



JAVA PROGRAMMING

제 17장 JAVAFX 프로그래밍

장안대학교
인터넷정보통신과

Java GUI 프로그래밍 개요 -850p

■ AWT (Abstract Windows Toolkit)

- 초기 자바 GUI 툴킷 – OS마다 다른 윈도우 컴퍼넌트에 대응시키는 방법
- 실행되는 시스템에 따라 다른 모양을 나타낼수 있다.
- 중량(heavyweight) 컴포넌트
- 테이블, 트리, 진행 바 같은 컴포넌트가 지원되지 않는다.

■ Swing

- OS 컴퍼넌트를 사용않고 java 코드로만 구현된 gui 컴퍼넌트 제공
- 시스템에 상관없이 일관된 모양을 나타낸다. -Look & feel
- 경량(lightweight) 컴포넌트

■ SWT(Standard Widget Toolkit)

- AWT와 Swing 장점만을 가진 gui 시스템을 구현
- 가능한 native 위젯은 사용하고 나머지만 자바코드 위젯 사용 - gui 개선
- IBM의 공개 프로젝트인 Eclipse에서 제공하는 라이브러리

■ JavaFX

- RIA(Rich Internet Application) – 디자인,만들고,테스트,디버그
- CSS(Cascading Style Sheet)를 이용한 디자인과 코드 분리
- FXML - 위젯의 배치를 직관적으로 표현해주기 위한 XML문서
- 씬 빌더(Scene Builder)를 이용한 개발

JavaFX 개요 - 850p

- RIA(Rich Internet Application) 전쟁
 - 웹 애플리케이션의 장점은 유지하면서 기존 웹 브라우저의 단점인 낮은 응답 속도, 데스크톱 애플리케이션에 비해 떨어지는 조작성 등을 개선하기 위한 기술 - x 인터넷
 - Adobe(플래쉬), Microsoft(Silverlight), Sun
- JavaFX는 RIA(Rich Internet Application)를 디자인하고 만들고 테스트하고 디버그하고 배포할 수 있는 일련의 그래픽과 미디어 패키지
- JavaFX는 Java SE의 표준 GUI 라이브러리였던 Swing을 대체하기 위해 Sun에서도 2008년에 JavaFX 1.0을 발표
- 2011년 JavaFX 2.0, JDK 8에 포함 → JavaFX 8
- JavaFX는 더 가벼워지고 더 강력한 기능을 가지며 데스크탑 뿐 아니라 웹 브라우저에서도 실행이 가능
- HTML5가 대세
- JavaFX는 Swing GUI를 대체할 새로운 엔진 → HTML5와 관계없이 지속적으로 개발.



SceneBuilder-869p

- JavaFx를 이용한 제작에 있어, 보다 생산성을 높일 수 있도록 오라클에서 별도로 제공하는 TOOLS
- 드래그 앤 드롭으로 손쉽게 JavaFx Ui를 제작 가능 .
- 오라클에서는 NetBeam에서 Scene Builder를 사용하도록 권장
- Eclipse에서 javaFx UI제작을 하기 위해선 이클립스와 Scene Builder를 연동해야 함
- <http://www.oracle.com/technetwork/java/javafxscenebuilder-1x-archive-2199384.html>
- Gluon 버전 : <http://gluonhq.com/products/scene-builder/>
- Download → 실행
- C:\Program Files (x86)\Oracle
폴더에 설치됨

JavaFX Scene Builder 2.0 Related Downloads

You must accept the [Oracle BSD](#) to download this software.

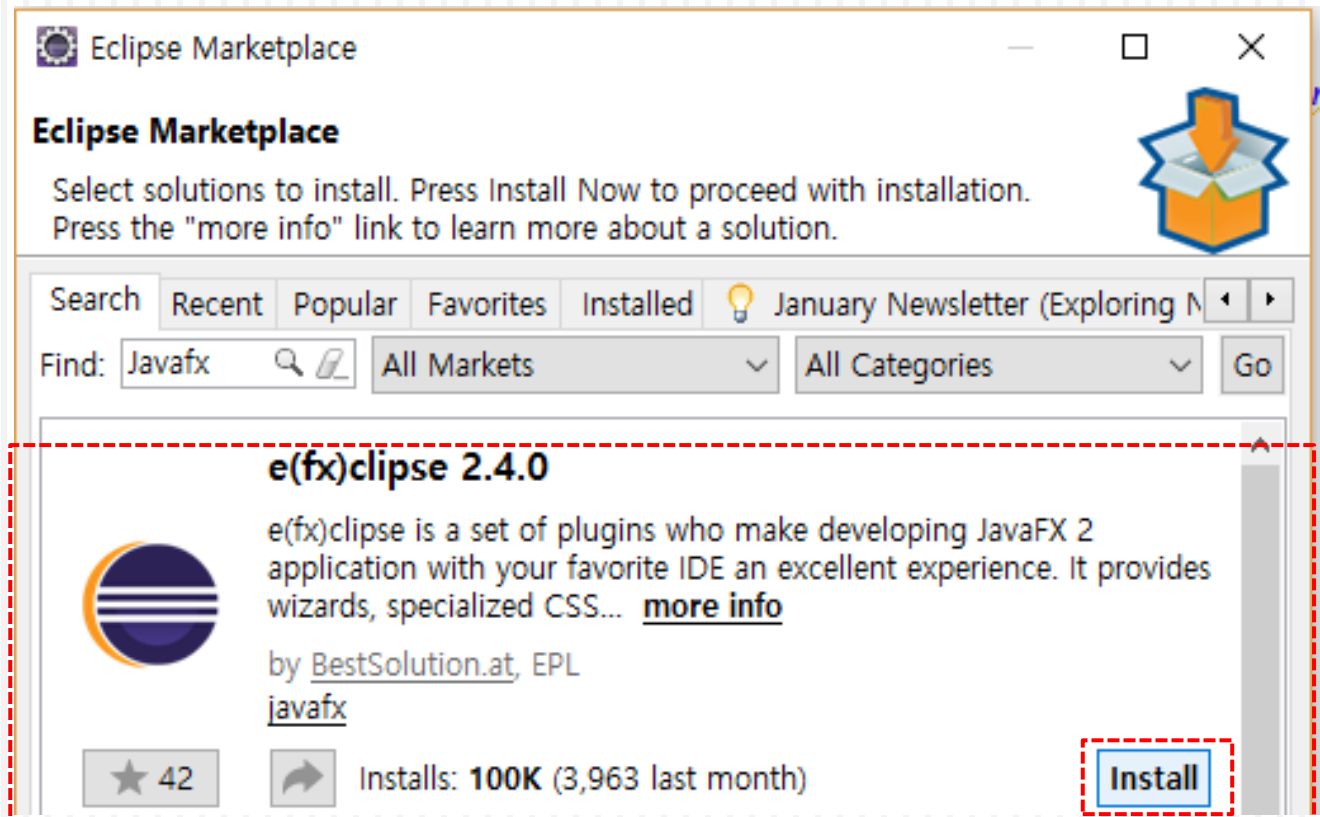
☐ Accept License Agreement ☒ Decline License Agreement

Product / File Description	File Size	Download
Windows 32/64 bit (msi)	56.1 MB	javafx_scenebuilder-2_0-windows.msi
Mac OS X (dmg)	68.6 MB	javafx_scenebuilder-2_0-macosx-universal.dmg
Linux 32-bit (deb)	61.5 MB	javafx_scenebuilder-2_0-linux-i586.deb
Linux 32-bit (tar.gz)	61.8 MB	javafx_scenebuilder-2_0-linux-i586.tar.gz
Linux 64-bit (tar.gz)	60.7 MB	javafx_scenebuilder-2_0-linux-x64.tar.gz
Linux 64-bit (deb)	60.5 MB	javafx_scenebuilder-2_0-linux-x64.deb
JavaFX Scene Builder 2.0 Samples	0.3 MB	javafx_scenebuilder_samples-2_0.zip
JavaFX Scene Builder 2.0 Kit API Documentation	1.2 MB	javafx_scenebuilder_kit_javadoc-2_0.zip
JavaFX Scene Builder 2.0 Kit Samples	68 KB	javafx_scenebuilder_kit_samples-2_0.zip

[Back to top](#)

JavaFX plug-in 설치 -870p

- JavaFX는 JDK 8에 포함되어 있으나 Eclipse에서 사용하기 위해서는 e(fx)clipse 플러그인 필요

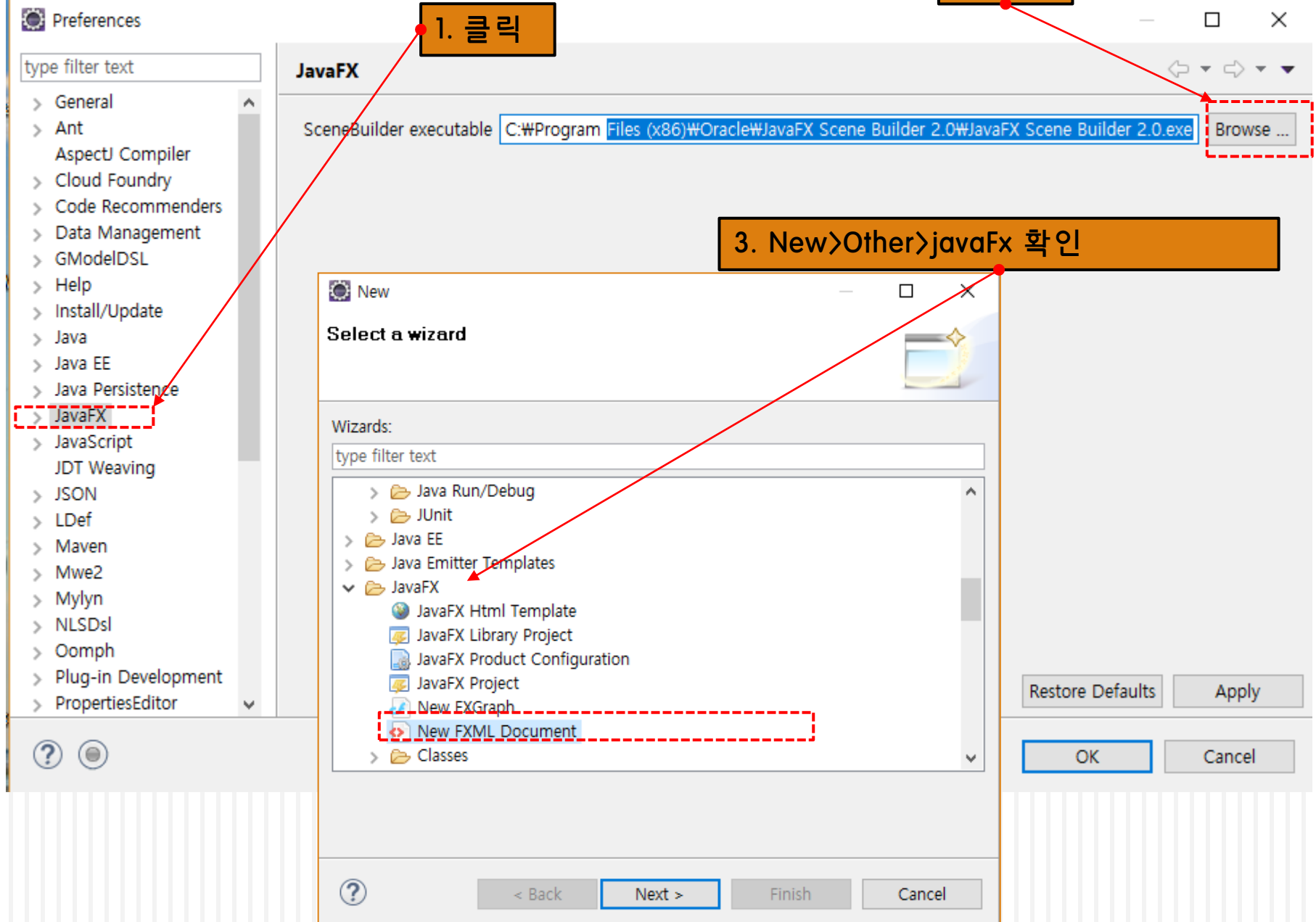


SceneBuilder 와 eclipse 연동

2. 검색

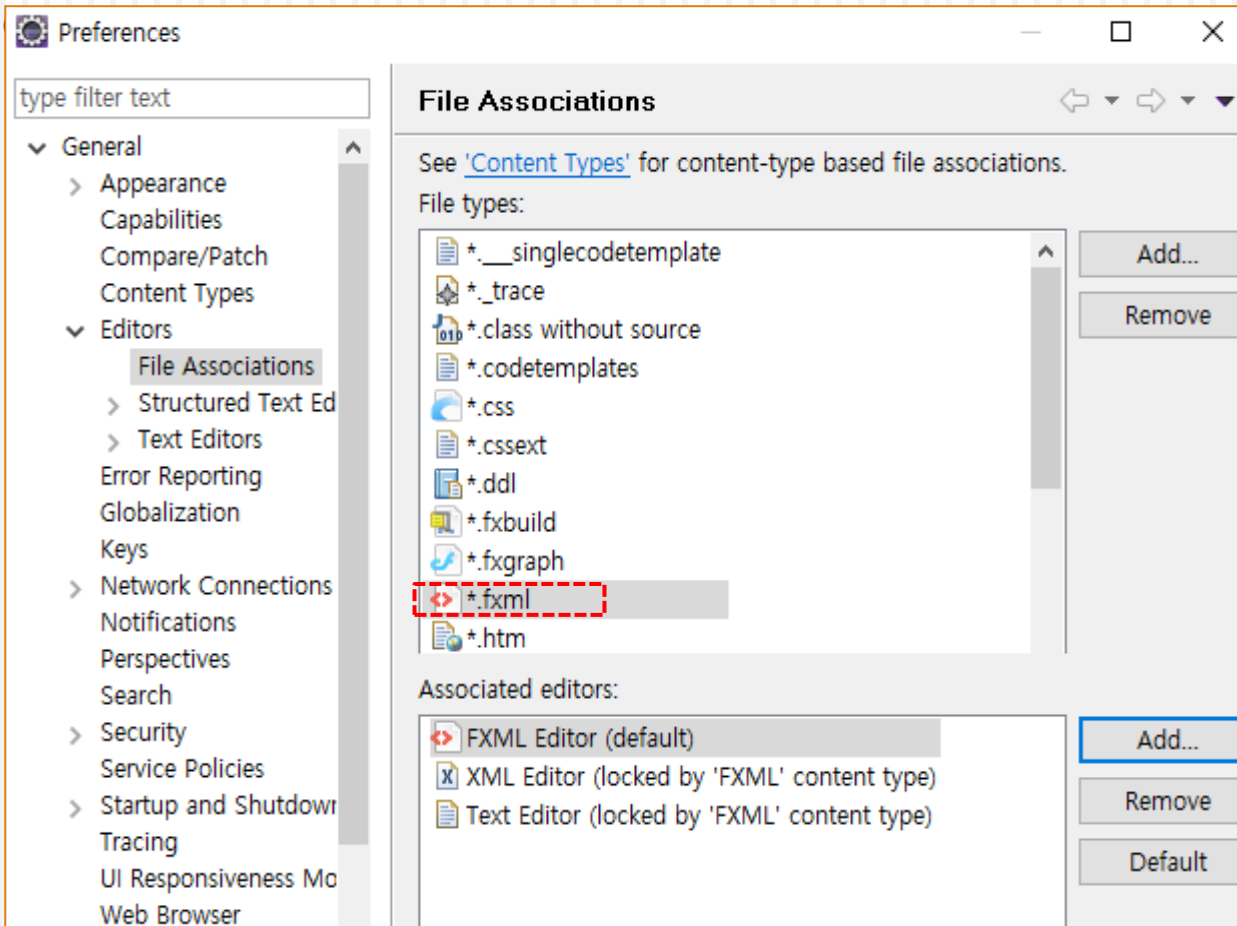
1. 클릭

3. New>Other>javaFx 확인



e(fx)clipse과 Scene Builder 연동 확인

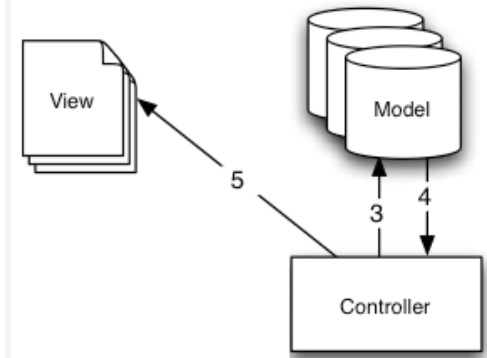
- Window > Preferences
- General > Editors > File associations
- *.fxml을 선택 ➔ Associated editors 필드에 editor 확인
- FXML이 없는 경우 설치한 Scene Builder의 실행파일 (JavaFX Scene Builder 2.0.exe)를 찾아 선택



MVC 모델 -851p

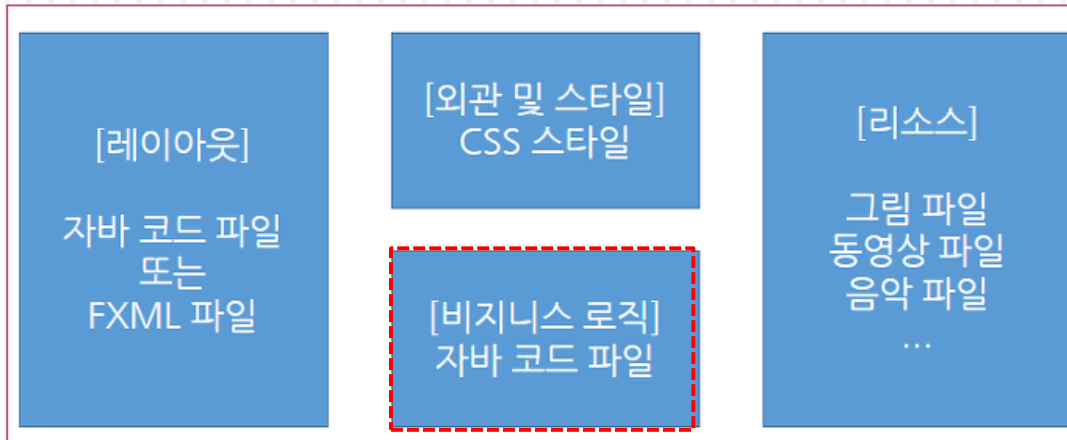
- 자바 프로그램에서 gui를 구축하는 기본 구조
- GUI를 만들 때 화면설계, 데이터저장, 이벤트처리 등을 한곳에 모아 두는 것 보다는 분리하는 것이 개발이나 유지보수에 효과적

- View – 화면표시, 데이터를 시각적으로 표현, 유저 인터페이스
- Controller – 사용자 입력을 처리, 로직구현, 이벤트처리 기능
- Model – 데이터의 저장과 접근기능



- 이벤트처리기에서 처리한 결과는 대부분 이벤트소스(화면)에 표현되나 이벤트처리기에서 이벤트소스(화면) 자원에 접근이 어렵다.
- 완전한 MVC 구조는 구현하기 어렵기 때문에 View가 하는 역할과 Controller의 역할을 합쳐서 수행하는 Delegate(View + Controller) 구조를 구현한다. - 복잡성감소, 생산성 증가, 컴퍼넌트 설계 단순화 – 프로그램적 레이아웃
- FXML 레이아웃에서는 완전한 MVC 모델 구현 가능

JavaFX 애플리케이션 구성요소



프로그래밍적 레이아웃 - 857p

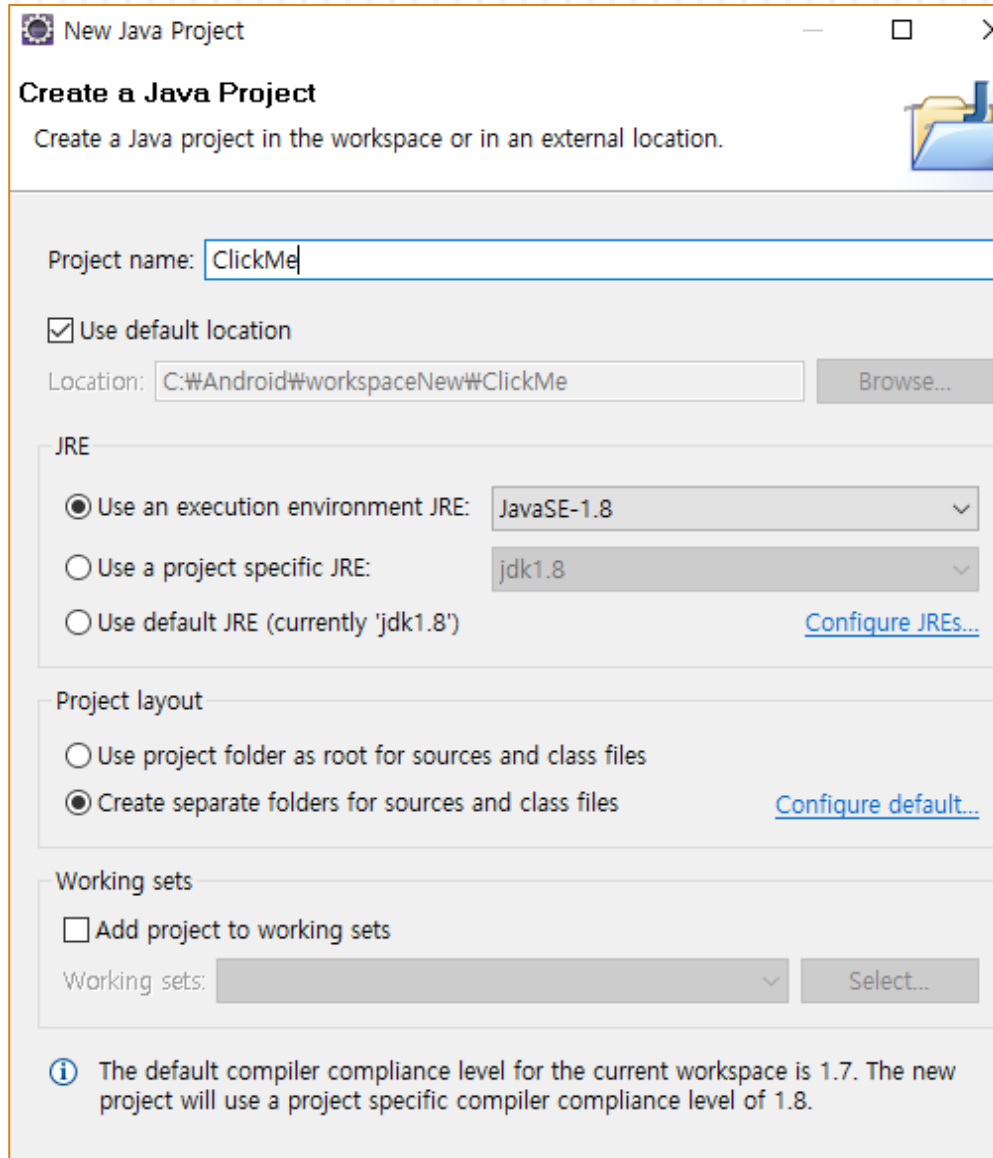
- 자바코드로 UI 컨트롤을 배치하는 것
- 컨트롤 배치, 스타일 지정, 이벤트 처리 등을 모두 자바 코드로 작성

FXML 레이아웃 - 859p

- FXML은 XML 기반의 마크업 언어
- JavaFX App의 UI 레이아웃을 자바 코드에서 분리해서 XML 태그로 선언
- SceneBuilder를 이용하여 컨트롤의 드래그 앤 드롭으로 화면 설계 가능

JavaFX 애플리케이션 시작 - 851p

■ File>New >other >JavaFX>JavaFX Project



New Java Project

Create a Java Project
Create a Java project in the workspace or in an external location.

