side="LEFT" />
20              </yAxis>
21          </BarChart>

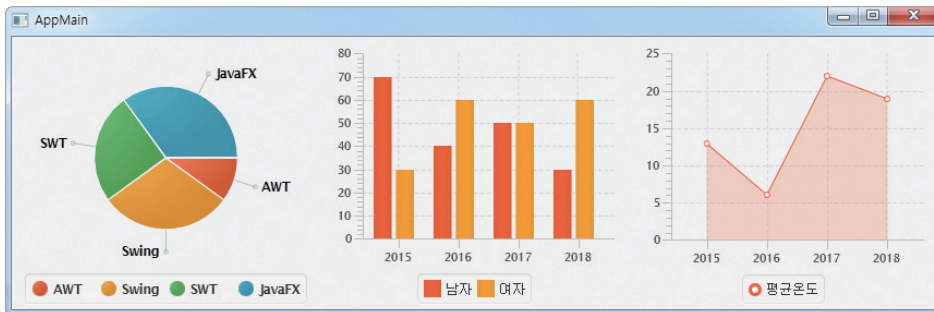
```



```

22     <AreaChart fx:id="areaChart">
23         <xAxis>
24             <CategoryAxis side="BOTTOM" />
25         </xAxis>
26         <yAxis>
27             <NumberAxis side="LEFT" />
28         </yAxis>
29     </AreaChart>
30 </children>
31 </HBox>

```



>>> RootController.java

```

1  package sec07.exam06_chart;
2
3  import java.net.URL;
4  import java.util.ResourceBundle;
5  import javafx.collections.FXCollections;
6  import javafx.fxml.FXML;
7  import javafx.fxml.Initializable;
8  import javafx.scene.chart.AreaChart;
9  import javafx.scene.chart.BarChart;
10 import javafx.scene.chart.PieChart;
11 import javafx.scene.chart.XYChart;
12
13 public class RootController implements Initializable {

```

```

14  @FXML private PieChart pieChart;
15  @FXML private BarChart barChart;
16  @FXML private AreaChart areaChart;
17
18  @Override
19  public void initialize(URL location, ResourceBundle resources) {
20      //PieChart 데이터 세팅
21      pieChart.setData(FXCollections.observableArrayList(
22          new PieChart.Data("AWT", 10),
23          new PieChart.Data("Swing", 30),
24          new PieChart.Data("SWT", 25),
25          new PieChart.Data("JavaFX", 35)
26      ));
27
28      //BarChart 데이터 세팅
29      XYChart.Series series1 = new XYChart.Series();
30      series1.setName("남자");
31      series1.setData(FXCollections.observableArrayList(
32          new XYChart.Data("2015", 70),
33          new XYChart.Data("2016", 40),
34          new XYChart.Data("2017", 50),
35          new XYChart.Data("2018", 30)
36      ));
37      XYChart.Series series2 = new XYChart.Series();
38      series2.setName("여자");
39      series2.setData(FXCollections.observableArrayList(
40          new XYChart.Data("2015", 30),
41          new XYChart.Data("2016", 60),
42          new XYChart.Data("2017", 50),
43          new XYChart.Data("2018", 60)
44      ));
45      barChart.getData().add(series1);
46      barChart.getData().add(series2);
47
48      //AreaChart 데이터 세팅
49      XYChart.Series series3 = new XYChart.Series();
50      series3.setName("평균온도");
51      series3.setData(FXCollections.observableArrayList(
52          new XYChart.Data("2015", 13),
53          new XYChart.Data("2016", 6),

```

• 두 개의 XYChart.Series 추가

```

54     new XYChart.Data("2017", 22),
55     new XYChart.Data("2018", 19)
56 ));
57 areaChart.getData().add(series3);
58 }
59 }

```

>>> AppMain.java

```

1  package sec07.exam06_chart;
2
3  import javafx.application.Application;
4  import javafx.fxml.FXMLLoader;
5  import javafx.scene.Parent;
6  import javafx.scene.Scene;
7  import javafx.stage.Stage;
8
9  public class AppMain extends Application {
10     @Override
11     public void start(Stage primaryStage) throws Exception {
12         Parent root = (Parent)FXMLLoader.load(getClass().getResource("root.
            fxml"));
13         Scene scene = new Scene(root);
14
15         primaryStage.setTitle("AppMain");
16         primaryStage.setScene(scene);
17         primaryStage.show();
18     }
19
20     public static void main(String[] args) {
21         launch(args);
22     }
23 }

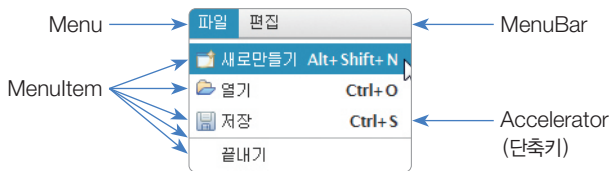
```

08 JavaFX 메뉴바와 툴바

UI 애플리케이션에서 메뉴바와 툴바는 빠질 수 없는 요소이다. JavaFX도 메뉴바와 툴바를 생성할 수 있도록 MenuBar 컨트롤과 Toolbar 컨트롤을 제공한다.

MenuBar 컨트롤

MenuBar 컨트롤은 컨테이너 상단에 배치되어 다양한 작업을 쉽게 선택하도록 해준다. MenuBar에는 Menu들이 배치되는데, 다음 화면에서 “파일”, “편집”이 Menu에 해당된다.



Menu에는 메뉴 아이템으로 MenuItem, CheckMenuItem, RadioMenuItem, CustomMenuItem, SeparatorMenuItem을 추가할 수 있고, 서브 메뉴를 갖는 Menu도 추가할 수 있다. 위 화면에서 “새로만들기”, “열기”, “저장”, “끝내기”가 MenuItem에 해당되고, 가로 구분선은 SeparatorMenuItem이다.

CheckMenuItem, RadioMenuItem은 두 가지 상태를 가지는 메뉴 아이템으로 CheckMenuItem을 클릭하면 선택, 다시 클릭하면 미선택이 된다. 동일한 ToggleGroup을 참조하는 RadioMenuItem들은 하나를 클릭하면 다른 것들이 선택 해제된다.

다음은 FXML로 MenuBar를 선언하는 방법을 보여준다.

```
<MenuBar>
  <menus>
    <Menu text="파일" ... </Menu>
    <Menu text="편집" ... </Menu>
  </menus>
</MenuBar>
```

다음은 “파일” Menu의 “새로만들기” MenuItem을 추가하는 방법을 보여준다.

```

<Menu text="파일">
  <items>
    <MenuItem text="새로만들기" onAction="#handleNew" >
      <accelerator>
        <KeyCodeCombination alt="DOWN" code="N" control="UP" meta="UP"
          shift="DOWN" shortcut="UP" />
      </accelerator>
      <graphic>
        <ImageView×image×Image url="@icons/new.png" />×/image×/ImageView
      </graphic>
    </MenuItem>
  </items>
</Menu>

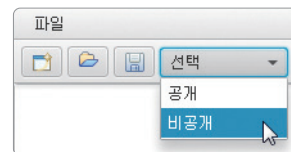
```

MenuItem도 Button과 마찬가지로 클릭하면 onAction 속성에 지정된 컨트롤러의 메소드를 호출해서 ActionEvent를 처리한다. MenuItem은 추가적으로 단축키와 아이콘을 설정할 수 있는데 <accelerator>는 단축키를, <graphic>은 아이콘을 설정한다.

단축키는 KeyCodeCombination 객체로 생성하는데, [Alt] 키, [Control] 키, [Shift] 키, Code 키의 조합으로 구성할 수 있다. DOWN으로 설정된 키와 code 키를 동시에 누르면 MenuItem이 선택되어 onAction 속성에 지정된 메소드가 호출된다. “새로만들기” 메뉴 아이템일 경우 [Alt] + [Shift] + [N]을 동시에 누르면 handleNew() 메소드가 실행된다.

Toolbar 컨트롤

계층적인 작업 선택 기능은 MenuBar 컨트롤이 편리하나, 빠르게 작업을 선택하고 싶다면 Toolbar 컨트롤이 편리하다. Toolbar 컨트롤은 Button, ComboBox와 같은 다른 컨트롤도 배치할 수 있는 컨테이너이다.



다음은 FXML로 Toolbar를 선언하는 방법을 보여준다.

```

<ToolBar>
  <items>
    <Button onAction="#handleNew">

```

```

        <graphic>
            <ImageView><image><Image url="@icons/new.png" /></image></ImageView>
        </graphic>
    </Button>
</items>
</ToolBar>

```

다음 예제는 상단에 MenuBar와 ToolBar를 추가하고 중앙에는 TextArea를 추가해서 간단한 메모장을 만든 것이다. MenuBar의 MenuItem과 ToolBar의 Button을 클릭하면 TextArea에 선택된 작업 내용이 출력된다.

>>> root.fxml

```

1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <?import javafx.scene.layout.*?>
4  <?import javafx.scene.control.*?>
5  <?import javafx.scene.input.*?>
6  <?import javafx.scene.image.*?>
7  <?import javafx.collections.*?>
8  <?import java.lang.*?>
9
10 <BorderPane xmlns:fx="http://javafx.com/fxml"
11     fx:controller="sec08.exam01_menubar_toolbar.RootController"
12     prefHeight="200.0" prefWidth="400.0" >
13     <top>
14         <VBox>
15             <children>
16                 <MenuBar>
17                     <menus>
18                         <Menu text="파일">
19                             <items>
20                                 <MenuItem text="새로만들기" onAction="#handleNew" >
21                                     <accelerator>
22                                         <KeyCodeCombination alt="DOWN" code="N"
23                                             control="UP" meta="UP" shift="DOWN" shortcut="UP" />
24                                     </accelerator>
25                                 <graphic>

```

```

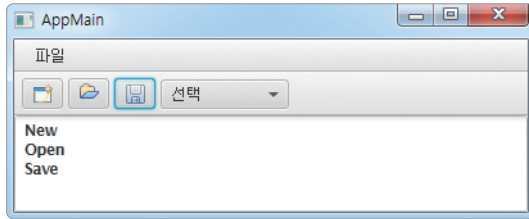
26         <ImageView>
27             <image><Image url="@icons/new.png" /></image>
28         </ImageView>
29     </graphic>
30 </MenuItem>
31 <MenuItem text="열기" onAction="#handleOpen" >
32     <accelerator>
33         <KeyCodeCombination alt="UP" code="O" control="DOWN"
34             meta="UP" shift="UP" shortcut="UP" />
35     </accelerator>
36     <graphic>
37         <ImageView>
38             <image><Image url="@icons/open.png" /></image>
39         </ImageView>
40     </graphic>
41 </MenuItem>
42 <MenuItem text="저장" onAction="#handleSave" >
43     <accelerator>
44         <KeyCodeCombination alt="UP" code="S" control="DOWN"
45             meta="UP" shift="UP" shortcut="UP" />
46     </accelerator>
47     <graphic>
48         <ImageView>
49             <image> <Image url="@icons/save.png" /></image>
50         </ImageView>
51     </graphic>
52 </MenuItem>
53 <SeparatorMenuItem />
54 <MenuItem text="끝내기" onAction="#handleExit"/>
55 </items>
56 </Menu>
57 </menus>
58 </MenuBar>
59
60 <ToolBar>
61     <items>
62         <Button onAction="#handleNew">
63             <graphic>
64                 <ImageView>
65                     <image><Image url="@icons/new.png" /></image>

```

```

66         </ImageView>
67     </graphic>
68 </Button>
69 <Button onAction="#handleOpen">
70     <graphic>
71         <ImageView>
72             <image><Image url="@icons/open.png" /></image>
73         </ImageView>
74     </graphic>
75 </Button>
76 <Button onAction="#handleSave">
77     <graphic>
78         <ImageView>
79             <image><Image url="@icons/save.png" /></image>
80         </ImageView>
81     </graphic>
82 </Button>
83 <ComboBox prefWidth="100" promptText="선택" >
84     <items>
85         <FXCollections fx:factory="observableArrayList">
86             <String fx:value="공개"/>
87             <String fx:value="비공개"/>
88         </FXCollections>
89     </items>
90 </ComboBox>
91 </items>
92 </ToolBar>
93 </children>
94 </VBox>
95 </top>
96
97 <center>
98     <TextArea fx:id="textArea"/>
99 </center>
100 </BorderPane>

```

>>> RootController.java

```
1  package sec08.exam01_menubar_toolbar;
2
3  import java.net.URL;
4  import java.util.ResourceBundle;
5  import javafx.application.Platform;
6  import javafx.event.ActionEvent;
7  import javafx.fxml.FXML;
8  import javafx.fxml.Initializable;
9  import javafx.scene.control.TextArea;
10
11 public class RootController implements Initializable {
12     @FXML private TextArea textArea;
13
14     @Override
15     public void initialize(URL location, ResourceBundle resources) {
16     }
17
18     public void handleNew(ActionEvent e) {
19         textArea.appendText("New\n");
20     }
21
22     public void handleOpen(ActionEvent e) {
23         textArea.appendText("Open\n");
24     }
25
26     public void handleSave(ActionEvent e) {
27         textArea.appendText("Save\n");
28     }
29 }
```

```
29
30     public void handleExit(ActionEvent e) {
31         Platform.exit();
32     }
33 }
```

>>> AppMain.java

```
1     package sec08.exam01_menubar_toolbar;
2
3     import javafx.application.Application;
4     import javafx.fxml.FXMLLoader;
5     import javafx.scene.Parent;
6     import javafx.scene.Scene;
7     import javafx.stage.Stage;
8
9     public class AppMain extends Application {
10         @Override
11         public void start(Stage primaryStage) throws Exception {
12             Parent root = (Parent)FXMLLoader.load(getClass().getResource("root.
13                 fxml"));
14             Scene scene = new Scene(root);
15
16             primaryStage.setTitle("AppMain");
17             primaryStage.setScene(scene);
18             primaryStage.show();
19         }
20
21         public static void main(String[] args) {
22             launch(args);
23         }
24     }
```

09 JavaFX 다이얼로그

다이얼로그(Dialog)는 주 윈도우에서 알림 또는 사용자의 입력을 위해서 실행되는 서브 윈도우라고 볼 수 있다. 다이얼로그는 자체적으로 실행될 수 없고 주 윈도우에 의해 실행되는데, 다이얼로그를 띄우는 주 윈도우를 다이얼로그의 소유자(owner) 윈도우라고 한다.

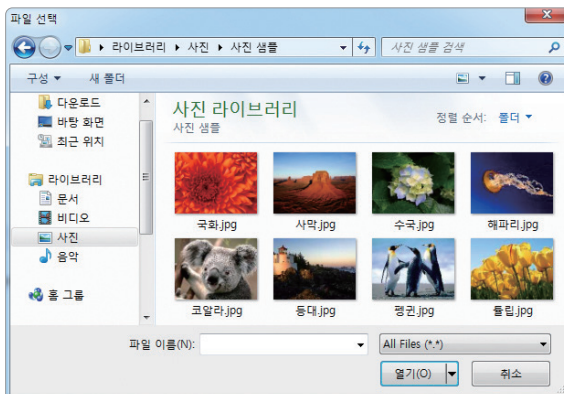
다이얼로그에는 모달(modal)과 모달리스(modeless) 두 가지 종류가 있다. 모달 다이얼로그는 다이얼로그를 닫기 전까지 소유자 윈도우를 사용할 수 없고, 모달리스 다이얼로그는 소유자 윈도우를 계속 사용할 수 있다.

JavaFX에서 제공하는 다이얼로그 종류에는 파일을 선택하는 FileChooser, 디렉토리를 선택하는 DirectoryChooser, 팝업창을 띄우는 Popup 등이 있다. 이 다이얼로그들은 javafx.stage 패키지에 모두 포함되어 있다. 다이얼로그도 윈도우이므로 Stage로 생성되기 때문이다.

FileChooser, DirectoryChooser

FileChooser는 PC의 파일을 선택할 수 있는 다이얼로그이다. 열기 또는 저장 옵션으로 실행할 수 있고, 파일 확장명을 필터링해서 원하는 파일만 볼 수 있다. FileChooser는 컨트롤이 아니기 때문에 FXML에서 선언할 수 없고, Java 코드로 FileChooser를 생성하고 showOpenDialog() 또는 showSaveDialog()를 호출해야 띄울 수 있다. 다음은 열기 옵션으로 FileChooser를 띄우는 코드이다.

```
FileChooser fileChooser = new FileChooser();
File selectedFile = fileChooser.showOpenDialog(primaryStage);
```



`showOpenDialog()` 또는 `showSaveDialog()`를 호출할 때에는 소유자 윈도우를 제공해야 한다. 위 코드에서는 `primaryStage`를 제공했다. `FileChooser`는 모달 다이얼로그이므로 [열기] 또는 [저장] 버튼을 클릭하거나 [취소] 버튼을 클릭하기 전까지는 소유자 윈도우를 사용할 수 없다.

기본적으로 All Files (*.*)가 선택되어 있기 때문에 모든 파일을 볼 수 있다. 만약 파일 확장명으로 필터링해서 파일을 보려면 다음과 같이 `ExtensionFilter`를 생성해서 `FileChooser`에 추가하면 된다.

```
FileChooser fileChooser = new FileChooser();

//파일 확장명으로 필터링 정보 추가
fileChooser.getExtensionFilters().addAll(
    new ExtensionFilter("Text Files", "*.txt"),
    new ExtensionFilter("Image Files", "*.png", "*.jpg", "*.gif"),
    new ExtensionFilter("Audio Files", "*.wav", "*.mp3", "*.aac"),
    new ExtensionFilter("All Files", "*.*)");

File selectedFile = fileChooser.showOpenDialog(primaryStage);
```

파일을 선택하고 [열기] 또는 [저장] 버튼을 클릭하면, `showOpenDialog()` 또는 `showSaveDialog()` 메소드는 선택된 파일의 `File` 객체를 리턴한다. 선택된 파일의 전체 경로를 문자열로 얻고 싶다면 `File`의 `getPath()`를 호출하면 된다.

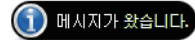
```
File selectedFile = fileChooser.showOpenDialog(primaryStage);
String selectedFilePath = selectedFile.getPath();
```

파일이 아니라 디렉토리(폴더)를 선택하고 싶다면 `FileChooser` 대신 `DirectoryChooser`를 사용하면 된다. 사용 방법은 `FileChooser`와 비슷하다.

```
DirectoryChooser directoryChooser = new DirectoryChooser();
File selectedDir = directoryChooser.showDialog(primaryStage);
String selectedDirPath = selectedDir.getPath();
```

Popup

Popup은 투명한 컨테이너를 제공하는 모달리스 다이얼로그이다. 따라서 소유자 윈도우는 계속 사용될 수 있다. Popup은 컨트롤의 툴팁^{tooltip}, 메시지 통지^{notification}, 드롭다운 박스^{drop down boxes}, 마우스 오른쪽 버튼을 클릭했을 때 나타나는 메뉴 등을 만들 때 사용될 수 있다.



예를 들어 오른쪽 그림과 같이 메시지 통지를 알려주는 Popup을 만들 수 있다.

Popup은 윈도우의 기본 장식(아이콘, 제목, 최소화 및 복원 버튼, 닫기 버튼)을 가지고 있지 않다. Popup의 내용은 Java 코드로 작성하거나 FXML 파일로 작성할 수 있다. 다음은 FXML 파일을 로딩해서 Popup의 내용으로 추가하는 코드이다.

```
Popup popup = new Popup();
popup.getContent().add(FXMLLoader.load(getClass().getResource("popup.fxml")));
```

Popup을 실행하려면 다음과 같이 show() 메소드를 호출하면 된다.

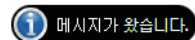
```
popup.show(primaryStage); //PC 화면 정중앙에서 실행
popup.show(primaryStage, anchorX, anchorY); //지정된 좌표에서 실행
```

anchorX와 anchorY를 지정하지 않으면 PC 화면의 중앙에 Popup이 실행된다. 다른 위치에서 실행하고 싶다면 PC 화면의 좌상단(0,0)을 기준으로 anchorX, anchorY를 지정하면 된다.

Popup은 다른 윈도우보다 최상위 위치에 놓이게 되므로 화면에서 항상 제일 위에 나타난다. Popup을 숨기려면 Popup 안에 있는 컨트롤의 이벤트를 처리해서 hide() 메소드를 호출해야 한다. 또 다른 방법은 setAutoHide(true)를 설정해서 다른 윈도우로 포커스가 이동할 때 자동으로 Popup을 숨기도록 해야 한다.

```
popup.hide(); //수동으로 닫을 때 호출
popup.setAutoHide(true); //다른 윈도우로 포커스가 이동할 때 자동으로 닫히도록 설정
```

다음은 Popup에 들어갈 내용으로 HBox를 FXML 파일로 정의한 것이다. HBox에 CSS를 적용해서 배경을 검은색으로, 모서리는 둥글게 변경하고 Label의 글자색을 흰색으로 변경했다. 주목할 점은 컨트롤에 id 속성이 있다는 점이다.



```

<HBox xmlns:fx="http://javafx.com/fxml"
      alignment="CENTER_LEFT"
      style="-fx-background-color: black; -fx-background-radius: 20;" >
    <children>
      <ImageView id="imgMessage"
                  fitHeight="30" fitWidth="30" preserveRatio="true"/>
      <Label id="lblMessage" style="-fx-text-fill: white;"
            <HBox.margin>
              <Insets left="5.0" right="5.0" />
            </HBox.margin>
      </Label>
    </children>
</HBox>

```

Popup의 정적인 내용은 FXML 파일에 정의하면 되지만, 상황에 따라 변경되는 동적 내용은 FXML 파일을 로딩한 후에 변경해야 한다. 변경이 필요한 컨트롤은 컨테이너의 lookup("#id") 메소드를 이용해서 찾을 수 있는데, 이 메소드는 컨트롤의 id 속성값을 이용해서 컨트롤을 찾는다.

다음 코드는 HBox가 로딩된 후에 내부의 ImageView와 Label의 참조를 얻어 동적으로 설정을 바꾸는 방법을 보여준다.

```

//FXML 파일 로딩
HBox hbox = (HBox) FXMLLoader.load(getClass().getResource("xxx.fxml"));

//HBox 내부에 있는 ImageView와 Label 컨트롤 참조 얻기
ImageView imageView = (ImageView) hbox.lookup("#imgMessage");
Label lblMessage = (Label)parent.lookup("#lblMessage");

//ImageView에 이미지를 설정하고 마우스로 클릭했을 때 Popup을 닫도록 이벤트 처리
imageView.setImage(new Image(getClass().getResource("images/dialog-info.png").
    toString()));
imageView.setOnMouseClicked(event->popup.hide());

//Label의 글자를 설정
lblMessage.setText("메시지가 왔습니다.");

```

Popup 내용이 완성되었다면 다음 코드로 Popup을 띄우면 된다.

```
//Popup 생성
Popup popup = new Popup();

//Popup의 내용으로 hbox를 설정
popup.getContent().add(hbox);

//다른 윈도우로 포커스가 이동될 때 Popup을 자동으로 숨기도록 설정
popup.setAutoHide(true);

//Popup 띄우기(소유자 윈도우는 primaryStage로 설정)
popup.show(primaryStage);
```

커스텀 다이얼로그

다양한 내용의 다이얼로그를 만들고 싶다면 Stage로 직접 생성하면 된다. 그리고 기본적인 다이얼로그 설정을 다음과 같이 하면 된다.

```
//Stage 객체 생성
Stage dialog = new Stage(StageStyle.UTILITY);
//모달리스로 설정
dialog.initModality(Modality.WINDOW_MODAL);
//소유자 윈도우 설정
dialog.initOwner(primaryStage);
//다이얼로그 제목 설정
dialog.setTitle("확인");
```

Stage 생성자 매개값에는 윈도우 스타일을 결정짓는 StageStyle 열거 상수가 온다. 다음은 StageStyle 열거 상수와 윈도우 스타일을 설명한 표이다.

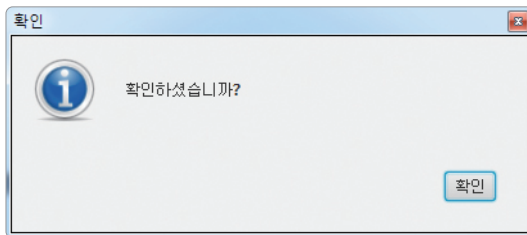
StageStyle 열거 상수	설명
DECORATED	일반적인 윈도우 스타일. 배경이 흰색, 제목줄에 장식(아이콘, 타이틀, 축소, 복원, 닫기 버튼 장식)이 있음
UNDECORATED	배경이 흰색, 제목줄 없음

TRANSPARENT	배경이 투명, 제목줄 없음
UNIFIED	DECORATED와 동일하나 내용물의 경계선이 없음
UTILITY	배경이 흰색이고, 제목줄에 타이틀, 종료 버튼만 있음

만약 매개값이 없는 기본 생성자로 Stage를 생성했다면 DECORATED 윈도우가 생성된다. `initModality(Modality.WINDOW_MODAL)`은 모달 다이얼로그로 설정한다. 이 설정을 하지 않으면 기본적으로 모달리스가 된다. `initOwner(primaryStage)`는 소유자 윈도우^{Stage}를 설정한다.

다음은 커스텀 다이얼로그에 들어갈 내용을 `AnchorPane`으로 정의한 FXML 파일이다.

```
<AnchorPane xmlns:fx="http://javafx.com/fxml"
  prefHeight="150.0" prefWidth="400.0" >
  <children>
    <ImageView fitHeight="50" fitWidth="50"
      layoutX="15" layoutY="15" preserveRatio="true">
      <image><Image url="@images/dialog-info.png" /></image>
    </ImageView>
    <Label id="txtTitle" prefHeight="15.0" prefWidth="290.0"
      layoutX="87.0" layoutY="33.0" />
    <Button id="btnOk" layoutX="336.0" layoutY="104.0" text="확인" />
  </children>
</AnchorPane>
```



다음 코드는 `AnchorPane`가 로딩된 후에 내부의 `ImageView`와 `Label`의 참조를 얻어 동적으로 설정을 바꾸는 방법을 보여준다.


```
//FXML 파일 로딩
AnchorPane anchorPane = (AnchorPane) FXMLLoader.load(getClass().getResource("xxx.fxml"));

//Label의 글자를 변경
Label txtTitle = (Label) parent.lookup("#txtTitle");
txtTitle.setText("확인하셨습니다까?");

//버튼의 이벤트 처리
Button btnOk = (Button) parent.lookup("#btnOk");
btnOk.setOnAction(event->dialog.close());
```

커스텀 다이얼로그의 내용이 완성되었다면 다음과 같이 커스텀 다이얼로그의 내용을 설정하고 띄우면 된다.

```
//다이얼로그 내용을 Scene 객체로 만들고 다이얼로그에 설정
Scene scene = new Scene(parent);
dialog.setScene(scene);

//다이얼로그 창 크기를 변경하지 못하도록 설정(내용이 AnchorPane일 경우 필요)
dialog.setResizable(false);

//다이얼로그 띄우기
dialog.show();
```

primaryStage 참조 얻기

컨트롤러에서 다이얼로그를 실행할 때는 소유자 윈도우가 될 primaryStage가 필요하다. 컨트롤러에서 primaryStage를 얻는 방법은 두 가지가 있다.

1) 메인 클래스에서 전달하는 방법

primaryStage는 메인 클래스의 start() 매개값으로 전달되기 때문에 start() 메소드에서 컨트롤러로 primaryStage를 전달하면 된다.

FXML 루트 태그의 fx:controller 속성에 지정된 컨트롤러 클래스는 FXMLLoader가 FXML을 로

딩할 때 객체로 생성된다. 그리고 FXMLLoader의 `getController()` 메소드로 참조를 얻을 수 있다. 그러나 `getController()` 메소드는 인스턴스 메소드이기 때문에 FXMLLoader 객체가 필요하다. FXML을 FXMLLoader의 정적 메소드인 `load()`로 로딩하면 FXMLLoader 객체를 얻을 수 없기 때문에 FXMLLoader 객체를 직접 생성하고 다음과 같이 인스턴스 메소드인 `load()`를 이용해서 FXML을 로딩해야 한다.

```
public class AppMain extends Application {
    @Override
    public void start(Stage primaryStage) throws Exception {
        //FXML 로딩
        FXMLLoader loader = new FXMLLoader(getClass().getResource("root.fxml"));
        Parent root = loader.load();

        //생성된 Controller 객체를 얻어 primaryStage 전달
        RootController controller = loader.getController();
        controller.setPrimaryStage(primaryStage);

        Scene scene = new Scene(root);
        primaryStage.setTitle("AppMain");
        primaryStage.setScene(scene);
        primaryStage.show();
    }
    ...
}
```

위 코드는 컨트롤러의 `setPrimaryStage()` 메소드를 이용해서 `primaryStage`를 전달하고 있기 때문에 컨트롤러는 다음과 같이 필드와 Setter 메소드를 추가해야 한다.

```
public class RootController implements Initializable {
    private Stage primaryStage;
    public void setPrimaryStage(Stage primaryStage) {
        this.primaryStage = primaryStage;
    }
    ...
}
```

2) 컨테이너 또는 컨트롤로부터 얻는 방법

컨테이너나 컨트롤의 `getScene()` 메소드는 자신이 포함된 `Scene` 객체를 리턴한다. 그리고 `Scene`의 `getWindow()` 메소드는 자신을 보여주는 `Stage` 객체를 리턴한다. 따라서 다음과 같은 코드를 이용하면 컨트롤러에서 `primaryStage`를 얻을 수 있다.

```
Stage primaryStage = (Stage) 컨트롤.getScene().getWindow();
```

주의할 점은 위 코드는 `initialize()` 메소드 안에서는 사용할 수 없다. 그 이유는 아직 `primaryStage`가 준비되지 않았기 때문이다. 메인 클래스의 `start()` 메소드에서 `Scene` 객체를 생성하고, `primaryStage`에 `Scene`을 설정해야만 위 코드가 정상적으로 동작한다. 따라서 이벤트 처리 메소드 내에서만 위 코드를 사용해야 한다.

예를 들어 `Button`을 클릭했을 때 실행하는 메소드가 `handleButton()`이라면 `handleButton()` 메소드 내부에서 다음과 같이 `primaryStage`를 얻어야 한다.

