

# Advanced Programming

## CMP 102

Dr. Perihan PEHLİVANOĞLU

Biruni Univarsity

Department of Computer Engineering

2023

Spring

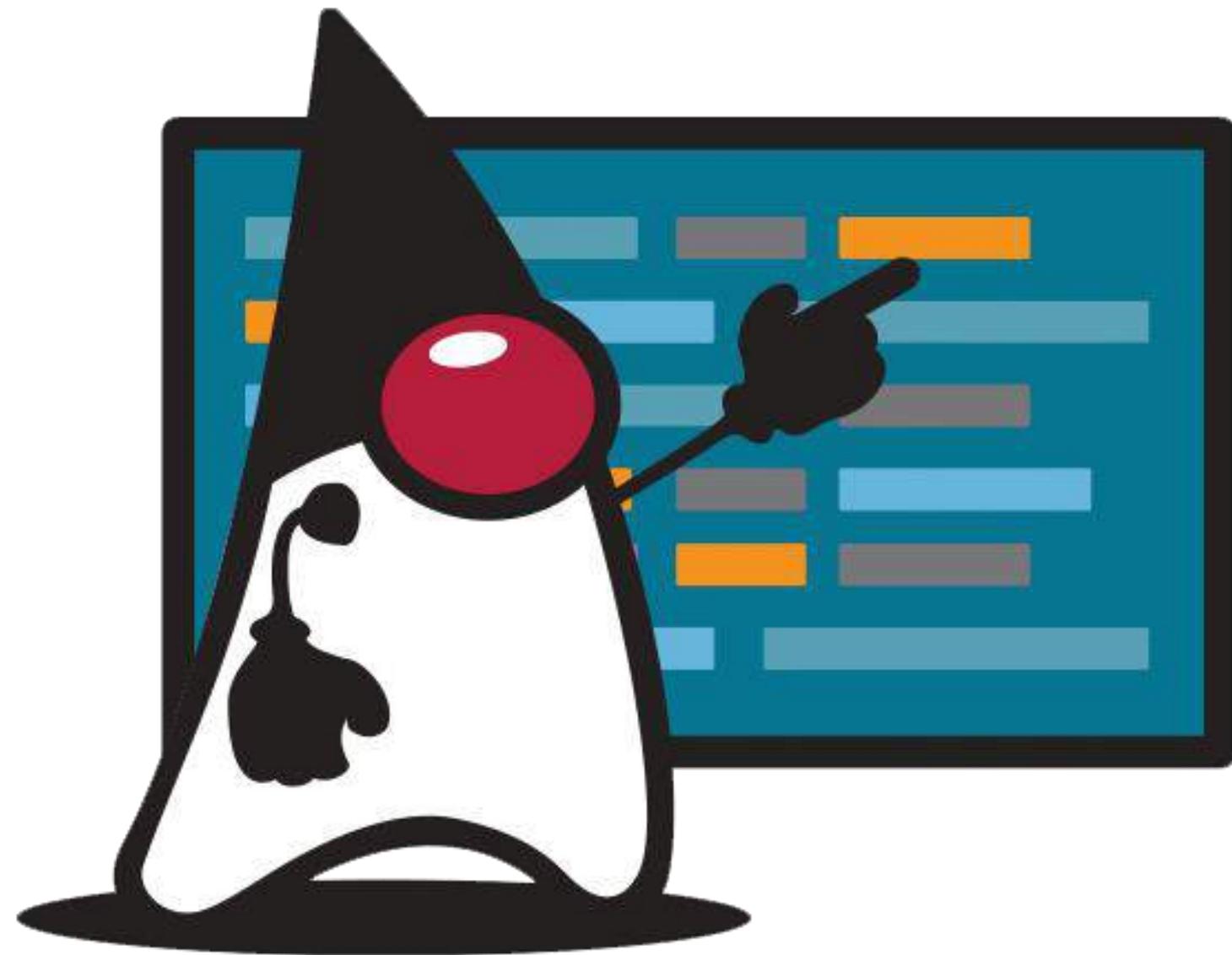
# Course Plan

- Introduction - Functions (Recursive functions)
- Arrays, ArrayList, Vector
- Collections (List, Queue, Stack, Set, Map, Tree)
- Streams, Files
- Generics and Exceptions
- Midterm Exam

# Course Plan

- Visual Programming Fundamentals
- Designing a Form
- Controls and Forms
- Properties of Components and Events
- Final Exam

# Last Lecture



- Midterm Exam.

# Introduction

- In this lecture, you will learn generics and visual programming fundamentals.

