

## 17장 JavaFX

### 17.1 JavaFX 개요

자바 UI 변천사

- AWT(Abstract Window Toolkit)
  - 운영 체제가 제공하는 네이티브 UI 컴포넌트 이용
  - 운영 체제에 따라 UI의 모양 서로 달랐고, 종류도 제한적
- Swing
  - 모든 운영체제상에서 동일한 UI 갖도록
  - 사용자는 애니메이션 추가된 시각적 운영 체제의 네이티브 UI 더 선호
    - 네이티브 UI로 보여지도록 자신의 UI 재정비
    - 실행 성능이 느려지고, 메모리 더 많이 사용
- JavaFX
  - 가볍고 풍부한 UI 제공
  - 레이아웃, 스타일, 애플리케이션 로직 분리 개발
  - 자바7 업데이트6버전부터 JavaFX2.2를 JDK와 JRE에 포함

JavaFX 애플리케이션 구성하는 파일단위 구성요소



### 17.2 JavaFX 애플리케이션 개발 시작

#### 17.2.1. 메인 클래스

- JavaFX 애플리케이션을 시작시키는 메인 클래스는 추상 클래스인 `javafx.application.Application`을 상속받고, `start()` 메소드를 재정의해야 한다.

[AppMain.java] 메인 클래스

```
package sec02.exam01_application_start;

import javafx.application.Application;
import javafx.stage.Stage;

public class AppMain extends Application {
    @Override
    public void start(Stage primaryStage) throws Exception {
        primaryStage.show(); // 윈도우 보여주기
    }
}
```

```

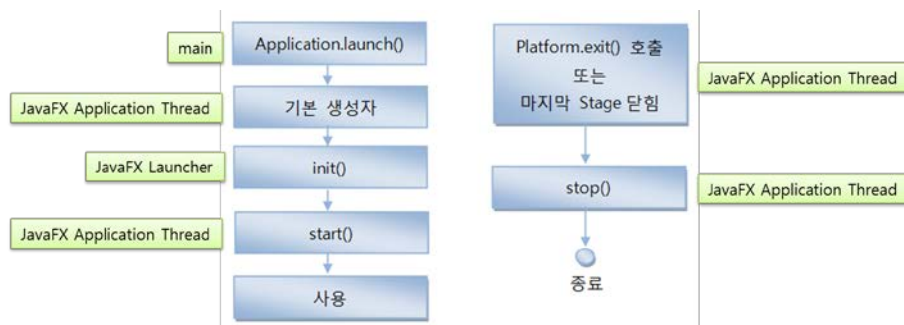
    }

    public static void main(String[] args) {
        launch(args); // AppMain 객체 생성 및 메인 윈도우 생성
    }
}

```

### 17.2.2. JavaFX 라이프사이클

- 라이프사이클의 각 단계에서 호출되는 메소드는 서로 다른 스레드상에서 실행된다.
  - main : Application.launch() 실행
  - JavaFX Application Thread : 메인 클래스 기본 생성자, start() 및 stop() 실행
  - JavaFX-Launcher : init() 실행



[AppMain.java] 라이프 사이클

```

package sec02.exam02_application_lifecycle;

import javafx.application.Application;
import javafx.stage.Stage;

public class AppMain extends Application {
    public AppMain() {
        System.out.println(Thread.currentThread().getName()+" : AppMain() 호출");
    }

    @Override
    public void init() throws Exception {
        System.out.println(Thread.currentThread().getName()+" : init() 호출");
    }

    @Override
    public void start(Stage primaryStage) throws Exception {
        System.out.println(Thread.currentThread().getName()+" : start() 호출");
        primaryStage.show();
    }

    @Override
    public void stop() throws Exception {
        System.out.println(Thread.currentThread().getName()+" : stop() 호출");
    }

    public static void main(String[] args) {
        System.out.println(Thread.currentThread().getName()+" : main() 호출");
        launch(args);
    }
}

```

### 17.2.3. 메인 클래스 실행 매개값 얻기

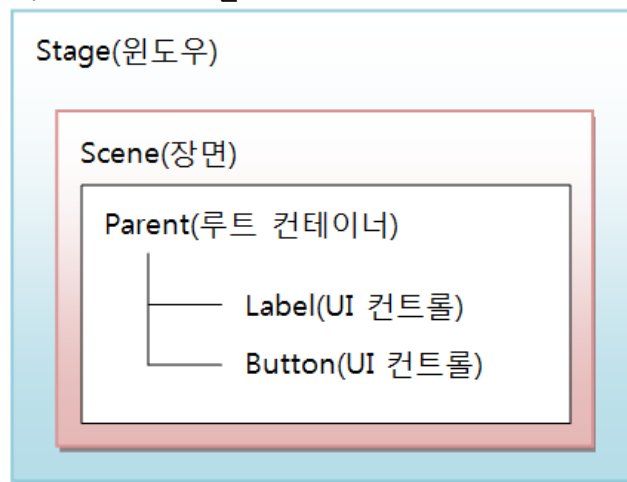
- `init()` 메소드에서 다음 두 가지 방법으로 매개값 얻기
  - `getRaw()` 메소드 : "--ip=192.168.8.5"와 "--port=50001"을 요소로 갖는 list 컬렉션을 리턴한다.
  - `getNamed()` 메소드 : "ip"을 키로해서 "192.168.8.5"를 저장하고, "port"을 키로해서 "50001"을 저장하는 Map 컬렉션을 리턴한다.

```
java AppMain --ip=192.168.8.5 --port=50001

Parameters params = getParameters();
List<String> list = params.getRaw();
Map<String, String> map = params.getNamed();
```

### 17.2.4. 무대(Stage)와 장면(Scene)

- 무대는 윈도우 하나에 하나의 장면 가질 수 있음
- 장면은 `javafx.scene.Scene` 으로 표현



```
Scene scene = new Scene(Parent root);
primaryStage.setScene(scene);
```

[AppMain.java] Stage와 Scene

```
package sec02.exam03_stage_scene;

import javafx.application.Application;
import javafx.application.Platform;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.layout.VBox;
```

```
import javafx.scene.text.Font;
import javafx.stage.Stage;

public class AppMain extends Application {
    @Override
    public void start(Stage primaryStage) throws Exception {
        VBox root = new VBox(); // Parent 하위 객체인 VBox 생성
        root.setPrefWidth(350); // VBox의 폭 설정
        root.setPrefHeight(150); // VBox의 높이 설정
        root.setAlignment(Pos.CENTER); // 컨트롤을 중앙으로 정렬
        root.setSpacing(20); // 컨트롤의 수직 간격

        Label label = new Label(); // Label 컨트롤 생성
        label.setText("Hello, JavaFX"); // text 속성 설정
        label.setFont(new Font(50)); // font 속성 설정

        Button button = new Button(); // Button 컨트롤 생성
        button.setText("확인"); // text 속성 설정
        button.setOnAction(event -> Platform.exit()); // 클릭 이벤트 처리

        root.getChildren().add(label); // VBox에 Label 컨트롤 추가
        root.getChildren().add(button); // VBox에 Button 컨트롤 추가

        Scene scene = new Scene(root); // VBox를 루트 컨테이너로 해서 Scene 생성

        primaryStage.setTitle("AppMain"); // 윈도우의 제목 설정
        primaryStage.setScene(scene); // 윈도우에 장면 설정
        primaryStage.show(); // 윈도우 보여주기
    }

    public static void main(String[] args) {
        launch(args);
    }
}
```

## 17.3 JavaFX 레이아웃

### 17.3.1. 프로그램적 레이아웃

- 자바 코드로만 개발
- 간단하게 쉽게 만들 것 - 코드를 잘 정리하지 않으면 난해한 프로그램
- 개발자가 직접 작성 - 디자이너와 협력 개발 어려움
- 개발 완료 후 간단한 레이아웃 변경이나 스타일 변경이라도 자바 소스 수정 후 재 컴파일

[AppMain.java] 프로그램적 레이아웃

```
package sec03.exam01_programmatical_layout;

import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.layout.HBox;
import javafx.geometry.Insets;
import javafx.scene.control.TextField;
import javafx.scene.control.Button;
import javafx.scene.Scene;
```

```
import javafx.collections.ObservableList;

public class AppMain extends Application {
    @Override
    public void start(Stage primaryStage) throws Exception {
        HBox hbox = new HBox(); // HBox 컨테이너 생성
        hbox.setPadding(new Insets(10, 10, 10, 10)); // 안쪽 여백 설정
        hbox.setSpacing(10); // 컨트롤간의 수평 간격 설정

        TextField textField = new TextField(); // TextField 컨트롤 생성
        textField.setPrefWidth(200); // TextField의 폭 설정

        Button button = new Button(); // Button 컨트롤 생성
        button.setText("확인"); // Button 글자 설정

        ObservableList list = hbox.getChildren(); // HBox의 ObservableList 얻기
        list.add(textField); // TextField 컨트롤 배치
        list.add(button); // Button의 컨트롤 배치

        Scene scene = new Scene(hbox); // 화면의 루트 컨테이너로 HBox 지정

        primaryStage.setTitle("AppMain"); // 윈도우 창 제목 설정
        primaryStage.setScene(scene); // 윈도우 창에 화면 설정
        primaryStage.show(); // 윈도우 창 보여주기
    }

    public static void main(String[] args) {
        launch(args);
    }
}
```

### 17.3.2. FXML 레이아웃

- FXML은 XML 기반의 마크업 언어
- JavaFX UI 레이아웃 자바 코드에서 분리 - 태그로 선언하는 방법 제공
- 웹 애플리케이션 및 안드로이드(Android) 앱 개발법과 유사
- 디자이너와 협업 가능
- 간단한 레이아웃 변경이나 스타일 변경 시 자바 소스 수정 X
- 레이아웃이 비슷한 장면(Scene)들간에 재사용 가능

[AppMain.java] FXML 레이아웃

```
package sec03.exam02_fxml_layout;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

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);

        primaryStage.setTitle("AppMain");
        primaryStage.setScene(scene);
        primaryStage.show();
    }

    public static void main(String[] args) {
        launch(args);
    }
}

```

[root.fxml] FXML 레이아웃

```

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.layout.HBox?>
<?import javafx.geometry.Insets?>
<?import javafx.scene.control.*?>

<HBox xmlns:fx="http://javafx.com/fxml" <!-- HBox 컨테이너 선언 -->
    <padding> <!-- 안쪽 여백 설정 -->
        <Insets top="10" right="10" bottom="10" left="10" />
    </padding>
    <spacing>10</spacing> <!-- 컨트롤간의 수평 간격 설정 -->

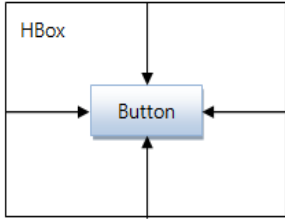
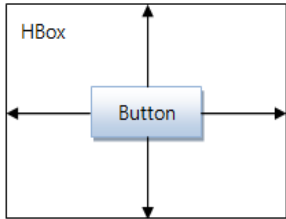
    <children> <!-- 자식 컨트롤 추가 -->
        <TextField> <!-- TextField 선언 -->
            <prefWidth>200</prefWidth> <!-- TextField의 폭 설정 -->
        </TextField>

        <Button> <!-- Button 컨트롤 선언 -->
            <text>확인</text> <!-- Button 글자 설정 -->
        </Button>
    </children>
</HBox>