```
public class RootController implements Initializable {
    @FXML private Button button;

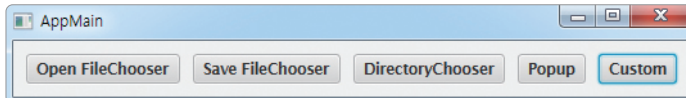
    @Override
    public void initialize(URL location, ResourceBundle resources) {
        ...
    }

    public void handleButton(ActionEvent e) {
        Stage primaryStage = (Stage) button.getScene().getWindow();
        ...
    }
}
```

다음 예제는 지금까지 설명한 다이얼로그들을 차례대로 버튼을 클릭해서 띄운다. 첫 번째 버튼은 파일 오픈 다이얼로그를, 두 번째 버튼은 파일 저장 다이얼로그를, 세 번째 버튼은 디렉토리 선택 다이얼로그를, 네 번째 버튼은 팝업을, 다섯 번째 버튼은 커스텀 다이얼로그를 띄운다.

>>> root.fxml

```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <?import javafx.scene.layout.*?>
4  <?import javafx.scene.control.*?>
5  <?import javafx.geometry.*?>
6
7  <HBox xmlns:fx="http://javafx.com/fxml"
8      fx:controller="sec09.exam01_dialog.RootController"
9      alignment="TOP_LEFT" spacing="10.0" >
10     <children>
11         <Button text="Open FileChooser" onAction="#handleOpenFileChooser"/>
12         <Button text="Save FileChooser" onAction="#handleSaveFileChooser"/>
13         <Button text="DirectoryChooser" onAction="#handleDirectoryChooser"/>
14         <Button text="Popup" onAction="#handlePopup"/>
15         <Button text="Custom" onAction="#handleCustom"/>
16     </children>
17     <padding>
18         <Insets bottom="10.0" left="10.0" right="10.0" top="10.0" />
19     </padding>
20 </HBox>
```



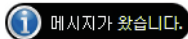
>>> popup.xml

```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <?import javafx.geometry.*?>
4  <?import java.lang.*?>
5  <?import javafx.scene.control.*?>
6  <?import javafx.scene.image.*?>
7  <?import javafx.scene.layout.*?>
```

```

8
9 <HBox xmlns:fx="http://javafx.com/fxml"
10     alignment="CENTER_LEFT"
11     style="-fx-background-color: black; -fx-background-radius: 20;" >
12     <children>         HBox에 CSS를 적용해서 배경을 검은색으로, 모서리는 둥글게 변경
13         <ImageView id="imgMessage"
14             fitHeight="30" fitWidth="30" preserveRatio="true"/>
15         <Label id="lblMessage" style="-fx-text-fill: white;">
16             <HBox.margin>         Label의 글자색을 흰색으로 변경
17                 <Insets left="5.0" right="5.0" />
18             </HBox.margin>
19         </Label>
20     </children>
21 </HBox>

```

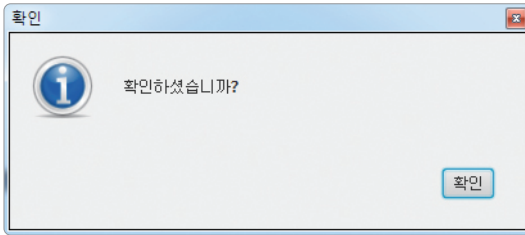


>>> custom_dialog.xml

```

1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <?import javafx.scene.layout.*?>
4 <?import javafx.scene.control.*?>
5 <?import javafx.scene.image.*?>
6
7 <AnchorPane xmlns:fx="http://javafx.com/fxml"
8     prefHeight="150.0" prefWidth="400.0" >
9     <children>
10         <ImageView fitHeight="50" fitWidth="50"
11             layoutX="15" layoutY="15" preserveRatio="true">
12             <image><Image url="@images/dialog-info.png" /></image>
13         </ImageView>
14         <Label id="txtTitle" prefHeight="15.0" prefWidth="290.0"
15             layoutX="87.0" layoutY="33.0" />
16         <Button id="btnOk" layoutX="336.0" layoutY="104.0" text="확인" />
17     </children>
18 </AnchorPane>

```



>>> RootController.java

```
1  package sec09.exam01_dialog;
2
3  import java.io.File;
4  import java.net.URL;
5  import java.util.ResourceBundle;
6  import javafx.event.ActionEvent;
7  import javafx.fxml.FXMLLoader;
8  import javafx.fxml.Initializable;
9  import javafx.scene.Parent;
10 import javafx.scene.Scene;
11 import javafx.scene.control.Button;
12 import javafx.scene.control.Label;
13 import javafx.scene.image.Image;
14 import javafx.scene.image.ImageView;
15 import javafx.stage.DirectoryChooser;
16 import javafx.stage.FileChooser;
17 import javafx.stage.FileChooser.ExtensionFilter;
18 import javafx.stage.Modality;
19 import javafx.stage.Popup;
20 import javafx.stage.Stage;
21 import javafx.stage.StageStyle;
22
23 public class RootController implements Initializable {
24     @Override
25     public void initialize(URL location, ResourceBundle resources) {
26     }
27
28     private Stage primaryStage;
```

```

29     public void setPrimaryStage(Stage primaryStage) {
30         this.primaryStage = primaryStage;
31     }
32
33     //파일 열기 다이얼로그 실행
34     public void handleOpenFileChooser(ActionEvent e) {
35         FileChooser fileChooser = new FileChooser();
36         fileChooser.getExtensionFilters().addAll(
37             new ExtensionFilter("Text Files", "*.txt"),
38             new ExtensionFilter("Image Files", "*.png", "*.jpg", "*.gif"),
39             new ExtensionFilter("Audio Files", "*.wav", "*.mp3", "*.aac"),
40             new ExtensionFilter("All Files", "*.*"));
41         File selectedFile = fileChooser.showOpenDialog(primaryStage);
42         if (selectedFile != null) {
43             System.out.println(selectedFile.getPath());
44         }
45     }
46
47     //파일 저장 다이얼로그 실행
48     public void handleSaveFileChooser(ActionEvent e) {
49         FileChooser fileChooser = new FileChooser();
50         fileChooser.getExtensionFilters().add(new ExtensionFilter("All
51             Files", "*.*"));
52         File selectedFile = fileChooser.showSaveDialog(primaryStage);
53         if (selectedFile != null) {
54             System.out.println(selectedFile.getPath());
55         }
56     }
57
58     //디렉토리 선택 다이얼로그 실행
59     public void handleDirectoryChooser(ActionEvent e) {
60         DirectoryChooser directoryChooser = new DirectoryChooser();
61         File selectedDir = directoryChooser.showDialog(primaryStage);
62         if (selectedDir != null) {
63             System.out.println(selectedDir.getPath());
64         }
65     }
66
67     //Popup 다이얼로그 실행
68     public void handlePopup(ActionEvent e) throws Exception {

```

```

68     Popup popup = new Popup();
69
70     Parent parent = FXMLLoader.load(getClass().getResource("popup.fxml"));
71     ImageView imageView = (ImageView) parent.lookup("#imgMessage");
72     imageView.setImage(new Image(
73         getClass().getResource("images/dialog-info.png").toString());
74     imageView.setOnMouseClicked(event->popup.hide());
75     Label lblMessage = (Label)parent.lookup("#lblMessage");
76     lblMessage.setText("메시지가 왔습니다.");
77
78     popup.getContent().add(parent);
79     popup.setAutoHide(true);
80     popup.show(primaryStage);
81 }
82
83 //커스텀 다이얼로그 실행
84 public void handleCustom(ActionEvent e) throws Exception {
85     Stage dialog = new Stage(StageStyle.UTILITY);
86     dialog.initModality(Modality.WINDOW_MODAL);
87     dialog.initOwner(primaryStage);
88     dialog.setTitle("확인");
89
90     Parent parent = FXMLLoader.load(getClass().getResource("custom_dialog.
91         fxml"));
92     Label txtTitle = (Label) parent.lookup("#txtTitle");
93     txtTitle.setText("확인하셨습니까?");
94     Button btnOk = (Button) parent.lookup("#btnOk");
95     btnOk.setOnAction(event->dialog.close());
96     Scene scene = new Scene(parent);
97
98     dialog.setScene(scene);
99     dialog.setResizable(false);
100     dialog.show();
101 }

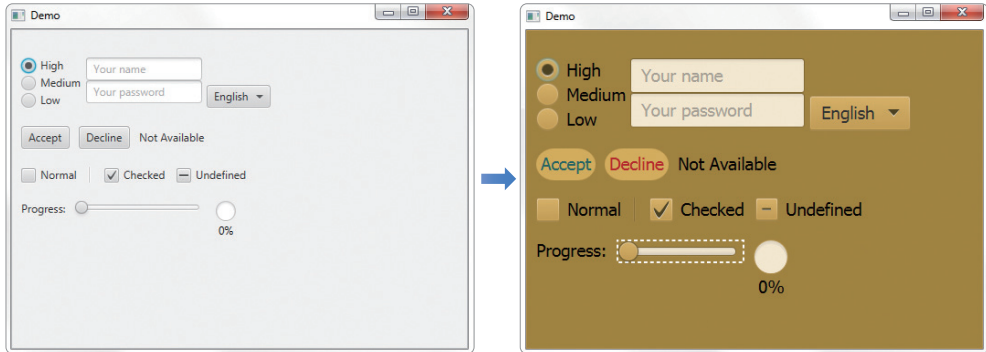
```


>>> AppMain.java

```
1  package sec09.exam01_dialog;
2
3  import javafx.application.Application;
4  import javafx.fxml.FXMLLoader;
5  import javafx.scene.Parent;
6  import javafx.scene.Scene;
7  import javafx.stage.Stage;
8
9  public class AppMain extends Application {
10     @Override
11     public void start(Stage primaryStage) throws Exception {
12         FXMLLoader loader = new FXMLLoader(getClass().getResource("root.fxml"));
13         Parent root = loader.load();
14         RootController controller = loader.getController();
15         controller.setPrimaryStage(primaryStage);
16
17         Scene scene = new Scene(root);
18
19         primaryStage.setTitle("AppMain");
20         primaryStage.setScene(scene);
21         primaryStage.show();
22     }
23
24     public static void main(String[] args) {
25         launch(args);
26     }
27 }
```

10 JavaFX CSS 스타일

JavaFX UI를 담당하는 컨테이너 및 컨트롤은 CSS(Cascading Style Sheets)를 적용해서 모양 및 색상 등을 변경할 수 있다. 이것은 HTML에 CSS를 적용하는 것과 유사하다. 다음은 JavaFX 애플리케이션에 CSS를 적용하면 UI 스킨이 어떻게 달라지는지 보여준다.



JavaFX CSS는 W3C CSS 버전 2.1 스펙(<http://www.w3.org/TR/CSS21>)에 따르기 때문에 CSS로 HTML 문서의 스타일을 작성해 본 개발자는 쉽게 JavaFX CSS를 정의할 수 있다. 단, JavaFX CSS의 속성명은 W3C CSS 속성명 앞에 “-fx-”가 더 붙는 것이 차이점이다.

JavaFX 컨트롤의 기본 CSS는 `modena.css` 파일에 작성되어 있다. 이 파일은 `javafx.controls.jar` 모듈에 포함되어 있다.

```
com/sun/javafx/scene/control/skin/modena/modena.css
```

커스텀 CSS는 기본 CSS의 속성값을 변경하거나 새로운 속성을 정의한 것을 말한다. 커스텀 CSS를 적용하는 방법에는 컨테이너 또는 컨트롤을 정의할 때 직접 설정하는 인라인 스타일 방식과 외부 CSS 파일을 생성하고 Scene에 적용하는 방식이 있다.

인라인 스타일

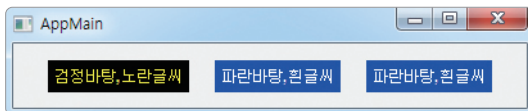
인라인^{inline} 스타일은 컨테이너 또는 컨트롤의 `style` 속성값으로 직접 CSS를 작성하는 방식을 말한다. 작성이 쉬우며 빠르게 모양과 색상을 변경할 수 있다는 장점이 있다.

```
<컨테이너 style="속성:값; 속성:값; ...">
<컨트롤 style="속성:값; 속성:값; ...">
```

다음 예제는 세 개의 Label을 FXML로 선언하고, Label의 style 속성을 이용해서 첫 번째 Label은 배경을 검은색, 글자는 노란색으로 CSS를 적용하고, 두 번째, 세 번째 Label은 배경을 파란색, 글자는 흰색으로 CSS를 적용했다.

>> root.fxml

```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <?import javafx.scene.layout.*?>
4  <?import javafx.scene.control.*?>
5  <?import java.lang.*?>
6
7  <HBox xmlns:fx="http://javafx.com/fxml"
8      prefHeight="50" prefWidth="400" alignment="CENTER" spacing="20">
9      <children>
10         <Label text="검정바탕,노란글씨"
11             style="-fx-background-color: black; -fx-text-fill: yellow;
12                 -fx-padding: 5;"/>
13         <Label text="파란바탕,흰글씨"
14             style="-fx-background-color: blue; -fx-text-fill: white;
15                 -fx-padding: 5;"/>
16         <Label text="파란바탕,흰글씨"
17             style="-fx-background-color: blue; -fx-text-fill: white;
18                 -fx-padding: 5;"/>
19     </children>
20 </HBox>
```



>> AppMain.java

```
1  package sec10.exam01_inline;
2
```

```

3  import javafx.application.Application;
4  import javafx.fxml.FXMLLoader;
5  import javafx.scene.Parent;
6  import javafx.scene.Scene;
7  import javafx.stage.Stage;
8
9  public class AppMain extends Application {
10     @Override
11     public void start(Stage primaryStage) throws Exception {
12         Parent root = FXMLLoader.load(getClass().getResource("root.fxml"));
13         Scene scene = new Scene(root);
14
15         primaryStage.setTitle("AppMain");
16         primaryStage.setScene(scene);
17         primaryStage.show();
18     }
19
20     public static void main(String[] args) {
21         launch(args);
22     }
23 }

```

외부 CSS 파일

인라인 스타일은 컨테이너와 컨트롤의 style 속성으로 CSS를 직접 적용하기 때문에 동일한 스타일을 적용하는 컨테이너와 컨트롤이 많아질수록 중복 코드가 늘어나는 단점이 있다. 또한 레이아웃 FXML과 CSS가 뒤섞여 유지보수가 어렵다.

인라인 스타일보다는 외부 CSS 파일 방식을 사용하면 중복 CSS 작성을 줄이고 재사용성을 높이면서 유지보수도 편리해진다. 외부 CSS 파일 방식은 컨테이너와 컨트롤에 사용되는 CSS를 별도 파일에 작성하고, Scene을 생성할 때 추가하는 방식이다.

여러 가지 장점이 많기 때문에 보통 JavaFX 애플리케이션을 개발할 때는 외부 CSS 파일 방식을 사용한다. 외부 CSS 파일을 작성하기 위해서는 몇 가지 선택자 문법을 알아야 한다.

1) 선택자

인라인 스타일은 해당 컨테이너 또는 컨트롤에 직접 스타일을 적용하기 때문에 선택자가 필요 없지만, 외부 CSS 파일은 스타일을 적용할 컨테이너와 컨트롤을 선택해주는 선택자가 필요하다. 다음은 선택자를 작성하는 방법을 보여준다.

```
선택자 {  
  속성:값; 속성:값; ...  
}
```

선택자는 중괄호 {}에 정의된 CSS 속성을 적용할 컨테이너 또는 컨트롤을 선택하는 역할을 하는데, 다음 세 가지가 있다.

선택자	작성 방법
Type 선택자	Type { 속성:값; 속성:값; .. }
id 선택자	#id { 속성:값; 속성:값; ... }
class 선택자	.class { 속성:값; 속성:값; ... }

Type 선택자는 같은 타입의 컨테이너 또는 컨트롤을 모두 선택한다. 예를 들어 모든 Label 컨트롤의 안쪽 여백을 5만큼 주고 싶다면 다음과 같이 정의하면 된다.

```
Label {  
  -fx-padding: 5;  
}
```

```
<Label ...>  
<Label ...>
```

#id 선택자는 동일한 id 속성값을 가지고 있는 컨테이너 또는 컨트롤을 선택한다. 예를 들어 id 속성값이 lblId인 Label의 배경을 검은색으로, 글자를 노란색으로 설정하고 싶다면 다음과 같이 정의하면 된다.

```
#lblId {  
  -fx-background-color: black;  
  -fx-text-fill: yellow;  
}
```

```
<Label id="lblId">
```

.class 선택자는 동일한 styleClass 속성값을 가지고 있는 컨테이너 또는 컨트롤을 선택한다. 예를 들어 styleClass 속성값이 lblClass인 Label의 배경을 파란색으로, 글자를 흰색으로 설정하고 싶다면 다음과 같이 정의하면 된다.

```
#lblId {
    -fx-background-color: black;
    -fx-text-fill: yellow;
}
```

```
<Label styleClass="lblClass">
<Label styleClass="lblClass">
```

FXML 파일에서 id 속성은 유일한 값을 가져야 하지만, styleClass 속성은 중복된 값을 가질 수 있다. 이 말은 id 선택자는 하나의 컨트롤만 선택할 수 있는 반면, class 선택자는 동시에 여러 컨트롤을 선택할 수 있다는 뜻이다.

Type 선택자와 class 선택자는 조합이 가능하다. 예를 들어 Label 컨트롤 중에서 styleClass="className"을 가진 것만 CSS를 적용하고 싶다면 다음과 같이 정의하면 된다. 만약 Button 컨트롤이 styleClass="className"을 가지고 있다면 CSS는 적용되지 않는다.

```
Label.className {
    -fx-background-color: blue;
    -fx-text-fill: white;
}
```

컨트롤은 세 가지 상태를 가질 수 있다. 입력 가능한 상태(focused), 마우스가 위에 있는 상태(hover), 마우스로 클릭한 상태(pressed)를 말하는데, 각 상태에 따라 스타일을 다르게 적용하고 싶다면 선택자 다음에 :focused, :hover, :pressed를 붙이면 된다. 이러한 것들을 유사 클래스(pseudo-class)라고 한다.

상태	상태별 선택자
입력 가능한 상태	선택자:focus { 속성:값 속성:값 ... }
마우스가 컨트롤 위에 있는 상태	선택자:hover { 속성:값 속성:값 ... }
마우스로 컨트롤을 누른 상태	선택자:pressed { 속성:값 속성:값 ... }

2) CSS 파일 적용

이렇게 작성된 외부 CSS 파일은 개별 컨테이너 또는 컨트롤에 적용하거나 Scene에 추가하여 Scene 내부의 모든 컨테이너와 컨트롤에 적용할 수 있다.

개별 컨테이너 또는 컨트롤에 CSS 파일을 적용하려면 다음과 같이 FXML 파일에서 해당 태그의 `stylesheets` 속성으로 CSS 파일 경로를 지정하면 된다.

```
<컨테이너 stylesheets= "@app.css">
```

Scene 내부의 모든 컨테이너와 컨트롤에 적용하려면 Scene의 `getStylesheets()` 메소드로 `ObservableList`를 얻고, 여기에 CSS 파일 경로를 추가한다. CSS 파일은 메인 클래스와 동일한 경로에서 흔히 작성되므로 다음과 같이 CSS 파일 경로를 얻어 추가하면 된다.

```
public class AppMain extends Application {
    @Override
    public void start(Stage primaryStage) throws Exception {
        Parent root = FXMLLoader.load(getClass().getResource("root.fxml"));
        Scene scene = new Scene(root);
        scene.getStylesheets().add(getClass().getResource("app.css").toString());
        primaryStage.setScene(scene);
        primaryStage.show();
    }
    ...
}
```

다음 예제는 외부 CSS 파일을 이용해서 Label들의 배경색과 글자색을 변경한다. `root.xml`에서 첫 번째 Label에는 `id` 속성이 있고, 두 번째와 세 번째 Label에는 `styleClass` 속성이 있다는 것을 주목하자. 이들 속성을 이용해서 선택자를 작성한다.

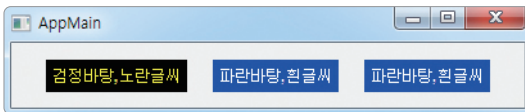
```
>>> root.fxml

1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <?import javafx.scene.layout.*?>
```

```

4  <?import javafx.scene.control.*?>
5  <?import java.lang.*?>
6
7  <HBox xmlns:fx="http://javafx.com/fxml"
8      prefHeight="50" prefWidth="400" alignment="CENTER" spacing="20">
9      <children>
10         <Label id="lblId" text="검정바탕,노란글씨"/>
11         <Label styleClass="lblClass" text="파란바탕,흰글씨"/>
12         <Label styleClass="lblClass" text="파란바탕,흰글씨"/>
13     </children>
14 </HBox>

```



>>> app.css

```

1  /* 전체 Label 선택 */
2  Label {
3      -fx-padding: 5;
4  }
5
6  /* id="lblId"을 가진 컨트롤 선택 */
7  #lblId {
8      -fx-background-color: black;
9      -fx-text-fill: yellow;
10 }
11
12 /* styleClass="lblClass"을 가진 컨트롤 선택 */
13 .lblClass {
14     -fx-background-color: blue;
15     -fx-text-fill: white;
16 }

```

CSS의 주석은 /*로 시작해서 */로 끝난다.
자바처럼 // 주석은 사용할 수 없다.

>>> AppMain.java

```
1  package sec10.exam02_css_file;
2
3  import javafx.application.Application;
4  import javafx.fxml.FXMLLoader;
5  import javafx.scene.Parent;
6  import javafx.scene.Scene;
7  import javafx.stage.Stage;
8
9  public class AppMain extends Application {
10     @Override
11     public void start(Stage primaryStage) throws Exception {
12         Parent root = FXMLLoader.load(getClass().getResource("root.fxml"));
13         Scene scene = new Scene(root);
14         scene.getStylesheets().add(getClass().getResource("app.css").toString());
15
16         primaryStage.setTitle("AppMain");
17         primaryStage.setScene(scene);
18         primaryStage.show();
19     }
20
21     public static void main(String[] args) {
22         launch(args);
23     }
24 }
```

다음 예제는 TextField와 Button 컨트롤의 상태에 따라서 스타일을 변경한다. TextField가 입력 가능한 상태가 되면 배경을 노란색으로, 마우스가 Button 위에 있으면 배경을 노란색 그리고 클릭 하면 빨간색으로 변경한다.

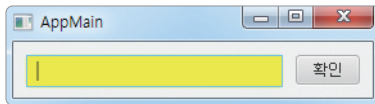
>>> root.fxml

```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <?import javafx.scene.layout.*?>
4  <?import javafx.geometry.*?>
```

```

5  <?import javafx.scene.control.*?>
6
7  <HBox xmlns:fx="http://javafx.com/fxml" spacing="10">
8      <padding>
9          <Insets topRightBottomLeft="10"/>
10     </padding>
11
12     <children>
13         <TextField prefWidth="200"/>
14         <Button prefWidth="50" text="확인"/>
15     </children>
16 </HBox>

```



>>> app.css

```

1  TextField:focused {
2      -fx-border-color: skyblue;
3      -fx-background-color: yellow;
4  }
5
6  Button:hover {
7      -fx-border-color: skyblue;
8      -fx-background-color: yellow;
9  }
10
11 Button:pressed {
12     -fx-border-color: skyblue;
13     -fx-background-color: red;
14 }

```

>>> AppMain.java

```
1  package sec10.exam03_state_selector;
2
3  import javafx.application.Application;
4  import javafx.fxml.FXMLLoader;
5  import javafx.scene.Parent;
6  import javafx.scene.Scene;
7  import javafx.stage.Stage;
8
9  public class AppMain extends Application {
10     @Override
11     public void start(Stage primaryStage) throws Exception {
12         Parent root = FXMLLoader.load(getClass().getResource("root.fxml"));
13         Scene scene = new Scene(root);
14         scene.getStylesheets().add(getClass().getResource("app.css").toString());
15
16         primaryStage.setTitle("AppMain");
17         primaryStage.setScene(scene);
18         primaryStage.show();
19     }
20
21     public static void main(String[] args) {
22         launch(args);
23     }
24 }
```

border 속성

border 속성은 컨테이너 및 컨트롤의 경계선 스타일을 설정한다. 다음과 같이 경계선의 굵기, 색상, 스타일, 내부 경계선의 위치를 설정할 수 있는 세부 속성들이 있다.

속성	설명
-fx-border-color	경계선의 색상
-fx-border-insets	내부 경계선의 위치
-fx-border-radius	둥근 모서리를 위한 원의 반지름

-fx-border-style	경계선의 스타일(실선, 점선)
-fx-border-width	경계선의 굵기

-fx-border-color 속성은 경계선의 색상을 설정하는데, 다음과 같은 다양한 값들을 사용할 수 있다.

```
-fx-border-color: red;           //색이름
-fx-border-color: #ff0000;      //#색상번호
-fx-border-color: rgba(255,0,0,0); //rgba(red값, green값, blue값, 투명도)
```

rgba() 함수의 red, green, blue는 0~255 값을 가질 수 있고, 투명도는 0.0(투명)~1.0(불투명) 값을 가진다. 다음은 경계선을 빨간색으로, 굵기를 1픽셀로 설정한 것이다.

```
-fx-border-color: red;
-fx-border-width: 1;
```

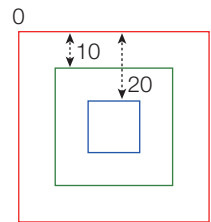


-fx-border-radius 속성은 원의 반지름을 설정해서 둥근 모서리를 만든다.

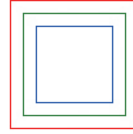
```
-fx-border-color: red;
-fx-border-width: 1;
-fx-border-radius: 20;
```



JavaFX 컨테이너 및 컨트롤은 바깥 경계선 외에 내부 경계선을 여러 개 정의할 수 있다. -fxborder-insets 속성으로 경계선이 나타날 깊이를 쉽표로 구분해서 나열하면 다른 속성들도 경계선의 개수에 맞게 굵기, 색상, 스타일을 지정할 수 있다. 다음은 3개의 경계선을 정의하고 각각 색상 및 굵기를 설정한 것이다.

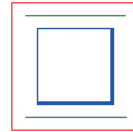


```
-fx-border-insets: 0, 10, 20;  
-fx-border-color: red, green, blue;  
-fx-border-width: 1, 1, 1;
```



경계선은 top, right, bottom, left 별로 굵기, 색상, 스타일을 지정할 수 있다. 값의 순서는 top부터 시작해 시계 방향으로 top, right, bottom, left로 나열해주면 된다. 다음은 두 번째 경계선 top, bottom의 색상을 녹색으로 설정하고, 세 번째 경계선 right, bottom의 굵기를 3으로 설정한 것이다.

```
-fx-border-insets: 0, 10, 20;  
-fx-border-color: red, green white green white, blue;  
-fx-border-width: 1, 1, 1 3 3 1;
```



-fx-border-style 속성은 실선과 점선을 설정하는데, solid(실선), dotted(점선), dashed(대시선)와 선의 길이 및 공백을 설정할 수 있는 segments()를 값으로 줄 수 있다. segments()의 매개값은 홀수 번째 값은 선의 길이를, 짝수 번째 값은 공백의 길이를 주면 된다. 다음은 바깥 경계선의 top, right, bottom, left를 다른 스타일로 설정한 것이다.

```
-fx-border-color: red;  
-fx-border-width: 2;  
-fx-border-style: solid dotted dashed segments(3, 2, 8, 2);
```



다음 예제는 FXML로 네 개의 VBox를 배치하고 border 속성을 다르게 적용하였다.

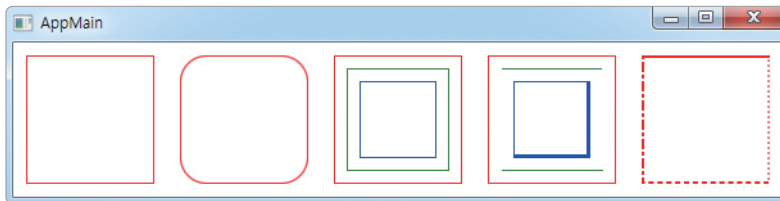
```
>>> root.fxml
```

```
1  <?xml version="1.0" encoding="UTF-8"?>  
2  
3  <?import java.lang.*?>  
4  <?import javafx.geometry.*?>  
5  <?import javafx.scene.layout.*?>  
6  <?import javafx.scene.control.*?>
```

```

7
8 <HBox xmlns:fx="http://javafx.com/fxml" spacing="20">
9   <padding>
10     <Insets bottom="10,0" left="10,0" right="10,0" top="10,0" />
11   </padding>
12   <children>
13     <VBox id="vbox1" prefHeight="100" prefWidth="100"/>
14     <VBox id="vbox2" prefHeight="100" prefWidth="100"/>
15     <VBox id="vbox3" prefHeight="100" prefWidth="100"/>
16     <VBox id="vbox4" prefHeight="100" prefWidth="100"/>
17     <VBox id="vbox5" prefHeight="100" prefWidth="100"/>
18   </children>
19 </HBox>

```



>>> app.css

```

1  #vbox1 {
2    -fx-border-color: red;
3    -fx-border-width: 1;
4  }
5
6  #vbox2 {
7    -fx-border-color: red;
8    -fx-border-width: 1;
9    -fx-border-radius: 20;
10 }
11
12 #vbox3 {
13   -fx-border-insets: 0, 10, 20;

```

```

14     -fx-border-color: red, green, blue;
15     -fx-border-width: 1, 1, 1;
16 }
17
18 #vbox4 {
19     -fx-border-insets: 0, 10, 20;
20     -fx-border-color: red, green white green white, blue;
21     -fx-border-width: 1, 1, 1 3 3 1;
22 }
23
24 #vbox5 {
25     -fx-border-color: red;
26     -fx-border-width: 2;
27     -fx-border-style: solid dotted dashed segments(3, 2, 8, 2);
28 }

```

>>> AppMain.java

```

1  package sec10.exam04_border;
2
3  import javafx.application.Application;
4  import javafx.fxml.FXMLLoader;
5  import javafx.scene.Parent;
6  import javafx.scene.Scene;
7  import javafx.stage.Stage;
8
9  public class AppMain extends Application {
10     @Override
11     public void start(Stage primaryStage) throws Exception {
12         Parent root = FXMLLoader.load(getClass().getResource("root.fxml"));
13         Scene scene = new Scene(root);
14         scene.getStylesheets().add(getClass().getResource("app.css").toString());
15
16         primaryStage.setTitle("AppMain");
17         primaryStage.setScene(scene);
18         primaryStage.show();
19     }
20

```

```

21     public static void main(String[] args) {
22         launch(args);
23     }
24 }

```

background 속성

background 속성은 컨테이너 및 컨트롤의 배경 스타일을 설정한다. 다음과 같이 배경 색상, 배경 이미지를 설정할 수 있는 세부 속성들이 있다.

속성	설명
-fx-background-color	배경 색상
-fx-background-image	배경 이미지
-fx-background-position	배경 이미지 위치(top, right, bottom, left, center)
-fx-background-repeat	이미지 반복 여부(no-repeat: 반복하지 않음)

-fx-background-color 속성은 배경 색상을 설정하는데, 단일 색상을 지정하는 방법은 다음과 같다.

```

-fx-background-color: red;           //색 이름
-fx-background-color: #ff0000;      //#색상 번호
-fx-background-color: rgba(255,0,0,0); //rgba(red값, green값, blue값, 투명도)

```

-fx-background-color에는 단일 색상뿐만 아니라 선형 및 원형 그라디언트도 설정할 수 있다. 선형 그라디언트는 시작 색에서부터 끝 색까지 진행 방향으로 서서히 색상 변화를 준다. 선형 그라디언트를 작성하는 방법은 다음과 같다.

```

linear-gradient(to 진행방향, 시작색 S%, 중간색 M%..., 끝색);

```


진행 방향은 to bottom, to right, to bottom right 등과 같이 밑으로, 오른쪽으로, 대각선으로 지정할 수 있다. 각각의 색은 몇 % 정도가 나올지 S, M 값으로 지정할 수 있다. 단, 끝 색은 % 값을 줄 수 없다. % 값이 생략되면 균등하게 색상이 나온다. 오른쪽 그림은 다음과 같이 작성된 선형 그라디언트가 적용된 것이다.

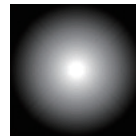


```
-fx-background-color: linear-gradient(to right, black, white);
```

원형 그라디언트는 시작 색에서부터 끝 색까지 원형으로 서서히 색상 변화를 준다. 원형 그라디언트를 작성하는 방법은 다음과 같다.

```
radial-gradient(center X% Y%, radius R%, 시작색 S%, 중간색 M%, 끝색);
```

X%와 Y%는 컨테이너 및 컨트롤의 좌상단을 0%, 0%로 보고, 원의 중심점이 위치하는 곳을 지정한다. 예를 들어 center 50% 50%는 원의 정중앙을 중심점으로 설정한다. R%는 중심점에서부터 색상 변화가 종료되는 위치이다. 예를 들어 radius 50%는 컨테이너 및 컨트롤 전체 영역에 원형 그라디언트를 적용한다. 각각의 색이 몇 % 정도로 나올지 S, M 값으로 지정할 수 있다. 단, 끝 색은 % 값을 줄 수 없다. % 값이 생략되면 균등하게 색상이 나온다. 오른쪽 그림은 다음과 같이 작성된 원형 그라디언트가 적용된 것이다.



```
-fx-background-color: radial-gradient(center 50% 50%, radius 50%, #ffffff 10%, #000000);
```

-fx-background-image 속성은 배경 이미지를 설정한다. 배경 이미지보다 컨테이너 및 컨트롤의 사이즈가 더 크면 이미지는 반복적으로 드로잉된다. 한 번만 드로잉하려면 -fx-backgroundrepeat 속성값을 no-repeat로 설정하면 된다. 그리고 이미지의 위치는 -fx-backgroundposition으로 설정한다. 다음은 배경 이미지를 설정하는 방법이다.

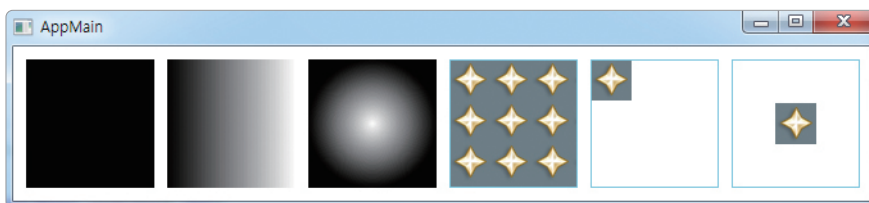


```
-fx-background-image: url("이미지 경로"); //이미지 파일 지정
-fx-background-repeat: no-repeat; //한 번만 드로잉하도록 지정
-fx-background-position: center; //정중앙에 위치하도록 지정
```

다음 예제는 6개의 VBox에 차례대로 단색, 선형 그라디언트, 원형 그라디언트, 반복 이미지, 단일 이미지, 정중앙 단일 이미지를 적용했다.

>> root.fxml

```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <?import javafx.scene.layout.*?>
4  <?import javafx.geometry.*?>
5
6  <HBox xmlns:fx="http://javafx.com/fxml" spacing="10">
7      <padding>
8          <Insets bottom="10.0" left="10.0" right="10.0" top="10.0" />
9      </padding>
10     <children>
11         <VBox id="vbox1" prefHeight="100" prefWidth="100" />
12         <VBox id="vbox2" prefHeight="100" prefWidth="100" />
13         <VBox id="vbox3" prefHeight="100" prefWidth="100" />
14         <VBox id="vbox4" prefHeight="100" prefWidth="100" />
15         <VBox id="vbox5" prefHeight="100" prefWidth="100" />
16         <VBox id="vbox6" prefHeight="100" prefWidth="100" />
17     </children>
18 </HBox>
```



>>> app.css

```
1  #vbox1 {
2    -fx-background-color: rgba(0, 0, 0, 1);
3  }
4
5  #vbox2 {
6    -fx-background-color: linear-gradient(to right, #000000, #ffffff);
7  }
8
9  #vbox3 {
10   -fx-background-color: radial-gradient(center 50% 50%, radius 50%,
11     #ffffff, #000000);
12 }
13
14 #vbox4 {
15   -fx-border-color: skyblue;
16   -fx-background-image: url("images/icon.gif");
17 }
18
19 #vbox5 {
20   -fx-border-color: skyblue;
21   -fx-background-image: url("images/icon.gif");
22   -fx-background-repeat: no-repeat;
23 }
24
25 #vbox6 {
26   -fx-border-color: skyblue;
27   -fx-background-image: url("images/icon.gif");
28   -fx-background-repeat: no-repeat;
29   -fx-background-position: center;
30 }
```

>>> AppMain.java

```
1  package sec10.exam05_background;
2
3  import javafx.application.Application;
4  import javafx.fxml.FXMLLoader;
```

```

5  import javafx.scene.Parent;
6  import javafx.scene.Scene;
7  import javafx.stage.Stage;
8
9  public class AppMain extends Application {
10     @Override
11     public void start(Stage primaryStage) throws Exception {
12         Parent root = FXMLLoader.load(getClass().getResource("root.fxml"));
13         Scene scene = new Scene(root);
14         scene.getStylesheets().add(getClass().getResource("app.css").toString());
15
16         primaryStage.setTitle("AppMain");
17         primaryStage.setScene(scene);
18         primaryStage.show();
19     }
20
21     public static void main(String[] args) {
22         launch(args);
23     }
24 }

```

font 속성

font 속성은 글자의 스타일을 설정한다. 여기에는 폰트의 크기, 종류, 굵기, 색상 등을 설정할 수 있는 다음과 같은 세부 속성들이 있다.

속성	설명
-fx-font-size	폰트 크기
-fx-font-family	폰트 종류
-fx-font-weight	폰트 굵기(bold)
-fx-text-fill	폰트 색상(단색, 선형 그라디언트, 원형 그라디언트)

다음 예제는 Label 컨트롤의 폰트를 설정한다. 폰트 종류는 Arial Black, 크기는 35, 굵게, 색상은 선형 그라디언트를 적용했다.

>>> root.fxml

```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <?import javafx.scene.layout.*?>
4  <?import javafx.geometry.*?>
5  <?import javafx.scene.control.*?>
6
7  <VBox id="root" xmlns:fx="http://javafx.com/fxml" >
8      <children>
9          <Label id="welcome-text" text="Welcome" />
10     </children>
11 </VBox>
```



>>> app.css

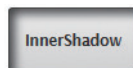
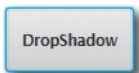
```
1  #root {
2      -fx-padding: 10;
3  }
4
5  #welcome-text {
6      -fx-font-size: 35;
7      -fx-font-family: "Arial Black";
8      -fx-font-weight: bold;
9      -fx-text-fill: linear-gradient(to bottom, blue, white);
10 }
```

>>> AppMain.java

```
1  package sec10.exam06_font;
2
3  import javafx.application.Application;
4  import javafx.fxml.FXMLLoader;
5  import javafx.scene.Parent;
6  import javafx.scene.Scene;
7  import javafx.stage.Stage;
8
9  public class AppMain extends Application {
10     @Override
11     public void start(Stage primaryStage) throws Exception {
12         Parent root = FXMLLoader.load(getClass().getResource("root.fxml"));
13         Scene scene = new Scene(root);
14         scene.getStylesheets().add(getClass().getResource("app.css").toString());
15
16         primaryStage.setTitle("AppMain");
17         primaryStage.setScene(scene);
18         primaryStage.show();
19     }
20
21     public static void main(String[] args) {
22         launch(args);
23     }
24 }
```

shadow 효과

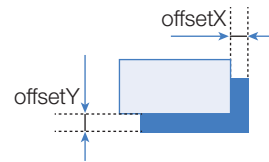
JavaFX CSS는 그림자 효과를 주기 위한 `-fx-effect` 속성을 제공한다. 속성값으로는 `dropshadow()`와 `innershadow()`를 줄 수 있는데, `dropshadow()`는 바깥 그림자를 주어 튀어 나오는 느낌을 주고, `innershadow()`는 안쪽 그림자를 주어 움푹 들어간 느낌을 준다.



-fx-effect 속성을 작성하는 방법은 다음과 같다.