Project name:

☒ Use default location
Location: [Browse...](#)

JRE

☒ Use an execution environment JRE:

☐ Use a project specific JRE:

☐ Use default JRE (currently 'jdk1.8') [Configure JREs...](#)

Project layout

☐ Use project folder as root for sources and class files

☒ Create separate folders for sources and class files [Configure default...](#)

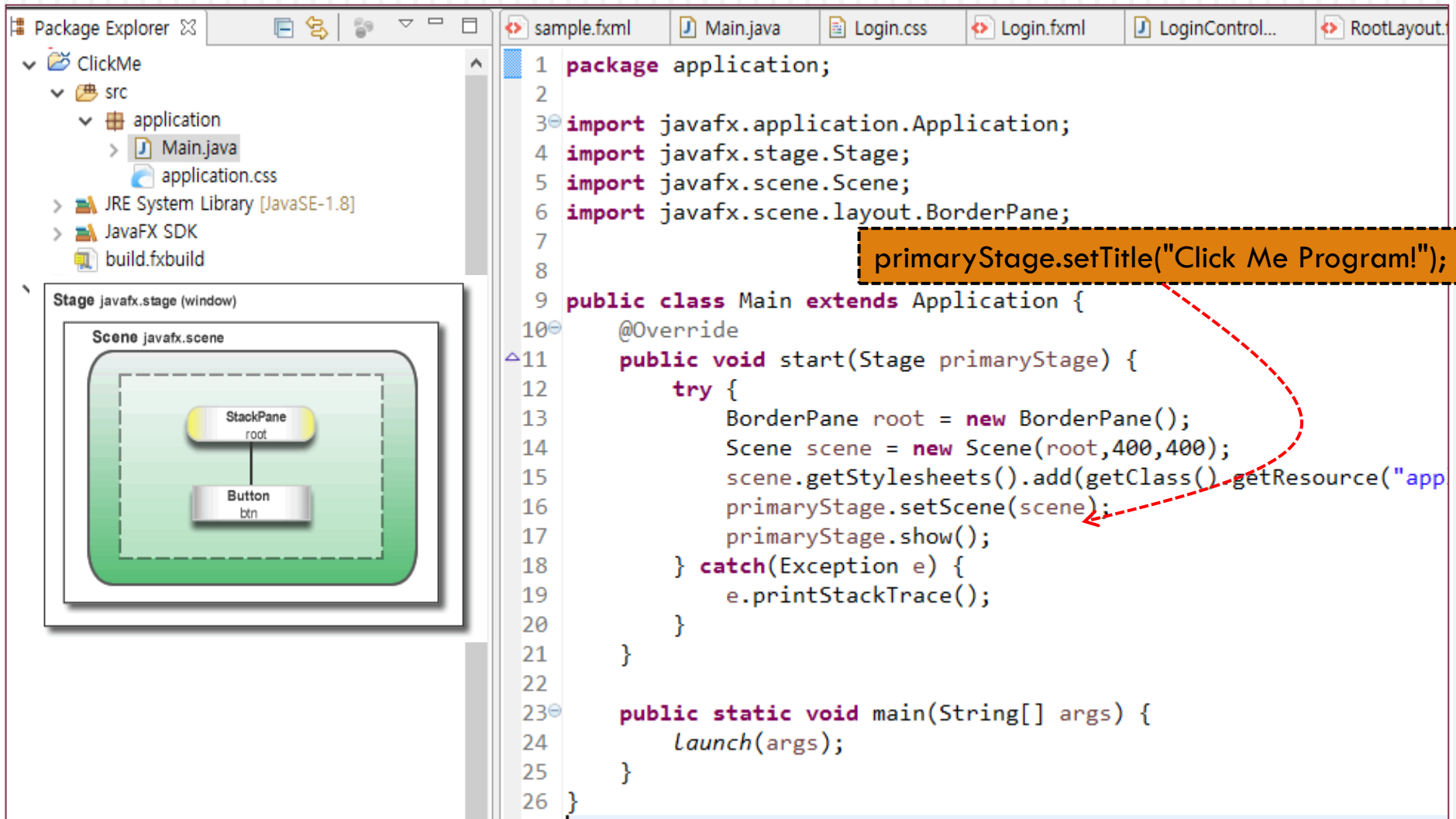
Working sets

☐ Add project to working sets

Working sets: [Select...](#)

i The default compiler compliance level for the current workspace is 1.7. The new project will use a project specific compiler compliance level of 1.8.

JavaFX 애플리케이션 시작 프로그램 - 852p



The screenshot displays an IDE with a Package Explorer on the left and a code editor on the right. The Package Explorer shows a project named 'ClickMe' with a source folder 'src' containing 'application' and 'Main.java'. The code editor shows the 'Main.java' file with the following code:

```
1 package application;
2
3 import javafx.application.Application;
4 import javafx.stage.Stage;
5 import javafx.scene.Scene;
6 import javafx.scene.layout.BorderPane;
7
8
9 public class Main extends Application {
10     @Override
11     public void start(Stage primaryStage) {
12         try {
13             BorderPane root = new BorderPane();
14             Scene scene = new Scene(root, 400, 400);
15             scene.getStylesheets().add(getClass().getResource("app
16             primaryStage.setScene(scene);
17             primaryStage.show();
18         } catch (Exception e) {
19             e.printStackTrace();
20         }
21     }
22
23     public static void main(String[] args) {
24         launch(args);
25     }
26 }
```

A red dashed arrow points from the `primaryStage.setScene(scene);` line in the code to a visual diagram of the scene. The diagram shows a 'Stage' containing a 'Scene', which contains a 'StackPane' (labeled 'root') and a 'Button' (labeled 'btn').

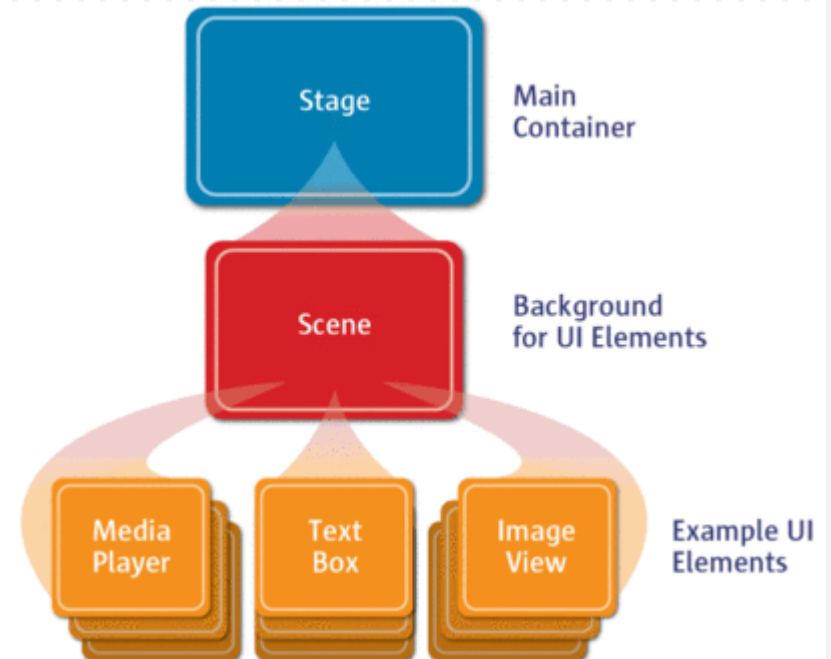
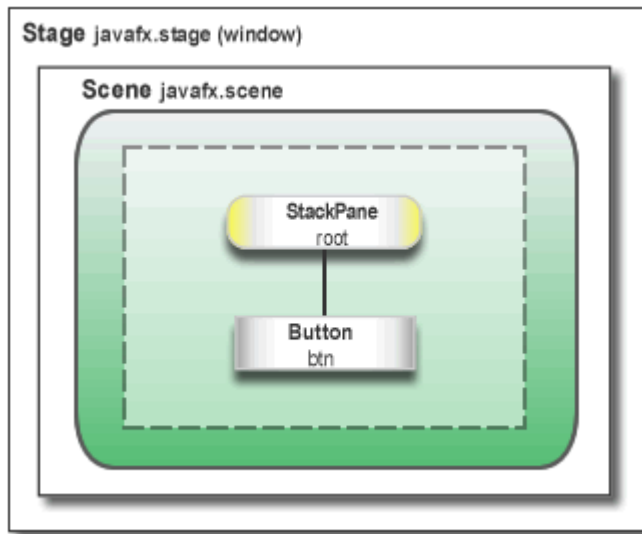
primaryStage.setTitle("Click Me Program!");

@Override : 오버라이드 (재정의)

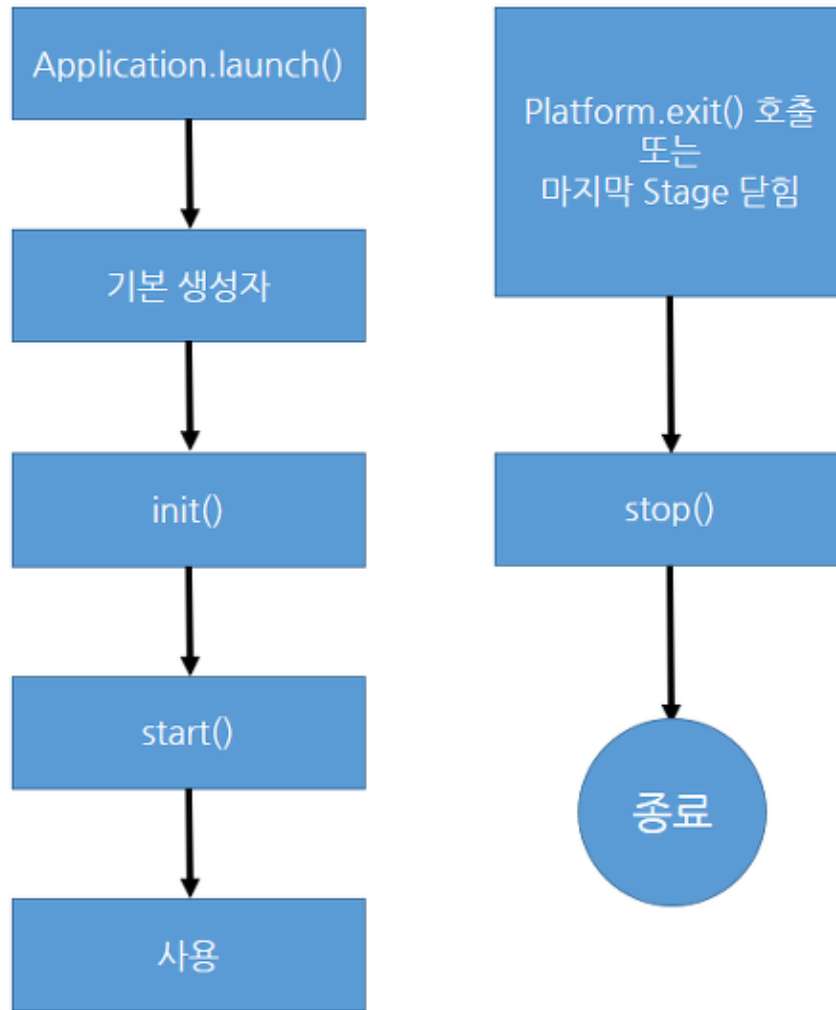
추상메서드 (body가 없는 메서드)를 상속받은 클래스에서 body를 재정의 하는 메서드라는 표기

어플리케이션 구성요소

- **JavaFX 어플리케이션을 시작시키는 메인(Main) 클래스**
 - 추상 클래스인 `javafx.application.Application`을 상속
 - `launch()`, `init()`, `start()`, `stop()`,
 - `start()` 메소드를 재정의(Overriding)
- **main() 메소드에서 Application의 launch() 메소드를 호출**
 - `launch()` 메소드는 메인 클래스의 객체를 생성하고, 메인 윈도우를 생성한 다음 `start()` 메소드를 호출하는 역할



Life-cycle - 853p



Life-cycle 확인

Console

=> main () 시작-----
: 생성자 Main() 호출
: init() 호출
: start() 호출
: start() 끝...
: stop() 호출
=> main () 끝-----

```
1 package application;
2
3 import java.util.List;
4
5
6
7
8
9
10
11
12 public class Main extends Application {
13
14     public Main() {
15         System.out.println(": 생성자 Main() 호출");
16     }
17
18     @Override
19     public void init() throws Exception {
20         System.out.println(" : init() 호출");
21     }
22
23     @Override
24     public void start(Stage primaryStage) {
25         System.out.println(" : start() 호출");
26         try {
27             BorderPane root = new BorderPane();
28             Scene scene = new Scene(root,400,400);
29             scene.getStylesheets().add(getClass().getResource("applica
30             primaryStage.setScene(scene);
31             primaryStage.show();
32         } catch (Exception e) {
33             e.printStackTrace();
34         }
35         System.out.println(" : start() 끝...");
36     }
37     @Override
38     public void stop() throws Exception {
39         System.out.println(": stop() 호출");
40     }
41
42     public static void main(String[] args) {
43         System.out.println(" : main () 시작-----");
44         Launch(args);
45         System.out.println(" : main () 끝----- ");
46     }
47 }
```

레이아웃 (Layout) - 857p

■ 프로그램적 레이아웃

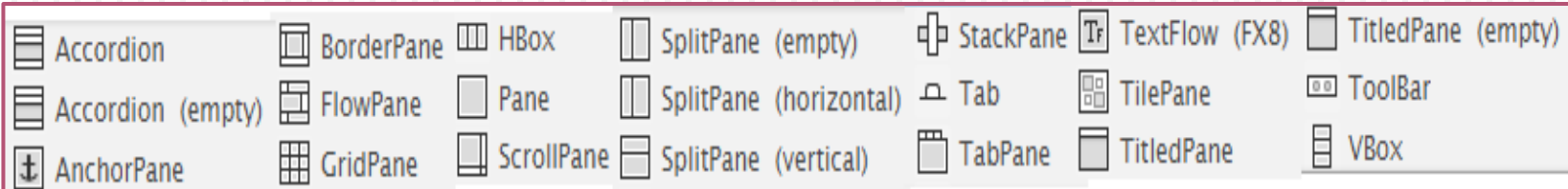
- 자바코드로 UI 컨트롤을 배치하는 것
- 컨트롤 배치, 스타일 지정, 이벤트 처리 등을 모두 자바 코드로 작성.

■ FXML 레이아웃

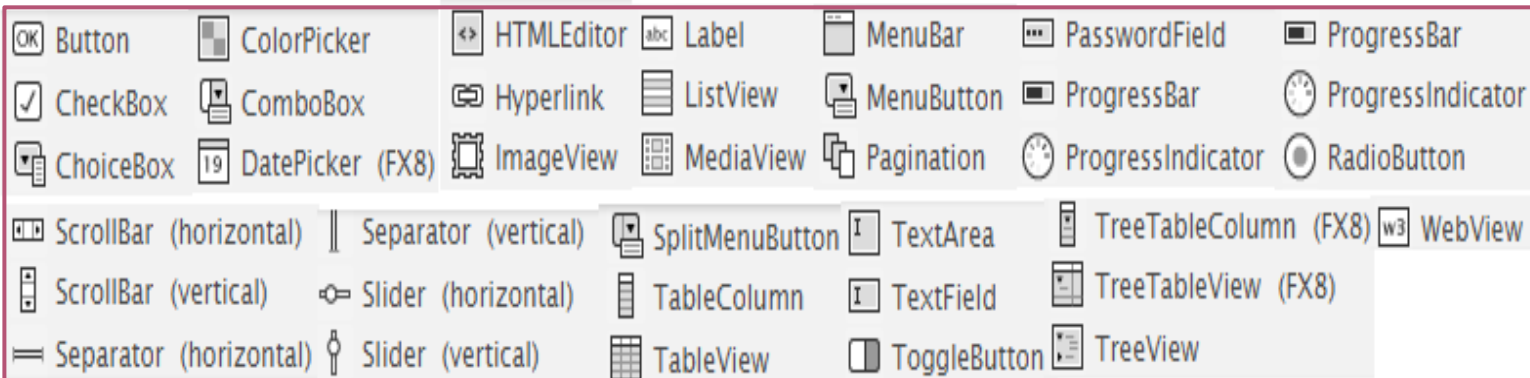
- FXML은 XML로 레이아웃을 작성하고, 자바로 이벤트 처리 및 어플리케이션 로직 작성 - XML 기반의 마크업 언어
- JavaFX App의 UI 레이아웃을 자바 코드에서 분리해서 태그로 선언
- 드래그 앤 드롭으로 손쉽게 JavaFx Ui를 제작할 수 있는 JavaFX Scene Builder 제공
- **JavaFX Scene Builder**
 - 드래그 앤 드롭으로 화면을 디자인하면 FXML을 자동으로 생성
 - JavaFx를 이용한 제작에 있어, 보다 생산성을 높일 수 있도록 오라클에서 별도로 제공하는 툴

레이아웃 (Layout)

Containers



Controls

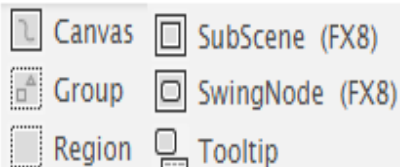


Menu



레이아웃 (Layout)

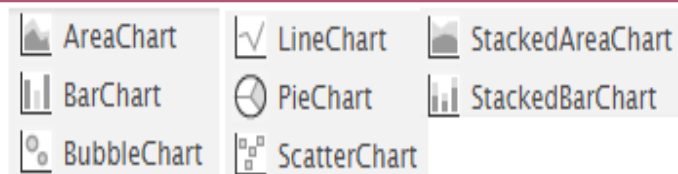
Miscellaneous



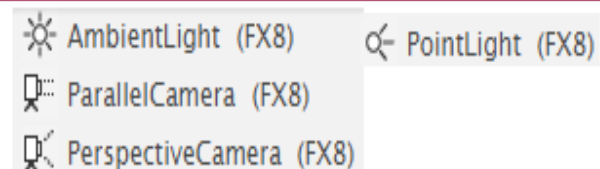
Shape



Charts



3D























Controls



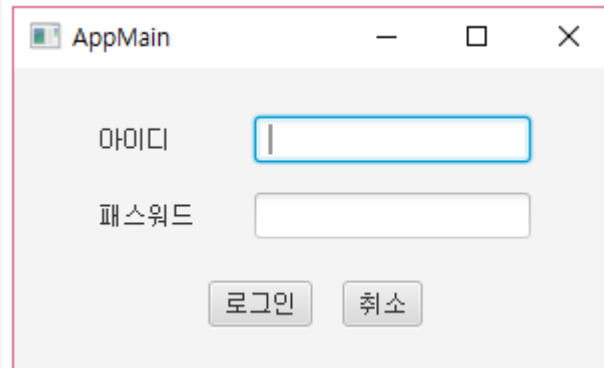
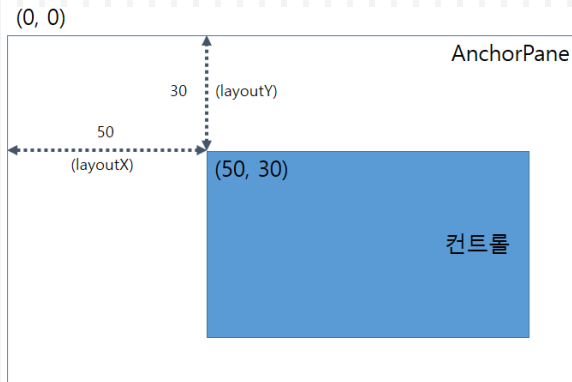
Container

Container : 다른 컴포넌트를 포함하는 컴포넌트

 Accordion	 BorderPane	 HBox	 SplitPane (empty)	 StackPane	 TextFlow (FX8)	 TitledPane (empty)
 Accordion (empty)	 FlowPane	 Pane	 SplitPane (horizontal)	 Tab	 TilePane	 ToolBar
 AnchorPane	 GridPane	 ScrollPane	 SplitPane (vertical)	 TabPane	 TitledPane	 VBox

컨테이너	설명
<u>AnchorPane</u>	컨트롤을 좌표로 배치하는 레이아웃
<u>BorderPane</u>	위, 아래, 오른쪽, 왼쪽, 중앙에 컨트롤을 배치하는 레이아웃
FlowPane	행으로 배치하되 공간이 부족하면 새로운 행에 배치하는 레이아웃
<u>GridPane</u>	그리드로 배치하되 셀의 크기가 고정적이지 않은 레이아웃
StackPane	컨트롤을 겹쳐서 배치하는 레이아웃
TilePane	그리드로 배치하되 고정된 셀의 크기를 갖는 레이아웃
<u>Hbox</u>	수평으로 배치하는 레이아웃
<u>Vbox</u>	수직으로 배치하는 레이아웃

AnchorPane



```
<AnchorPane xmlns:fx="http://javafx.com/fxml/1" prefHeight="150.0" prefWidth="
  <children>
    <Label layoutX="42.0" layoutY="28.0" text="아이디" />
    <Label layoutX="42.0" layoutY="66.0" text="패스워드" />
    <TextField layoutX="120.0" layoutY="24.0" />
    <PasswordField layoutX="120.0" layoutY="62.0" />
    <Button layoutX="97.0" layoutY="106.0" text="로그인" />
    <Button layoutX="164.0" layoutY="106.0" text="취소" />
  </children>
</AnchorPane>
```

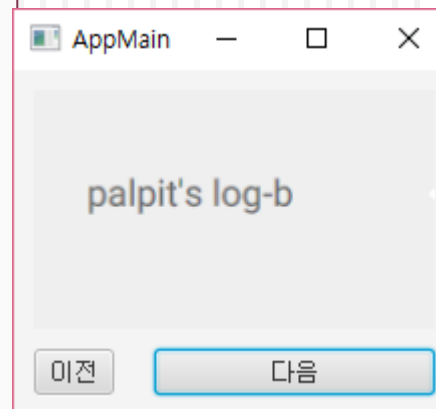
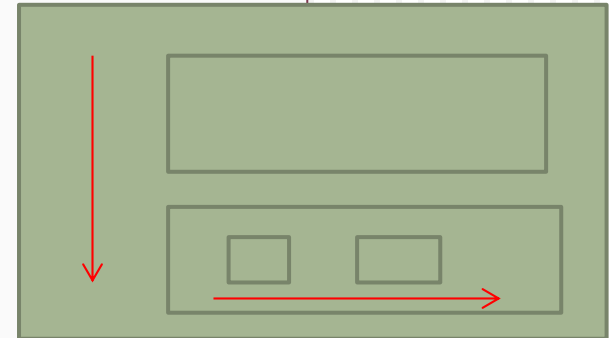
HBox와 VBox 컨테이너

