



Java FX



- ❖ Java 기반의 데스크톱, 모바일 및 임베디드 시스템을 위한 오픈 소스 차세대 UI 클라이언트 애플리케이션 플랫폼이다.
- ❖ 자바 코드와 분리해서 스타일 시트(CSS)
  - 테마를 쉽게 변경할 수 있고
  - 안드로이드처럼 화면 레이아웃과 비즈니스 로직을 분리할 수 있음
  - Swing보다는 편리하게 디자인을 할 수 있다



## 개발 환경

- ❖ JavaFX 설치
  - [https ://gluonhq.com/products/javafx](https://gluonhq.com/products/javafx)
- ❖ Scene Builder 설치
  - [https ://gluonhq.com/products/scene-builder](https://gluonhq.com/products/scene-builder)
- ❖ e(fx)clipse 설치

## 프로젝트 생성

### ❖ module-info.java

>>> 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  }
```

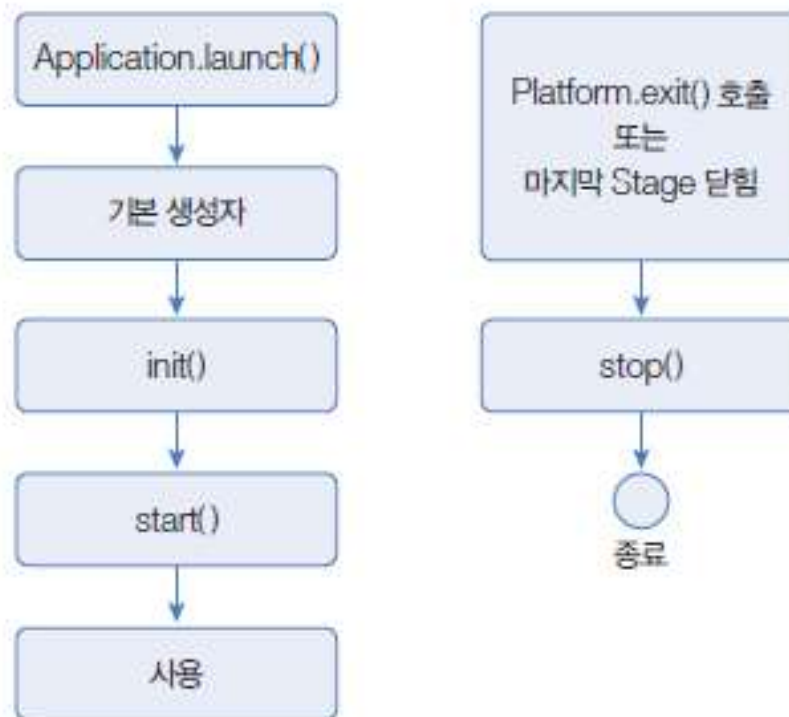
>>> AppMain.java

```
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 }
```

# JavaFX 라이프사이클

❖ JavaFX 애플리케이션이 종료되는 경우는 다음과 같이 세 가지가 있다.

- ⊖ 마우스로 마지막 윈도우(Stage)의 우측 상단 닫기 버튼을 클릭했을 때
- ⊖ 자바 코드로 마지막 윈도우(Stage)의 `close()` 메소드를 호출했을 때
- ⊗ 자바 코드로 `Platform.exit()` 또는 `System.exit(0)`을 호출했을 때



>>> 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
```

# 무대와 장면

## ❖ 무대와 장면

- JavaFX는 윈도우를 무대 Stage로 표현한다
- Stage는 한 번에 하나의 장면(Scene)을 가질 수 있음,
- 메인 윈도우는 start ( ) 메소드의 primaryStage 매개값으로 전달되지만 장면Scene은 직접 생성해야 한다.
- Scene을 생성하려면 UI의 루트 컨테이너인 javafx.scene.Parent가 필요하다.
- Scene scene = new Scene(Parent root);
- primaryStage.setScene(scene);





## ❖ 예제

- Parent의 하위 클래스인 VBox 컨테이너를 이용해서 Scene을 생성
- 메인 윈도우 (primaryStage)의 장면으로 설정했다.
- VBox에는 Label과 Button 컨트롤을 배치했다.

```

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
8 import javafx.scene.control.Button;
9 import javafx.scene.control.Label;
10 import javafx.scene.layout.VBox;
11 import javafx.scene.text.Font;
12 import javafx.stage.Stage;

