JavaFX I

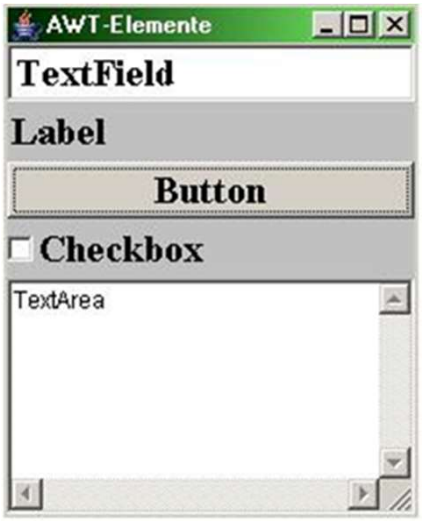
Shin-Jie Lee (李信杰)
Associate Professor
Computer and Network Center
Department of Computer Science and Information Engineering
National Cheng Kung University

GUI

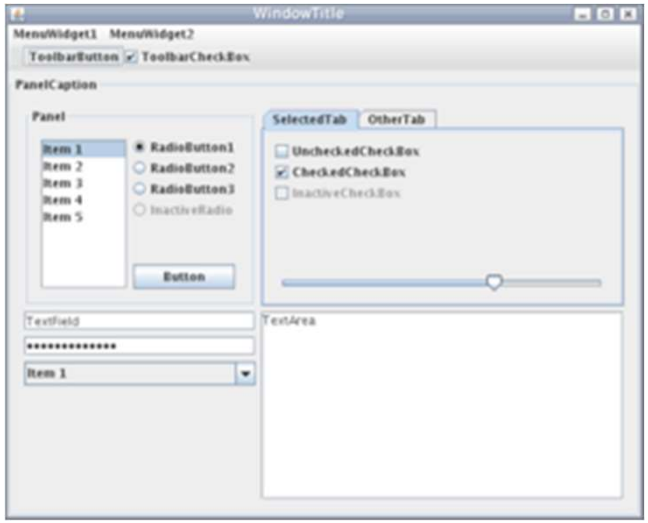
Graphical User Interface

Java GUI Tools

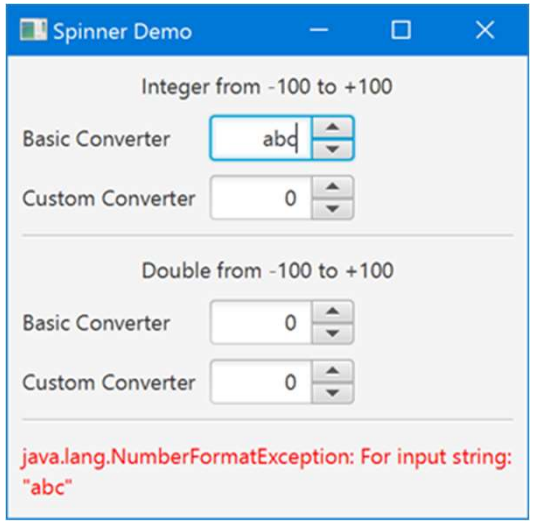
AWT



Swing



FX



Removed from JDK 11

JavaFX Feature 1

Cross-Platform

Windows, macOS, Linux

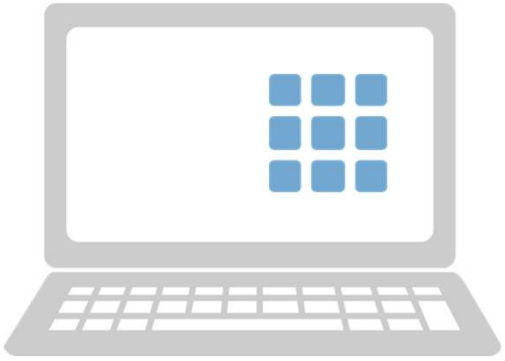
WebView

iOS, Android, Windows Mobile

"write once, run anywhere"

JavaFX Feature 2

Native Packaging



Distribute as a native app

No dependencies on Java version

Minimized app size

Why JavaFX was removed from JDK11?

Modular System of JDK

more flexibility for contributors to
engage in the open source community

Cannot Compete with HTML5

HTML5 is a larger web standard
which cannot be easily replaced

JavaFX removed from JDK

=

Manual Installation

1.
Installation

Download JavaFX SDK

- https://gluonhq.com/products/javafx/
- Long Term Support (LTS) ☜
- Latest Release
-

Product	Public version	LTS version	Platform	Download
JavaFX Windows SDK	11.0.2	11.0.11 More info	Windows	Download [SHA256]
JavaFX Windows jmods	11.0.2	11.0.11 More info	Windows	Download [SHA256]
JavaFX Mac OS X SDK	11.0.2	11.0.11 More info	Mac	Download [SHA256]
JavaFX Mac OS X jmods	11.0.2	11.0.11 More info	Mac	Download [SHA256]
JavaFX Linux SDK	11.0.2	11.0.11 More info	Linux	Download [SHA256]
JavaFX Linux jmods	11.0.2	11.0.11 More info	Linux	Download [SHA256]
JavaFX armv6hf SDK	11.0.2	11.0.11 More info	Embedded armv6hf	Download [SHA256]
JavaFX Documentation	11.0.2	11.0.11 More info	Javadoc	Download [SHA256]

Download JavaFX SDK

Extract the downloaded file, you will find three folders：

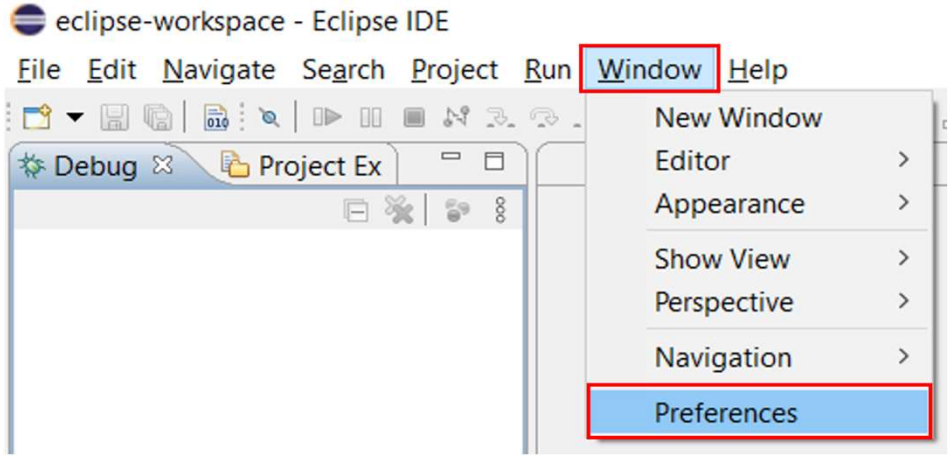
- bin/
- legal/
- lib/

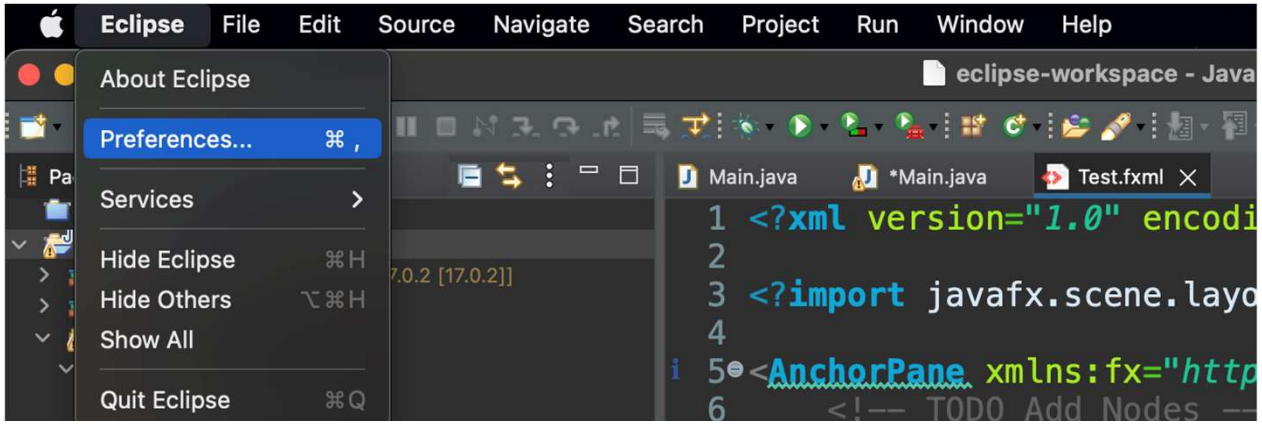
Install JavaFX

- https://openjfx.io/openjfx-docs/#install-javafx

Install JavaFX in Eclipse IDE

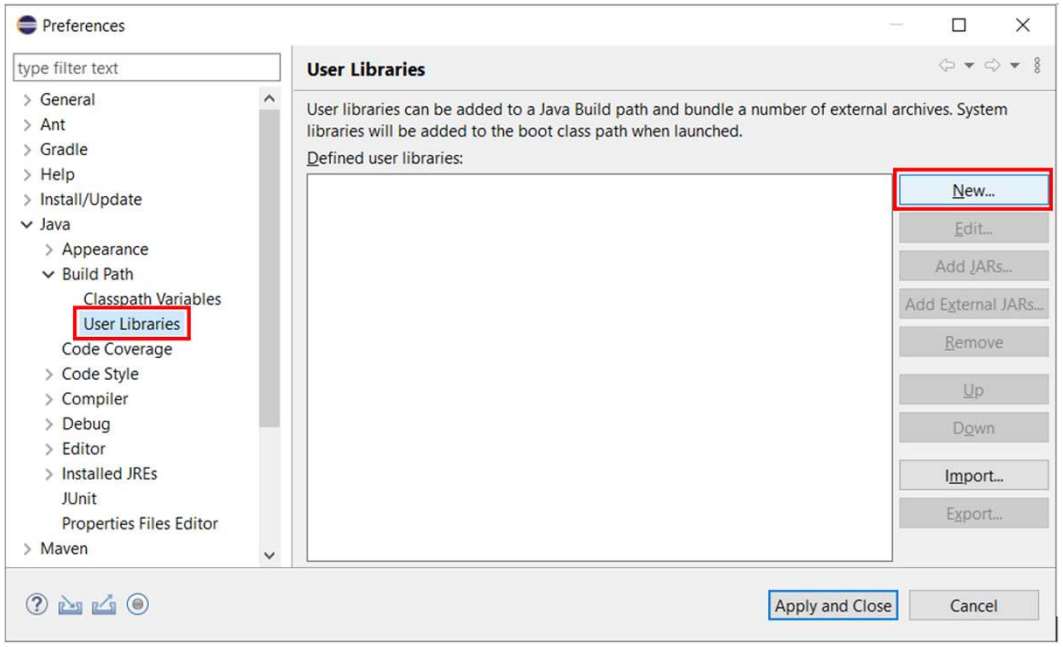
- Window → Preferences





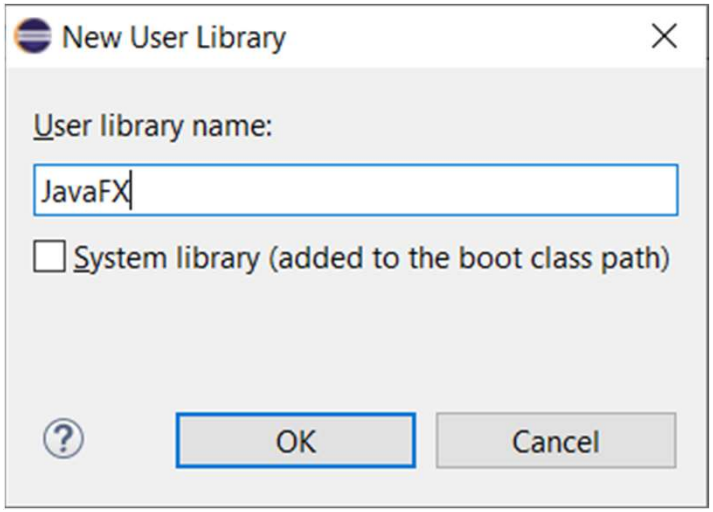
Install JavaFX in Eclipse IDE

- Create New User Library
  - Java → Build Path → User Libraries → New...



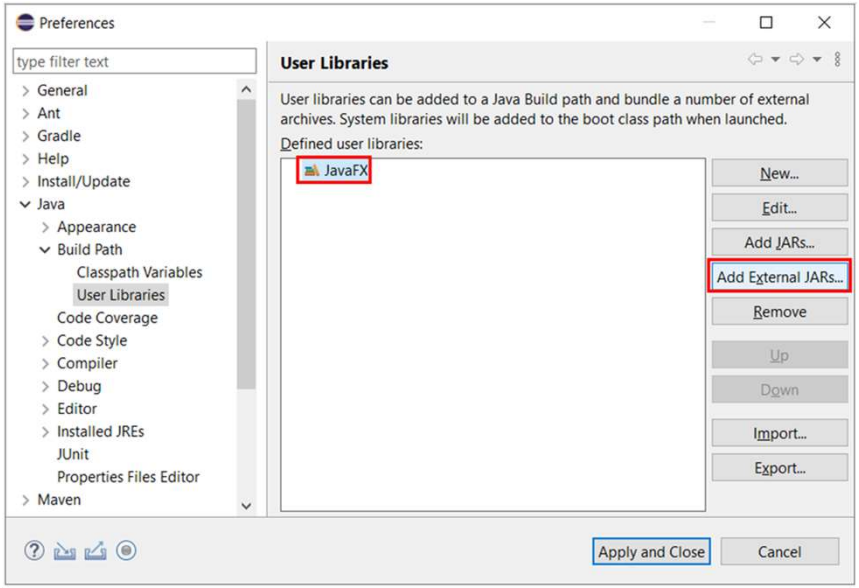
Install JavaFX in Eclipse IDE

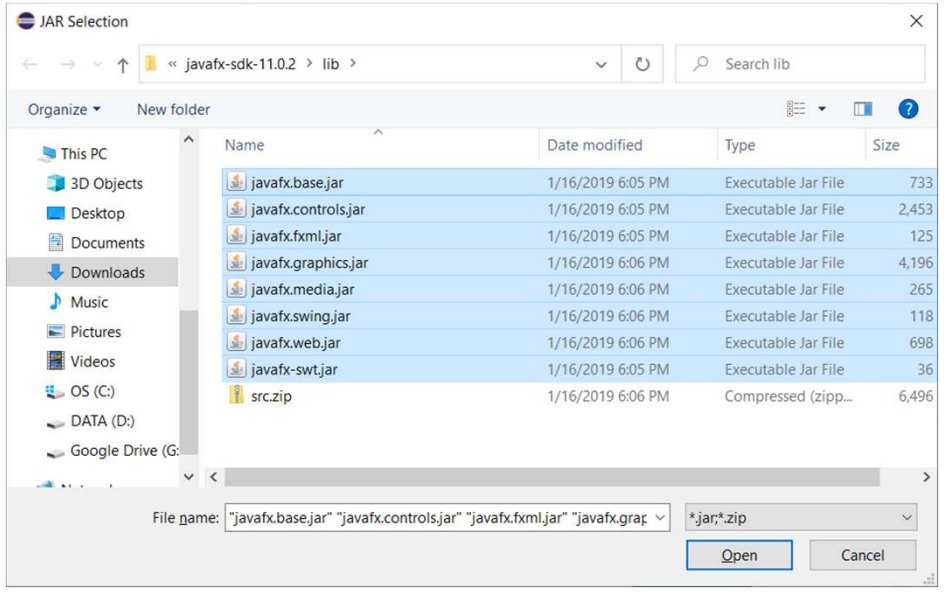
- Enter User Library Name (“JavaFX” , "javafx-11.0.2” , …)



Install JavaFX in Eclipse IDE

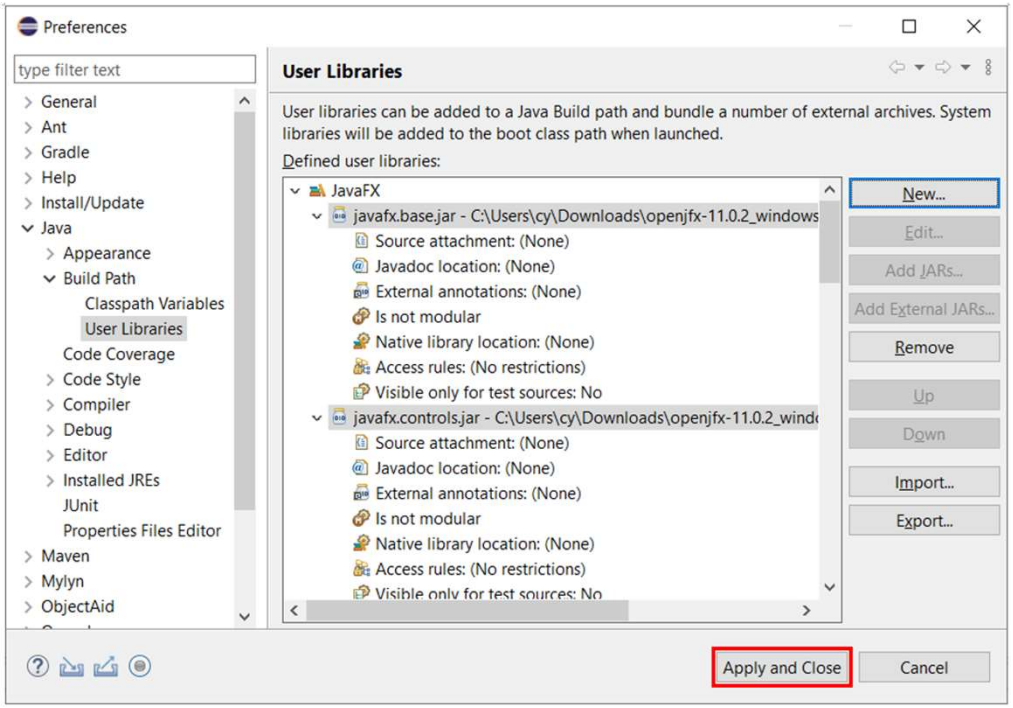
1. Add External JARs…
2. Select All *.jar in lib/