- HBox와 VBox는 수평과 수직으로 컨트롤을 배치하는 컨테이너

```
<VBox xmlns:fx="http://javafx.com/fxml/1">
  <padding>
    <Insets bottom="10.0" left="10.0" right="10.0" top="10.0" />
  </padding>

  <children>
    <ImageView fitWidth="200.0" preserveRatio="true">
      <image>
        <Image url="@images/pal.png" />
      </image>
    </ImageView>

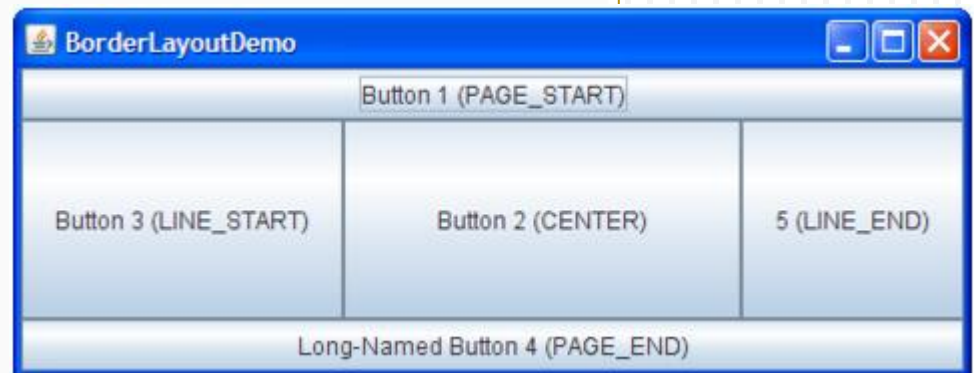
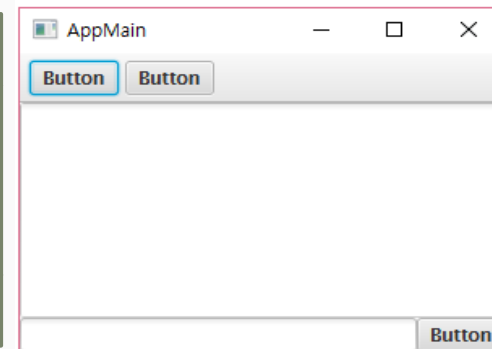
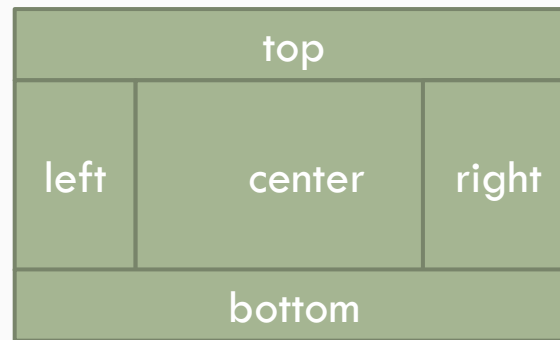
    <HBox alignment="CENTER" spacing="20.0">
      <children>
        <Button text="이전" />
        <Button text="다음">
          <HBox.hgrow><Priority fx:constant="ALWAYS"/></HBox.hgrow>
          <maxWidth><Double fx:constant="MAX_VALUE"/></maxWidth>
        </Button>
      </children>
      <VBox.margin>
        <Insets top="10.0" />
      </VBox.margin>
    </HBox>
  </children>
</VBox>
```



BorderPane 컨테이너

- top, bottom, left, right, center 셀에 컨트롤을 배치하는 컨테이너.
- 컨트롤만 배치하는 것이 아니라 다른 컨테이너도 배치할 수 있음
- 각 셀에는 하나의 컨트롤 또는 컨테이너만 배치 가능

```
<BorderPane xmlns:fx="http://javafx.com/fxml/1" prefHeight="200" prefWidth="300"
  <top>
    <ToolBar>
      <items>
        <Button text="Button" />
        <Button text="Button" />
      </items>
    </ToolBar>
  </top>
  <center>
    <TextArea />
  </center>
  <bottom>
    <BorderPane>
      <center>
        <TextField />
      </center>
      <right>
        <Button text="Button" />
      </right>
    </BorderPane>
  </bottom>
</BorderPane>
```

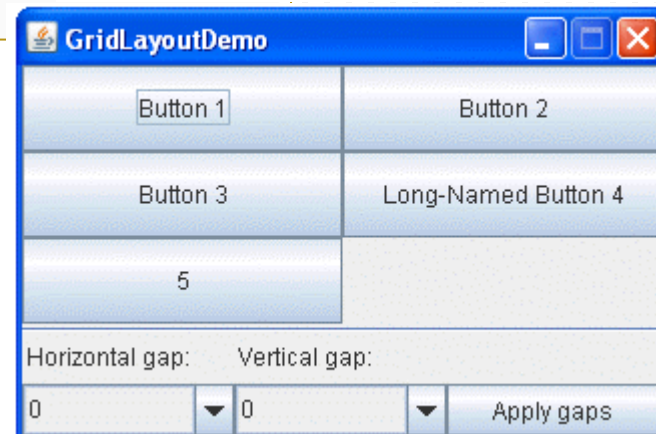
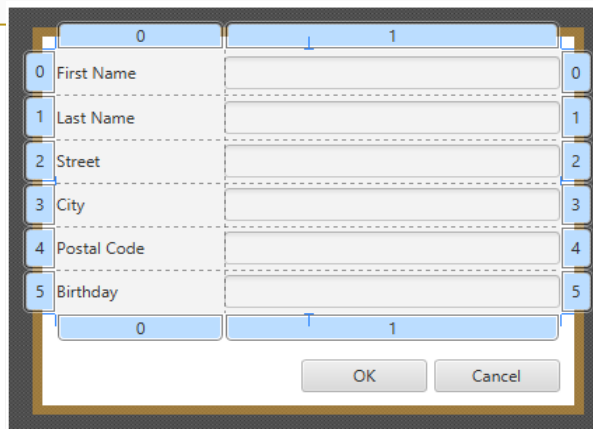
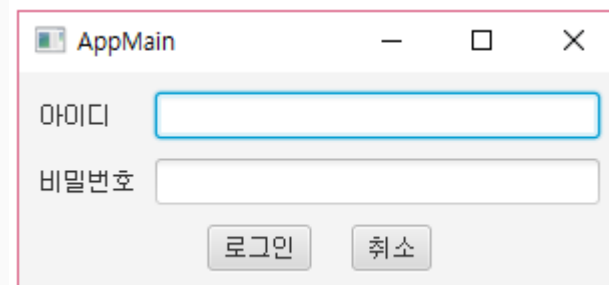


GridPane 컨테이너

```
<GridPane xmlns:fx="http://javafx.com/fxml/1" prefWidth="300" hgap="10" vgap="10">
  <padding>
    <Insets topRightBottomLeft="10"/>
  </padding>
  <children>
    <Label text="아이디" GridPane.rowIndex="0" GridPane.columnIndex="0" />
    <TextField GridPane.rowIndex="0" GridPane.columnIndex="1" GridPane.hgrow="ALWAYS" />

    <Label text="비밀번호" GridPane.rowIndex="1" GridPane.columnIndex="0" />
    <TextField GridPane.rowIndex="1" GridPane.columnIndex="1" GridPane.hgrow="ALWAYS" />

    <HBox GridPane.rowIndex="2" GridPane.columnIndex="0"
      GridPane.columnSpan="2" GridPane.hgrow="ALWAYS"
      alignment="CENTER" spacing="20">
      <children>
        <Button text="로그인" />
        <Button text="취소" />
      </children>
    </HBox>
  </children>
</GridPane>
```



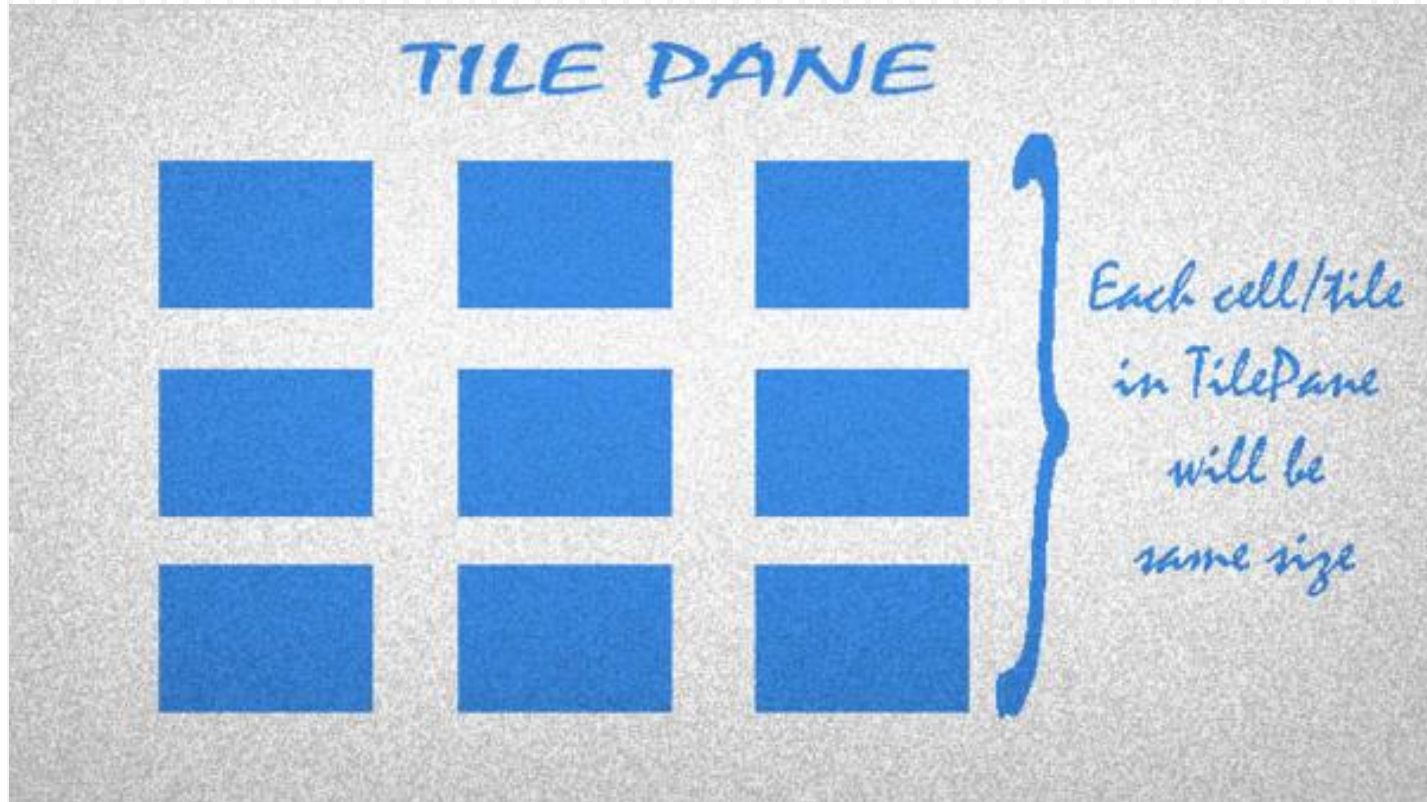
FlowPane

회과 종으로 배치하되 부족하면 새로운 회과 종으로 배치

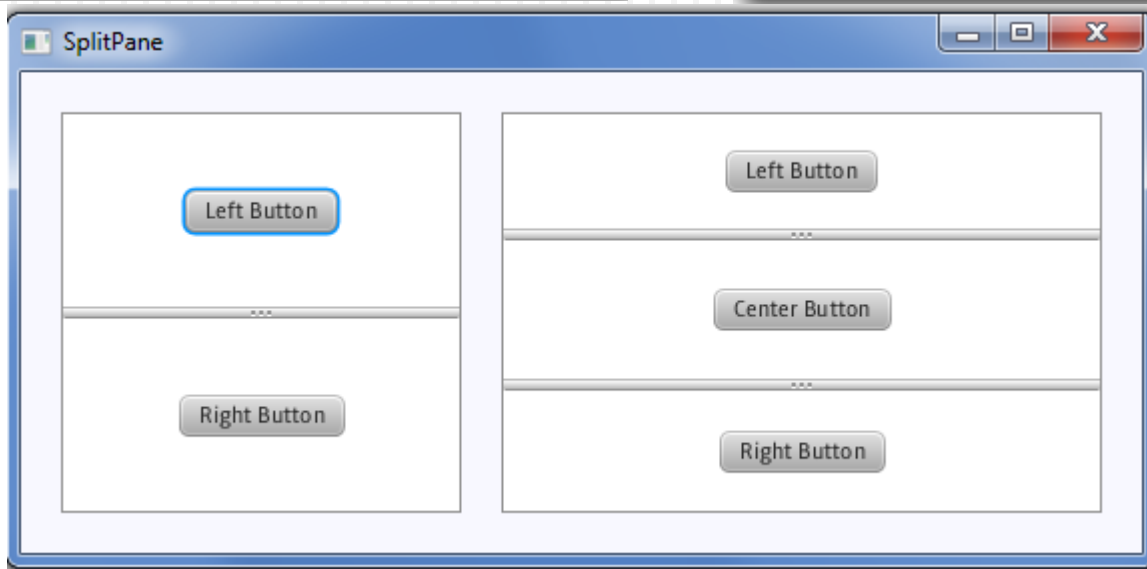
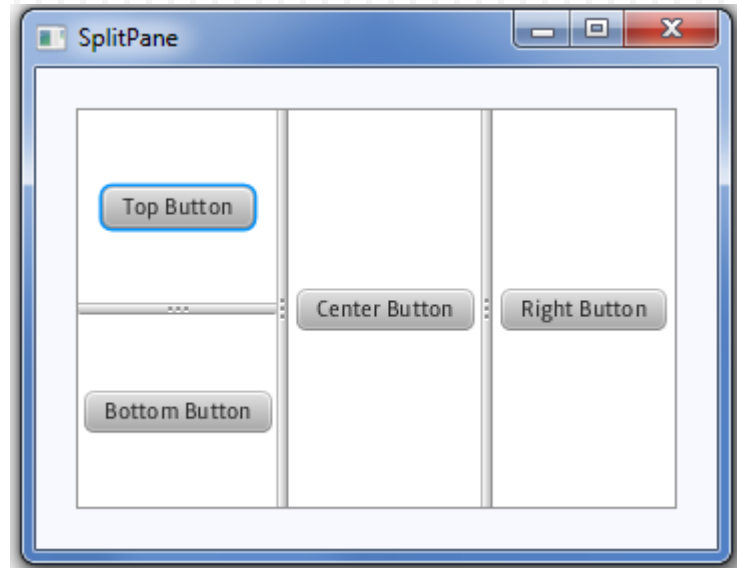
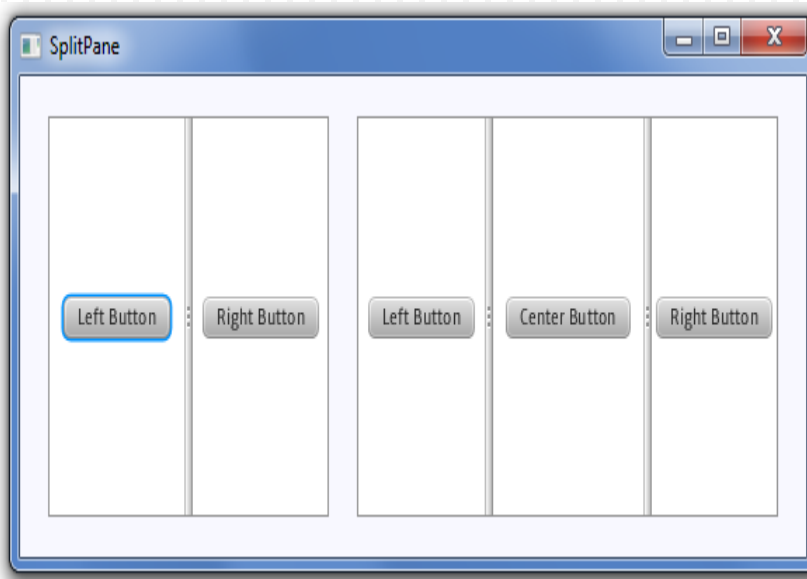


TilePane

- 그리드로 배치하며 고정된 크기를 갖는 컨테이너



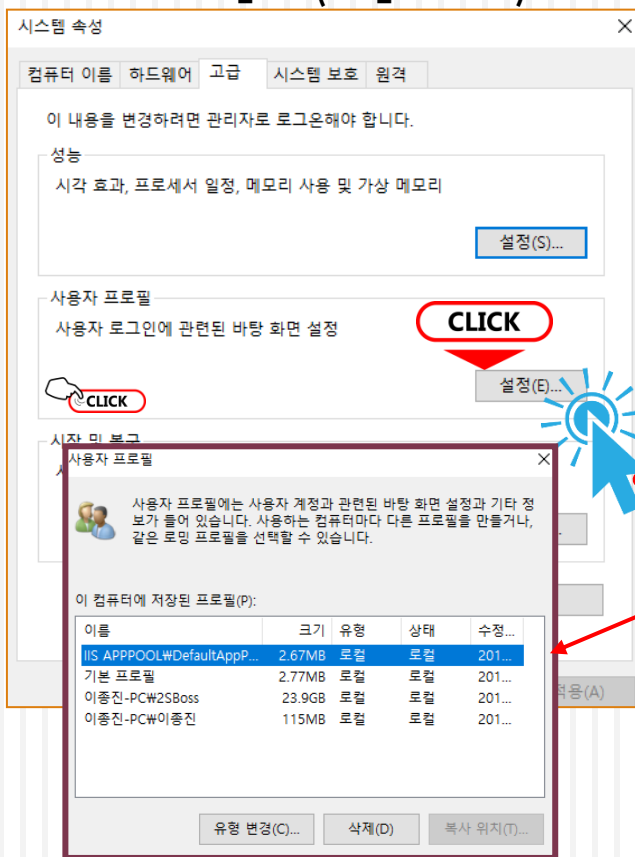
SplitPane



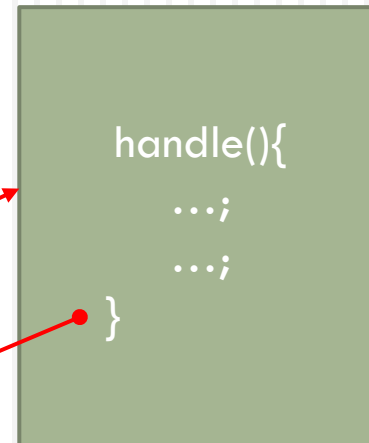
JavaFx 이벤트 처리

- 사용자가 UI 컨트롤을 사용하면 이벤트가 발생하고 프로그램은 그 이벤트를 처리하기 위해 특정 코드를 실행한다.
 - 이벤트소스 : 이벤트를 발생 시키는 화면
 - 이벤트처리기 : 이벤트가 발생하면 이벤트를 처리하기 위한 코드 - 클래스나 함수로 구현됨

이벤트소스(화면 :view)



이벤트처리기(EventHandler)



이벤트 처리(위임형:delegation 처리)- 883p

위임형 이벤트 처리

컨트롤에서 이벤트가 발생하면 컨트롤이 직접 처리하지 않고 이벤트핸들러에게 이벤트를 처리를 위임하는 방식

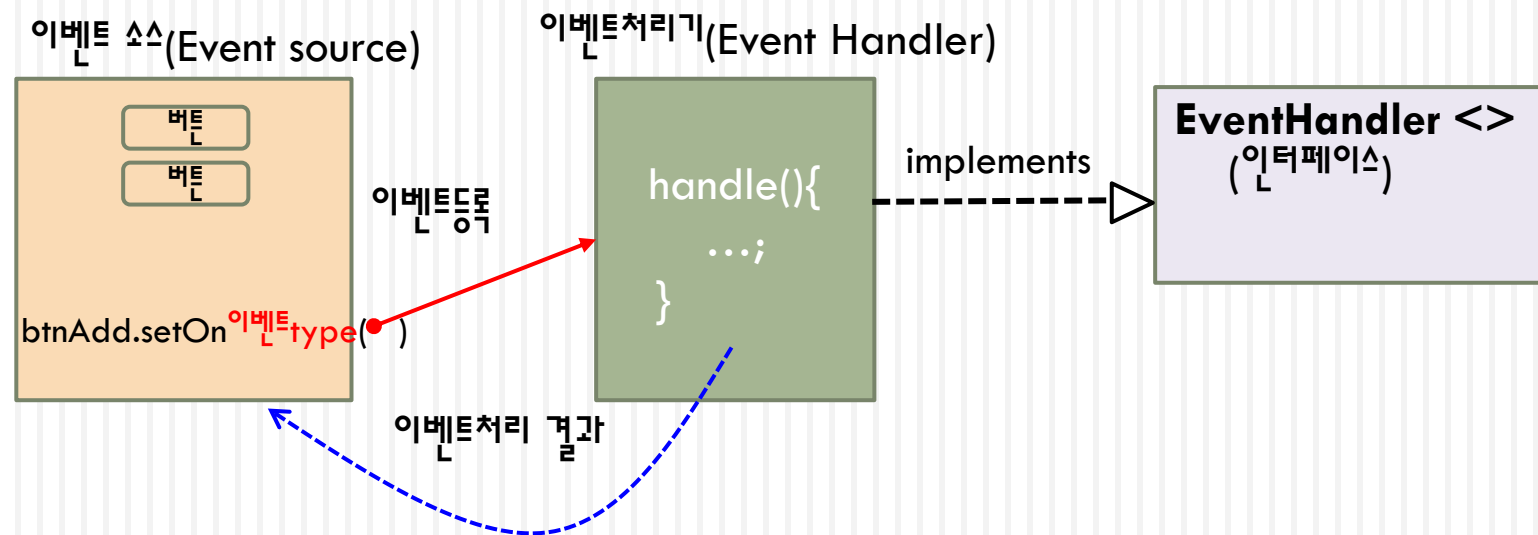
단계 1 : 이벤트소스(컨트롤)에 이벤트핸들러 type 등록(setOnXXX(*))

예: btn.setOnMouseClicked(***)

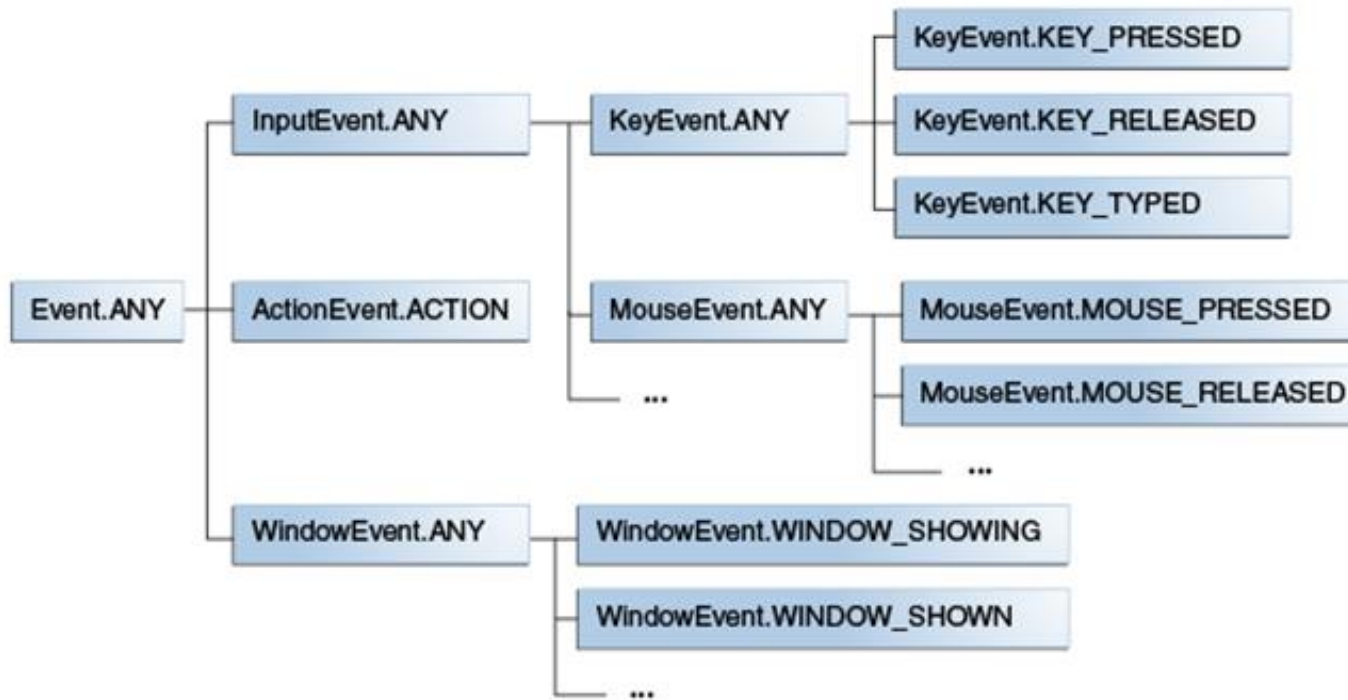
단계 2 : 이벤트핸들(처리: 메서드) 구현 (btn.setOnMouseClicked(이벤트처리객체))