```



```

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 }

```

## 03 JavaFX 레이아웃

### ❖ 프로그램적 레이아웃

```
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 }
```

## 03 JavaFX 레이아웃

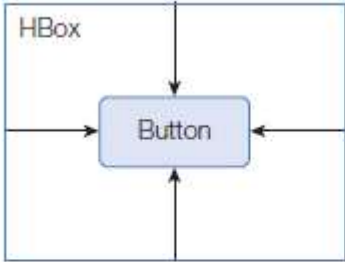
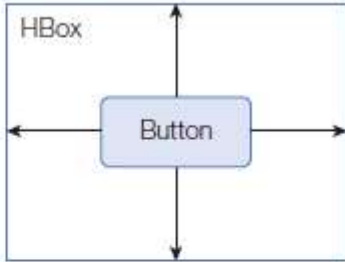
### ❖ FXML 레이아웃

```
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>
```



## 03 JavaFX 레이아웃-레이아웃 여백

### ❖ 패딩과 마진

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>&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>



//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)`

## 03 JavaFX 레이아웃

```
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 }
```

## 03 JavaFX 레이아웃- FXML 태그와 자바 코드 매핑

- ❖ FXML로 선언된 태그는 자바 코드로 변환되어 실행  
→ 자바 코드와 매핑 관계가 존재한다

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>&lt;HBox xmlns:fx="http://javafx.com/fxml"&gt;   &lt;padding&gt;     &lt;Insets top="10" right="10"             bottom="10" left="10"/&gt;   &lt;/padding&gt;   &lt;spacing&gt;10&lt;/spacing&gt; &lt;/HBox&gt;</pre>
<pre>TextField textField = new TextField(); textField.setPrefWidth(200);</pre>	<pre>&lt;TextField&gt;   &lt;prefWidth&gt;200&lt;/prefWidth&gt; &lt;/TextField&gt;</pre>
<pre>Button button = new Button(); button.setText("확인");</pre>	<pre>&lt;Button &gt;   &lt;text&gt;확인&lt;/text&gt; &lt;/Button&gt;</pre>
<pre>ObservableList list = hbox.getChildren(); list.add(textField); list.add(button);</pre>	<pre>&lt;children&gt;   &lt;TextField&gt;...&lt;/TextField&gt;   &lt;Button &gt;...&lt;/Button&gt; &lt;/children&gt;</pre>

## 03 JavaFX 레이아웃

### ❖ 패키지 선언

자바 코드	FXML 태그
<code>import javafx.scene.layout.HBox;</code>	<code>&lt;?import javafx.scene.layout.HBox?&gt;</code>
<code>import javafx.scene.control.*;</code>	<code>&lt;?import javafx.scene.control.*?&gt;</code>

### ❖ <?import?> 태그

- 위치는 XML 선언 태그인 `<?xml version="1.0" encoding="UTF-8"?>`과 루트 컨테이너 태그 사이

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

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

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



## 03 JavaFX 레이아웃- FXML 태그와 자바 코드 매핑

### ❖ 태그 선언

- FXML 태그는 < 와 > 사이에 태그 이름을 작성
- 시작 태그가 있으면 끝 태그도 있어야 한다. 그렇지 않으면 `javax.xml.stream.XMLStreamException` 예외가 발생한다.
- <태그이름> ... </태그이름>
- <태그이름/>

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

## 03 JavaFX 레이아웃

### ❖ 속성 선언

- 속성값은 큰따옴표(") 또는 작은따옴표(')로 반드시 감싸야 한다  
→ `javax.xml.stream.XMLStreamException` 예외
- `<태그이름 속성명="값" 속성명= <값>> ... </태그이름>`
- 속성명은 Setter 메소드명, 기본 타입(`boolean`, `byte`, `short`, `char`, `int`, `long`, `float`, `double`)의 값을 세팅

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

## 03 JavaFX 레이아웃 - 객체 선언

### 1) <클래스 속성="값"/>

- Setter 메소드가 기본 타입과 String 타입이 아닌 다른 타입의 객체를 매개값으로 갖는다면 속성으로 작성할 수 없음  
→ 태그로 작성

```
<클래스 속성="값">
```

```
<클래스>  
  <매개변수>값</매개변수>  
</클래스>
```

- Insets 객체를 FXML로 선언하면

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>

## 03 JavaFX 레이아웃-객체 선언

### 2) <클래스 fx:value="값"/>

- 클래스가 `valueOf(String)` 메소드를 제공해서 객체를 생성하는 경우가
- `String`, `Integer`, `Double`, `Boolean` 객체를 FXML로 선언하면