```
-fx-effect: dropshadow(three-pass-box , 그림자색상 , radius, spread , offsetX , offsetY);  
-fx-effect: innershadow(three-pass-box , 그림자색상 , radius, choke, offsetX, offsetY);
```

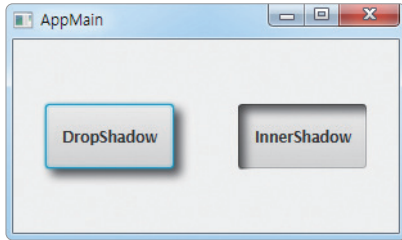
three-pass-box는 blur-type의 종류로 gaussian, one-pass-box, two-pass-box, three-pass-box 등이 있다. JavaFX는 three-pass-box를 기본으로 사용한다. radius는 blurkernel의 반지름으로 0.0~127.0 사이의 값을 가지는데, 기본값은 10이다. spread와 choke는 각각 그림자의 spread와 choke로 0.0~1.0 사이의 값을 가지는데, 기본값은 0.0이다. three-pass-box, radius, spread, choke는 이해하기가 쉽지 않기 때문에 보통은 기본값을 그대로 이용한다. offsetX와 offsetY는 그림자의 편차인데, 오른쪽 그림을 보면 쉽게 이해가 될 것이다.



다음 예제는 두 개의 버튼에 shadow 효과를 주었다. 첫 번째 버튼은 dropshadow()를 적용했고 두 번째 버튼은 innershadow()를 적용했다.

>> root.fxml

```
1  <?xml version="1.0" encoding="UTF-8"?>  
2  
3  <?import javafx.scene.layout.*?>  
4  <?import javafx.geometry.*?>  
5  <?import javafx.scene.control.*?>  
6  
7  <HBox xmlns:fx="http://javafx.com/fxml" prefWidth="300" prefHeight="150"  
8    spacing="50" fillHeight="false" alignment="CENTER">  
9    <padding>  
10     <Insets topRightBottomLeft="10"/>  
11    </padding>  
12    <children>  
13     <Button id="btn1" prefWidth="100" prefHeight="50" text="DropShadow"/>  
14     <Button id="btn2" prefWidth="100" prefHeight="50" text="InnerShadow"/>  
15    </children>  
16  </HBox>
```



>>> app.css

```
1  #btn1 {
2      -fx-effect: dropshadow(three-pass-box , rgba(0,0,0,0.7) , 10, 0 , 5 , 5);
3  }
4
5  #btn2 {
6      -fx-effect: innershadow(three-pass-box , rgba(0,0,0,0.7) , 10, 0 , 3, 3);
7  }
```

>>> AppMain.java

```
1  package sec10.exam07_shadow;
2
3  import javafx.application.Application;
4  import javafx.fxml.FXMLLoader;
5  import javafx.scene.Parent;
6  import javafx.scene.Scene;
7  import javafx.stage.Stage;
8
9  public class AppMain extends Application {
10     @Override
11     public void start(Stage primaryStage) throws Exception {
12         Parent root = FXMLLoader.load(getClass().getResource("root.fxml"));
13         Scene scene = new Scene(root);
14         scene.getStylesheets().add(getClass().getResource("app.css").toString());
15
16         primaryStage.setTitle("AppMain");
```



```

17     primaryStage.setScene(scene);
18     primaryStage.show();
19 }
20
21 public static void main(String[] args) {
22     launch(args);
23 }
24 }

```

화면 스킨 입히기

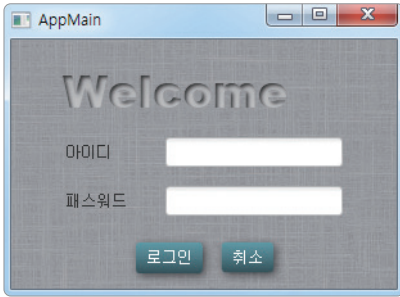
다음 예제에서는 지금까지 학습한 내용을 활용해서 로그인 폼 화면을 만들고 CSS를 적용해서 스킨을 입혀본다. 이 예제는 오라클의 JavaFX CSS 샘플 예제로 소개된 것이다. 마우스를 버튼 위로 올리거나 클릭하면 색상 변화를 볼 수 있다

>>> root.fxml

```

1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <?import javafx.scene.layout.*?>
4  <?import javafx.scene.control.*?>
5
6  <AnchorPane styleClass="root" xmlns:fx="http://javafx.com/fxml"
7      prefHeight="194.0" prefWidth="300.0" >
8      <children>
9          <Label id="welcome-text" layoutX="40.0" layoutY="14.0" text="Welcome" />
10         <Label layoutX="42.0" layoutY="80.0" text="아이디" />
11         <Label layoutX="42.0" layoutY="118.0" text="패스워드" />
12         <TextField layoutX="120.0" layoutY="76.0" />
13         <PasswordField layoutX="120.0" layoutY="114.0" />
14         <Button layoutX="97.0" layoutY="158.0" styleClass="button" text=
            "로그인" />
15         <Button layoutX="164.0" layoutY="158.0" styleClass="button" text=
            "취소" />
16     </children>
17 </AnchorPane>

```



>> app.css

```
1  .root {
2      -fx-background-image: url("images/background.jpg");
3  }
4
5  Label {
6      -fx-font-size: 12px;
7      -fx-font-weight: bold;
8      -fx-text-fill: #333333;
9  }
10
11 #welcome-text {
12     -fx-font-size: 35px;
13     -fx-font-family: "Arial Black";
14     -fx-text-fill: linear-gradient(darkgray, lightgray);
15     -fx-effect: innershadow( three-pass-box , rgba(0,0,0,0.7) , 3, 0 , 2 , 2.1 );
16 }
17
18 .button {
19     -fx-text-fill: white;
20     -fx-font-family: "Arial Narrow";
21     -fx-font-weight: bold;
22     -fx-background-color: linear-gradient(#61a2b1, #2A5058);
23     -fx-effect: dropshadow( three-pass-box , rgba(0,0,0,0.6) , 10, 0 , 2 , 2 );
24 }
25
26 .button:hover {
```

```

27     -fx-background-color: linear-gradient(#2A5058, #61a2b1);
28 }
29
30 .button:pressed {
31     -fx-background-color: linear-gradient(yellow, #61a2b1);
32 }

```

>>> AppMain.java

```

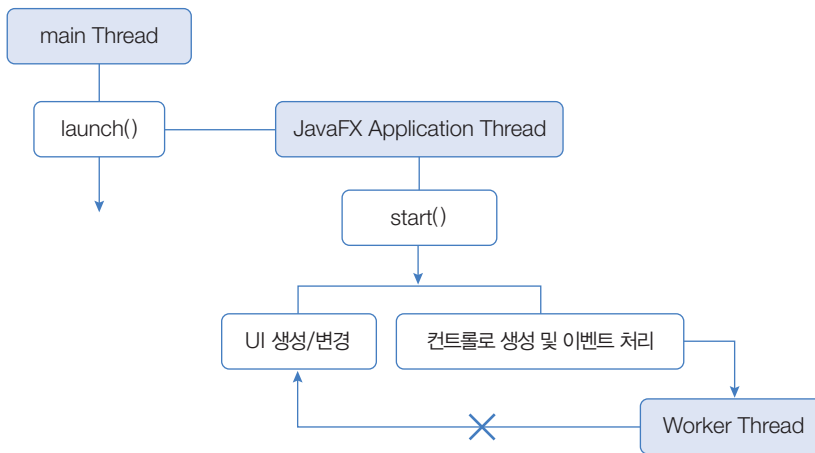
1  package sec10.exam08_login_skin;
2
3  import javafx.application.Application;
4  import javafx.fxml.FXMLLoader;
5  import javafx.scene.Parent;
6  import javafx.scene.Scene;
7  import javafx.stage.Stage;
8
9  public class AppMain extends Application {
10     @Override
11     public void start(Stage primaryStage) throws Exception {
12         Parent root = FXMLLoader.load(getClass().getResource("root.fxml"));
13         Scene scene = new Scene(root);
14         scene.getStylesheets().add(getClass().getResource("app.css").toString());
15
16         primaryStage.setTitle("AppMain");
17         primaryStage.setScene(scene);
18         primaryStage.show();
19     }
20
21     public static void main(String[] args) {
22         launch(args);
23     }
24 }

```

11 JavaFX 스레드 UI 변경

JavaFX UI는 스레드에 안전하지 않기 때문에 UI를 생성하고 변경하는 작업은 JavaFX Application Thread가 담당하고, 다른 작업 스레드들은 UI를 생성하거나 변경할 수 없다.

main 스레드가 Application의 `launch()` 메소드를 호출하면서 생성된 JavaFX Application Thread는 `start()` 메소드를 실행시키면서 모든 UI를 생성한다. 컨트롤에서 이벤트가 발생할 경우 컨트롤러의 이벤트 처리 메소드를 실행하는 것도 JavaFX Application Thread이다.



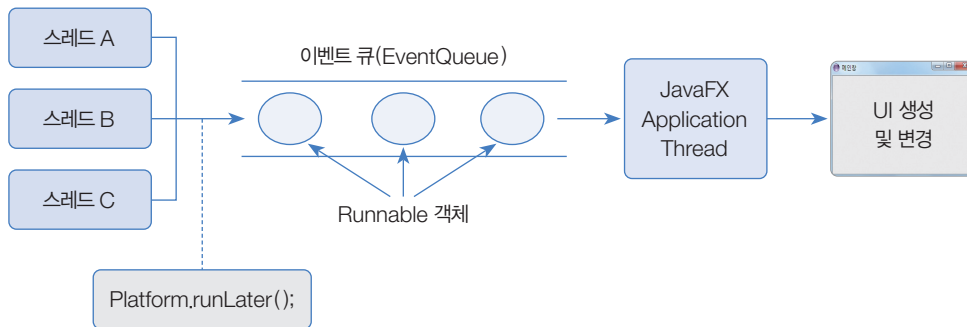
JavaFX 애플리케이션을 개발할 때 주의할 점은 JavaFX Application Thread가 시간을 요하는 작업을 하지 않도록 하는 것이다. 시간을 요하는 작업을 하게 되면 이 시간 동안에 UI는 반응하지 않고 멈춰있는 상태가 되기 때문에 다른 작업 스레드를 생성해서 처리하는 것이 좋다.

예를 들어 파일을 읽고 쓰거나 네트워크상에서 데이터를 주고받을 때 얼마만큼의 시간이 필요한지 모르기 때문에 반드시 작업 스레드를 생성해서 처리해야 한다.

만약 작업 스레드에서 UI 변경 작업이 필요하다면 작업 스레드가 직접 UI를 변경할 수 없기 때문에 UI 변경 코드를 Runnable로 생성하고, 이것을 매개값으로 해서 Platform의 정적 메소드인 `runLater()`를 호출해야 한다.

Runnable 익명 구현 객체 이용	람다식 이용
<pre>Platform.runLater(new Runnable() { @Override public void run() { //UI 생성 및 변경 코드 } });</pre>	<pre>Platform.runLater()->{ //UI 생성 및 변경 코드 };</pre>

Platform.runLater() 메소드는 매개값으로 받은 Runnable을 이벤트 큐(event queue)에 저장하고 즉시 리턴된다.



이벤트 큐에 저장된 Runnable은 JavaFX Application Thread에 의해 순차적으로 하나씩 실행 처리되어 UI를 변경한다. runLater는 지금 당장 실행하는 것이 아니고 순서에 따라 조금 후에 실행한다는 뜻에서 지어진 이름이다.

다음 예제는 작업 스레드가 0.1초 주기로 얻어낸 시간을 Label 컨트롤의 text 속성값으로 사용한다. Label은 UI 요소이므로 작업 스레드에서 setText() 메소드로 text 속성값을 변경할 수 없다. 대신 Runnable을 생성해서 Platform.runLater() 메소드를 호출한다.

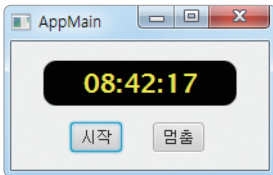
```
>>> root.fxml

1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <?import javafx.scene.layout.*?>
4  <?import javafx.scene.control.*?>
5
```

```

6  <AnchorPane xmlns:fx="http://javafx.com/fxml" prefHeight="100.0"
    prefWidth="200.0"
7    fx:controller="sec11.exam01_runlater.RootController">
8    <children>
9    <Label fx:id="lblTime" alignment="CENTER" layoutX="25.0"
        layoutY="15.0"
10       prefHeight="35.0" prefWidth="150.0"
11       style="-fx-background-color: black; -fx-text-fill: yellow;
12       -fx-font-size: 20; -fx-background-radius: 10;" text="00:00:00" />
13    <Button fx:id="btnStart" layoutX="46.0" layoutY="63.0" text="시작" />
14    <Button fx:id="btnStop" layoutX="110.0" layoutY="63.0" text="멈춤" />
15    </children>
16 </AnchorPane>

```



>> RootController.java

```

1  package sec11.exam01_runlater;
2
3  import java.net.URL;
4  import java.text.SimpleDateFormat;
5  import java.util.Date;
6  import java.util.ResourceBundle;
7  import javafx.application.Platform;
8  import javafx.event.ActionEvent;
9  import javafx.fxml.FXML;
10 import javafx.fxml.Initializable;
11 import javafx.scene.control.Button;
12 import javafx.scene.control.Label;
13
14 public class RootController implements Initializable {

```

```

15     @FXML private Label lblTime;
16     @FXML private Button btnStart;
17     @FXML private Button btnStop;
18
19     private boolean stop;
20
21     @Override
22     public void initialize(URL location, ResourceBundle resources) {
23         btnStart.setOnAction(event->handleBtnStart(event));
24         btnStop.setOnAction(event->handleBtnStop(event));
25     }
26
27     public void handleBtnStart(ActionEvent e) {
28         stop = false;
29         Thread thread = new Thread() {
30             @Override
31             public void run() {
32                 SimpleDateFormat sdf = new SimpleDateFormat("HH:mm:ss");
33                 while(!stop) {
34                     String strTime = sdf.format(new Date());
35                     Platform.runLater(()->{
36                         lblTime.setText(strTime);
37                     });
38                 }
39             }
40         };
41         thread.setDaemon(true);
42         thread.start();
43     }
44
45     public void handleBtnStop(ActionEvent e) {
46         stop = true;
47     }
48 }
49

```

UI 변경 작업

>>> AppMain.java

```
1  package sec11.exam01_runlater;
2
3  import javafx.application.Application;
4  import javafx.fxml.FXMLLoader;
5  import javafx.scene.Parent;
6  import javafx.scene.Scene;
7  import javafx.stage.Stage;
8
9  public class AppMain extends Application {
10     @Override
11     public void start(Stage primaryStage) throws Exception {
12         Parent root = (Parent)FXMLLoader.load(getClass().getResource("root.
            fxml"));
13         Scene scene = new Scene(root);
14
15         primaryStage.setTitle("AppMain");
16         primaryStage.setScene(scene);
17         primaryStage.show();
18     }
19
20     public static void main(String[] args) {
21         launch(args);
22     }
23 }
```

다음 예제는 작업 스레드에서 0부터 100까지 합을 구할 동안 ProgressBar와 Label 컨트롤에 진행 정도를 표시한다. 그리고 작업이 완료되면 결과를 Label 컨트롤에 나타낸다. ProgressBar의 진행 정도와 Label의 글자를 변경하는 작업은 Platform.runlater() 메소드로 처리하는 것을 볼 수 있다.

>>> root.fxml

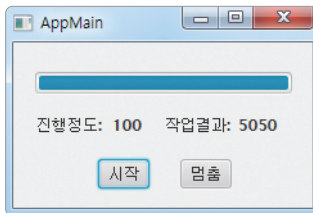
```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <?import java.lang.*?>
4  <?import javafx.scene.layout.*?>
```



```

5  <?import javafx.scene.control.*?>
6
7  <AnchorPane xmlns:fx="http://javafx.com/fxml" prefHeight="129.0"
    prefWidth="233.0"
8      fx:controller="sec11.exam02_runlater.RootController">
9      <children>
10         <ProgressBar fx:id="progressBar" layoutX="17.0" layoutY="23.0"
11             prefWidth="200.0" progress="0.0" />
12         <Label layoutX="18.0" layoutY="57.0" text="진행정도:" />
13         <Label fx:id="lblWorkDone" layoutX="77.0" layoutY="57.0" />
14         <Label layoutX="118.0" layoutY="57.0" text="작업결과:" />
15         <Label fx:id="lblResult" layoutX="175.0" layoutY="57.0" />
16         <Button fx:id="btnStart" layoutX="66.0" layoutY="91.0" text="시작" />
17         <Button fx:id="btnStop" layoutX="130.0" layoutY="91.0" text="멈춤" />
18     </children>
19 </AnchorPane>

```



>> RootController.java

```

1  package sec11.exam02_runlater;
2
3  import java.net.URL;
4  import java.util.ResourceBundle;
5  import javafx.application.Platform;
6  import javafx.event.ActionEvent;
7  import javafx.fxml.FXML;
8  import javafx.fxml.Initializable;
9  import javafx.scene.control.Button;
10 import javafx.scene.control.Label;

```

```

11 import javafx.scene.control.ProgressBar;
12
13 public class RootController implements Initializable {
14     @FXML private ProgressBar progressBar;
15     @FXML private Label lblWorkDone;
16     @FXML private Label lblResult;
17     @FXML private Button btnStart;
18     @FXML private Button btnStop;
19
20     private Thread thread;
21
22     @Override
23     public void initialize(URL location, ResourceBundle resources) {
24         btnStart.setOnAction(event->handleBtnStart(event));
25         btnStop.setOnAction(event->handleBtnStop(event));
26     }
27
28     public void handleBtnStart(ActionEvent e) {
29         thread = new Thread() {
30             @Override
31             public void run() {
32                 int result = 0;
33                 for(int i=0; i<=100; i++) {
34                     result += i;
35
36                     double progress = i/100.0;
37                     String workDone = String.valueOf(i);
38
39                     Platform.runLater(() -> {
40                         progressBar.setProgress(progress);
41                         lblWorkDone.setText(workDone);
42                     });
43
44                     try {Thread.sleep(100); } catch(InterruptedException e) {
45                         break;
46                     }
47                 }
48
49                 String strResult = String.valueOf(result);

```

```

50         Platform.runLater(() -> {
51             lblResult.setText(String.valueOf(strResult));
52         });
53     }
54 };
55 thread.setDaemon(true);
56 thread.start();
57 }
58
59 public void handleBtnStop(ActionEvent e) {
60     thread.interrupt();
61 }
62 }

```

UI 변경 작업

>>> AppMain.java

```

1  package sec11.exam02_runlater;
2
3  import javafx.application.Application;
4  import javafx.fxml.FXMLLoader;
5  import javafx.scene.Parent;
6  import javafx.scene.Scene;
7  import javafx.stage.Stage;
8
9  public class AppMain extends Application {
10     @Override
11     public void start(Stage primaryStage) throws Exception {
12         Parent root = (Parent)FXMLLoader.load(getClass().getResource("root.
            fxml"));
13         Scene scene = new Scene(root);
14
15         primaryStage.setTitle("AppMain");
16         primaryStage.setScene(scene);
17         primaryStage.show();
18     }
19
20     public static void main(String[] args) {
21         launch(args);

```

```
22     }
23 }
```

12 장면 이동과 애니메이션

애플리케이션은 다양한 화면을 가지고 있다. 메인 화면에서 시작해 사용자의 선택에 따라 가입 화면, 로그인 화면, 목록 화면 등으로 이동된다. 화면이 이동될 때 애니메이션을 적용하면 눈을 즐겁게 만들어준다. 같은 화면에서도 마우스의 위치에 따라 컨트롤의 속성을 변화시켜 다이나믹한 애니메이션을 만들 수도 있다.

이번 절에서는 화면을 이동시키는 방법과 슬라이드^{slide}와 페이드^{fade} 애니메이션을 적용하는 방법을 알아보기로 하자.

화면 이동

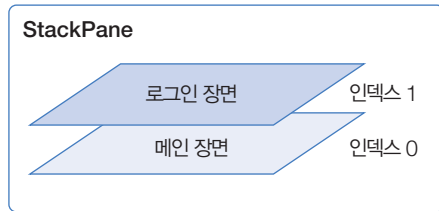
JavaFX에서 화면은 장면이므로 화면을 이동한다는 것은 즉 장면을 변경하는 것이다. 장면을 변경하는 가장 쉬운 방법은 Stage에 새로운 Scene을 세팅하는 것이다.

메인 화면에서 로그인 화면으로 이동하기 위해 버튼을 클릭했다고 가정해보자. 컨트롤러의 이벤트 처리 메소드는 로그인 Scene을 생성하고, primaryStage의 setScene() 메소드로 현재 Scene을 로그인 Scene으로 변경할 수 있다.

```
public void handlebtnLogin(ActionEvent event) {
    try {
        Parent login = FXMLLoader.load(getClass().getResource("login.fxml"));
        Scene scene = new Scene(login);
        Stage primaryStage = (Stage) btnLogin.getScene().getWindow();
        primaryStage.setScene(scene);    //화면 변경
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

setScene() 메소드로 화면을 이동하면 이전 Scene은 Stage에서 제거되므로 애니메이션을 적용할 수 없다. 화면 이동 애니메이션을 적용하려면 이전 화면과 다음 화면이 일시적으로 공존해야 한다.

따라서 화면 이동을 위한 다른 방법이 필요한데, StackPane을 이용하면 화면 이동 효과와 함께 애니메이션을 적용할 수 있다. 다음과 같이 Scene은 하나만 생성하고 StackPane을 루트 컨테이너로 해서 메인 화면과 로그인 화면을 추가하는 것이다.



메인 화면만 StackPane에 추가되어 있는 상태에서 로그인 화면이 생성되고 StackPane에 추가되면 StackPane은 나중에 추가된 로그인 화면만 사용자에게 보여주고 메인 화면은 뒤로 숨긴다. 반대로 StackPane에서 로그인 화면을 제거하면 밑에 있는 메인 화면이 보여진다.

이와 같은 방법으로 화면을 이동하게 되면 이전 화면과 다음 화면이 일시적으로 공존하게 되고 애니메이션을 적용할 수 있게 된다. 다음 예제는 StackPane을 이용해서 로그인 화면과 메인 화면을 번갈아가면서 보여준다.

>>> root.fxml

```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <?import javafx.scene.layout.*?>
4  <?import javafx.scene.control.*?>
5  <?import javafx.geometry.*?>
6
7  <StackPane xmlns:fx="http://javafx.com/fxml"
8      prefHeight="500" prefWidth="350"
9      fx:controller="sec12.exam01_stackpane.RootController">
10     <children>
11         <BorderPane>
12             <top>
13                 <BorderPane style="-fx-background-color: #eaeaea;">
```

```

14         <center>
15             <Label alignment="CENTER" prefWidth="215" text="메인 화면" />
16         </center>
17
18         <right>
19             <Button fx:id="btnLogin" text="로그인"/>
20         </right>
21         <padding>
22             <Insets bottom="10.0" left="10.0" right="10.0" top="10.0" />
23         </padding>
24     </BorderPane>
25 </top>
26
27     <center>
28         <VBox style="-fx-background-color: #0000ff;">
29             <VBox>
30             </center>
31         </BorderPane>
32     </children>
33 </StackPane>

```



>>> RootController.java

```
1  package sec12.exam01_stackpane;
2
3  import java.net.URL;
4  import java.util.ResourceBundle;
5  import javafx.event.ActionEvent;
6  import javafx.fxml.FXML;
7  import javafx.fxml.FXMLLoader;
8  import javafx.fxml.Initializable;
9  import javafx.scene.Parent;
10 import javafx.scene.control.Button;
11 import javafx.scene.layout.StackPane;
12
13 public class RootController implements Initializable {
14     @FXML private Button btnLogin;
15
16     @Override
17     public void initialize(URL location, ResourceBundle resources) {
18         btnLogin.setOnAction(e->handleBtnLogin(e));
19     }
20
21     public void handleBtnLogin(ActionEvent event) {
22         try {
23             Parent login= FXMLLoader.load(getClass().getResource("login.fxml"));
24             StackPane root = (StackPane) btnLogin.getScene().getRoot();
25             root.getChildren().add(login);
26         } catch (Exception e) {
27             e.printStackTrace();
28         }
29     }
30 }
```



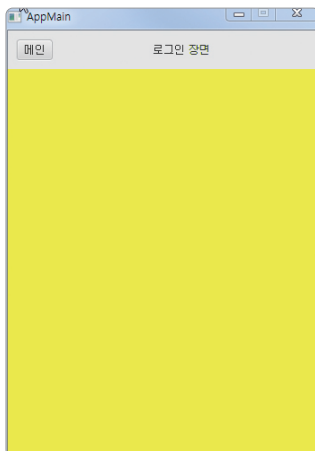
>>> login.xml

```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <import javafx.scene.layout.*?>
4  <import javafx.scene.control.*?>
```

```

5  <?import javafx.geometry.*?>
6
7  <BorderPane xmlns:fx="http://javafx.com/fxml"
8      fx:id="login" prefHeight="500" prefWidth="350"
9      fx:controller="sec12.exam01_stackpane.LoginController">
10     <top>
11         <BorderPane style="-fx-background-color: #eaeaea;">
12             <left>
13                 <Button fx:id="btnMain" text="메인"/>
14             </left>
15             <center>
16                 <Label alignment="CENTER" prefWidth="215" text="로그인 화면" />
17             </center>
18             <padding>
19                 <Insets bottom="10.0" left="10.0" right="10.0" top="10.0" />
20             </padding>
21         </BorderPane>
22     </top>
23
24     <center>
25         <VBox style="-fx-background-color: #ffff00;">
26             </VBox>
27         </center>
28 </BorderPane>

```



>>> LoginController.java

```
1  package sec12.exam01_stackpane;
2
3  import java.net.URL;
4  import java.util.ResourceBundle;
5  import javafx.event.ActionEvent;
6  import javafx.fxml.FXML;
7  import javafx.fxml.Initializable;
8  import javafx.scene.control.Button;
9  import javafx.scene.layout.BorderPane;
10 import javafx.scene.layout.StackPane;
11
12 public class LoginController implements Initializable {
13     @FXML private BorderPane login;
14     @FXML private Button btnMain;
15
16     @Override
17     public void initialize(URL location, ResourceBundle resources) {
18         btnMain.setOnAction(e -> handleBtnMain(e));
19     }
20
21     public void handleBtnMain(ActionEvent event) {
22         try {
23             StackPane root = (StackPane) btnMain.getScene().getRoot();
24             root.getChildren().remove(login);
25         } catch (Exception e) {
26             e.printStackTrace();
27         }
28     }
29 }
```

StackPane에서 로그인 화면 제거, 메인 화면으로 이동

>>> AppMain.java

```
1  package sec12.exam01_stackpane;
2
3  import javafx.application.Application;
4  import javafx.fxml.FXMLLoader;
5  import javafx.scene.Parent;
```

```

6  import javafx.scene.Scene;
7  import javafx.stage.Stage;
8
9  public class AppMain extends Application {
10     @Override
11     public void start(Stage primaryStage) throws Exception {
12         primaryStage.setTitle("AppMain");
13         Parent root= FXMLLoader.load(getClass().getResource("root.fxml"));
14         Scene scene = new Scene(root);
15         primaryStage.setScene(scene);
16         primaryStage.setWidth(350);           //윈도우의 고정 폭 설정
17         primaryStage.setHeight(500);          //윈도우의 고정 높이 설정
18         primaryStage.setResizable(false);     //윈도우 크기를 조정할 수 없도록 함
19         primaryStage.show();
20     }
21
22     public static void main(String[] args) {
23         launch(args);
24     }
25 }

```

애니메이션

JavaFX에서 애니메이션은 컨트롤 또는 컨테이너의 속성^{Property} 변화를 주어진 시간 동안 진행함으로써 구현한다. 다음은 애니메이션과 관련된 클래스를 설명한 표이다.

클래스	설명
KeyValue	값이 변경될 타겟 Property와 종료값을 설정하는 객체
KeyFrame	애니메이션의 지속 시간과 KeyValue를 설정하는 객체 (지속시간 동안 타겟 Property의 값을 종료값까지 변화시킴)
Timeline	KeyFrame에 설정된 내용대로 애니메이션을 플레이하는 객체

다음은 컨테이너의 `translateX` 속성을 주어진 종료값까지 변화시켜 수평 방향으로 슬라이드하는 애니메이션을 구현한 코드이다. 종료값은 x 좌표값으로 음수, 0, 양수 값을 모두 가질 수 있다.

```

① Timeline timeline = new Timeline();
② KeyValue keyValue = new KeyValue(컨테이너.translateXProperty(), 종료값);
③ KeyFrame keyFrame = new KeyFrame(Duration.millis(지속시간), keyValue);
④ timeline.getKeyFrames().add(keyFrame);
⑤ timeline.play();    //애니메이션 시작

```

다음 코드는 이전 예제의 [로그인] 버튼 이벤트 처리 메소드를 수정한 것으로, 로그인 화면이 나올 때 우측에서 좌측으로 수평 애니메이션이 진행되도록 했다.