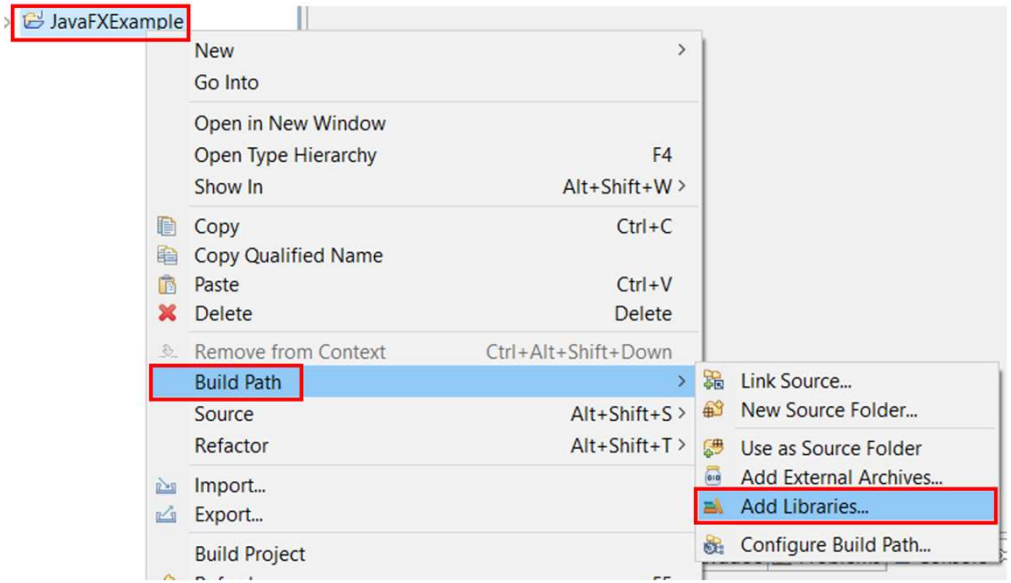
Install JavaFX in Eclipse IDE

- Apply and Close



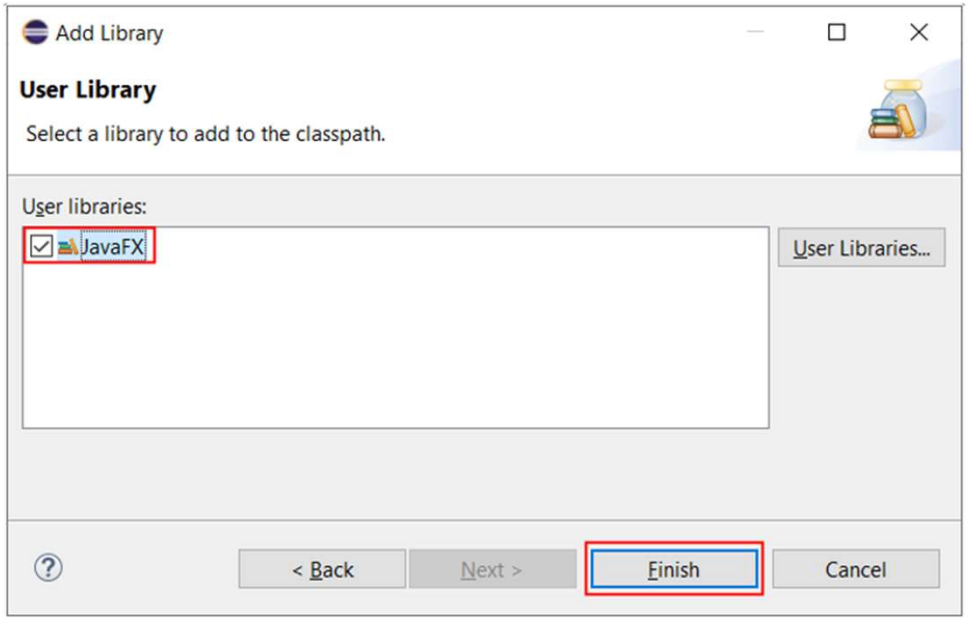
Install JavaFX in Eclipse IDE

1. Create a New Java Project
2. Right Click the Project
3. Bulid Path → Add Libraries...



Install JavaFX in Eclipse IDE

1. Select “User Library” → Next
2. Select JavaFX User Library → Finish



Try it !

```java
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;

public class HelloJavaFX extends Application {

    @Override
    public void start(Stage stage) {
        Label label = new Label("Hello, JavaFX");
        Scene scene = new Scene(new StackPane(label), 300, 200);
        stage.setScene(scene);
        stage.show();
    }

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

Oops...

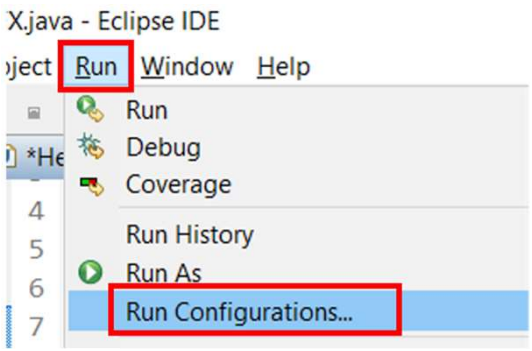
Warning: If you now run the project it will compile but you will get this error:

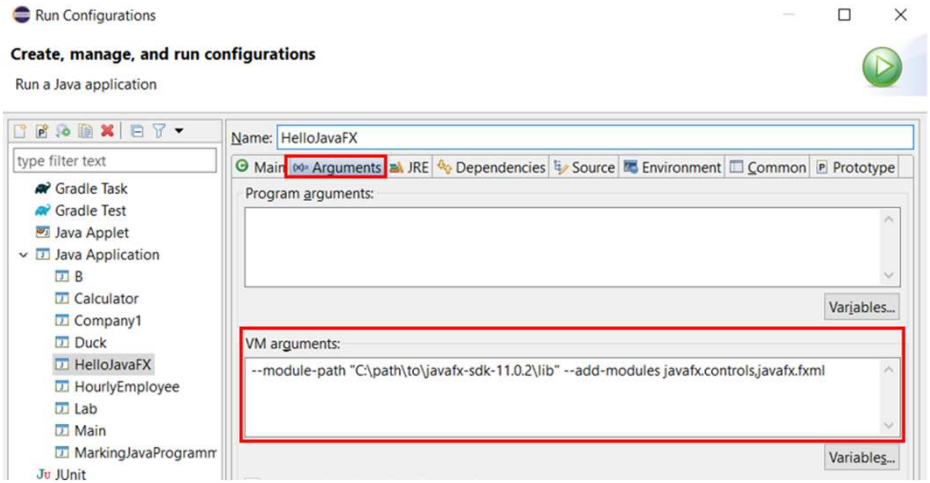
```
Error: JavaFX runtime components are missing, and are required to run this application
```

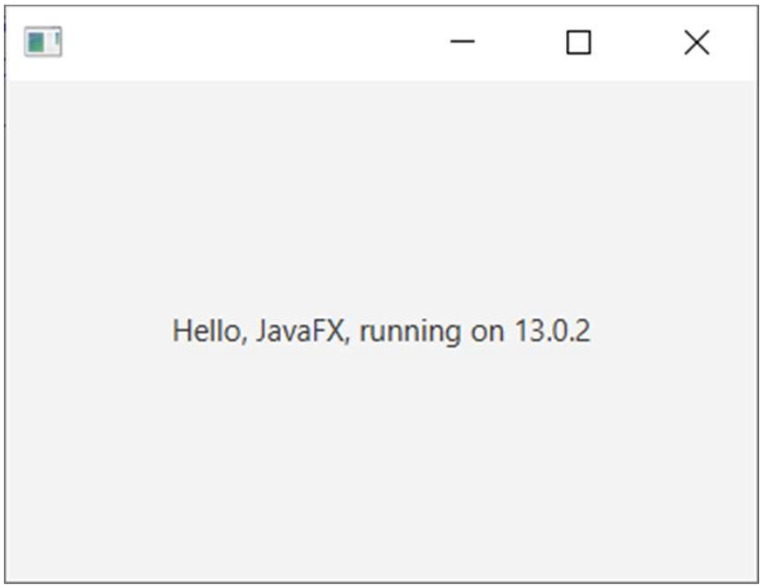
This error is shown since the Java 15 launcher checks if the main class extends `javafx.application.Application`. If that is the case, it is required to have the `javafx.graphics` module on the module-path.

Add VM arguments

1. Run → Run Configurations…
2. Arguments → VM arguments
   - Linux/Mac: `--module-path /path/to/javafx-sdk-15.0.1/lib --add-modules javafx.controls,javafx.fxml`
   - Windows: `--module-path "\path\to\javafx-sdk-15.0.1\lib" --add-modules javafx.controls,javafx.fxml`



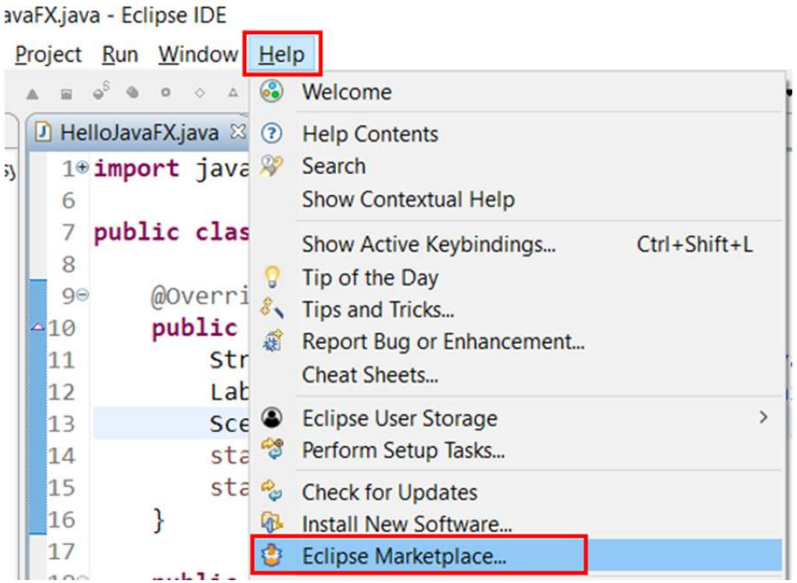



Hello, JavaFX, running on 13.0.2

Install e(fx)clipse in Eclipse IDE

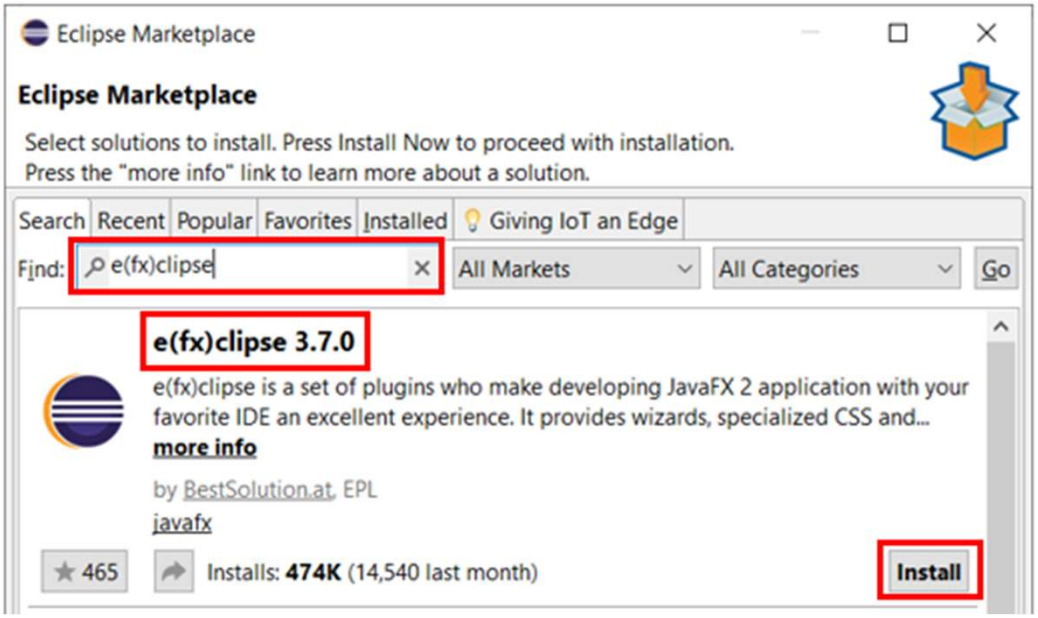
e(fx)clipse is an extension that provides JavaFX tooling for the Eclipse IDE.

- Help → Eclipse Marketplace...



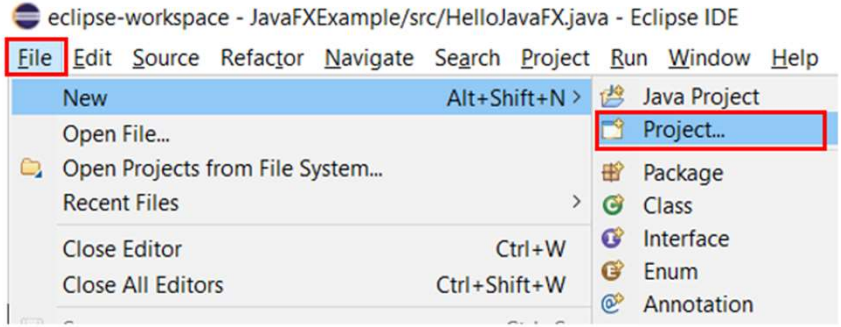
Install e(fx)clipse in Eclipse IDE

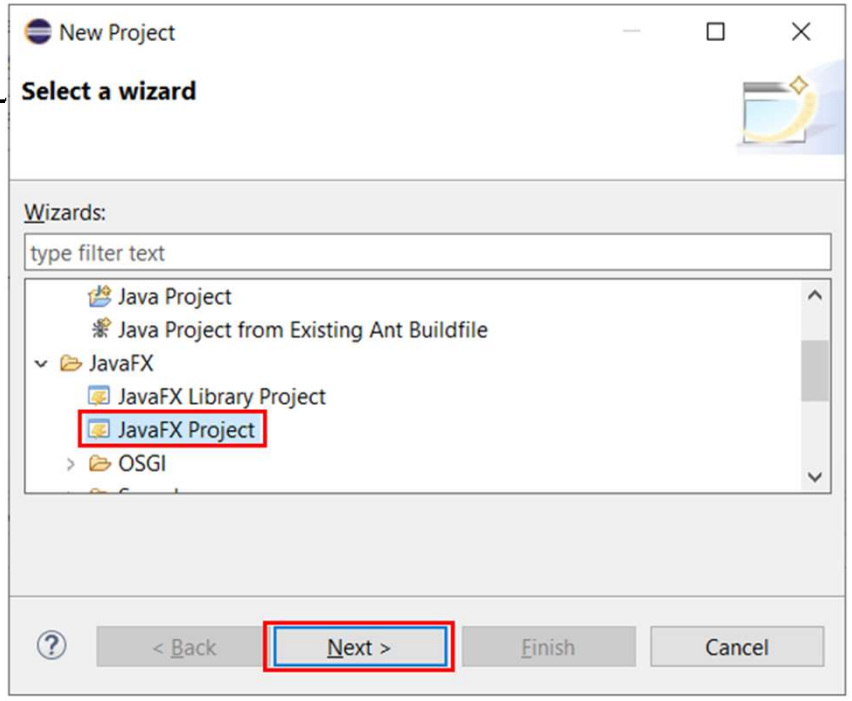
- Find "e(fx)clipse" → Install



Install e(fx)clipse in Eclipse IDE

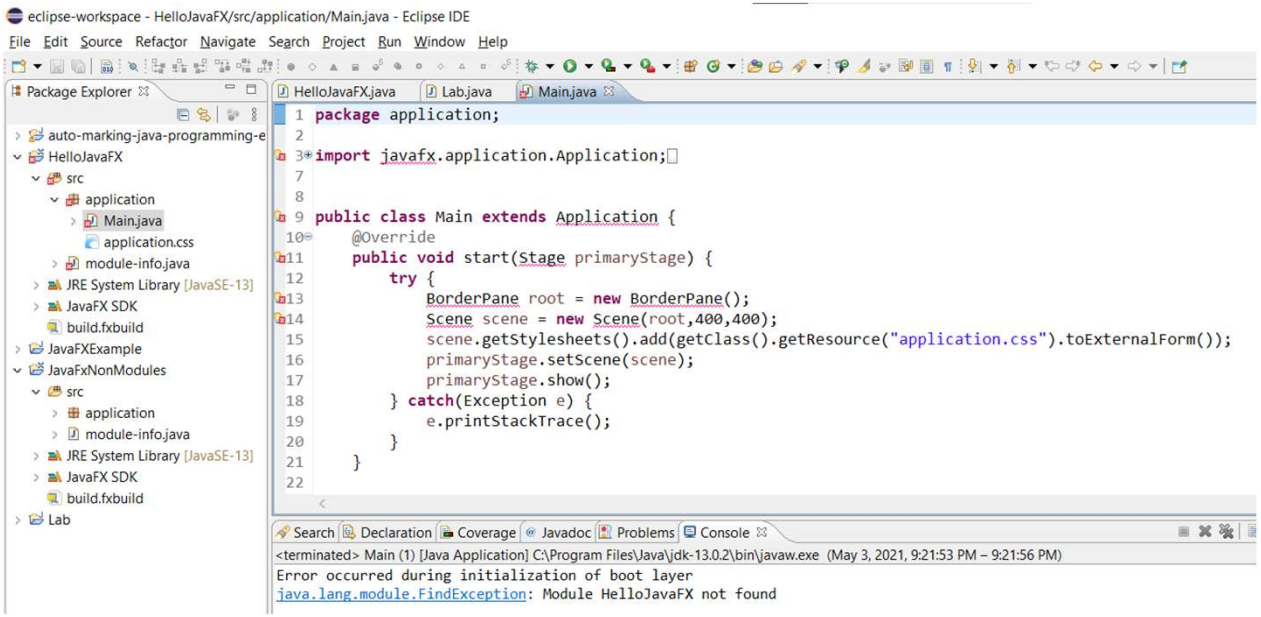
- File → Project
- JavaFX Project → Next





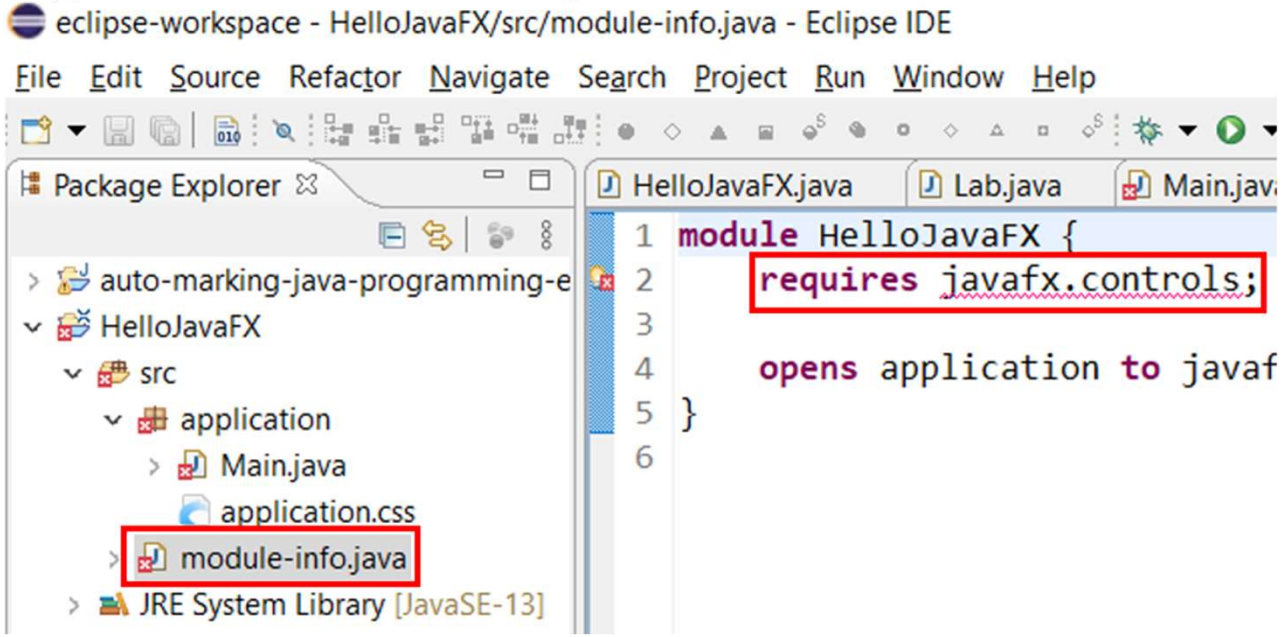
Oops...

- 「Error occurred during initialization of boot layer java.lang.module.FindException: Module HelloJavaFX not found」



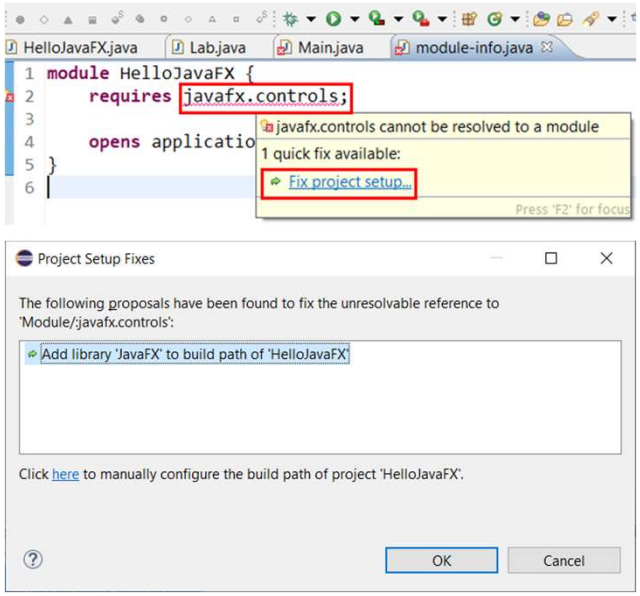
Solution 1: Quick Fixes in Eclipse

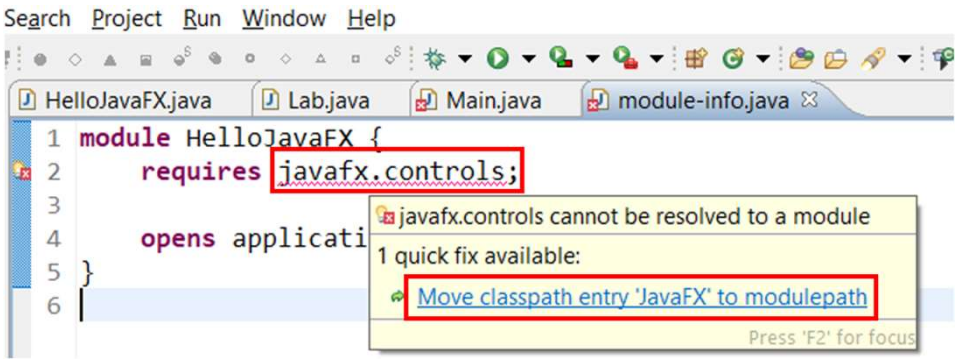
- Open module-info.java



Solution 1: Quick Fixes in Eclipse

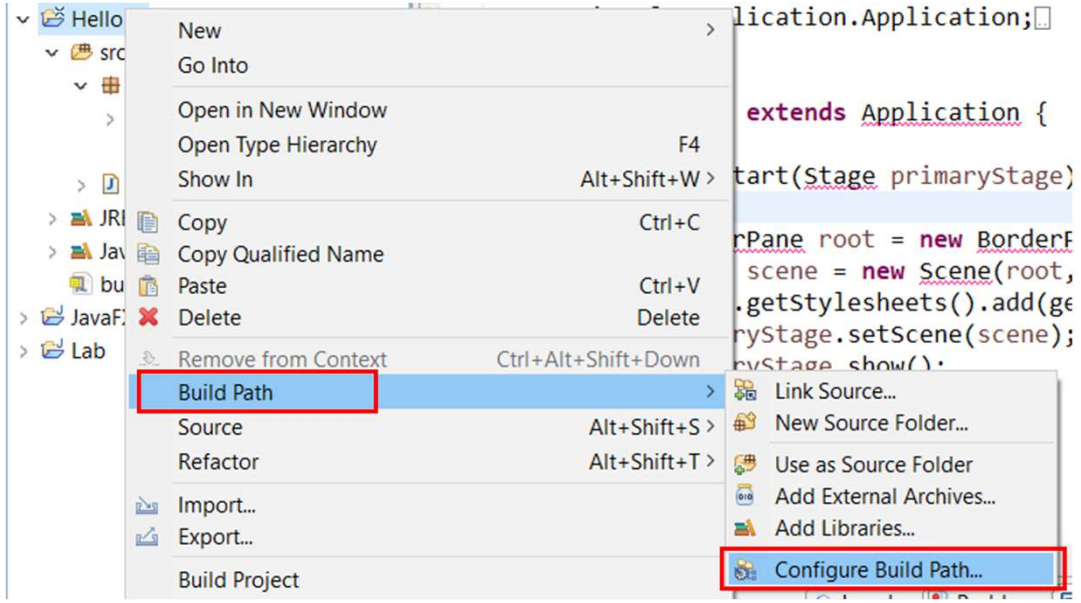
a. Fix project setup
b. Move classpath entry 'JavaFX' to modulepath：





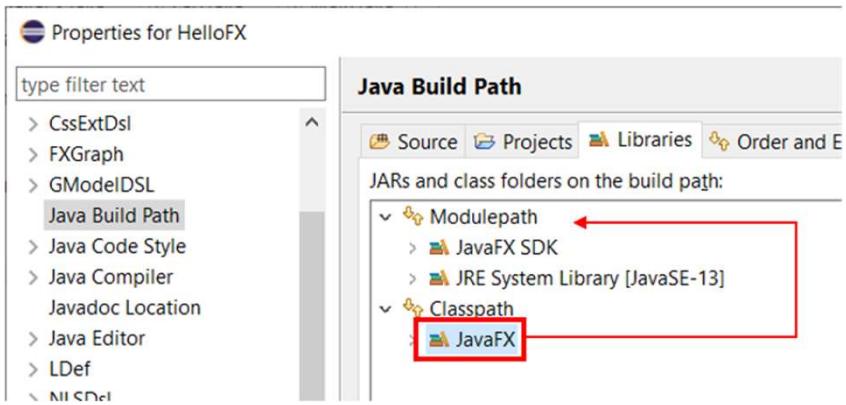
Solution 2: Manually Fix Ref Problems

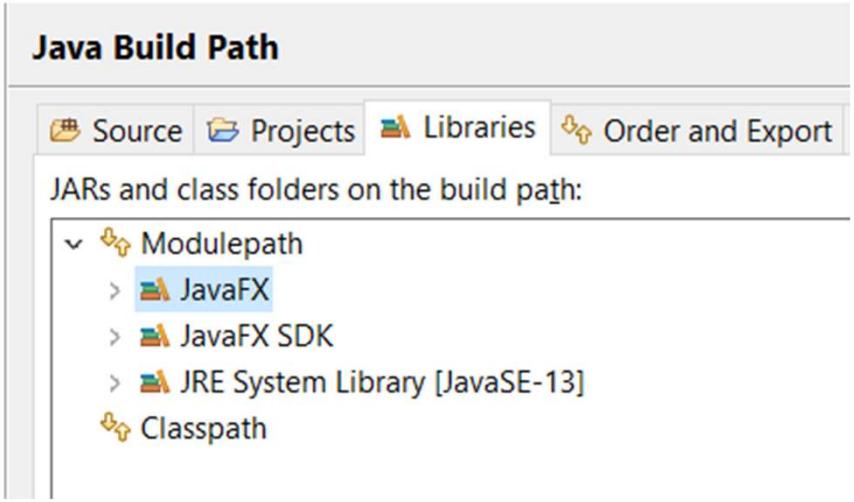
1. Add JavaFX User Library
2. Right Click Project → Build Path → Configure Build Path：



Solution 2: Manually Fix Ref Problems

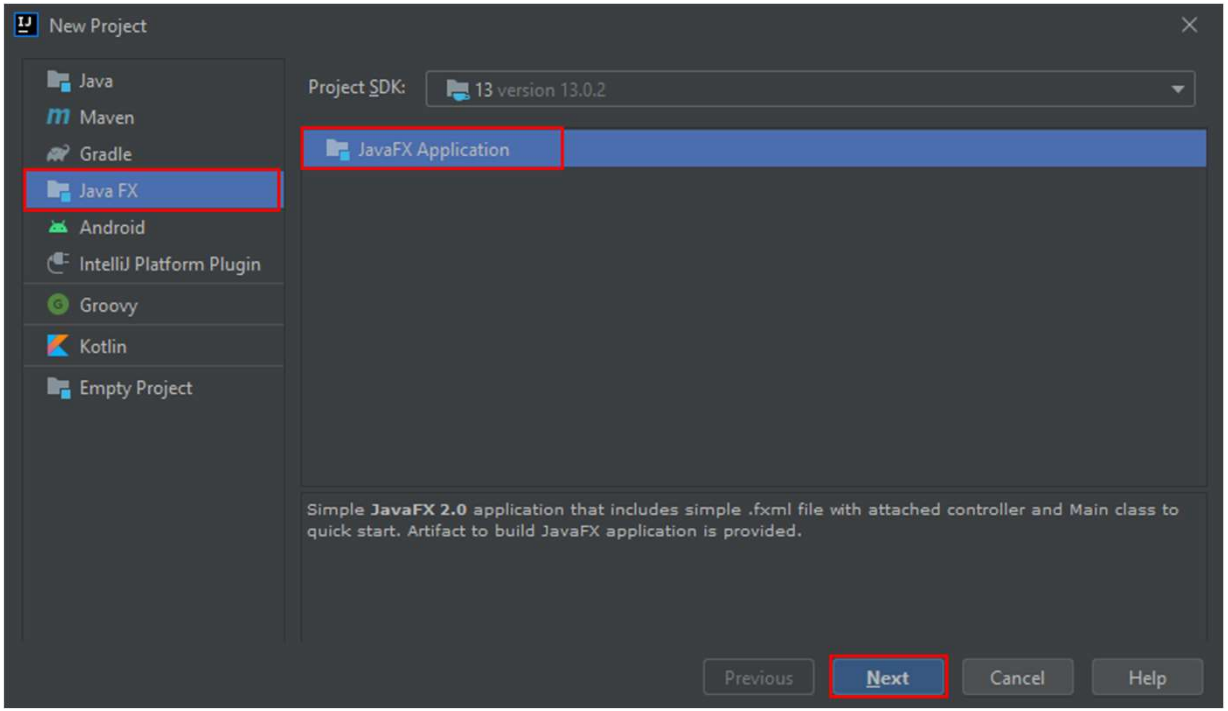
3. Move JavaFX User From classpath to module path：





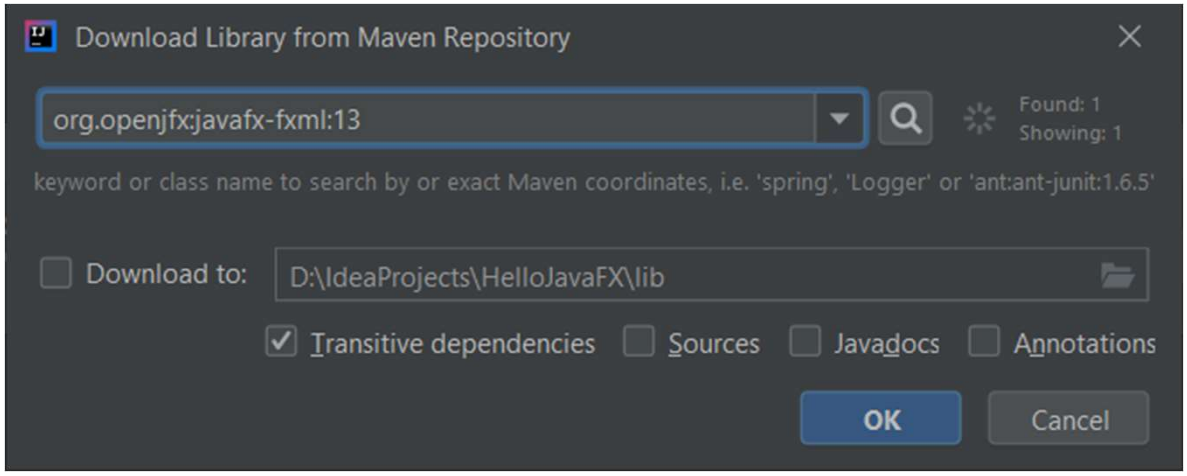
Install JavaFX in IntelliJ IDEA

1. New Project → Java FX → JavaFX Application → Next：



Install JavaFX in IntelliJ IDEA

- If this is your first time creating a JavaFX Application, IntelliJ IDEA should "Download Library from Maven Repository" automatically



Install JavaFX in IntelliJ IDEA

- You will see Intellij IDEA create sample code for you in Main.java