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>

### 3) <클래스 fx:constant="상수"/>

- `Double.MAX_VALUE` 상수값을 `Button` 컨트롤의 `maxWidth` 속성값으로 설정할 경우

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>

## 03 JavaFX 레이아웃 - 객체 선언

### 4) <클래스 fx:factory="정적메소드">

- 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

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>

## 03 JavaFX 레이아웃- Scene Builder 사용



## 04 JavaFX 컨테이너( p 276)

- ❖ 레이아웃을 작성할 때 컨트롤들을 쉽게 배치할 수 있도록 도와주는 클래스
- ❖ javafx.scene.layout 패키지
- ❖ 접미사가 Pane으로 끝나는 클래스는 모두 컨테이너이고, 그 이외에 Hbox, VBox가 있다

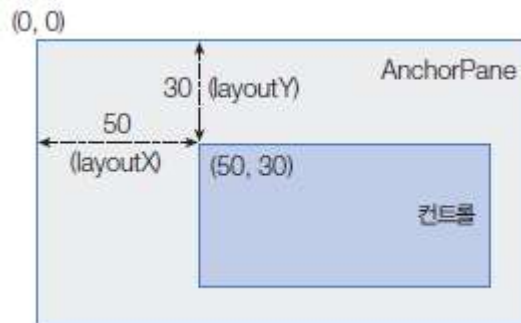
컨테이너	설명
AnchorPane	컨트롤을 좌표를 이용해서 배치하는 레이아웃
BorderPane	위, 아래, 오른쪽, 왼쪽, 중앙에 컨트롤을 배치하는 레이아웃
FlowPane	행으로 배치하되 공간이 부족하면 새로운 행에 배치하는 레이아웃
GridPane	그리드로 배치하되 셀의 크기가 고정적이지 않은 레이아웃
StackPane	컨트롤을 겹쳐 배치하는 레이아웃
TilePane	그리드로 배치하되 고정된 셀의 크기를 갖는 레이아웃
HBox	수평으로 배치하는 레이아웃
VBox	수직으로 배치하는 레이아웃



## 4 JavaFX 컨테이너

### ❖ AnchorPane 컨테이너

- 좌표를 이용하여 AnchorPane의 좌상단(0, 0)을 기준으로 컨트롤을 배치한다



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



## 4 JavaFX 컨테이너

### ❖ HBox와 VBox 컨테이너

- 수평과 수직으로 컨트롤을 배치하는 컨테이너이다

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

## 4 JavaFX 컨테이너

### ❖ BorderPane 컨테이너

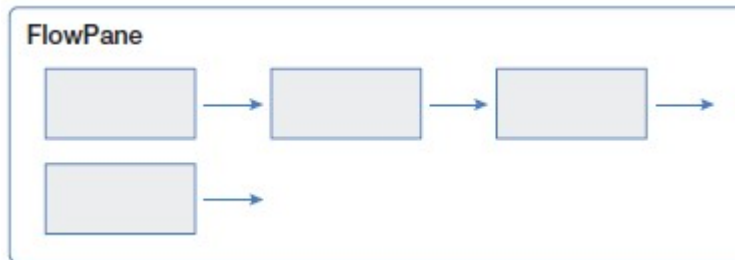
- top, bottom, left, right, center 셀에 컨트롤을 배치하는 컨테이너이다