```

### 17.3.3. 레이아웃 여백: 패딩과 마진

- 패딩은 안쪽 여백, 마진은 바깥 여백

구분	HBox 의 패딩	Button 의 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>&lt;HBox&gt;   &lt;padding&gt;     &lt;Insets topRightBottomLeft="50"/&gt;   &lt;/padding&gt; &lt;/HBox&gt;</pre>	<pre>&lt;Button&gt;   &lt;HBox.margin&gt;     &lt;Insets topRightBottomLeft="50"/&gt;   &lt;/HBox.margin&gt; &lt;/Button&gt;</pre>

[AppMain.java] 패딩과 마진 적용

```
package sec03.exam03_margin_padding;

import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.HBox;
import javafx.stage.Stage;

public class AppMain extends Application {
    @Override
    public void start(Stage primaryStage) throws Exception {
        // 패딩 설정-----
        /*
        * HBox hbox = new HBox(); hbox.setPadding(new Insets(50, 10, 10, 50));
        * Button button = new Button(); button.setPrefSize(100, 100);
        */

        // 마진 설정-----
        HBox hbox = new HBox();
        Button button = new Button();
        button.setPrefSize(100, 100);
        HBox.setMargin(button, new Insets(10, 10, 50, 50));

        hbox.getChildren().add(button);

        Scene scene = new Scene(hbox);

        primaryStage.setTitle("AppMain");
        primaryStage.setScene(scene);
        primaryStage.show();
    }

    public static void main(String[] args) {
        launch(args);
    }
}
```

#### 17.3.4. FXML 작성 규칙

- FXML 태그는 자바 코드로 변환되어 실행: 자바 코드와 매핑 관계 존재
- 매핑 관계 잘 이해하면 JavaFX API 참조해 FXML 태그 쉽게 작성

##### (1) 패키지 선언

- 해당 클래스가 존재하지 않으면 에러
- 위치 중요! - <?xml version ..>과 <루트컨테이너> 태그 사이에 선언되어 한다.

자바 코드	FXML 태그
import javafx.scene.layout.HBox;	<?import javafx.scene.layout.HBox?>
import javafx.scene.control.*;	<?import javafx.scene.control.*?>

```

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.layout.HBox?>
<?import javafx.scene.control.*?>

<루트컨테이너 xmlns:fx="http://javafx.com/fxml" >
    ...
</루트컨테이너>
    
```

##### (2) 태그 선언

- FXML 태그는 < 와 > 사이에 태그 이름 작성
- 반드시 시작 태그가 있으면 끝 태그도 있어야

자바 코드	FXML
Button button = new Button(); button.setText("확인");	<Button > <text>확인</text> </Button>

##### (3) 속성 선언

- 속성값은 큰따옴표(") 또는 작은따옴표(')로 반드시 감싸야 한다.

자바 코드	FXML (Setter 태그)	FXML (Setter 속성)
Button button = new Button(); button.setText("확인");	<Button > <text>확인</text> </Button>	<Button text="확인"/>

##### (4) 객체 선언

- | <클래스 속성="값"/> |



자바 코드	FXML
<pre>HBox hbox = new HBox(); hbox.setPadding(new Insets(10,10,10,10));</pre>	<pre>&lt;HBox&gt;   &lt;padding&gt;     &lt;Insets top="10" right="10"             bottom="10" left="10"/&gt;   &lt;/padding&gt; &lt;/HBox&gt;</pre>

■ | <클래스 fx:value="값"/> |

자바 코드	FXML
<pre>String.valueOf("Hello, World!"); Integer.valueOf("1"); Double.valueOf("1.0"); Boolean.valueOf("false");</pre>	<pre>&lt;String fx:value="Hello, World!"/&gt; &lt;Integer fx:value="1"/&gt; &lt;Double fx:value="1.0"/&gt; &lt;Boolean fx:value="false"/&gt;</pre>

■ | <클래스 fx:constant="상수"/> |

자바 코드	FXML
<pre>Button button = new Button(); button.setMaxWidth(     Double.MAX_VALUE );</pre>	<pre>&lt;Button&gt;   &lt;maxWidth&gt;     &lt;Double fx:constant="MAX_VALUE"/&gt;   &lt;/maxWidth&gt; &lt;/Button&gt;</pre>

■ | <클래스 fx:factory="정적메소드"/> |

자바 코드	FXML
<pre>ComboBox combo = new ComboBox(); combo.setItems(     FXCollections.observableArrayList(         "공개", "비공개"     ) );</pre>	<pre>&lt;ComboBox&gt;   &lt;items&gt;     &lt;FXCollections fx:factory="observableArrayList"&gt;       &lt;String fx:value="공개"/&gt;       &lt;String fx:value="비공개"/&gt;     &lt;/FXCollections&gt;   &lt;/items&gt; &lt;/ComboBox&gt;</pre>

### 17.3.5. FXML 로딩과 Scene 생성

- FXML 로딩: FXML 파일을 읽어 들여 선언된 내용을 객체화하는 것
- Scene 생성: FXML 로딩 후 얻은 루트 컨테이너는 Scene을 생성할 때 매개값으로 사용

```
Parent root = FXMLLoader.load(getClass().getResource("xxx.fxml"));
HBox hbox = (HBox) FXMLLoader.load(getClass().getResource("xxx.fxml")); //루트 태그가 <HBox>인 경우

Scene scene = new Scene(root);
```

### 17.3.6. JavaFX Scene Builder

- 드래그 앤 드롭 방식의 WYSIWYG 디자인 툴

- E(fx)clipse 플러그인 설치하면 더 편리하게 사용가능

- 주의할 점은 PC에 JavaFX Scene Builder를 먼저 설치한 다음 e(fx)clipse 플러그인을 설치해야 한다.

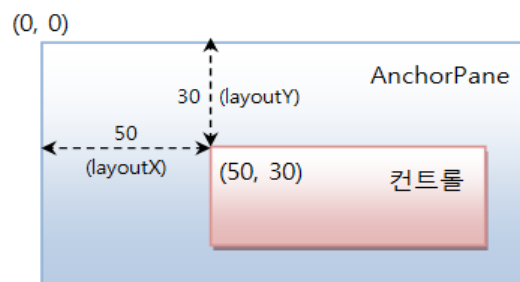
## 17.4 JavaFX 컨테이너

- 레이아웃 작성시 다양한 컨트롤들을 쉽게 배치하도록 해주는 역할
- javafx.scene.layout 패키지에 속함
- 컨테이너의 종류

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 않는 레이아웃
StackPane	컨트롤을 겹쳐 배치하는 레이아웃
TilePane	그리드로 배치하되 고정된 셀의 크기를 갖는 레이아웃
HBox	수평으로 배치하는 레이아웃
VBox	수직으로 배치하는 레이아웃

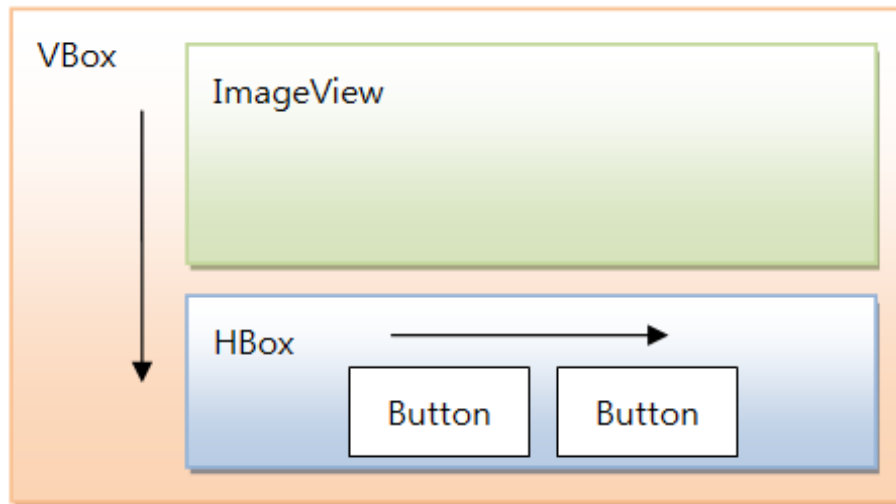
### 17.4.1. AnchorPane 컨테이너

AnchorPane 컨테이너는 AnchorPane 의 좌상단(0, 0)을 기준으로 컨트롤을 좌표로 배치한다. 컨트롤 좌표는 좌상단 (layoutX, layoutY) 값을 말하는데 (0,0) 에서 떨어진 거리이다.



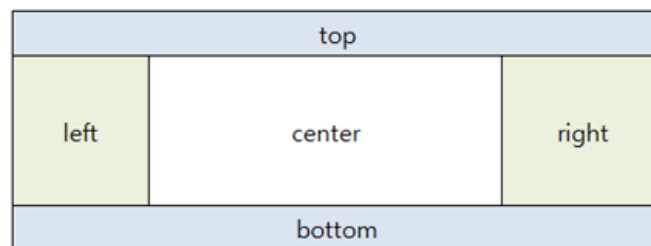
### 17.4.2. HBox와 VBox 컨테이너

- 수평과 수직으로 컨트롤을 배치하는 컨테이너
- 자식 컨트롤의 크기 재조정에 쓰임
  - HBox는 컨트롤의 높이 확장하고, 컨트롤의 폭은 유지
  - VBox는 컨트롤의 폭 확장하고 컨트롤의 높이는 유지

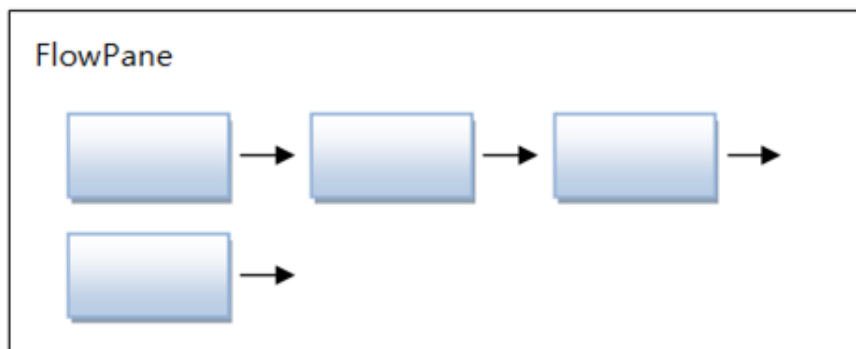


### 17.4.3. BorderPane 컨테이너

- top, bottom, left, right, center 셀에 컨트롤 배치하는 컨테이너
- 각 셀에는 하나의 컨트롤 또는 컨테이너만 배치
- top, bottom, left, right에 배치하지 않으면 center에 배치된 컨트롤이 사방으로 자동 확장



### 17.4.4. FlowPane 컨테이너



### 17.4.5. TilePane 컨테이너



#### 17.4.6. GridPane 컨테이너



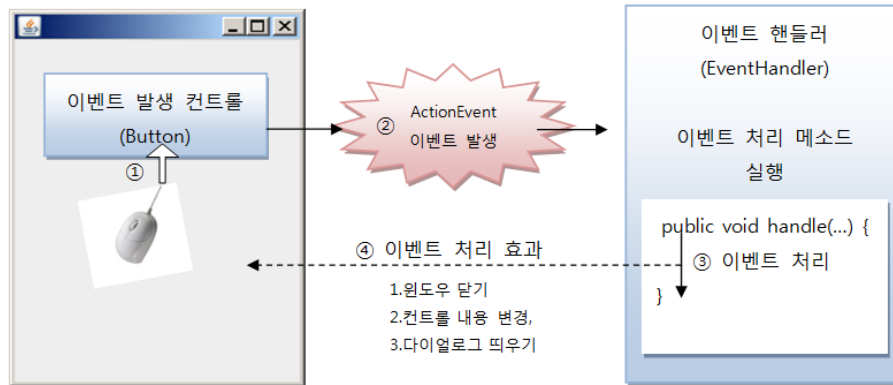
#### 17.4.7. StackPane 컨테이너



### 17.5 JavaFX 이벤트 처리

#### 17.5.1 이벤트 핸들러(EventHandler)

- 이벤트 발생 컨트롤과 이벤트 핸들러를 분리하는 위임형 방식



```

Button button = new Button();
button.setOnAction(new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent event) {...}
});

TableView tableView = new TableView();
tableView.setOnMouseClicked(new EventHandler<MouseEvent>() {
    @Override
    public void handle(MouseEvent event) {...}
});

stage.setOnCloseRequest(new EventHandler<WindowEvent>() {
    @Override
    public void handle(WindowEvent event) {...}
});
    
```

### 17.5.2 FXML 컨트롤러(Controller)

- FXML 레이아웃은 FXML 파일당 별도의 컨트롤러(Controller) 지정해서 이벤트를 처리할 수 있기 때문에 FXML 레이아웃과 이벤트 처리 코드를 완전히 분리할 수 있다.