# GENERICS

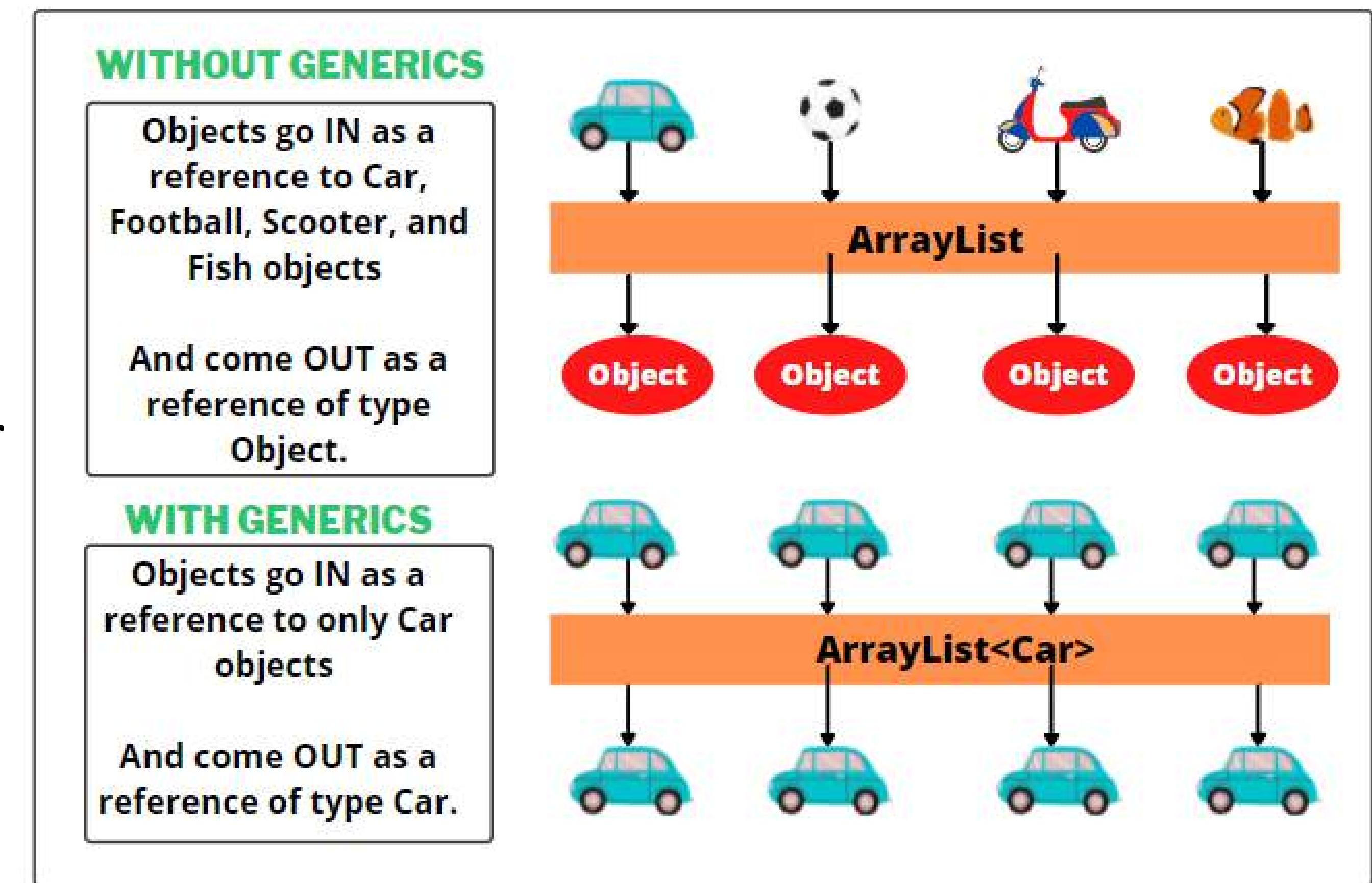
# Generics

- From generic class, you can create an ArrayList object for holding strings, and an ArrayList object for holding numbers.



# Generics

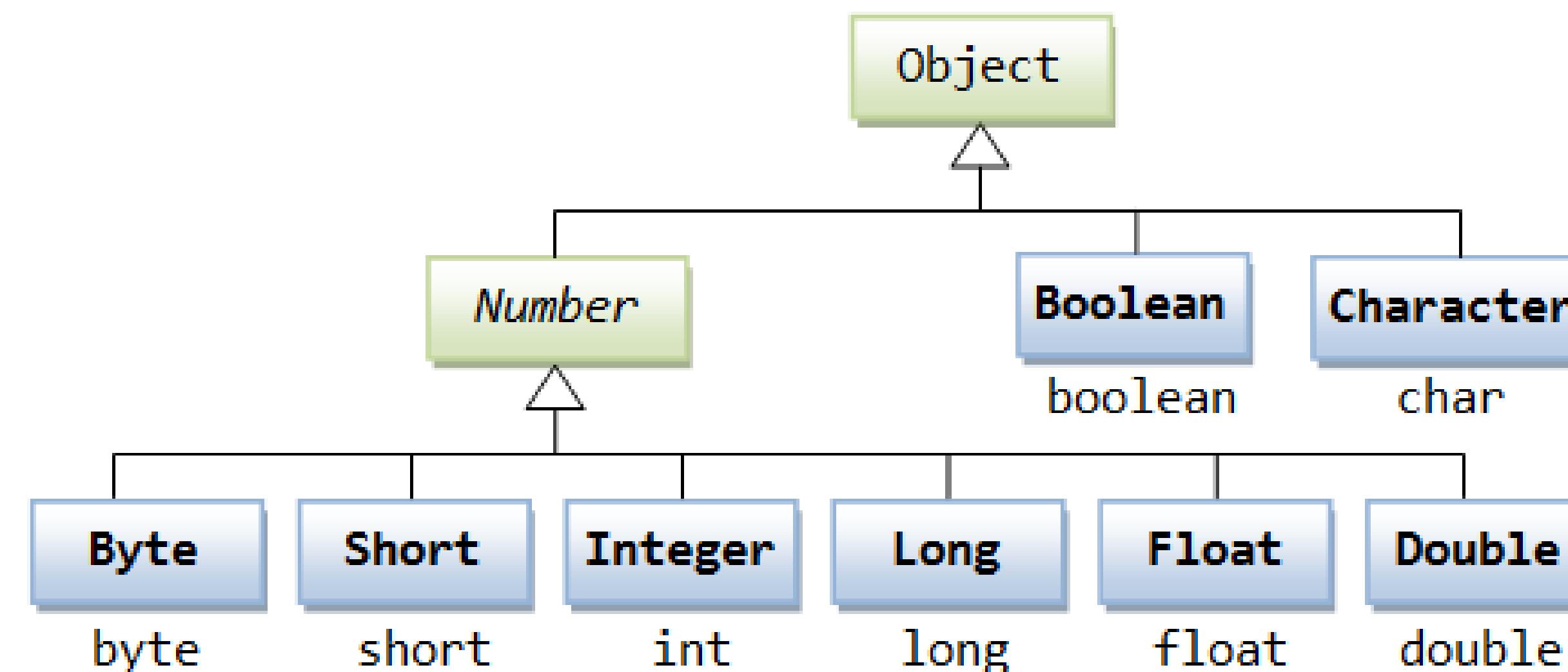
- The key benefit of generics is to enable errors to be detected at compile time rather than at runtime.
- A generic class or method permits you to specify allowable types of objects that the class or method can work with.
- If you attempt to use an incompatible object, the compiler will detect that error.



# Generics

- Abbreviations:

- E - Element (used extensively by the Java Collections Framework)
- K - Key
- N - Number
- T - Type
- V - Value
- S,U,V etc. - 2nd, 3rd, 4th types



# Generics

- Generic types have been added to the Java language to reduce code duplication. We can specify the data types of the classes, methods or interfaces we write with generic types at run time.
- For example, you have written a method and it takes a parameter. If you want this parameter to be of any data type, you can use generic types.
- This could be done with the object class before generic types, but it makes much more sense to do it with generic types because it requires extra operations such as type checking and type conversion.

# Generics

- Generic types have many advantages such as code simplicity, allowing us to develop code quickly and type safety.
- Generic types, when used with classes, can be used as follows:.
- The general definition can be class `ClassName<T,K,L,...>`. We can specify as many different types as we want, such as T, K, L, all of them represent the type we specify when creating the object.
- Let's recall the use of `ArrayList` for example.

# Generics

```
ArrayList<String> items = new ArrayList<String>();
```

- In between the above <> we make use of generics.

# Arrays and Overloaded Methods

```
public class OverloadedMethods {
    public static void main(String[] args) {
        // create arrays of Integer, Double and Character
        Integer[] integerArray = {1, 2, 3, 4, 5, 6};
        Double[] doubleArray = {1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7};
        Character[] characterArray = {'H', 'E', 'L', 'L', 'O'};

        System.out.printf("Array integerArray contains: ");
        printArray(integerArray); // pass an Integer array
        System.out.printf("Array doubleArray contains: ");
        printArray(doubleArray); // pass a Double array
        System.out.printf("Array characterArray contains: ");
        printArray(characterArray); // pass a Character array
    }

    // method printArray to print Integer array
    public static void printArray(Integer[] inputArray) {
        // display array elements
        for (Integer element : inputArray) {
            System.out.printf("%s ", element);
        }

        System.out.println();
    }

    // method printArray to print Double array
    public static void printArray(Double[] inputArray) {
        // display array elements
        for (Double element : inputArray) {
            System.out.printf("%s ", element);
        }

        System.out.println();
    }

    // method printArray to print Character array
    public static void printArray(Character[] inputArray) {
        // display array elements
        for (Character element : inputArray) {
            System.out.printf("%s ", element);
        }

        System.out.println();
    }
}
```

```
Array integerArray contains: 1 2 3 4 5 6
Array doubleArray contains: 1.1 2.2 3.3 4.4 5.5 6.6 7.7
Array characterArray contains: H E L L O
```