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

## 4 JavaFX 컨테이너

### ❖ FlowPane 컨테이너



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

## 4 JavaFX 컨테이너

### ❖ TilePane 컨테이너

- 그리드로 컨트롤을 배치하되 고정된 셀(타일) 크기를 갖는 컨테이너이다

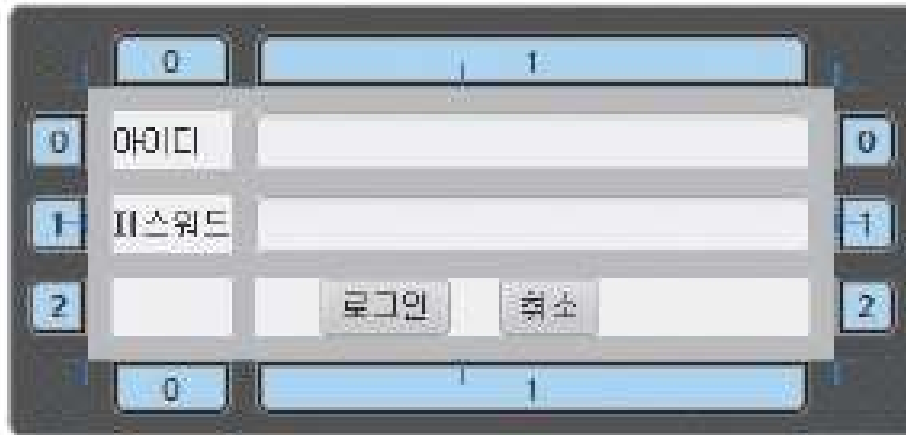


태그 및 속성	설명	적용
prefWidth	폭을 설정	TilePane
prefHeight	높이를 설정	TilePane
prefTileWidth	타일의 폭을 설정	TilePane
prefTileHeight	타일의 높이를 설정	TilePane
<children>	컨트롤을 포함	TilePane

## 4 JavaFX 컨테이너

### ❖ GridPane 컨테이너

- 그리드로 컨트롤을 배치하되 셀의 크기가 고정적이지 않고 유동적인 컨테이너이다
- 셀

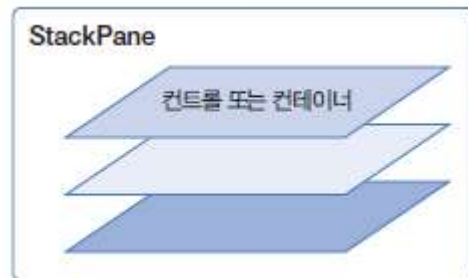


vgap	수직 컨트롤 간격을 설정	GridPane
<children>	컨트롤을 포함	GridPane
GridPane.rowIndex	컨트롤이 위치하는 행 인덱스를 설정	컨트롤
GridPane.columnIndex	컨트롤이 위치하는 컬럼 인덱스를 설정	컨트롤
GridPane.rowSpan	행 병합 수를 설정	컨트롤
GridPane.columnSpan	컬럼 병합 수를 설정	컨트롤
GridPane.hgrow	수평 빈 공간 채우기를 설정	컨트롤
GridPane.vgrow	수직 빈 공간 채우기를 설정	컨트롤
GridPane.halignment	컨트롤의 수평 정렬을 설정	컨트롤
GridPane.valignment	컨트롤의 수직 정렬을 설정	컨트롤

## 4 JavaFX 컨테이너

### ❖ StackPane 컨테이너

- 컨트롤을 겹쳐 배치하는 컨테이너이다, 카드 레이아웃



## 5. JavaFX 이벤트 처리

### ❖ 이벤트 핸들러

- 이벤트 발생 컨트롤과 이벤트 처리를 분리하기 위해 위임형(Delegation) 방식을 사용한다
- 위임형(Delegation) 방식
  - 컨트롤에서 이벤트가 발생하면 컨트롤이 직접 처리하지 않고 이벤트 핸들러에게 이벤트 처리를 위임



## 5. JavaFX 이벤트 처리

### ❖ FXML 컨트롤러

- 프로그램적 레이아웃
  - 레이아웃 코드와 이벤트 처리 코드를 모두 자바 코드로 작성
  - 코드가 복잡해지고 유지보수도 힘들어지며
  - 디자이너와 협력해서 개발하는 것도 쉽지 않다.
- FXML 파일당 별도의 컨트롤러를 지정해서 이벤트를 처리할 수 있음
- FXML 레이아웃과 이벤트 처리 코드를 완전히 분리함

- 1) fx:controller 속성과 컨트롤러 클래스
- 2) fx:id 속성과 @FXML 컨트롤 주입
- 3) EventHandler 등록
- 4) 이벤트 처리 메소드 매핑



## 06. JavaFX 속성 감시와 바인딩

### ❖ 속성 감시

- 컨트롤의 속성(property)을 감시하는 리스너를 설정할 수 있다
- XXXProperty 객체

```
//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; }
```

```
StringProperty stringProperty = textField.textProperty();
stringProperty.addListener( new ChangeListener<String>() {
    @Override
    public void changed(ObservableValue<? extends String> observable,
        String oldValue, String newValue) { ... }
} );
```