```
package sample;

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

public class Main extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception{
        Parent root = FXMLLoader.load(getClass().getResource("sample.fxml"));
        primaryStage.setTitle("Hello World");
        primaryStage.setScene(new Scene(root, 300, 275));
        primaryStage.show();
    }

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

Oops...

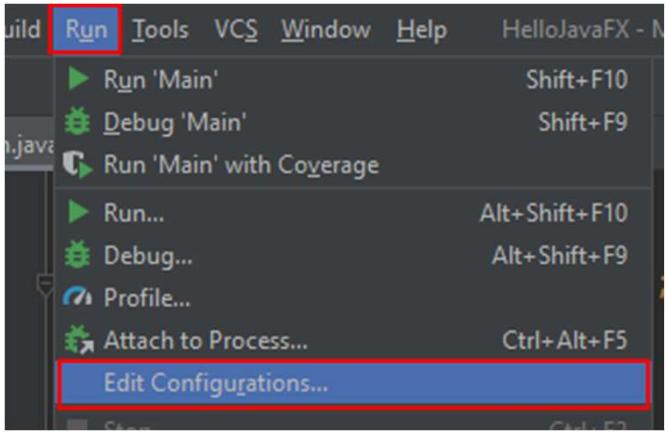
Warning: If you now run the project it will compile but you will get this error:

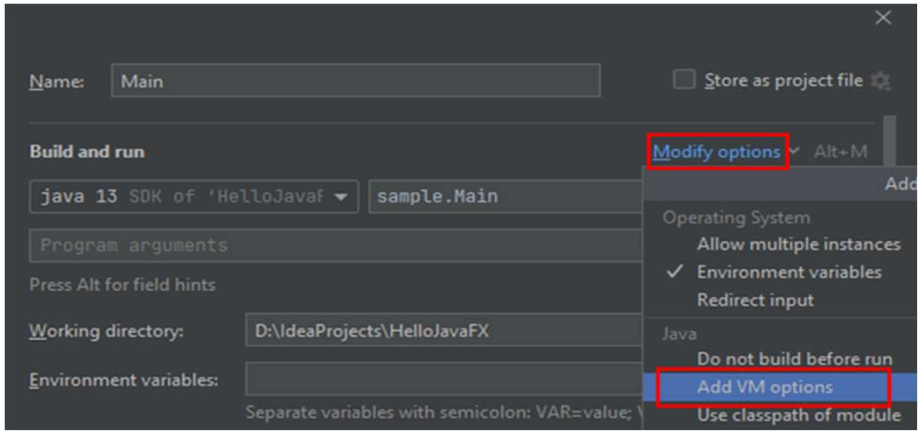
```
Error: JavaFX runtime components are missing, and are required to run this application
```

This error is shown since the Java 15 launcher checks if the main class extends `javafx.application.Application`. If that is the case, it is required to have the `javafx.graphics` module on the module-path.

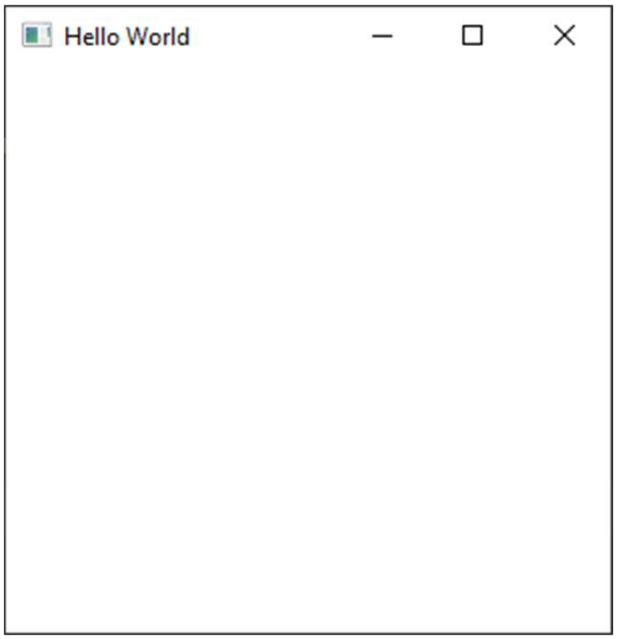
Add VM arguments

1. Run → Run Configurations…
2. Modify Options → VM arguments
   - Linux/Mac: `--module-path /path/to/javafx-sdk-15.0.1/lib --add-modules javafx.controls,javafx.fxml`
   - Windows: `--module-path "\path\to\javafx-sdk-15.0.1\lib" --add-modules javafx.controls,javafx.fxml`





Install JavaFX in IntelliJ IDEA



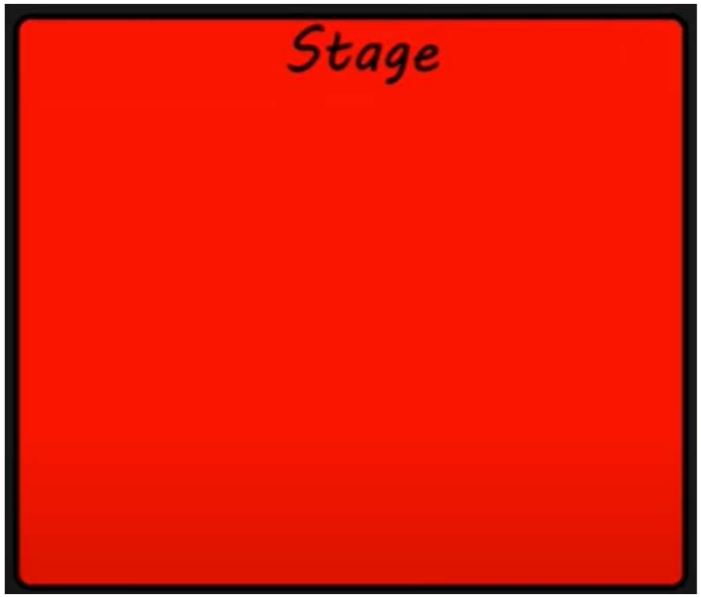
2.
Stage

Stage

Top-level container to hold the FX GUI applications

The Window of your application

Similar to JFrame

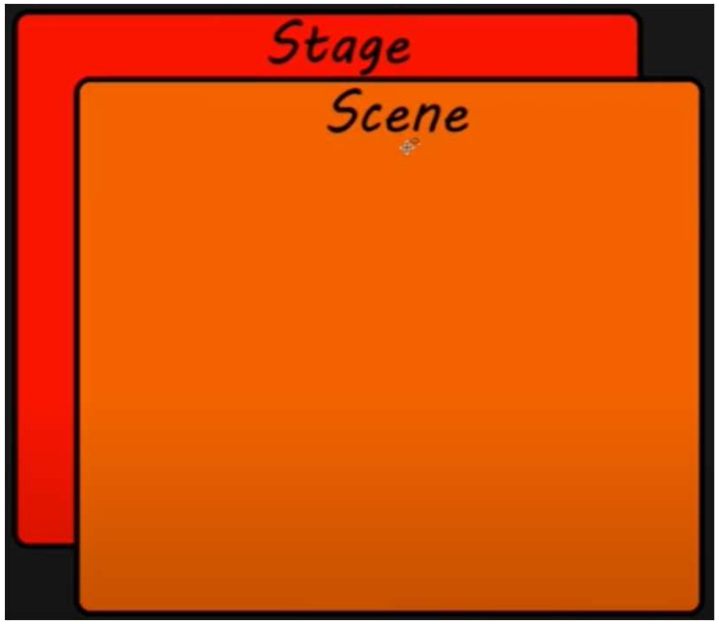


Scene

Add a scene to the “stage”

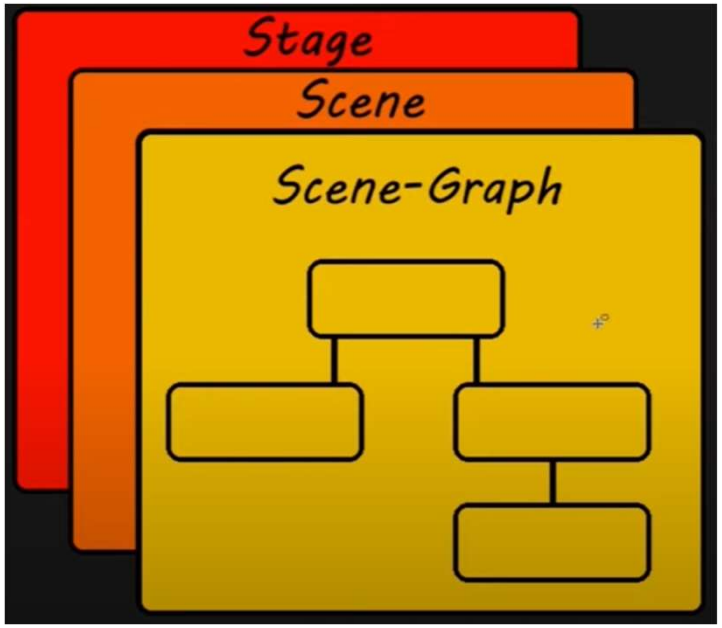
A drawing surface for graphical content

Similar to JPanel



Scene-Graph

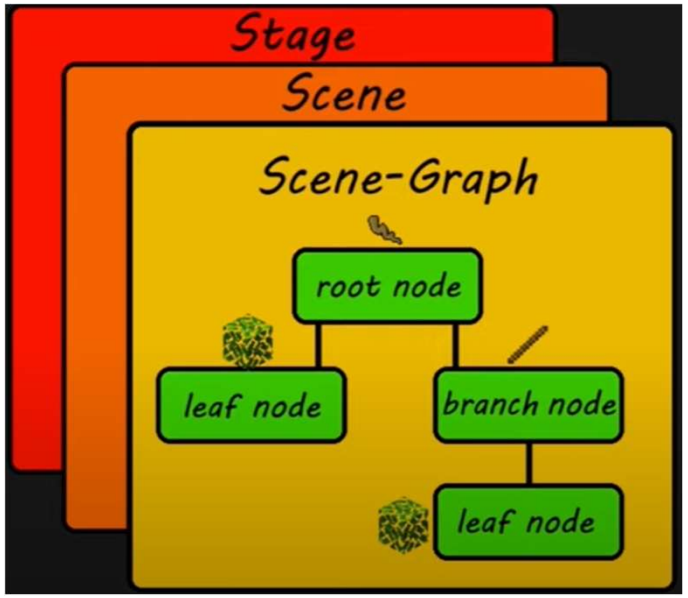
A hierarchical tree that can hold and organize nodes