- * 이벤트처리 클래스는 EventHandler <T> 인터페이스를 상속받아야 함
- * 이벤트소스 내부에 구현 : 무명내부 클래스, 람다식, 내부클래스 이용 -884p
- * controller : 별도의 외부 controller 클래스 지정 - 886p

단계 3: 이벤트처리 결과 반영

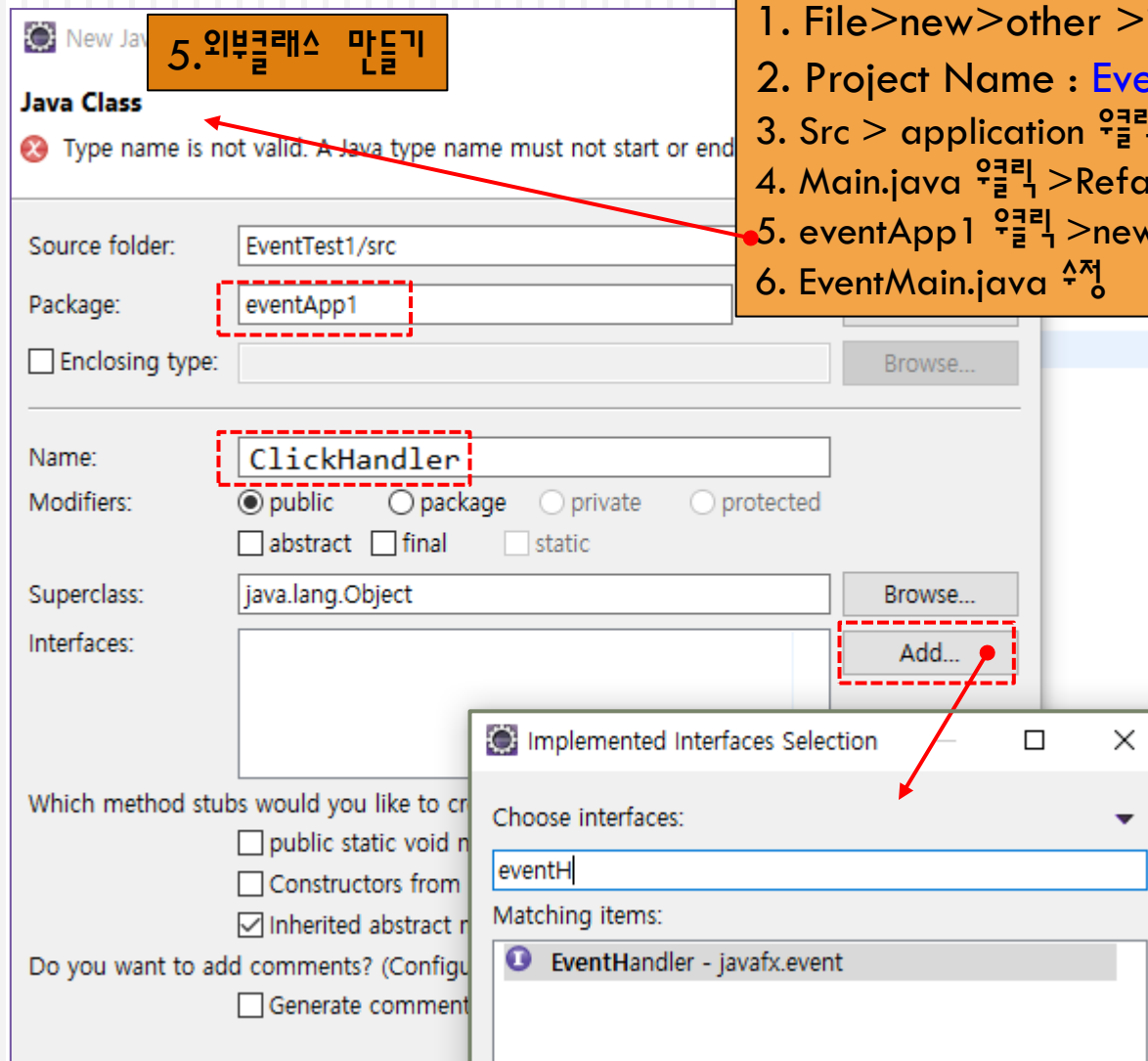


이벤트 type



- `setOnAction(EventHandler<ActionEvent>`
- `setOnContextMenuRequested(EventHandl`
- `setOnDragDetected(EventHandler<? supe`
- `setOnDragDone(EventHandler<? super Dr`
- `setOnDragDropped(EventHandler<? super`
- `setOnDragEntered(EventHandler<? super`
- `setOnDragExited(EventHandler<? super D`
- `setOnDragOver(EventHandler<? super Dra`
- `setOnInputMethodTextChanged(EventHar`
- `setOnKeyPressed(EventHandler<? super K`
- `setOnKeyReleased(EventHandler<? super I`
- `setOnKeyTyped(EventHandler<? super Key`
- `setOnMouseClicked(EventHandler<? supe`
- `setOnMouseDragEntered(EventHandler<? s`
- `setOnMouseDragExited(EventHandler<? s`
- `setOnMouseDragged(EventHandler<? sup`
- `setOnMouseDragOver(EventHandler<? su`
- `setOnMouseDragReleased(EventHandler<`
- `setOnMouseEntered(EventHandler<? supe`
- `setOnMouseExited(EventHandler<? super`
- `setOnMouseMoved(EventHandler<? super`
- `setOnMousePressed(EventHandler<? supe`
- `setOnMouseReleased(EventHandler<? sup`
- `setOnRotate(EventHandler<? super Rotate`
- `setOnRotationFinished(EventHandler<? su`
- `setOnRotationStarted(EventHandler<? sup`
- `setOnScroll(EventHandler<? super Scrolle`
- `setOnScrollFinished(EventHandler<? supe`
- `setOnScrollStarted(EventHandler<? super`
- `setOnSwipeDown(EventHandler<? super S`
- `setOnSwipeLeft(EventHandler<? super Sw`
- `setOnSwipeRight(EventHandler<? super S`
- `setOnSwipeUp(EventHandler<? super Swi`
- `setOnTouchMoved(EventHandler<? super`
- `setOnTouchPressed(EventHandler<? super`
- `setOnTouchReleased(EventHandler<? supe`
- `setOnTouchStationary(EventHandler<? su`
- `setOnZoom(EventHandler<? super ZoomI`

외부클래스를 이용한 이벤트핸들러 구현 - eventApp1 (885p)



1. File>new>other >javaFx> JavaFx Project
2. Project Name : EventTest
3. Src > application 우클릭 > Refactor>Rename>ok>eventApp1
4. Main.java 우클릭 > Refactor>Rename>ok>EventMain
5. eventApp1 우클릭 > new > class > ClickHandler
6. EventMain.java 수정

클래스 수정 - eventApp1 (885p)

```
10 public class EventMain extends Application {
11
12     @Override
13     public void start(Stage primaryStage) throws Exception {
14
15         HBox root = new HBox();
16         root.setPrefSize(200,50);
17         root.setAlignment(Pos.CENTER);
18         root.setSpacing(20);
19
20         ClickHandler ch = new ClickHandler();
21         Button btn1 = new Button("버튼1");
22         btn1.setOnAction(ch);
23
24         Button btn2 = new Button("버튼2");
25
26
27         root.getChildren().addAll(btn1,btn2);
28         Scene scene = new Scene(root);
29
30         primaryStage.setTitle("EventApp");
31         primaryStage.setScene(scene);
32         primaryStage.show();
33
34     }
35
36
37     public static void main(String[] args) {
38         launch(args);
39     }
40 }
```

1. 외부 클래스 입력

```
1 package eventApp1;
2
3 import javafx.event.ActionEvent;
4 import javafx.event.EventHandler;
5
6 public class ClickHandler implements EventHandler<ActionEvent> {
7     @Override
8     public void handle(ActionEvent e) {
9         System.out.println("버튼1 클릭");
10    }
11 }
```

2. Main 클래스 수정

3. 별도의 이벤트 처리
클래스 생성 필요

버튼1

버튼2

내부중첩클래스를 이용한 이벤트처리 - eventApp2 (885p)

New Java Class

Java Class

Create a new Java class.

Source folder: EventApp/src Browse...

Package: eventApp0 Browse...

☐ Enclosing type: Browse...

Name: AppMain

Modifiers: ☒ public ☐ package ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass: java.lang.Object Browse...

Interfaces:

Which method stubs would you like to create?

☒ public static void main(String[] args)
☐ Constructors from superclass
☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))
☐ Generate comments

Finish Cancel

1. Src 우클릭 > new > Package > eventApp2 입력
2. eventApp2 우클릭 > new > class > event EventMain 입력
3. Application 클래스 상속, main() 포함 체크
4. start() 와 main() 에 eventApp1 의 내용을 그대로 복사
5. 실행 확인
6. ClickHandler.java 파일의 클래스를 EventMain.java의 내부 클래스로 복사
7. ClickHandler.java 파일 삭제

Superclass Selection

Choose a type:

a

Matching items:

Application - javafx.application - [jdk1.8_112]

AATextListener - javax.swing.plaf.metal.tk1.8.0_112]

AATextListener - javax.swing.plaf.synth.tk1.8.0_112]

AbstractChronology

javafx.application - [jdk1.8.0_112]

OK Cancel

내부 클래스를 이용한 이벤트처리 - eventApp2 (885p)

```
13 public class EventMain extends Application {
14
15     @Override
16     public void start(Stage primaryStage) throws Exception {
17
18         HBox root = new HBox();
19         root.setPrefSize(200, 50);
20         root.setAlignment(Pos.CENTER);
21         root.setSpacing(20);
22
23         Button btn1 = new Button("버튼1");
24
25         ClickHandler ch = new ClickHandler();
26         btn1.setOnAction(ch);
27
28         Button btn2 = new Button("버튼2");
29
30
31         root.getChildren().addAll(btn1, btn2);
32         Scene scene = new Scene(root);
33
34         primaryStage.setTitle("EventApp");
35         primaryStage.setScene(scene);
36         primaryStage.show();
37
38     }
39
40     // 내부 클래스
41     private class ClickHandler implements EventHandler<ActionEvent> {
42         @Override
43         public void handle(ActionEvent e) {
44             System.out.println("버튼1 클릭");
45         }
46     }
47
48     public static void main(String[] args) {
49         launch(args);
50     }
51 }
```

2. 이벤트처리 내부 클래스의 객체를 생성하고
리스너에 연결 (import 확인)

3. ClickHandler.java 파일 삭제
실행 및 결과 확인

4. ClickHandler2 새 내부 클래스를
생성하여 테스트

1. 복사

전역변수를 이용한 내부클래스와 데이터 교환

```
public class EventMain extends Application {
```

```
    Button btn1; //전역변수  
    Button btn2;
```

```
    @Override
```

```
    public void start(Stage primaryStage) throws Exception {
```

```
        HBox root = new HBox();  
        root.setPrefSize(200,50);  
        root.setAlignment(Pos.CENTER);  
        root.setSpacing(20);
```

```
        ClickHandler ch = new ClickHandler();
```

```
        btn1 = new Button("버튼1");  
        btn1.setOnAction(ch);
```

```
        btn2 = new Button("버튼2");  
        btn2.setOnAction(ch);
```

```
        root.getChildren().addAll(btn1, btn2);  
        Scene scene = new Scene(root);
```

```
        primaryStage.setTitle("EventApp");  
        primaryStage.setScene(scene);  
        primaryStage.show();
```

```
    }
```

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

```
//내부 클래스
```

```
private class ClickHandler implements EventHandler<ActionEvent> {
```

```
    @Override
```

```
    public void handle(ActionEvent e) {
```

```
        if (e.getSource()==btn1){  
            System.out.println("버튼1 클릭");
```

```
        }
```

```
        else{  
            System.out.println("버튼2 클릭");
```

```
        }
```

```
    }
```

```
}
```

무명클래스와 람다식을 이용한 이벤트처리 – eventApp3 (885p)

```
30 import javafx.application.Application;
12
13
14 public class EventMain extends Application {
15     @Override
16     public void start(Stage primaryStage) {
17         HBox root = new HBox();
18         root.setPrefSize(200,50);
19         root.setAlignment(Pos.CENTER);
20         root.setSpacing(20);
21
22         Button btn1 = new Button("버튼1");
23         //ClickHandler ch = new ClickHandler();
24         //btn1.setOnAction(ch);
25
26         //무명 클래스 사용
27         btn1.setOnAction(new EventHandler<ActionEvent>(){
28             @Override
29             public void handle(ActionEvent e){
30                 System.out.println("버튼1 클릭");
31             }
32         });
33
34         Button btn2 = new Button("버튼2");
35         //람다식 (무명함수) 사용
36         btn2.setOnAction(e-> System.out.println("버튼2 클릭"));
37
38         root.getChildren().addAll(btn1,btn2);
39         Scene scene = new Scene(root);
40
41         primaryStage.setTitle("EventApp");
42         primaryStage.setScene(scene);
43         primaryStage.show();
44     }
45
46     public static void main(String[] args) {
47         launch(args);
48     }
49 }
50 }
```

1. Src 우클릭 > new > Package > eventApp3 입력
2. eventApp3 우클릭 > new > class > event EventMain 입력
3. Application 클래스 상속, main() 포함 체크
4. start() 와 main() 에 eventApp2의 내용을 그대로 복사
5. 내부클래스(ClickHandler) 복사
6. 내부클래스(ClickHandler) 주석 표시

무명 클래스를 이용한 이벤트 처리

람다식을 이용한 이벤트 처리

버튼1 버튼2

- 실행 결과와 exception 은 console 에 나타남
- Console 이 나타나지 않으면 메뉴 > window > Show View 에서 console 선택

무명클래스와 무명함수(람다식)- eventApp3 (885p)

```
3+ import javafx.application.Application;
12
13
14 public class EventMain extends Application {
15     @Override
16     public void start(Stage primaryStage) {
17         HBox root = new HBox();
18         root.setPrefSize(200,50);
19         root.setAlignment(Pos.CENTER);
20         root.setSpacing(20);
21
22         Button btn1 = new Button("버튼1");
23         //ClickHandler ch = new ClickHandler();
24         btn1.setOnAction(ch);
25
26         //무명 클래스 사용
27         btn1.setOnAction(new EventHandler<ActionEvent>(){
28             @Override
29             public void handle(ActionEvent e){
30                 System.out.println("버튼1 클릭");
31             }
32         });
33
34         Button btn2 = new Button("버튼2");
35         //람다식 (무명함수) 사용
36         btn2.setOnAction(e-> System.out.println("버튼2 클릭"));
37
38         root.getChildren().addAll(btn1,btn2);
39         Scene scene = new Scene(root);
40
41         primaryStage.setTitle("EventApp");
42         primaryStage.setScene(scene);
43         primaryStage.show();
44     }
45
46     public static void main(String[] args) {
47         launch(args);
48     }
49 }
50 }
```

1

2: 클래스명 생략

Btn1.setOnAction (new ClickHadler())

이벤트리스너에 클래스정의와 객체 생성을 동시에 한다.
이때 클래스 이름 생략한 형태

람다식 : 클래스명과 함수명까지 생략한 형태

//내부 클래스

```
private class ClickHandler implements EventHandler<ActionEvent> {
    @Override
    public void handle(ActionEvent e) {
        System.out.println("버튼1 클릭");
    }
}
```

람다식

- 1936년 알론조 처치(Alonzo Church) : 람다 대수 체계 발표
- 제자 존 메카시(mccarthy)가 함수형 프로그래밍 언어에 도입
- 람다식은 익명함수를 생성하기 위한 식 → 함수 지향 프로그램

```
void kim ( )  
{  
    ... ... ;  
}
```

- (매개변수) -> { body } //람다식 기본형
 - (int a) -> { Syatem.out.println(a); }
 - (a) -> { Syatem.out.println(a); } //람다식에서는 매개변수 type 자동인식
 - a -> Syatem.out.println(a) //매개변수 1개:() 생략, 실행문 1개:{ } ;생략
 - () -> Syatem.out.println(a) //매개변수 없다면 빈괄호

람다식과 별도함수를 사용한 이벤트처리 eventApp3 (885p)

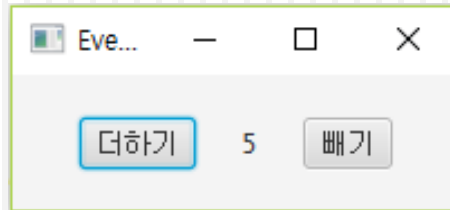
```
public class AppMain extends Application {  
  
    @Override  
    public void start(Stage primaryStage) throws Exception {  
        HBox root = new HBox();  
        root.setPrefSize(200,50);  
        root.setAlignment(Pos.CENTER);  
        root.setSpacing(20);  
  
        Button btn1 = new Button("버튼1");  
        btn1.setOnAction(e -> btn1Click());  
  
        Button btn2 = new Button("버튼2");  
        btn2.setOnAction(e -> System.out.println("버튼2 클릭"));  
  
        root.getChildren().addAll(btn1, btn2);  
        Scene scene = new Scene(root);  
  
        primaryStage.setTitle("EventApp");  
        primaryStage.setScene(scene);  
        primaryStage.show();  
    }  
  
    public void btn1Click(){  
        System.out.println("버튼1 클릭");  
    }  
  
    public static void main(String[] args) {  
        launch(args);  
    }  
}
```

1. Src **우클릭** > new > Package > eventApp4 **입력**
2. eventApp3 **우클릭** > new > class > event **EventMain** **입력**
3. Application **클래스 상속**, main() **포함 체크**
4. start() 와 main() 에 eventApp3의 내용을 그대로 복사

- 이벤트처리에 람다식을 사용할 경우 이벤트처리 문장이 많아질 수록 직접 정의가 어려워진다.
- 별도의 이벤트함수를 정의하여 사용하면 가독성이 높아지고 편리하다.
- 가장 일반적 형태이나 이벤트소스와 이벤트처리가 분리되지 않아 코드가 복잡해지고, 유지보수가 어려워지며, 디자이너와 협력개발이 쉽지 않다.

Event handler 연습

- 그림과 같이 작동하는 이벤트처리 프로그램을 내부클래스, 익명클래스, 람다식 별도 함수 방식으로 구현



- 이벤트처리 결과를 이벤트소스의 Label 컨트롤에 나타나게 한다.

1. “CountEvent” javaFx project 생성
2. application package 를 countApp1 로 변경
 - countApp1 – 외부 클래스를 이용한 이벤트핸들러 작성 (구현 어려움)
 - countApp2 – 내부 클래스를 이용한 이벤트핸들러 작성
 - countApp3 – 무명 클래스와 람다식을 이용한 이벤트핸들러 작성
 - countApp4 – 람다식 이벤트처리 함수를 이용한 이벤트핸들러 작성

내부클래스를 이용한 이벤트핸들러 구현(1) countApp2

```
15 public class EventMain extends Application {
16
17     Label lbl;
18     int iCounter = 0;
19     Button btnAdd ;
20     Button btnSub ;
21
22     @Override
23     public void start(Stage primaryStage) throws Exception {
24
25         HBox root = new HBox();
26         root.setPrefSize(200,60);
27         root.setAlignment(Pos.CENTER);
28         root.setSpacing(20);
29
30         lbl = new Label("0");
31         btnAdd = new Button("더하기");
32         btnSub = new Button("빼기");
33
34         ClickHandler ch = new ClickHandler();
35         btnAdd.setOnAction(ch);
36         btnSub.setOnAction(ch);
37
38         root.getChildren().addAll(btnAdd, lbl, btnSub);
39         Scene scene = new Scene(root);
40
41         primaryStage.setTitle("CountApp");
42         primaryStage.setScene(scene);
43         primaryStage.show();
44
45     }
```

```
47     public static void main(String[] args) {
48         launch(args);
49     }
50
51     private class ClickHandler implements EventHandler<ActionEvent>
52     {
53         @Override public void handle(ActionEvent e) {
54             if (e.getSource()==btnAdd){
55                 iCounter++;
56             }
57             else{
58                 iCounter--;
59             }
60
61             lbl.setText(Integer.toString(iCounter));
62         }
63     }
64 }
```

이벤트핸들러와 이벤트소스사이 데이터 교환을
위하여 전역 변수 선언

무명클래스와 랴다식을 이뚱한 핸들러 구현(2) countApp3

```
13 public class AppMain extends Application {
14
15     Label lbl;
16     int iCounter = 0;
17     Button btnAdd ;
18     Button btnSub ;
19
20     @Override
21     public void start(Stage primaryStage) throws Exception {
22         HBox root =new HBox();
23         root.setPrefSize(200,60);
24         root.setAlignment(Pos.CENTER);
25         root.setSpacing(20);
26
27         lbl =new Label("0");
28         btnAdd =new Button("더하기");
29         btnSub =new Button("빼기");
30
31         btnAdd.setOnAction(new EventHandler<ActionEvent>() {
32             @Override public void handle(ActionEvent e) {
33                 iCounter++;
34                 lbl.setText(Integer.toString(iCounter));
35             }
36         });
37
38         btnSub.setOnAction(e-> {
39             iCounter--;
40             lbl.setText(Integer.toString(iCounter));
41         });
42
43         root.getChildren().addAll(btnAdd, lbl, btnSub);
44         Scene scene = new Scene(root);
45
46         primaryStage.setTitle("CVEventApp");
47         primaryStage.setScene(scene);
48         primaryStage.show();
49     }
50 }
```

1. Src **윅** > new > Package> **countApp3** **입력**
2. countApp3 **윅** >new> class > **AppMain** **입력**
3. Application 클래스 상속, main() 포함 체크
5. 전역변수, start() 와 main() 에 countApp2 의 내용 그대로 복사
6. 실행 확인 - 수정 시작

btnAdd: 무명클래스 로 구현
btnSub: 랴다식으로 구현

```
51
52 public static void main(String[] args) {
53     launch(args);
54 }
55 }
```