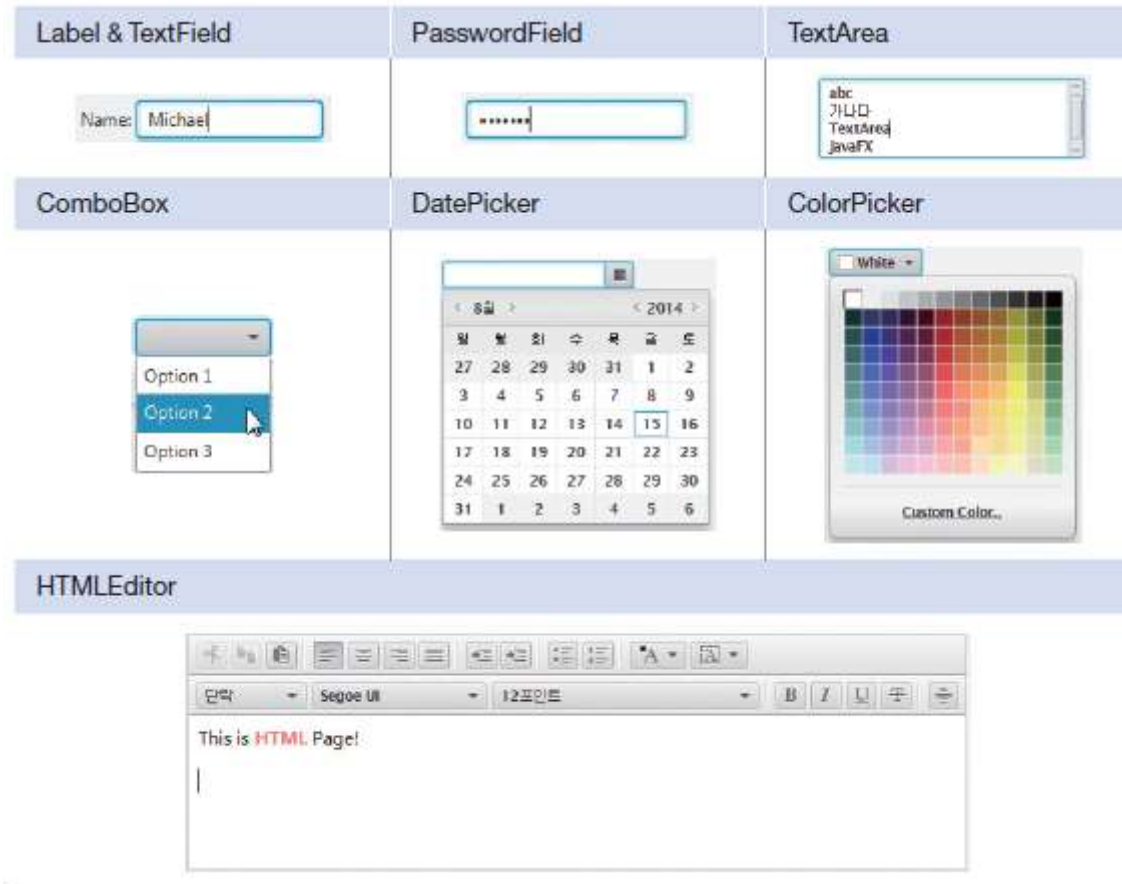
- p. 303 예제

## 06. JavaFX 속성 감시와 바인딩

- ❖ 속성 바인딩
- ❖ Bindings 클래스

## 7 JavaFX 컨트롤

### ❖ 입력 컨트롤



## 7 JavaFX 컨트롤

### ❖ 뷰 컨트롤



## 7 JavaFX 컨트롤

### ❖ 미디어 컨트롤

- 1) MediaPlayer와 MediaView 컨트롤
- 2) Slider 컨트롤
- 3) ProgressBar와 ProgressIndicator 컨트롤