#### (1) fx:controller 속성과 컨트롤러 클래스

- UI 컨트롤에서 발생하는 이벤트를 컨트롤러가 처리

```

<루트컨테이너 xmlns:fx="http://javafx.com/fxml"
    fx:controller="packageName.ControllerName">
    ...
</루트컨테이너>

public class ControllerName implements Initializable {
    @Override
    public void initialize(URL location, ResourceBundle resources) {}
}
    
```

#### (2) fx:id 속성과 @FXML 컨트롤 주입

- 컨트롤러의 @FXML 어노테이션이 적용된 필드에 자동 주입.
- fx:id 속성값과 필드명은 동일해야 함

```
<HBox xmlns:fx="http://javafx.com/fxml"
      fx:controller="sec05.exam02_fxml_controller.RootController"
      prefHeight="50.0" refWidth="200.0"
      alignment="CENTER" spacing="20.0">
  <children>
    <Button fx:id="btn1" text="버튼1" />
    <Button fx:id="btn2" text="버튼2" />
    <Button fx:id="btn3" text="버튼3" />
  </children>
</HBox>

public class ControllerName implements Initializable {
    @FXML private Button btn1;
    @FXML private Button btn2;
    @FXML private Button btn3;
    @Override
    public void initialize(URL location, ResourceBundle resources) {}
}
```

### (3) EventHandler 등록

- 컨트롤에서 발생하는 이벤트를 처리하려면 컨트롤러의 initialize() 메소드에서 EventHandler를 생성하고 등록해야 한다.

### (4) 이벤트 처리 메소드 매핑 <-- 다른 방법

- 컨트롤러에서 EventHandler를 생성하지 않고도 바로 이벤트 처리 메소드와 연결할 수 있는 방법이 있다.
- Button 컨트롤을 작성할 때 onAction 속값으로 "#메소드명"을 주면 내부적으로 EventHandler 객체가 생성되기 때문에 컨트롤러에서는 해당 메소드만 작성하면 된다.

```
// FXML 파일
<Button fx:id="btn1" text="버튼1" onAction="#handleBtn1Action"/>

// Controller 클래스
public void handleBtn1Action(ActionEvent event) { ... }
```

[root.fxml] FXMLLoader가 FXML 파일을 로딩할 때, 태그로 선언된 컨트롤 객체와 컨트롤러 객체를 함께 생성한다.

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.layout.*?>
<?import javafx.scene.control.*?>

<HBox xmlns:fx="http://javafx.com/fxml"
      fx:controller="sec05.exam02_fxml_controller.RootController"
      prefHeight="50.0" prefWidth="200.0"
      alignment="CENTER" spacing="20.0">
```

```
<children>
  <Button fx:id="btn1" text="버튼1" />
  <Button fx:id="btn2" text="버튼2" />
  <Button fx:id="btn3" text="버튼3" />
</children>
</HBox>
```

[RootController.java] fx:id 속성을 가진 컨트롤들은 컨트롤러의 @FXML 어노테이션이 적용된 필드에 자동 주입된다.

```
package sec05.exam02_fxml_controller;

import java.net.URL;
import java.util.ResourceBundle;

import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Button;

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) {
        btn1.setOnAction(new EventHandler<ActionEvent>() { // 직접 EventHandler 생성 후 등록
            @Override
            public void handle(ActionEvent event) {
                handleBtn1Action(event);
            }
        });
        btn2.setOnAction(event -> handleBtn2Action(event)); // 람다식 이용
        btn3.setOnAction(event -> handleBtn3Action(event));
    }

    public void handleBtn1Action(ActionEvent event) {
        System.out.println("버튼1 클릭");
    }

    public void handleBtn2Action(ActionEvent event) {
        System.out.println("버튼2 클릭");
    }

    public void handleBtn3Action(ActionEvent event) {
        System.out.println("버튼3 클릭");
    }
}
```

## 17.6 JavaFX 속성 감시와 바인딩

### 17.6.1 속성 감시

- 컨트롤의 속성값 변화 감시하는 리스너(ChangeListener)를 설정할 수 있다.

- 속성값에 변화가 생기면 ChangeListener의 changed() 메소드가 자동으로 실행된다.
- JavaFX 컨트롤 속성의 구성
  - Setter
  - Getter
  - Property 객체 리턴하는 메소드

```
// 예) text 속성은 getText(), setText(), 그리고 StringProperty를 리턴하는 textProperty()를 가지고 있다.
private StringProperty text = new SimpleStringProperty();
public void setText(String newValue) { text.set(newValue); }
public String getText() { return text.get(); }
public StringProperty textProperty() { return text; }

// text 속성값을 감시하는 ChangeListener를 설정하는 코드이다.
textProperty().addListener(new ChangeListener<String>() {
    @Override
    public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {
    }
});
```

[RootController.java] 속성 감시 리스너 설정

```
package sec06.exam01_property_listener;

import java.net.URL;
import java.util.ResourceBundle;

import javafx.beans.value.ChangeListener;
import javafx.beans.value.ObservableValue;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Label;
import javafx.scene.control.Slider;
import javafx.scene.text.Font;

public class RootController implements Initializable {
    @FXML
    private Slider slider;
    @FXML
    private Label label;

    @Override
    public void initialize(URL location, ResourceBundle resources) {
        slider.valueProperty().addListener(new ChangeListener<Number>() {
            @Override
            public void changed(ObservableValue<? extends Number> observable, Number
oldValue, Number newValue) {
                label.setFont(new Font(newValue.doubleValue()));
            }
        });
    }
}
```

## 17.6.2 속성 바인딩

- 두 컨트롤의 속성을 서로 연결하는 것



- 바인드 된 속성들은 하나가 변경되면 자동적으로 다른 하나도 변경
- 단방향 바인드 - bind()
- 양방향 바인드 - bindBidirectional()
- 언바인드 - unbind(), 바인드 해제

```
TextArea textArea1 = new TextArea();
TextArea textArea2 = new TextArea();
textArea2.textProperty().bind(textArea1.textProperty());
textArea2.textProperty().bindBidirectional(textArea1.textProperty());
Bindings.bindBidirectional(textArea1.textProperty(),textArea2.textProperty());
textArea2.textProperty().unbind(); // 단방향 해제
```

### 17.6.3 Bindings 클래스

- 속성 연산하거나, 다른 타입으로 변환 후 바인딩하는 기능 제공

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, greaterThanOrEqual	속성값이 어떤 값보다 큰지, 같거나 큰지를 조사해서 true/false 로 변환하여 바인딩함
lessThan, lessThanOrEqual	속성값이 어떤 값보다 적거나, 같거나 적은지를 조사해서 true/false 로 변환하여 바인딩함
equal, notEquals	속성값이 어떤 값과 같은지, 다른지를 조사해서 true/false 로 변환하여 바인딩함
equalsIgnoreCase, notEqualIgnoreCase	대소문자와 상관없이 속성값이 어떤 문자열과 같은지, 다른지를 조사해서 true/false 로 변환하여 바인딩함
isEmpty, isEmpty	속성값이 비어있는지, 아닌지를 조사해서 true/false 로 변환하여 바인딩함
isNull, isNotNull	속성값이 null 또는 not null 인지를 조사해서 true/false 로 변환하여 바인딩함
length	속성값이 문자열일 경우 문자수를 얻어 바인딩함
size	속성 타입이 배열, List, Map, Set 일 경우 요소수를 얻어 바인딩함
and, or	속성값이 boolean 일 경우, 논리곱, 논리합을 얻어 바인딩함
not	속성값이 boolean 일 경우, 반대값으로 바인딩함
convert	속성값을 문자열로 변환해서 바인딩함
valueAt	속성이 List, Map 일 경우 해당 인덱스 또는 키의 값을 얻어 바인딩함

[RootController.java] 연산된 속성 바인딩

```
package sec06.exam03_bindings;

import java.net.URL;
import java.util.ResourceBundle;

import javafx.beans.binding.Bindings;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.layout.AnchorPane;
import javafx.scene.shape.Circle;

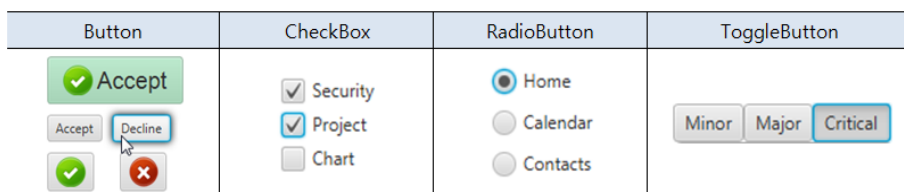
public class RootController implements Initializable {
    @FXML
    private AnchorPane root;
    @FXML
```

```
private Circle circle;

@Override
public void initialize(URL location, ResourceBundle resources) {
    circle.centerXProperty().bind(Bindings.divide(root.widthProperty(), 2));
    circle.centerYProperty().bind(Bindings.divide(root.heightProperty(), 2));
}
}
```

## 17.7 JavaFX 컨트롤

### 17.7.1 버튼 컨트롤



[root.fxml] 버튼 컨트롤 배치의 이벤트 처리

```
<?xml version="1.0" encoding="UTF-8"?>

<?import java.lang.*?>
<?import javafx.scene.layout.*?>
<?import javafx.geometry.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.image.*?>

<BorderPane prefHeight="150.0" prefWidth="420.0" xmlns:fx="http://javafx.com/fxml/1"
    xmlns="http://javafx.com/javafx/8" fx:controller="sec07.exam01_button.RootController">
    <padding>
        <Insets bottom="10.0" left="10.0" right="10.0" top="10.0" />
    </padding>