람다식과 별도 함수로 구현 (3)- countApp4

```
13 public class AppMain extends Application {
14
15     Label lbl;
16     int iCounter = 0;
17     Button btnAdd ;
18     Button btnSub ;
19
20     @Override
21     public void start(Stage primaryStage) throws Exception {
22
23         HBox root =new HBox();
24         root.setPrefSize(200,60);
25         root.setAlignment(Pos.CENTER);
26         root.setSpacing(20);
27
28         lbl =new Label("0");
29         btnAdd =new Button("더하기");
30         btnSub =new Button("빼기");
31
32         btnAdd.setOnAction(e-> btn1Click());
33         btnSub.setOnAction(e-> btn2Click());
34
35         root.getChildren().addAll(btnAdd, lbl, btnSub);
36         Scene scene = new Scene(root);
37
38         primaryStage.setTitle("CVEventApp");
39         primaryStage.setScene(scene);
40         primaryStage.show();
41     }
```

1. Src 우클릭 > new > Package > countApp4 입력
2. countApp2 우클릭 > new > class > AppMain 입력
3. Application 클래스 상속, main() 포함
4. start() 와 main()에 countApp3 의 내용을 복사
5. 실행 확인 - 수정 시작

```
43     public void btn1Click() {
44         iCounter++;
45         lbl.setText(Integer.toString(iCounter));
46     }
47     public void btn2Click() {
48         iCounter--;
49         lbl.setText(Integer.toString(iCounter));
50     }
51
52     public static void main(String[] args) {
53         Launch(args);
54     }
```

Event Delivery Process(이벤트 전달 처리)

1.Target selection :

마우스의 경우 위치, 키보드의 경우 포커스가 잡힌 노드 (UI객체)

2.Route construction :

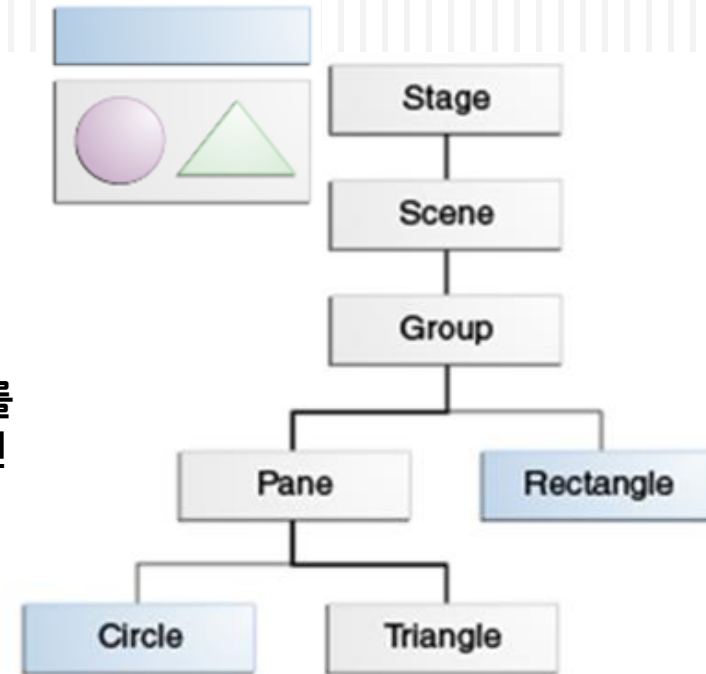
경로 선택: Event Dispatch Chain stage 에서 이벤트 발생노드 까지

3.Event capturing:

상위노드 부터 이벤트 필터 호출을 통해 등록된 이벤트를 확인, 없으면 하위 노드로 전달, 결국 이벤트를 발생시킨 노드까지 이벤트 필터 호출

4. Event bubbling:

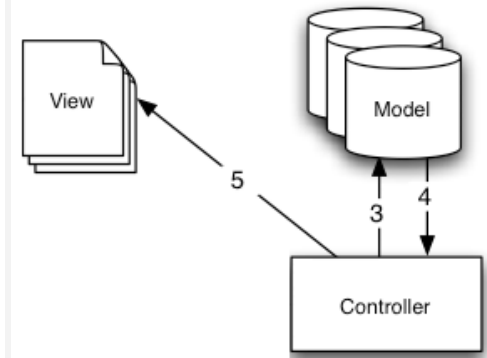
하위 노드에서 부터 등록된 이벤트 처리, 완료되면 상위노드 이벤트 처리, 마지막 루트 노드가 이벤트를 수신하고 처리가 완료된다.



MVC 모델

- 자바 프로그램에서 gui를 구축하는 기본 구조
- GUI를 만들 때 **화면설계, 데이터저장, 이벤트처리** 등을 한곳에 모아 두는 것 보다는 분리하는 것이 개발이나 유지보수에 효과적

- **View** – 화면표시, 데이터를 시각적으로 표현, 유저 인터페이스를
- **Controller** – 사용자 입력을 처리, 로직구현, 이벤트처리 기능
- **Model** – 데이터의 저장과 접근기능

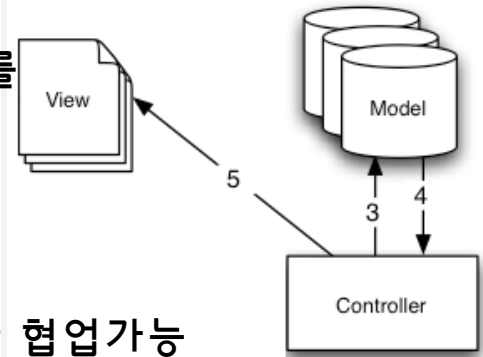


- 이벤트처리기에서 처리한 결과는 대부분 이벤트소스에 표현되나 이벤트처리기에서 이벤트소스 자원(화면)에 접근이 어렵다.
- 완전한 MVC 구조는 구현하기 어렵기 때문에 View가 하는 역할과 Controller의 역할을 합쳐서 수행하는 **Delegate(View + Controller) 구조**를 구현한다. - 복잡성감소, 생산성 증가, 컴퍼넌트 설계 단순화 – 프로그램적 레이아웃
- FXML 레이아웃에서는 완전한 MVC 모델 구현 가능

MVC 모델과 FXML 레이아웃

- MVC 모델 : 자바 프로그램에서 gui를 구축하는 기본 구조
- GUI를 만들 때 **화면설계, 데이터저장, 이벤트처리** 등을 한곳에 모아 두는 것 보다는 분리하는 것이 개발이나 유지보수에 효과적

- **View** – 화면표시 ,데이터를 시각적으로 표현, 유저 인터페이스를
- **Controller** – 사용자 입력을 처리, 로직구현, 이벤트처리 기능
- **Model** - 데이터의 저장과 접근기능

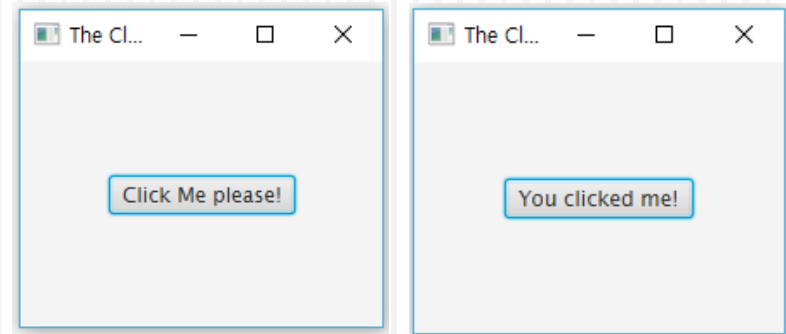


- **화면 (View)** - FXML layout(XML 기반의 마크업언어) 으로 디자이너와 협업가능
 - FXML layout - FXML 코드(XML)로 작성하거나 SceneBuilder로 작성
- **이벤트** - FXML layout에서 별도의 컨트롤러(controller)를 지정해서 이벤트 처리
- **데이터의 저장과 접근** : 컨트롤러에서 별도의 Model 프로그램을 지정해서 연동
- 프로그램과 디자인을 분리
 - 프로그램(controller, main, model) : 디자인(fxml, css)

프로그래믹 Layout 예제 (ClickMe)

```
public class Main extends Application {  
  
    Button btn;  
  
    @Override  
  
    public void start(Stage primaryStage) {  
  
        btn = new Button();  
        btn.setText("Click Me Please");  
  
        btn.setOnAction(e->buttonClick());  
  
        BorderPane root = new BorderPane();  
        root.setCenter(btn);  
  
        Scene scene = new Scene(root,400,400);  
        primaryStage.setScene(scene);  
  
        primaryStage.setTitle("The ClickMe Application");  
        primaryStage.show();  
  
    }  
  
    public void buttonClick(){  
        if(btn.getText()=="Click Me please!"){  
            btn.setText("You clicked me!");  
        }  
        else{  
            btn.setText("Click Me please!");  
        }  
    }  
}
```

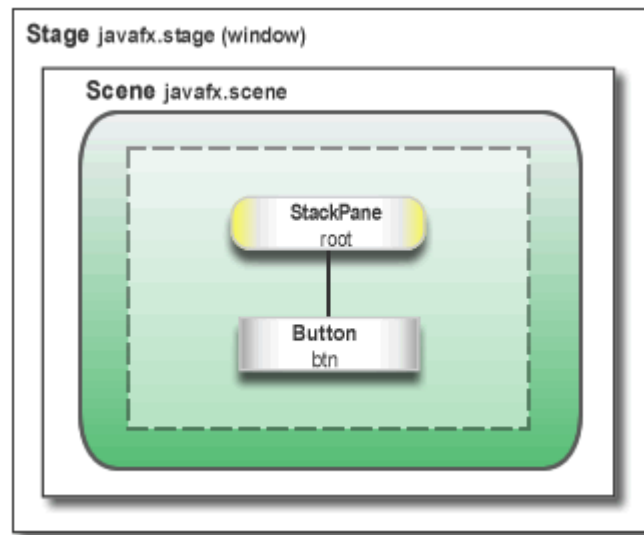
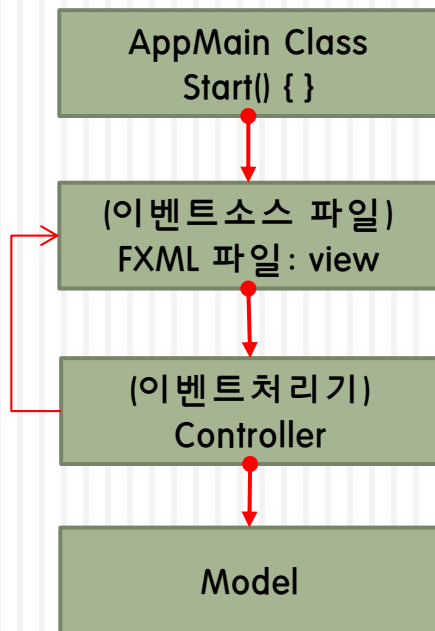
1. "ClickMe" JavaFx project 생성
2. Package 명 proApp 로 변경
3. Main.java → AppMain 변경



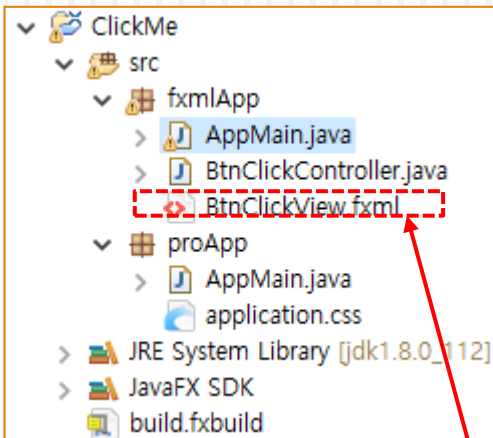
```
public static void main(String[] args) {  
    launch(args);  
}
```

FXML Layout

- FXML로 이벤트소스를 작성하면 코드가 단순해지고 디자이너와 협업이 가능해진다.
- JavaFx를 시작시키는 메인 클래스는 start() 에서 root Pane을 Scene에 고정하여 창의 크기를 결정하고 Stage에 보내 화면에 출력한다.
- FXML로 이벤트소스(화면)를 작성하면 root Pane이 ~.FXML 파일로 결정되므로 start() 에서 이 파일을 읽어와야 한다(FXMLLoader.load)



FXML Layout 예제 (ClickMe)



1. src 우클릭 > new > package > fxmlApp 입력
2. proApp AppMain.java 파일 복사
3. src > fxmlApp 우클릭 > New > other > javaFX > New FXML Document

```
1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <?import javafx.scene.layout.AnchorPane?>
4
5 <AnchorPane xmlns:fx="http://javafx.com/fxml/1">
6     <!-- TODO Add Nodes -->
7 </AnchorPane>
8
9
```

5. 확인

4. BtnClickView 입력

FXML File
Create a new FXML File

Source folder: ClickMe/src

Package: fxmlApp

Name: BtnClickView

Root Element: AnchorPane - javafx.scene.layout

Dynamic Root (fx:root) ☐

< Back Next > Finish Cancel

FXMLApp : AppMain 클래스 start() 메소드 수정

```
Button btn; //전역변수
```

```
@Override
```

```
public void start(Stage primaryStage) {
```

```
    btn = new Button();
```

```
    btn.setText("Click Me Please");
```

```
    btn.setOnAction(e->buttonClick());
```

```
    BorderPane root = new BorderPane();
```

```
    root.setCenter(btn);
```

```
    Scene scene = new Scene(root,400,400);
```

```
    primaryStage.setScene(scene);
```

```
    primaryStage.setTitle("The ClickMe Application");
```

```
    primaryStage.show();
```

```
}
```

FXML로 이벤트소스(화면)를 작성하면 root Pane이 ~.FXML 파일로 대체되므로 start() 에서 이 파일을 읽어와야 한다(FXMLLoader.load)

AppMain Class
Start() { }

(이벤트소스 파일)
FXML 파일: view

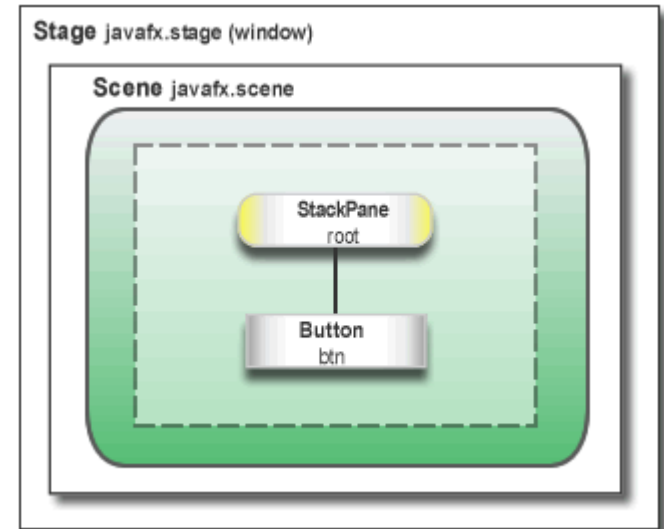
(이벤트처리기)
Controller

```
Parent root = FXMLLoader.load(getClass().getResource("/FXMLApp/BtnClickView.fxml"));  
Scene scene = new Scene(root);
```

```
AnchorPane root =(AnchorPane) FXMLLoader.load(getClass().getResource("/FXMLApp/BtnClickView.fxml));
```

FXMLApp : AppMain 클래스 start() 메소드 수정

```
public class AppMain extends Application {  
  
    // Button btn;  
  
    @Override  
    public void start(Stage primaryStage) throws Exception {  
  
        /*  
        btn = new Button();  
        btn.setText("Click Me Please");  
  
        btn.setOnAction(e->buttonClick());  
  
        BorderPane root = new BorderPane();  
        root.setCenter(btn);  
        Scene scene = new Scene(root,400,400);  
        */  
  
        Parent root = FXMLLoader.load(getClass().getResource("/FXMLApp/BtnClickView.fxml"));  
        Scene scene = new Scene(root);  
  
        primaryStage.setScene(scene);  
  
        primaryStage.setTitle("The FXML ClickMe Application");  
        primaryStage.show();  
    }  
}
```

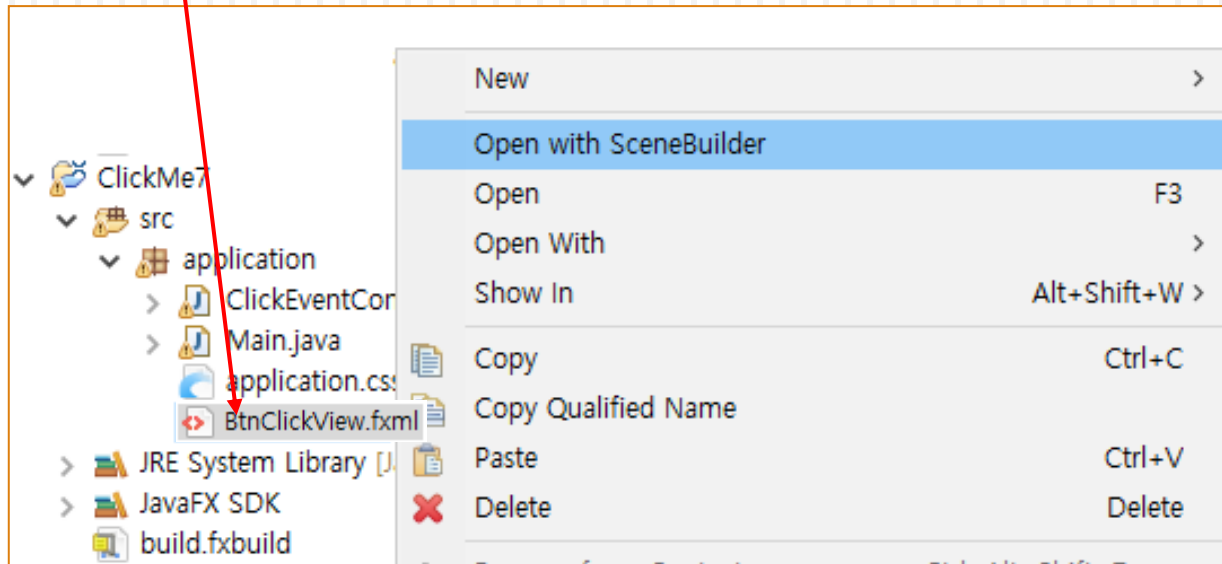


실행 → 컴파일

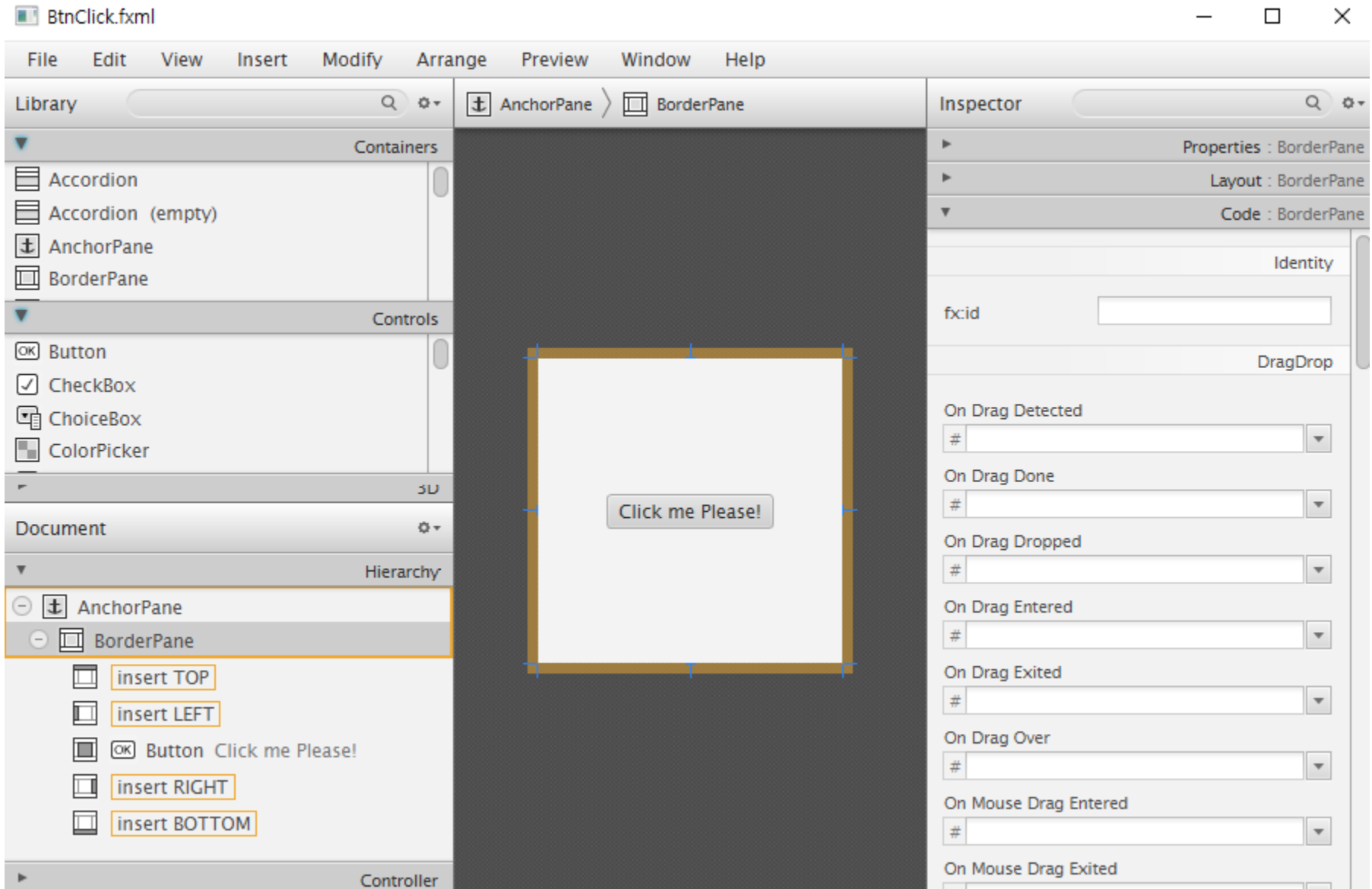
```
/* 컨트롤러에서 구현 - 외부 클래스  
public void buttonClick() {  
    if (btn.getText()=="Click me please!") {  
        btn.setText("You clicked me");  
    }  
    else {  
        btn.setText("Click me please!");  
    }  
}  
*/  
  
public static void main(String[] args) {  
    launch(args);  
}
```

FXML 레이아웃 작성 (SceneBuilder 사용)