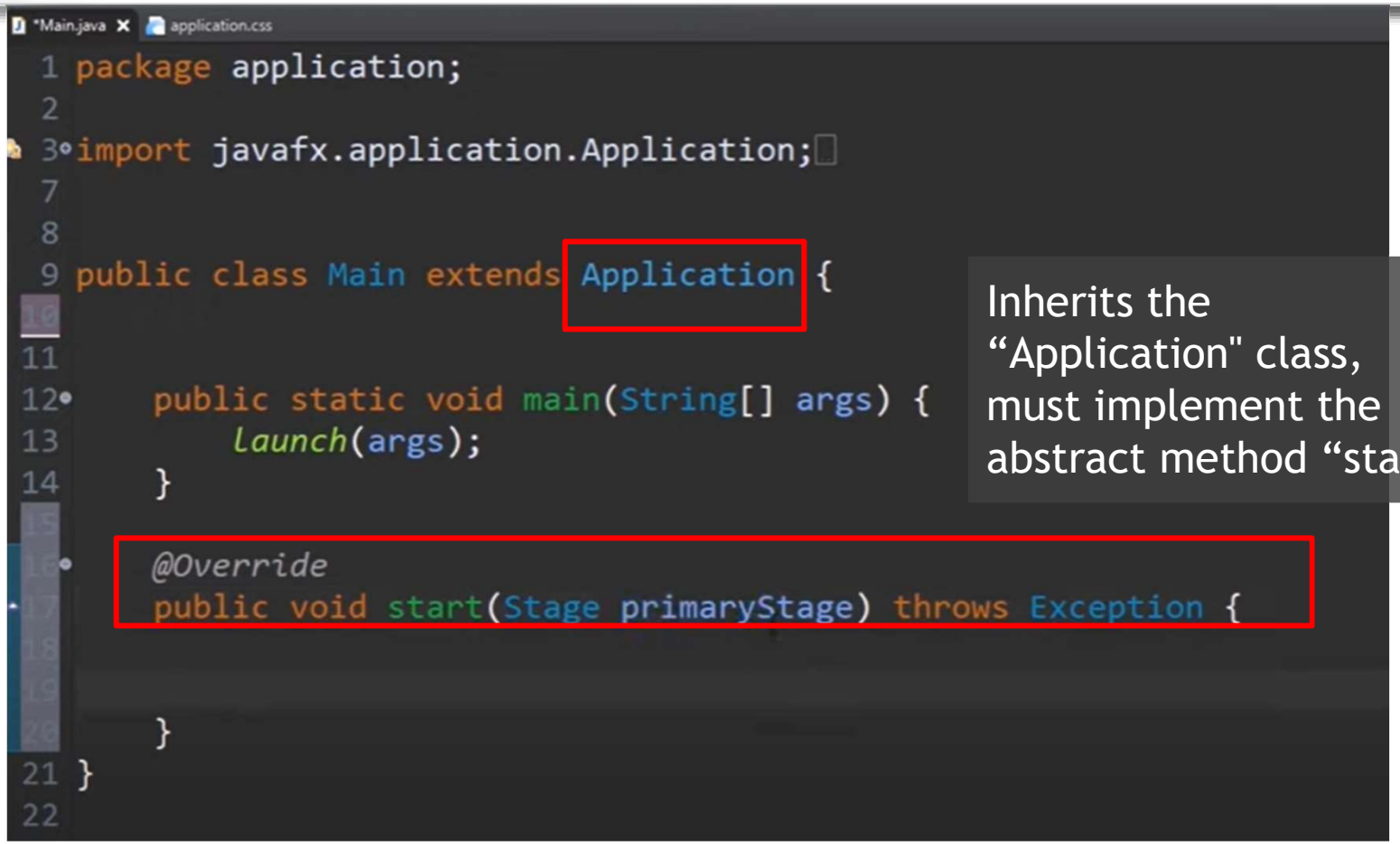


Nodes

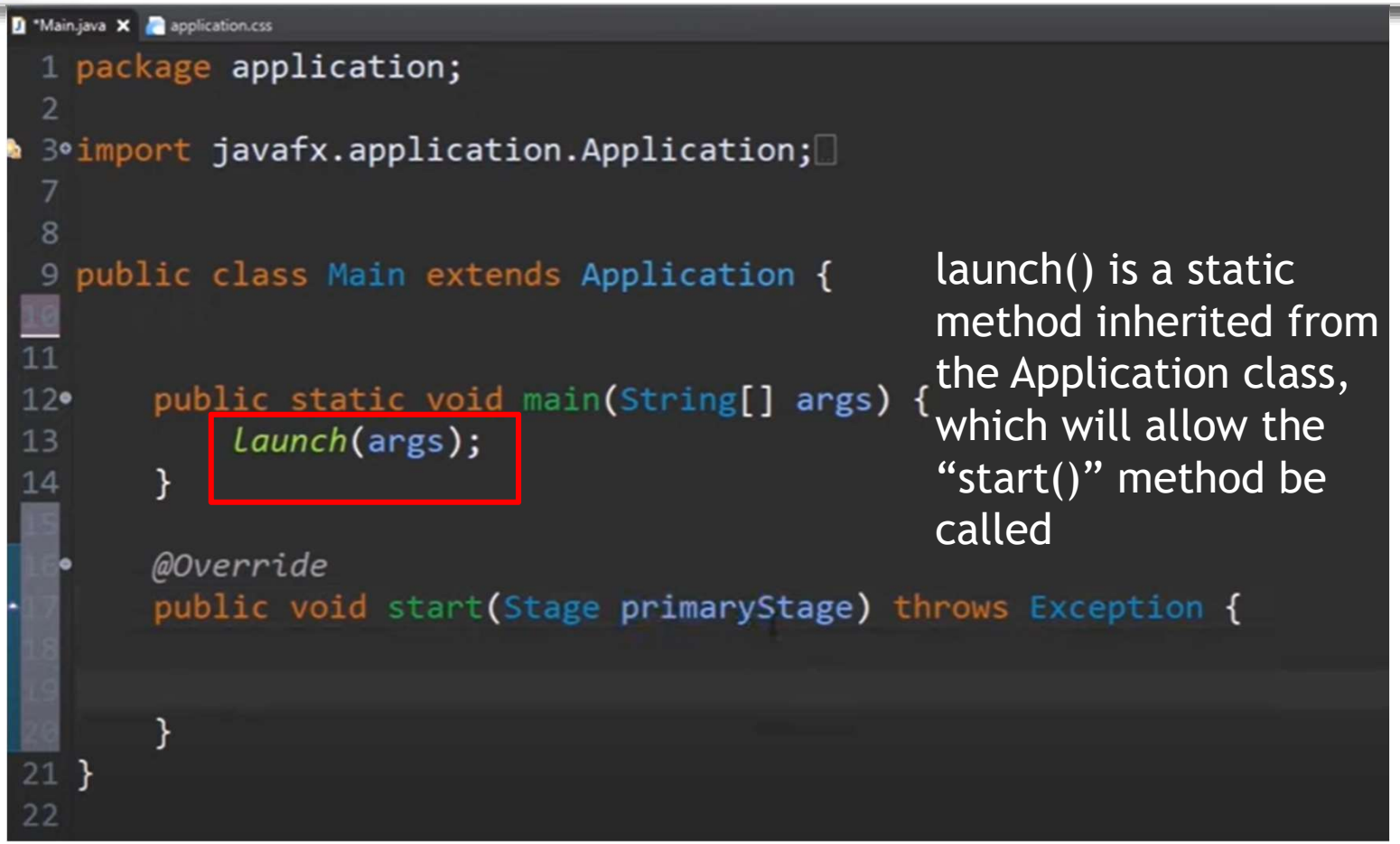
Different components that can be added to the scene

Buttons, textbox, image...





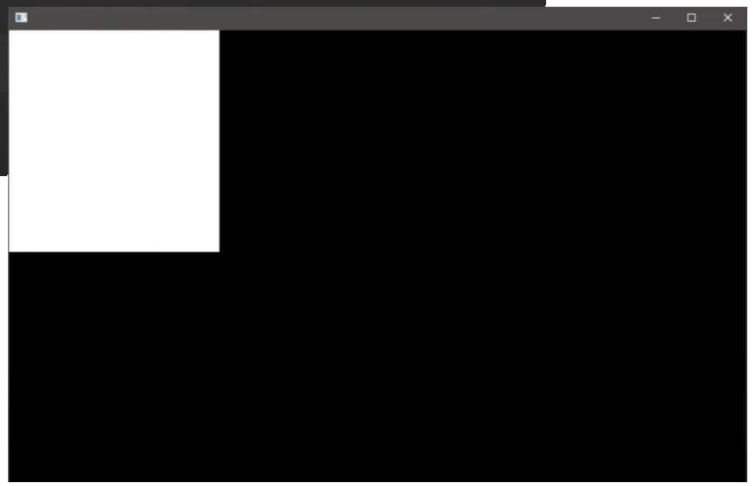
Inherits the "Application" class, must implement the abstract method "start"



launch() is a static method inherited from the Application class, which will allow the “start()” method be called

Stage

```
@Override
public void start(Stage stage) throws Exception {

    //Stage stage = new Stage();

    stage.show();
}
```

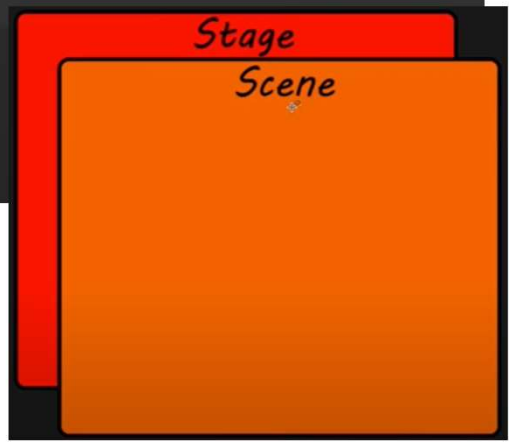


If we run this, a blank window will appear, but when you resize it, the background will be weird

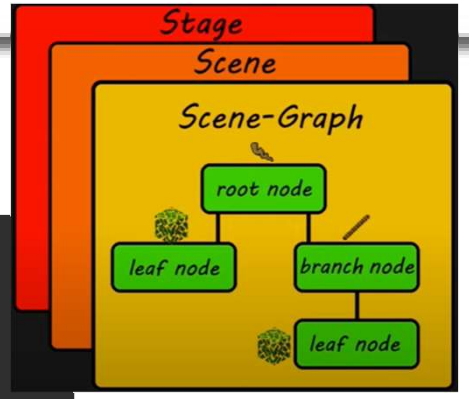
Scene

```
16    @Override
17    public void start(Stage stage) throws Exception {
18
19        //Stage stage = new Stage();
20        Scene scene = new Scene();
21
22        stage.show();
23    }
```

But we need at least a node for it to show



Group




```
@Override
public void start(Stage stage) throws Exception {

    //Stage stage = new Stage();
    Group root = new Group();
    Scene scene = new Scene(root);


    stage.show();
```

"Group" can be think as a group of nodes

Put your group into the Scene

```
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Group;
import javafx.scene.Scene;
```

Remember to import the corresponding libraries

Group

```
@Override
public void start(Stage stage) throws Exception {

    //Stage stage = new Stage();
    Group root = new Group();
    Scene scene = new Scene(root);

    stage.setScene(scene);
    stage.show();
}
```

Set the scene into the stage

Now if you run your app, the background will be resizable

Other Customization

```
@Override
public void start(Stage stage) throws Exception {

    //Stage stage = new Stage();
    Group root = new Group();
    Scene scene = new Scene(root,Color.BLACK);

    stage.setTitle("Stage Demo Program w00t w00t");
    stage.setWidth(420);
    stage.setHeight(420);
    stage.setResizable(false);
    stage.setX(50);
    stage.setY(50);
```

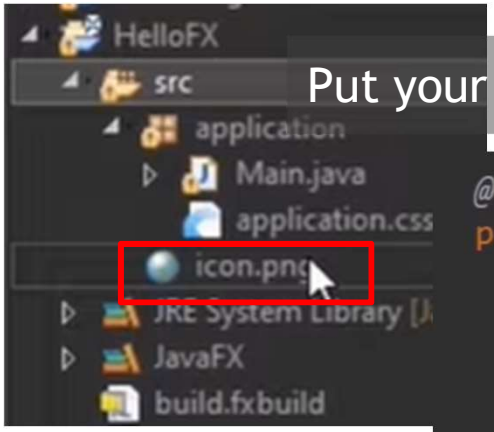
Set the color of the scene, remember to import the color package

Set the title of the stage

Make the window not resizable

Set where the window will appear on screen

Changing App Icon

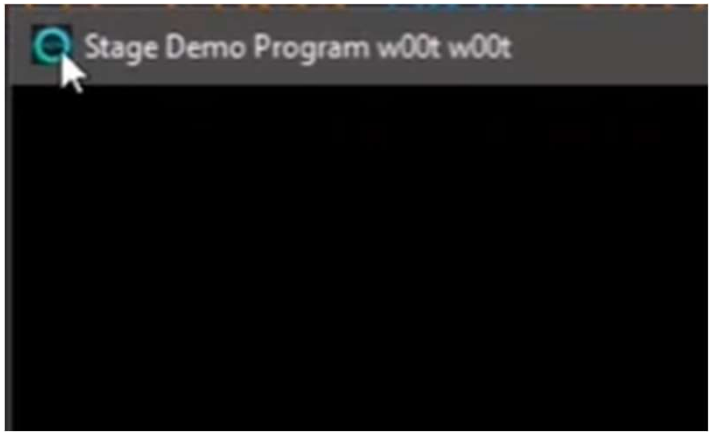


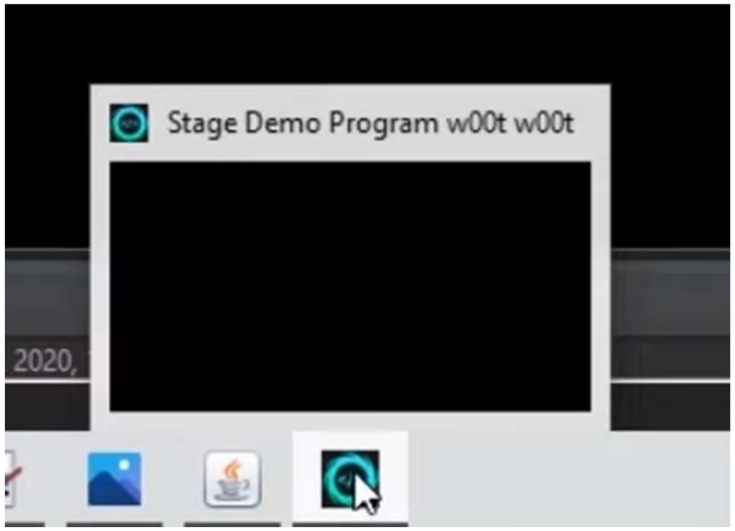
Put your file under the src folder

```
@Override
public void start(Stage stage) throws Exception {

    //Stage stage = new Stage();
    Group root = new Group();
    Scene scene = new Scene(root,Color.BLACK);

    Image icon = new Image("icon.png");
    stage.getIcons().add(icon);
    stage.setTitle("Stage Demo Program w00t w00t");

    stage.setScene(scene);
    stage.show();
}
```

Make an Image object with the path to your icon file and add it to the stage

Changing App Icon



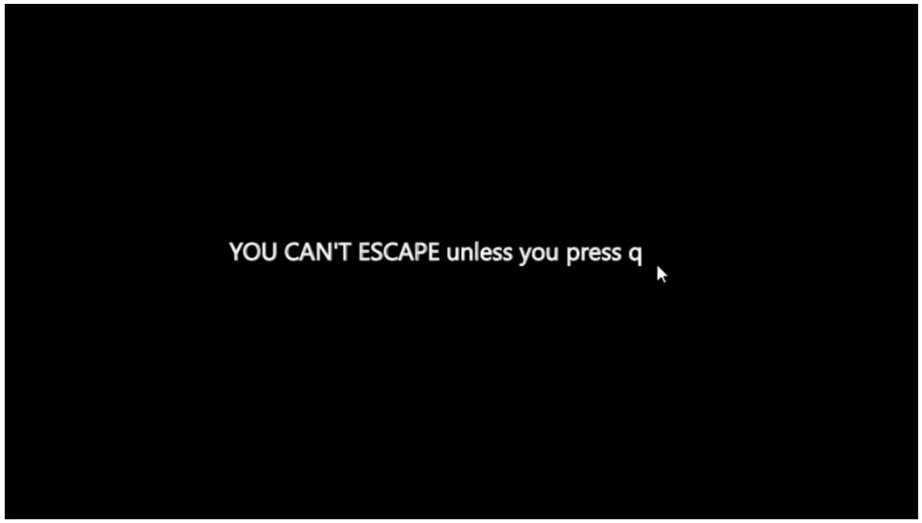


Full Screen Mode

```
stage.setFullScreen(true);
stage.setFullScreenExitHint("YOU CAN'T ESCAPE unless you press q");
stage.setFullScreenExitKeyCombination(KeyCombination.valueOf("q"));
```

The hint message shown to users

change the key combination to exit full screen



3.
Scenes

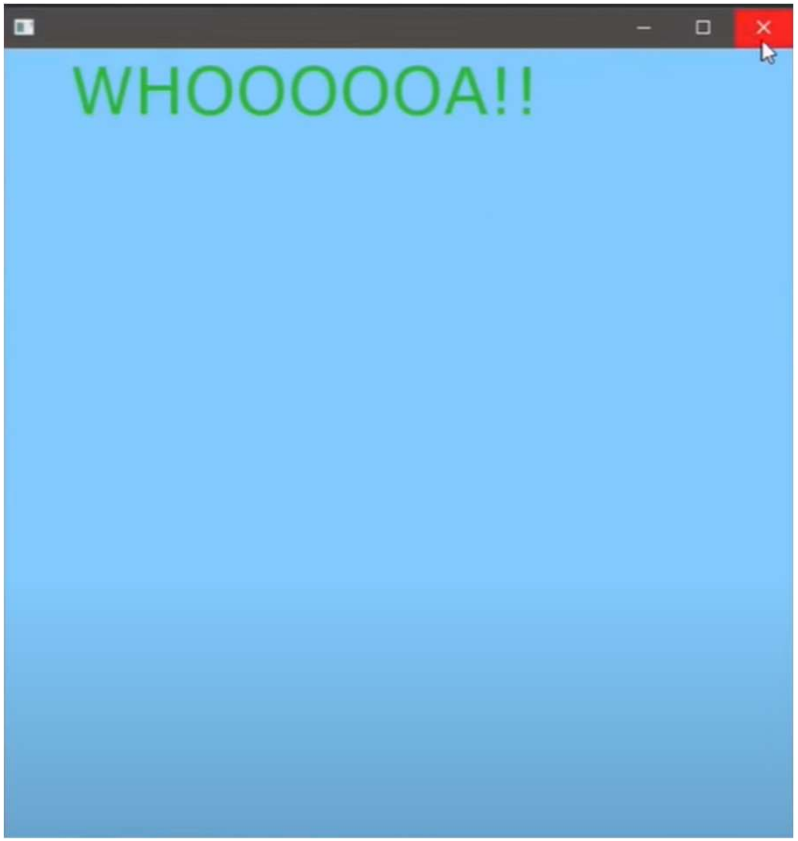
Text

```java
@Override
public void start(Stage primaryStage) throws Exception {

    Group root = new Group();
    Scene scene = new Scene(root,600,600,Color.LIGHTSKYBLUE);
    Stage stage = new Stage();

    Text text = new Text();
    text.setText("WHOOOOOA!!");
    text.setX(50);
    text.setY(50);
    text.setFont(Font.font("Verdana",50));
    text.setFill(Color.LIMEGREEN);

    root.getChildren().add(text);
    stage.setScene(scene);
    stage.show();
```