# Collections

```
import java.util.ArrayList;

public class ArrayListCollection {
    public static void main(String[] args) {
        // create a new ArrayList of Strings with an initial capacity of 10
        ArrayList<String> items = new ArrayList<String>();

        items.add("red"); // append an item to the list
        items.add(0, "yellow"); // insert "yellow" at index 0

        // header
        System.out.print(
            "Display list contents with counter-controlled loop:");

        // display the colors in the list
        for (int i = 0; i < items.size(); i++) {
            System.out.printf(" %s", items.get(i));
        }

        // display colors using enhanced for in the display method
        display(items,
            "%nDisplay list contents with enhanced for statement:");

        items.add("green"); // add "green" to the end of the list
        items.add("yellow"); // add "yellow" to the end of the list
        display(items, "List with two new elements:");

        items.remove("yellow"); // remove the first "yellow"
        display(items, "Remove first instance of yellow:");

        items.remove(1); // remove item at index 1
        display(items, "Remove second list element (green):");

        // check if a value is in the List
        System.out.printf("\\"red\\" is %sin the list%n",
            items.contains("red") ? "" : "not ");

        // display number of elements in the List
        System.out.printf("Size: %s%n", items.size());
    }

    // display the ArrayList's elements on the console
    public static void display(ArrayList<String> items, String header) {
        System.out.printf(header); // display header

        // display each element in items
        for (String item : items) {
            System.out.printf(" %s", item);
        }

        System.out.println();
    }
}
```

Display list contents with counter-controlled loop: yellow red  
Display list contents with enhanced for statement: yellow red  
List with two new elements: yellow red green yellow  
Remove first instance of yellow: red green yellow  
Remove second list element (green): red yellow  
"red" is in the list  
Size: 2

# Generics Methods

```
public class GenericMethodTest {  
    public static void main(String[] args) {  
        // create arrays of Integer, Double and Character  
        Integer[] integerArray = {1, 2, 3, 4, 5};  
        Double[] doubleArray = {1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7};  
        Character[] characterArray = {'H', 'E', 'L', 'L', 'O'};  
  
        System.out.printf("Array integerArray contains: ");  
        printArray(integerArray); // pass an Integer array  
        System.out.printf("Array doubleArray contains: ");  
        printArray(doubleArray); // pass a Double array  
        System.out.printf("Array characterArray contains: ");  
        printArray(characterArray); // pass a Character array  
    }  
  
    // generic method printArray  
    public static <T> void printArray(T[] inputArray) {  
        // display array elements  
        for (T element : inputArray) {  
            System.out.printf("%s ", element);  
        }  
  
        System.out.println();  
    }  
}
```

```
Array integerArray contains: 1 2 3 4 5  
Array doubleArray contains: 1.1 2.2 3.3 4.4 5.5 6.6 7.7  
Array characterArray contains: H E L L O
```

Here, **<T>** represents a *formal generic type*, which can be replaced later with an *actual concrete type*. Replacing a generic type is called a *generic instantiation*. By convention, a single capital letter such as **E** or **T** is used to denote a formal generic type.

# Generics

- There are too many examples in open sources on the subject. You can review it.

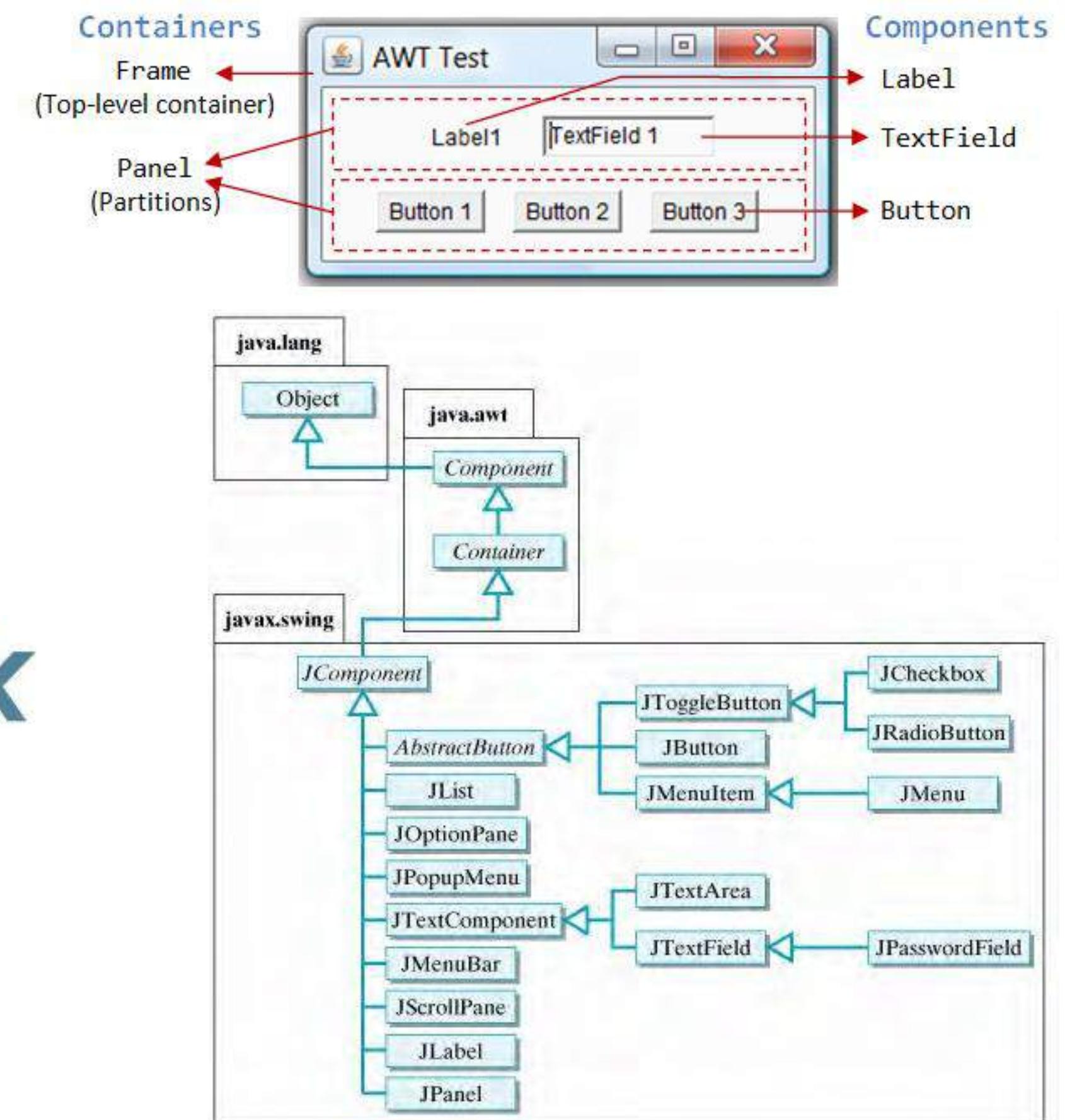
# **VISUAL PROGRAMMING**

# Visual Programming

- In this part, we will learn the basics of JavaFX for visual programming in Java.

# Visual Programming in Java

- Development of visuality in Java:
    - ✓ Abstract Windows Toolkit (AWT),
    - ✓ Swing,
    - ✓ JavaFX, after Java 8.