    <center>
        <HBox alignment="CENTER" prefHeight="100.0" prefWidth="400.0" spacing="10">
            <children>
                <VBox alignment="CENTER_LEFT" prefHeight="100.0" prefWidth="100.0" spacing="20.0">
                    <children>
                        <CheckBox fx:id="chk1" onAction="#handleChkAction" text="안경" />
                        <CheckBox fx:id="chk2" onAction="#handleChkAction" text="모자" />
                    </children>
                </VBox>

                <ImageView fx:id="checkImageView" fitWidth="100.0" preserveRatio="true">
                    <image><Image url="@images/geek.gif" /></image>
                </ImageView>

                <Separator orientation="VERTICAL" prefHeight="100.0" />

                <VBox alignment="CENTER_LEFT" prefHeight="100" prefWidth="150" spacing="20.0">
                    <!-- <fx:define><ToggleGroup fx:id="group" /></fx:define> -->
                    <children>
                        <RadioButton fx:id="rad1" text="BubbleChart" userData="BubbleChart">
                            <toggleGroup>
                                <ToggleGroup fx:id="group" />
                            </toggleGroup>
                        </RadioButton>
                    </children>
                </VBox>
            </children>
        </HBox>
    </center>
</BorderPane>
```

```

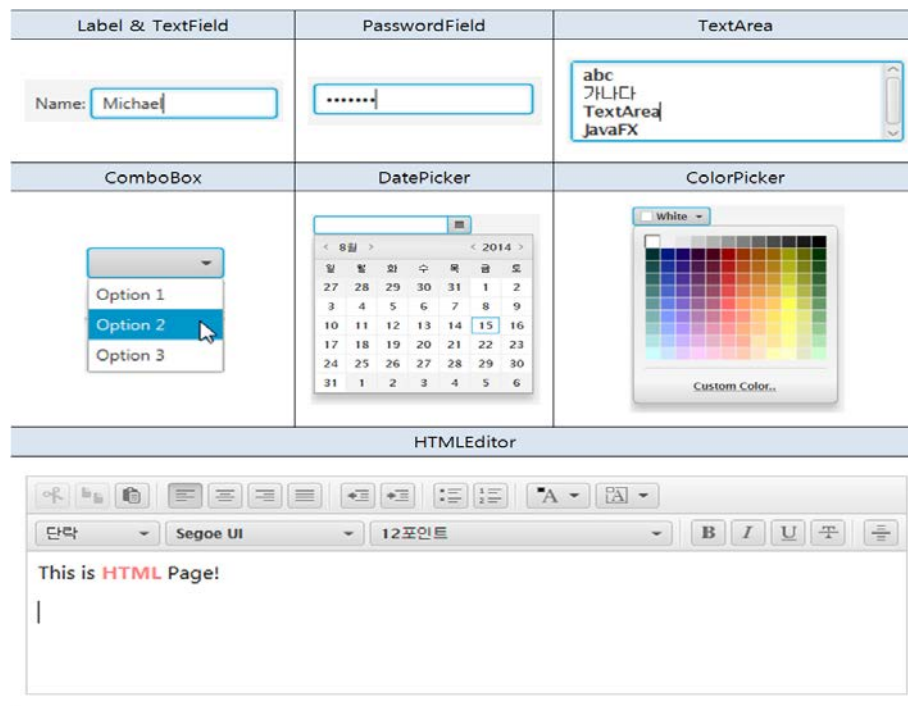
        </RadioButton>
        <RadioButton fx:id="rad2" selected="true" text="BarChart" toggleGroup="$group"
userData="BarChart" />
        <RadioButton fx:id="rad3" text="AreaChart" toggleGroup="$group" userData="AreaChart"
/>
    </children>
</VBox>

<ImageView fx:id="radioImageView" fitWidth="100.0" preserveRatio="true">
    <image><Image url="@images/BarChart.png" /></image>
</ImageView>
</children>
</HBox>
</center>

<bottom>
    <Button fx:id="btnExit" onAction="#handleBtnExitAction" BorderPane.alignment="CENTER">
        <graphic>
            <ImageView>
                <image>
                    <Image url="@images/exit.png" />
                </image></ImageView>
            </graphic>
            <BorderPane.margin><Insets top="20.0" /></BorderPane.margin>
        </Button>
    </bottom>
</BorderPane>

```

## 17.7.2 입력 컨트롤



[root.fxml] 입력 컨트롤 배치

```

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.text.*?>

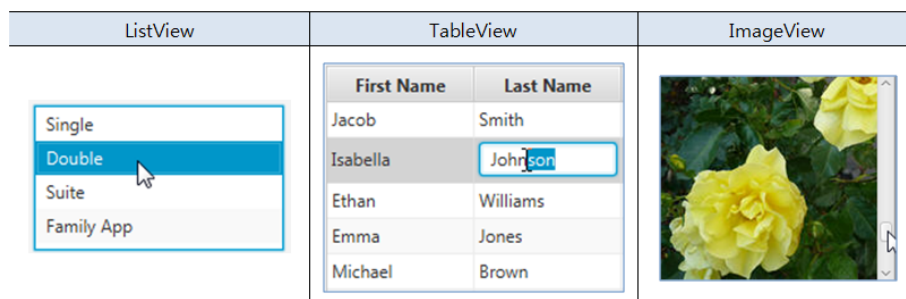
```

```
<?import javafx.scene.shape.*?>
<?import javafx.scene.web.*?>
<?import javafx.geometry.*?>
<?import javafx.scene.image.*?>
<?import java.lang.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>
<?import javafx.collections.*?>

<AnchorPane xmlns:fx="http://javafx.com/fxml" fx:controller="sec07.exam02_input.RootController"
    prefHeight="380" prefWidth="485" >
    <children>
        <Label layoutX="22.0" layoutY="36.0" text="제목" />
        <TextField fx:id="txtTitle" layoutX="84.0" layoutY="32.0" prefHeight="23.0" prefWidth="375.0" />
        <Label layoutX="22.0" layoutY="69.0" text="비밀번호" />
        <PasswordField fx:id="txtPassword" layoutX="86.0" layoutY="65.0" prefHeight="23.0" prefWidth="132.0" />
        <Label layoutX="22.0" layoutY="104.0" text="공개" />
        <ComboBox fx:id="comboPublic" layoutX="86.0" layoutY="100.0" prefHeight="23.0"
            prefWidth="132.0" promptText="선택하세요" >
            <items>
                <FXCollections fx:factory="observableArrayList">
                    <String fx:value="공개"/>
                    <String fx:value="비공개"/>
                </FXCollections>
            </items>
        </ComboBox>
        <Label layoutX="240.0" layoutY="104.0" text="게시종료" />
        <DatePicker fx:id="dateExit" layoutX="296.0" layoutY="100.0" promptText="날짜를 선택하세요" />
        <Label layoutX="22.0" layoutY="135.0" text="내용" />
        <TextArea fx:id="txtContent" layoutX="22.0" layoutY="154.0" prefHeight="132.0" prefWidth="440.0" />
        <Separator layoutX="13.0" layoutY="320" prefHeight="0.0" prefWidth="457.0" />
        <Button fx:id="btnReg" layoutX="189.0" layoutY="340" text="등록" onAction="#handleBtnRegAction"/>
        <Button fx:id="btnCancel" layoutX="252.0" layoutY="340" text="취소" onAction="#handleBtnCancelAction"/>
    </children>
</AnchorPane>
```

### 17.7.3 뷰 컨트롤

- 목록 형태로 보여주는 ListView
- 테이블 형태로 보여주는 TableView
- 이미지를 보여주는 ImageView



```
<ImageView fitWidth="폭" fitHeight="높이" preserveRatio="true"/>
```

```
<ListView prefWidth="폭" prefHeight="높이"/>
```

```
<TableView prefHeight="100" prefWidth="300">
  <columns>
    <TableColumn prefWidth="150" text="스마트폰" />
    <TableColumn prefWidth="150" text="이미지" />
  </columns>
</TableView>
```

[root.fxml] 뷰 컨트롤 배치


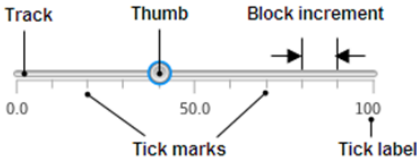
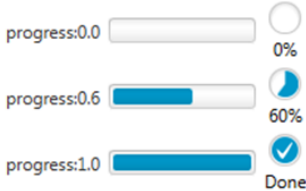
```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.text.*?>
<?import javafx.scene.shape.*?>
<?import javafx.scene.web.*?>
<?import javafx.geometry.*?>
<?import javafx.scene.image.*?>
<?import java.lang.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>
<?import javafx.collections.*?>

<AnchorPane xmlns:fx="http://javafx.com/fxml" fx:controller="sec07.exam03_view.RootController"
  prefHeight="180.0" prefWidth="500.0">
  <children>
    <Label layoutX="11.0" layoutY="9.0" text="ListView" />
    <ListView fx:id="listView" layoutX="10.0" layoutY="30.0" prefHeight="100.0" prefWidth="100.0" />
    <Label layoutX="125.0" layoutY="9.0" text="TableView" />
    <TableView fx:id="tableView" layoutX="120.0" layoutY="30.0" prefHeight="100.0" prefWidth="290.0">
      <columns>
        <TableColumn prefWidth="100.0" text="스마트폰" />
        <TableColumn prefWidth="100.0" text="이미지" />
      </columns>
    </TableView>
    <Label layoutX="425.0" layoutY="9.0" text="ImageView" />
    <ImageView fx:id="imageView" fitHeight="100.0" fitWidth="60.0" layoutX="430.0" layoutY="30.0"
      preserveRatio="true" />
    <Button layoutX="190.0" layoutY="145.0" onAction="#handleBtnOkAction" text="확인" />
    <Button layoutX="260.0" layoutY="145.0" onAction="#handleBtnCancelAction" text="취소" />
  </children>
</AnchorPane>
```

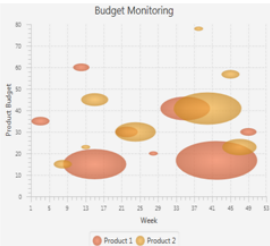
#### 17.7.4 미디어 컨트롤

- 비디오를 재생할 수 있는 MediaView 컨트롤
- 볼륨 조절 및 재생 위치 조절을 위한 slider 컨트롤
- 현재 진행 상태 보여주는 ProgressBar, ProgressIndicator

MediaView	
	
Slider	ProgressBar 와 ProgressIndicator
	

## 17.7.5 차트 컨트롤

- javafx.scene.chart 패키지에 포함

PieChart	LineChart	AreaChart
		
BarChart	BubbleChart	ScatterChart
		

## 17.8 JavaFX 메뉴바와 툴바

### 17.8.1. MenuBar 컨트롤

- MenuBar에는 Menu들이 배치
- Menu에는 메뉴 아이템 추가
  - MenuItem,

- CheckMenuItem,
  - RadioMenuItem,
  - CustomMenuItem,
  - SeparatorMenuItem
  - Menu(서브 메뉴)
- 계층적인 작업 선택 기능 구현에 주로 쓰임

### 17.8.2. Toolbar 컨트롤

- 빠르게 작업을 선택하고 싶을 때 사용
- Toolbar 컨트롤은 UI 컨트롤이면서 컨테이너
- Button이 추가되지만, ComboBox와 같은 다른 컨트롤도 배치

## 17.9 JavaFX 다이얼로그

### 17.9.1. FileChooser, DirectoryChooser

- XXXChooser 는 컨트롤이 아니라 FXML 에서 선언 불가
- 모달 다이얼로그 - 버튼 클릭하기 전에는 소유자 윈도우 사용 불가

### 17.9.2. Popup

- 투명한 컨테이너 제공하는 모달리스 다이얼로그
- 윈도우의 기본 장식(아이콘, 제목, 최소화 및 복원 버튼, 닫기 버튼) 없음
- 용도
- 컨트롤의 툴팁(tooltip), 메시지 통지(notification), 드롭 다운 박스(drop down boxes)
- Popup의 내용은 자바 코드로 작성하거나, FXML 파일로 작성
- Popup은 최상위 윈도우
- 소유자 윈도우를 닫거나, hide()를 호출하면 닫힘
  - setAutoHide(true): 다른 윈도우로 포커스 이동시 Popup은 자동 닫힘

### 17.9.3. 커스텀 다이얼로그

- 다양한 내용의 다이얼로그를 만들고 싶다면 Stage로 직접 생성
- StageStyle 열거 상수와 윈도우 스타일

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