Make a Text object

Add the text object to the root group

Text



Line

```
Line line = new Line();
line.setStartX(200);
line.setStartY(200);
line.setEndX(500);
line.setEndY(200);
line.setStrokeWidth(5);
line.setStroke(Color.RED);
line.setOpacity(0.5);
line.setRotate(45);

root.getChildren().add(text);
root.getChildren().add(line);
```

Make a Line object

Add the line object to the root group

Line

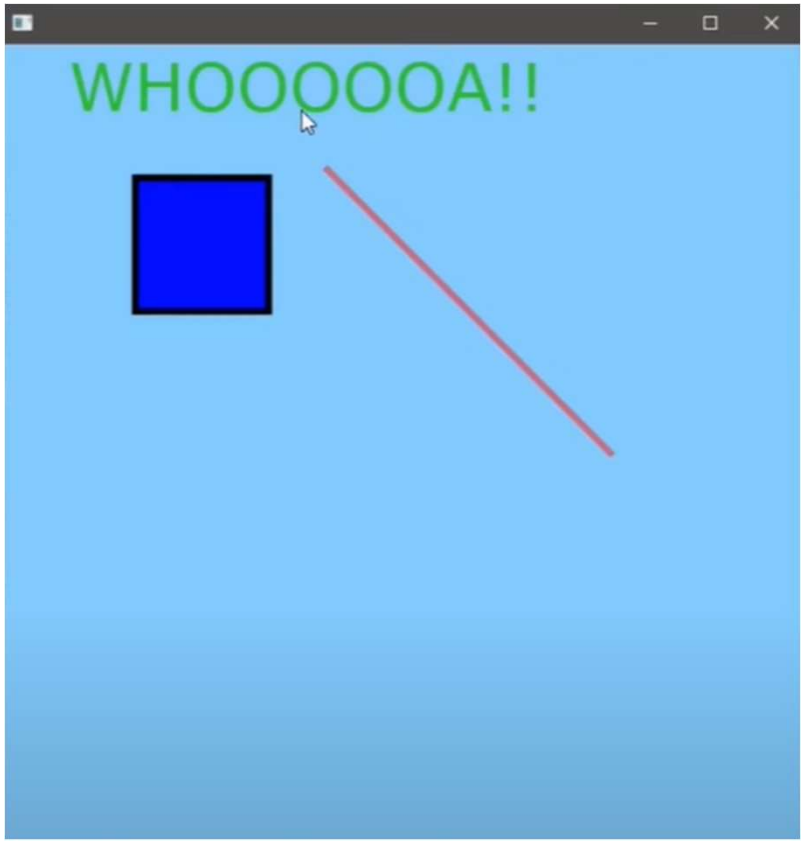
Rectangle

```
Rectangle rectangle = new Rectangle();
rectangle.setX(100);
rectangle.setY(100);
rectangle.setWidth(100);
rectangle.setHeight(100);
rectangle.setFill(Color.BLUE);
rectangle.setStrokeWidth(5);
rectangle.setStroke(Color.BLACK);

root.getChildren().add(text);
root.getChildren().add(line);
root.getChildren().add(rectangle);
stage.setScene(scene);
stage.show();
```

Make a Rectangle object

Add the rectangle object to the root group

Rectangle



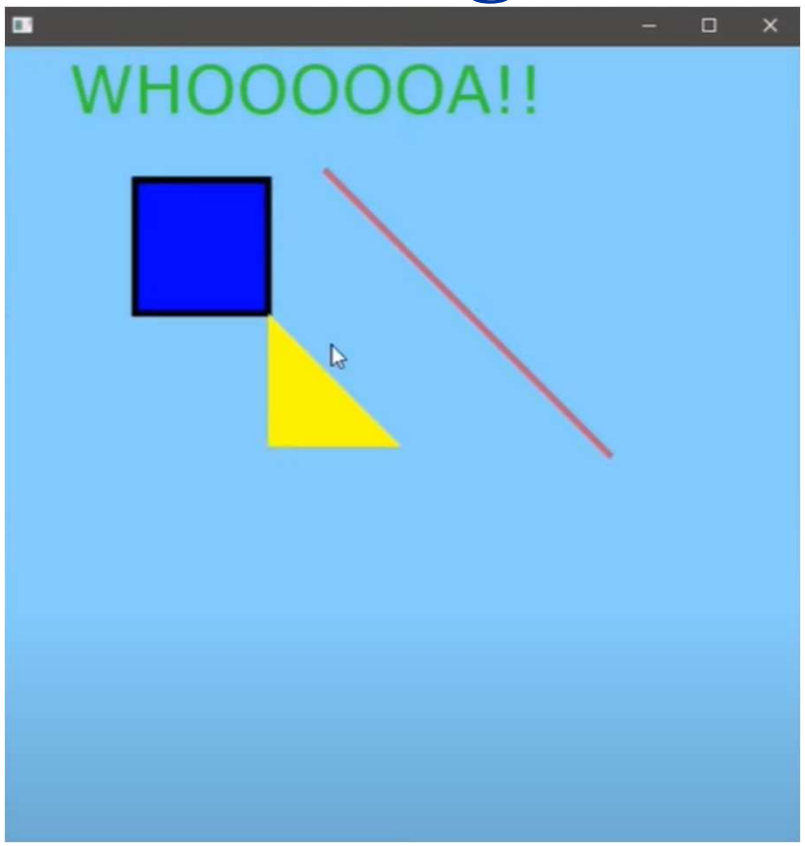
Triangle

```
Polygon triangle = new Polygon();
triangle.getPoints().setAll(
        200.0,200.0,
        300.0,300.0,
        200.0,300.0
        );
triangle.setFill(Color.YELLOW);


root.getChildren().add(text);
root.getChildren().add(line);
root.getChildren().add(rectangle);
root.getChildren().add(triangle);
```

Make a Polygon object and define all of its points

Add the polygon object to the root group

Triangle



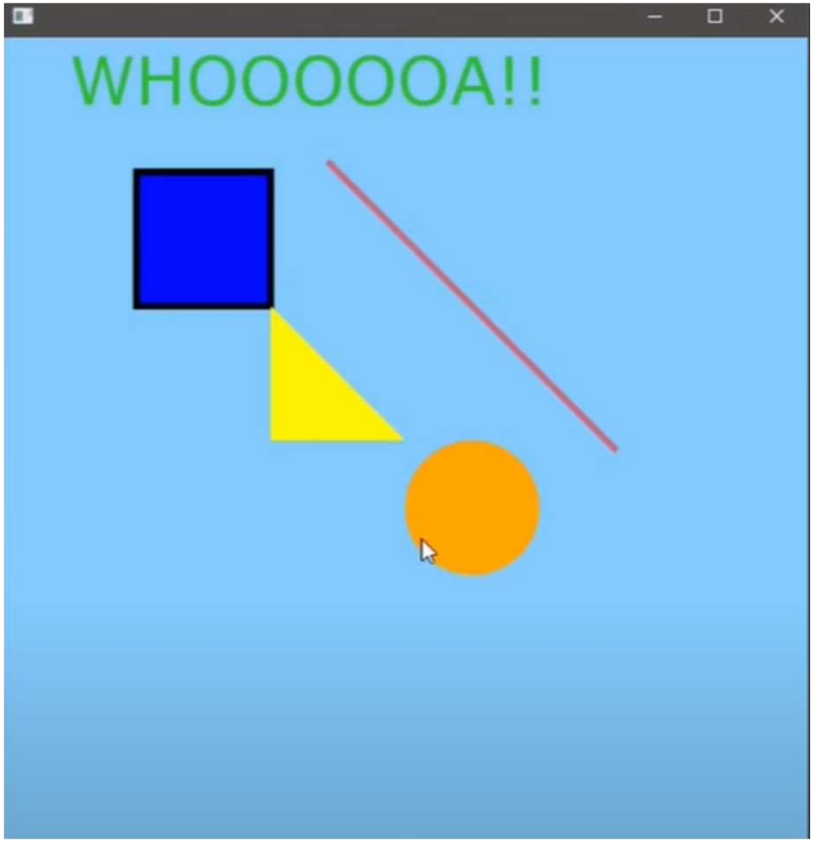
Circle

```
Circle circle = new Circle();
circle.setCenterX(350);
circle.setCenterY(350);
circle.setRadius(50);
circle.setFill(Color.ORANGE);

root.getChildren().add(text);
root.getChildren().add(line);
root.getChildren().add(rectangle);
root.getChildren().add(triangle);
root.getChildren().add(circle);
```

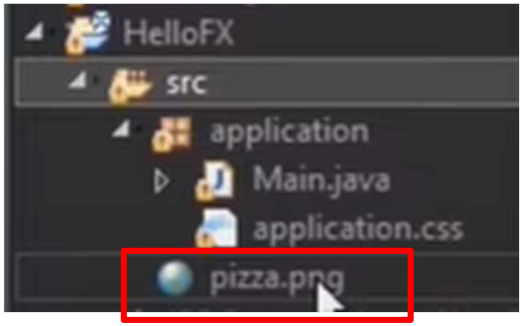
Make a Circle object

Add the circle object to the root group

Circle



Insert Image



Place file under src folder

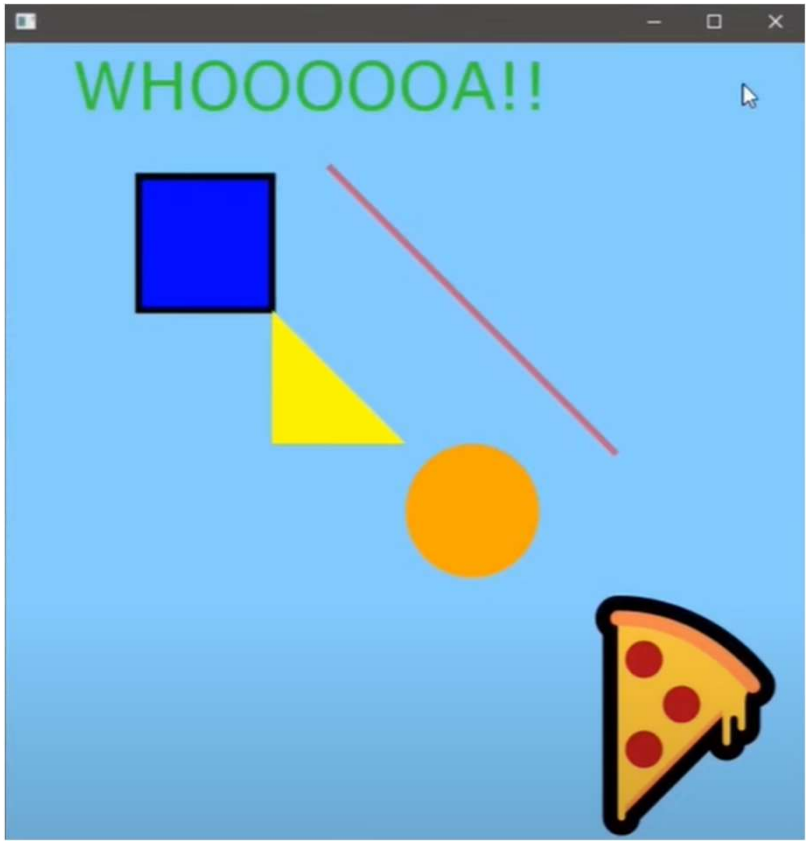
```
Image image = new Image("pizza.png");
ImageView imageView = new ImageView(image);
imageView.setX(400);
imageView.setY(400);

root.getChildren().add(text);
root.getChildren().add(line);
root.getChildren().add(rectangle);
root.getChildren().add(triangle);
root.getChildren().add(circle);
root.getChildren().add(imageView);
stage.setScene(scene);
stage.show();
```

Create an Image object with your file

Create an imageView object with your image object

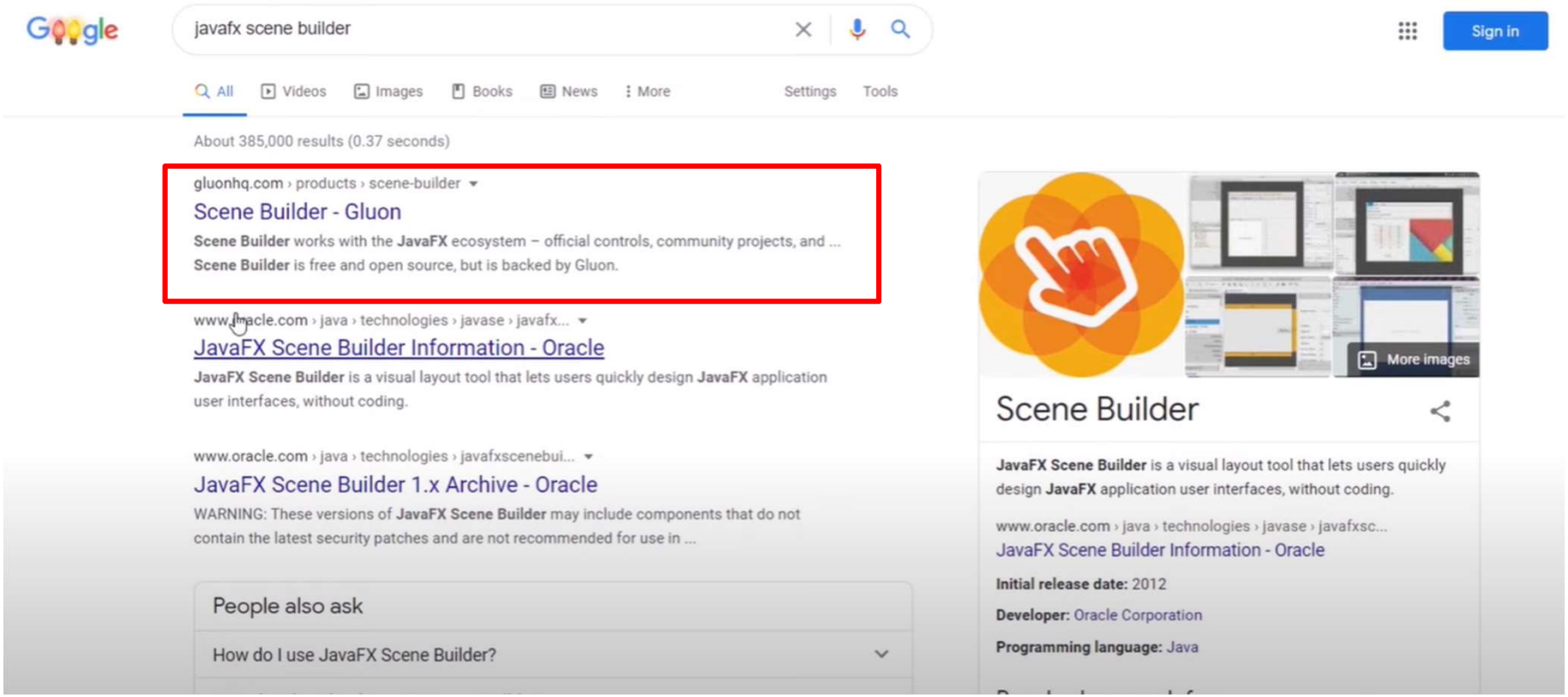
Add the imageView object to the root group

Image



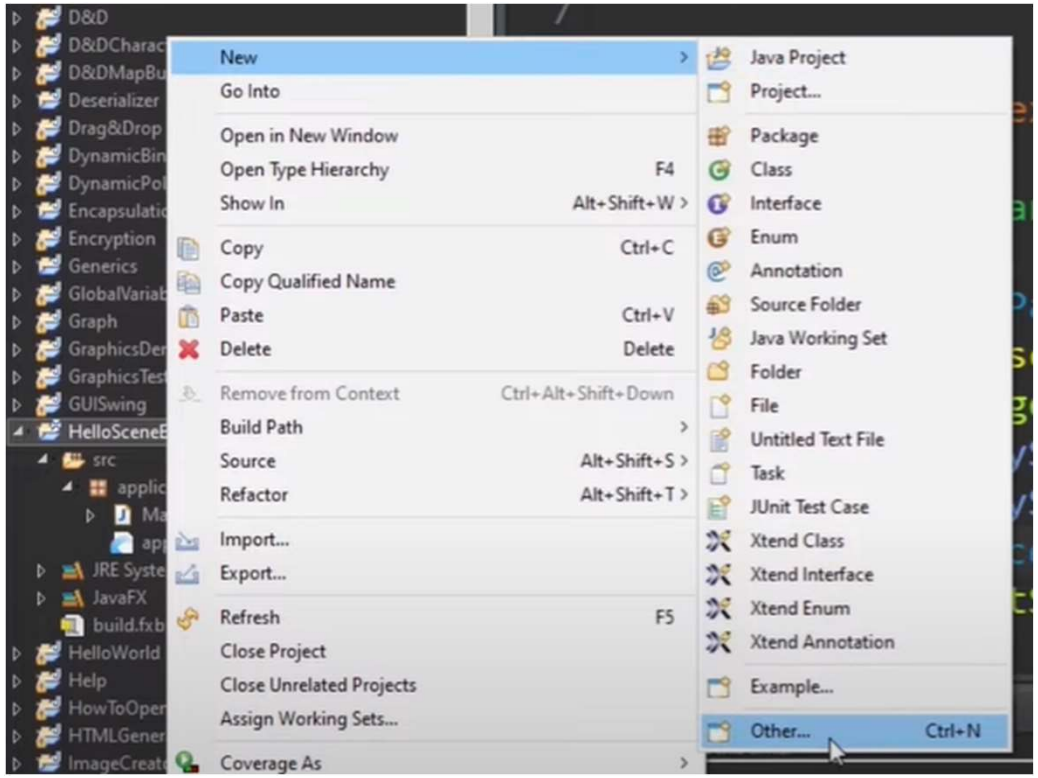
4.
Scene Builders

Download and Install Scene Builder



Eclipse Installation

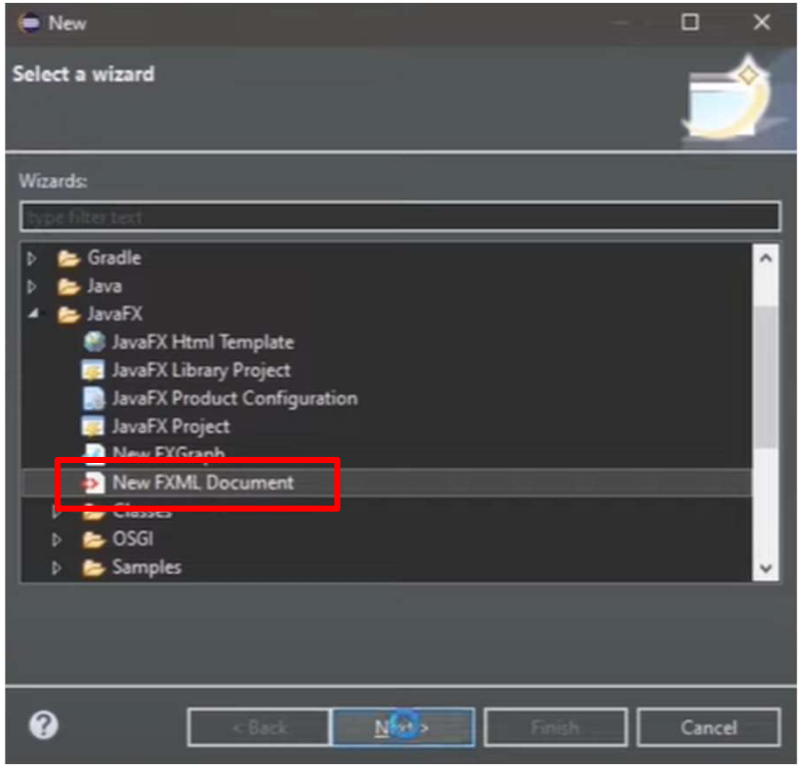
1. Create a FX project
2. Right click on the project > **New** > **Other**