»» RootController.java

```

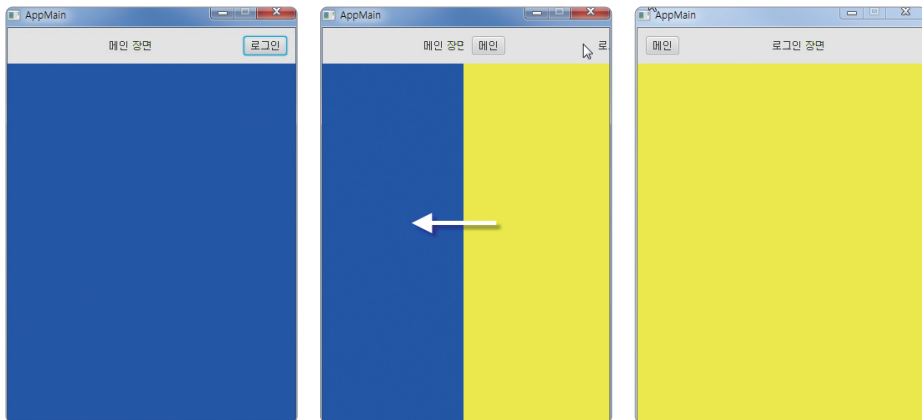
1  package sec12.exam02_move_animation;
2
3  import java.net.URL;
4  import java.util.ResourceBundle;
5  import javafx.animation.KeyFrame;
6  import javafx.animation.KeyValue;
7  import javafx.animation.Timeline;
8  import javafx.event.ActionEvent;
9  import javafx.fxml.FXML;
10 import javafx.fxml.FXMLLoader;
11 import javafx.fxml.Initializable;
12 import javafx.scene.Parent;
13 import javafx.scene.control.Button;
14 import javafx.scene.layout.StackPane;
15 import javafx.util.Duration;
16
17 public class RootController implements Initializable {
18     @FXML private Button btnLogin;
19
20     @Override
21     public void initialize(URL location, ResourceBundle resources) {
22         btnLogin.setOnAction(e->handleBtnLogin(e));
23     }
24
25     public void handleBtnLogin(ActionEvent event) {
26         try {
27             Parent login= FXMLLoader.load(getClass().getResource("login.fxml"));

```

```

28     StackPane root = (StackPane) btnLogin.getScene().getRoot();
29     root.getChildren().add(login);
30
31     //로그인 화면의 x 좌표를 350(윈도우 폭) 만큼 이동(안보이게 하기 위해서)
32     login.setTranslateX(350);
33
34     //Timeline 객체 생성
35     Timeline timeline = new Timeline();
36     //로그인 화면의 x 좌표 종료값으로 0을 설정
37     KeyValue keyValue = new KeyValue(login.translateXProperty(), 0);
38     //0.1초간 종료값까지 x 좌표를 변화시키도록(0 <-- 350) KeyFrame 생성
39     KeyFrame keyFrame = new KeyFrame(Duration.millis(100), keyValue);
40     //Timeline에 KeyFrame 추가
41     timeline.getKeyFrames().add(keyFrame);
42     //애니메이션 시작
43     timeline.play();
44 } catch (Exception e) {
45     e.printStackTrace();
46 }
47 }
48 }

```



KeyFrame은 지속시간 동안 속성을 종료값까지 변화시키고 나서 콜백 메소드를 자동 호출하는 기능이 있다. KeyFrame을 생성할 때 두 번째 매개값으로 EventHandler<ActionEvent> 객체를 제공해주면 애니메이션이 종료될 때 handle() 메소드가 자동 실행된다.

```
KeyFrame keyFrame = new KeyFrame(
    Duration.millis(지속시간),
    new EventHandler<ActionEvent>() {
        @Override
        public void handle(ActionEvent event) {
            //애니메이션이 종료된 후 실행할 코드
        }
    },
    keyValue
);
```

애니메이션이 종료되면 발생하는 이벤트를 처리하는 핸들러 지정

다음 코드는 이전 예제에서 로그인 화면의 [메인] 버튼 이벤트 처리 메소드를 수정한 것으로, 로그인 화면이 사라질 때 좌측에서 우측으로 수평 애니메이션이 진행되도록 했다. 그리고 애니메이션을 종료한 후에는 StackPane에서 로그인 화면을 완전히 제거한다.

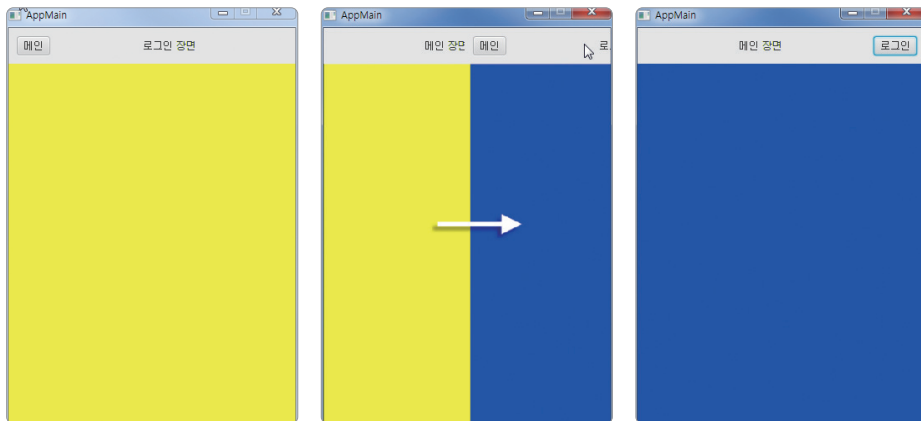
>> LoginController.java

```
1  package sec12.exam02_move_animation;
2
3  import java.net.URL;
4  import java.util.ResourceBundle;
5  import javafx.animation.KeyFrame;
6  import javafx.animation.KeyValue;
7  import javafx.animation.Timeline;
8  import javafx.event.ActionEvent;
9  import javafx.event.EventHandler;
10 import javafx.fxml.FXML;
11 import javafx.fxml.Initializable;
12 import javafx.scene.control.Button;
13 import javafx.scene.layout.BorderPane;
14 import javafx.scene.layout.StackPane;
15 import javafx.util.Duration;
16
```

```

17 public class LoginController implements Initializable {
18     @FXML private BorderPane login;
19     @FXML private Button btnMain;
20
21     @Override
22     public void initialize(URL location, ResourceBundle resources) {
23         btnMain.setOnAction(e -> handleBtnMain(e));
24     }
25
26     public void handleBtnMain(ActionEvent event) {
27         try {
28             StackPane root = (StackPane) btnMain.getScene().getRoot();
29
30             //로그인 화면의 x 좌표를 0으로 이동시켜놓음(원래 0임)
31             login.setTranslateX(0);
32
33             //Timeline 객체 생성
34             Timeline timeline = new Timeline();
35             //로그인 화면의 x 좌표 종료값으로 350을 설정
36             KeyValue keyValue = new KeyValue(login.translateXProperty(), 350);
37             //0.1초간 종료값까지 x 좌표를 변화시키도록(0 --> 350) KeyFrame 생성
38             KeyFrame keyFrame = new KeyFrame(
39                 Duration.millis(100),
40                 new EventHandler<ActionEvent>() {
41                     @Override
42                     public void handle(ActionEvent event) {
43                         //애니메이션이 종료된 후, StackPane에서 로그인 화면 제거
44                         root.getChildren().remove(login);
45                     }
46                 },
47                 keyValue
48             );
49             //Timeline에 KeyFrame 추가
50             timeline.getKeyFrames().add(keyFrame);
51             //애니메이션 시작
52             timeline.play();
53         } catch (Exception e) {
54             e.printStackTrace();
55         }
56     }
57 }

```



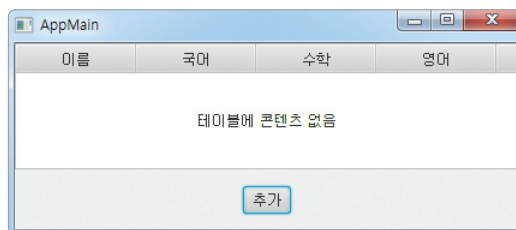
13 JavaFX 과제

지금까지 학습한 내용을 기반으로 5가지 과제를 수행해 보자.

과제 1

학생들의 점수를 보여주는 애플리케이션의 메인 윈도우를 다음과 같이 만든다.

생성해야 할 파일과 역할은 다음 표와 같다.

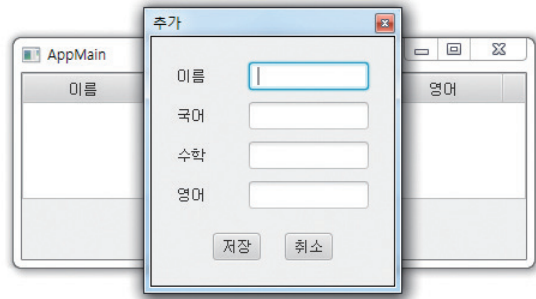


파일명	역할
AppMain.java	메인 클래스
root.fxml	메인 윈도우 레이아웃
RootController.java	컨트롤러 클래스

root.fxml의 루트 태그는 BorderPane 컨테이너로 작성한다. 중앙에는 TableView를 배치하고 하단에는 HBox를 배치한다. 그리고 HBox에는 [추가] 버튼을 배치한다. TableView에는 이름, 국어, 수학, 영어 컬럼을 추가한다.

과제 2

과제 1에 이어서 메인 윈도우에서 [추가] 버튼을 클릭하면 학생 정보 추가 다이얼로그가 다음과 같이 실행되도록 만든다.
생성해야 할 파일과 역할은 다음 표와 같다.

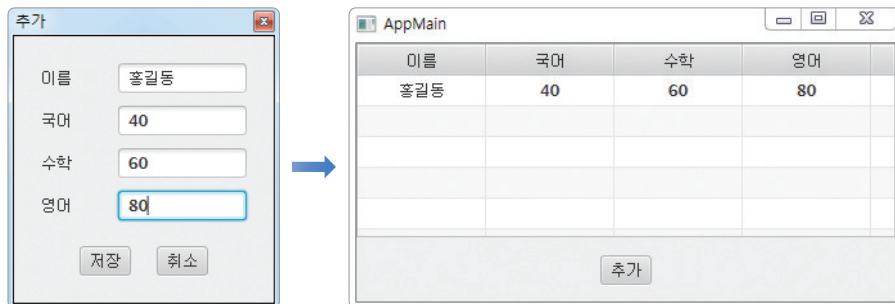


파일명	역할
form.fxml	학생 정보 추가 다이얼로그 레이아웃

form.fxml의 루트 태그는 BorderPane 컨테이너로 작성한다. 중앙에는 GridPane 컨테이너를 배치하고 네 개의 Label과 네 개의 TextField를 배치한다. 하단에는 HBox를 배치하고 [저장], [취소] 버튼을 배치한다. 그리고 학생 정보 추가 다이얼로그에서 [취소] 버튼을 클릭하면 다이얼로그가 닫히도록 이벤트 처리한다.

과제 3

과제 2에 이어서 학생 정보 추가 다이얼로그에서 정보를 입력하고 [저장] 버튼을 클릭하면 메인 윈도우의 TableView에 행이 추가되도록 한다.

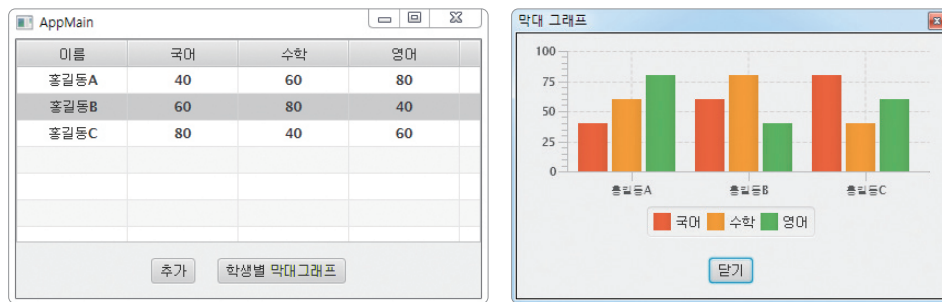


생성해야 할 파일과 역할은 다음 표와 같다.

파일명	역할
Student.java	테이블 행의 데이터를 가지고 있는 모델 클래스

과제 4

과제 3에 이어서 메인 윈도우 하단에 [학생별 막대 그래프] 버튼을 추가하고, 이 버튼을 클릭하면 다음 그림과 같이 막대 그래프 다이얼로그가 실행되도록 한다.



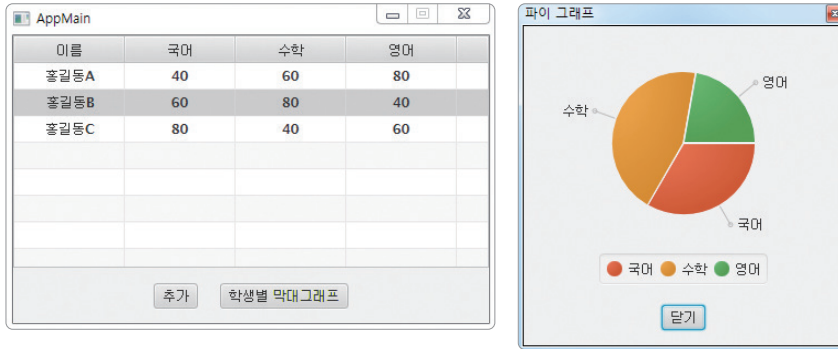
생성해야 할 파일과 역할은 다음 표와 같다.

파일명	역할
barchart.fxml	막대 그래프 다이얼로그의 레이아웃

barchart.fxml의 루트 태그는 BorderPane 컨테이너로 작성한다. 중앙에는 BarChart를 배치하고 하단에는 HBox를 배치한다. 그리고 HBox에는 [닫기] 버튼을 배치한다. [닫기] 버튼을 클릭하면 막대 그래프 다이얼로그가 닫히도록 이벤트 처리를 한다.

과제 5

과제 4에 이어서 메인 윈도우의 TableView에서 한 행을 더블 클릭하면 오른쪽 그림과 같이 해당 학생의 과목 점수를 보여주는 파이 그래프 다이얼로그가 실행되도록 한다.



생성해야 할 파일과 역할은 다음 표와 같다.

파일명	역할
piechart.fxml	파이 그래프 다이얼로그의 레이아웃

piechart.fxml의 루트 태그는 BorderPane 컨테이너로 작성한다. 중앙에는 PieChart를 배치하고 하단에는 HBox를 배치한다. 그리고 HBox에는 [닫기] 버튼을 배치한다. [닫기] 버튼을 클릭하면 파이 그래프 다이얼로그가 닫히도록 이벤트 처리한다.

TableView의 setOnMouseClicked() 메소드로 마우스 이벤트 핸들러를 등록하고, 이벤트 핸들러에서 MouseEvent의 getClickCount() 메소드가 2를 리턴할 경우(더블 클릭)에만 파이 그래프 다이얼로그가 실행되도록 한다.

Appendix

04

▶ NIO 기반 입출력 및 네트워킹

- 01 NIO 소개
- 02 파일과 디렉토리
- 03 버퍼
- 04 파일 입출력
- 05 파일 비동기 입출력
- 06 TCP 네트워크 입출력
- 07 TCP 비동기 네트워크 입출력
- 08 UDP 네트워크 입출력
- 09 NIO 과제

01 NIO 소개

Java 4부터 새로운 입출력(New Input/Output, NIO)이라는 뜻에서 `java.nio` 패키지가 포함되었는데, Java 7로 버전 업그레이드가 되면서 자바 IO와 NIO 사이의 일관성 없는 클래스 설계를 바로 잡고 비동기 채널 등의 네트워크 지원을 대폭 강화한 NIO.2 API가 추가되었다.

NIO.2는 `java.nio2` 패키지로 제공되지 않고 기존 `java.nio`의 하위 패키지(`java.nio.channels`, `java.nio.charset`, `java.nio.file`)에 통합되어 있다. 이 책에서는 NIO와 NIO.2를 구별하지 않고 그냥 NIO로 부르겠다. 다음은 NIO에서 제공하는 패키지에 대해 간략히 설명한 표이다.

NIO 패키지	포함되어 있는 내용
<code>java.nio</code>	다양한 버퍼 클래스
<code>java.nio.channels</code>	파일 채널, TCP 채널, UDP 채널 등의 클래스
<code>java.nio.channels.spi</code>	<code>java.nio.channels</code> 패키지를 위한 서비스 제공자 클래스
<code>java.nio.charset</code>	문자셋, 인코더, 디코더 API
<code>java.nio.charset.spi</code>	<code>java.nio.charset</code> 패키지를 위한 서비스 제공자 클래스
<code>java.nio.file</code>	파일 및 파일 시스템에 접근하기 위한 클래스
<code>java.nio.file.attribute</code>	파일 및 파일 시스템의 속성에 접근하기 위한 클래스
<code>java.nio.file.spi</code>	<code>java.nio.file</code> 패키지를 위한 서비스 제공자 클래스

IO와 NIO의 차이점

IO와 NIO는 데이터를 입출력한다는 목적은 동일하지만 방식에 있어서 크게 차이가 난다. 다음 표는 IO와 NIO의 차이점을 정리한 것이다.

구분	IO	NIO
입출력 방식	스트림 방식(단방향)	채널 방식(양방향)
버퍼 방식	넌버퍼(non-buffer)	버퍼(buffer)
비동기 방식	지원 안 함	지원

1) 스트림 vs. 채널

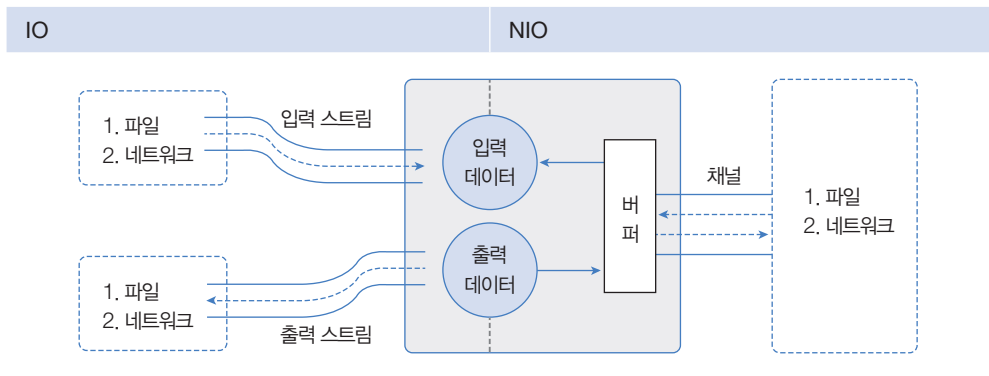
IO는 단방향 스트림^{Stream} 기반이다. 스트림은 입력 스트림과 출력 스트림으로 구분되어 있기 때문에 데이터를 읽기 위해서는 입력 스트림을 생성해야 하고, 데이터를 출력하기 위해서는 출력 스트림을 생성해야 한다. 예를 들어 하나의 파일에서 데이터를 읽고 저장하는 작업을 모두 해야 한다면 `FileInputStream`과 `FileOutputStream`을 별도로 생성해야 한다.

NIO는 채널^{Channel} 기반이다. 채널은 스트림과 달리 양방향으로 입력과 출력이 가능하다. 그렇기 때문에 입력과 출력을 위한 별도의 채널을 만들 필요가 없다. 예를 들어 하나의 파일에서 데이터를 읽고 저장하는 작업을 모두 해야 한다면 `FileChannel` 하나만 생성하면 된다.

2) 넌버퍼 vs. 버퍼

IO에서는 출력 스트림이 1바이트를 쓰면 입력 스트림이 1바이트를 읽는다. 이런 시스템은 대체로 속도가 느리다. 따라서 이것보다는 버퍼^{Buffer}: 메모리 저장소를 사용해서 복수 개의 바이트를 한꺼번에 입력받고 출력하는 것이 빠른 성능을 낸다. 그래서 IO는 버퍼를 제공해주는 보조 스트림인 `BufferedInputStream`, `BufferedOutputStream`을 연결해서 사용하기도 한다.

NIO는 기본적으로 버퍼를 사용해서 입출력을 하기 때문에 IO보다는 입출력 성능이 좋다. 채널은 버퍼에 저장된 데이터를 출력하고, 입력된 데이터를 버퍼에 저장한다.



IO는 스트림에서 입력된 데이터를 별도로 저장하지 않으면 입력된 데이터의 위치를 이동해가면서 자유롭게 읽을 수 없다. 반면 NIO는 읽은 데이터를 무조건 버퍼에 저장하기 때문에 버퍼 내에서 데이터의 위치를 이동해가면서 필요한 부분만 읽을 수 있다.

IO와 NIO의 선택

NIO는 스레드풀을 이용한 비동기로 처리할 수 있기 때문에 스레드를 효과적으로 재사용한다는 점에서 큰 장점이 있다. 또한 운영체제의 버퍼(다이렉트 버퍼)를 이용한 입출력이 가능하기 때문에 입출력 성능이 향상된다.

NIO는 처리 작업 수가 많고 하나의 작업이 오래 걸리지 않는 경우에 사용하는 것이 좋다. 반대로 처리 작업 수가 적고, 데이터가 대용량이면서 순차적으로 처리될 필요성이 있을 경우에는 IO로 구현하는 것이 좋다. 대용량 데이터일 경우에는 NIO 버퍼 크기가 문제가 되기 때문이다.

02 파일과 디렉토리

IO는 파일의 속성 정보를 읽기 위해 File 클래스만 제공하지만, NIO는 좀 더 다양한 파일의 속성 정보를 제공해주는 클래스와 인터페이스를 `java.nio.file`, `java.nio.file.attribute` 패키지에서 제공한다.

Path 클래스

Path는 IO의 `java.io.File` 클래스에 대응되는 NIO 인터페이스이다. NIO에서는 파일 경로를 지정할 때 Path를 사용하기 때문에 Path 사용 방법을 잘 익혀두어야 한다. Path 구현 객체를 얻기 위해서는 `java.nio.file.Paths` 클래스의 정적 메소드인 `get()` 메소드를 호출하면 된다.

```
Path path = Paths.get(String first, String... more)
Path path = Paths.get(URI uri);
```

`get()` 메소드의 매개값은 파일의 경로인데, 문자열로 지정할 수도 있고 URI 객체로도 지정할 수 있다. 문자열로 지정할 경우 전체 경로를 한꺼번에 지정해도 좋고, 상위 디렉토리나 하위 디렉토리를 나열해서 지정해도 좋다.

다음은 'C:\Temp\dir\file.txt' 경로를 이용해서 Path 객체를 얻는 방법이다.

```
Path path = Paths.get("C:/Temp/dir/file.txt");
Path path = Paths.get("C:/Temp/dir", "file.txt");
Path path = Paths.get("C:", "Temp", "dir", "file.txt");
```

파일의 경로는 절대 경로와 상대 경로를 모두 사용할 수 있다. 만약 현재 디렉토리 위치가 'C:\Temp' 일 경우 'C:\Temp\dir\file.txt'는 다음과 같이 지정이 가능하다.

```
Path path = Paths.get("dir/file.txt");
Path path = Paths.get("../dir/file.txt");
```

현재 위치가 'C:\Temp\dir1'이라면 'C:\Temp\dir2\file.txt'는 다음과 같이 지정할 수 있다.

```
Path path = Paths.get("../dir2/file.txt");
```

Path 인터페이스에는 다음과 같이 파일 경로에서 얻을 수 있는 여러 가지 정보를 제공하는 메소드가 있다.

리턴 타입	메소드(매개변수)	설명
int	compareTo(Path other)	파일 경로가 동일하면 0을 리턴, 상위 경로면 음수, 하위 경로면 양수를 리턴. 음수와 양수의 값은 차이나는 문자열의 수
Path	getFileName()	부모 경로를 제외한 파일 이름만 가진 Path 리턴
FileSystem	getFileSystem()	FileSystem 객체 리턴
Path	getName(int index)	C:/Temp/dir/file.txt일 경우 index가 0이면 'Temp'의 Path 객체 리턴 index가 1이면 'dir'의 Path 객체 리턴 index가 2이면 'file.txt'의 Path 객체 리턴
int	getNameCount()	중첩 경로의 수. C:/Temp/dir/file.txt일 경우 3을 리턴
Path	getParent()	바로 위 부모 폴더의 Path 리턴
Path	getRoot()	루트 디렉토리의 Path 리턴
Iterator<Path>	iterator()	경로에 있는 모든 디렉토리와 파일을 Path 객체로 생성하고 반복자를 리턴
Path	normalize()	상대 경로로 표기할 때 불필요한 요소를 제거 C:/Temp/dir1/./dir2/file.txt → C:/Temp/dir2/file.txt
WatchKey	register(...)	WatchService를 등록(와치 서비스에서 설명함)
File	toFile()	java.io.File 객체로 리턴
String	toString()	파일 경로를 문자열로 리턴
URI	toUri()	파일 경로를 URI 객체로 리턴

다음 예제는 상대 경로를 이용해서 Path 객체를 얻고 파일명, 부모 디렉토리명, 중첩 경로 수, 경로 상에 있는 모든 디렉토리를 출력한다.

>>> PathExample.java

```
1  package sec02.exam01_path;
2
3  import java.nio.file.Path;
4  import java.nio.file.Paths;
5  import java.util.Iterator;
6
7  public class PathExample {
8      public static void main(String[] args) throws Exception {
9          Path path = Paths.get("src/sec02/exam01_path/PathExample.java");
10         System.out.println("[파일명] " + path.getFileName());
11         System.out.println("[부모 디렉토리명]: " + path.getParent().getFileName());
12         System.out.println("[중첩 경로 수]: " + path.getNameCount());
13
14         System.out.println();
15         for(int i=0; i<path.getNameCount(); i++) {
16             System.out.println(path.getName(i));
17         }
18
19         System.out.println();
20         Iterator<Path> iterator = path.iterator();
21         while(iterator.hasNext()) {
22             Path temp = iterator.next();
23             System.out.println(temp.getFileName());
24         }
25     }
26 }
```

실행 결과

```
[파일명] PathExample.java
[부모 디렉토리명]: exam01_path
[중첩 경로 수]: 4
```

```
src
sec02
exam01_path
```

```
PathExample.java
```

```
src
sec02
exam01_path
PathExample.java
```

FileSystem 클래스

운영체제의 파일 시스템은 `FileSystem` 인터페이스를 통해서 접근할 수 있다. `FileSystem` 구현 객체는 `FileSystems`의 정적 메소드인 `getDefault()`로 얻을 수 있다.

```
FileSystem fileSystem = FileSystems.getDefault();
```

`FileSystem`은 다음과 같은 메소드를 제공한다.

리턴타입	메소드(매개변수)	설명
<code>Iterable<FileStore></code>	<code>getFileStores()</code>	드라이버 정보를 가진 <code>FileStore</code> 객체들을 리턴
<code>Iterable<Path></code>	<code>getRootDirectories()</code>	루트 디렉토리 정보를 가진 <code>Path</code> 객체들을 리턴
<code>String</code>	<code>getSeparator()</code>	디렉토리 구분자 리턴

`FileStore`는 드라이버를 표현한 객체로 다음과 같은 메소드를 제공한다.

리턴 타입	메소드(매개변수)	설명
<code>long</code>	<code>getTotalSpace()</code>	드라이버 전체 공간 크기(단위: 바이트) 리턴
<code>long</code>	<code>getUnallocatedSpace()</code>	할당되지 않은 공간 크기(단위: 바이트) 리턴
<code>long</code>	<code>getUsableSpace()</code>	사용 가능한 공간 크기, <code>getUnallocatedSpace()</code> 와 동일값
<code>boolean</code>	<code>isReadOnly()</code>	읽기 전용 여부
<code>String</code>	<code>name()</code>	드라이버명 리턴
<code>String</code>	<code>type()</code>	파일 시스템 종류

다음 예제는 드라이버명과 파일 시스템에 대한 정보를 보여준다.

>>> PathExample.java

```
1  package sec02.exam02_filesystem;
2
3  import java.nio.file.FileStore;
4  import java.nio.file.FileSystem;
5  import java.nio.file.FileSystems;
6  import java.nio.file.Path;
7
8  public class FileSystemExample {
9      public static void main(String[] args) throws Exception {
10         FileSystem fileSystem = FileSystems.getDefault();
11         for(FileStore store : fileSystem.getFileStores()) {
12             System.out.println("드라이버명: " + store.name());
13             System.out.println("파일시스템: " + store.type());
14             System.out.println("전체 공간: \t\t" + store.getTotalSpace() + " 바이트");
15             System.out.println("사용 중인 공간: \t" +
16                 (store.getTotalSpace() - store.getUnallocatedSpace()) + " 바이트");
17             System.out.println("사용 가능한 공간: \t" + store.getUsableSpace() +
18                 " 바이트");
19             System.out.println();
20
21             System.out.println("파일 구분자: " + fileSystem.getSeparator());
22             System.out.println();
23
24             for(Path path : fileSystem.getRootDirectories()) {
25                 System.out.println(path.toString());
26             }
27         }
28     }
```

실행 결과

드라이버명: 운영체제
파일시스템: NTFS
전체 공간: 255382777856 바이트
사용 중인 공간: 135942008832 바이트
사용 가능한 공간: 119440769024 바이트

드라이버명: 작업디스크

파일시스템: NTFS
전체 공간: 256058060800 바이트
사용 중인 공간: 197464064 바이트
사용 가능한 공간: 255860596736 바이트

드라이버명: 로컬디스크
파일시스템: NTFS
전체 공간: 8001545039872 바이트
사용 중인 공간: 1677118210048 바이트
사용 가능한 공간: 6324426829824 바이트

파일 구분자: \

C:\
D:\
E:\

Files 클래스

Files 클래스는 파일과 디렉토리 생성 및 삭제, 그리고 이들의 속성을 읽는 메소드를 제공하고 있다. 여기서 속성이란 파일이나 디렉토리가 숨김인지, 디렉토리인지, 크기가 어떻게 되는지, 소유자가 누구인지에 대한 정보를 말한다.

다음은 Files 클래스가 제공하는 정적 메소드들이다. 매개변수에 대한 자세한 설명은 API 도큐먼트를 참조하길 바란다.

리턴 타입	메소드(매개변수)	설명
long 또는 Path	copy(...)	복사
Path	createDirectories(...)	경로에 있는 모든 디렉토리 생성
Path	createDirectory(...)	경로의 마지막 디렉토리만 생성
Path	createFile(...)	파일 생성
void	delete(...)	삭제
boolean	deleteIfExists(...)	존재한다면 삭제
boolean	exists(...)	존재 여부
FileStore	getFileStore(...)	파일이 위치한 FileStore(드라이브) 리턴

FileTime	getLastModifiedTime(...)	마지막 수정 시간을 리턴
UserPrincipal	getOwner(...)	소유자 정보를 리턴
boolean	isDirectory(...)	디렉토리인지 여부
boolean	isExecutable(...)	실행 가능 여부
boolean	isHidden(...)	숨김 여부
boolean	isReadable(...)	읽기 가능 여부
boolean	isRegularFile(...)	일반 파일인지 여부
boolean	isSameFile(...)	같은 파일인지 여부
boolean	isWritable(...)	쓰기 가능 여부
Path	move(...)	파일 이동
BufferedReader	newBufferedReader(...)	텍스트 파일을 읽는 BufferedReader 리턴
BufferedWriter	newBufferedWriter(...)	텍스트 파일에 쓰는 BufferedWriter 리턴
SeekableByteChannel	newByteChannel(...)	파일에 읽고 쓰는 바이트 채널을 리턴
DirectoryStream<Path>	newDirectoryStream(...)	디렉토리의 모든 내용을 스트림으로 리턴
InputStream	newInputStream(...)	파일의 InputStream 리턴
OutputStream	newOutputStream(...)	파일의 OutputStream 리턴
boolean	notExists(...)	존재하지 않는지 여부
String	probeContentType(...)	파일의 MIME 타입을 리턴
byte[]	readAllBytes(...)	파일의 모든 바이트를 읽고 배열로 리턴
List<String>	readAllLines(...)	텍스트 파일의 모든 라인을 읽고 리턴
long	size(...)	파일의 크기 리턴
Path	write(...)	파일에 바이트나 문자열을 저장

다음 예제는 파일의 속성을 읽고 출력한다.

» FileExample.java

```

1  package sec02.exam03_file_directory;
2
3  import java.nio.file.Files;
4  import java.nio.file.Path;
5  import java.nio.file.Paths;

```

```

6
7  public class FileExample {
8      public static void main(String[] args) throws Exception {
9          Path path = Paths.get("src/sec02/exam03_file_directory/
              FileExample.java");
10         System.out.println("디렉토리 여부: " + Files.isDirectory(path));
11         System.out.println("파일 여부: " + Files.isRegularFile(path));
12         System.out.println("마지막 수정 시간: " + Files.getLastModifiedTime
              (path));
13         System.out.println("파일 크기: " + Files.size(path));
14         System.out.println("소유자: " + Files.getOwner(path).getName());
15         System.out.println("숨김 파일 여부: " + Files.isHidden(path));
16         System.out.println("읽기 가능 여부: " + Files.isReadable(path));
17         System.out.println("쓰기 가능 여부: " + Files.isWritable(path));
18     }
19 }

```

실행 결과

```

디렉토리 여부: false
파일 여부: true
마지막 수정 시간: 2022-03-15T11:13:32.1599634Z
파일 크기: 883
소유자: BLUESKII-REMOTE\blueskii
숨김 파일 여부: false
읽기 가능 여부: true
쓰기 가능 여부: true

```

다음 예제는 디렉토리와 파일을 생성하고, 디렉토리의 내용을 출력한다. 예제를 실행하려면 C:/Temp 디렉토리가 존재해야 한다. 없다면 생성 후 실행하자.