## 7 JavaFX 컨트롤

### ❖ 차트 컨트롤



## 8 JavaFX 메뉴바와 툴바

### ❖ MenuBar 컨트롤



### ❖ Toolbar 컨트롤





## 9 JavaFX 다이얼로그

### ❖ FileChooser, DirectoryChooser





## 9 JavaFX 다이얼로그

❖ Popup

❖ 커스텀 다이얼로그

## 9 JavaFX 다이얼로그

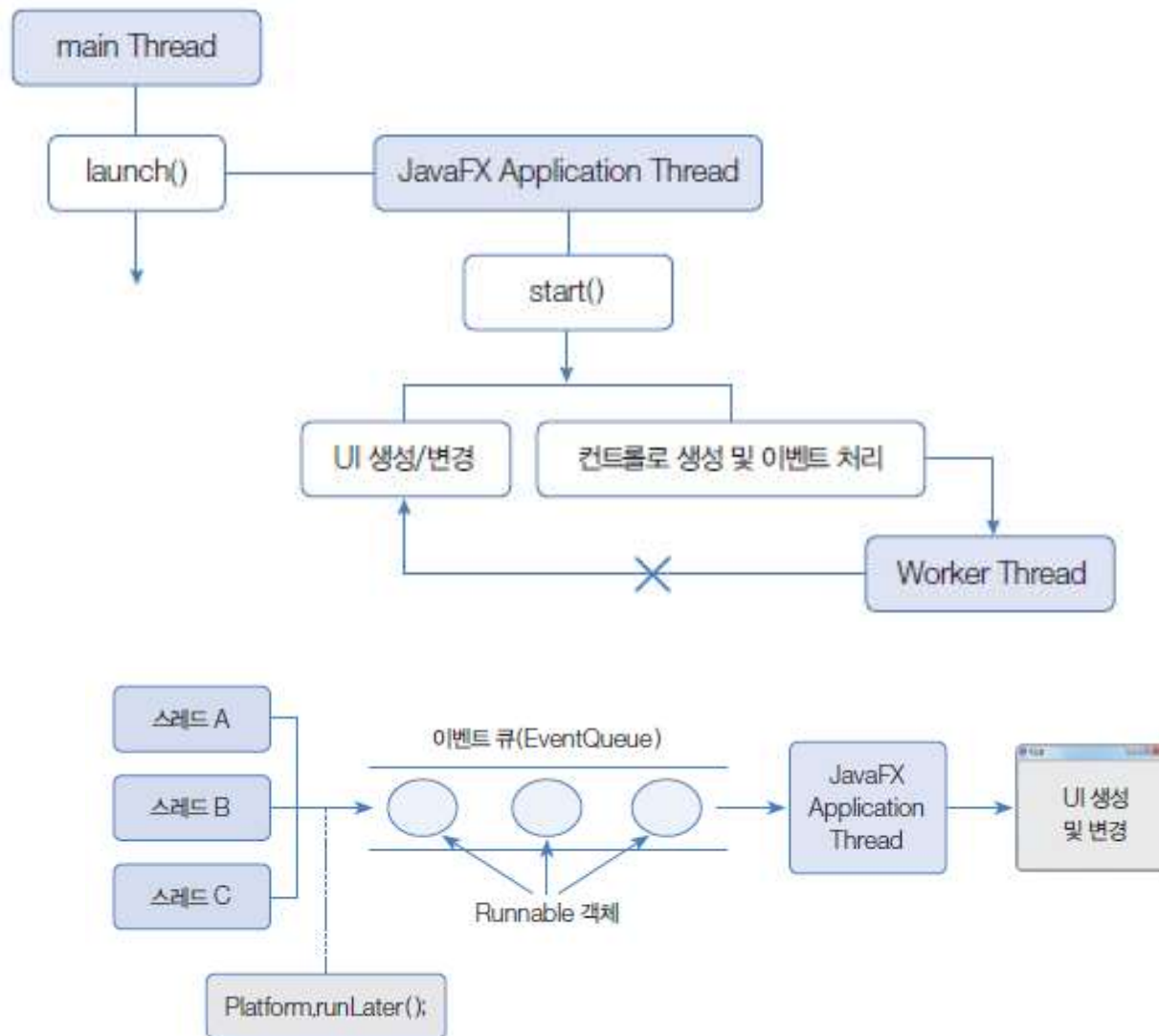
### ❖ primaryStage 참조 얻기

- 컨트롤러에서 다이얼로그를 실행할 때는 소유자 윈도우가 될 primaryStage가 필요하다.
- 컨트롤러에서 primaryStage를 얻는 방법은 두 가지가 있다
  - 1) 메인 클래스에서 전달하는 방법
  - 2) 컨테이너 또는 컨트롤로부터 얻는 방법

## 10 JavaFX CSS 스타일(376)

- ❖ W3C CSS 버전 2.1 스펙(<http://www.w3.org/TR/CSS21>)
- ❖ 인라인 스타일
- ❖ 외부 CSS 파일

## 11 JavaFX 스레드 UI 변경



## 12 장면 이동과 애니메이션

### ❖ 화면 이동

## 13 JavaFX 과제

❖ 학생들의 점수를 보여주는 애플리케이션