# Visual Programming in Java



# Visual Programming in Java



# Visual Programming in Java



# Visual Programming in Java



# Visual Programming in Java



# Visual Programming in Java

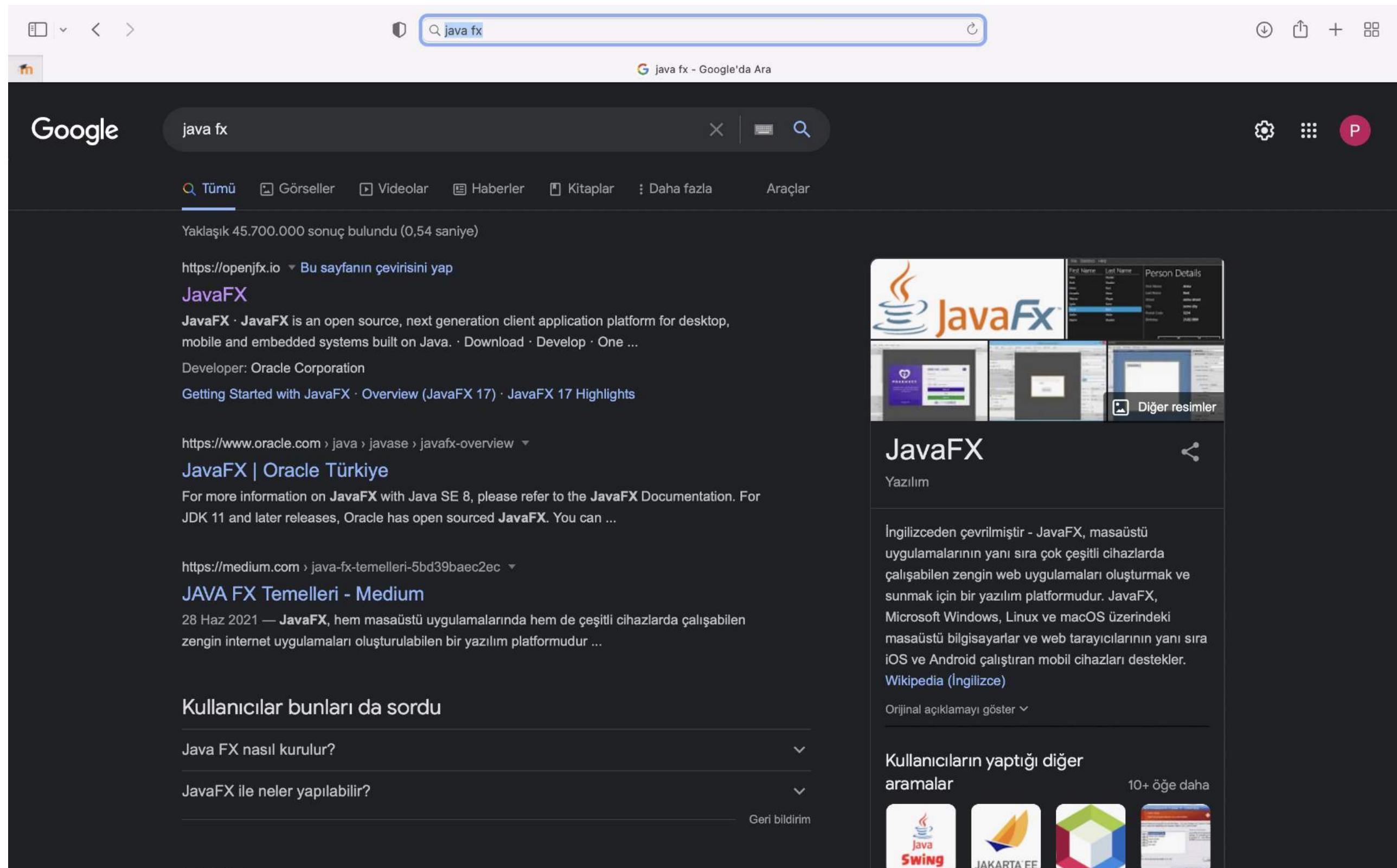


# JavaFX Program Structure

- In this lecture, you will learn the basics of JavaFX for visual programming in Java.
- Coding made easy with IntelliJ IDEA.
- Eclipse is a little more challenging.

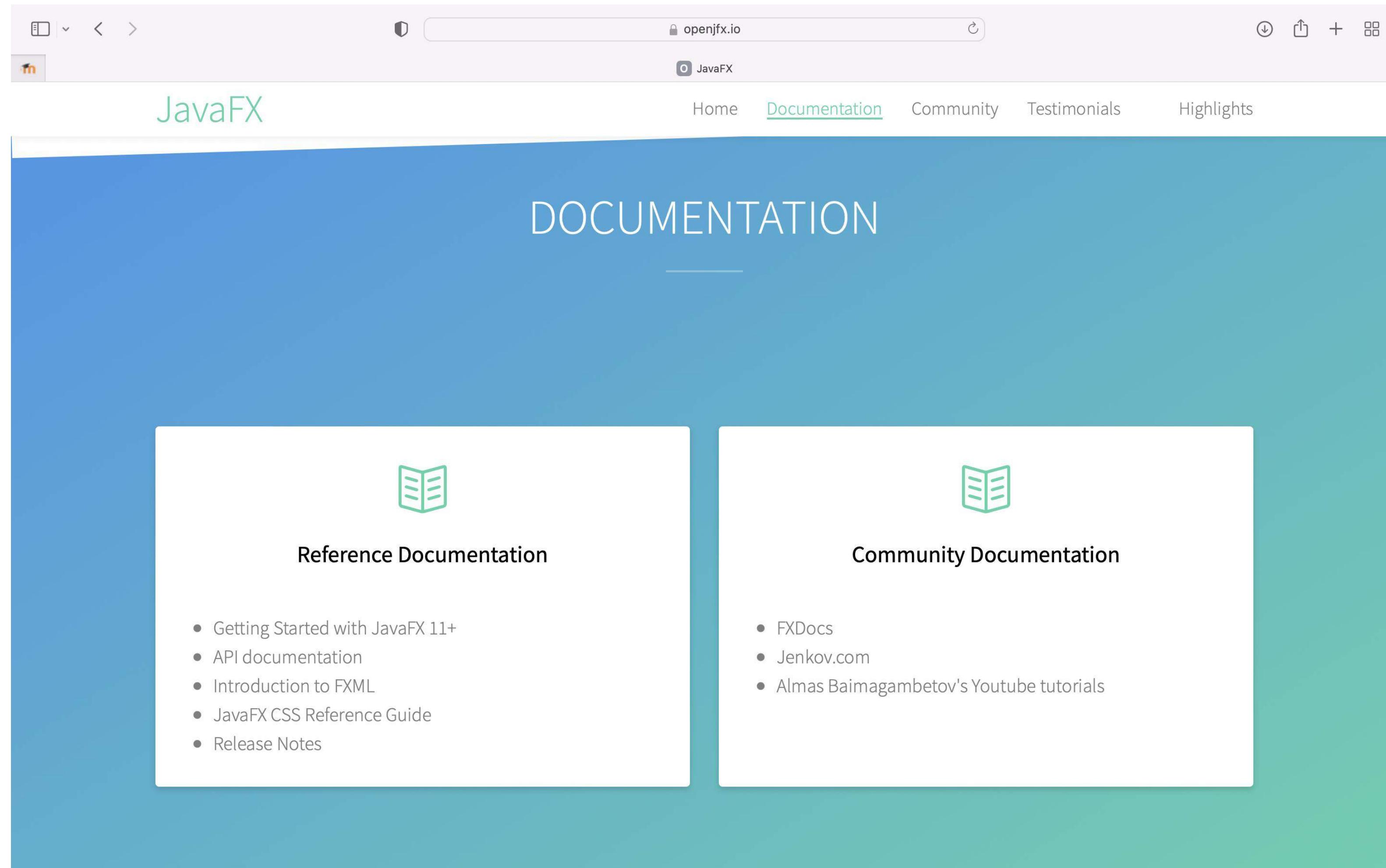
# Installing JavaFX

- Let's visit the JavaFX website.