»» DirectoryExample.java

```

1  package sec02.exam03_file_directory;
2
3  import java.io.IOException;
4  import java.nio.file.DirectoryStream;
5  import java.nio.file.Files;

```

```

6  import java.nio.file.Path;
7  import java.nio.file.Paths;
8
9  public class DirectoryExample {
10     public static void main(String[] args) {
11         try {
12             //디렉토리 및 파일 생성
13             Path path = Paths.get("C:/Temp/file1.txt");
14             if(!Files.exists(path)) { Files.createFile(path); }
15             path = Paths.get("C:/Temp/dir1");
16             if(!Files.exists(path)) { Files.createDirectories(path); }
17             path = Paths.get("C:/Temp/dir1/file2.txt");
18             if(!Files.exists(path)) { Files.createFile(path); }
19             path = Paths.get("C:/Temp/dir1/dir2");
20             if(!Files.exists(path)) { Files.createDirectories(path); }
21             path = Paths.get("C:/Temp/dir1/dir2/file3.txt");
22             if(!Files.exists(path)) { Files.createFile(path); }
23
24             path = Paths.get("C:/Temp");
25             printDirContent(path, 0);
26         } catch(IOException e) {
27             System.out.println(e.getMessage());
28         }
29     }
30
31     //디렉토리 내용 출력
32     public static void printDirContent(Path path, int indent) {
33         try {
34             //파일들만 출력
35             DirectoryStream<Path> directoryStream = Files.newDirectoryStream
36                 (path);
37             directoryStream.forEach(p -> {
38                 if(!Files.isDirectory(p)) {
39                     //들여쓰기
40                     for(int i=0; i<indent; i++) { System.out.print("\t"); }
41                     //파일 이름과 크기 출력
42                     try {
43                         System.out.println(p.getFileName() + " (크기:" + Files.size(p)

```

```

44         System.out.println(e.getMessage());
45     }
46 }
47 });
48
49 //디렉토리들만 출력
50 directoryStream = Files.newDirectoryStream(path);
51 directoryStream.forEach(p -> {
52     if(Files.isDirectory(p)) {
53         //들여쓰기
54         for(int i=0; i<indent; i++) { System.out.print("\t"); }
55         //디렉토리 이름 출력
56         System.out.println("[ " + p.getFileName() + " ]");
57         //재귀호출, 들여쓰기를 1 추가함
58         printDirContent(p, indent+1);
59     }
60 });
61 } catch(IOException e) {
62     System.out.println(e.getMessage());
63 }
64 }
65 }

```

실행 결과

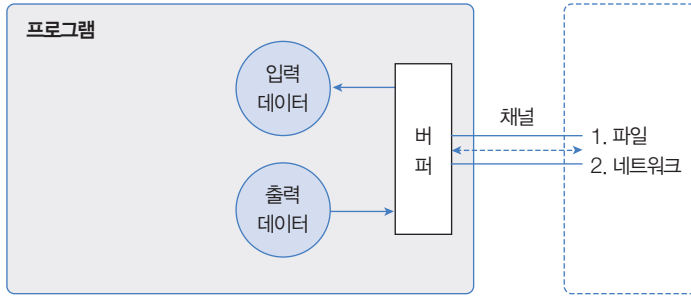
```

file1.txt (크기:0)
[dir1]
  file2.txt (크기:0)
  [dir2]
    file3.txt (크기:0)

```

03 버퍼

NIO에서는 데이터를 입출력하기 위해 항상 버퍼를 사용해야 한다. 버퍼(Buffer)는 읽고 쓰기가 가능한 메모리 배열이다. 버퍼를 이해하고 잘 사용할 수 있어야 NIO에서 제공하는 API를 올바르게 활용할 수 있다.

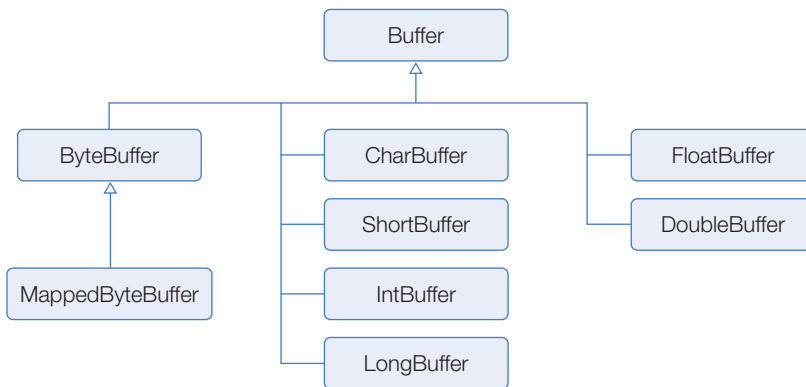


버퍼 종류

버퍼는 저장되는 데이터 타입에 따라 분류될 수 있고, 어떤 메모리를 사용하느냐에 따라 **다이렉트Direct**와 **넌다이렉트NonDirect**로 분류할 수도 있다.

1) 데이터 타입에 따른 버퍼

NIO 버퍼는 저장되는 데이터 타입에 따라서 별도의 클래스로 제공된다. 이 버퍼 클래스들은 `Buffer` 추상 클래스를 모두 상속하고 있다.



버퍼 클래스의 이름을 보면 어떤 데이터가 저장되는 버퍼인지 쉽게 알 수 있다. `ByteBuffer`는 `byte` 데이터가 저장되고, `CharBuffer`, `ShortBuffer`, `IntBuffer`, `LongBuffer`, `FloatBuffer`, `DoubleBuffer`는 각각 `char`, `short`, `int`, `long`, `float`, `double` 데이터가 저장되는 버퍼이다.

MappedByteBuffer는 ByteBuffer의 하위 클래스로 파일의 내용에 랜덤하게 접근하기 위해서 파일의 내용을 메모리와 맵핑시킨 버퍼이다.

2) 넌다이렉트와 다이렉트 버퍼

버퍼가 생성되는 메모리의 위치에 따라서 넌다이렉트^{non-direct} 버퍼와 다이렉트^{direct} 버퍼로 분류된다. 넌다이렉트 버퍼는 JVM이 관리하는 힙 메모리 공간에 생성되는 버퍼이고, 다이렉트 버퍼는 운영체제가 관리하는 메모리 공간에 생성되는 버퍼이다. 두 버퍼의 특징은 다음과 같다.

구분	넌다이렉트 버퍼	다이렉트 버퍼
사용하는 메모리 공간	JVM의 힙 메모리	운영체제의 메모리
버퍼 생성 시간	버퍼 생성이 빠르다.	버퍼 생성이 느리다.
버퍼의 크기	작다.	크다(큰 데이터를 처리할 때 유리).
입출력 성능	낮다.	높다(입출력이 빈번할 경우 유리).

- 넌다이렉트 버퍼는 JVM 힙 메모리를 사용하므로 버퍼 생성 시간이 빠르지만, 다이렉트 버퍼는 운영체제로부터 메모리를 할당받아야 하므로 상대적으로 버퍼 생성이 느리다. 그렇기 때문에 다이렉트 버퍼는 자주 생성하기보다는 한 번 생성해 놓고 재사용하는 것이 적합하다.
- 넌다이렉트 버퍼는 JVM의 제한된 힙메모리를 사용하므로 버퍼의 크기를 작게 잡는 것이 좋고, 다이렉트 버퍼는 운영체제가 관리하는 메모리를 사용하므로 운영체제가 허용하는 범위 내에서 대용량 버퍼를 생성시킬 수 있다.
- 넌다이렉트 버퍼는 I/O(입출력)를 하기 위해 임시 다이렉트 버퍼를 생성하고 넌다이렉트 버퍼에 있는 내용을 임시 다이렉트 버퍼에 복사한다. 그리고 나서 임시 다이렉트 버퍼의 내용으로 운영체제의 I/O기능을 수행한다. 그렇기 때문에 직접 다이렉트 버퍼를 사용하는 것보다는 I/O 성능이 낮다.

버퍼 생성

각 데이터 타입별로 넌다이렉트 버퍼를 생성하기 위해서는 각 Buffer 클래스의 `allocate()`와 `wrap()` 메소드를 이용하고, 다이렉트 버퍼는 ByteBuffer의 `allocateDirect()` 메소드를 이용한다.

1) `allocate()`와 `wrap()` 메소드

데이터 타입별로 Buffer의 `allocate()` 메소드를 호출하면 JVM 힙 메모리에 넌다이렉트 버퍼를 생성한다. capacity 매개값은 데이터 저장 개수이다.

리턴 타입	메소드(매개변수)	설명
ByteBuffer	ByteBuffer.allocate(int capacity)	capacity개의 byte 값을 저장하는 버퍼 생성
CharBuffer	CharBuffer.allocate(int capacity)	capacity개의 char 값을 저장하는 버퍼 생성
DoubleBuffer	DoubleBuffer.allocate(int capacity)	capacity개의 double 값을 저장하는 버퍼 생성
FloatBuffer	FloatBuffer.allocate(int capacity)	capacity개의 float 값을 저장하는 버퍼 생성
IntBuffer	IntBuffer.allocate(int capacity)	capacity개의 int 값을 저장하는 버퍼 생성
LongBuffer	LongBuffer.allocate(int capacity)	capacity개의 long 값을 저장하는 버퍼 생성
ShortBuffer	ShortBuffer.allocate(int capacity)	capacity개의 short 값을 저장하는 버퍼 생성

다음은 100개의 byte[] 값을 저장하는 ByteBuffer와 100개의 int값을 저장하는 CharBuffer를 생성하는 코드이다.

```
ByteBuffer buffer = ByteBuffer.allocate(100);
```

```
CharBuffer buffer = CharBuffer.allocate(100);
```

각 데이터 타입별 Buffer 클래스는 모두 wrap() 메소드를 가지고 있는데, wrap() 메소드는 배열을 래핑해서 Buffer 객체를 생성한다. 배열은 JVM 힙 메모리에 생성되므로 wrap()은 넌다이렉트 버퍼를 생성한다.

다음은 길이가 100인 byte[]를 이용해서 ByteBuffer를 생성하고, 길이가 100인 int[]를 이용해서 CharBuffer를 생성하는 코드이다.

```
byte[] array = new byte[100];
ByteBuffer buffer = ByteBuffer.wrap(array);
```

```
char[] array = new char[100];
CharBuffer buffer = CharBuffer.wrap(array);
```

배열 전체가 아니라 일부 데이터만 가지고 Buffer 객체를 생성할 수도 있다. 이 경우 시작 인덱스와 길이를 추가적으로 지정하면 된다. 다음은 0 인덱스부터 50개만 버퍼로 생성하는 코드이다.

```
byte[] byteArray = new byte[100];
ByteBuffer byteBuffer = ByteBuffer.wrap(byteArray, 0, 50);
```

CharBuffer는 추가적으로 문자열을 제공해서 CharBuffer를 생성할 수 있는 wrap() 메소드도 제공한다.

```
CharBuffer charBuffer = CharBuffer.wrap("NIO 입출력은 버퍼를 이용합니다.");
```

wrap() 메소드로 배열을 래핑하는 버퍼를 생성할 경우에는, 버퍼의 array() 메소드로 다시 배열을 얻을 수 있다. 하지만 배열을 래핑하지 않는 버퍼는 사용할 수 없다. 다음은 ByteBuffer와 CharBuffer가 래핑하고 있는 배열을 얻는 코드이다.

```
byte[] byteArray1 = new byte[100];
ByteBuffer byteBuffer = ByteBuffer.wrap(byteArray);
byte[] byteArray2 = byteBuffer.array();
```

```
char[] charArray1 = new char[100];
CharBuffer charBuffer = CharBuffer.wrap(charArray);
char[] charArray2 = charBuffer.array();
```

다음 예제는 allocate() 메소드와 wrap() 메소드를 이용해서 넌다이렉트 버퍼를 생성하는 방법을 보여준다.

»» NonDirentBufferExample.java

```
1  package sec03.exam01_create_buffer;
2
3  import java.nio.ByteBuffer;
4  import java.nio.CharBuffer;
5  import java.nio.IntBuffer;
6  import java.util.Arrays;
7
```

```

8  public class NonDirentBufferExample {
9      public static void main(String[] args) {
10         //넌다이렉트 ByteBuffer 생성
11         ByteBuffer buffer1 = ByteBuffer.allocate(100);
12         System.out.println(buffer1);
13         System.out.println();
14
15         //넌다이렉트 IntBuffer 생성
16         IntBuffer buffer2 = IntBuffer.allocate(100);
17         System.out.println(buffer2);
18         System.out.println();
19
20         //배열을 래핑해서 넌다이렉트 ByteBuffer 생성
21         byte[] array3 = { 10, 20 };
22         ByteBuffer buffer3 = ByteBuffer.wrap(array3);
23         System.out.println(buffer3 + ", ");
24         //래핑된 배열을 얻어 출력하기
25         System.out.println(Arrays.toString(buffer3.array()));
26         System.out.println();
27
28         //배열을 래핑해서 넌다이렉트 CharBuffer 생성
29         char[] array4 = "This is Java".toCharArray();
30         CharBuffer buffer4 = CharBuffer.wrap(array4);
31         System.out.println(buffer4);
32         //래핑된 배열을 얻어 출력하기
33         System.out.println(Arrays.toString(buffer4.array()));
34     }
35 }

```

실행 결과

```
java.nio.HeapByteBuffer[pos=0 lim=100 cap=100]
```

```
java.nio.HeapIntBuffer[pos=0 lim=100 cap=100]
```

```
java.nio.HeapByteBuffer[pos=0 lim=2 cap=2],
[10, 20]
```

```
This is Java
```

```
[T, h, i, s, , i, s, , J, a, v, a]
```

3) allocateDirect() 메소드

ByteBuffer의 allocateDirect() 메소드는 운영체제가 관리하는 메모리에 다이렉트 버퍼를 생성한다. 이 메소드는 각 타입별 Buffer 클래스에는 없고, ByteBuffer에서만 제공된다. 다음 예제는 100개의 byte를 저장하는 다이렉트 ByteBuffer를 생성하는 코드이다.

» DirectBufferExample.java

```
1 package sec03.exam01_create_buffer;
2
3 import java.nio.ByteBuffer;
4
5 public class DirectBufferExample {
6     public static void main(String[] args) {
7         //다이렉트 ByteBuffer 생성
8         ByteBuffer buffer = ByteBuffer.allocateDirect(100);
9         System.out.println(buffer);
10    }
11 }
```

실행 결과

```
java.nio.DirectByteBuffer[pos=0 lim=100 cap=100]
```

버퍼 위치 속성

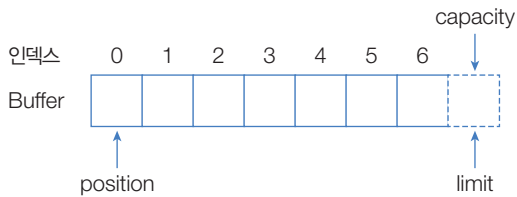
버퍼를 생성하는 방법을 알았으니 이제는 사용하는 방법을 알아보자. 버퍼를 사용하려면 먼저 버퍼의 위치 속성에 대해 알고 있어야 한다. 다음은 버퍼의 네 가지 위치 속성이다.

속성	설명
capacity	- 버퍼에 저장할 수 있는 최대 데이터 수
limit	- 읽거나 쓸 수 있는 한계 - 버퍼 생성 시 limit과 capacity는 같은 값을 가짐
position	- 현재 읽거나 쓰는 위치 - position이 limit이 되면 더 이상 데이터를 쓰거나 읽을 수 없음
mark	- reset() 메소드를 실행했을 때에 되돌아올 위치 - position나 limit의 값이 mark 값보다 작은 경우 mark는 자동 제거

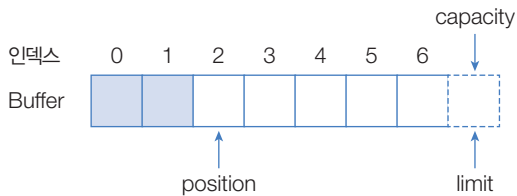
position, limit, capacity, mark 속성의 크기 관계는 다음과 같다. mark는 position보다 클 수 없고, position은 limit보다 클 수 없으며, limit은 capacity보다 클 수 없다.

$$0 \leq \text{mark} \leq \text{position} \leq \text{limit} \leq \text{capacity}$$

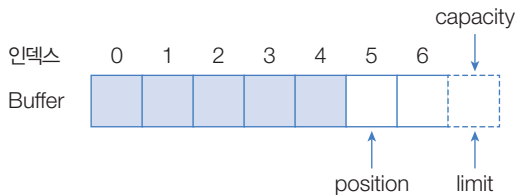
예를 들어 다음 그림처럼 7바이트 크기의 버퍼를 생성했다고 가정해 보자. 처음에는 limit과 capacity가 동일하게 7이 되고 position은 0이 된다. 버퍼의 크기가 7이므로 인덱스는 6까지이다.



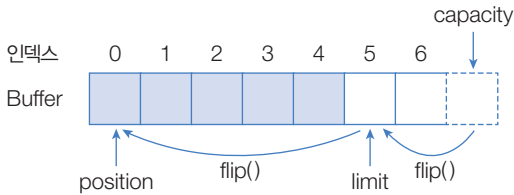
버퍼가 생성된 후 2바이트를 버퍼에 저장하면 다음 그림과 같이 position이 위치한 0 인덱스부터 2바이트가 저장되고 position은 2 인덱스로 이동한다.



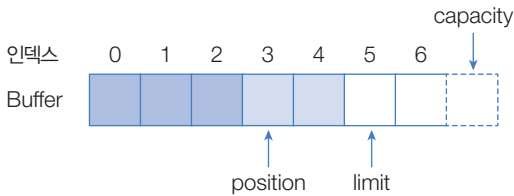
계속해서 3바이트를 저장하면 다음 그림과 같이 position 2 인덱스에서 3바이트가 저장되고, position은 5 인덱스로 이동한다.



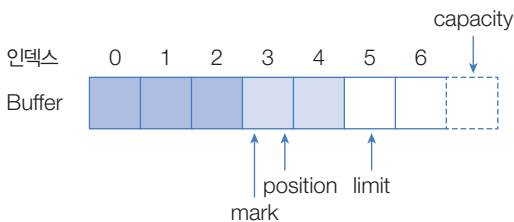
저장이 완료된 후 버퍼에 저장된 바이트를 읽으려면 먼저 `flip()` 메소드를 호출해야 한다. `flip()`을 호출하면 다음 그림과 같이 `limit`을 현재 `position` 5 인덱스로 설정하고, `position`을 0 인덱스로 설정한다.



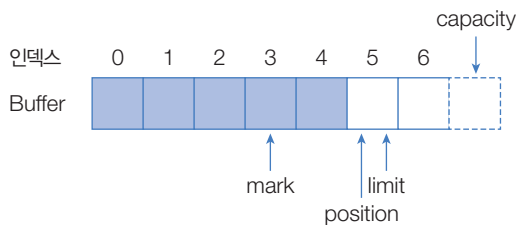
버퍼에서 3바이트를 읽으면 다음 그림과 같이 `position`이 위치한 0 인덱스부터 3바이트가 읽혀지고 `position`은 3번 인덱스로 이동한다.



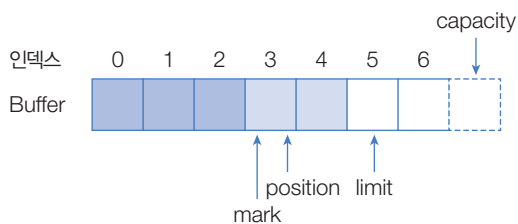
`position`이 3번 인덱스를 가리키고 있을 때 `mark()` 메소드를 호출해서 다음 그림과 같이 3번 인덱스를 마킹해놓는다.



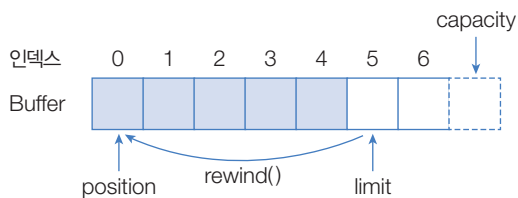
버퍼에서 2바이트를 더 읽으면 다음 그림과 같이 `position`이 위치한 3 인덱스부터 2바이트가 읽혀지고 `position`은 5 인덱스로 이동한다.



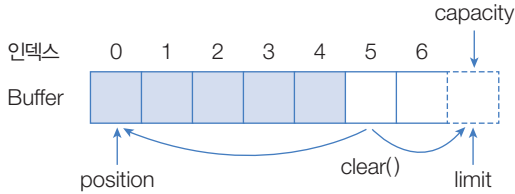
position을 mark 위치로 이동하기 위해 `reset()` 메소드를 호출한다. 다음 그림과 같이 position은 mark가 있는 3 인덱스로 이동한다. mark가 없는 상태에서 `reset()` 메소드를 호출하면 `InvalidMarkException` 예외가 발생한다.



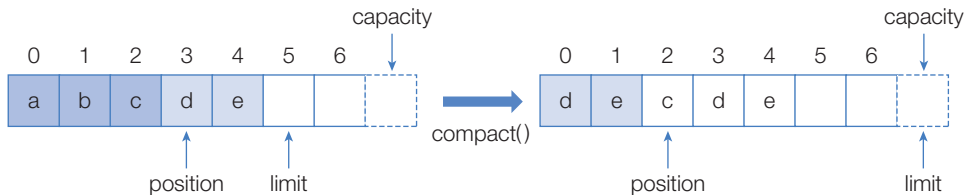
버퍼 처음으로 되돌아가기 위해 `rewind()` 메소드를 호출하면 다음 그림과 같이 limit은 변하지 않지만 position은 0 인덱스로 다시 설정된다. mark는 position이나 limit이 mark보다 작은 값으로 조정되면 자동으로 없어진다.



`rewind()` 대신 `clear()` 메소드를 호출하면 Buffer의 세 가지 속성은 초기화된다. limit은 capacity로, position은 0으로 설정되고 mark는 자동으로 없어진다. 하지만 데이터는 삭제되지 않는다.



`compact()` 메소드는 `position`에서 `limit` 사이의 데이터를 맨 앞으로 복사시키고 복사된 데이터 다음 위치로 `position`을 이동시킨다. 예를 들어 다음과 같이 `position`이 3 인덱스 위치에 있을 때 `compact()`가 호출되면 3 인덱스와 4 인덱스 데이터는 0 인덱스와 1 인덱스로 복사되고 `position`은 2 인덱스로 이동한다. 그리고 `limit`은 `capacity`로 이동한다. 0번과 1번 인덱스를 제외한 나머지 인덱스의 데이터는 삭제되지 않고 남아있다.



`compact()`를 호출하는 이유는 읽지 않은 데이터 뒤에 새로운 데이터를 저장하기 위해서이다.

버퍼 메소드

버퍼를 생성하고 사용하려면 버퍼가 제공하는 메소드를 잘 활용해야 한다. 버퍼들마다 공통으로 제공하는 메소드들도 있고, 각 타입별로 제공되는 메소드들도 있다.

1) 공통 메소드

모든 버퍼 클래스는 `Buffer` 추상 클래스를 상속하고 있기 때문에 `Buffer` 클래스에서 제공하는 메소드는 모든 버퍼에서 사용할 수 있다. 위치 속성을 변경하는 `flip()`, `rewind()`, `clear()`, `mark()`, `reset()` 메소드는 모두 `Buffer` 추상 클래스가 제공한다.

다음은 `Buffer` 추상 클래스가 가지고 있는 메소드를 정리한 표이다.

리턴 타입	메소드(매개변수)	설명
Object	array()	버퍼가 래핑(wrap)한 배열을 리턴
int	arrayOffset()	버퍼의 첫 번째 요소가 있는 내부 배열의 인덱스를 리턴
int	capacity()	버퍼의 전체 크기를 리턴
Buffer	clear()	버퍼의 위치 속성을 초기화(position=0, limit=capacity)
Buffer	flip()	limit을 position으로, position을 0 인덱스로 이동
boolean	hasArray()	버퍼가 래핑(wrap)한 배열을 가지고 있는지 여부
boolean	hasRemaining()	position과 limit 사이에 요소가 있는지 여부
boolean	isDirect()	운영체제의 버퍼를 사용하는지 여부
boolean	isReadOnly()	버퍼가 읽기 전용인지 여부
int	limit()	limit 위치를 리턴
Buffer	limit(int newLimit)	newLimit으로 limit 위치를 설정
Buffer	mark()	현재 위치를 mark로 표시
int	position()	position 위치를 리턴
Buffer	position(int newPosition)	newPosition으로 position 위치를 설정
int	remaining()	position과 limit 사이의 요소의 개수
Buffer	reset()	position을 mark 위치로 이동
Buffer	rewind()	position을 0 인덱스로 이동

2) 데이터를 읽고 저장하는 메소드

버퍼에 데이터를 저장하는 메소드는 put()이고, 데이터를 읽는 메소드는 get()이다. 이 메소드들은 Buffer 추상 클래스에는 없고 각 타입별 버퍼 클래스가 가지고 있다.

get()과 put() 메소드는 상대적^{Relative}과 절대적^{Absolute}으로 구분된다. position에서 데이터를 읽고 저장할 경우는 상대적이고, position과 상관없이 주어진 인덱스에서 데이터를 읽고 저장하면 절대적이다.

다음은 ByteBuffer와 CharBuffer에서 제공하는 get()과 put() 메소드이다. ShortBuffer, IntBuffer, LongBuffer, FloatBuffer, DoubleBuffer도 데이터 타입만 다를 뿐 비슷한 메소드를 가지고 있다.

구분		ByteBuffer	CharBuffer
get()	상대적	get() get(byte[] dst) get(byte[] dst, int offset, int length) getChar() getDouble() getFloat() getInt() getLong() getShort()	get() get(char[] dst) get(char[] dst, int offset, int length)
	절대적	get(int index) getChar(int index) getDouble(int index) getFloat(int index) getInt(int index) getLong(int index) getShort(int index)	get(int index)
put()	상대적	put(byte b) put(byte[] src) put(byte[] src, int offset, int length) put(ByteBuffer src) putChar(char value) putDouble(double value) putFloat(float value) putInt(int value) putLong(long value) putShort(short value)	put(char c) put(char[] src) put(char[] src, int offset, int length) put(CharBuffer src) put(String src) put(String src, int start, int end)
	절대적	put(int index, byte b) putChar(int index, char value) putDouble(int index, double value) putFloat(int index, float value) putInt(int index, int value) putLong(int index, long value) putShort(int index, short value)	put(int index, char c)

상대적 메소드와 절대적 메소드를 쉽게 구분하는 방법은 다음과 같다. index 매개변수가 없으면 상대적이고, index 매개변수가 있으면 절대적이다. 상대적 get()과 put() 메소드를 호출하면 position이 이동하지만, 절대적 get()과 put() 메소드를 호출하면 position은 이동되지 않는다.

다음 예제는 데이터를 버퍼에 쓰고 읽기 위해 `put()`과 `get()` 메소드를 호출할 때와 위치 속성을 변경하는 메소드를 호출할 때 버퍼의 위치 속성의 변화를 보여준다.

>>> `BufferMethodExample.java`

```
1  package sec03.exam02_buffer_method;
2
3  import java.nio.Buffer;
4  import java.nio.ByteBuffer;
5
6  public class BufferMethodExample {
7      public static void main(String[] args) {
8          System.out.println("[7바이트 크기로 버퍼 생성]");
9          ByteBuffer buffer = ByteBuffer.allocateDirect(7);
10         printState(buffer);
11
12         buffer.put((byte)10);
13         buffer.put((byte)11);
14         System.out.println("[2바이트 저장 후]");
15         printState(buffer);
16
17         buffer.put((byte)12);
18         buffer.put((byte)13);
19         buffer.put((byte)14);
20         System.out.println("[3바이트 저장 후]");
21         printState(buffer);
22
23         buffer.flip();
24         System.out.println("[flip() 실행 후]");
25         printState(buffer);
26
27         buffer.get(new byte[3]);
28         System.out.println("[3바이트 읽은 후]");
29         printState(buffer);
30
31         buffer.mark();
32         System.out.println("-----[현재 위치를 마크해놓음]");
33
34         buffer.get(new byte[2]);
35         System.out.println("[2바이트 읽은 후]");
36         printState(buffer);
```

```

37
38     buffer.reset();
39     System.out.println("-----[position을 마크 위치로 옮김]");
40     printState(buffer);
41
42     buffer.rewind();
43     System.out.println("[rewind() 실행 후]");
44     printState(buffer);
45
46     buffer.clear();
47     System.out.println("[clear() 실행 후]");
48     printState(buffer);
49 }
50
51 public static void printState(Buffer buffer) {
52     System.out.print("\tposition:" + buffer.position() + ", ");
53     System.out.print("\tlimit:" + buffer.limit() + ", ");
54     System.out.println("\tcapacity:" + buffer.capacity());
55 }
56 }

```

실행 결과

```

[7바이트 크기로 버퍼 생성]
  position:0, limit:7, capacity:7
[2바이트 저장 후]
  position:2, limit:7, capacity:7
[3바이트 저장 후]
  position:5, limit:7, capacity:7
[flip() 실행 후]
  position:0, limit:5, capacity:7
[3바이트 읽은 후]
  position:3, limit:5, capacity:7
-----[현재 위치를 마크해놓음]
[2바이트 읽은 후]
  position:5, limit:5, capacity:7
-----[position을 마크 위치로 옮김]
  position:3, limit:5, capacity:7
[rewind() 실행 후]
  position:0, limit:5, capacity:7
[clear() 실행 후]
  position:0, limit:7, capacity:7

```

3) 버퍼 예외의 종류

버퍼가 데이터로 꽉 찼을 때 `put()`으로 새 데이터를 저장하려고 하면 `BufferOverflowException` 이 발생한다. 그리고 버퍼에서 더 이상 읽을 데이터가 없을 때 `get()`으로 데이터를 읽으려고 하면 `BufferUnderflowException`이다. 다음은 버퍼 메소드에서 발생할 수 있는 예외들을 보여준다.

예외	설명
<code>BufferOverflowException</code>	<code>position</code> 이 <code>limit</code> 에 도달했을 때 <code>put()</code> 을 호출하면 발생
<code>BufferUnderflowException</code>	<code>position</code> 이 <code>limit</code> 에 도달했을 때 <code>get()</code> 을 호출하면 발생
<code>InvalidMarkException</code>	<code>mark</code> 가 없는 상태에서 <code>reset()</code> 메소드를 호출하면 발생
<code>ReadOnlyBufferException</code>	읽기 전용 버퍼에서 <code>put()</code> 또는 <code>compact()</code> 메소드를 호출하면 발생

버퍼 변환

채널을 통해 데이터를 입출력할 때에는 반드시 `ByteBuffer`를 사용한다. 그렇기 때문에 문자열이나 배열을 채널을 통해 출력하려면 `ByteBuffer`로 변환해야 한다. 그리고 채널을 통해 읽은 `ByteBuffer`를 원래 타입으로 변환해서 사용해야 한다.

1) String ↔ ByteBuffer

프로그램에서 가장 많이 처리되는 데이터는 문자열이다. 채널을 통해 문자열을 파일이나 네트워크로 입출력하려면 특정 문자셋으로 문자열을 인코딩해서 `ByteBuffer`를 얻어야 하며, 반대로 `ByteBuffer`를 디코딩해서 문자열로 복원해야 한다.

특정 문자셋으로 인코딩과 디코딩을 하는 `Charset`은 다음 두 가지 방법으로 얻을 수 있다.

```
Charset charset = Charset.forName("UTF-8"); //UTF-8로 인코딩 및 디코딩
Charset charset = Charset.defaultCharset(); //운영체제의 기본 문자셋으로 인코딩 및 디코딩
```

문자열을 `ByteBuffer`로 변환하려면 다음과 같이 `Charset`의 `encode()` 메소드를 호출하면 된다.

```
String data = ...;
ByteBuffer byteBuffer = charset.encode(data);
```

반대로 ByteBuffer를 문자열로 변환하려면 다음과 같이 `decode()` 메소드를 호출하면 된다.

```
ByteBuffer byteBuffer = ...;
String data = charset.decode(byteBuffer).toString();
```

다음 예제는 문자열을 UTF-8로 인코딩해서 ByteBuffer를 얻고, 다시 UTF-8로 디코딩해서 문자열로 복원한다.

»» `ByteBufferToStringExample.java`

```
1  package sec03.exam03_convert_buffer;
2
3  import java.nio.ByteBuffer;
4  import java.nio.charset.Charset;
5
6  public class ByteBufferToStringExample {
7      public static void main(String[] args) {
8          Charset charset = Charset.forName("UTF-8");
9
10         String data = "안녕하세요";
11
12         //문자열 -> 인코딩 -> ByteBuffer
13         ByteBuffer byteBuffer = charset.encode(data);
14         System.out.println(byteBuffer);
15
16         //ByteBuffer -> 디코딩 -> 문자열
17         data = charset.decode(byteBuffer).toString();
18         System.out.println(data);
19     }
20 }
```

실행 결과

```
java.nio.HeapByteBuffer[pos=0 lim=15 cap=23]
안녕하세요
```


2) 배열 → ByteBuffer

byte[], int[], double[] 배열을 ByteBuffer로 변환하는 방법을 알아보자.

byte[] 배열을 ByteBuffer로 변환할 때에는 간단하게 wrap() 메소드를 이용하면 된다.

```
byte[] byteArray = { 10, 20 };
ByteBuffer byteBuffer = ByteBuffer.wrap(byteArray);
System.out.println(byteBuffer);
```

int[] 배열을 ByteBuffer로 변환하기 위해서는 int[]의 length보다 4배 큰 capacity를 가진 ByteBuffer가 필요하다. 그 이유는 int 타입은 4byte 크기를 가지기 때문이다. 다음은 Int[] 배열을 ByteBuffer로 변환하는 방법을 보여준다.

```
int[] intArray = { 10, 20 };
ByteBuffer byteBuffer= ByteBuffer.allocate(intArray.length * 4);
byteBuffer.asIntBuffer().put(intArray);
```

ByteBuffer의 asIntBuffer() 메소드는 ByteBuffer의 IntBuffer 뷰를 리턴한다. 뷰^{view}란 실제 값을 가지지 않는 가상 객체이다. IntBuffer 뷰의 put() 메소드를 통해 int[]을 저장하면 실제 저장되는 곳은 ByteBuffer이다. 다음은 ByteBuffer가 가지고 있는 타입별 버퍼 뷰를 제공하는 메소드들이다.

리턴 타입	변환 메소드	설명
ShortBuffer	asShortBuffer()	2byte short으로 구성된 ShortBuffer 뷰를 리턴
IntBuffer	asIntBuffer()	4byte int로 구성된 IntBuffer 뷰를 리턴
LongBuffer	asLongBuffer()	8byte long으로 구성된 LongBuffer 뷰를 리턴
FloatBuffer	asFloatBuffer()	4byte float으로 구성된 FloatBuffer 뷰를 리턴
DoubleBuffer	asDoubleBuffer()	8byte double로 구성된 DoubleBuffer 뷰를 리턴

다음은 double[] 배열을 ByteBuffer로 변환하는 방법을 보여준다.

```
double[] doubleArray = { 10.0, 20.0 };
ByteBuffer byteBuffer= ByteBuffer.allocate(doubleArray.length * 8);
byteBuffer.asDoubleBuffer().put(doubleArray);
```

다음은 byte[], int[], double[] 배열을 ByteBuffer로 변환하는 방법을 보여준다.