Eclipse Installation

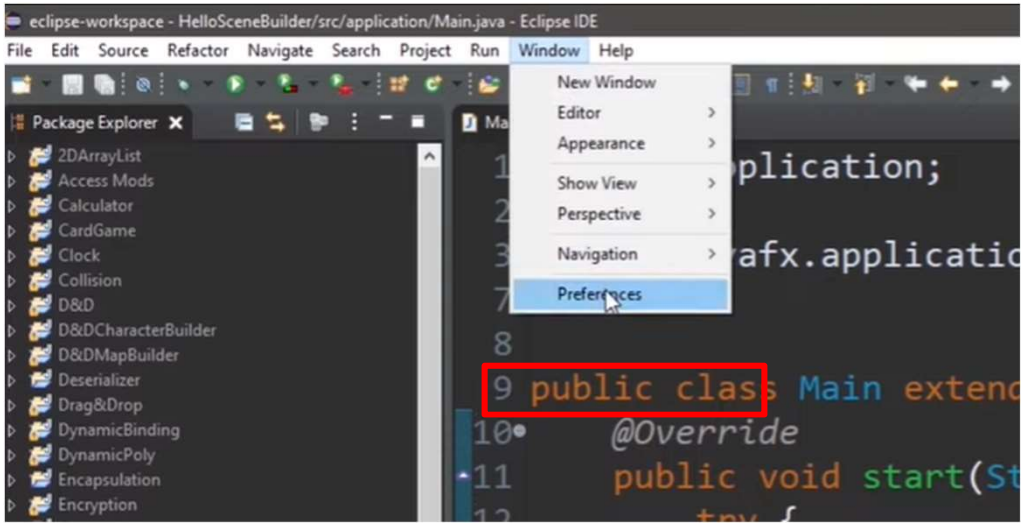
3. Select **New FXML Document**

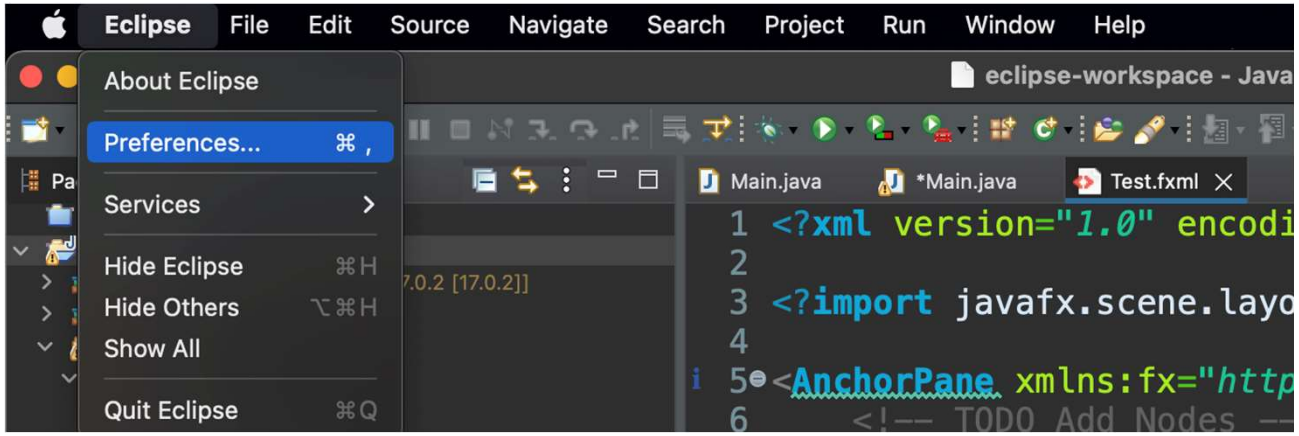
4. Type in the Name for your FXML File and Finish



Eclipse Installation

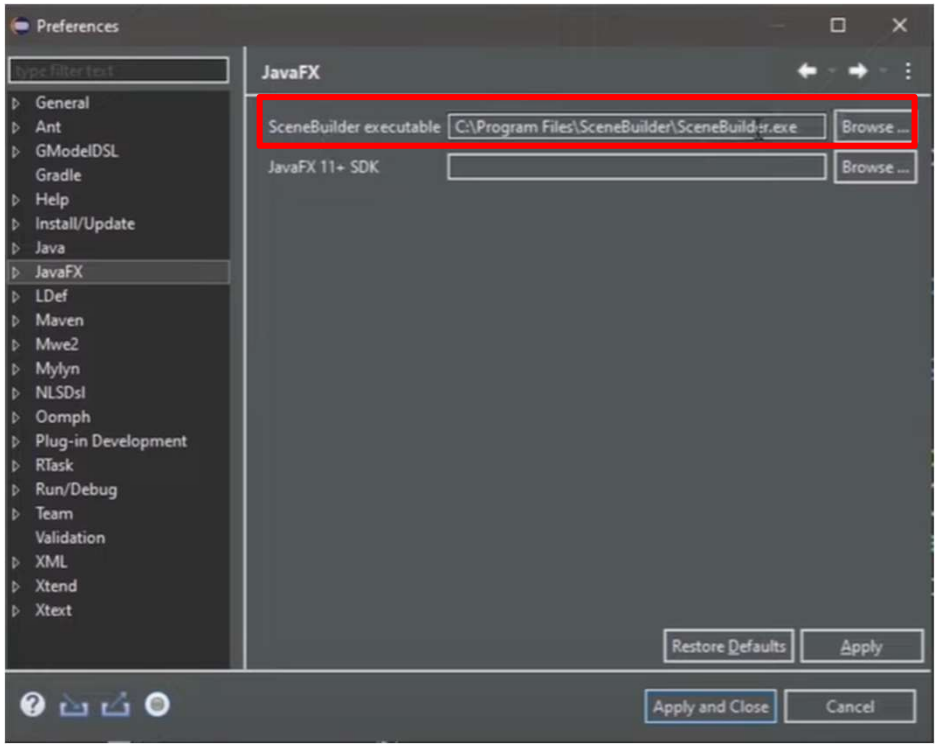
5. Window > Preferences





Eclipse Installation

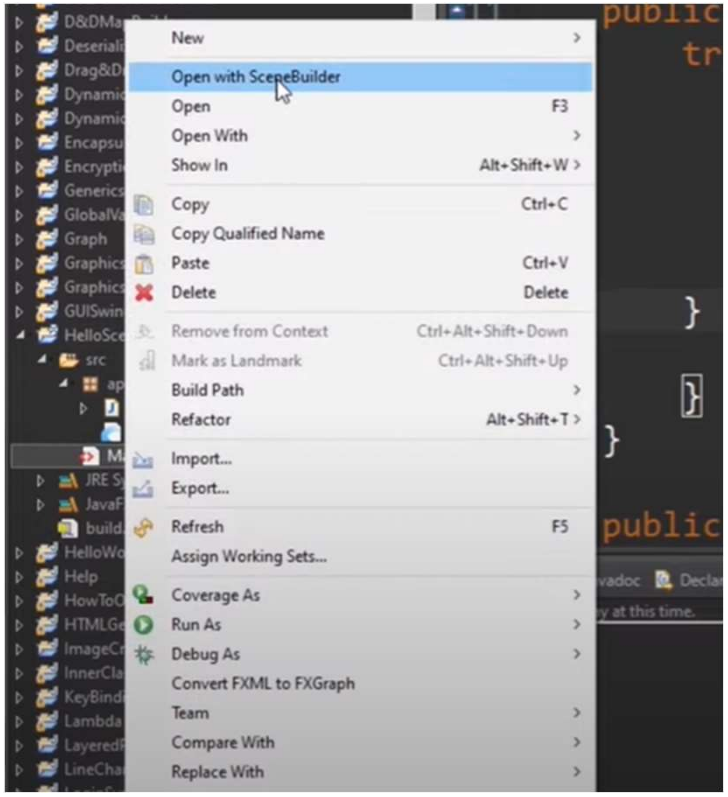
6. Browse for the path of the SceneBuilder executable file you downloaded



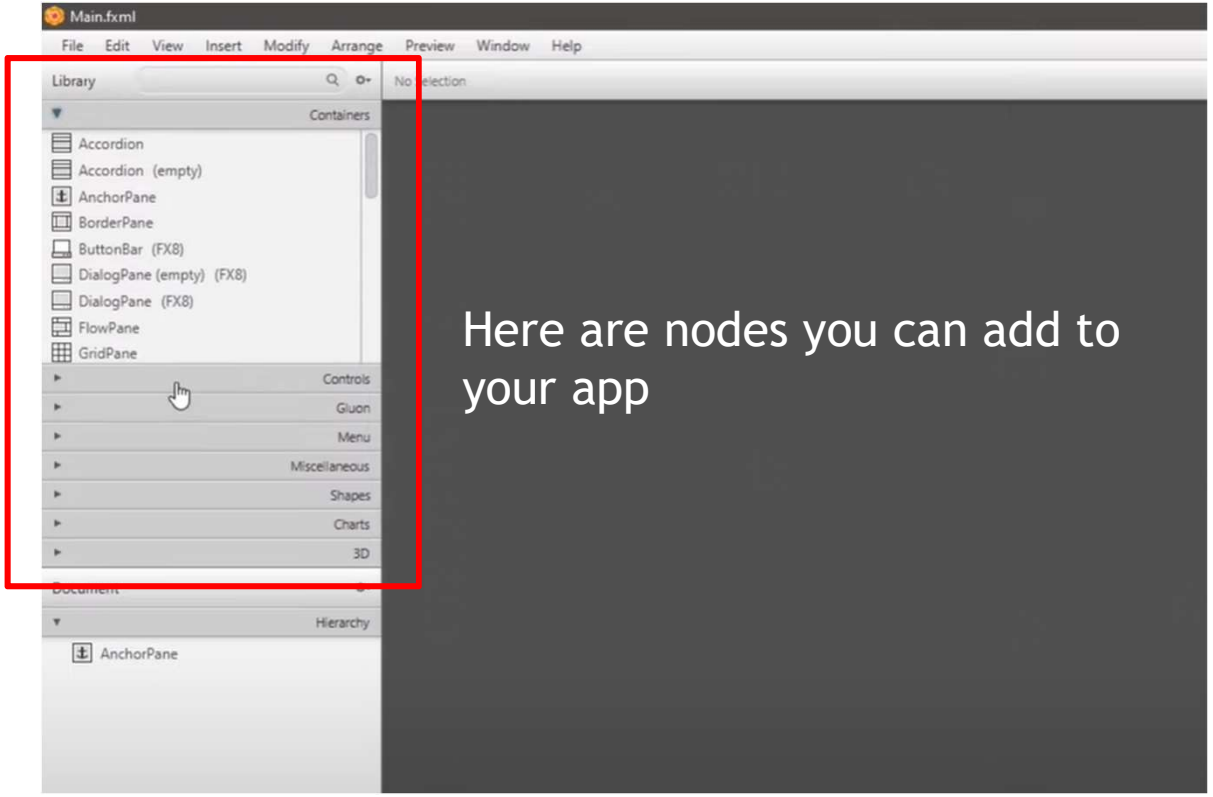
Mac: /Applications/SceneBuilder.app

Eclipse Installation

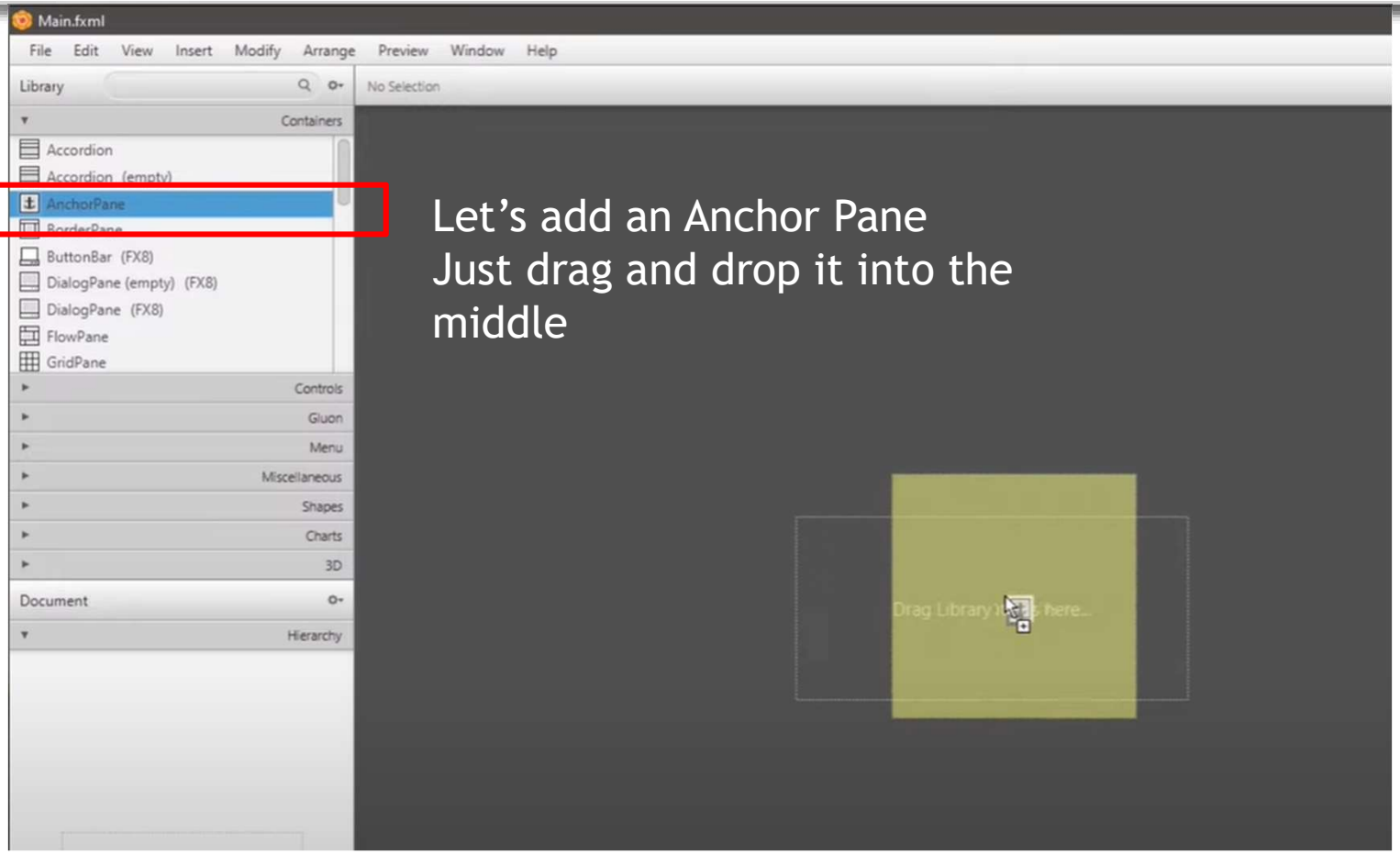
7. Right click on the FXML file > Open with SceneBuilder

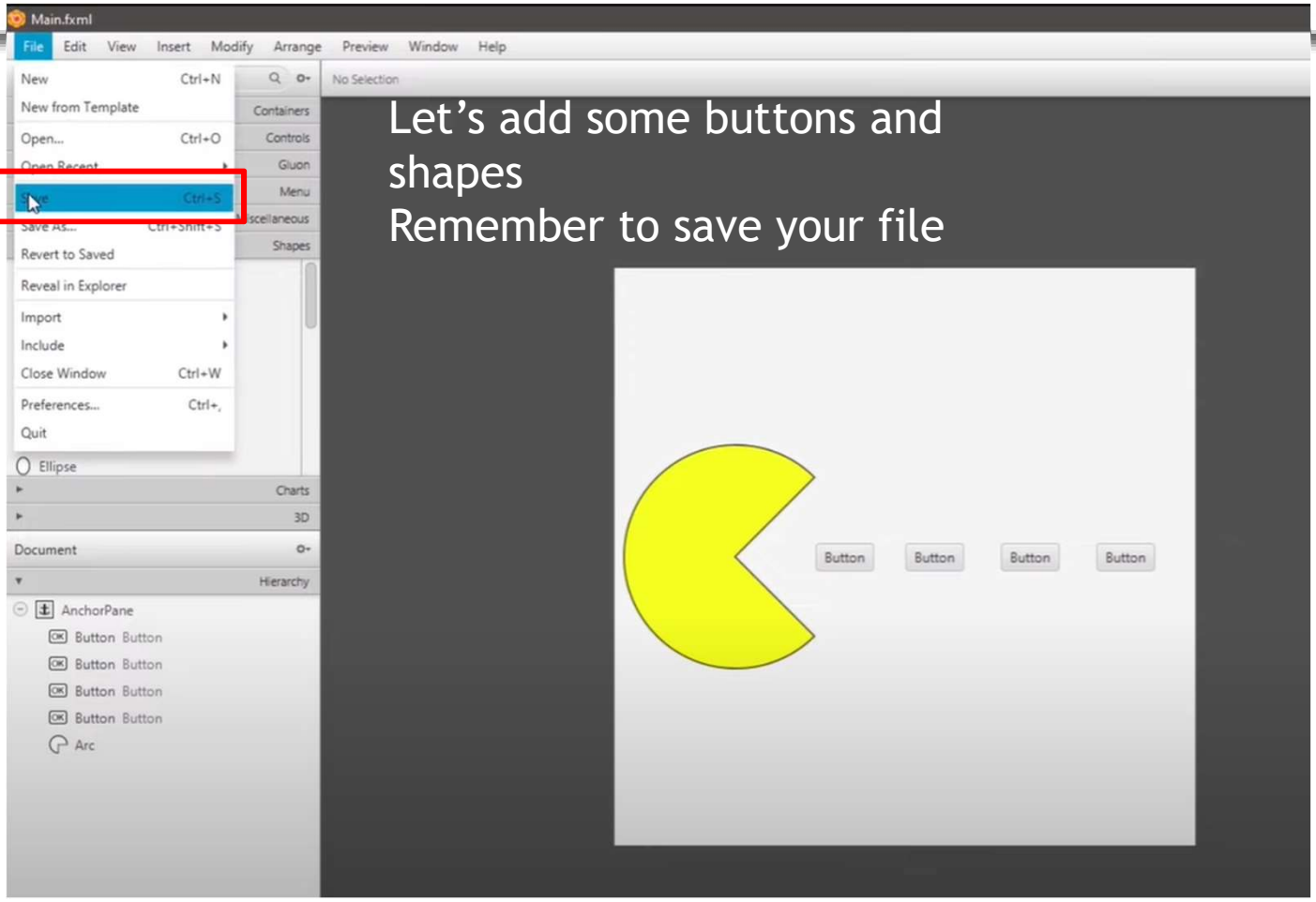


SceneBuilder



Here are nodes you can add to your app


Main.fxml
File Edit View Insert Modify Arrange Preview Window Help
Library
No Selection
Containers
Accordion
Accordion (empty)
AnchorPane
BorderPane
ButtonBar (FX8)
DialogPane (empty) (FX8)
DialogPane (FX8)
FlowPane
GridPane
Controls
Gluon
Menu
Miscellaneous
Shapes
Charts
3D
Document
Hierarchy
Let’s add an Anchor Pane
Just drag and drop it into the middle


Main.fxml
File Edit View Insert Modify Arrange Preview Window Help
New Ctrl+N
New from Template
Open... Ctrl+O
Revert to Saved
Reveal in Explorer
Import
Include
Close Window Ctrl+W
Preferences... Ctrl+,
Quit
Containers
Controls
Menu
Shapes
Ellipse
Charts
3D
Document
Hierarchy
AnchorPane
Button Button
Button Button
Button Button
Button Button
Arc
No Selection
Let’s add some buttons and shapes
Remember to save your file
Button
Button
Button
Button