# Installing JavaFX

- Let's visit the JavaFX website.



# **IntelliJ and Java FX**

# Installing JavaFX

- Let's visit the JavaFX website.



## Getting Started with JavaFX

Introduction

Install Java

Run HelloWorld using JavaFX

Run HelloWorld via Maven

Run HelloWorld via Gradle

Runtime images

JavaFX and IntelliJ

Non-modular from IDE

Non-modular with Maven

Non-modular with Gradle

Modular from IDE

Modular with Maven

Modular with Gradle

JavaFX and NetBeans

JavaFX and Eclipse

JavaFX and Visual Studio Code

Next Steps

### JavaFX and IntelliJ IDEA

This section explains how to create a JavaFX application in IntelliJ IDEA. JavaFX 15.0.1 and IntelliJ IDEA 2020.1 were used for the IDE screenshots.

Download an appropriate JDK for your operating system and set `JAVA_HOME` to the JDK directory. Refer to [Install Java](#) section for more information.

You can create a JavaFX modular or non-modular project and use the IDE tools, Maven or Gradle build tools.

#### Non-modular projects

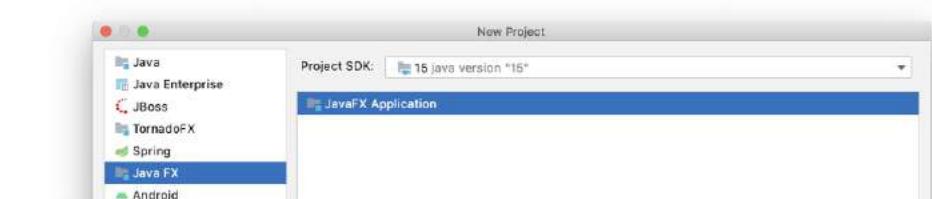
##### IDE

Follow these steps to create a JavaFX non-modular project and use the IDE tools to build it and run it.

Alternatively, you can download a similar project from [here](#).

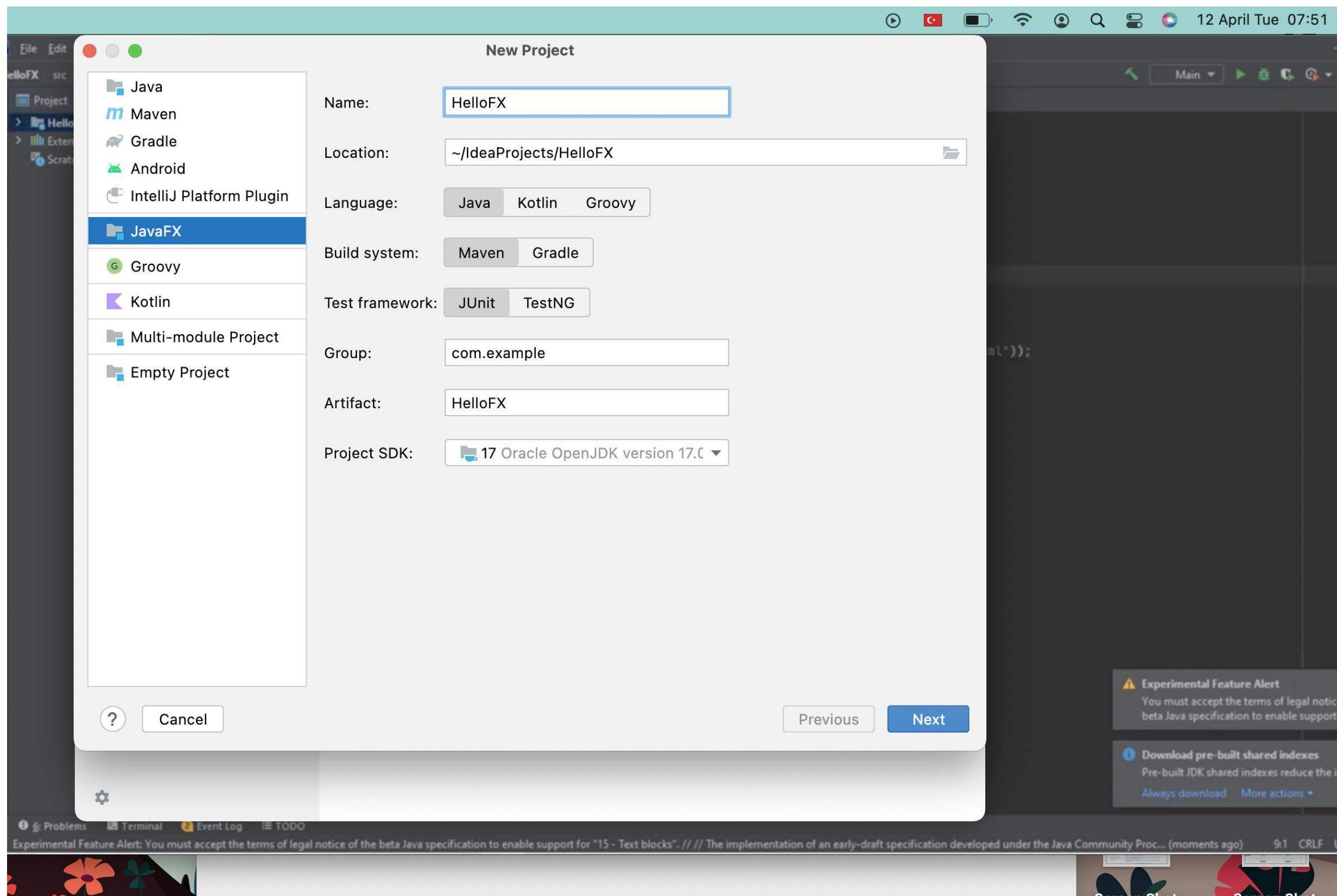
Download the appropriate [JavaFX SDK](#) for your operating system and unzip it to a desired location, for instance `/Users/your-user/Downloads/javafx-sdk-20`.