»» ArrayToIntBufferExample.java

```
1  package sec03.exam03_convert_buffer;
2
3  import java.nio.ByteBuffer;
4
5  public class ArrayToIntBufferExample {
6      public static void main(String[] args) throws Exception {
7          //byte[] -> ByteBuffer
8          byte[] byteArray = {10, 20};
9          ByteBuffer byteBuffer1 = ByteBuffer.wrap(byteArray);
10         System.out.println(byteBuffer1);
11
12         //int[] -> ByteBuffer
13         int[] intArray = { 10, 20 };
14         ByteBuffer byteBuffer2= ByteBuffer.allocate(intArray.length * 4);
15         byteBuffer2.asIntBuffer().put(intArray);
16         System.out.println(byteBuffer2);
17
18         //int[] -> ByteBuffer
19         double[] doubleArray = { 10.0, 20.0 };
20         ByteBuffer byteBuffer3= ByteBuffer.allocate(doubleArray.length * 8);
21         byteBuffer3.asDoubleBuffer().put(doubleArray);
22         System.out.println(byteBuffer3);
23     }
24 }
```

실행 결과

```
java.nio.HeapByteBuffer[pos=0 lim=2 cap=2]
java.nio.HeapByteBuffer[pos=0 lim=8 cap=8]
java.nio.HeapByteBuffer[pos=0 lim=16 cap=16]
```

3) ByteBuffer → 배열

ByteBuffer에 데이터를 저장한 후, position이 0이고 limit이 마지막 데이터 다음 위치에 있을 때 ByteBuffer를 배열로 변환하는 방법에 대해 알아보자.

ByteBuffer의 capacity까지 byte[] 배열로 얻으려면 array() 메소드를 사용할 수 있다.

```
byte[] byteArray = byteBuffer.array();
```

ByteBuffer의 limit까지 byte[] 배열로 얻으려면 ByteBuffer limit 길이만큼 byte[] 배열을 생성하고, get() 메소드를 이용해서 읽은 byte 값을 배열에 저장하면 된다.

```
byteArray = new byte[byteBuffer.limit()];  
byteBuffer.get(byteArray);
```

int[] 배열을 얻으려면 ByteBuffer의 IntBuffer 뷰 limit 길이만큼 int[] 배열을 생성하고, IntBuffer 뷰의 get() 메소드로 읽은 int 값을 배열에 저장하면 된다.

```
IntBuffer intBuffer = byteBuffer.asIntBuffer();  
int[] intArray = new int[intBuffer.limit()];  
intBuffer.get(intArray); //IntBuffer 뷰를 통해 읽은 int 값을 배열에 저장
```

double[] 배열을 얻으려면 ByteBuffer의 DoubleBuffer 뷰 capacity 길이만큼 double[] 배열을 생성한 뒤에, DoubleBuffer 뷰의 get() 메소드로 읽은 double 값을 배열에 저장하면 된다.

```
DoubleBuffer doubleBuffer = byteBuffer.asDoubleBuffer();  
double[] doubleArray = new double[doubleBuffer.limit()];  
doubleBuffer.get(doubleArray); //DoubleBuffer 뷰를 통해 읽은 double 값을 배열에 저장
```

다음 예제는 ByteBuffer를 byte[], int[], double[] 배열로 변환하는 방법을 보여준다.

>>> ByteBufferToArrayExample.java

```
1  package sec03.exam03_convert_buffer;  
2  
3  import java.nio.ByteBuffer;  
4  import java.nio.DoubleBuffer;
```

```

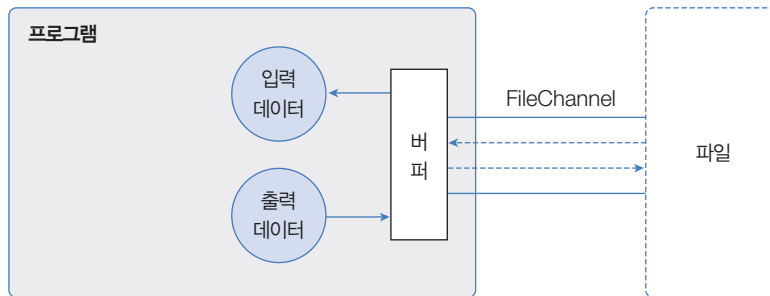
5  import java.nio.IntBuffer;
6  import java.util.Arrays;
7
8  public class ByteBufferToArrayExample {
9      public static void main(String[] args) throws Exception {
10         //ByteBuffer -> byte[] -----
11         ByteBuffer byteBuffer1 = ByteBuffer.allocate(3);
12         byte b1 = 10; byteBuffer1.put(b1);
13         byte b2 = 20; byteBuffer1.put(b2);
14         byteBuffer1.flip();
15         System.out.print(byteBuffer1 + " -> ");
16
17         byte[] byteArray = new byte[byteBuffer1.limit()];
18         byteBuffer1.get(byteArray);
19         System.out.println(Arrays.toString(byteArray));
20
21         //ByteBuffer -> int[] -----
22         ByteBuffer byteBuffer2 = ByteBuffer.allocate(16);
23         byteBuffer2.putInt(10);
24         byteBuffer2.putInt(20);
25         byteBuffer2.flip();
26         System.out.print(byteBuffer2 + " -> ");
27
28         IntBuffer intBuffer = byteBuffer2.asIntBuffer();
29         int[] intArray = new int[intBuffer.capacity()];
30         intBuffer.get(intArray);
31         System.out.println(Arrays.toString(intArray));
32
33         //ByteBuffer -> double[] -----
34         ByteBuffer byteBuffer3 = ByteBuffer.allocate(24);
35         byteBuffer3.putDouble(10.0);
36         byteBuffer3.putDouble(20.0);
37         byteBuffer3.flip();
38         System.out.print(byteBuffer3 + " -> ");
39
40         DoubleBuffer doubleBuffer = byteBuffer3.asDoubleBuffer();
41         double[] doubleArray = new double[doubleBuffer.capacity()];
42         doubleBuffer.get(doubleArray);
43         System.out.println(Arrays.toString(doubleArray));
44     }
45 }

```

```
java.nio.HeapByteBuffer[pos=0 lim=2 cap=3] -> [10, 20]
java.nio.HeapByteBuffer[pos=0 lim=8 cap=16] -> [10, 20]
java.nio.HeapByteBuffer[pos=0 lim=16 cap=24] -> [10.0, 20.0]
```

04 파일 입출력

NIO에서 제공하는 파일 채널 `FileChannel`을 이용하면 파일 읽기와 쓰기를 할 수 있다. 파일 채널은 동기화 처리가 되어 있기 때문에 멀티 스레드 환경에서 사용해도 안전하다.



파일 채널

`FileChannel`은 정적 메소드인 `open()`을 호출해서 얻을 수도 있지만, IO의 `FileInputStream`, `FileOutputStream`의 `getChannel()` 메소드를 호출해서 얻을 수도 있다. 다음은 `open()` 메소드로 `FileChannel`을 얻는 방법을 보여준다.

```
FileChannel fileChannel = FileChannel.open(Path path, OpenOption... options);
```

첫 번째 매개값은 열고자 하는 파일의 `Path` 객체이고, 두 번째 매개값부터는 다음 표에 나와있는 `StandardOpenOption`의 열거 상수를 나열해주면 된다.

열거 상수	설명
READ	읽기용으로 파일을 연다.
WRITE	쓰기용으로 파일을 연다.
CREATE	파일이 없다면 새 파일을 생성한다.
CREATE_NEW	새 파일을 만든다. 파일이 이미 있으면 예외와 함께 실패한다.
APPEND	파일 끝에 데이터를 추가한다(WRITE나 CREATE와 함께 사용됨).
DELETE_ON_CLOSE	스트림을 닫을 때 파일을 삭제한다(임시파일을 삭제할 때 사용).
TRUNCATE_EXISTING	파일을 0바이트로 잘라낸다(WRITE 옵션과 함께 사용됨).

예를 들어 'C:\Temp\file.txt' 파일을 생성하고, 내용을 쓰고 싶다면 다음과 같이 매개값을 지정하면 된다.

```
FileChannel fileChannel = FileChannel.open(
    Paths.get("C:/Temp/file.txt"),
    StandardOpenOption.CREATE_NEW,
    StandardOpenOption.WRITE
);
```

다음은 'C:\Temp\file.txt' 파일을 읽고, 쓸 수 있도록 FileChannel을 생성한다.

```
FileChannel fileChannel = FileChannel.open(
    Paths.get("C:/Temp/file.txt"),
    StandardOpenOption.READ,
    StandardOpenOption.WRITE
);
```

FileChannel을 더 이상 이용하지 않을 경우에는 다음과 같이 close() 메소드를 호출해서 닫아주어야 한다.

```
fileChannel.close();
```

파일 입출력

파일로 내용을 출력(저장)하려면 `FileChannel`의 `write()` 메소드를 호출하면 된다. 매개값으로 `ByteBuffer` 객체를 주면 되는데, 파일에 쓰여지는 바이트는 `ByteBuffer`의 `position`부터 `limit`까 지이다. `position`이 0이고 `limit`이 `capacity`와 동일하다면 `ByteBuffer`의 모든 바이트가 파일에 쓰여진다. `write()` 메소드의 리턴값은 `ByteBuffer`에서 실제로 파일로 출력된 바이트 수이다.

```
int byteNum = fileChannel.write(ByteBuffer buffer);
```

다음 예제는 `FileChannel`을 이용해서 문자열을 `C:\Temp\file.txt` 파일로 출력(저장)한다.

>>> `FileChannelWriteExample.java`

```
1  package sec04.exam01_file_read_write;
2
3  import java.io.IOException;
4  import java.nio.ByteBuffer;
5  import java.nio.channels.FileChannel;
6  import java.nio.charset.Charset;
7  import java.nio.file.Files;
8  import java.nio.file.Path;
9  import java.nio.file.Paths;
10 import java.nio.file.StandardOpenOption;
11
12 public class FileChannelWriteExample {
13     public static void main(String[] args) throws IOException {
14         //Path 생성과 디렉토리 생성
15         Path path = Paths.get("C:/Temp/file.txt");
16         Files.createDirectories(path.getParent());
17
18         //FileChannel 열기
19         FileChannel fileChannel = FileChannel.open(
20             path, StandardOpenOption.CREATE, StandardOpenOption.WRITE);
21
22         //문자열을 ByteBuffer로 변환
23         String data = "안녕하세요";
24         Charset charset = Charset.forName("UTF-8");
```

```

25     ByteBuffer byteBuffer = charset.encode(data);
26
27     //FileChannel을 통해 ByteBuffer 출력하기
28     int byteCount = fileChannel.write(byteBuffer);
29     System.out.println("file.txt : " + byteCount + " bytes written");
30
31     //FileChannel 닫기
32     fileChannel.close();
33 }
34 }

```

실행 결과

```
file.txt : 15 bytes written
```

파일에서 내용을 읽기 위해서는 FileChannel의 read() 메소드를 호출하면 된다. 매개값으로 ByteBuffer를 주면, 파일에서 읽은 바이트가 position부터 저장된다. read() 메소드의 리턴값은 파일에서 읽은 바이트 수이다. 한 번 읽을 수 있는 최대 바이트 수는 ByteBuffer의 capacity이다. 더 이상 읽을 바이트가 없다면 read() 메소드는 -1을 리턴한다.

```
int byteNum = fileChannel.read(ByteBuffer buffer);
```

read() 메소드로 ByteBuffer에 바이트가 저장될 때마다 position이 1씩 증가하게 된다. 따라서 ByteBuffer에 저장한 마지막 바이트의 위치는 position-1이다. 다음 예제는 이전 예제에서 생성한 C:\Temp\file.txt 파일을 읽고 콘솔에 출력한다.

»» FileChannelReadExample.java

```

1     package sec04.exam01_file_read_write;
2
3     import java.io.IOException;
4     import java.nio.ByteBuffer;
5     import java.nio.channels.FileChannel;
6     import java.nio.charset.Charset;

```



```

7  import java.nio.file.Path;
8  import java.nio.file.Paths;
9  import java.nio.file.StandardOpenOption;
10
11 public class FileChannelReadExample {
12     public static void main(String[] args) throws IOException {
13         //Path 생성
14         Path path = Paths.get("C:/Temp/file.txt");
15
16         //FileChannel 열기
17         FileChannel fileChannel = FileChannel.open(path,
18             StandardOpenOption.READ);
19
20         //읽은 바이트가 저장될 ByteBuffer 생성
21         ByteBuffer byteBuffer = ByteBuffer.allocate(100);
22
23         //FileChannel로부터 입력받기
24         Charset charset = Charset.forName("UTF-8");
25         String data = "";
26         while(true) {
27             int byteNum = fileChannel.read(byteBuffer);
28             if(byteNum == -1) break;
29             byteBuffer.flip();
30             data += charset.decode(byteBuffer).toString();
31             byteBuffer.clear();
32         }
33
34         //FileChannel 닫기
35         fileChannel.close();
36
37         //읽은 내용을 콘솔에 출력
38         System.out.println("file.txt : " + data);
39     }

```

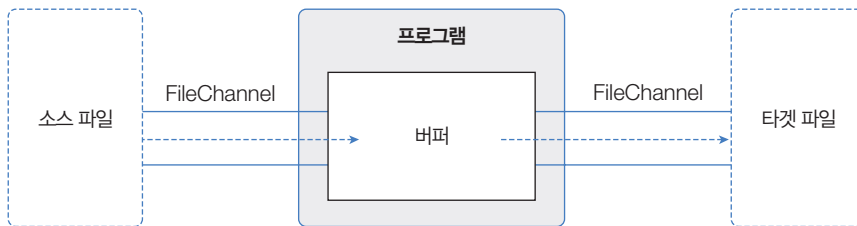
실행 결과

```
file.txt : 안녕하세요
```

28라인에서 flip()을 호출한 이유는 limit을 현재 position으로 설정하고 position을 0으로 설정하기 위해서이다. 29라인은 position에서 limit까지 읽고 문자열로 변환한다. 30라인에서 clear() 메소드는 position을 0으로 limit을 capacity로 설정해서 ByteBuffer를 초기화한다.

파일 복사

파일 복사를 구현하기 위해서는 하나의 ByteBuffer를 사이에 두고 파일 읽기용 FileChannel과 파일 쓰기용 FileChannel이 읽기와 쓰기를 교대로 번갈아 수행하도록 하면 된다.



다음 예제는 FileChannel을 이용해서 이미지 파일을 복사한다. 다이렉트 버퍼를 사용하는데, FileChannel의 입출력 성능을 향상시키기 위해서이다.

>> FileCopyExample.java

```

1  package sec04.exam02_file_copy;
2
3  import java.io.IOException;
4  import java.nio.ByteBuffer;
5  import java.nio.channels.FileChannel;
6  import java.nio.file.Path;
7  import java.nio.file.Paths;
8  import java.nio.file.StandardOpenOption;
9
10 public class FileCopyExample {
11     public static void main(String[] args) throws IOException {
12         //Path 생성
13         Path from = Paths.get("src/sec04/exam02_file_copy/house.jpg");
14         Path to = Paths.get("C:/Temp/house.jpg");
15     }

```

```

16      //입력용 FileChannel 열기
17      FileChannel fileChannel_from = FileChannel.open(
18          from, StandardOpenOption.READ);
19
20      //출력용 FileChannel 열기
21      FileChannel fileChannel_to = FileChannel.open(
22          to, StandardOpenOption.CREATE, StandardOpenOption.WRITE);
23
24      //다이렉트 ByteBuffer를 이용해서 데이터 입출력
25      ByteBuffer buffer = ByteBuffer.allocateDirect(100);
26      while(true) {
27          int byteCount = fileChannel_from.read(buffer);
28          if(byteCount == -1) break;
29          buffer.flip();
30          fileChannel_to.write(buffer);
31          buffer.clear();
32      }
33
34      //FileChannel 닫기
35      fileChannel_from.close();
36      fileChannel_to.close();
37      System.out.println("파일 복사 성공");
38  }
39  }

```

실행 결과

파일 복사 성공

이번 예제처럼 ByteBuffer와 FileChannel 2개를 직접 생성해서 복사를 구현해도 좋지만, 단순히 파일을 복사할 목적이라면 NIO의 Files 클래스의 copy() 메소드를 사용하는 것이 더 편리하다.

```
Files.copy(Path source, Path target, CopyOption... options);
```

첫 번째 source 매개값에는 원본 파일의 Path 객체를 지정하고 두 번째 target 매개값에는 타겟 파일의 Path 객체를 지정하면 된다. 세 번째 매개값은 다음 세 가지 StandardCopyOption 열거 상수를 목적에 맞게 나열해주면 된다.

열거 상수	설명
REPLACE_EXISTING	타겟 파일이 존재하면 대체한다.
COPY_ATTRIBUTES	파일 속성까지도 복사한다.
NOFOLLOW_LINKS	링크 파일일 경우 링크 파일만 복사하고 링크된 파일은 복사하지 않는다.

다음 예제는 Files 클래스의 copy() 메소드를 이용해서 이미지 파일을 복사한다.

»» FilesCopyMethodExample.java

```

1  package sec04.exam02_file_copy;
2
3  import java.io.IOException;
4  import java.nio.file.Files;
5  import java.nio.file.Path;
6  import java.nio.file.Paths;
7  import java.nio.file.StandardCopyOption;
8
9  public class FilesCopyMethodExample {
10     public static void main(String[] args) throws IOException {
11         Path from = Paths.get("src/sec04/exam02_file_copy/house.jpg");
12         Path to = Paths.get("C:/Temp/house.jpg");
13
14         Files.copy(from, to, StandardCopyOption.REPLACE_EXISTING);
15         System.out.println("파일 복사 성공");
16     }
17 }

```

실행 결과

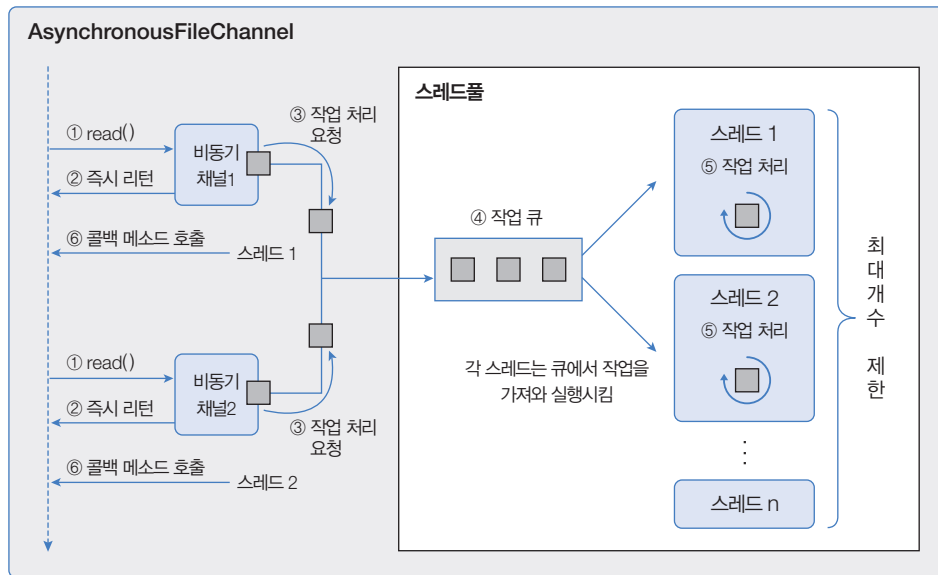
파일 복사 성공

05 파일 비동기 입출력

FileChannel의 read()와 write() 메소드는 파일 입출력 작업 동안 블로킹된다. UI 및 이벤트를 처리하는 스레드에서 이 메소드들을 호출하면 블로킹되는 동안에 UI 갱신이나 이벤트 처리를 할 수 없다. 따라서 별도의 작업 스레드를 생성해서 이 메소드들을 호출해야 한다.

NIO는 스레드풀을 이용해서 동시에 여러 개의 파일을 입출력할 수 있도록 비동기 파일 채널 (AsynchronousFileChannel)을 제공하고 있다. 비동기 파일 채널의 특징은 read()와 write() 메소드를 호출하면 스레드풀에게 입출력 처리를 요청하고 즉시 리턴된다는 점이다.

입출력 처리는 스레드풀의 작업 스레드가 담당하는데, 작업 스레드가 입출력을 완료하게 되면 콜백 callback 메소드가 자동 호출된다. 따라서 입출력 완료 후 실행해야 할 코드가 있다면 콜백 메소드에 작성하면 된다.



파일 비동기 채널

AsynchronousFileChannel은 두 가지 정적 메소드인 open()을 호출해서 얻을 수 있다. 첫 번째 open() 메소드는 다음과 같이 파일의 Path와 열기 옵션을 매개값으로 받는다.

```
AsynchronousFileChannel fileChannel = AsynchronousFileChannel.open(
    Path file,
    OpenOption... options
);
```

이렇게 생성된 `AsynchronousFileChannel`은 내부적으로 생성되는 기본 스레드풀을 이용해서 스레드를 관리한다. 기본 스레드풀의 최대 스레드 개수는 개발자가 지정할 수 없기 때문에 다음과 같이 두 번째 `open()` 메소드로 `AsynchronousFileChannel`을 만들 수도 있다.

```
AsynchronousFileChannel fileChannel = AsynchronousFileChannel.open(
    Path file,
    Set<? extends OpenOption> options,
    ExecutorService executor,
    FileAttribute<?>... attrs
);
```

`file` 매개값은 파일의 `Path`이고, `options` 매개값은 열기 옵션이 저장된 `Set`이다. `executor` 매개값은 스레드풀인 `ExecutorService`이다. `attrs` 매개값은 파일 생성 시 파일 속성이 될 `FileAttribute`를 나열하면 된다.

예로 'C:\Temp\file.txt' 파일을 입출력할 수 있는 `AsynchronousFileChannel`은 다음과 같이 생성할 수 있다.

```
ExecutorService executorService = Executors.newFixedThreadPool(
    Runtime.getRuntime().availableProcessors()
);

AsynchronousFileChannel fileChannel = AsynchronousFileChannel.open(
    Paths.get("C:/Temp/file.txt"),
    EnumSet.of(StandardOpenOption.CREATE, StandardOpenOption.WRITE),
    executorService
);
```

`Runtime.getRuntime().availableProcessors()`는 CPU의 코어 수를 리턴한다. 쿼드 코어 CPU 일 경우는 4를 리턴, 하이퍼 스레딩일 경우는 8을 리턴한다. `EnumSet.of()` 메소드는 매개값으로 나열된 열거 상수를 `Set` 객체에 담아 리턴한다. `AsynchronousFileChannel`을 더 이상 사용하지 않을 경우에는 다음과 같이 `close()` 메소드를 호출해서 닫아준다.

```
fileChannel.close();
```

파일 입출력

AsynchronousFileChannel이 생성되었다면 read(), write() 메소드를 이용해서 입출력할 수 있다.

```
read(  
    ByteBuffer dst,  
    long position,  
    A attachment,  
    CompletionHandler<Integer, A> handler  
);
```

```
write(  
    ByteBuffer src,  
    long position,  
    A attachment,  
    CompletionHandler<Integer, A> handler  
);
```

이 메소드들을 호출하면 즉시 리턴되고, 스레드풀의 스레드가 입출력 작업을 진행한다. dst와 src 매개값은 읽거나 쓰기 위한 ByteBuffer이고, position 매개값은 파일에서 읽을 위치이거나 쓸 위치이다. 파일의 첫 번째 바이트부터 읽거나 첫 번째 위치에 바이트를 쓰고 싶다면 position을 0으로 주면 된다.

attachment 매개값은 입출력 연산에 사용될 수 있는 첨부 객체이다. 첨부 객체는 콜백 메소드에서도 사용할 수 있는데, 주로 입출력 후의 정보를 얻고자 할 때 사용된다. 만약 첨부 객체가 필요 없다면 null을 대입해도 된다.

handler 매개값은 CompletionHandler<Integer, A> 구현 객체이다. Integer는 입출력 작업의 결과 타입으로, read()와 write()가 읽거나 쓴 바이트 수이다. A는 첨부 객체 타입으로 개발자가 CompletionHandler 구현 클래스를 작성할 때 임의로 지정이 가능하다. attachment가 null일 경우, A는 Void로 해야 한다.

CompletionHandler<Integer, A> 구현 객체는 비동기 작업이 정상적으로 완료된 경우와 예외 발생으로 실패된 경우에 자동으로 콜백되는 다음 두 가지 메소드를 재정의해야 한다.

리턴 타입	메소드명(매개변수)	설명
void	completed(Integer result, A attachment)	작업이 정상적으로 완료된 경우 콜백
void	failed(Throwable exc, A attachment)	예외 때문에 작업이 실패된 경우 콜백

completed() 메소드의 result 매개값은 작업 결과가 대입되는데, read()와 write() 작업 결과는 읽거나 쓴 바이트 수이다. attachment 매개값은 read()와 write() 호출 시 제공된 첨부 객체이다.

failed() 메소드의 exc 매개값은 작업 처리 도중 발생한 예외이다. 주목할 점은 콜백 메소드를 실행하는 스레드는 read()와 write()를 호출한 스레드가 아니고 스레드풀의 작업 스레드라는 것이다. 그렇기 때문에 JavaFX 또는 Swing 애플리케이션에서 UI 생성 및 변경 작업을 이 메소드에서 직접 할 수 없고 Platform.runLater() 또는 SwingUtilities.invokeLater()를 사용해야 한다.

다음은 CompletionHandler 구현 클래스를 작성하는 방법을 보여준다.

```
new CompletionHandler<Integer, A>() {
    @Override
    public void completed(Integer result, A attachment) { ... }

    @Override
    public void failed(Throwable exc, A attachment) { ... }
};
```

다음은 AsynchronousFileChannel을 이용해서 비동기적으로 'C:\Temp' 디렉토리에 file0.txt ~ file9.txt까지 총 10개의 파일을 생성한 후 '안녕하세요'라는 내용을 쓴다. 그리고 비동기 작업이 완료되었을 때 출력된 바이트 수와 처리를 담당했던 스레드 이름을 콘솔에 출력한다.

» AsynchronousFileChannelWriteExample.java

```
1 package sec05;
2
3 import java.io.IOException;
4 import java.nio.ByteBuffer;
5 import java.nio.channels.AsynchronousFileChannel;
```



```

6  import java.nio.channels.CompletionHandler;
7  import java.nio.charset.Charset;
8  import java.nio.file.Files;
9  import java.nio.file.Path;
10 import java.nio.file.Paths;
11 import java.nio.file.StandardOpenOption;
12 import java.util.EnumSet;
13 import java.util.concurrent.ExecutorService;
14 import java.util.concurrent.Executors;
15
16 public class AsynchronousFileChannelWriteExample {
17     public static void main(String[] args) throws Exception {
18         //스레드풀 생성
19         ExecutorService executorService = Executors.newFixedThreadPool(3);
20
21         for (int i = 0; i < 10; i++) {
22             Path path = Paths.get("C:/Temp/file" + i + ".txt");
23             Files.createDirectories(path.getParent());
24
25             //비동기 파일 채널 생성
26             AsynchronousFileChannel fileChannel = AsynchronousFileChannel.open(
27                 path,
28                 EnumSet.of(StandardOpenOption.CREATE, StandardOpenOption.WRITE),
29                 executorService
30             );
31
32             Charset charset = Charset.defaultCharset();
33             ByteBuffer byteBuffer = charset.encode("안녕하세요");
34
35             //첨부 객체 생성
36             class Attachment {
37                 Path path;
38                 AsynchronousFileChannel fileChannel;
39             }
40             Attachment attachment = new Attachment();
41             attachment.path = path;
42             attachment.fileChannel = fileChannel;
43
44             //CompletionHandler 객체 생성
45             CompletionHandler<Integer, Attachment> completionHandler =

```

```

46         new CompletionHandler<Integer, Attachment>() {
47             @Override
48             public void completed(Integer result, Attachment attachment) {
49                 System.out.println(
50                     attachment.path.getFileName() + " : " +
51                     result + " bytes written : " +
52                     Thread.currentThread().getName());
53                 try {
54                     attachment.fileChannel.close();
55                 } catch (IOException e) {
56                 }
57             }
58
59             @Override
60             public void failed(Throwable exc, Attachment attachment) {
61                 exc.printStackTrace();
62                 try {
63                     attachment.fileChannel.close();
64                 } catch (IOException e) {
65                 }
66             }
67         };
68
69         //ByteBuffer에 있는 내용을 파일에 출력
70         fileChannel.write(byteBuffer, 0, attachment, completionHandler);
71     }
72
73     //스레드풀 종료
74     executorService.shutdown();
75 }
76 }

```

실행 결과

```

file0.txt : 15 bytes written : pool-1-thread-1
file2.txt : 15 bytes written : pool-1-thread-3
file1.txt : 15 bytes written : pool-1-thread-2
file3.txt : 15 bytes written : pool-1-thread-3
file4.txt : 15 bytes written : pool-1-thread-1
file5.txt : 15 bytes written : pool-1-thread-3
file6.txt : 15 bytes written : pool-1-thread-2

```

```
file7.txt : 15 bytes written : pool-1-thread-1
file8.txt : 15 bytes written : pool-1-thread-3
file9.txt : 15 bytes written : pool-1-thread-2
```

이 예제에서 주의할 점은 70라인에서 `write()` 메소드가 즉시 리턴되더라도 뒤에서는 작업 스레드가 파일 쓰기 작업을 하고 있기 때문에 바로 `AsynchronousFileChannel`을 닫으면 안 된다. 작업이 정상적으로 완료되었거나 실패일 경우 채널을 닫아야 하므로 `completed()`와 `failed()` 메소드에서 `AsynchronousFileChannel`의 `close()`를 호출해야 한다.

다음 예제는 이전 예제에서 생성한 `file0.txt ~ file9.txt`를 읽고 콘솔에 출력한다.

>>> `AsynchronousFileChannelReadExample.java`

```
1  package sec05;
2
3  import java.io.IOException;
4  import java.nio.ByteBuffer;
5  import java.nio.channels.AsynchronousFileChannel;
6  import java.nio.channels.CompletionHandler;
7  import java.nio.charset.Charset;
8  import java.nio.file.Path;
9  import java.nio.file.Paths;
10 import java.nio.file.StandardOpenOption;
11 import java.util.EnumSet;
12 import java.util.concurrent.ExecutorService;
13 import java.util.concurrent.Executors;
14
15 public class AsynchronousFileChannelReadExample {
16     public static void main(String[] args) throws Exception {
17         //스레드풀 생성
18         ExecutorService executorService = Executors.newFixedThreadPool(3);
19
20         for (int i = 0; i < 10; i++) {
21             Path path = Paths.get("C:/Temp/file" + i + ".txt");
22
23             //비동기 파일 채널 생성
24             AsynchronousFileChannel fileChannel = AsynchronousFileChannel.
25                 open(path,
26                     EnumSet.of(StandardOpenOption.READ), executorService);
```

```

26
27     ByteBuffer byteBuffer = ByteBuffer.allocate((int) fileChannel.size());
28
29     //첨부 객체 생성
30     class Attachment {
31         Path path;
32         AsynchronousFileChannel fileChannel;
33         ByteBuffer byteBuffer;
34     }
35     Attachment attachment = new Attachment();
36     attachment.path = path;
37     attachment.fileChannel = fileChannel;
38     attachment.byteBuffer = byteBuffer;
39
40     //CompletionHandler 객체 생성
41     CompletionHandler<Integer, Attachment> completionHandlernew =
42         new CompletionHandler<Integer, Attachment>() {
43             @Override
44             public void completed(Integer result, Attachment attachment) {
45                 attachment.byteBuffer.flip();
46
47                 Charset charset = Charset.defaultCharset();
48                 String data = charset.decode(attachment.byteBuffer).toString();
49
50                 System.out.println(
51                     attachment.path.getFileName() + " : " +
52                     data + " : " +
53                     Thread.currentThread().getName());
54                 try {
55                     fileChannel.close();
56                 } catch (IOException e) {
57                     //e.printStackTrace();
58                 }
59             }
60
61             @Override
62             public void failed(Throwable exc, Attachment attachment) {
63                 exc.printStackTrace();
64                 try {
65                     fileChannel.close();

```

```

66         } catch (IOException e) {
67         }
68     }
69 };
70
71     //파일을 읽고 ByteBuffer에 저장
72     fileChannel.read(byteBuffer, 0, attachment, completionHandlernew);
73 }
74
75     //스레드풀 종료
76     executorService.shutdown();
77 }
78 }

```

실행 결과

```

file1.txt : 안녕하세요 : pool-1-thread-2
file3.txt : 안녕하세요 : pool-1-thread-2
file4.txt : 안녕하세요 : pool-1-thread-2
file0.txt : 안녕하세요 : pool-1-thread-1
file2.txt : 안녕하세요 : pool-1-thread-3
file5.txt : 안녕하세요 : pool-1-thread-2
file6.txt : 안녕하세요 : pool-1-thread-1
file8.txt : 안녕하세요 : pool-1-thread-2
file7.txt : 안녕하세요 : pool-1-thread-3

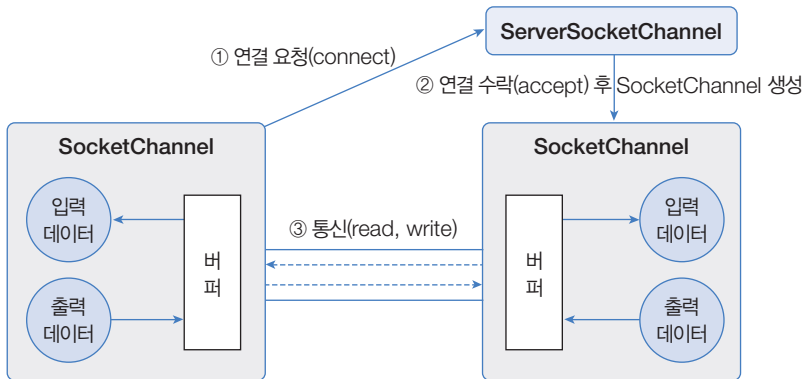
```

이 예제에서도 72라인의 `read()` 메소드가 즉시 리턴되더라도 뒤에서는 작업 스레드가 파일 읽기 작업을 하고 있기 때문에 바로 `AsynchronousFileChannel`을 닫으면 안 된다. 작업이 정상적으로 완료되었거나 실패일 경우 채널을 닫아야 하므로 `completed()`와 `failed()` 메소드에서 `AsynchronousFileChannel`의 `close()`를 호출해야 한다.