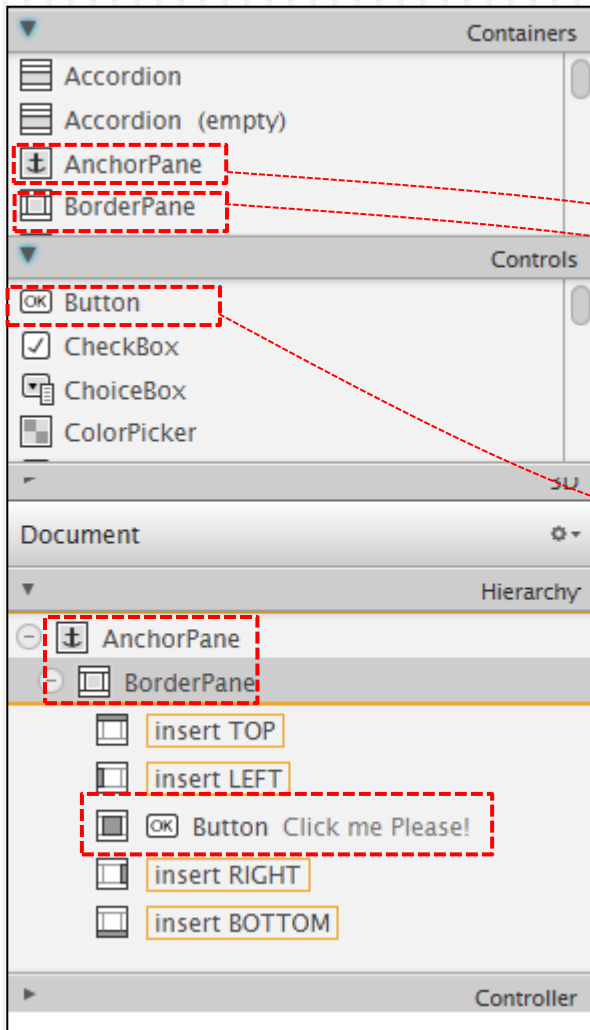
2. Fxml 파일 우클릭 > Open with SceneBuilder



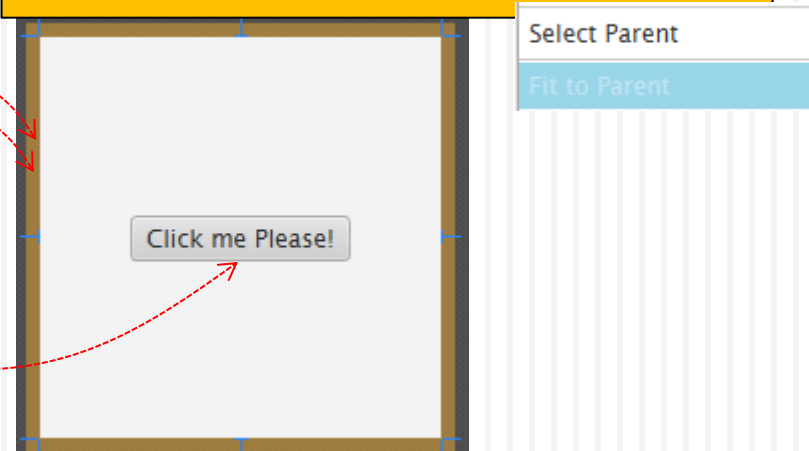
FXML 작성 (SceneBuilder)



FXML 작성



1. AnchorPane 드래그앤드롭
2. BorderPane 드래그앤드롭
3. BorderPane 우클릭 > Fit to Parent



4. BorderPane Center에 Button 컨트롤 드래그앤드롭

```
BorderPane root = new BorderPane();  
root.setCenter(btn);  
Scene scene = new Scene(root, 400, 400);
```

Properties, Layout, Code - BorderPane

Properties : AnchorPane

Node

Disable ☐

Opacity

Node Orientation **INHERIT**

Visible ☒

Focus Traversable ☐

Cache Shape ☒

Center Shape ☒

Scale Shape ☒

Opaque Insets ☐ ☐ ☐ ☐

0 > 0 0 0

Cursor **Inherited (Default)**

Effect **+**

JavaFX CSS

Style **+** **▼**

Style Class **+** **▼**

Stylesheets **+**

Id

Extras

Blend Mode **SRC_OVER**

Cache ☐

Cache Hint **DEFAULT**

Depth Test **INHERIT**

☐ ☐ ☐ ☐

0 > 0 0 0

Mouse Transpar... ☐

Pick On Bounds ☒

Layout : BorderPan

Anchor Pane Constraints

Internal

Padding ☐ ☐ ☐ ☐

0 > 0 0 0

Size

Min Width **USE_COMPUTED_SIZE**

Min Height **USE_COMPUTED_SIZE**

Pref Width 400

Pref Height 400

Max Width **USE_COMPUTED_SIZE**

Max Height **USE_COMPUTED_SIZE**

Width 400

Height 400

Position

Layout X -107

Layout Y -112

Transforms

Code : AnchorPane

Identity

fxid

DragDrop

On Drag Detected **#** **▼**

On Drag Done **#** **▼**

On Drag Dropped **#** **▼**

On Drag Entered **#** **▼**

On Drag Exited **#** **▼**

On Drag Over **#** **▼**

On Mouse Drag Entered **#** **▼**

On Mouse Drag Exited **#** **▼**

On Mouse Drag Over **#** **▼**

On Mouse Drag Released **#** **▼**

Keyboard

On Input Method Text Changed **#** **▼**

On Key Pressed **#** **▼**

On Key Released **#** **▼**

On Key Typed **#** **▼**

Mouse

Properties, Layout, Code - Button

Properties : Button





Text

Text: Click me please!

Font: System 12px

Text Fill: BLACK

Wrap Text: ☐

Text Alignment:    

Text Overrun: ELLIPSIS

Ellipsis String: ...

Underline: ☐

Line Spacing: 0

Specific

Default Button: ☐

Cancel Button: ☐

Graphic

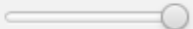
Graphic Text ...: 4

Content Display: CENTER

Node

Alignment: CENTER

Disable: ☐

Opacity: 1 

Node Orientation: INHERIT

Visible: ☒

Focus Travers...: ☒

Inspector

Properties : Button

Layout : Button

Border Pane Constraints

Alignment: CENTER

Margin: 0 > 0 0 0

Internal

Padding: 0 > 0 0 0

Size

Min Width: USE_COMPUTED_SIZE

Min Height: USE_COMPUTED_SIZE

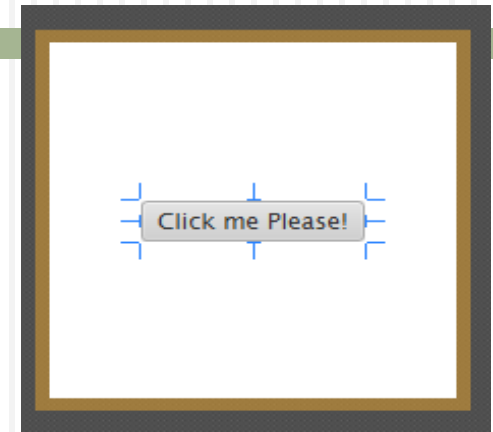
Pref Width: USE_COMPUTED_SIZE

Pref Height: USE_COMPUTED_SIZE

Max Width: USE_COMPUTED_SIZE

Max Height: USE_COMPUTED_SIZE

```
BorderPane root = new BorderPane();  
root.setCenter(btn);  
Scene scene = new Scene(root,400,400);
```



Code : Button

Identity

fx:id: btn

Main

On Action: # btnClick

DragDrop

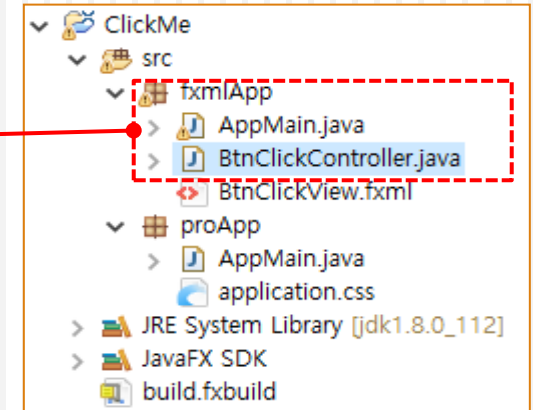
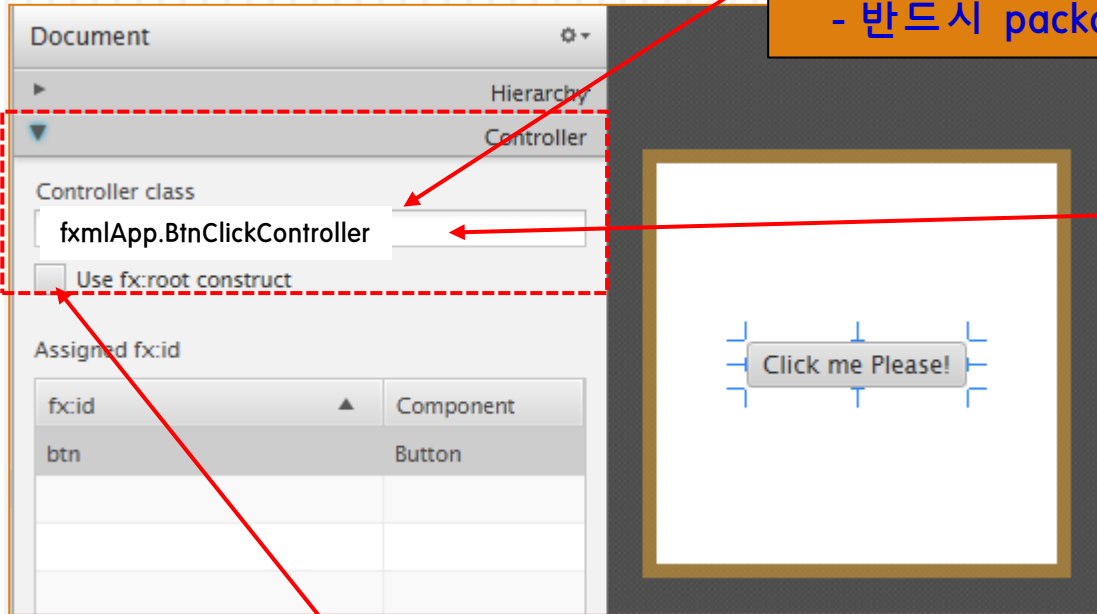
On Drag Detected: #

On Drag Done: #

On Drag Drropped: #

Controller 지정

1. 이벤트 처리 Controller 지정
 - `FXMLApp.BtnClickController`
 - 반드시 package 포함



2. Use fx:root constructor 체크해제 확인

3. Save

SceneBuilder가 작성한 FXML code

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

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

```
<?import java.lang.*?>
```

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

```
<?import javafx.scene.layout.AnchorPane?>
```

```
<AnchorPane xmlns:fx="http://javafx.com/fxml/1" xmlns="http://javafx.com/javafx/8" fx:controller="FXMLApp.BtnClickController">
```

```
  <children>
```

```
    <BorderPane layoutX="-107.0" layoutY="-112.0" prefHeight="400.0" prefWidth="400.0" AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0" AnchorPane.topAnchor="0.0">
```

```
      <center>
```

```
        <Button fx:id="btn" alignment="CENTER" mnemonicParsing="false" onAction="#btnclick" text="Click me please!">
```

```
      </center>
```

```
    </BorderPane>
```

```
  </children>
```

```
</AnchorPane>
```

FXML code에서 controller skeleton(골격) 생성

1. Code 화면 오른쪽 클릭 > Source > Generate Controller

```
fx:controller="FXMLApp.BtnClickController">
```

```
<AnchorPane>
  <children>
    <BorderPane layoutX="-107.0" layoutY="-112.0" prefHeight="400.0" prefWidth="400.0" AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0" AnchorPane.topAnchor="0.0">
      <center>
        <Button fx:id="btn" alignment="CENTER" mnemonicParsing="false" onAction="#btnclick" text="Click me please!" BorderPane.alignment="CENTER">
        </center>
      </BorderPane>
    </children>
  </AnchorPane>
```

2. BtnClickController 확인 !!!

Source folder: ClickMe/src Browse...

Package: FXMLApp Browse...

Name: BtnClickController

Fields & Methods

- ☒ btn : Button
- ☒ btnclick : Button [#btn] - onAction(ActionEvent)

OK Cancel

Context menu options:

- Undo Text Change (Ctrl+Z)
- Revert File
- Save (Ctrl+S)
- Open With >
- Show In (Alt+Shift+W >)
- Cut (Ctrl+X)
- Copy (Ctrl+C)
- Paste (Ctrl+V)
- Quick Fix (Ctrl+1)
- Source >
- Refactor >
- Properties
- Open Selection (F3)
- Configure Namespaces...
- Run As >
- Toggle Comment
- Add Block Comment
- Remove Block Comment
- Cleanup Document...
- Format
- Format Active Element
- Generate Controller

FXML : controller 이벤트 핸들러 작성

```
package fxmlApp;
|
import javafx.fxml.FXML;

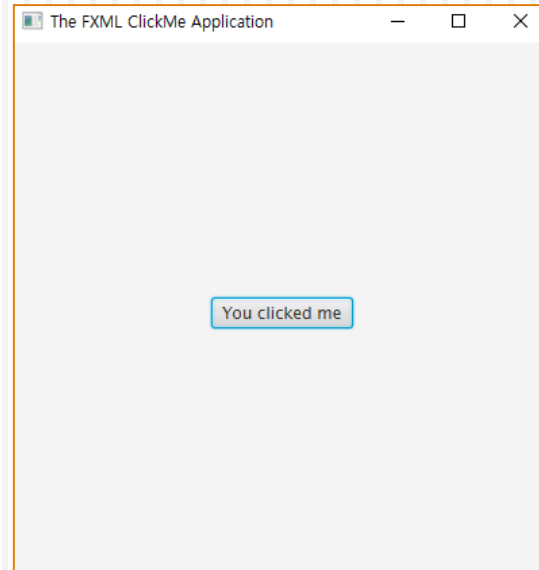
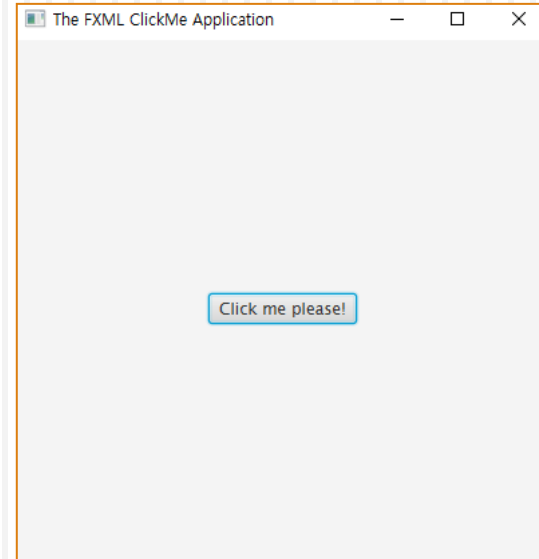
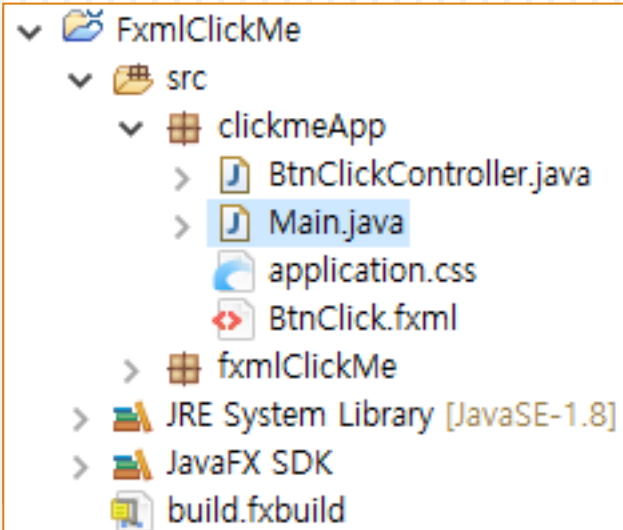
public class BtnClickController {
    @FXML
    private Button btn;

    // Event Listener on Button[#btn].onAction
    @FXML
    public void btnclick(ActionEvent event) {
        if (btn.getText().equals("Click me please!")) {
            btn.setText("You clicked me");
        }
        else {
            btn.setText("Click me please!");
        }
    }
}
```

FXML 파일과 공유되는 자원

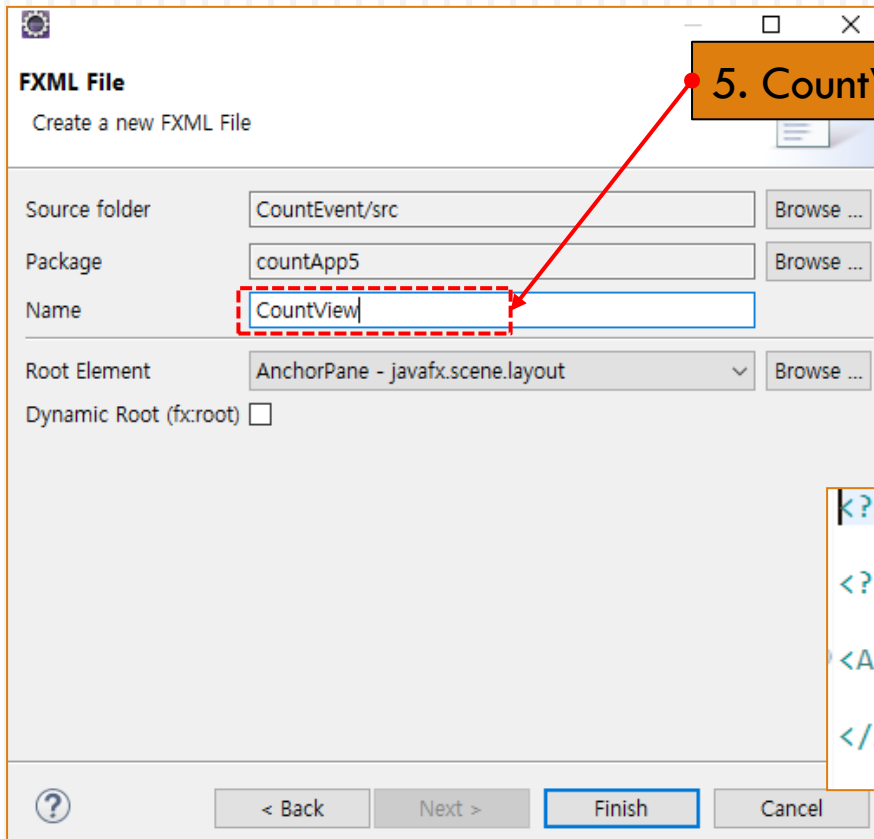
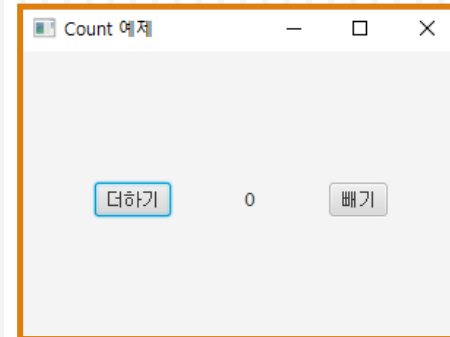
1. 복사 및 수정

실행

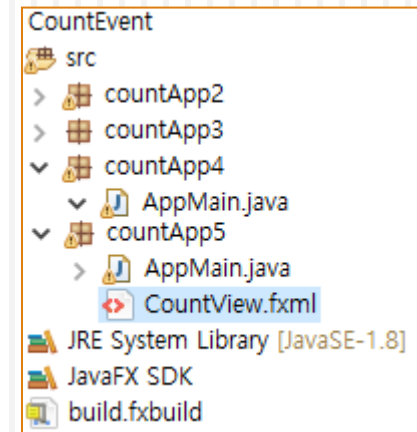


FXML 을 이용한 CountEvent 예제

1. CountEvent javaFx Project import
2. Src 우클릭 > new > package > countApp5 입력
3. countApp4의 AppMain 파일 복사
4. countApp5 우클릭 > New > other > javaFX > New FXML Document



5. CountView 입력



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

<?import javafx.scene.layout.AnchorPane?>

<AnchorPane xmlns:fx="http://javafx.com/fxml/1">
    <!-- TODO Add Nodes -->
</AnchorPane>
```

Start() 함수 수정

```
public class AppMain extends Application {
    /*
    Label lbl;
    int iCounter = 0;
    Button btnAdd ;
    Button btnSub ;
    */
    @Override
    public void start(Stage primaryStage) throws Exception {
        /*
        HBox root =new HBox();
        root.setPrefSize(200,60);
        root.setAlignment(Pos.CENTER);
        root.setSpacing(20);
```

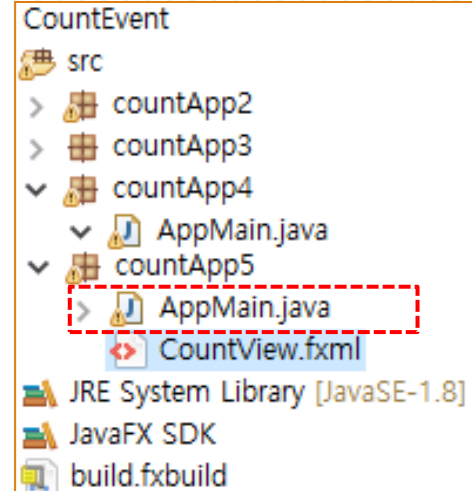
Parent root = FXMLLoader.load(getClass().getResource("/countApp5/CountView.fxml"));

```
        btnAdd =new Button("더하기");
        btnSub =new Button("빼기");

        btnAdd.setOnAction(e-> btn1Click());
        btnSub.setOnAction(e-> btn2Click());

        root.getChildren().addAll(btnAdd, lbl, btnSub);
        /*
        Scene scene = new Scene(root);

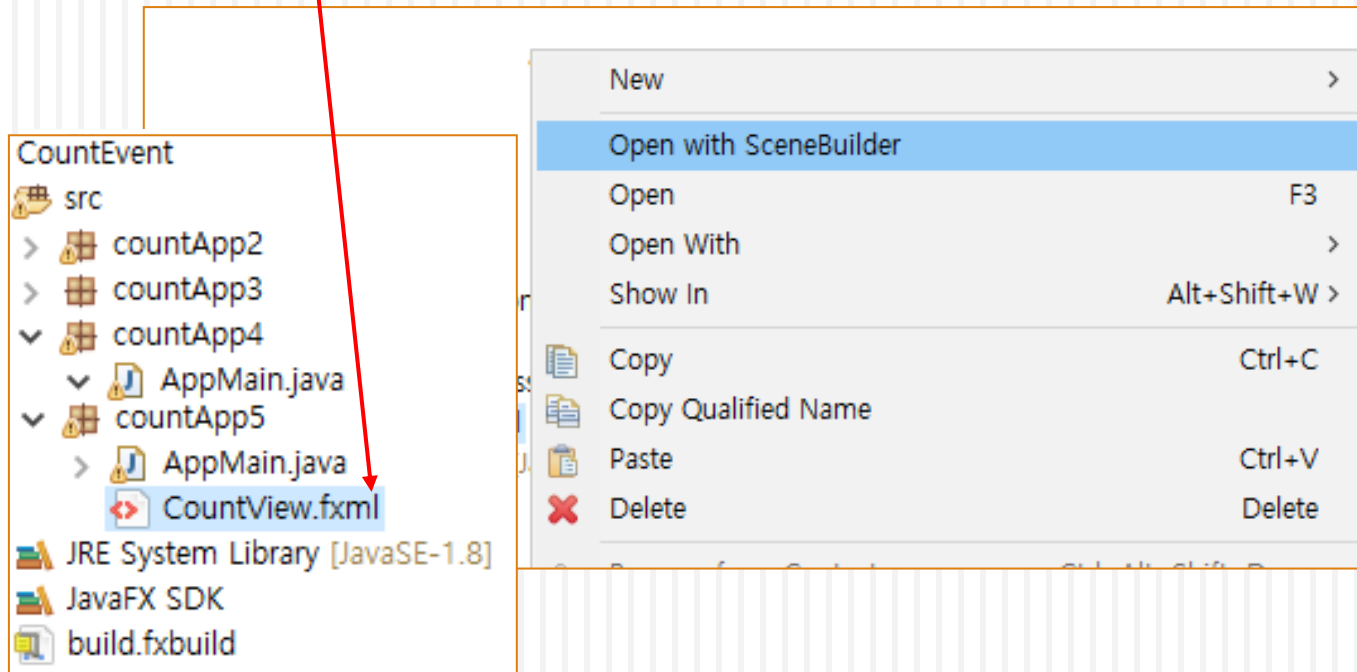
        primaryStage.setTitle("CVEventApp");
        primaryStage.setScene(scene);
        primaryStage.show();
    }
}
```