##### 1. Create a JavaFX project



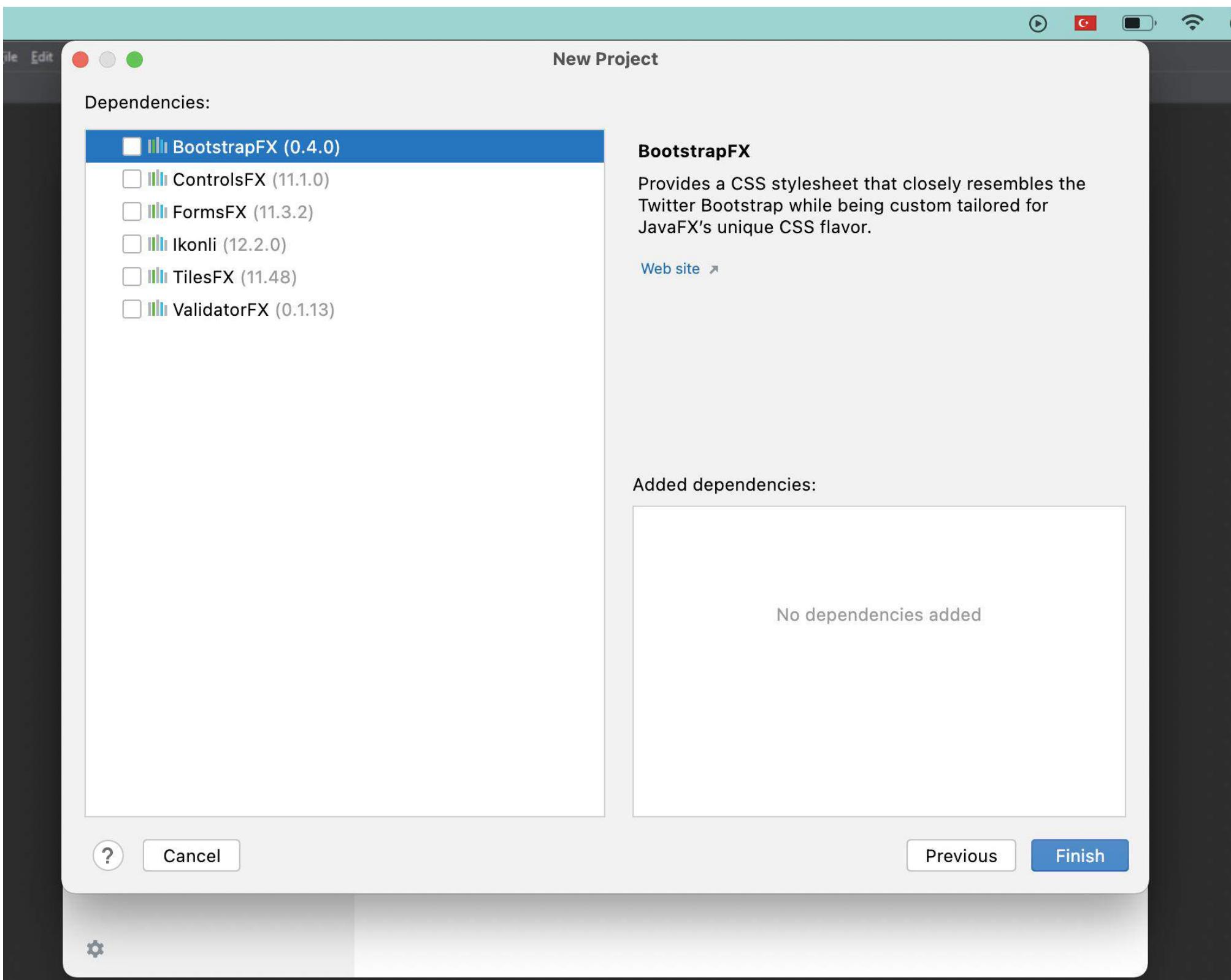
# Installing JavaFX

- With IntelliJ IDEA create a new Project.