```

Stage dialog = new Stage(StageStyle.UTILITY);
dialog.initModality(Modality.WINDOW_MODAL);
dialog.initOwner(primaryStage);
dialog.setTitle("확인");

Parent parent = FXMLLoader.load(getClass().getResource("custom_dialog.fxml"));
Scene scene = new Scene(parent);

dialog.setScene(scene);
dialog.setResizable(false);
dialog.show();

```

#### 17.9.4. 컨트롤러에서 primaryStage 사용

- 컨트롤러에서 다이얼로그 실행
  - 소유자 윈도우로 primaryStage 필요
- 컨트롤러에서 primaryStage 얻는 방법
  - 메인 클래스에서 전달하는 방법
  - 컨테이너 또는 컨트롤로부터 얻는 방법
    - initialize() 메소드 안에서는 사용 불가

### 17.10 JavaFX CSS 스타일

#### 17.10.1. 인라인(inline) 스타일

- 컨테이너 또는 컨트롤의 style 속성값으로 직접 CSS 정의
- 쉽고, 빠르게 모양과 색상 변경

#### 17.10.2. 외부 CSS 파일

- 인라인 스타일 문제점
  - 동일한 스타일을 적용하는 컨트롤 많을수록 중복 코드가 많이 늘어남
  - FXML과 CSS가 뒤섞여 추후 유지 보수가 어려움
- 선택자:
  - 외부 CSS 파일
    - 스타일 적용할 컨테이너와 컨트롤 선택해주는 선택자 필요
  - 선택자의 종류



- Type 선택자: Type { 속성:값; 속성:값; ... }
- id 선택자: #id { 속성:값; 속성:값; ... }
- class 선택자: .class { 속성:값; 속성:값; ... }
- Type 선택자와 class 선택자 조합
- 상태별 선택자

■ CSS 파일 적용

- Scene에 추가하여 Scene 내부의 모든 컨테이너와 컨트롤에 적용

### 17.10.3. border 속성

■ 컨테이너 및 컨트롤의 경계선의 스타일 설정

■ 세부 속성

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

### 17.10.4. background 속성

■ 컨테이너 및 컨트롤의 배경 스타일을 설정한다.

■ 세부 속성

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)

### 17.10.5. font 속성

■ 세부 속성

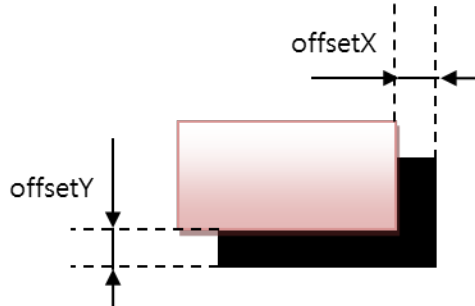
속성	설명
-fx-font-size	폰트 크기
-fx-font-family	폰트 종류
-fx-font-weight	폰트 굵기(bold)
-fx-text-fill	폰트 색상(단색, 선형 그라디언트, 원형그라디언트)

### 17.10.6. shadow 효과

■ blur-type : gaussian, one-pass-box, three-pass-box, two-pass-box

■ radius: blur kernel의 반지름, 0.0~127.0 사이의 값 , 기본값 10

- spread, choke: 그림자의 spread와 choke, 0.0~1.0 사이의 값. 기본값은 0.0
- offsetX, offsetY: 그림자의 편차



### 17.10.7. 화면 스킨 입히기

- 로그인 폼 화면을 만들고 CSS를 적용해서 스킨을 입힌다.

[root.fxml] 로그인 폼 배치하기

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.layout.*?>
<?import javafx.scene.control.*?>

<AnchorPane styleClass="root" xmlns:fx="http://javafx.com/fxml"
    prefHeight="194.0" prefWidth="300.0">
    <children>
        <Label id="welcome-text" layoutX="40.0" layoutY="14.0" text="Welcome" />
        <Label layoutX="42.0" layoutY="80.0" text="아이디" />
        <Label layoutX="42.0" layoutY="118.0" text="패스워드" />
        <TextField layoutX="120.0" layoutY="76.0" />
        <PasswordField layoutX="120.0" layoutY="114.0" />
        <Button layoutX="97.0" layoutY="158.0" styleClass="button" text="로그인" />
        <Button layoutX="164.0" layoutY="158.0" styleClass="button" text="취소" />
    </children>
</AnchorPane>
```

[app.css] CSS 파일

```
.root {
    -fx-background-image: url("images/background.jpg");
}

Label {
    -fx-font-size: 12px;
    -fx-font-weight: bold;
    -fx-text-fill: #333333;
}

#welcome-text {
    -fx-font-size: 35px;
    -fx-font-family: "Arial Black";
    -fx-text-fill: linear-gradient(darkgray, lightgray);
    -fx-effect: innershadow( three-pass-box , rgba(0,0,0,0.7) , 3, 0 , 2 , 2.1 );
}
```

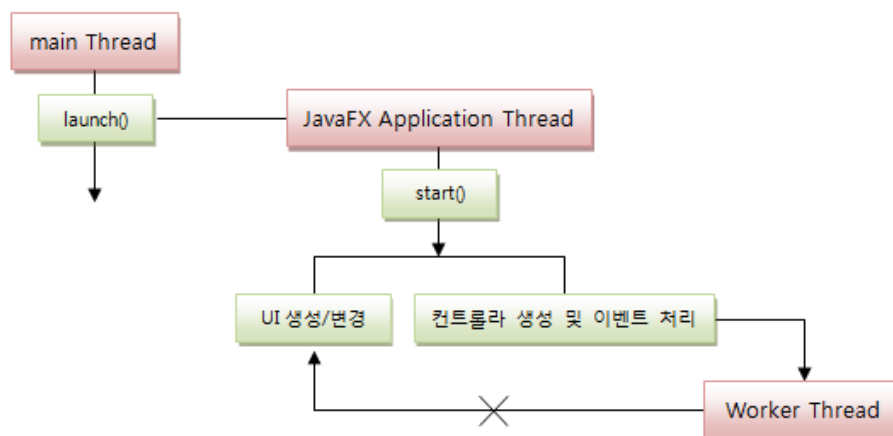
```
.button {
    -fx-text-fill: white;
    -fx-font-family: "Arial Narrow";
    -fx-font-weight: bold;
    -fx-background-color: linear-gradient(#61a2b1, #2A5058);
    -fx-effect: dropshadow( three-pass-box , rgba(0,0,0,0.6) , 10, 0 , 2 , 2 );
}

.button:hover {
    -fx-background-color: linear-gradient(#2A5058, #61a2b1);
}

.button:pressed {
    -fx-background-color: linear-gradient(yellow, #61a2b1);
}
```

## 17.11 JavaFX 스레드 동시성

- JavaFX UI는 스레드에 안전하지 않음
  - UI를 생성하고 변경하는 작업은 JavaFX Application Thread가 담당
  - 다른 작업 스레드들은 UI를 생성하거나 변경할 수 없음

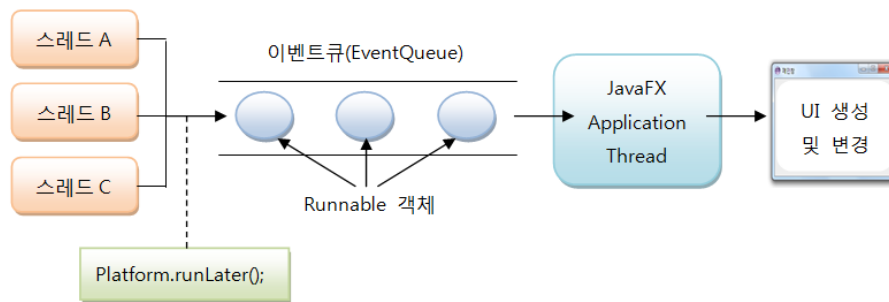


- JavaFX Application Thread는 시간을 요하는 작업 하지 않도록
  - 시간을 요하는 작업을 하게 되면 UI는 멈춰있는 상태. Ex) 파일 읽고 쓰기, 네트워크상에서 데이터를 주고 받을 경우
  - 다른 작업 스레드 생성해 처리

### 17.11.1. Platform.runLater() 메소드

- 작업 스레드는 UI 를 변경할 수 없음
- UI 변경 필요한 경우 Runnable 로 생성해 실행

## JAVA 프로그래밍 (프로그래밍 언어 활용)



### [root.fxml] 컨트롤 배치

```

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.layout.*?>
<?import javafx.scene.control.*?>

<AnchorPane xmlns:fx="http://javafx.com/fxml" prefHeight="100.0" prefWidth="200.0"
    fx:controller="sec11.exam01_runlater.RootController">
    <children>
        <Label fx:id="lblTime" alignment="CENTER" layoutX="25.0" layoutY="15.0"
            prefHeight="35.0" prefWidth="150.0"
            style="-fx-background-color: black; -fx-text-fill: yellow;
                -fx-font-size: 20; -fx-background-radius: 10;"
            text="00:00:00" />
        <Button fx:id="btnStart" layoutX="46.0" layoutY="63.0" text="시작" />
        <Button fx:id="btnStop" layoutX="110.0" layoutY="63.0" text="멈춤" />
    </children>
</AnchorPane>
  
```

### [RootController.java] 컨트롤러

```

package sec11.exam01_runlater;

import java.net.URL;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.ResourceBundle;

import javafx.application.Platform;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Button;
import javafx.scene.control.Label;

public class RootController implements Initializable {
    @FXML private Label lblTime;
    @FXML private Button btnStart;
    @FXML private Button btnStop;

    private boolean stop;

    @Override
    public void initialize(URL location, ResourceBundle resources) {
        btnStart.setOnAction(event -> handleBtnStart(event));
        btnStop.setOnAction(event -> handleBtnStop(event));
    }

    public void handleBtnStart(ActionEvent e) {
        stop = false;
    }
  