```
    /*
    public void btn1Click() {
        iCounter++;
        lbl.setText(Integer.toString(iCounter));
    }
    public void btn2Click() {
        iCounter--;
        lbl.setText(Integer.toString(iCounter));
    }
    */
    public static void main(String[] args) {
        launch(args);
    }
}
```

FXML 레이아웃 작성 (SceneBuilder 사용)

1. Fxml 파일 오른쪽 클릭 > Open with SceneBuilder



컨테이너 설정

```
HBox root = new HBox();  
root.setPrefSize(200, 60);  
root.setAlignment(Pos.CENTER);  
root.setSpacing(20);
```

```
lbl = new Label("0");  
btnAdd = new Button("더하기");  
btnSub = new Button("빼기");
```

CountView.fxml

File Edit View Insert Modify Arrange Preview Window Help

Library

Containers

Accordion
Accordion (empty)
AnchorPane
BorderPane
FlowPane
GridPane
HBox
Pane
ScrollPane
ScrollPane (empty)

Controls

Menu

Miscellaneous

Shapes

Charts

3D

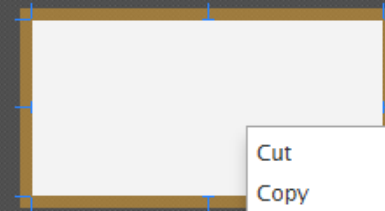
Document

Hierarchy

AnchorPane

HBox

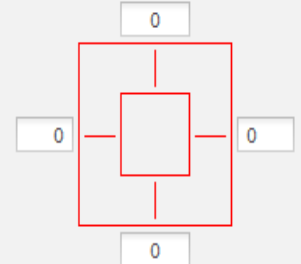
AnchorPane > HBox



Cut
Copy
Paste
Paste Into
Duplicate
Delete
Select Parent
Fit to Parent
Use Computed Sizes

Layout : HBox

Anchor Pane Constraints



Internal

Padding 0 0 0 0

Spacing 20

Specific

Fill Height ☒

Size

Min Width USE_COMPUTED_SIZE

Min Height USE_COMPUTED_SIZE

Pref Width 200

Pref Height 60

컨트롤 설정

```
HBox root = new HBox();  
root.setPrefSize(200, 60);  
root.setAlignment(Pos.CENTER);  
root.setSpacing(20);
```

```
lbl = new Label("0");  
btnAdd = new Button("더하기");  
btnSub = new Button("빼기");
```

3. 클릭

1. 드래깅

2. 선택

- Alignment
- TOP_LEFT
 - ✓ TOP_LEFT
 - TOP_CENTER
 - TOP_RIGHT
 - CENTER_LEFT
 - CENTER**
 - CENTER_RIGHT
 - BOTTOM_LEFT
 - BOTTOM_CENTER
 - BOTTOM_RIGHT
 - BASELINE_LEFT
 - BASELINE_CENTER
 - BASELINE_RIGHT

Style

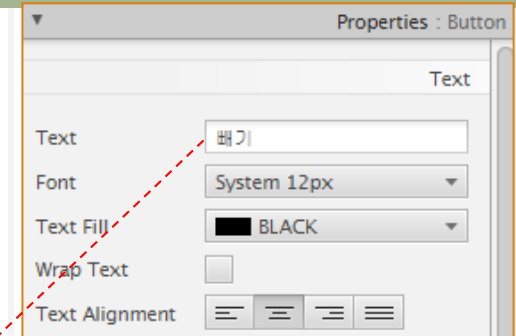
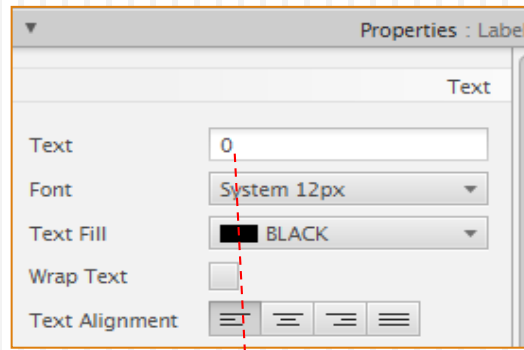
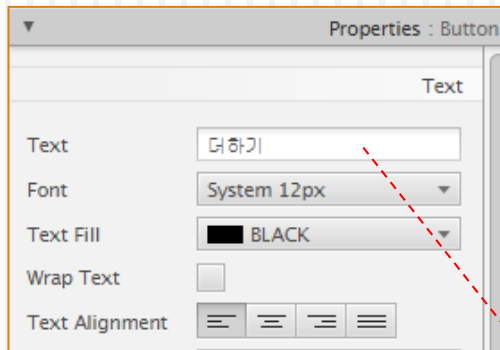
Style Class

Stylesheets

+

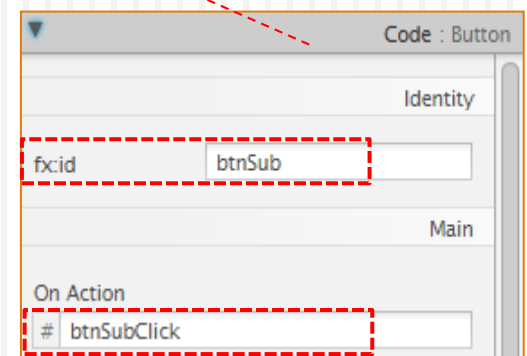
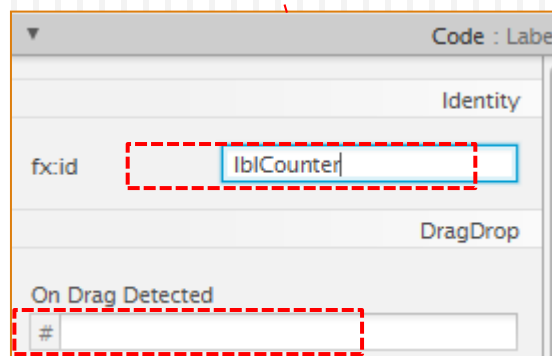
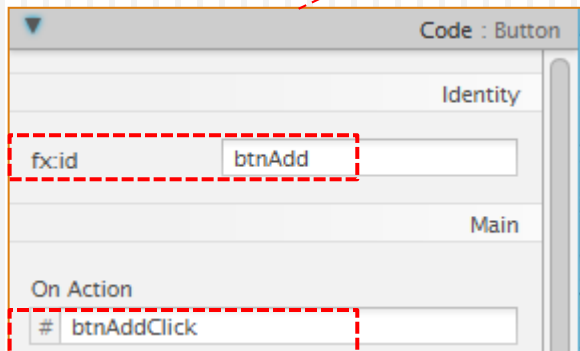
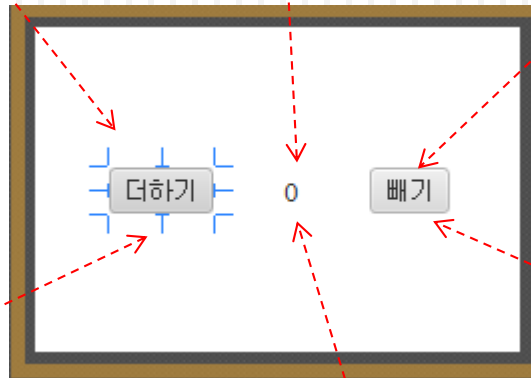
Id

Text, Fx:id 속성, 이벤트핸들러 설정



```
HBox root =new HBox();
root.setPrefSize(200,60);
root.setAlignment(Pos.CENTER);
root.setSpacing(20);

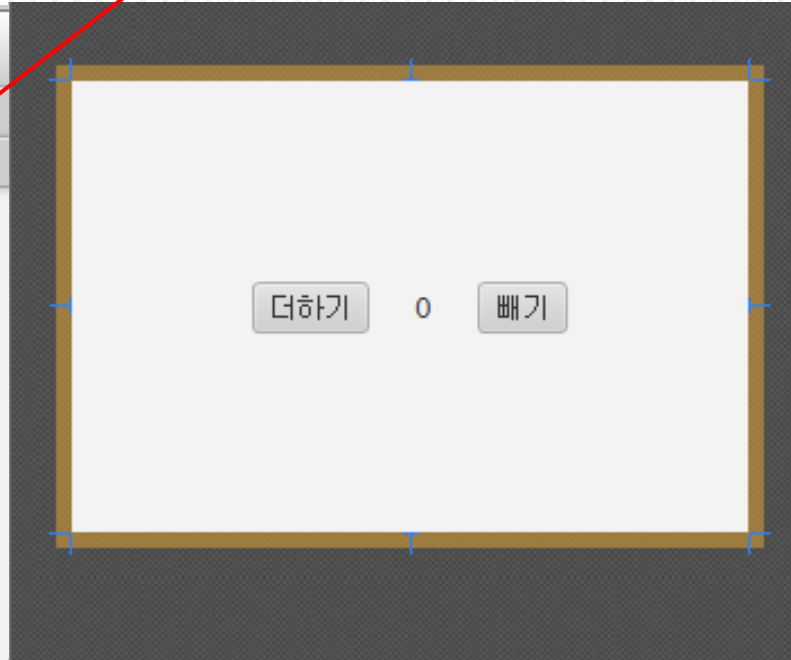
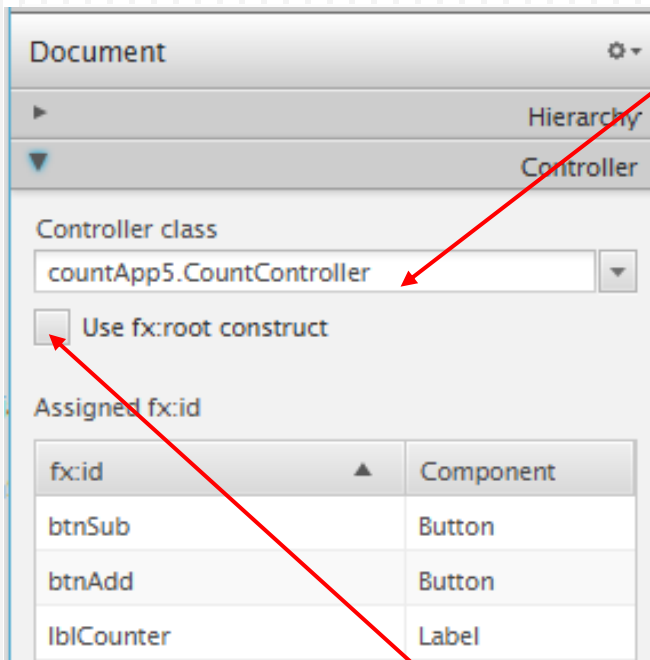
lbl =new Label("0");
btnAdd =new Button("더하기");
btnSub =new Button("빼기");
```



컨트롤러 입력 및 SceneBuilder

1. Controller 입력

countApp5.CountController



2. Use fx:root constructor 체크해제

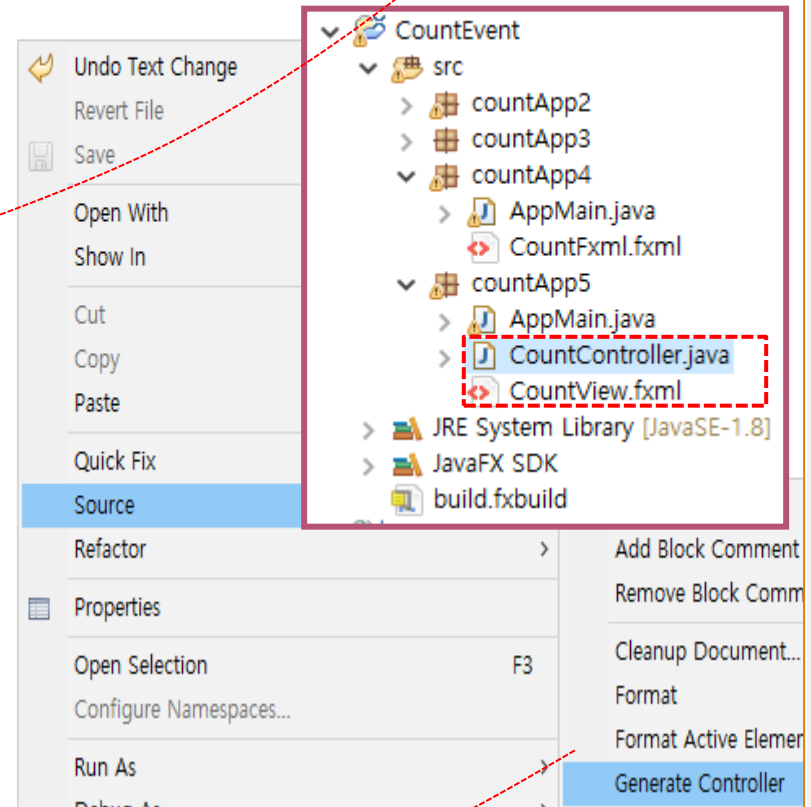
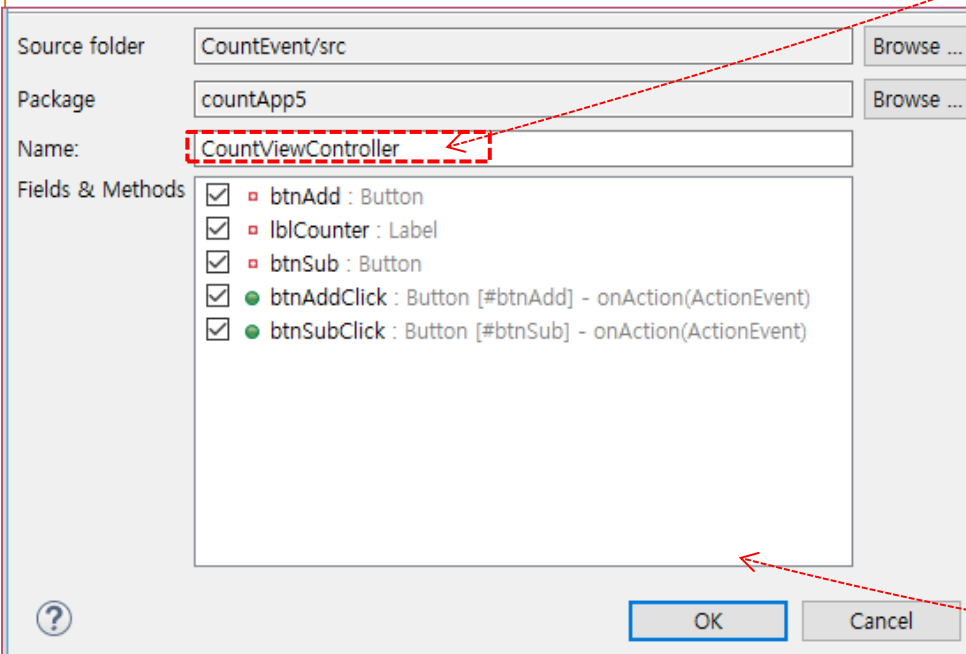
3. SceneBuilder save

Fxml 파일 확인과 Controller 골격 생성

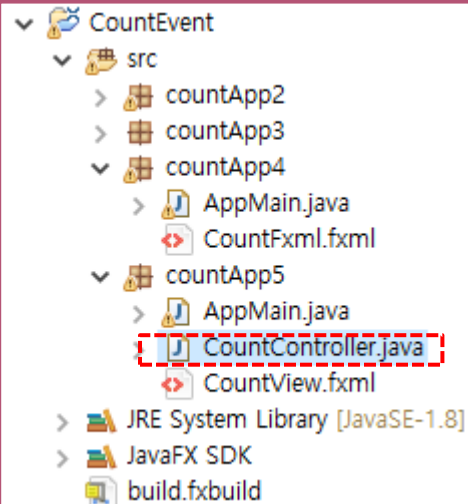
```
<AnchorPane xmlns="http://javafx.com/javafx/8" xmlns:fx="http://javafx.com/fxml/1" fx:controller="countApp5.CountController">
    <children>
        <HBox alignment="CENTER" prefHeight="200.0" prefWidth="300.0" spacing="20.0" AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0">
            <children>
                <Button fx:id="btnAdd" alignment="CENTER" contentDisplay="CENTER" mnemonicParsing="false" onAction="#btnAddClick" />
                <Label fx:id="lblCounter" alignment="CENTER" contentDisplay="CENTER" text="0" />
                <Button fx:id="btnSub" alignment="CENTER" contentDisplay="CENTER" mnemonicParsing="false" onAction="#btnSubClick" />
            </children>
        </HBox>
    </children>
</AnchorPane>
```

1. Code 화면바탕 오른쪽 클릭

2. 이름수정 일치확인 !!!



Controller 골격생성과 확인



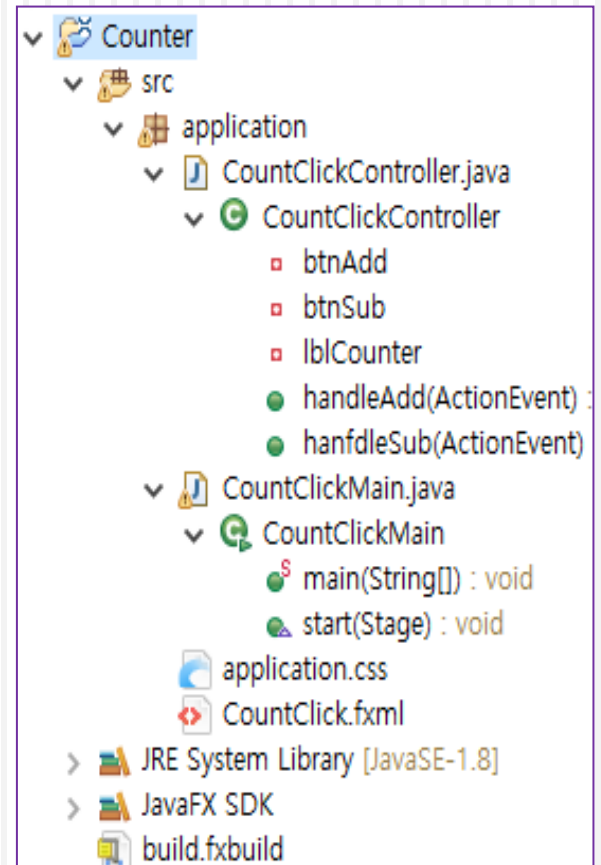
1. Controller 확인

```
public class CountController {  
    @FXML  
    private Button btnAdd;  
    @FXML  
    private Label lblCounter;  
    @FXML  
    private Button btnSub;  
  
    // Event Listener on Button[#btnAdd].onAction  
    @FXML  
    public void btnAddClick(ActionEvent event) {  
        // TODO Autogenerated  
    }  
    // Event Listener on Button[#btnSub].onAction  
    @FXML  
    public void btnSubClick(ActionEvent event) {  
        // TODO Autogenerated  
    }  
}
```

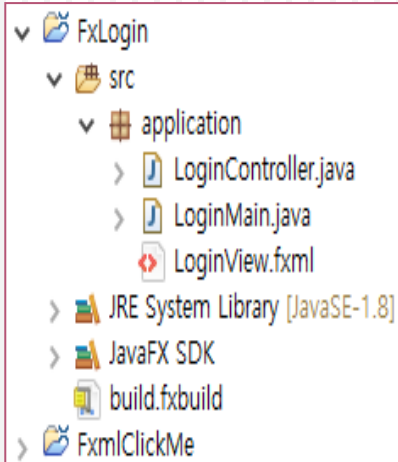
- 이벤트 소스(fxml)에서 가져온 자원 확인
- Controller 명칭 확인 - 3 부분 일치
- 컨트롤 개수, 명칭 확인
- 이벤트 개수, 명칭 확인

Controller 이벤트처리기 완성

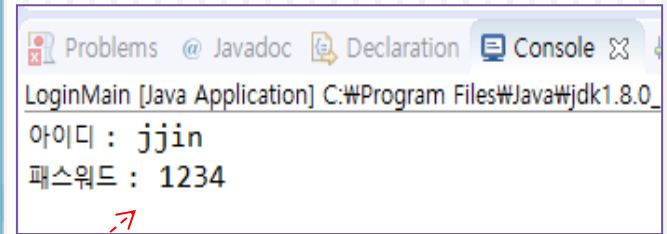
```
public class CountClickController {  
    @FXML  
    private Button btnAdd;  
    @FXML  
    private Button btnSub;  
    @FXML  
    private Label lblCounter;  
  
    int iCounter=0;  
    // Event Listener on Button[#btnAdd].onAction  
    @FXML  
    public void handleAdd(ActionEvent event) {  
        iCounter++;  
        lblCounter.setText(Integer.toString(iCounter));  
    }  
    // Event Listener on Button[#btnSub].onAction  
    @FXML  
    public void handleSub(ActionEvent event) {  
        iCounter--;  
        lblCounter.setText(Integer.toString(iCounter));  
    }  
}
```



Login 예제

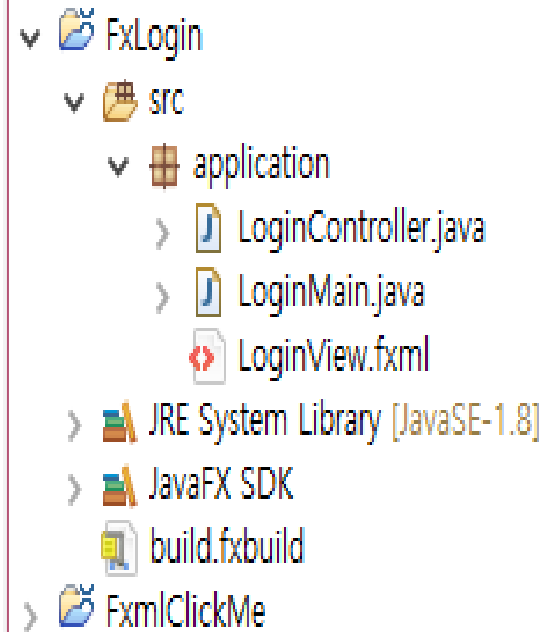
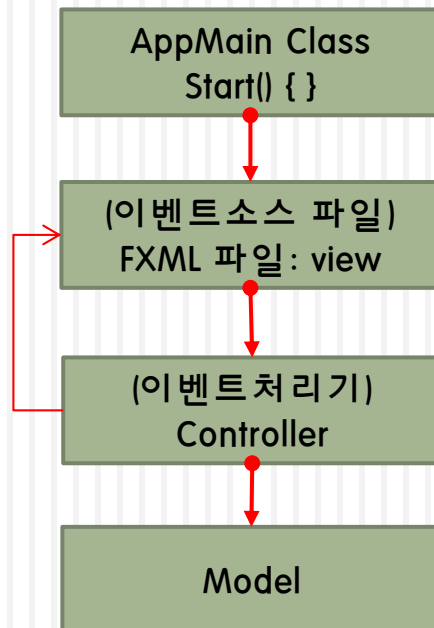


1. "FxLogin" JavaFx project 생성
2. Main.java → LoginMain 변경
3. LoginView.fxml 파일 생성 및 편집
4. Controller 생성
5. LoginMain.java, Controller 수정



작업 순서

- FXML 파일 생성
- Main 수정 (FXML 파일 로드)
- FXML 파일 편집
- Controller 생성
- Controller 이벤트 핸들러 작성
- 실행



Main 수정

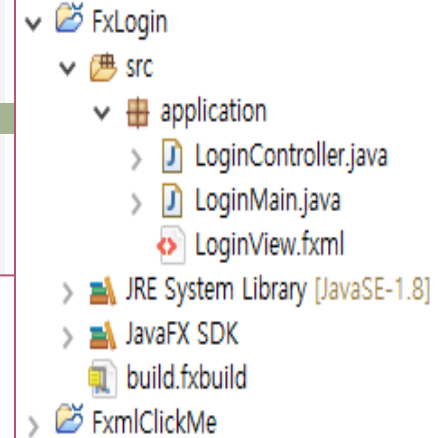
■ Main 수정