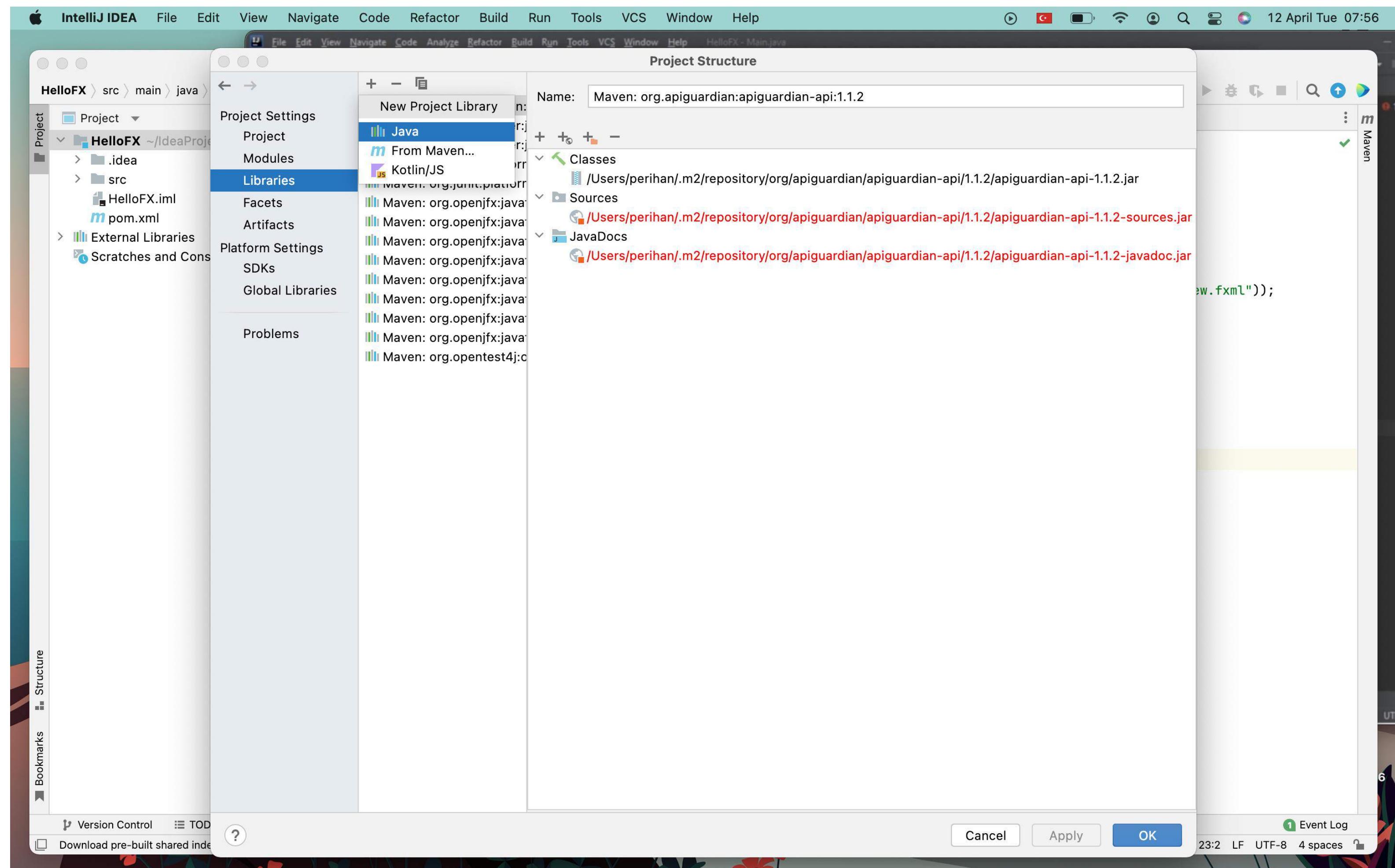
# Installing JavaFX

- IntelliJ IDEA



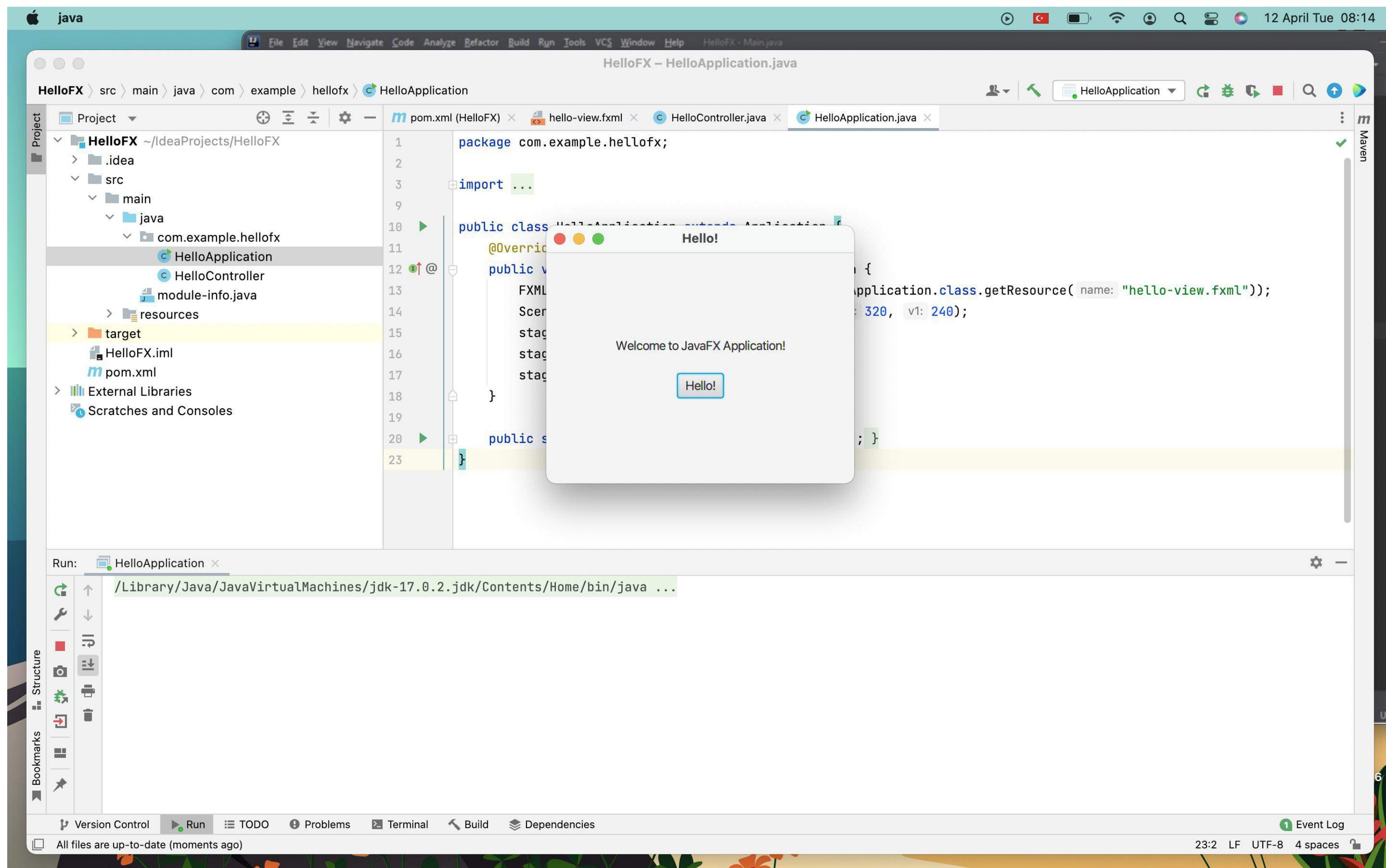
# Installing JavaFX

- Let's add libraries to IntelliJ IDEA.



# Installing JavaFX

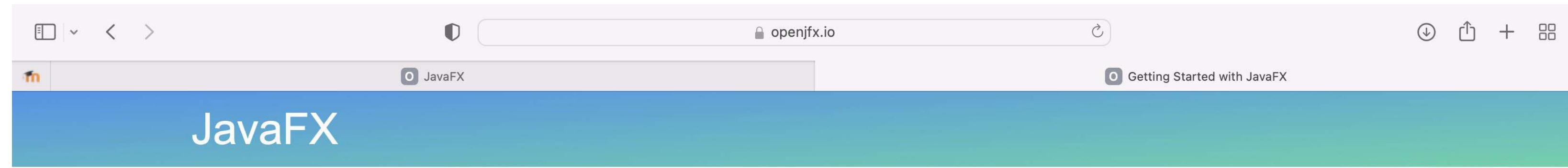
- Example.



# **IntelliJ and Eclipse**

# Installing JavaFX

- Let's visit the JavaFX website.



## Getting Started with JavaFX

Introduction

Install Java

Run HelloWorld using JavaFX

Run HelloWorld via Maven

Run HelloWorld via Gradle

Runtime images

JavaFX and IntelliJ

JavaFX and NetBeans

**JavaFX and Eclipse**

Non-modular from IDE

Non-modular with Maven

Non-modular with Gradle

Modular from IDE

Modular with Maven

Modular with Gradle

JavaFX and Visual Studio Code

Next Steps

### JavaFX and Eclipse

This section explains how to create a JavaFX application in Eclipse. JavaFX 15.0.1 and Eclipse 2020-12 (4.18.0) were used for the IDE screenshots.

Download an appropriate JDK for your operating system and set `JAVA_HOME` to the JDK directory. Refer to [Install Java](#) section for more information.

Include the new JDK as `Installed JREs` in `Eclipse -> Preferences -> Java -> Installed JREs -> Add`.

You can create a JavaFX modular or non-modular project and use the IDE tools, Maven or Gradle build tools.

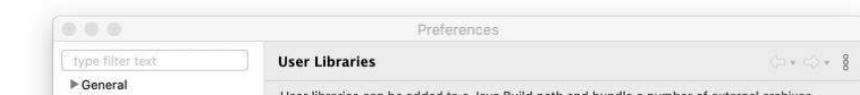
#### Non-modular projects

##### IDE

Follow these steps to create a JavaFX non-modular project and use the IDE tools to build it and run it. Alternatively, you can download a similar project from [here](#).