06 TCP 네트워크 입출력

NIO를 이용해서 TCP 서버/클라이언트 애플리케이션을 개발하는 방법을 알아보자. TCP를 사용해서 네트워크 통신을 하려면 서버 소켓 채널(`ServerSocketChannel`)과 소켓 채널(`SocketChannel`)을 알아야 한다.

ServerSocketChannel과 SocketChannel은 각각 IO의 ServerSocket과 Socket에 대응되는 클래스이다. IO는 버퍼를 사용하지 않고 입출력한다면 이들 클래스는 버퍼를 이용해서 입출력한다. 사용 방법은 IO와 큰 차이점이 없는데, 다음 그림처럼 ServerSocketChannel은 연결 요청을 수락하고 통신용 SocketChannel을 생성한다



서버 소켓 채널

TCP 서버를 개발하려면 먼저 ServerSocketChannel을 생성해야 한다. ServerSocketChannel은 정적 메소드인 `open()`으로 생성한다. 그리고 포트(port)에 바인딩하기 위해 `InetSocketAddress` 객체를 매개값으로 해서 `bind()` 메소드를 호출한다.

```
ServerSocketChannel serverSocketChannel = ServerSocketChannel.open();
serverSocketChannel.bind(new InetSocketAddress(50001));
```

포트 바인딩까지 끝났다면 클라이언트 연결 수락을 위해 `accept()` 메소드를 실행한다. `accept()` 메소드는 연결 요청이 들어오기 전까지 대기(블로킹)되고, 연결 요청이 들어오면 클라이언트와 통신할 SocketChannel을 만들고 리턴한다.

```
SocketChannel socketChannel = serverSocketChannel.accept();
```

연결된 클라이언트의 IP와 포트 정보를 알고 싶다면 SocketChannel의 getRemoteAddress() 메소드를 호출해서 SocketAddress를 얻으면 된다. 실제 리턴되는 것은 InetSocketAddress 인스턴스이므로 다음과 같이 타입 변환할 수 있다.

```
InetSocketAddress socketAddress = (InetSocketAddress) socketChannel.  
getRemoteAddress();
```

InetSocketAddress에는 다음과 같이 IP와 포트 정보를 리턴하는 메소드들이 있다.

리턴 타입	메소드명(매개변수)	설명
String	getHostName()	클라이언트 IP 리턴
int	getPort()	클라이언트 포트 번호 리턴
String	toString()	"IP:포트번호" 형태의 문자열 리턴

더 이상 클라이언트를 위해 연결 수락이 필요 없다면 ServerSocketChannel의 close() 메소드를 호출해서 포트를 언바인딩시켜야 한다. 그래야 다른 프로그램에서 해당 포트를 재사용할 수 있다.

```
serverSocketChannel.close();
```

다음 예제는 반복적으로 accept() 메소드를 호출해서 복수의 클라이언트 연결을 수락하는 가장 기본적인 TCP 서버 코드를 보여준다.

>>> ServerExample.java

```
1  package sec06.exam01_tcpchannel;  
2  
3  import java.io.IOException;  
4  import java.net.InetSocketAddress;  
5  import java.nio.channels.ServerSocketChannel;  
6  import java.nio.channels.SocketChannel;  
7  
8  public class ServerExample {  
9      public static void main(String[] args) {
```

```

10      //ServerSocketChannel 변수 선언
11      ServerSocketChannel serverSocketChannel = null;
12
13      try {
14          //ServerSocketChannel 열기
15          serverSocketChannel = ServerSocketChannel.open();
16
17          //ServerSocketChannel 포트 바인딩
18          serverSocketChannel.bind(new InetSocketAddress(50001));
19
20          System.out.println("[서버 시작]");
21
22          //클라이언트의 연결 요청을 수락
23          while (true) {
24              SocketChannel socketChannel = serverSocketChannel.accept();
25              InetSocketAddress isa = (InetSocketAddress) socketChannel.
                getRemoteAddress();
26              System.out.println(isa.getHostName() + " 연결 수락");
27              //연결 끊기
28              System.out.println(isa.getHostName() + " 연결 끊기");
29              socketChannel.close();
30          }
31      } catch (Exception e) {
32          e.printStackTrace();
33      } finally {
34          //ServerSocketChannel 닫기
35          try {
36              serverSocketChannel.close();
37          } catch (IOException e1) {}
38      }
39  }
40  }

```

실행 결과

[서버 시작]

[연결 기다림]만 출력되고, 클라이언트의 연결 요청이 있을 때까지 accept() 메소드에서 블로킹(대기) 상태가 된다. 실행을 종료하지 말고, 이어서 나오는 소켓 채널 생성을 학습하고 클라이언트 연결 요청 예제를 실행시켜보자.

소켓 채널

TCP 클라이언트를 개발하려면 먼저 `SocketChannel`을 생성해야 한다. `SocketChannel`은 정적 메소드인 `open()`으로 생성한다. 서버 연결 요청은 `connect()` 메소드를 호출하면 되는데, 서버 IP와 포트 정보를 가진 `InetSocketAddress` 객체를 매개값으로 주면 된다.

`connect()` 메소드는 연결이 완료될 때까지 블로킹(대기) 상태가 되고, 연결이 완료되면 리턴된다. 다음은 로컬 PC의 50001 포트에 바인딩된 서버로 연결을 요청하는 코드이다.

```
SocketChannel socketChannel = SocketChannel.open();
socketChannel.connect(new InetSocketAddress("localhost", 50001));
```

`connect()` 메소드는 서버와 연결이 될 때까지 블로킹된다. 연결된 후, 클라이언트 프로그램을 종료하거나 필요에 따라서 연결을 끊고 싶다면 다음과 같이 `SocketChannel`의 `close()` 메소드를 호출하면 된다.

```
socketChannel.close();
```

다음 예제는 이전 예제에서 실행 중인 TCP 서버로 연결 요청하는 코드이다. `connect()` 메소드가 정상적으로 리턴되면 연결 성공한 것이다.

» ClientExample.java

```
1  package sec06.exam01_tcpchannel;
2
3  import java.io.IOException;
4  import java.net.InetSocketAddress;
5  import java.nio.channels.SocketChannel;
6
7  public class ClientExample {
8      public static void main(String[] args) {
9          //SocketChannel 변수 선언
10         SocketChannel socketChannel = null;
11     }
```

```

12     try {
13         //SocketChannel 열기
14         socketChannel = SocketChannel.open();
15
16         //로컬 PC의 50001에서 실행 중인 ServerSocketChannel로 연결 요청
17         System.out.println("[연결 요청]");
18         socketChannel.connect(new InetSocketAddress("localhost", 50001));
19         System.out.println("[연결 성공]");
20     } catch (Exception e) {
21         e.printStackTrace();
22     } finally {
23         //SocketChannel 닫기
24         try {
25             System.out.println("[연결 끊기]");
26             socketChannel.close();
27         } catch (IOException e1) {}
28     }
29 }
30 }

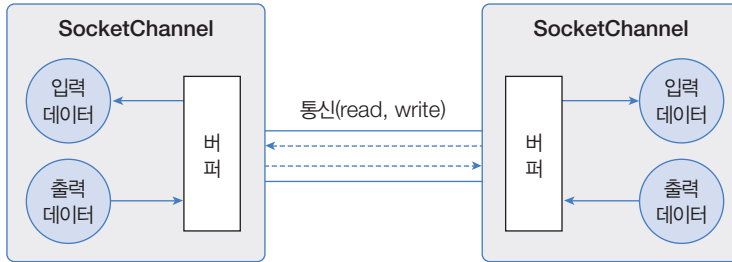
```

실행 결과

ClientExample.java	ServerExample.java
[연결 요청]	[서버 시작]
[연결 성공]	127.0.0.1 연결 수락
[연결 끊기]	127.0.0.1 연결 끊기

네트워크 입출력

클라이언트가 연결 요청(connect())하고 서버가 연결 수락(accept())했다면, 양쪽 SocketChannel 객체의 read(), write() 메소드를 호출해서 네트워크 입출력을 할 수 있다. 이 메소드들은 모두 버퍼를 이용한다.



다음은 SocketChannel의 write() 메소드를 이용해서 문자열을 보내는 코드이다.

```

Charset charset = Charset.forName("UTF-8");
ByteBuffer byteBuffer = charset.encode("Hello Server");
socketChannel.write(byteBuffer);

```

다음은 SocketChannel의 read() 메소드를 이용해서 문자열을 받는 코드이다.

```

ByteBuffer byteBuffer = ByteBuffer.allocate(100);
int byteNum = socketChannel.read(byteBuffer);
byteBuffer.flip();
Charset charset = Charset.forName("UTF-8");
String message = charset.decode(byteBuffer).toString();

```

다음 예제에서는 연결 성공 후 클라이언트가 'Hello Server'를 서버로 보내면 서버가 응답으로 'Hello Client'를 클라이언트로 보낸다.

>>> ServerExample.java

```

1  package sec06.exam02_data_read_write;
2
3  import java.io.IOException;
4  import java.net.InetSocketAddress;
5  import java.nio.ByteBuffer;
6  import java.nio.channels.ServerSocketChannel;
7  import java.nio.channels.SocketChannel;
8  import java.nio.charset.Charset;

```

```

9
10 public class ServerExample {
11     public static void main(String[] args) {
12         //ServerSocketChannel 변수 선언
13         ServerSocketChannel serverSocketChannel = null;
14
15         try {
16             //ServerSocketChannel 열기
17             serverSocketChannel = ServerSocketChannel.open();
18
19             //ServerSocketChannel 포트 바인딩
20             serverSocketChannel.bind(new InetSocketAddress(50001));
21
22             System.out.println("[서버 시작]");
23
24             while (true) {
25                 //클라이언트의 연결 요청을 수락
26                 SocketChannel socketChannel = serverSocketChannel.accept();
27                 InetSocketAddress isa = (InetSocketAddress)
28                     socketChannel.getRemoteAddress();
29                 System.out.println(isa.getHostName() + " 연결 수락");
30
31                 ByteBuffer byteBuffer = null;
32                 Charset charset = Charset.forName("UTF-8");
33
34                 //클라이언트가 보낸 데이터 받기
35                 byteBuffer = ByteBuffer.allocate(100);
36                 int byteNum = socketChannel.read(byteBuffer);
37                 byteBuffer.flip();
38                 String message = charset.decode(byteBuffer).toString();
39                 System.out.println(isa.getHostName() + " 데이터 받기: " + message);
40
41                 //클라이언트로 데이터 보내기
42                 byteBuffer = charset.encode("Hello Client");
43                 socketChannel.write(byteBuffer);
44                 System.out.println(isa.getHostName() + " 데이터 보냄");
45
46                 //연결 끊기
47                 System.out.println(isa.getHostName() + " 연결 끊기");
48                 socketChannel.close();

```

```

48     }
49     } catch (Exception e) {
50         e.printStackTrace();
51     } finally {
52         //ServerSocketChannel 닫기
53         try {
54             serverSocketChannel.close();
55         } catch (IOException e1) {
56         }
57     }
58 }
59 }

```

실행 결과

[서버 시작]

>>> ClientExample.java

```

1  package sec06.exam02_data_read_write;
2
3  import java.io.IOException;
4  import java.net.InetSocketAddress;
5  import java.nio.ByteBuffer;
6  import java.nio.channels.SocketChannel;
7  import java.nio.charset.Charset;
8
9  public class ClientExample {
10     public static void main(String[] args) {
11         //SocketChannel 변수 선언
12         SocketChannel socketChannel = null;
13
14         try {
15             //SocketChannel 열기
16             socketChannel = SocketChannel.open();
17
18             //로컬 PC의 50001에서 실행 중인 ServerSocketChannel로 연결 요청
19             System.out.println("[연결 요청]");

```

```

20     socketChannel.connect(new InetSocketAddress("localhost", 50001));
21     System.out.println("[연결 성공]");
22
23     ByteBuffer byteBuffer = null;
24     Charset charset = Charset.forName("UTF-8");
25
26     //서버로 데이터 보내기
27     byteBuffer = charset.encode("Hello Server");
28     socketChannel.write(byteBuffer);
29     System.out.println("서버로 데이터 보냄");
30
31     //서버가 보낸 데이터 받기
32     byteBuffer = ByteBuffer.allocate(100);
33     int byteNum = socketChannel.read(byteBuffer);
34     byteBuffer.flip();
35     String message = charset.decode(byteBuffer).toString();
36     System.out.println("서버에서 데이터 받기: " + message);
37 } catch (Exception e) {
38     e.printStackTrace();
39 } finally {
40     //SocketChannel 닫기
41     try {
42         System.out.println("[연결 끊기]");
43         socketChannel.close();
44     } catch (IOException e1) {
45     }
46 }
47 }
48 }

```

실행 결과

ClientExample.java	ServerExample.java
[연결 요청]	[서버 시작]
[연결 성공]	127.0.0.1 연결 수락
데이터 보냄	127.0.0.1 데이터 받기: Hello Server
데이터 받기: Hello Client	127.0.0.1 데이터 보냄
[연결 끊기]	127.0.0.1 연결 끊기

데이터를 받기 위해 `read()` 메소드를 호출하면 상대방이 데이터를 보내기 전까지는 블로킹(대기) 상태가 된다. `read()` 메소드가 블로킹 상태에서 해제되고 리턴되는 경우는 다음 세 가지이다.

블로킹이 해제되는 경우	리턴값
상대방이 데이터를 보냄	읽은 바이트 수
상대방이 정상적으로 <code>SocketChannel</code> 의 <code>close()</code> 를 호출	-1
상대방이 비정상적으로 종료	<code>IOException</code> 발생

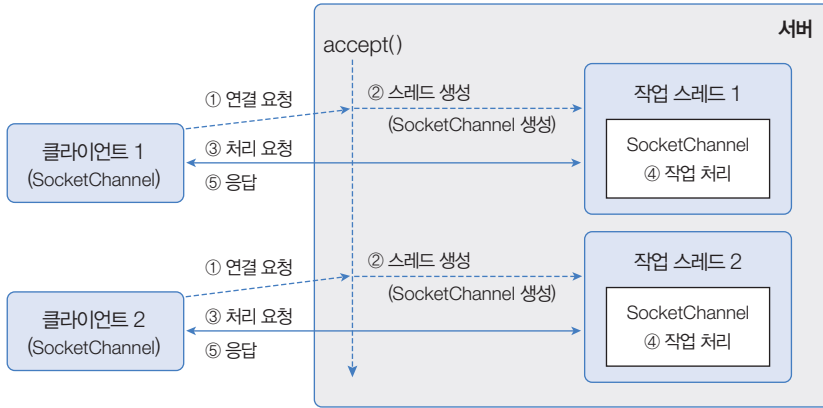
상대방이 정상적으로 `SocketChannel`의 `close()`를 호출하고 연결을 끊었을 경우와 상대방이 비정상적으로 종료된 경우는 예외 처리를 해서 `SocketChannel`을 닫기 위해 `close()` 메소드를 호출하는 것이 좋다.

```
try {
    ...
    //상대방이 비정상적으로 종료했을 경우 IOException 발생
    int byteNum = socketChannel.read(byteBuffer);

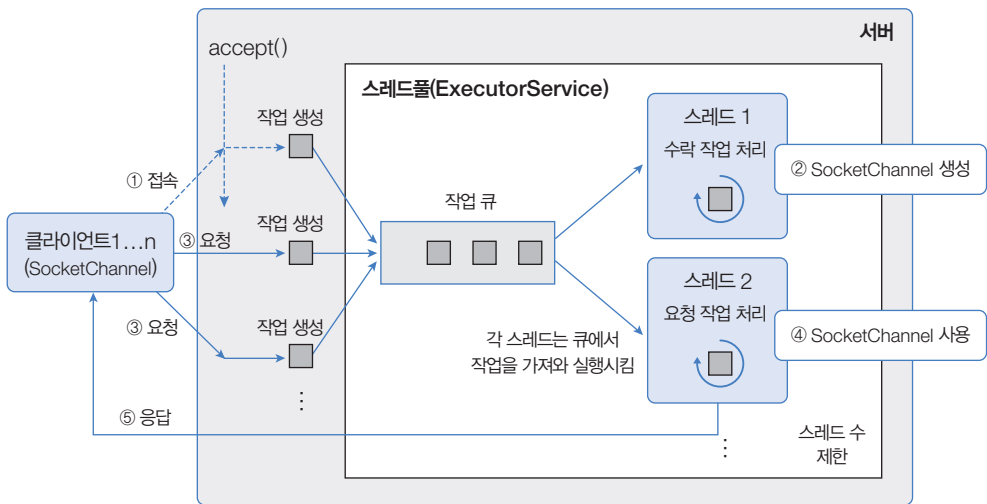
    //상대방이 정상적으로 SocketChannel의 close()를 호출했을 경우
    if(byteNum == -1) {
        throw new IOException(); //강제로 IOException 발생시킴
    }
    ...
} catch (Exception e) {
    ...
} finally {
    //연결 끊기
    try { socketChannel.close(); } catch (Exception e2) { }
}
```

스레드 병렬 처리

TCP 채널을 이용할 경우 데이터 입출력이 완료되기 전까지 `read()`와 `write()` 메소드는 블로킹된다. 따라서 서버가 동시에 여러 클라이언트와 통신을 하기 위해서는 멀티 스레드를 이용해서 병렬 처리해야 한다.



위 그림과 같이 클라이언트별로 스레드를 생성해서 병렬 처리를 한다면 클라이언트의 폭증이 있을 때 과도한 스레드가 생성되어 서버 성능이 급격히 저하된다. 좋은 해결 방법은 서버에서 스레드풀을 사용해서 병렬 처리하는 것이다.



스레드풀을 이용할 경우 클라이언트의 폭증은 작업 큐의 작업량만 증가시킬 뿐, 전체 스레드의 수에는 변함이 없기 때문에 서버 성능은 완만히 저하된다. 다만 대기하는 작업량이 증가하기 때문에 개별 클라이언트에서 응답을 늦게 받을 수 있다.

다음 예제는 100개의 클라이언트가 동시에 서버로 데이터를 보내고 받는 예제이다. 서버는 모든 클라이언트의 요청을 처리하기 위해 단 10개의 스레드만 사용한다.

>>> ServerExample.java

```
1  package sec06.exam03_threadpool;
2
3  import java.io.IOException;
4  import java.net.InetSocketAddress;
5  import java.nio.ByteBuffer;
6  import java.nio.channels.ServerSocketChannel;
7  import java.nio.channels.SocketChannel;
8  import java.nio.charset.Charset;
9  import java.util.concurrent.ExecutorService;
10 import java.util.concurrent.Executors;
11
12 public class ServerExample {
13     public static void main(String[] args) {
14         try {
15             //스레드풀 생성
16             ExecutorService executorService = Executors.newFixedThreadPool(10);
17
18             //ServerSocketChannel 열기
19             ServerSocketChannel serverSocketChannel = ServerSocketChannel.open();
20
21             //ServerSocketChannel 포트 바인딩
22             serverSocketChannel.bind(new InetSocketAddress(50001));
23
24             System.out.println("[서버 시작]");
25
26             //스레드풀 작업 처리
27             executorService.execute(() -> {
28                 try {
29                     //지속적인 클라이언트 연결 요청 수락
30                     while (true) {
31                         SocketChannel socketChannel = serverSocketChannel.accept();
32                         System.out.println();
33                         InetSocketAddress isa =
34                             (InetSocketAddress) socketChannel.getRemoteAddress();
35                         System.out.println(isa.getHostName() + " 연결 수락");
36
37                         //스레드풀 작업 처리
38                         executorService.execute(() -> {
```

```

39         //작업 스레드 이름 얻기
40         String threadName = Thread.currentThread().getName();
41
42         try {
43             Charset charset = Charset.forName("UTF-8");
44
45             //클라이언트가 보낸 데이터 받기
46             ByteBuffer byteBuffer = ByteBuffer.allocate(100);
47             int byteNum = socketChannel.read(byteBuffer);
48             if (byteNum == -1) {
49                 throw new IOException();
50             }
51             byteBuffer.flip();
52             String message = charset.decode(byteBuffer).toString();
53             System.out.println "[" + threadName + "]" +
54                 isa.getHostName() + " 데이터 받기: " + message);
55
56             //클라이언트로 데이터 보내기
57             byteBuffer = charset.encode("Hello Client");
58             socketChannel.write(byteBuffer);
59             System.out.println "[" + threadName + "]" +
60                 isa.getHostName() + " 데이터 보냄");
61         } catch (Exception e) {
62         } finally {
63             try {
64                 //연결 끊기
65                 System.out.println "[" + threadName + "]" +
66                     isa.getHostName() + " 연결 끊기");
67                 socketChannel.close();
68             } catch (Exception e) {
69             }
70         }
71     });
72 }
73 } catch (Exception e) {
74     e.printStackTrace();
75 } finally {
76     try {
77         //ServerSocketChannel 닫기

```

```

78         serverSocketChannel.close();
79         //스레드풀 종료
80         executorService.shutdown();
81     } catch (Exception e) {
82     }
83 }
84 });
85 } catch (Exception e) {
86     e.printStackTrace();
87 }
88 }
89 }

```

실행 결과

[서버 시작]

>>> ClientExample.java

```

1  package sec06.exam03_threadpool;
2
3  import java.io.IOException;
4  import java.net.InetSocketAddress;
5  import java.nio.ByteBuffer;
6  import java.nio.channels.SocketChannel;
7  import java.nio.charset.Charset;
8
9  public class ClientExample {
10     public static void main(String[] args) {
11         for(int i=1; i<=100; i++) {
12             //SocketChannel 변수 선언
13             SocketChannel socketChannel = null;
14
15             try {
16                 //SocketChannel 열기
17                 socketChannel = SocketChannel.open();
18
19                 //로컬 PC의 50001에서 실행 중인 ServerSocketChannel로 연결 요청

```

```

20         System.out.println("[연결 요청]");
21         socketChannel.connect(new InetSocketAddress("localhost", 50001));
22         System.out.println("[연결 성공]");
23
24         ByteBuffer byteBuffer = null;
25         Charset charset = Charset.forName("UTF-8");
26
27         //서버로 데이터 보내기
28         byteBuffer = charset.encode("Hello Server " + i);
29         socketChannel.write(byteBuffer);
30         System.out.println(i + "번째 데이터 보냄");
31
32         //서버가 보낸 데이터 받기
33         byteBuffer = ByteBuffer.allocate(100);
34         int byteNum = socketChannel.read(byteBuffer);
35         if(byteNum == -1) {
36             throw new IOException();
37         }
38         byteBuffer.flip();
39         String message = charset.decode(byteBuffer).toString();
40         System.out.println(i + "번째 데이터 받기: " + message);
41     } catch (Exception e) {
42         e.printStackTrace();
43     } finally {
44         //SocketChannel 닫기
45         try {
46             System.out.println("[연결 끊기]");
47             socketChannel.close();
48         } catch (IOException e1) {
49             }
50     }
51     System.out.println();
52 }
53 }
54 }

```

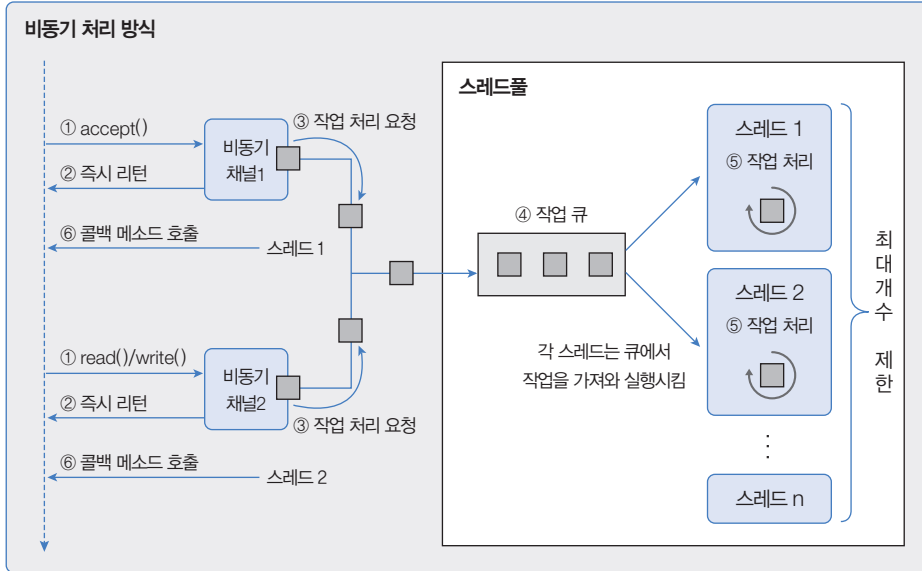
ClientExample.java	ServerExample.java
[연결 요청]	[서버 시작]
[연결 성공]	
1번째 데이터 보냄	127.0.0.1 연결 수락
1번째 데이터 받기: Hello Client	[pool-1-thread-2]127.0.0.1 데이터 받기: Hello Server 1
[연결 끊기]	[pool-1-thread-2]127.0.0.1 데이터 보냄
	[pool-1-thread-2]127.0.0.1 연결 끊기
[연결 요청]	
[연결 성공]	127.0.0.1 연결 수락
2번째 데이터 보냄	[pool-1-thread-3]127.0.0.1 데이터 받기: Hello Server 2
2번째 데이터 받기: Hello Client	[pool-1-thread-3]127.0.0.1 데이터 보냄
[연결 끊기]	[pool-1-thread-3]127.0.0.1 연결 끊기
[연결 요청]	127.0.0.1 연결 수락
[연결 성공]	[pool-1-thread-4]127.0.0.1 데이터 받기: Hello Server 3
3번째 데이터 보냄	[pool-1-thread-4]127.0.0.1 데이터 보냄
3번째 데이터 받기: Hello Client	[pool-1-thread-4]127.0.0.1 연결 끊기
[연결 끊기]	
...	...

서버 실행 결과를 보면 100개의 클라이언트가 보내는 데이터를 받고 다시 보내는 작업을 처리하기 위해 단 10개의 스레드(pool-1-thread1 ~ pool-1-thread-10)만 사용하는 것을 볼 수 있다.

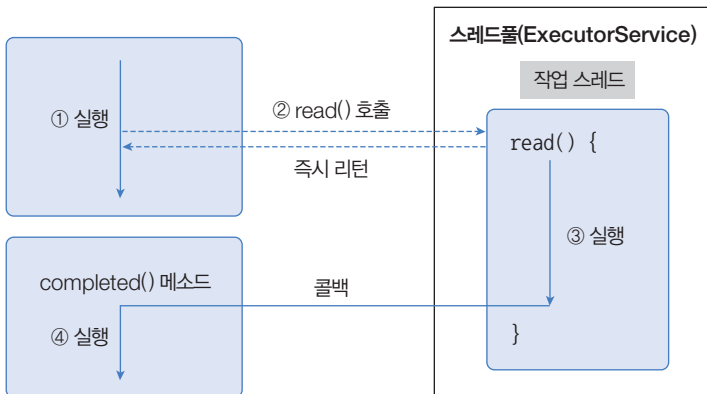
07 TCP 비동기 네트워크 입출력

NIO는 TCP 채널 이외에 TCP 비동기 채널(AsynchronousServerSocketChannel과 AsynchronousSocketChannel)도 제공한다. TCP 비동기 채널은 연결 요청(connect()), 연결 수락(accept()), 읽기(read()), 쓰기(write())를 호출하면 스레드풀에게 작업 처리를 요청하고 즉시 리턴된다.

실질적인 작업 처리는 스레드풀의 작업 스레드가 담당한다. 작업 스레드가 작업을 완료하게 되면 콜백callback 메소드가 자동 호출되기 때문에 작업 완료 후 실행해야 할 코드가 있다면 콜백 메소드에서 작성할 수 있다.

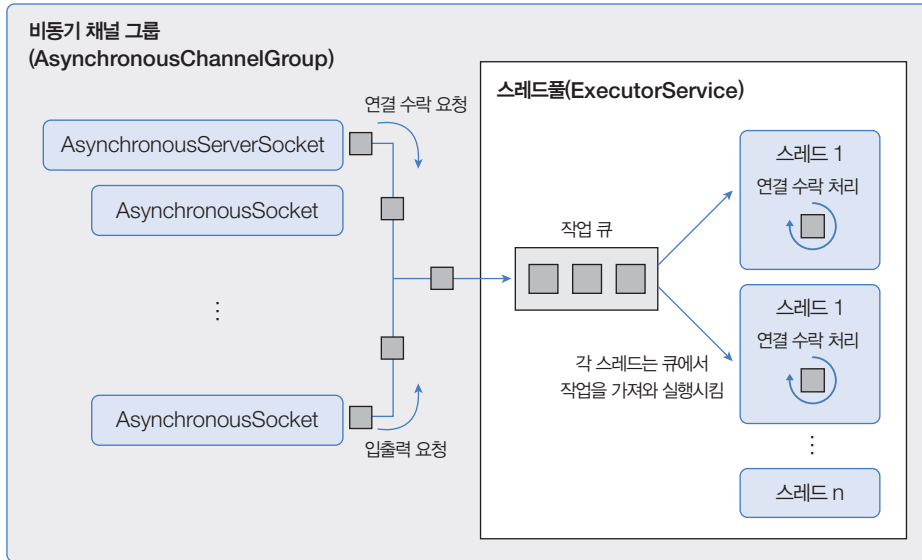


애플리케이션에서 `read()` 메소드를 호출하면 즉시 리턴되지만, 내부에서는 스레드풀의 작업 스레드가 읽기 작업을 수행한다. 작업이 완료되면 콜백 메소드인 `completed()`가 자동 호출된다.



비동기 채널 그룹

비동기 서버 소켓 채널과 비동기 소켓 채널을 살펴보기 전에 우선 비동기 채널 그룹에 대해서 이해해 보자. 비동기 채널 그룹(`AsynchronousChannelGroup`)은 같은 스레드풀을 공유하는 비동기 채널들의 묶음이라고 볼 수 있다.



비동기 채널을 생성할 때 채널 그룹을 지정하지 않으면 기본 비동기 채널 그룹이 생성된다. 기본 비동기 채널 그룹은 내부적으로 다음과 같이 스레드풀을 생성한다

```
new ThreadPoolExecutor(
    0, Integer.MAX_VALUE,
    Long.MAX_VALUE, TimeUnit.MILLISECONDS,
    new SynchronousQueue<Runnable>(),
    threadFactory
);
```

이론적으로 Integer.MAX_VALUE개만큼 스레드가 증가할 수 있도록 되어 있다. 하지만 스레드풀은 대부분 최대 스레드 수를 지정해서 사용하므로 다음과 같이 사용자 비동기 채널 그룹을 직접 생성하는 것이 일반적이다.

```
AsynchronousChannelGroup channelGroup = AsynchronousChannelGroup.withFixedThreadPool(
    maxThreadNum,
    Executors.defaultThreadFactory()
);
```

다음은 CPU 코어의 수만큼 스레드를 관리하는 스레드풀을 생성하고 이것을 이용하는 비동기 채널 그룹을 생성한다.

```
AsynchronousChannelGroup channelGroup = AsynchronousChannelGroup.withFixedThreadPool(  
    Runtime.getRuntime().availableProcessors(),  
    Executors.defaultThreadFactory()  
);
```

이렇게 생성된 비동기 채널 그룹은 비동기 채널을 생성할 때 매개값으로 사용된다. 비동기 채널 그룹을 더 이상 사용하지 않고 종료할 경우에는 `shutdown()`과 `shutdownNow()` 메소드를 호출할 수 있다.

```
channelGroup.shutdown();  
channelGroup.shutdownNow();
```

`shutdown()`은 비동기 채널 그룹을 종료하겠다는 의사만 전달할 뿐 즉시 비동기 채널 그룹을 종료하지 않는다. 비동기 채널 그룹에 포함된 모든 비동기 채널이 닫히면 비로소 비동기 채널 그룹을 종료시킨다.

`shutdownNow()`는 강제로 비동기 채널 그룹에 포함된 모든 비동기 채널을 닫아버리고 비동기 채널 그룹을 종료한다.