```
public class LoginMain extends Application {
    @Override
    public void start(Stage primaryStage) throws Exception {

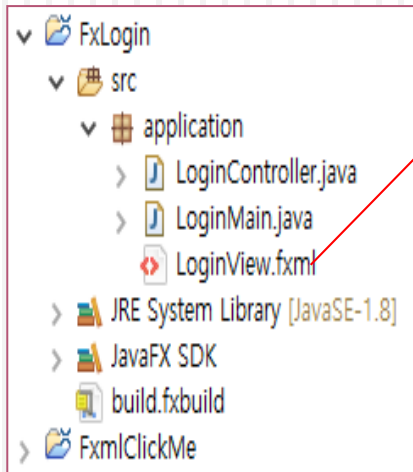
        Parent root = FXMLLoader.load(getClass().getResource("/application/LoginView.fxml"));
        Scene scene = new Scene(root);

        primaryStage.setTitle("Login App");
        primaryStage.setScene(scene);
        primaryStage.show();
    }

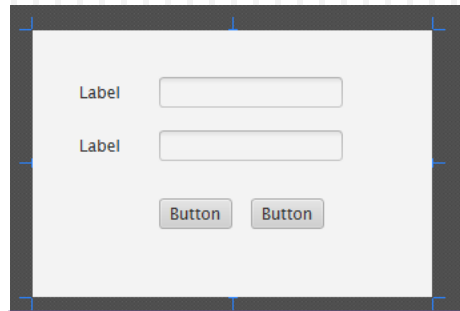
    public static void main(String[] args) {
        launch(args);
    }
}
```



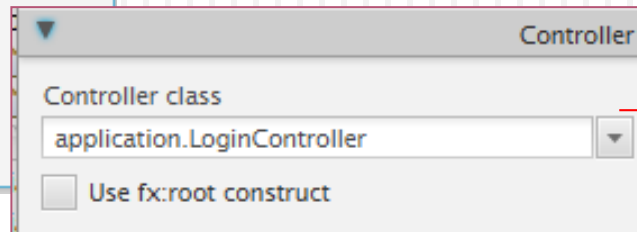
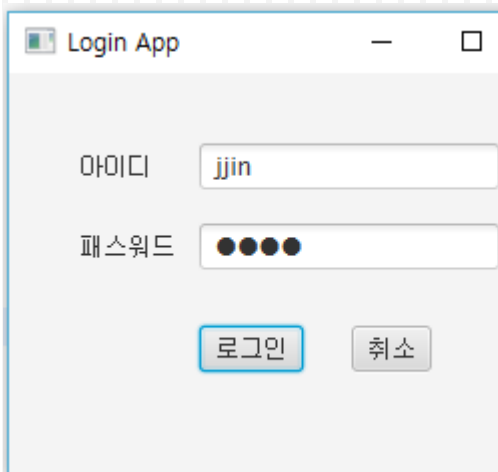
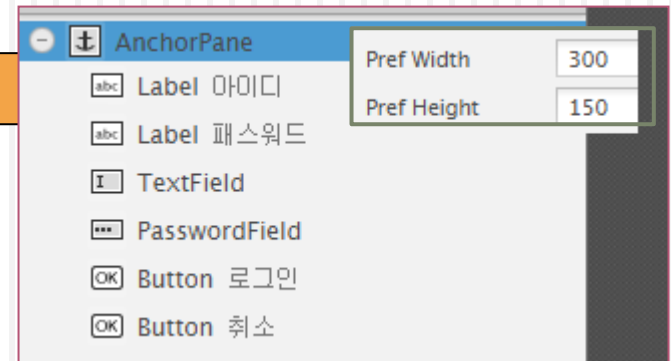
FXML 파일 편집



우클릭 > open with scenebuilder



```
<TextField fx:id="txtID" layout="column" />
<PasswordField fx:id="txtPWD" layout="column" />
<Button fx:id="btnLogin" layout="center" text="로그인" onAction="#clickLogin" />
<Button fx:id="btnCancel" layout="center" text="취소" onAction="#clickCancel" />
```



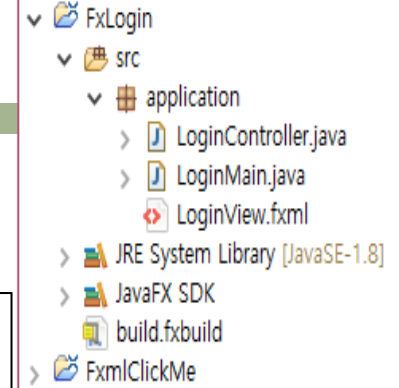
Save

Controller

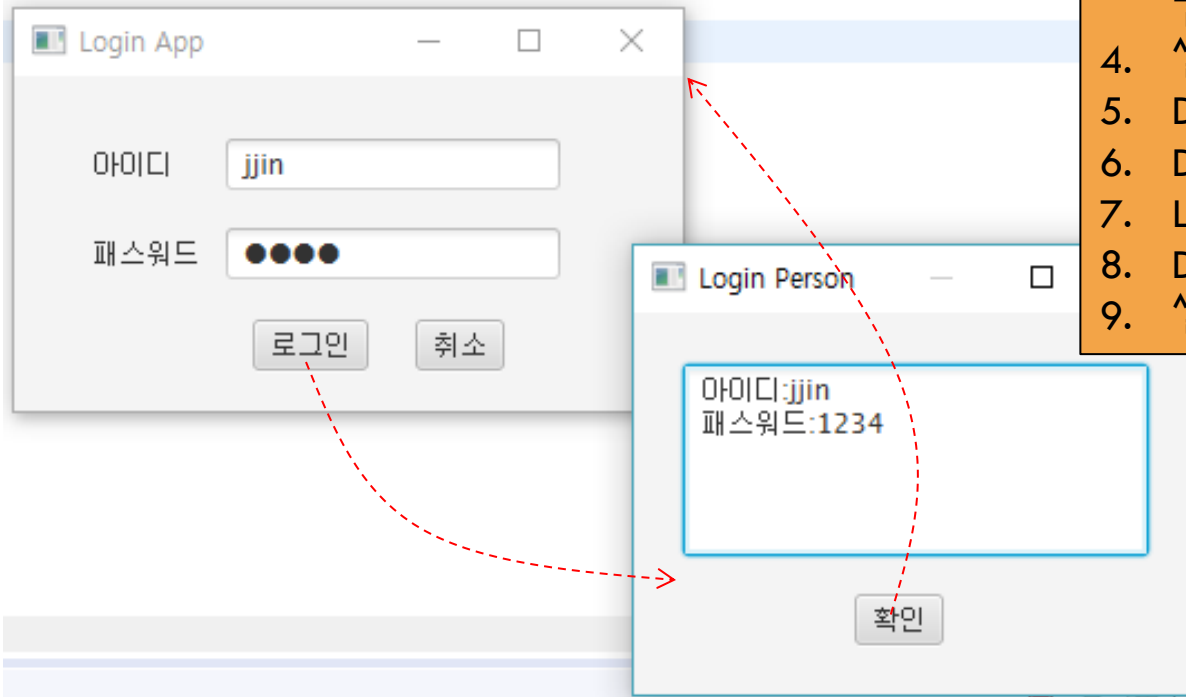
- Controller 생성, 확인, 이벤트 핸들러 작성
 - LoginView.fxml 우클릭 > source > generate Controller

```
public class LoginController {  
    @FXML  
    private TextField txtID;  
    @FXML  
    private PasswordField txtPWD;  
    @FXML  
    private Button btnLogin;  
    @FXML  
    private Button btnCancel;  
  
    // Event Listener on Button[#btnLogin].onAction  
    @FXML  
    public void clickLogin(ActionEvent event) {  
        System.out.println("아이디 : "+txtID.getText().toString());  
        System.out.println("패스워드 : "+txtPWD.getText().toString());  
    }  
    // Event Listener on Button[#btnCancel].onAction  
    @FXML  
    public void clickCancel(ActionEvent event) {  
        System.exit(0);  
    }  
}
```

- 실행



DiaLog 예 제 - 932p



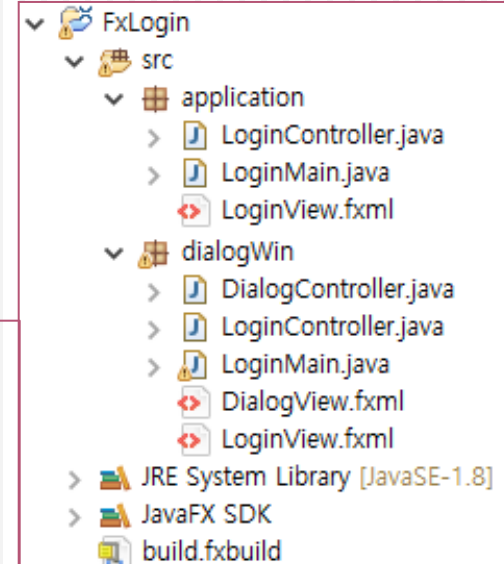
1. Src>"dialogWin" Package 생성
2. Application package 의 파일 복사
3. Controller, Main 파일의 package 경로명 변경
4. 실행
5. DialogView.fxml 파일 생성 및 편집
6. DialogController 생성
7. LoginMain.java, Controller 수정
8. DialogController 수정
9. 실행

Dialog box or window — 사용자와 프롬프트간 정보를 전달하기 위한 작은 window

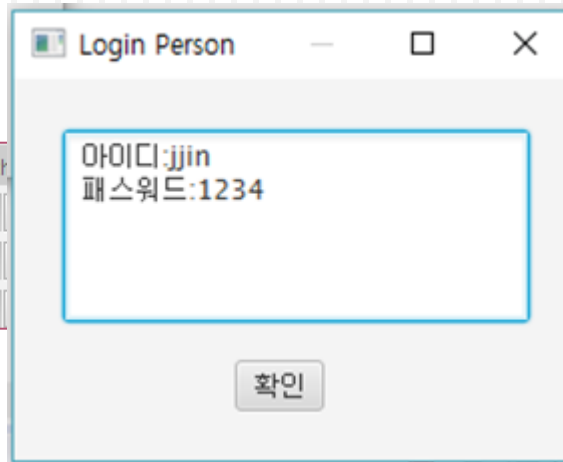
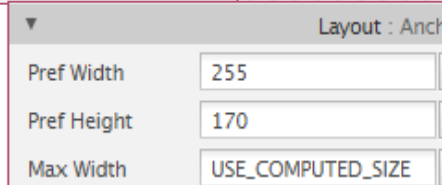
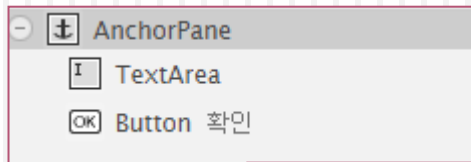
Modal : 다이얼로그 창을 닫기전까지 부모 창을 사용할 수 없음

Modeless : 다이얼로그 창을 닫지 않아도 부모 창을 계속 사용할 수 있음

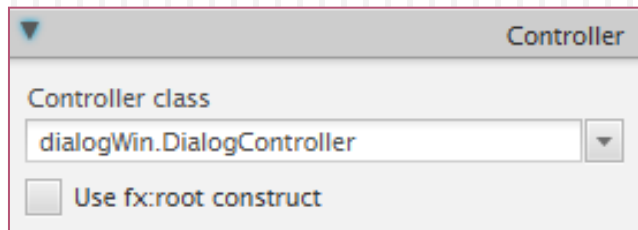
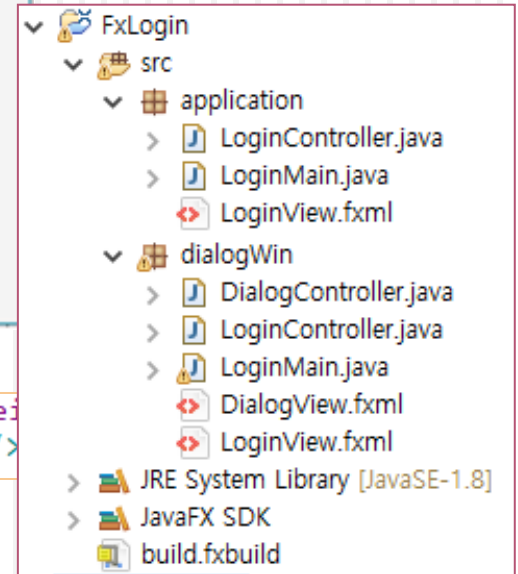
Dialog box도 윈도우 이므로 Stage로 취급된다.



DialogView.fxml 파일 생성 및 편집

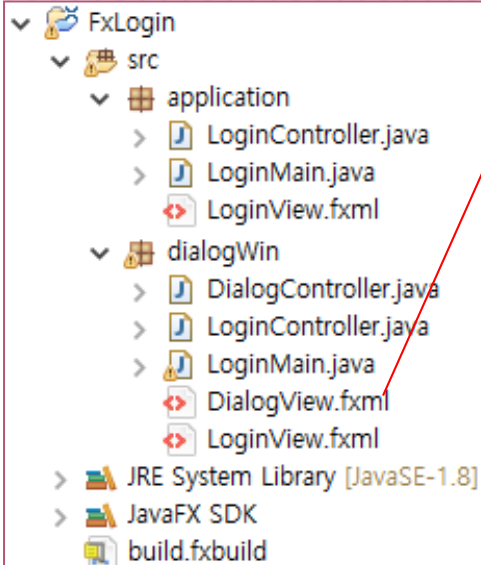


```
<TextArea fx:id="txtArea" layoutX="22.0" layoutY="23.0" prefHeight="170.0" />
<Button fx:id="btnConfirm" onAction="#hdlConfirm" text="확인" />
```



Save

DialogController 생성 및 편집



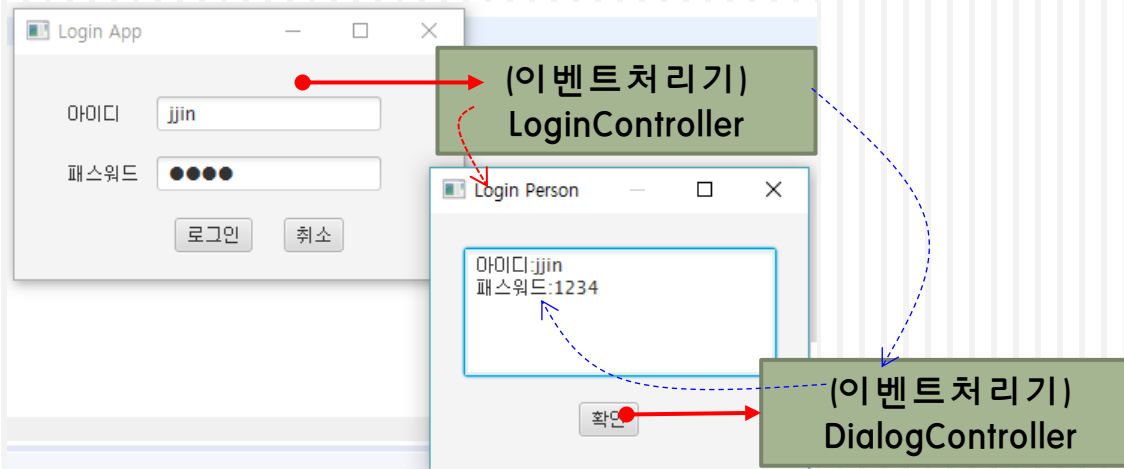
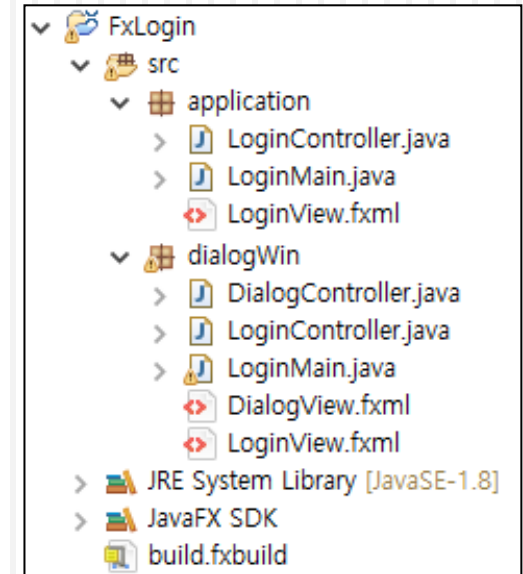
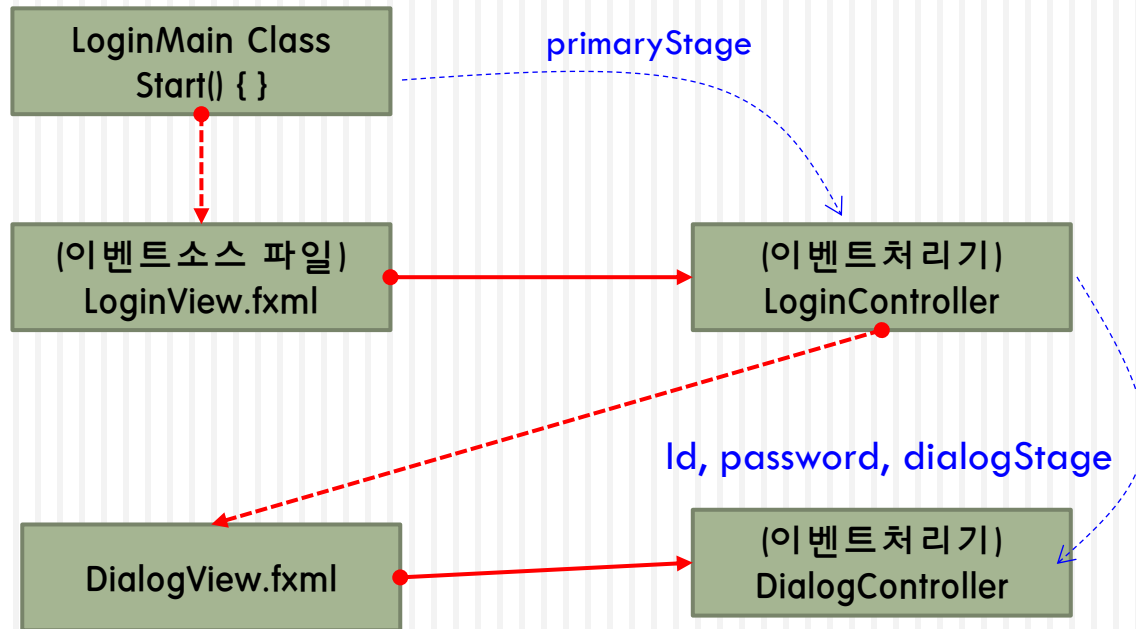
DialogView.fxml 우클릭 > source > generate Controller

```
import javafx.fxml.FXML;

public class DialogController {
    @FXML
    private TextArea txtArea;
    @FXML
    private Button btnConfirm;

    // Event Listener on Button[#btnConfirm].onAction
    @FXML
    public void hdlConfirm(ActionEvent event) {
        // TODO Autogenerated
    }
}
```

프로그램 구조



LoginMain의 변경

```
public class LoginMain extends Application {  
  
    @Override  
    public void start(Stage primaryStage) throws Exception {  
  
        FXMLLoader loader = new FXMLLoader();  
        loader.setLocation(getClass().getResource("/dialogWin/LoginView.fxml"));  
        Parent root = loader.load();  
        //Parent root = FXMLLoader.load(getClass().getResource("/application/L  
  
        Scene scene = new Scene(root);  
  
        primaryStage.setTitle("Login App");  
        primaryStage.setScene(scene);  
  
        LoginController controller = loader.getController();  
        controller.setPrimaryStage(primaryStage);  
  
        primaryStage.show();  
    }  
  
    public static void main(String[] args) {  
        launch(args);  
    }  
}
```

(이벤트소스 파일)
LoginView.fxml

(이벤트처리기)
LoginController

LoginController 변경

LoginMain.start()

//부모 윈도우 스테이지 전달 받기

```
private Stage primaryStage;
```

```
public void setPrimaryStage(Stage primaryStage) {  
    this.primaryStage = primaryStage;  
}
```

@FXML // #btnLogin 버튼의 이벤트처리

```
public void clickLogin(ActionEvent event) throws Exception {
```

```
    System.out.println("아이디 : "+txtID.getText().toString());
```

```
    System.out.println("패스워드 : "+txtPWD.getText().toString());
```

//FXML에 의한 Pane 생성

```
FXMLLoader loader = new FXMLLoader();
```

```
loader.setLocation(getClass().getResource("/dialogWin/DialogView.fxml"));
```

```
AnchorPane page = (AnchorPane) loader.load();
```

//Scene 생성

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

//dialog Stage 생성

```
Stage dialogStage = new Stage();
```

```
dialogStage.setTitle("Login Person");
```

```
dialogStage.initModality(Modality.WINDOW_MODAL);
```

```
dialogStage.initOwner(primaryStage);
```

```
dialogStage.setScene(scene);
```

//다이얼로그 controller에 id, pwd 전달

```
DialogController controller = loader.getController();
```

```
controller.setDialogStage(dialogStage);
```

```
controller.setPerson(txtID.getText(), txtPWD.getText());
```

```
dialogStage.showAndWait();
```

```
}
```

```
LoginController controller = loader.getController();  
controller.setPrimaryStage(primaryStage);
```

나머지 부분은 그대로

(이벤트처리기)
LoginController

id, password, dialogStage

DialogView.fxml

(이벤트처리기)
DialogController

DialogController 수정

```
public class DialogController {
```

```
@FXML
```

```
private TextArea txtArea;
```

```
@FXML
```

```
private Button btnConfirm;
```

```
private Stage dialogStage;
```

```
public void setDialogStage(Stage dialogStage) {  
    this.dialogStage = dialogStage;  
}
```

```
public void setPerson(String id, String pwd) {  
    txtArea.setText("아이디:" + id + "\n");  
    txtArea.setText(txtArea.getText() + "패스워드:" + pwd);  
}
```

```
// Event Listener on Button[#btnConfirm].onAction
```

```
@FXML
```

```
public void hdlConfirm(ActionEvent event) {  
    dialogStage.close();  
}
```

```
}
```

//다이얼로그 controller에 id, pwd 전달

```
LoginDialController controller = loader.getController();  
controller.setDialogStage(dialogStage);  
controller.setPerson(txtID.getText(), txtPWD.getText());
```

실행