```

```

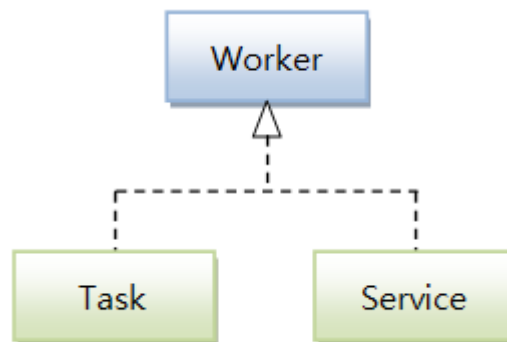
        Thread thread = new Thread() {
            @Override
            public void run() {
                SimpleDateFormat sdf = new SimpleDateFormat("HH:mm:ss");
                while(!stop) {
                    String strTime = sdf.format(new Date());
                    Platform.runLater(()->{ // UI 변경 작업
                        lblTime.setText(strTime);
                    });
                    try { Thread.sleep(100); } catch (InterruptedException e) {}
                }
            }
        };
        thread.setDaemon(true);
        thread.start();
    }

    public void handleBtnStop(ActionEvent e) {
        stop = true;
    }
}

```

### 17.11.2. Task 클래스

- javafx.concurrent 패키지가 제공하는 스레드 동시성 API



- Worker 인터페이스: 작업 스레드가 UI 변경할 때 Task와 Service에서 공통적으로 사용할 수 있는 메소드 제공
- Task 추상 클래스: JavaFX 애플리케이션에서 비동기 작업을 표현한 클래스
- Service 추상 클래스: Task를 간편하게 시작, 취소, 재시작 할 수 있는 기능 제공하는 클래스

#### (1) Task 생성

- 작업 스레드에서 실행되는 하나의 작업을 표현한 추상 클래스이다.
- 하나의 작업을 정의할때에는 Task를 상속해서 클래스를 생성한 후, call() 메소드를 재정의 하면 된다.

```
// 리턴값(작업 결과)이 Integer
```

```
Task<Integer> task = new Task<Integer>() {
    @Override
    protected Integer call() throws Exception {
        int result = 0;
        // 작업 코드 ...
        return result;
    }
}

// 리턴값이 없는 작업
Task<Void> task = new Task<Void>() {
    @Override
    protected Void call() throws Exception {
        // 작업 코드 ...
        return null;
    }
}
```

## (2) Task 실행

- Task를 작업 스레드에서 실행하려면 작업 스레드를 생성할 때 매개값으로 Task를 전달하면 된다.

```
// 작업 스레드에서 Task를 처리
Task<Integer> task = new Task<Integer>() { ... }
Thread thread = new Thread(task);
thread.setDaemon(true);
thread.start();

// 스레드풀에서 Task를 처리
Task<Integer> task = new Task<Integer>() { ... };
executorService.submit(task);
```

## (3) Task 취소

- Task는 cancel() 메소드가 호출되면 언제든지 실행을 멈출 수 있어야 한다.
- Task는 주기적으로 isCancelled() 메소드를 호출해서 작업이 취소되었는지 확인하고, 작업을 종료해야 한다.

```
Task<Void> task = new Task<Void>() {
    @Override
    protected Void call() throws Exception {
        while (...) {
            if (isCancelled()) { break; }
            // 작업코드...
        }
        return null;
    }
}
```

#### (4) UI 변경

- Task의 call() 메소드는 작업 스레드상에서 호출되기 때문에 UI 변경 코드를 작성할 수 없다.
- 대신 call() 메소드 내부에서는 updateProgress(), updateMessage() 메소드를 호출할 수 있는데, 이 메소드들은 JavaFX Application Thread로 하여금 UI를 업데이트하도록 해준다.
- 복잡한 UI 변경이 필요한 경우에는 call() 메소드 내에서 Platform.runLater(new Runnable(){...}) 메소드를 호출하면 된다.

[RootController.java]

```
package sec11.exam02_task;

import java.net.URL;
import java.util.ResourceBundle;

import javafx.concurrent.Task;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.ProgressBar;

public class RootController implements Initializable {
    @FXML private ProgressBar progressBar;
    @FXML private Label lblWorkDone;
    @FXML private Button btnStart;
    @FXML private Button btnStop;

    private Task<Void> task;

    @Override
    public void initialize(URL location, ResourceBundle resources) {
        btnStart.setOnAction(event->handleBtnStart(event));
        btnStop.setOnAction(event->handleBtnStop(event));
    }

    public void handleBtnStart(ActionEvent e) {
        task = new Task<Void>() {
            @Override
            protected Void call() throws Exception {
                for(int i=0; i<=100; i++) {
                    if(isCancelled()) {
                        updateMessage("취소됨");
                        break;
                    }
                    updateProgress(i, 100); // Task의 progress 속성 업데이트
                    updateMessage(String.valueOf(i));
                    try {Thread.sleep(100); } catch (InterruptedException e) {
                        if(isCancelled()) {
                            updateMessage("취소됨");
                            break;
                        }
                    }
                }
            }
        };
        return null;
    }
};
```

```

        progressBar.progressProperty().bind(task.progressProperty()); // 속성 바인딩
        lblWorkDone.textProperty().bind(task.messageProperty());

        Thread thread = new Thread(task);
        thread.setDaemon(true);
        thread.start();
    }

    public void handleBtnStop(ActionEvent e) {
        task.cancel();
    }
}

```

## (5) 작업 상태별 콜백

- 작업이 처리 결과 따라 Task의 다음 세가지 메소드 중 하나 자동 콜백

콜백 메소드	설명
succeeded()	성공적으로 call() 메소드가 리턴되었을 때
cancelled()	cancel() 메소드로 작업이 취소되었을 때
failed()	예외가 발생되었을 때

- Task 클래스를 작성할 때 재정의해서 애플리케이션 로직으로 재구성 가능
- 작업 결과가 있는 Task일 경우(call() 메소드가 리턴값 있을 경우)
  - succeeded() 메소드 재정의해 작업 결과 얻는 것 가능
  - V는 Task의 타입 파라미터에 지정된 타입
- JavaFX Application Thread상에서 호출
  - 안전하게 UI 변경 코드 작성 가능

[RootController.java] 컨트롤러

```

package sec11.exam03_task_callback;

import java.net.URL;
import java.util.ResourceBundle;

import javafx.concurrent.Task;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.ProgressBar;

public class RootController implements Initializable {
    @FXML private ProgressBar progressBar;
    @FXML private Label lblWorkDone;
    @FXML private Label lblResult;
    @FXML private Button btnStart;
    @FXML private Button btnStop;

    private Task<Integer> task;

    @Override
    public void initialize(URL location, ResourceBundle resources) {
        btnStart.setOnAction(event->handleBtnStart(event));
    }
}

```



```

        btnStop.setOnAction(event->handleBtnStop(event));
    }

    public void handleBtnStart(ActionEvent e) {
        task = new Task<Integer>() {
            @Override
            protected Integer call() throws Exception {
                int result = 0;
                for(int i=0; i<=100; i++) {
                    if(isCancelled()) { break; }
                    result += i;
                    updateProgress(i, 100);
                    updateMessage(String.valueOf(i));
                    try {Thread.sleep(100); } catch(InterruptedException e) {
                        if(isCancelled()) { break; }
                    }
                }
                return result;
            }
        };

        @Override
        protected void succeeded() {
            lblResult.setText(String.valueOf(getValue()));
        }

        @Override
        protected void cancelled() {
            lblResult.setText("취소됨");
        }

        @Override
        protected void failed() {
            lblResult.setText("실패");
        }
    };

    progressBar.progressProperty().bind(task.progressProperty());
    lblWorkDone.textProperty().bind(task.messageProperty());
    lblResult.setText("");

    Thread thread = new Thread(task);
    thread.setDaemon(true);
    thread.start();
}

public void handleBtnStop(ActionEvent e) {
    task.cancel();
}
}

```

### 17.11.3. Service 클래스

- 작업 스레드상에서 Task를 간편하게 시작, 취소, 재시작 기능을 제공한다.

#### (1) Service 생성

- Service를 상속받고 createTask() 메소드 재정의

- createTask()는 작업 스레드가 실행할 Task를 생성해서 리턴

```
class CustomService extends Service<Integer> {
    @Override
    protected Task<Integer> createTask() {
        Task<Integer> task = new Task<Integer>() { ... };
        return task;
    }
}
```

## (2) Service 시작, 취소, 재시작

- Service를 시작하려면 start() 메소드를 호출하면 되고, 취소하려면 cancel() 메소드를 호출하면 된다. 그리고 재시작이 필요할 경우 restart()를 호출하면 된다.

```
public class RootController implements Initializable {
    private TimeService timeService;

    @Override
    public void initialize(URL location, ResourceBundle resources) {
        timeService = new TimeService();
        timeService.start();
    }

    public void handleBtnStart(ActionEvent e) {
        timeService.restart();
    }

    public void handleBtnStop(ActionEvent e) {
        timeService.cancel();
    }
}

class TimeService extends Service<Void> {
    @Override
    protected Task<Void> createTask() {
        Task<Void> task = new Task<Void>() {...};
        return task;
    }
}
```

## (3) 작업 상태별 콜백

- 작업이 어떻게 처리됐는지 따라 Service의 다음 세 가지 메소드 중 하나가 자동 콜백

콜백 메소드	설명
succeeded()	성공적으로 call() 메소드가 리턴되었을 때
cancelled()	cancel() 메소드로 작업이 취소되었을 때
failed()	예외가 발생되었을 때

- Service 클래스를 작성할 때 재정의해서 애플리케이션 로직으로 재구성
- 작업 결과가 있는 Task일 경우(call() 메소드가 리턴값이 있을 경우)
  - succeeded() 메소드를 재정의해 작업 결과 얻음

- V는 Task의 타입 파라미터에 지정된 타입

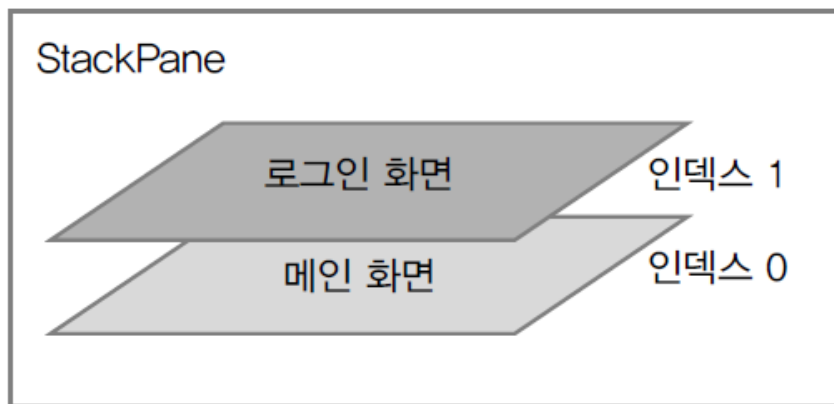
■ JavaFX Application Thread상에서 호출

- 안전하게 UI 변경 코드 작성 가능

## 17.12 화면 이동과 애니메이션

### 17.12.1 화면 이동

- 화면을 이동하는 가장 쉬운 방법은 Stage에 새로운 Scene을 세팅하는 것이다.
- StackPane을 루트 컨테이너로 사용하면 애니메이션도 사용 가능



### 17.12.2 애니메이션

- 컨트롤 또는 컨테이너의 속성(Property) 변화를 주어진 시간 동안 진행함으로써 구현
- 애니메이션과 관련된 클래스

클래스	설명
Timeline	KeyFrame에 설정된 내용대로 애니메이션을 진행시키는 객체
KeyValue	타겟 속성(Property)과 종료값을 설정하는 객체
KeyFrame	애니메이션의 지속 시간과 KeyValue를 설정하는 객체 (지속 시간 동안 타겟 속성의 값을 종료값까지 변화시킴)