비동기 서버 소켓 채널

`AsynchronousServerSocketChannel`은 두 가지 정적 메소드인 `open()`을 호출해서 얻을 수 있다. 다음과 같이 매개값 없는 `open()` 메소드를 호출하면 기본 비동기 채널 그룹에 포함되는 `AsynchronousServerSocketChannel`을 얻을 수 있다.

```
AsynchronousServerSocketChannel assc = AsynchronousServerSocketChannel.open();
```

별도로 비동기 채널 그룹을 생성하고 여기에 포함되는 `AsynchronousServerSocketChannel`을 얻고 싶다면 다음과 같이 비동기 채널 그룹을 매개값으로 갖는 `open()` 메소드를 호출하면 된다.


```
AsynchronousChannelGroup channelGroup = AsynchronousChannelGroup.withFixedThreadPool(
    최대스레드수, Executors.defaultThreadFactory()
);
AsynchronousServerSocketChannel assc = AsynchronousServerSocketChannel.open
(channelGroup);
```

AsynchronousServerSocketChannel을 생성하고 나서는 포트 바인딩을 위해 다음과 같이 bind() 메소드를 호출한다.

```
ascc.bind(new InetSocketAddress(50001));
```

바인딩 작업이 끝나면 클라이언트의 연결 요청을 수락할 수 있는 accept() 메소드를 다음과 같이 호출할 수 있다. 이 메소드는 호출 즉시 리턴되지만 스레드풀에서 연결 수락 작업을 진행한다.

```
ascc.accept(null, new CompletionHandler<AsynchronousSocketChannel, Void>() {
    @Override
    public void completed(AsynchronousSocketChannel asc, Void attachment) {
        //연결 수락 후 실행할 코드
        //...

        //다음 연결 수락을 위해 accept() 재호출
        ascc.accept(null, this);
    }

    @Override
    public void failed(Throwable exc, Void attachment) {
        //연결 수락 실패 시 실행할 코드
        //...
    }
});
```

첫 번째 매개값은 콜백 메소드의 매개값으로 제공할 첨부 객체인데, 연결 수락 작업에는 별도의 첨부 객체가 필요하지 않기 때문에 null을 지정한다. 두 번째 매개값은 콜백 객체로, CompletionHandler<AsynchronousSocketChannel, A> 구현 객체이다. A는 첨부 객체 타입인데, 첫 번째 매개값이 null이므로 Void로 지정한다.

completed()는 연결 수락이 되었을 때 스레드풀의 스레드에서 호출되는 콜백 메소드이다. 첫 번째 매개값은 클라이언트와 통신할 수 있는 AsynchronousSocketChannel이고, 두 번째 매개값은 첨부 객체인데 없기 때문에 null이 대입된다.

failed()는 연결 수락 시에 예외가 발생되면 스레드풀의 스레드에서 호출되는 콜백 메소드이다. 첫 번째 매개값은 예외 객체이고 두 번째 매개값은 첨부 객체인데 없기 때문에 null이 대입된다.

주목할 점은 클라이언트의 연결 수락을 계속하기 위해 accept() 메소드를 반복 호출하는 제어문이 없고, 대신 completed() 메소드 끝에 다음 연결 수락을 위해 accept()를 다시 호출한다.

클라이언트의 연결 수락 작업을 멈추고 싶다면 다음과 같이 AsynchronousServerSocketChannel의 close() 메소드를 호출해서 포트를 언바인딩할 수 있다.

```
assc.close();
```

다음 예제는 계속적으로 클라이언트의 연결 수락 작업을 수행하는 가장 기본적인 TCP 비동기 서버 코드를 보여준다.

>>> ServerExample.java

```
1  package sec07.exam01_asynchronous_tcpchannel;
2
3  import java.io.IOException;
4  import java.net.InetSocketAddress;
5  import java.nio.channels.AsynchronousChannelGroup;
6  import java.nio.channels.AsynchronousServerSocketChannel;
7  import java.nio.channels.AsynchronousSocketChannel;
8  import java.nio.channels.CompletionHandler;
9  import java.util.concurrent.Executors;
10
11 public class ServerExample {
12     private static AsynchronousChannelGroup channelGroup;
13     private static AsynchronousServerSocketChannel assc;
14
15     public static void main(String[] args) {
16         System.out.println("[서버 시작]");
17     }
18 }
```

```

18     try {
19         //비동기 채널 그룹 생성
20         channelGroup = AsynchronousChannelGroup.withFixedThreadPool(
21             10, Executors.defaultThreadFactory());
22
23         //비동기 서버 소켓 채널 생성
24         assc = AsynchronousServerSocketChannel.open(channelGroup);
25
26         //포트 바인딩
27         assc.bind(new InetSocketAddress(50001));
28
29         //클라이언트 연결 수락하기
30         assc.accept(
31             null,
32             new CompletionHandler<AsynchronousSocketChannel, Void>() {
33                 @Override
34                 public void completed(AsynchronousSocketChannel asc,
35                     Void attachment) {
36                     try {
37                         InetSocketAddress isa = (InetSocketAddress)
38                             asc.getRemoteAddress();
39                         System.out.println(isa.getHostName() + " 연결 수락");
40                         try { asc.close(); } catch (Exception e) {}
41                         System.out.println(isa.getHostName() + " 연결 종료");
42                     } catch (IOException e) {}
43
44                     //다음 클라이언트 연결 수락하기
45                     assc.accept(null, this);
46                 }
47             }
48         );
49
50         //키보드 입력이 있을 때까지 대기
51         try { System.in.read(); } catch (Exception e) {}
52     } catch (Exception e) {
53         e.printStackTrace();
54     } finally {

```

```

56         try {
57             assc.close();
58             channelGroup.shutdownNow();
59         } catch (Exception e) {}
60     }
61
62     System.out.println("[서버 종료]");
63 }
64 }

```

실행 결과

[서버 시작]

비동기 소켓 채널

AsynchronousSocketChannel은 서버와 클라이언트에 각각 존재하는데, 클라이언트가 AsynchronousSocketChannel을 생성해서 서버로 연결 요청을 하면 서버의 AsynchronousServerSocketChannel은 연결 수락 후 AsynchronousSocketChannel을 생성해서 서로 통신할 수 있도록 만들어준다.

AsynchronousServerSocketChannel이 생성하는 AsynchronousSocketChannel은 자동적으로 AsynchronousServerSocketChannel과 같은 비동기 채널 그룹에 속하게 된다.

클라이언트에서 AsynchronousSocketChannel을 생성하려면 두 가지 open() 메소드를 사용할 수 있다. 다음과 같이 매개값없는 open() 메소드를 호출하면 기본 비동기 채널에 포함되는 AsynchronousSocketChannel을 얻을 수 있다.

```
AsynchronousSocketChannel asc = AsynchronousSocketChannel.open();
```

별도로 비동기 채널 그룹을 생성하고 여기에 포함되는 AsynchronousSocketChannel을 얻고 싶다면 다음과 같이 비동기 채널 그룹을 매개값으로 갖는 open() 메소드를 호출하면 된다

```
AsynchronousChannelGroup channelGroup = AsynchronousChannelGroup.withFixedThreadPool(
    최대스레드수, Executors.defaultThreadFactory()
);
AsynchronousSocketChannel asc = AsynchronousSocketChannel.open(channelGroup);
```

AsynchronousSocketChannel은 서버 연결 요청 작업을 스레드풀을 이용해서 비동기로 처리한다. 다음은 connect() 메소드를 호출하는 코드이다.

```
asc.connect(
    new InetSocketAddress("localhost", 5001),
    null,
    new CompletionHandler<Void, Void>() {
        @Override
        public void completed(Void result, Void attachment) {
            //연결 성공 후 실행할 코드
            //...
        }

        @Override
        public void failed(Throwable e, Void attachment) {
            //연결 실패 후 실행할 코드
            //...
        }
    });
```

첫 번째 매개값은 서버 IP와 연결 포트 정보를 가진 InetSocketAddress 객체이다. 두 번째 매개값은 콜백 객체에서 사용할 첨부 객체인데, 연결 요청 작업에는 별도의 첨부 객체가 필요하지 않기 때문에 null을 지정한다.

세 번째 매개값은 CompletionHandler<Void, A>를 구현한 콜백 객체이다. A는 첨부 타입을 말하는데, 두 번째 매개값을 null로 지정했기 때문에 Void로 지정한다.

completed()는 연결이 성공했을 때 스레드풀의 스레드에서 호출되는 콜백 메소드이다. 첫 번째 매개값은 무조건 null이 대입되고, 두 번째 매개값은 첨부 객체인데, 없기 때문에 null이 대입된다.

failed()는 연결 요청 시 예외가 발생하면 스레드풀의 스레드에서 호출되는 콜백 메소드이다. 첫 번째 매개값은 예외 객체이고, 두 번째 매개값은 첨부 객체인데 없기 때문에 null이 대입된다.

AsynchronousSocketChannel을 더 이상 사용하지 않을 경우에는 close() 메소드를 호출해서 연결을 끊어준다.

```
asynchronousSocketChannel.close();
```

다음 예제는 서버로 연결 요청을 하는 가장 기본적인 TCP 비동기 클라이언트 코드를 보여준다.

» ClientExample.java

```
1  package sec07.exam01_asynchronous_tcpchannel;
2
3  import java.net.InetSocketAddress;
4  import java.nio.channels.AsynchronousSocketChannel;
5  import java.nio.channels.CompletionHandler;
6
7  public class ClientExample {
8      public static void main(String[] args) {
9          System.out.println("[클라이언트 시작]");
10
11         try {
12             //비동기 소켓 채널 생성
13             AsynchronousSocketChannel asc = AsynchronousSocketChannel.open();
14
15             //서버로 연결 요청하기
16             asc.connect(
17                 new InetSocketAddress("localhost", 50001),
18                 null,
19                 new CompletionHandler<Void, Void>() {
20                     @Override
21                     public void completed(Void result, Void attachment) {
22                         System.out.println("연결 성공");
23                         try { asc.close(); } catch (Exception e) {}
24                         System.out.println("연결 종료");
25                     }
26                     @Override
27                     public void failed(Throwable exc, Void attachment) {
28                         exc.printStackTrace();
29                         try { asc.close(); } catch (Exception e) {}
30                     }
31                 }
32             );
33         } catch (Exception e) {
34             e.printStackTrace();
35         }
36     }
37 }
```

```

30         }
31     }
32 );
33
34     //키보드 입력이 있을 때까지 대기
35     try { System.in.read(); } catch (Exception e) {}
36 } catch (Exception e) {
37     e.printStackTrace();
38 }
39
40 System.out.println("[클라이언트 종료]");
41 }
42 }

```

실행 결과

ClientExample.java	ServerExample.java
[클라이언트 시작]	[서버 시작]
연결 성공	127.0.0.1 연결 수락
연결 종료	127.0.0.1 연결 종료
[Enter]	[Enter]
[클라이언트 종료]	[서버 종료]

네트워크 입출력

클라이언트와 서버가 연결되면 양쪽 `AsynchronousSocketChannel`의 `read()`와 `write()` 메소드로 네트워크 입출력을 할 수 있다. 이 메소드들은 호출하는 즉시 리턴되고, 실질적인 입출력 작업은 스레드풀의 스레드가 담당한다. 다음은 `read()`와 `write()`를 호출하는 코드이다.

```

read(ByteBuffer dst, A attachment, CompletionHandler<Integer, A> handler);
write(ByteBuffer src, A attachment, CompletionHandler<Integer, A> handler);

```

첫 번째 매개값은 읽고 쓰기 위한 `ByteBuffer` 객체이고, 두 번째 매개값은 콜백 객체에서 사용할 첨부 객체이다. 세 번째 매개값은 `CompletionHandler<Integer, A>`를 구현한 콜백 객체인데, 다음과 같이 생성할 수 있다.

```

new CompletionHandler<Integer, A>() {
    @Override
    public void completed(Integer result, A attachment) {
        //입출력 작업이 완료되면 실행
        //...
    }

    @Override
    public void failed(Throwable exc, A attachment) {
        //입출력 작업 시 예외가 발생하면 실행
        //...
    }
}

```

completed()는 입출력이 완료되었을 때 스레드풀의 스레드에서 호출되는 콜백 메소드이다. 첫 번째 매개 값은 입출력된 바이트 수이고, 두 번째 매개 값은 첨부 객체인데, read()와 write()를 호출할 때 준 두 번째 매개 값에 해당한다.

failed()는 입출력 작업 시 예외가 발생하면 스레드풀의 스레드에서 호출되는 콜백 메소드이다. 첫 번째 매개 값은 예외 객체이고, 두 번째 매개 값은 첨부 객체인데, read()와 write()를 호출할 때 준 두 번째 매개 값에 해당한다.

read() 메소드를 호출할 경우에는 상대방이 보내는 데이터를 계속 입력받기 위해서 콜백 객체의 completed() 메소드 끝에 다음과 같이 read() 메소드를 다시 호출할 수도 있다.

```

@Override
public void completed(Integer result, A attachment) {
    //받은 데이터를 처리하는 코드
    //...
    asc.read(byteBuffer, attachment, this);
}

```

다음 예제는 100개의 클라이언트가 동시에 서버로 데이터를 보내고 받는 예제이다. 서버는 모든 클라이언트의 요청을 처리하기 위해 10개의 스레드로 비동기 처리한다.

>>> ServerExample.java

```
1  package sec07.exam02_data_read_write;
2
3  import java.io.IOException;
4  import java.net.InetSocketAddress;
5  import java.nio.ByteBuffer;
6  import java.nio.channels.AsynchronousChannelGroup;
7  import java.nio.channels.AsynchronousServerSocketChannel;
8  import java.nio.channels.AsynchronousSocketChannel;
9  import java.nio.channels.CompletionHandler;
10 import java.nio.charset.Charset;
11 import java.util.concurrent.Executors;
12
13 public class ServerExample {
14     private static AsynchronousChannelGroup channelGroup;
15     private static AsynchronousServerSocketChannel assc;
16
17     public static void main(String[] args) {
18         System.out.println("[서버 시작]");
19
20         try {
21             //비동기 채널 그룹 생성
22             channelGroup = AsynchronousChannelGroup.withFixedThreadPool(
23                 10, Executors.defaultThreadFactory());
24
25             //비동기 서버 소켓 채널 생성
26             assc = AsynchronousServerSocketChannel.open(channelGroup);
27
28             //포트 바인딩
29             assc.bind(new InetSocketAddress(50001));
30
31             //클라이언트 연결 수락하기
32             assc.accept(
33                 null,
34                 new CompletionHandler<AsynchronousSocketChannel, Void>() {
35                     @Override
36                     public void completed(AsynchronousSocketChannel asc,
37                         Void attachment) {
```

```

38         //클라이언트가 보낸 데이터 받기
39         receive(asc);
40
41         //다음 클라이언트 연결 수락하기
42         asc.accept(null, this);
43     }
44
45     @Override
46     public void failed(Throwable exc, Void attachment) {
47     }
48 }
49 );
50
51 //키보드 입력이 있을 때까지 대기
52 try { System.in.read(); } catch (Exception e) {}
53 } catch (Exception e) {
54     e.printStackTrace();
55 } finally {
56     try {
57         asc.close();
58         channelGroup.shutdownNow();
59     } catch (Exception e) {}
60 }
61
62 System.out.println("[서버 종료]");
63 }
64
65 //클라이언트가 보낸 데이터 받기
66 public static void receive(AsynchronousSocketChannel asc) {
67     ByteBuffer byteBuffer = ByteBuffer.allocate(100);
68     asc.read(byteBuffer, byteBuffer, new CompletionHandler<Integer,
69         ByteBuffer>() {
70         @Override
71         public void completed(Integer result, ByteBuffer attachment) {
72             try {
73                 attachment.flip();
74                 Charset charset = Charset.forName("utf-8");
75                 String receiveData = charset.decode(attachment).toString();
76
77                 String threadName = Thread.currentThread().getName();

```

```

77         System.out.println("[ " + threadName + " ] " + "데이터 받음: " +
            receiveData);
78
79         //클라이언트로 데이터 보내기
80         send(asc, receiveData);
81     } catch (Exception e) {}
82 }
83
84 @Override
85 public void failed(Throwable exc, ByteBuffer attachment) {
86     exc.printStackTrace();
87     try { asc.close(); } catch (IOException e) {}
88 }
89 });
90 }
91
92 //클라이언트로 데이터 보내기
93 public static void send(AsynchronousSocketChannel asc,
    String receiveData) {
94     String sendData = "Hello Client " + receiveData.substring(13);
95     Charset charset = Charset.forName("utf-8");
96     ByteBuffer byteBuffer = charset.encode(sendData);
97     asc.write(byteBuffer, sendData, new CompletionHandler<Integer,
        String>() {
98         @Override
99         public void completed(Integer result, String attachment) {
100             String threadName = Thread.currentThread().getName();
101             System.out.println("[ " + threadName + " ] " + "데이터 보냄: " +
                attachment);
102             try { asc.close(); } catch (IOException e) {}
103         }
104     }
105     @Override
106     public void failed(Throwable exc, String attachment) {
107         exc.printStackTrace();
108         try { asc.close(); } catch (IOException e) {}
109     }
110 });
111 }
112 }

```

>>> ClientExample.java

```
1  package sec07.exam02_data_read_write;
2
3  import java.net.InetSocketAddress;
4  import java.nio.ByteBuffer;
5  import java.nio.channels.AsynchronousSocketChannel;
6  import java.nio.channels.CompletionHandler;
7  import java.nio.charset.Charset;
8
9  public class ClientExample {
10     public static void main(String[] args) {
11         System.out.println("[클라이언트 시작]");
12
13         try {
14             for (int i = 1; i <= 100; i++) {
15                 //비동기 소켓 채널 생성
16                 AsynchronousSocketChannel asc = AsynchronousSocketChannel.open();
17
18                 //서버로 연결 요청하기
19                 int count = i;
20                 asc.connect(new InetSocketAddress("localhost", 50001), null,
21                     new CompletionHandler<Void, Void>() {
22                         @Override
23                         public void completed(Void result, Void attachment) {
24                             //서버로 데이터 보내기
25                             receive(asc, count);
26                         }
27
28                         @Override
29                         public void failed(Throwable exc, Void attachment) {
30                             exc.printStackTrace();
31                             try { asc.close(); } catch (Exception e) {}
32                         }
33                     }
34             );
15
```

```

35         }
36
37         //키보드 입력이 있을 때까지 대기
38         try { System.in.read(); } catch (Exception e) {}
39     } catch (Exception e) {
40         e.printStackTrace();
41     }
42
43     System.out.println("[클라이언트 종료]");
44 }
45
46 //서버로 데이터 보내기
47 public static void receive(AsynchronousSocketChannel asc, int count) {
48     Charset charset = Charset.forName("utf-8");
49     String sendData = "Hello Server " + count;
50     ByteBuffer byteBuffer = charset.encode(sendData);
51     asc.write(byteBuffer, null, new CompletionHandler<Integer, Void>() {
52         @Override
53         public void completed(Integer result, Void attachment) {
54             System.out.println("데이터 보냄: " + sendData);
55             //서버가 보낸 데이터 받기
56             send(asc);
57         }
58
59         @Override
60         public void failed(Throwable exc, Void attachment) {
61             exc.printStackTrace();
62             try {
63                 asc.close();
64             } catch (Exception e) {
65             }
66         }
67     });
68 }
69
70 //서버가 보낸 데이터 받기
71 public static void send(AsynchronousSocketChannel asc) {
72     ByteBuffer byteBuffer = ByteBuffer.allocate(100);
73     asc.read(byteBuffer, byteBuffer, new CompletionHandler<Integer,
74         ByteBuffer>() {

```

```

74         @Override
75         public void completed(Integer result, ByteBuffer attachment) {
76             try {
77                 attachment.flip();
78                 Charset charset = Charset.forName("utf-8");
79                 String receiveData = charset.decode(attachment).toString();
80                 System.out.println("데이터 받음: " + receiveData);
81                 asc.close();
82             } catch (Exception e) {
83             }
84         }
85
86         @Override
87         public void failed(Throwable exc, ByteBuffer attachment) {
88             exc.printStackTrace();
89             try { asc.close(); } catch (Exception e) {}
90         }
91     });
92 }
93 }

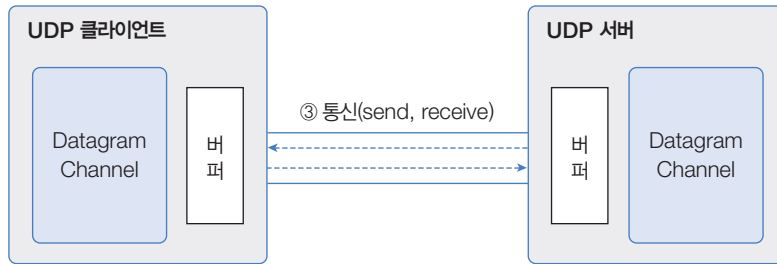
```

실행 결과

ClientExample.java	ServerExample.java
[클라이언트 시작]	[서버 시작]
...	...
데이터 보냄: Hello Server 92	[pool-1-thread-4] 데이터 받음: Hello Server 94
데이터 받음: Hello Client 70	[pool-1-thread-6] 데이터 보냄: Hello Client 94
데이터 보냄: Hello Server 93	[pool-1-thread-4] 데이터 받음: Hello Server 95
데이터 보냄: Hello Server 94	[pool-1-thread-6] 데이터 보냄: Hello Client 95
데이터 받음: Hello Client 71	[pool-1-thread-6] 데이터 받음: Hello Server 96
데이터 보냄: Hello Server 95	[pool-1-thread-3] 데이터 보냄: Hello Client 96
...	...
[Enter]	[Enter]
[클라이언트 종료]	[서버 종료]

08 UDP 네트워크 입출력

NIO에서 UDP 채널은 데이터그램 채널 `DatagramChannel`이다. 서버와 클라이언트는 데이터그램 채널을 이용해서 버퍼의 데이터를 입출력한다.



데이터그램 채널

`DatagramChannel`을 생성하려면 `open()` 메소드를 호출해야 한다. 매개값으로 `ProtocolFamily` 인터페이스 구현 객체가 필요한데, 이 객체는 `StandardProtocolFamily` 열거 상수를 사용한다. 다음은 IPv4를 사용하는 `DatagramChannel`을 생성하는 코드이다.

```
DatagramChannel datagramChannel = DatagramChannel.open(StandardProtocolFamily.INET);
```

UDP 서버가 되려면 `DatagramChannel`이 특정 포트_{port}와 바인딩되어야 한다. 이 포트는 클라이언트가 데이터를 보낼 때 사용된다. 다음은 50001번 포트와 바인딩하기 위해 `bind()` 메소드를 호출하는 방법을 보여준다.

```
datagramChannel.bind(new InetSocketAddress(50001));
```

네트워크 입출력

상대방이 메시지를 보내면 `DatagramChannel`의 `receive()` 메소드로 다음과 같이 읽을 수 있다.

```
InetSocketAddress isa = (InetSocketAddress) datagramChannel.receive(byteBuffer);
```

receive() 메소드의 매개값은 받은 데이터를 저장할 ByteBuffer이다. 데이터를 받기 전까지 receive() 메소드는 블로킹되고, 데이터를 받으면 리턴된다. 리턴 타입은 SocketAddress인데, 실제로는 InetSocketAddress 객체가 리턴된다. InetSocketAddress를 통해 상대방의 IP와 포트 정보를 다음과 같이 알 수 있다.

```
String clientIp = isa.getHostName();  
int clientPort = isa.getPort();
```

상대방에게 데이터를 보내기 위해서는 send() 메소드를 이용한다. 다음은 로컬 PC의 50001번에서 실행하고 있는 UDP 서버로 데이터를 보낸다.

```
int byteCount = datagramChannel.send(byteBuffer, new InetSocketAddress  
("localhost", 50001));
```

send()의 첫 번째 매개값은 보낼 데이터를 가지고 있는 ByteBuffer이고, 두 번째 매개값은 수신자 IP와 포트 정보를 가지고 있는 SocketAddress이다. SocketAddress는 추상 클래스이므로 하위 클래스인 InetSocketAddress 객체를 생성하고 대입하면 된다. send() 메소드의 리턴값은 실제로 보낸 바이트 수이다.

더 이상 네트워크 입출력이 필요없다면 DatagramChannel을 닫기 위해 close() 메소드를 호출한다.

```
datagramChannel.close();
```

다음 예제는 UDP 서버가 데이터를 받고 다시 UDP 클라이언트로 보내는 방법을 보여준다. UDP 클라이언트는 100번 데이터를 보내고, 100번 데이터를 받는다.

>>> ServerExample.java

```
1  package sec08;
2
3  import java.io.IOException;
4  import java.net.InetSocketAddress;
5  import java.net.StandardProtocolFamily;
6  import java.nio.ByteBuffer;
7  import java.nio.channels.DatagramChannel;
8  import java.nio.charset.Charset;
9
10 public class ServerExample {
11     public static void main(String[] args) {
12         System.out.println("[서버 시작]");
13
14         DatagramChannel datagramChannel = null;
15
16         try {
17             //DatagramChannel 생성
18             datagramChannel = DatagramChannel.open(StandardProtocolFamily.INET);
19
20             //포트 바인딩
21             datagramChannel.bind(new InetSocketAddress(50001));
22
23             Charset charset = Charset.forName("UTF-8");
24
25             while(true) {
26                 try {
27                     //클라이언트가 보낸 데이터 받기
28                     ByteBuffer byteBuffer = ByteBuffer.allocateDirect(100);
29
30                     InetSocketAddress isa =
31                         (InetSocketAddress) datagramChannel.receive(byteBuffer);
32                     String clientIp = isa.getHostName();
33                     int clientPort = isa.getPort();
34
35                     byteBuffer.flip();
36                     String receiveData = charset.decode(byteBuffer).toString();
37                     System.out.println "[" + clientIp + " ] 데이터 받음: " +
                        receiveData);
```

```

38
39         //클라이언트로 데이터 보내기
40         String sendData = "Hello Client " + receiveData.substring(13);
41         byteBuffer = charset.encode(sendData);
42
43         int byteCount = datagramChannel.send(
44             byteBuffer,
45             new InetSocketAddress(clientIp, clientPort)
46         );
47
48         System.out.println("[ " + clientIp + " ] 데이터 보냄: " + sendData);
49     } catch (IOException e) {
50         e.printStackTrace();
51         break;
52     }
53 }
54 } catch (Exception e) {
55     e.printStackTrace();
56 } finally {
57     //DatagramChannel 닫기
58     try { datagramChannel.close(); } catch (Exception e) {}
59 }
60
61     System.out.println("[서버 종료]");
62 }
63 }

```

실행 결과

[서버 시작]

>>> ClientExample.java

```

1     package sec08;
2
3     import java.io.IOException;
4     import java.net.InetSocketAddress;
5     import java.net.StandardProtocolFamily;

```

```

6  import java.nio.ByteBuffer;
7  import java.nio.channels.DatagramChannel;
8  import java.nio.charset.Charset;
9
10 public class ClientExample {
11     public static void main(String[] args) {
12         System.out.println("[클라이언트 시작]");
13
14         DatagramChannel datagramChannel = null;
15
16         try {
17             //DatagramChannel 생성
18             datagramChannel = DatagramChannel.open(StandardProtocolFamily.INET);
19
20             Charset charset = Charset.forName("UTF-8");
21
22             for(int i=1; i<=100; i++) {
23                 try {
24                     //서버로 데이터 보내기
25                     String sendData = "Hello Server " + i;
26                     ByteBuffer byteBuffer = charset.encode(sendData);
27                     int byteCount = datagramChannel.send(
28                         byteBuffer,
29                         new InetSocketAddress("localhost", 50001)
30                     );
31                     System.out.println("데이터 보냄: " + sendData);
32
33                     //서버가 보낸 데이터 받기
34                     byteBuffer = ByteBuffer.allocateDirect(100);
35                     InetSocketAddress isa =
36                         (InetSocketAddress) datagramChannel.receive(byteBuffer);
37                     byteBuffer.flip();
38                     String receiveData = charset.decode(byteBuffer).toString();
39                     System.out.println("데이터 받음: " + receiveData);
40                 } catch(IOException e) {
41                     e.printStackTrace();
42                     break;
43                 }
44             } catch(Exception e) {

```

```

45         e.printStackTrace();
46     } finally {
47         //DatagramChannel 닫기
48         try { datagramChannel.close(); } catch(Exception e) {}
49     }
50
51     System.out.println("[클라이언트 종료]");
52 }
53 }

```

실행 결과

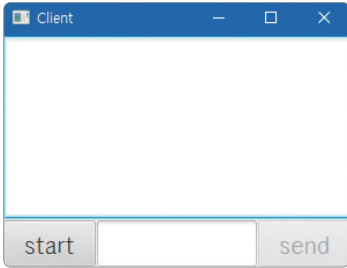
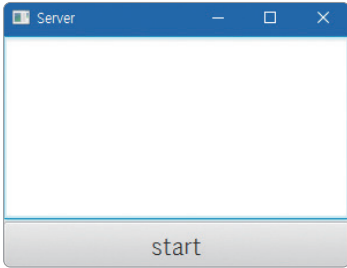
ClientExample.java	ServerExample.java
[클라이언트 시작]	[서버 시작]
데이터 보냄: Hello Server 1	[127.0.0.1] 데이터 받음: Hello Server 1
데이터 받음: Hello Client 1	[127.0.0.1] 데이터 보냄: Hello Client 1
...	...
데이터 보냄: Hello Server 100	[127.0.0.1] 데이터 받음: Hello Server 100
데이터 받음: Hello Client 100	[127.0.0.1] 데이터 보냄: Hello Client 100
[클라이언트 종료]	

09 NIO 과제

지금까지 학습한 내용을 기반으로 NIO 비동기 채널을 이용해서 다음과 같이 동작하는 채팅 서버와 채팅 클라이언트 과제를 수행해 보자.

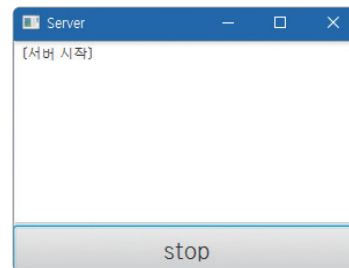
과제 1

UI 라이브러리는 JavaFX 또는 Swing 중 하나를 선택해서 사용하고, 서버와 클라이언트 UI를 다음과 같이 만든다.

ClientExample.java	ServerExample.java
	
<ul style="list-style-type: none"> - 중앙: TextArea(JTextArea) - [start] 버튼: Button(JButton) - 입력: TextField(JTextField) - [send] 버튼: Button(JButton), 비활성화 	<ul style="list-style-type: none"> - 중앙: TextArea(JTextArea) - [start] 버튼: Button(JButton)

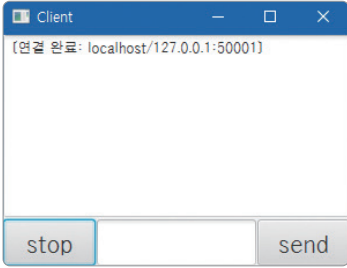
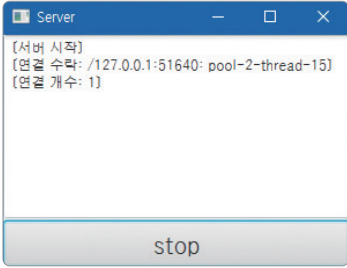
과제 2

서버에서 [start] 버튼을 클릭하면 [서버 시작]이라고 출력되고, [stop] 버튼으로 변경되도록 한다. 서버는 50001번 포트에 바인딩되도록 한다.



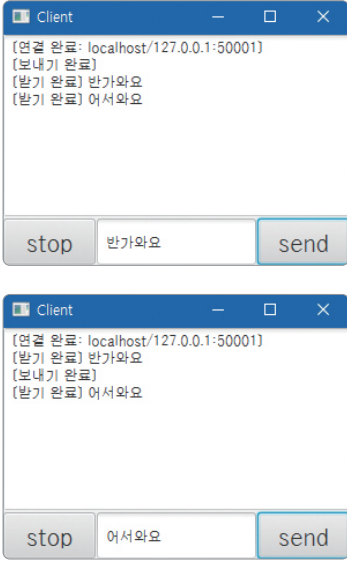
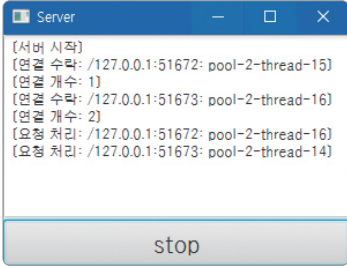
과제 3

클라이언트에서 [start] 버튼을 클릭하면 서버에 연결 요청을 한다. 연결이 성공되면 연결 완료 내용을 출력시키고, [stop] 버튼과 [send] 버튼을 활성화시킨다. 서버에서는 클라이언트 연결 수락 내용과 현재 연결 개수를 출력시킨다.

ClientExample.java	ServerExample.java
 <p>Client window showing: [연결 완료: localhost/127.0.0.1:50001]. Buttons: stop, send.</p>	 <p>Server window showing: [서버 시작], [연결 수락: /127.0.0.1:51640: pool-2-thread-15], [연결 개수: 1]. Button: stop.</p>

과제 4

클라이언트에서 글자를 입력하고 [send] 버튼을 클릭하면 입력된 글자는 서버로 전송되고, 서버는 연결된 모든 클라이언트로 글자를 보내도록 한다.

ClientExample.java	ServerExample.java
 <p>Two Client window screenshots showing the flow of messages. The first shows the client sending '반가워요' and receiving '어서와요'. The second shows the client sending '어서와요' and receiving '반가워요'.</p>	 <p>Server window showing: [서버 시작], [연결 수락: /127.0.0.1:51672: pool-2-thread-15], [연결 개수: 1], [연결 수락: /127.0.0.1:51673: pool-2-thread-16], [연결 개수: 2], [요청 처리: /127.0.0.1:51672: pool-2-thread-16], [요청 처리: /127.0.0.1:51673: pool-2-thread-14]. Button: stop.</p>

과제 5

클라이언트에서 [stop] 버튼을 클릭하면 [서버 통신 안됨]을 출력하고, [start] 버튼을 활성화, [send] 버튼을 비활성화시킨다. 서버에서는 클라이언트 통신 안됨으로 출력한다.

ClientExample.java	ServerExample.java