This is the default start() when you create a FX project

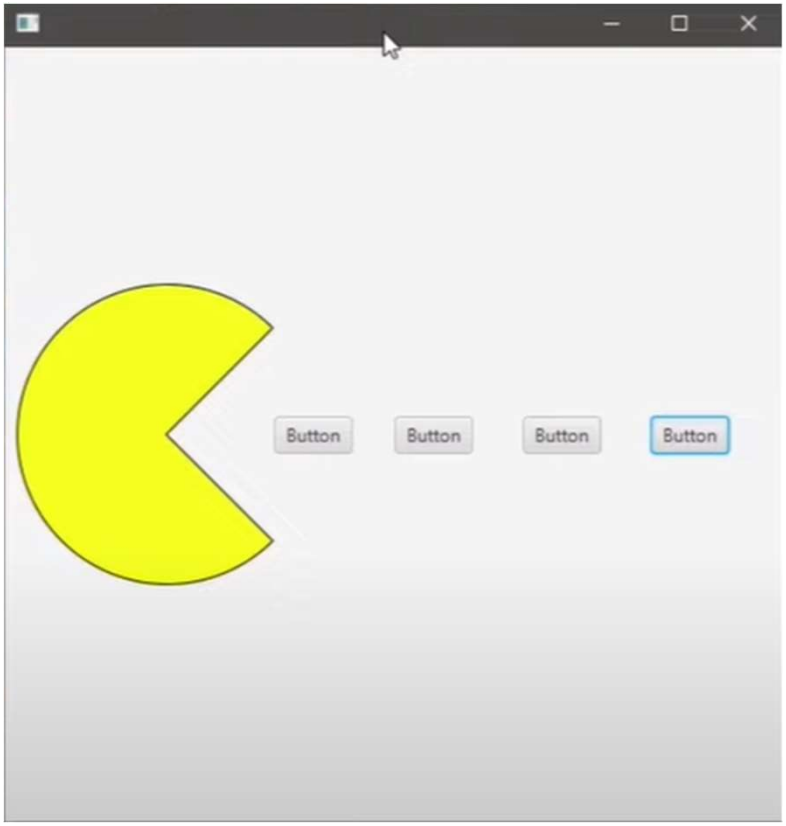
```
public void start(Stage primaryStage) {
    try {
        BorderPane root = new BorderPane();
        Scene scene = new Scene(root,400,400);
        scene.getStylesheets().add(getClass().getResource("application.css")
```

```
public void start(Stage primaryStage) {
    try {
        Parent root = FXMLLoader.load(getClass().getResource("/Main.fxml"));
        Scene scene = new Scene(root);
        scene.getStylesheets().add(getClass().getResource("application.css").
        primaryStage.setScene(scene);
        primaryStage.show();
    } catch(Exception e) {
        e.printStackTrace();
    }
}
```

Change the BorderPane root into a Parent object that will load your FXML file as the resource

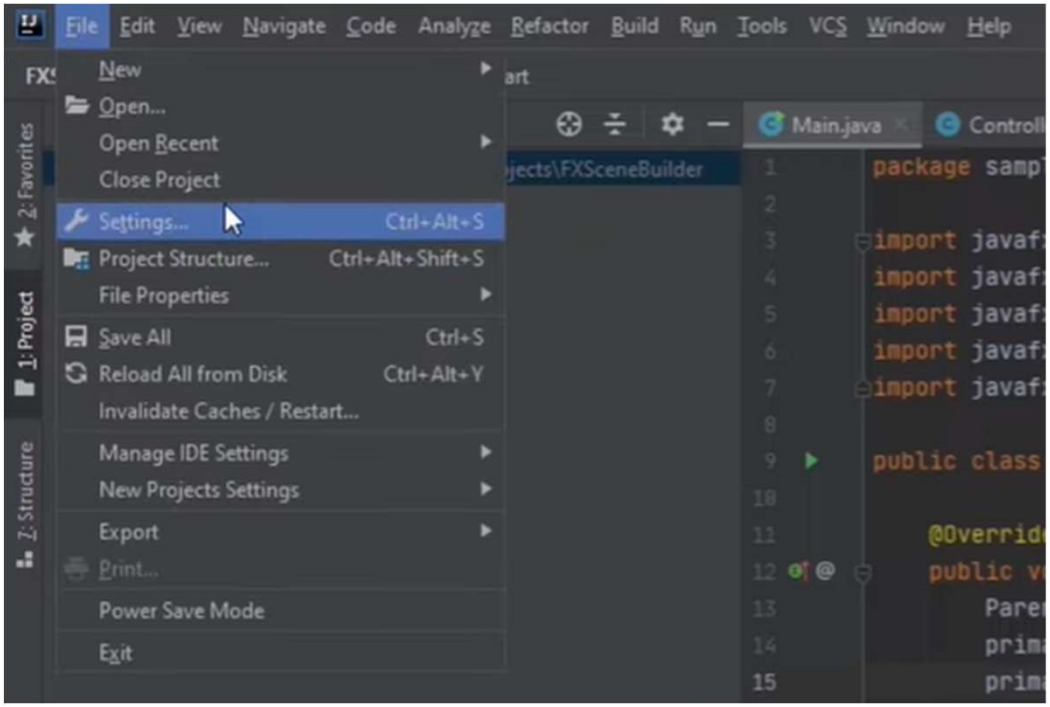
And remove the default scene size

And your app should be running!



IntelliJ Installation

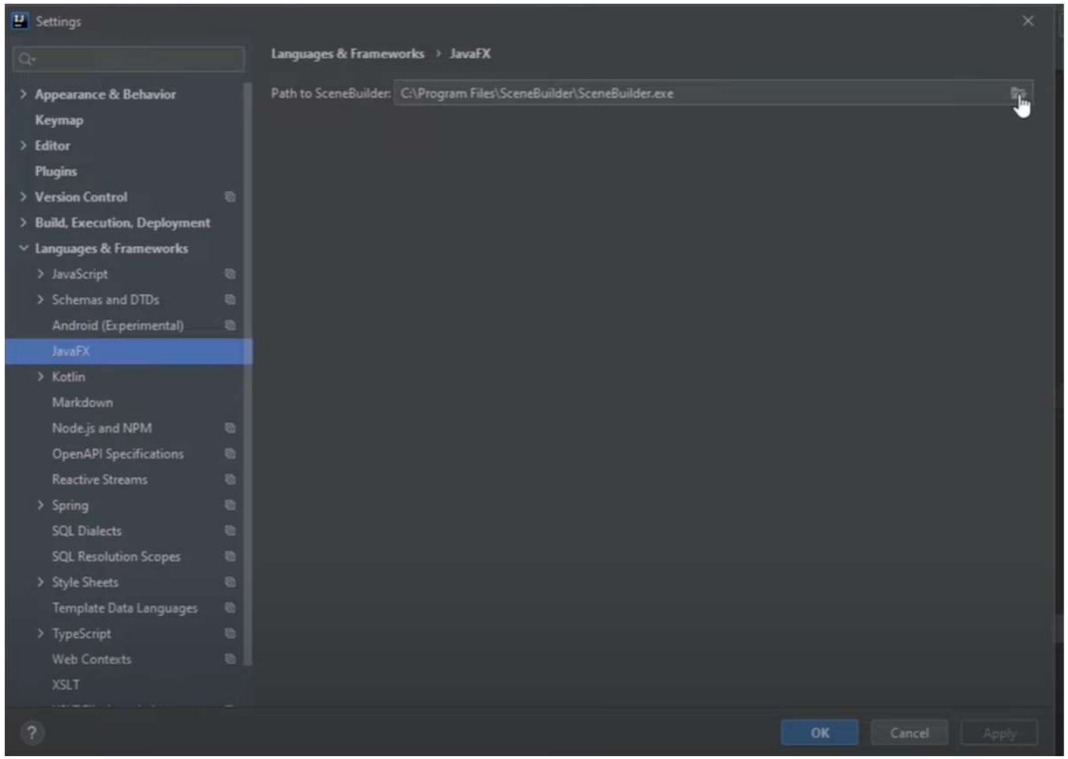
1. Create a FX project
2. Go to **File** > **Settings**



IntelliJ Installation

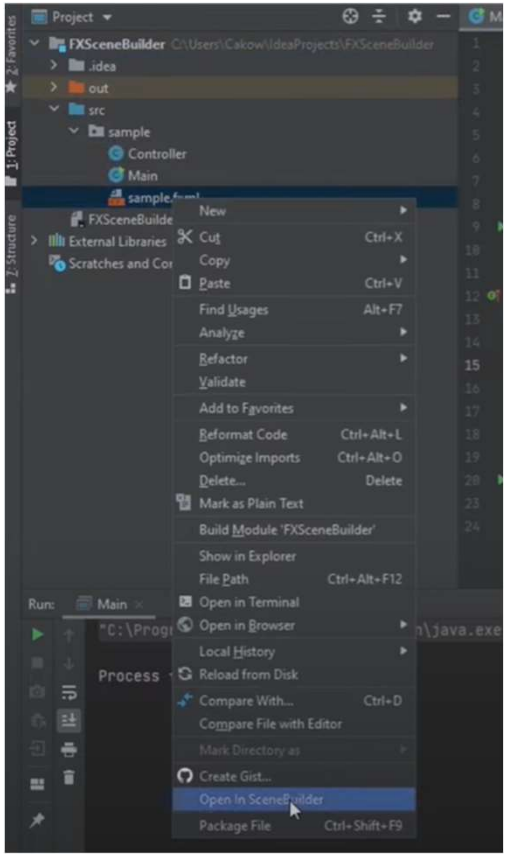
3. Under **Languages & Frameworks > JavaFX**, browse your path to the SceneBuilder executable

4. Click **Apply** and **OK**



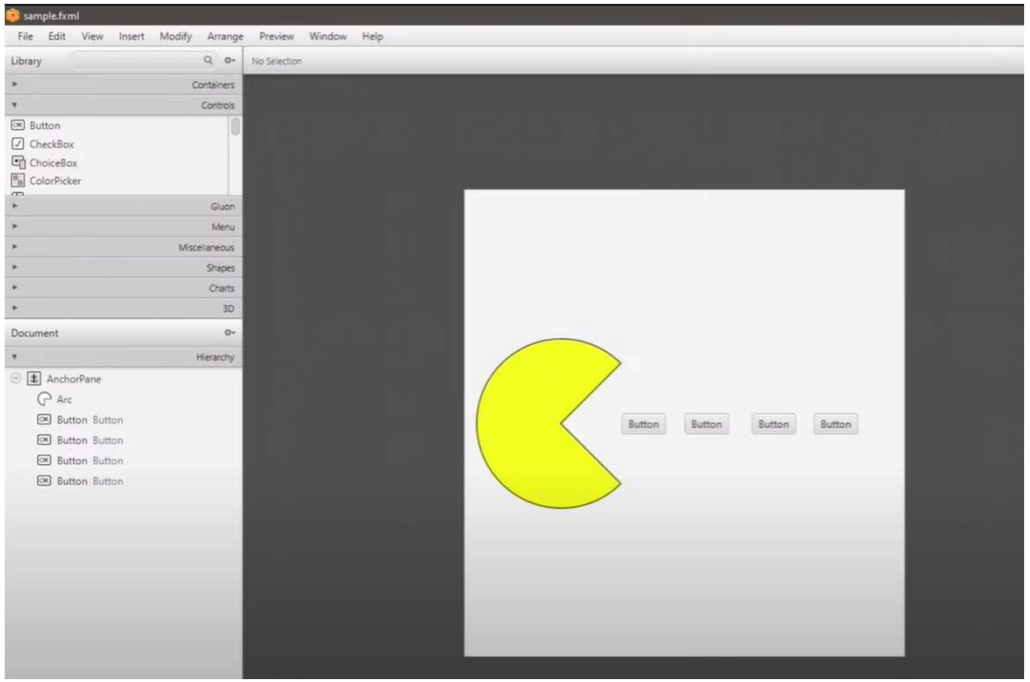
IntelliJ Installation

5. Right click on the FXML file > **Open in SceneBuilder**



IntelliJ Installation

6. Do the same thing in SceneBuilder



IntelliJ Installation

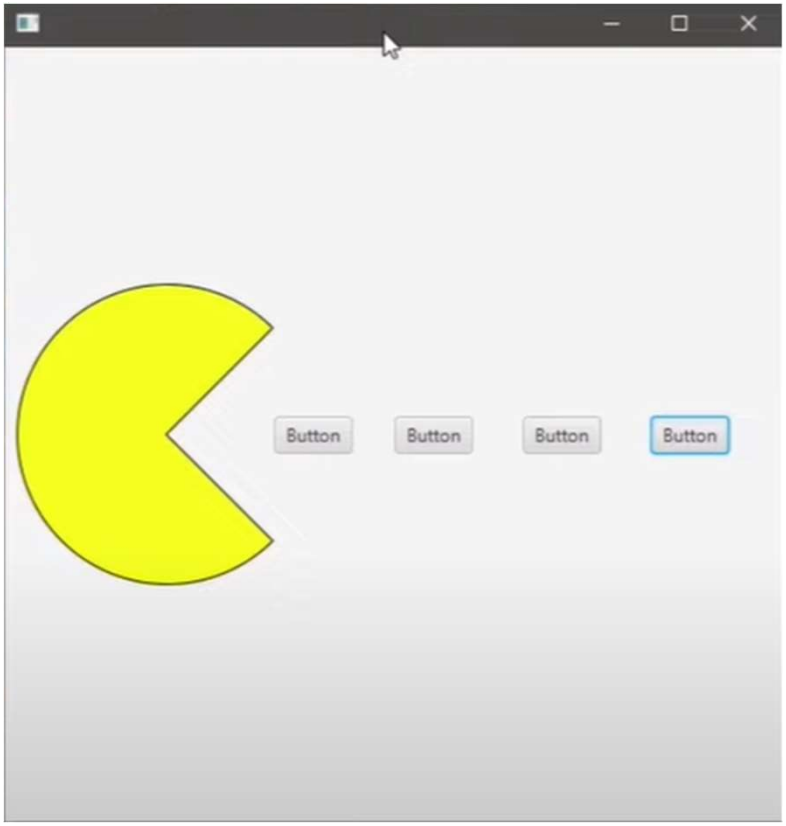
```
@Override
public void start(Stage primaryStage) throws Exception{
    Parent root = FXMLLoader.load(getClass().getResource( name: "sample.fxml"));
    primaryStage.setTitle("Hello World");
    primaryStage.setScene(new Scene(root));
    primaryStage.show();
}
```

link the fxml file to the root node

And delete the default size of the scene

5.
Event Handling

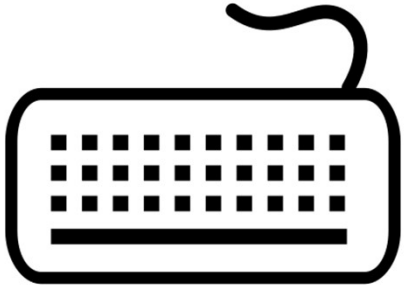
The buttons here have no reaction, why?



Because no events are handled

What are events?

Any events that happens

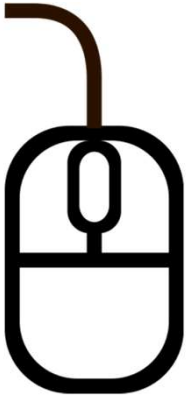


A button has been clicked

The mouse has moved

An arrow key has been pressed

A file has been uploaded



What happens with the events?

Programmers handle them

A button has been clicked → go to another page

The mouse has moved → change color

An arrow key has been pressed → change walking direction

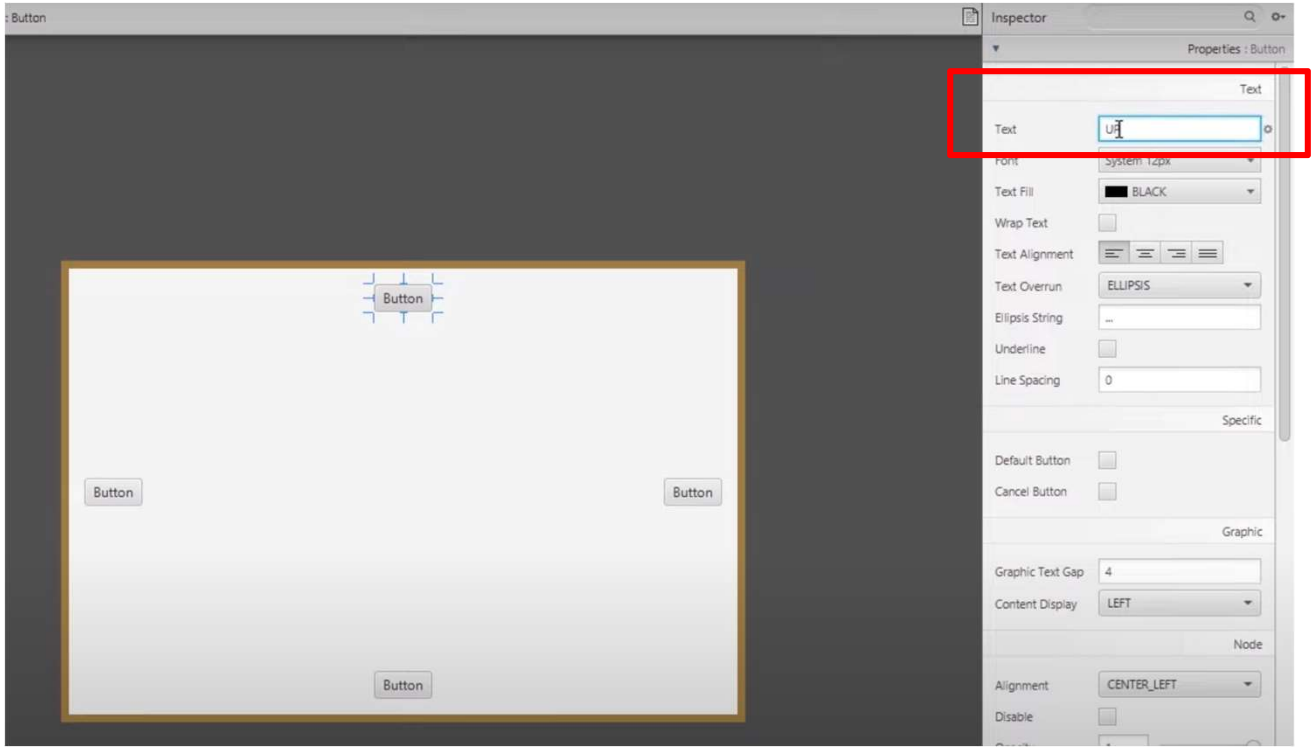
A file has been uploaded → show a popup window