[RootController.java] 메인 화면 컨트롤러

```

01 package sec12.exam01_move_animation;
02
03 import java.net.URL;
04 import java.util.ResourceBundle;
05
06 import javafx.animation.KeyFrame;
07 import javafx.animation.KeyValue;
08 import javafx.animation.Timeline;
09 import javafx.event.ActionEvent;
10 import javafx.fxml.FXML;
11 import javafx.fxml.FXMLLoader;

```

```

12 import javafx.fxml.Initializable;
13 import javafx.scene.Parent;
14 import javafx.scene.control.Button;
15 import javafx.scene.layout.StackPane;
16 import javafx.util.Duration;
17
18 public class RootController implements Initializable {
19     @FXML private Button btnLogin;
20
21     @Override
22     public void initialize(URL location, ResourceBundle resources) {
23         btnLogin.setOnAction(e->handleBtnLogin(e));
24     }
25
26     public void handleBtnLogin(ActionEvent event) {
27         try {
28             Parent login= FXMLLoader.load(getClass().getResource("login.fxml"));
29             StackPane root = (StackPane) btnLogin.getScene().getRoot();
30             root.getChildren().add(login);
31
32             login.setTranslateX(350);
33
34             Timeline timeline = new Timeline();
35             KeyValue keyValue = new KeyValue(login.translateXProperty(), 0);
36             KeyFrame keyFrame = new KeyFrame(Duration.millis(100), keyValue);
37             timeline.getKeyFrames().add(keyFrame);
38             timeline.play();
39         } catch(Exception e) {
40             e.printStackTrace();
41         }
42     }
43 }

```

## [과제] 확인문제

1. 학생들의 점수를 보여주는 애플리케이션의 메인 윈도우를 다음과 같이 만들어 보세요.

[정답]

```

[AppMain.java]

package verify.exam01;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

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);

        primaryStage.setTitle("AppMain");
        primaryStage.setScene(scene);
        primaryStage.setResizable(false);
        primaryStage.show();
    }

    public static void main(String[] args) {
        launch(args);
    }
}

```

```
}
}
```

[root.fxml]

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.layout.*?>
<?import javafx.scene.control.*?>

<BorderPane xmlns:fx="http://javafx.com/fxml" fx:controller="verify.exam01.RootController">
    <center>
        <TableView fx:id="tableView" prefHeight="200" prefWidth="420">
            <columns>
                <TableColumn prefWidth="100" resizable="false" text="이름" />
                <TableColumn prefWidth="100" resizable="false" text="국어" />
                <TableColumn prefWidth="100" resizable="false" text="수학" />
                <TableColumn prefWidth="100" resizable="false" text="영어" />
            </columns>
        </TableView>
    </center>
    <bottom>
        <HBox prefHeight="50" alignment="CENTER">
            <children>
                <Button fx:id="btnAdd" text="추가" />
            </children>
        </HBox>
    </bottom>
</BorderPane>
```

[RootController.java]

```
package verify.exam01;

import java.net.URL;
import java.util.ResourceBundle;

import javafx.fxml.Initializable;

public class RootController implements Initializable {
    @Override
    public void initialize(URL location, ResourceBundle resources) {
    }
}
```

2. 1번 예제에 이어서 메인 윈도우에서 [추가] 버튼을 클릭하면 학생 정보 추가 다이얼로그가 다음과 같이 실행되도록 해보세요.

[정답]

[RootController.java]

```
01 package verify.exam02;
02
03 import java.io.IOException;
04 import java.net.URL;
05 import java.util.ResourceBundle;
06
07 import verify.exam03.Student;
08 import javafx.event.ActionEvent;
```

```

09  import javafx.fxml.FXML;
10  import javafx.fxml.FXMLLoader;
11  import javafx.fxml.Initializable;
12  import javafx.scene.Parent;
13  import javafx.scene.Scene;
14  import javafx.scene.control.Button;
15  import javafx.scene.control.Label;
16  import javafx.scene.control.TextField;
17  import javafx.stage.Modality;
18  import javafx.stage.Stage;
19  import javafx.stage.StageStyle;
20
21  public class RootController implements Initializable {
22      @FXML private Button btnAdd;
23
24      @Override
25      public void initialize(URL location, ResourceBundle resources) {
26          btnAdd.setOnAction(event->handleBtnAddAction(event));
27      }
28
29      private void handleBtnAddAction(ActionEvent event) {
30          try {
31              Stage dialog = new Stage(StageStyle.UTILITY);
32              dialog.initModality(Modality.WINDOW_MODAL);
33              dialog.initOwner(btnAdd.getScene().getWindow());
34              dialog.setTitle("추가");
35
36              Parent parent = FXMLLoader.load(getClass().getResource("form.fxml"));
37
38              Button btnFormCancel = (Button) parent.lookup("#btnFormCancel");
39              btnFormCancel.setOnAction(e->dialog.close());
40
41              Scene scene = new Scene(parent);
42              dialog.setScene(scene);
43              dialog.setResizable(false);
44              dialog.show();
45          } catch (IOException e) {}
46      }
47  }

```

3. 2번 예제에 이어서 학생 정보 추가 다이얼로그에서 정보를 입력하고 [저장] 버튼을 클릭하면 메인 윈도우의 TableView에 행이 추가되도록 해보세요.

[정답]

```

[RootController.java]

01  package verify.exam03;
02
03  import java.io.IOException;
04  import java.net.URL;
05  import java.util.ResourceBundle;
06
07  import javafx.collections.FXCollections;
08  import javafx.collections.ObservableList;
09  import javafx.event.ActionEvent;
10  import javafx.fxml.FXML;
11  import javafx.fxml.FXMLLoader;
12  import javafx.fxml.Initializable;
13  import javafx.scene.Parent;
14  import javafx.scene.Scene;
15  import javafx.scene.control.Button;

```

```

16 import javafx.scene.control.TableColumn;
17 import javafx.scene.control.TableView;
18 import javafx.scene.control.TextField;
19 import javafx.scene.control.cell.PropertyValueFactory;
20 import javafx.stage.Modality;
21 import javafx.stage.Stage;
22 import javafx.stage.StageStyle;
23
24 public class RootController implements Initializable {
25     @FXML private TableView<Student> tableView;
26     @FXML private Button btnAdd;
27
28     private ObservableList<Student> list;
29
30     @Override
31     public void initialize(URL location, ResourceBundle resources) {
32         list = FXCollections.observableArrayList();
33
34         TableColumn tc = tableView.getColumns().get(0);
35         tc.setCellValueFactory(new PropertyValueFactory("name"));
36         tc.setStyle("--fx-alignment: CENTER;");
37
38         tc = tableView.getColumns().get(1);
39         tc.setCellValueFactory(new PropertyValueFactory("korean"));
40         tc.setStyle("--fx-alignment: CENTER;");
41
42         tc = tableView.getColumns().get(2);
43         tc.setCellValueFactory(new PropertyValueFactory("math"));
44         tc.setStyle("--fx-alignment: CENTER;");
45
46         tc = tableView.getColumns().get(3);
47         tc.setCellValueFactory(new PropertyValueFactory("english"));
48         tc.setStyle("--fx-alignment: CENTER;");
49
50         tableView.setItems(list);
51
52         btnAdd.setOnAction(event->handleBtnAddAction(event));
53     }
54
55     private void handleBtnAddAction(ActionEvent event) {
56         try {
57             Stage dialog = new Stage(StageStyle.UTILITY);
58             dialog.initModality(Modality.WINDOW_MODAL);
59             dialog.initOwner(btnAdd.getScene().getWindow());
60             dialog.setTitle("추가");
61
62             Parent parent = FXMLLoader.load(getClass().getResource("form.fxml"));
63
64             Button btnFormAdd = (Button) parent.lookup("#btnFormAdd");
65             btnFormAdd.setOnAction(e->{
66                 TextField txtName = (TextField) parent.lookup("#txtName");
67                 TextField txtKorean = (TextField) parent.lookup("#txtKorean");
68                 TextField txtMath = (TextField) parent.lookup("#txtMath");
69                 TextField txtEnglish = (TextField) parent.lookup("#txtEnglish");
70                 list.add(new Student(
71                     txtName.getText(),
72                     Integer.parseInt(txtKorean.getText()),
73                     Integer.parseInt(txtMath.getText()),
74                     Integer.parseInt(txtEnglish.getText())
75                 ));
76                 dialog.close();
77             });
78
79             Button btnFormCancel = (Button) parent.lookup("#btnFormCancel");
80             btnFormCancel.setOnAction(e->dialog.close());

```

```

81
82             Scene scene = new Scene(parent);
83             dialog.setScene(scene);
84             dialog.setResizable(false);
85             dialog.show();
86         } catch (IOException e) {}
87     }
88 }

```

4. 3번 예제에 이어서 메인 윈도우 하단에 [학생별 막대 그래프] 버튼을 추가하고, 이 버튼을 클릭하면 오른쪽 그림과 같이 막대 그래프 다이얼로그가 실행되도록 해보세요.

[정답]

[RootController.java]

```

01 package verify.exam04;
02
03 import java.io.IOException;
04 import java.net.URL;
05 import java.util.ResourceBundle;
06
07 import javafx.collections.FXCollections;
08 import javafx.collections.ObservableList;
09 import javafx.event.ActionEvent;
10 import javafx.fxml.FXML;
11 import javafx.fxml.FXMLLoader;
12 import javafx.fxml.Initializable;
13 import javafx.scene.Node;
14 import javafx.scene.Parent;
15 import javafx.scene.Scene;
16 import javafx.scene.chart.BarChart;
17 import javafx.scene.chart.XYChart;
18 import javafx.scene.control.Button;
19 import javafx.scene.control.TableColumn;
20 import javafx.scene.control.TableView;
21 import javafx.scene.control.TextField;
22 import javafx.scene.control.cell.PropertyValueFactory;
23 import javafx.stage.Modality;
24 import javafx.stage.Stage;
25 import javafx.stage.StageStyle;
26
27 public class RootController implements Initializable {
28     @FXML private TableView<Student> tableView;
29     @FXML private Button btnAdd;
30     @FXML private Button btnBarChart;
31
32     private ObservableList<Student> list;
33
34     @Override
35     public void initialize(URL location, ResourceBundle resources) {
36         //list = FXCollections.observableArrayList();
37         list = FXCollections.observableArrayList(
38             new Student("홍길동A", 40, 60, 80)        ,
39             new Student("홍길동B", 60, 80, 40)        ,
40             new Student("홍길동C", 80, 40, 60)
41         );
42
43         TableColumn tc = tableView.getColumns().get(0);
44         tc.setCellValueFactory(new PropertyValueFactory("name"));
45         tc.setStyle("-fx-alignment: CENTER;");
46

```



```

47         tc = tableView.getColumns().get(1);
48         tc.setCellValueFactory(new PropertyValueFactory("korean"));
49         tc.setStyle("-fx-alignment: CENTER;");
50
51         tc = tableView.getColumns().get(2);
52         tc.setCellValueFactory(new PropertyValueFactory("math"));
53         tc.setStyle("-fx-alignment: CENTER;");
54
55         tc = tableView.getColumns().get(3);
56         tc.setCellValueFactory(new PropertyValueFactory("english"));
57         tc.setStyle("-fx-alignment: CENTER;");
58
59         tableView.setItems(list);
60
61         btnAdd.setOnAction(event->handleBtnAddAction(event));
62         btnBarChart.setOnAction(event->handlebtnBarChartAction(event));
63     }
64
65     private void handleBtnAddAction(ActionEvent event) {
66         try {
67             Stage dialog = new Stage(StageStyle.UTILITY);
68             dialog.initModality(Modality.WINDOW_MODAL);
69             dialog.initOwner(btnAdd.getScene().getWindow());
70             dialog.setTitle("추가");
71
72             Parent parent = FXMLLoader.load(getClass().getResource("form.fxml"));
73
74             Button btnFormAdd = (Button) parent.lookup("#btnFormAdd");
75             btnFormAdd.setOnAction(e->{
76                 TextField txtName = (TextField) parent.lookup("#txtName");
77                 TextField txtKorean = (TextField) parent.lookup("#txtKorean");
78                 TextField txtMath = (TextField) parent.lookup("#txtMath");
79                 TextField txtEnglish = (TextField) parent.lookup("#txtEnglish");
80                 list.add(new Student(
81                     txtName.getText(),
82                     Integer.parseInt(txtKorean.getText()),
83                     Integer.parseInt(txtMath.getText()),
84                     Integer.parseInt(txtEnglish.getText())
85                 ));
86                 dialog.close();
87             });
88
89             Button btnFormCancel = (Button) parent.lookup("#btnFormCancel");
90             btnFormCancel.setOnAction(e->dialog.close());
91
92             Scene scene = new Scene(parent);
93             dialog.setScene(scene);
94             dialog.setResizable(false);
95             dialog.show();
96         } catch (IOException e) {}
97     }
98
99     private void handlebtnBarChartAction(ActionEvent event) {
100         try {
101             Stage dialog = new Stage(StageStyle.UTILITY);
102             dialog.initModality(Modality.WINDOW_MODAL);
103             dialog.initOwner(btnAdd.getScene().getWindow());
104             dialog.setTitle("막대 그래프");
105
106             Parent parent = FXMLLoader.load(getClass().getResource("barchart.fxml"));
107
108             BarChart barChart = (BarChart) parent.lookup("#barChart");
109
110             XYChart.Series seriesKorean = new XYChart.Series();
111             seriesKorean.setName("국어");

```

```

112         ObservableList koreanList = FXCollections.observableArrayList();
113         for(int i=0; i<list.size(); i++) {
114             koreanList.add(new XYChart.Data(list.get(i).getName(),
115 list.get(i).getKorean()));
116         }
117         seriesKorean.setData(koreanList);
118         barChart.getData().add(seriesKorean);
119
120         XYChart.Series seriesMath = new XYChart.Series();
121         seriesMath.setName("수학");
122         ObservableList mathList = FXCollections.observableArrayList();
123         for(int i=0; i<list.size(); i++) {
124             mathList.add(new XYChart.Data(list.get(i).getName(),
125 list.get(i).getMath()));
126         }
127         seriesMath.setData(mathList);
128         barChart.getData().add(seriesMath);
129
130         XYChart.Series seriesEnglish = new XYChart.Series();
131         seriesEnglish.setName("영어");
132         ObservableList englishList = FXCollections.observableArrayList();
133         for(int i=0; i<list.size(); i++) {
134             englishList.add(new XYChart.Data(list.get(i).getName(),
135 list.get(i).getEnglish()));
136         }
137         seriesEnglish.setData(englishList);
138         barChart.getData().add(seriesEnglish);
139
140         Button btnClose = (Button) parent.lookup("#btnClose");
141         btnClose.setOnAction(e->dialog.close());
142
143         Scene scene = new Scene(parent);
144         dialog.setScene(scene);
145         dialog.show();
146     } catch (IOException e) {}
147 }
148 }

```

5. 4번 예제에 이어서 메인 윈도우의 TableView에서 한 행을 더블클릭하면 오른쪽 그림과 같이 해당 학생의 과목 점수 비율을 보여주는 파이 그래프 다이얼로그가 실행되도록 해보세요.

[정답]

```

[RootController.java]
01 package verify.exam05;
02
03 import java.io.IOException;
04 import java.net.URL;
05 import java.util.ResourceBundle;
06
07 import javafx.collections.FXCollections;
08 import javafx.collections.ObservableList;
09 import javafx.event.ActionEvent;
10 import javafx.fxml.FXML;
11 import javafx.fxml.FXMLLoader;
12 import javafx.fxml.Initializable;
13 import javafx.scene.Node;
14 import javafx.scene.Parent;
15 import javafx.scene.Scene;
16 import javafx.scene.chart.BarChart;
17 import javafx.scene.chart.PieChart;

```

```

18 import javafx.scene.chart.XYChart;
19 import javafx.scene.control.Button;
20 import javafx.scene.control.TableColumn;
21 import javafx.scene.control.TableView;
22 import javafx.scene.control.TextField;
23 import javafx.scene.control.cell.PropertyValueFactory;
24 import javafx.scene.input.MouseEvent;
25 import javafx.stage.Modality;
26 import javafx.stage.Stage;
27 import javafx.stage.StageStyle;
28
29 public class RootController implements Initializable {
30     @FXML private TableView<Student> tableView;
31     @FXML private Button btnAdd;
32     @FXML private Button btnBarChart;
33
34     private ObservableList<Student> list;
35
36     @Override
37     public void initialize(URL location, ResourceBundle resources) {
38         //list = FXCollections.observableArrayList();
39         list = FXCollections.observableArrayList(
40             new Student("홍길동A", 40, 60, 80)        ,
41             new Student("홍길동B", 60, 80, 40)        ,
42             new Student("홍길동C", 80, 40, 60)
43         );
44
45         TableColumn tc = tableView.getColumns().get(0);
46         tc.setCellValueFactory(new PropertyValueFactory("name"));
47         tc.setStyle("-fx-alignment: CENTER;");
48
49         tc = tableView.getColumns().get(1);
50         tc.setCellValueFactory(new PropertyValueFactory("korean"));
51         tc.setStyle("-fx-alignment: CENTER;");
52
53         tc = tableView.getColumns().get(2);
54         tc.setCellValueFactory(new PropertyValueFactory("math"));
55         tc.setStyle("-fx-alignment: CENTER;");
56
57         tc = tableView.getColumns().get(3);
58         tc.setCellValueFactory(new PropertyValueFactory("english"));
59         tc.setStyle("-fx-alignment: CENTER;");
60
61         tableView.setItems(list);
62         tableView.setOnMouseClicked(event->handleTableViewMouseClicked(event));
63
64         btnAdd.setOnAction(event->handleBtnAddAction(event));
65         btnBarChart.setOnAction(event->handlebtnBarChartAction(event));
66     }
67
68     private void handleBtnAddAction(ActionEvent event) {
69         try {
70             Stage dialog = new Stage(StageStyle.UTILITY);
71             dialog.initModality(Modality.WINDOW_MODAL);
72             dialog.initOwner(btnAdd.getScene().getWindow());
73             dialog.setTitle("추가");
74
75             Parent parent = FXMLLoader.load(getClass().getResource("form.fxml"));
76
77             Button btnFormAdd = (Button) parent.lookup("#btnFormAdd");
78             btnFormAdd.setOnAction(e->{
79                 TextField txtName = (TextField) parent.lookup("#txtName");
80                 TextField txtKorean = (TextField) parent.lookup("#txtKorean");
81                 TextField txtMath = (TextField) parent.lookup("#txtMath");
82                 TextField txtEnglish = (TextField) parent.lookup("#txtEnglish");

```

```

83         list.add(new Student(
84             txtName.getText(),
85             Integer.parseInt(txtKorean.getText()),
86             Integer.parseInt(txtMath.getText()),
87             Integer.parseInt(txtEnglish.getText())
88         ));
89         dialog.close();
90     });
91
92     Button btnFormCancel = (Button) parent.lookup("#btnFormCancel");
93     btnFormCancel.setOnAction(e->dialog.close());
94
95     Scene scene = new Scene(parent);
96     dialog.setScene(scene);
97     dialog.setResizable(false);
98     dialog.show();
99     } catch (IOException e) {}
100 }
101
102 private void handlebtnBarChartAction(ActionEvent event) {
103     try {
104         Stage dialog = new Stage(StageStyle.UTILITY);
105         dialog.initModality(Modality.WINDOW_MODAL);
106         dialog.initOwner(btnAdd.getScene().getWindow());
107         dialog.setTitle("막대 그래프");
108
109         Parent parent = FXMLLoader.load(getClass().getResource("barchart.fxml"));
110
111         BarChart barChart = (BarChart) parent.lookup("#barChart");
112
113         XYChart.Series seriesKorean = new XYChart.Series();
114         seriesKorean.setName("국어");
115         ObservableList koreanList = FXCollections.observableArrayList();
116         for(int i=0; i<list.size(); i++) {
117             koreanList.add(new XYChart.Data(list.get(i).getName(),
118 list.get(i).getKorean()));
119         }
120         seriesKorean.setData(koreanList);
121         barChart.getData().add(seriesKorean);
122
123         XYChart.Series seriesMath = new XYChart.Series();
124         seriesMath.setName("수학");
125         ObservableList mathList = FXCollections.observableArrayList();
126         for(int i=0; i<list.size(); i++) {
127             mathList.add(new XYChart.Data(list.get(i).getName(),
128 list.get(i).getMath()));
129         }
130         seriesMath.setData(mathList);
131         barChart.getData().add(seriesMath);
132
133         XYChart.Series seriesEnglish = new XYChart.Series();
134         seriesEnglish.setName("영어");
135         ObservableList englishList = FXCollections.observableArrayList();
136         for(int i=0; i<list.size(); i++) {
137             englishList.add(new XYChart.Data(list.get(i).getName(),
138 list.get(i).getEnglish()));
139         }
140         seriesEnglish.setData(englishList);
141         barChart.getData().add(seriesEnglish);
142
143         Button btnClose = (Button) parent.lookup("#btnClose");
144         btnClose.setOnAction(e->dialog.close());
145
146         Scene scene = new Scene(parent);
147         dialog.setScene(scene);

```

```

148         dialog.show();
149     } catch (IOException e) {}
150 }
151
152 private void handleTableViewMouseClicked(MouseEvent event) {
153     if (event.getClickCount() != 2) return;
154
155     try {
156         Stage dialog = new Stage(StageStyle.UTILITY);
157         dialog.initModality(Modality.WINDOW_MODAL);
158         dialog.initOwner(btnAdd.getScene().getWindow());
159         dialog.setTitle("파이 그래프");
160
161         Parent parent = FXMLLoader.load(getClass().getResource("piechart.fxml"));
162
163         PieChart pieChart = (PieChart) parent.lookup("#pieChart");
164         Student student = tableView.getSelectionModel().getSelectedItem();
165         pieChart.setData(FXCollections.observableArrayList(
166             new PieChart.Data("국어", student.getKorean()),
167             new PieChart.Data("수학", student.getMath()),
168             new PieChart.Data("영어", student.getEnglish())
169         ));
170
171         Button btnClose = (Button) parent.lookup("#btnClose");
172         btnClose.setOnAction(e->dialog.close());
173
174         Scene scene = new Scene(parent);
175         dialog.setScene(scene);
176         dialog.show();
177     } catch (IOException e) {}
178 }
179 }

```