Programs use Listeners to listen for events

Listener must be
24H always
listening if an event
has been fired

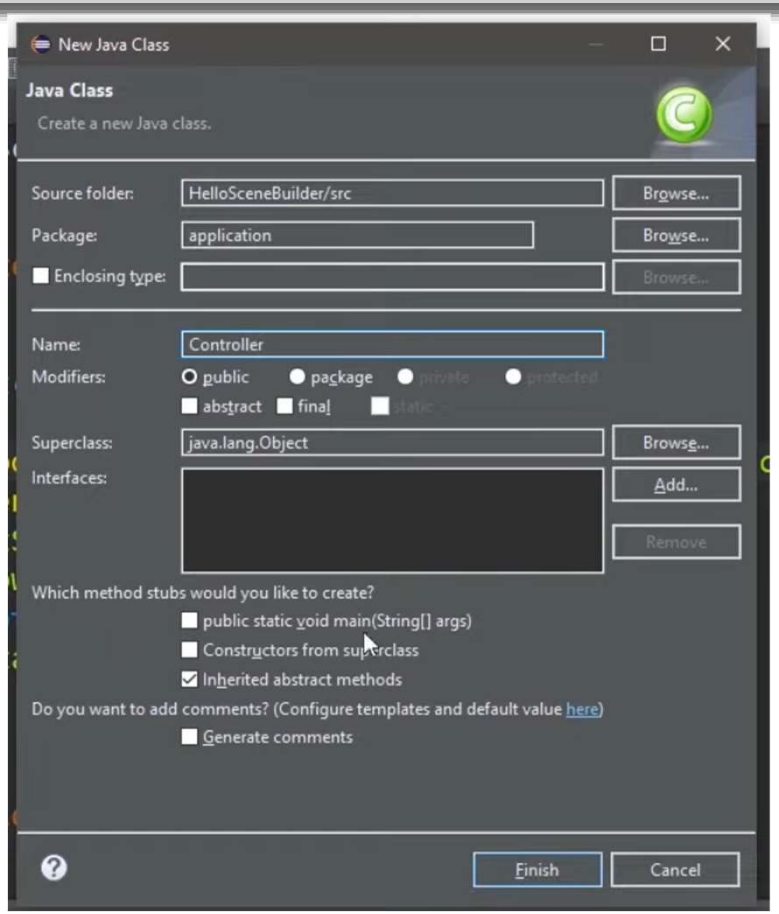
Let’s make some event handlers

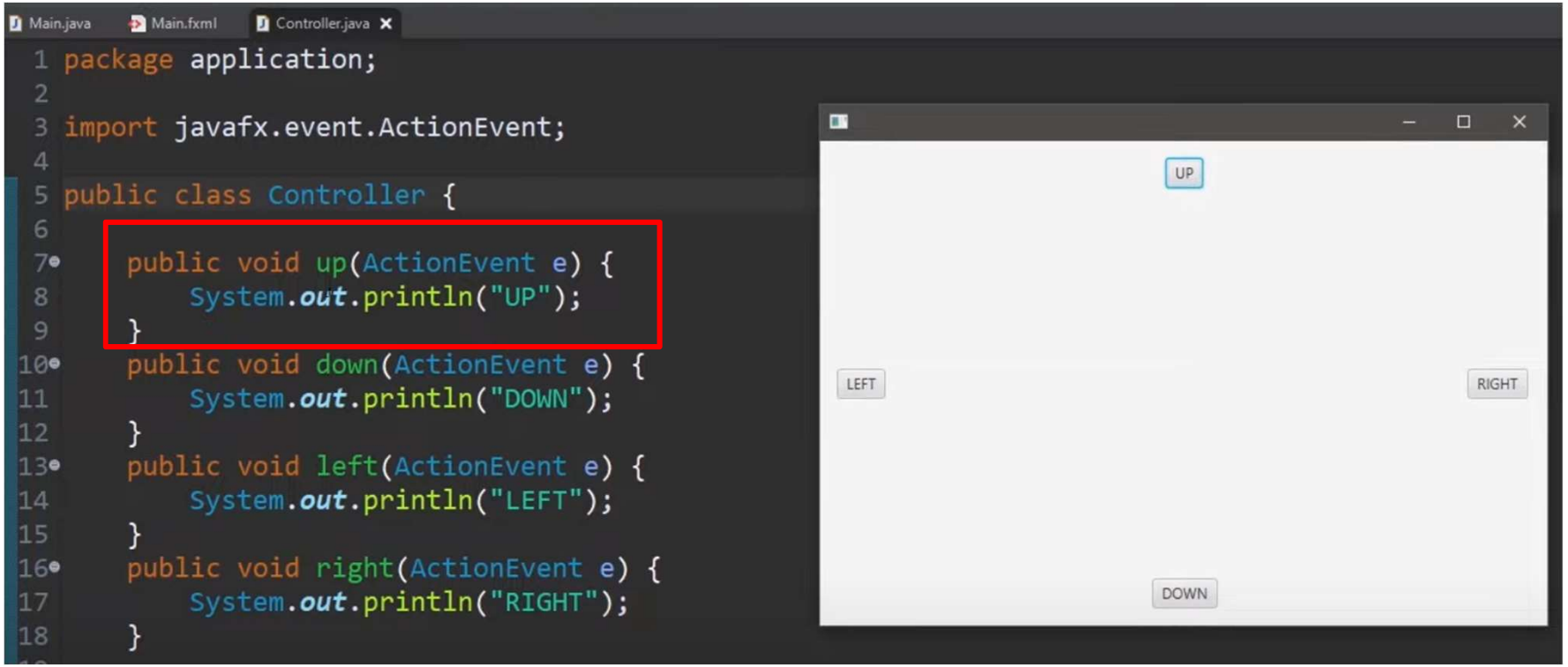
Make a FXML file with 4 buttons



change the text of the button under Properties > Text

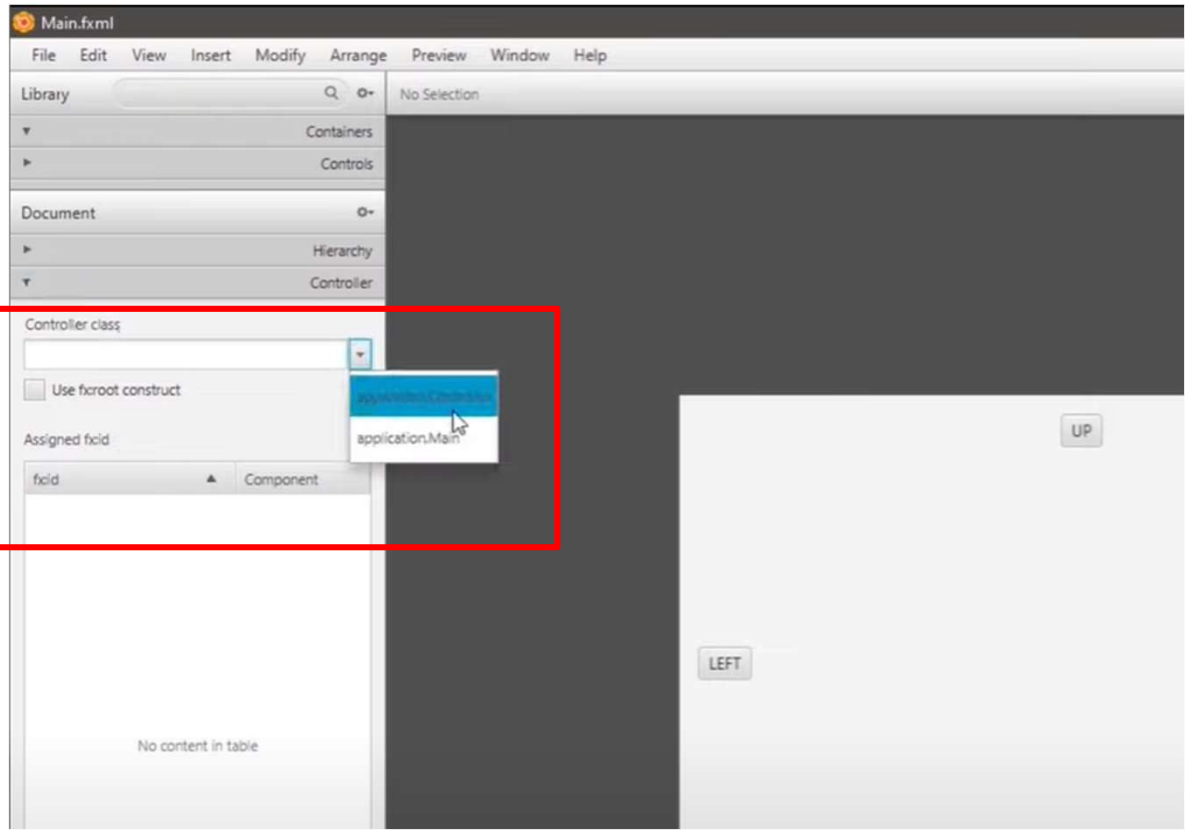
Add a new Class

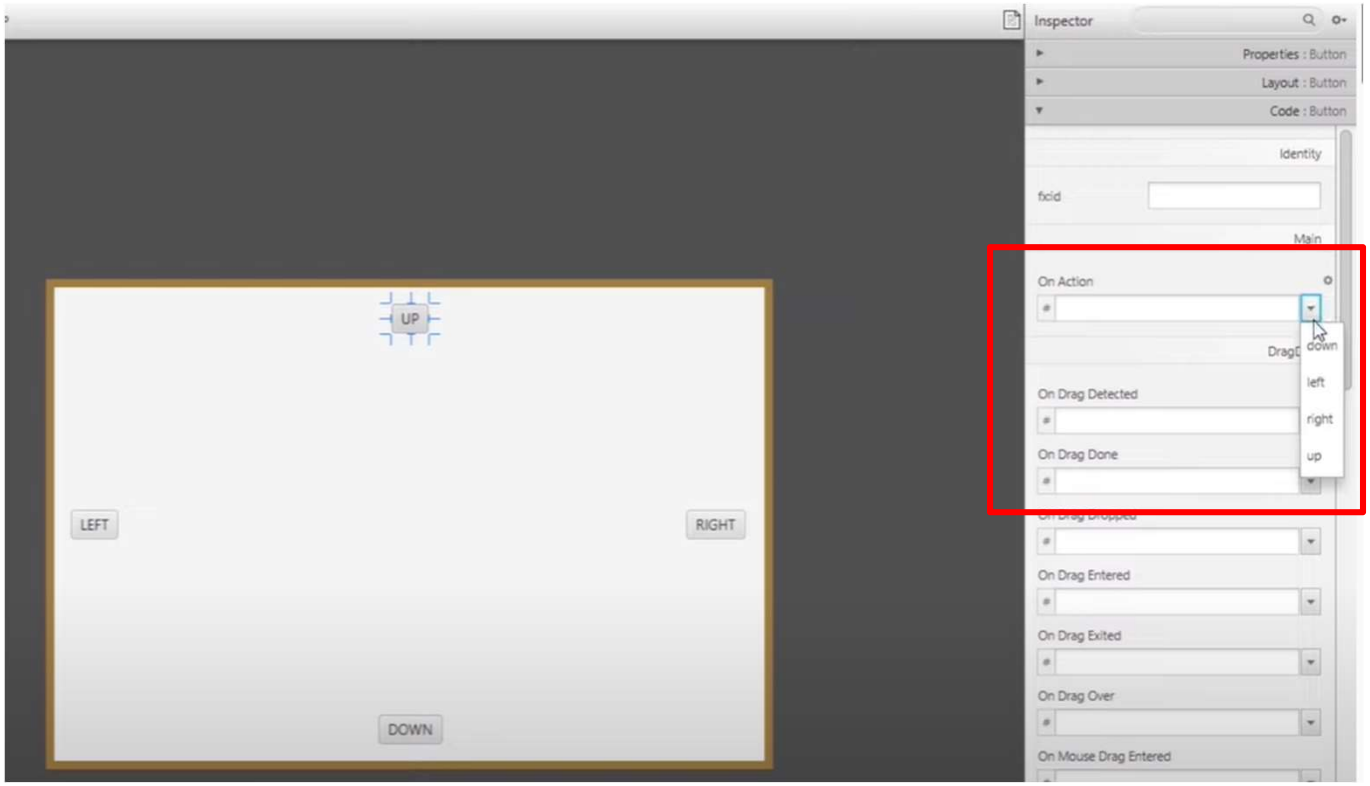




Write functions for each button that will have ActionEvent as the parameter
Make sure to import ActionEvent from javafx

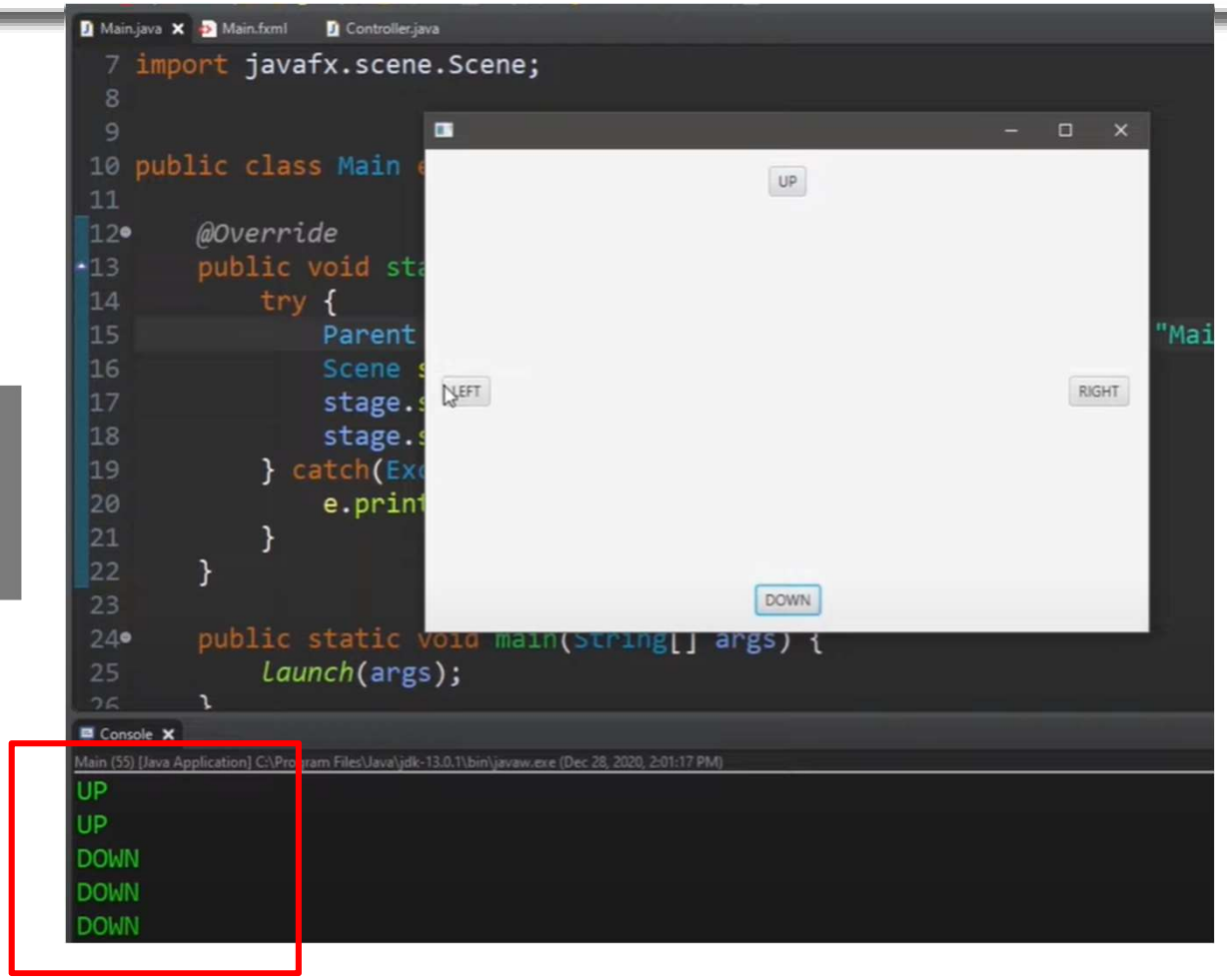
Go back to the SceneBuilder and select **Controller Class** > Select your controller class you just created



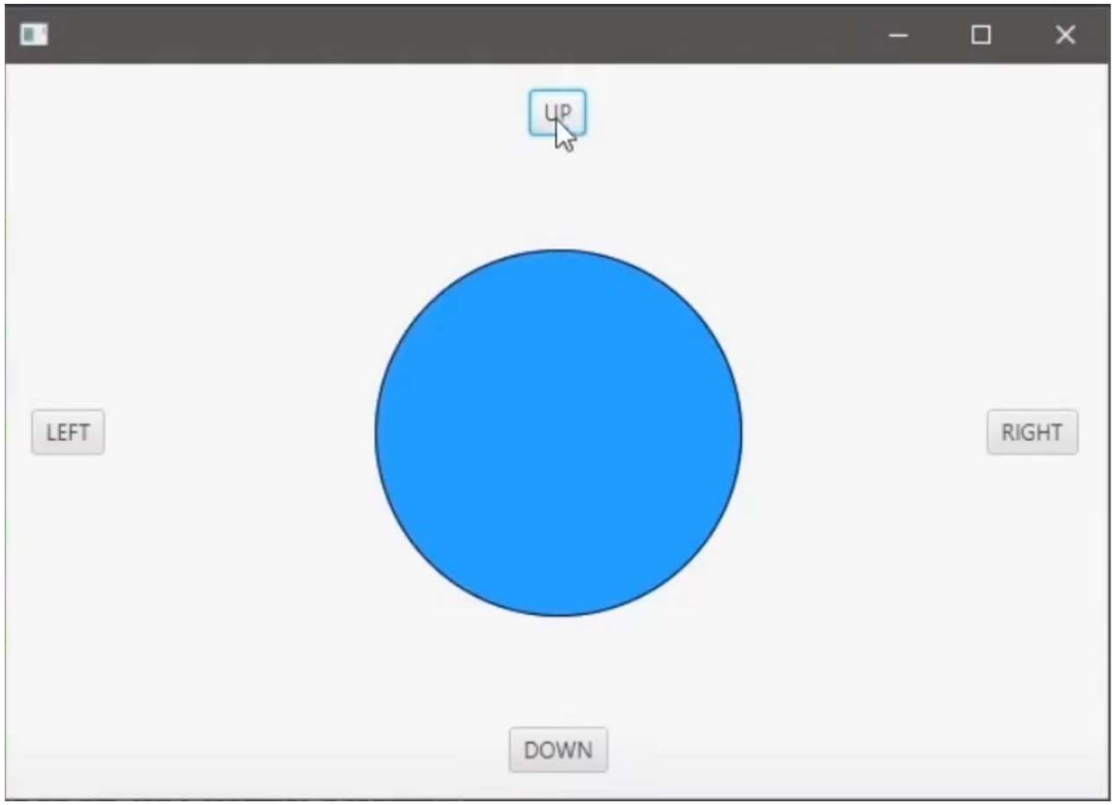


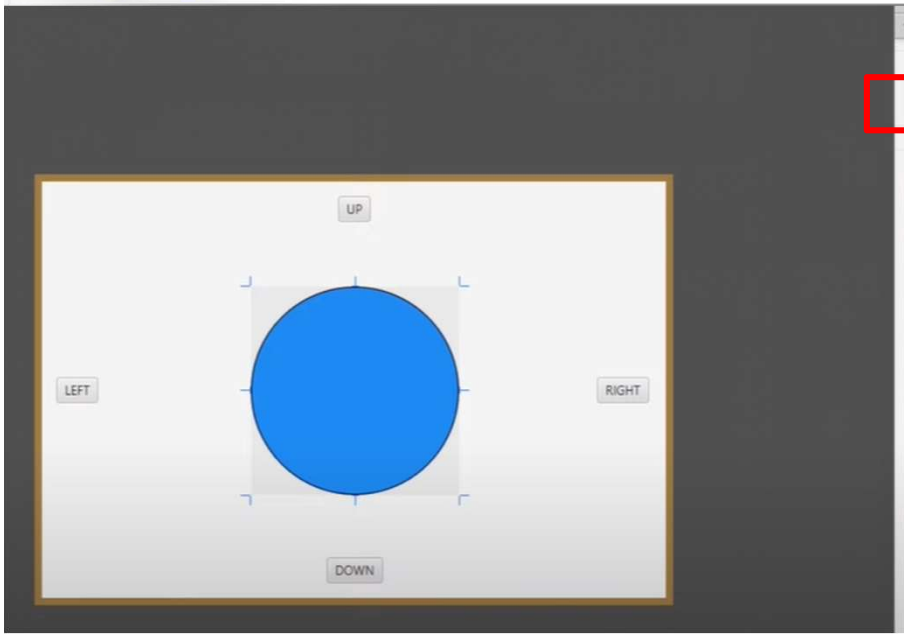
And select the corresponding function for **On Action**

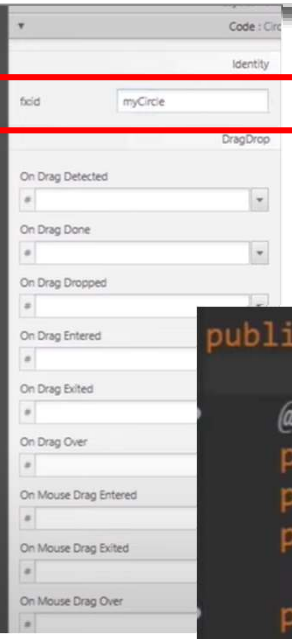
Now text should appear when you click on the buttons



Lab







Make a circle in SceneBuilder and name its fxid to “myCircle”

```
public class Controller {

    @FXML
    private Circle myCircle;
    private double x;
    private double y;

    public void up(ActionEvent e) {
        //System.out.println("UP");
        myCircle.setCenterY(y-=10);
    }
    public void down(ActionEvent e) {
        //System.out.println("DOWN");
        myCircle.setCenterY(y+=10);
    }
    public void left(ActionEvent e) {
        //System.out.println("LEFT");
        myCircle.setCenterX(x-=10);
    }
```

Your FXML file should look like this

Main.java | Main.fxml | *Controller.java

```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <?import javafx.scene.control.Button?>
4  <?import javafx.scene.layout.AnchorPane?>
5  <?import javafx.scene.shape.Circle?>
6
7  <AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" pref
8      <children>
9          <Button layoutX="285.0" layoutY="14.0" mnemonicParsing="false" onAction="#up" text="UP" />
10         <Button layoutX="534.0" layoutY="188.0" mnemonicParsing="false" onAction="#right" text="RIGHT" /
11         <Button layoutX="274.0" layoutY="361.0" mnemonicParsing="false" onAction="#down" text="DOWN" />
12         <Button layoutX="14.0" layoutY="188.0" mnemonicParsing="false" onAction="#left" text="LEFT" />
13         <Circle fx:id="myCircle" fill="DODGERBLUE" layoutX="301.0" layoutY="201.0" radius="100.0" stroke
14     </children>
15 </AnchorPane>
16
```

onAction = the function being fired when the button is on action

the fxid of the shape