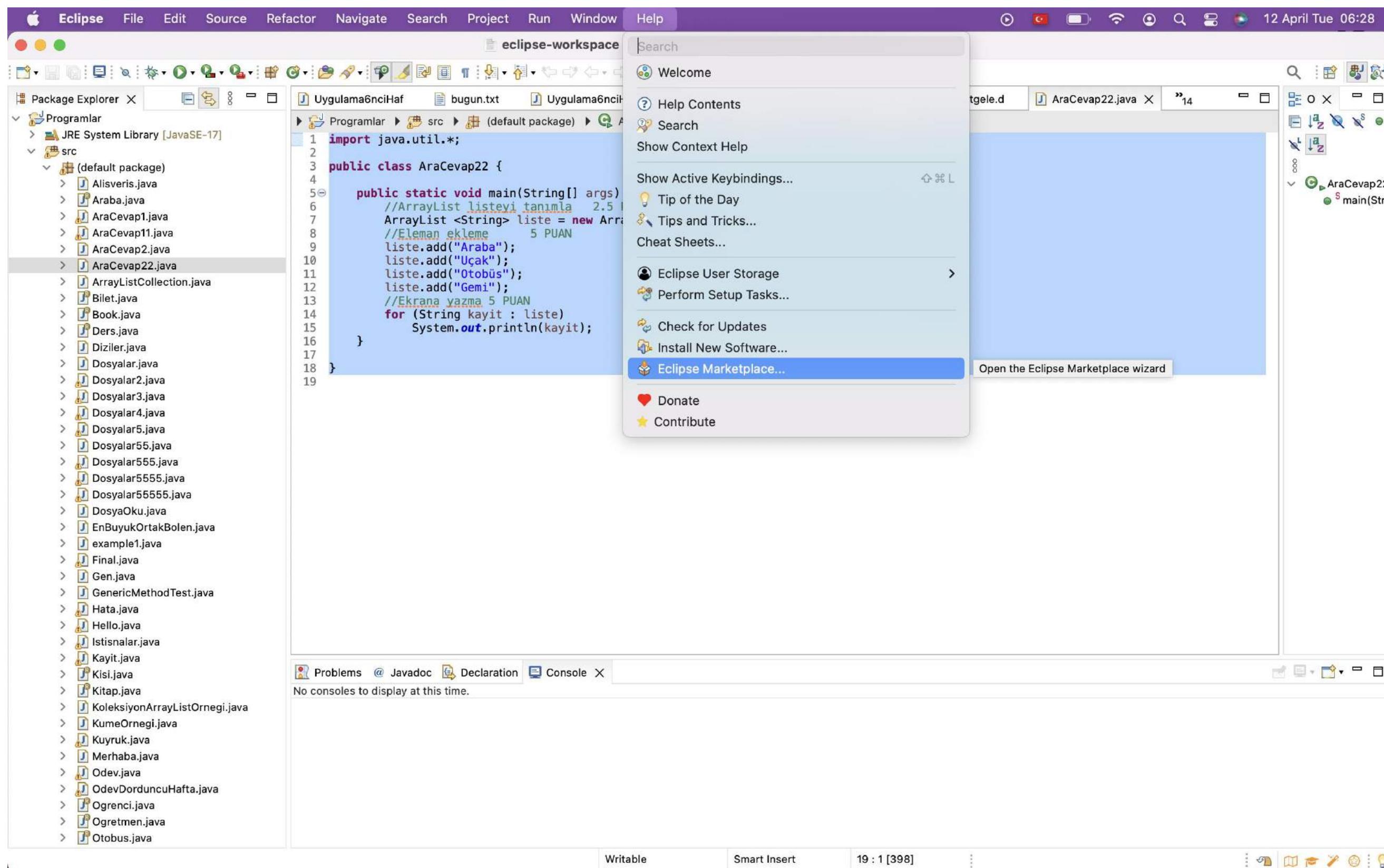
Download the appropriate [JavaFX SDK](#) for your operating system and unzip it to a desired location, for instance `/Users/your-user/Downloads/javafx-sdk-17.0.1`.

Create a new `User Library` under `Eclipse -> Window -> Preferences -> Java -> Build Path -> User Libraries -> New`.



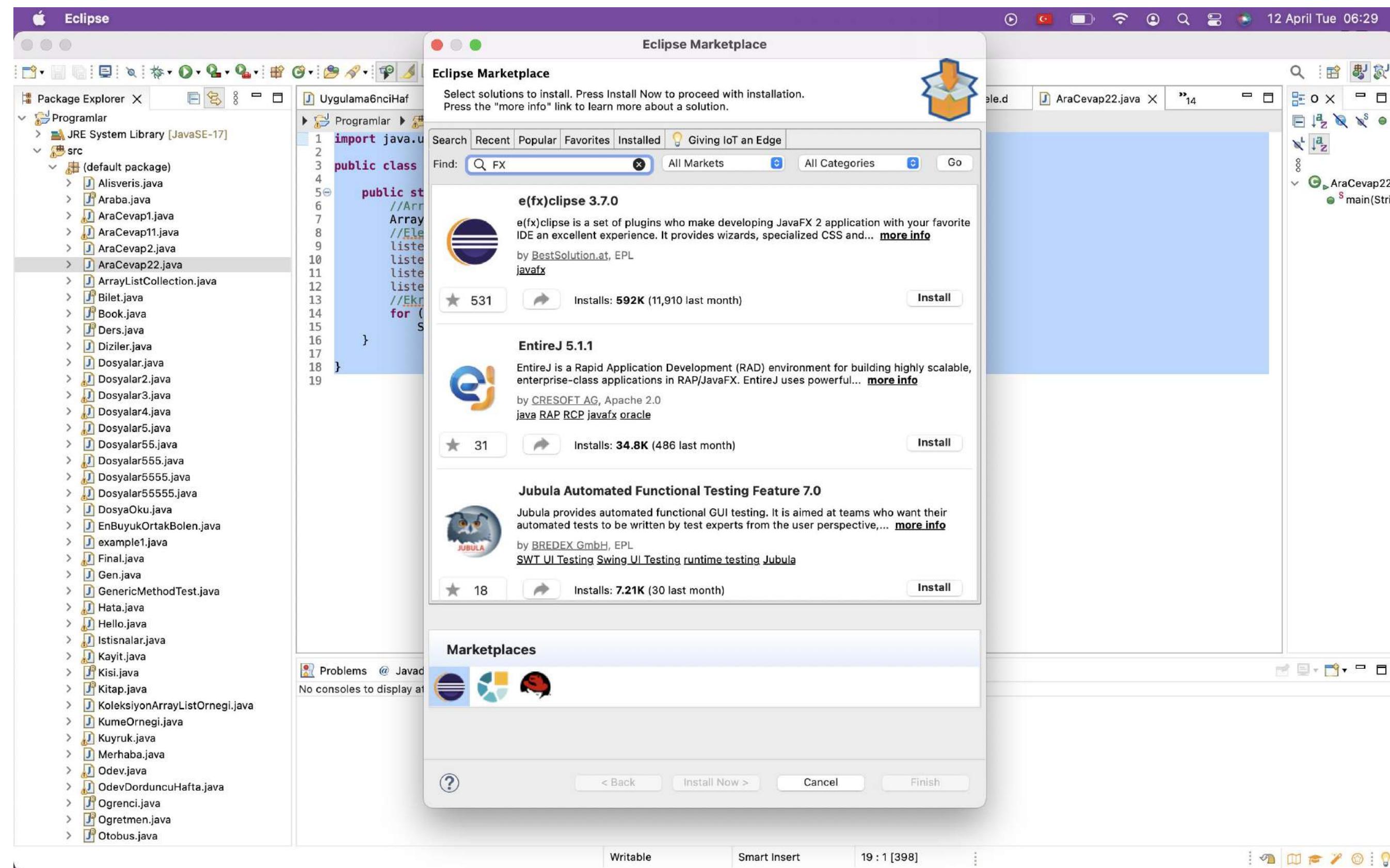
# Installing JavaFX

- Let's visit the JavaFX website.



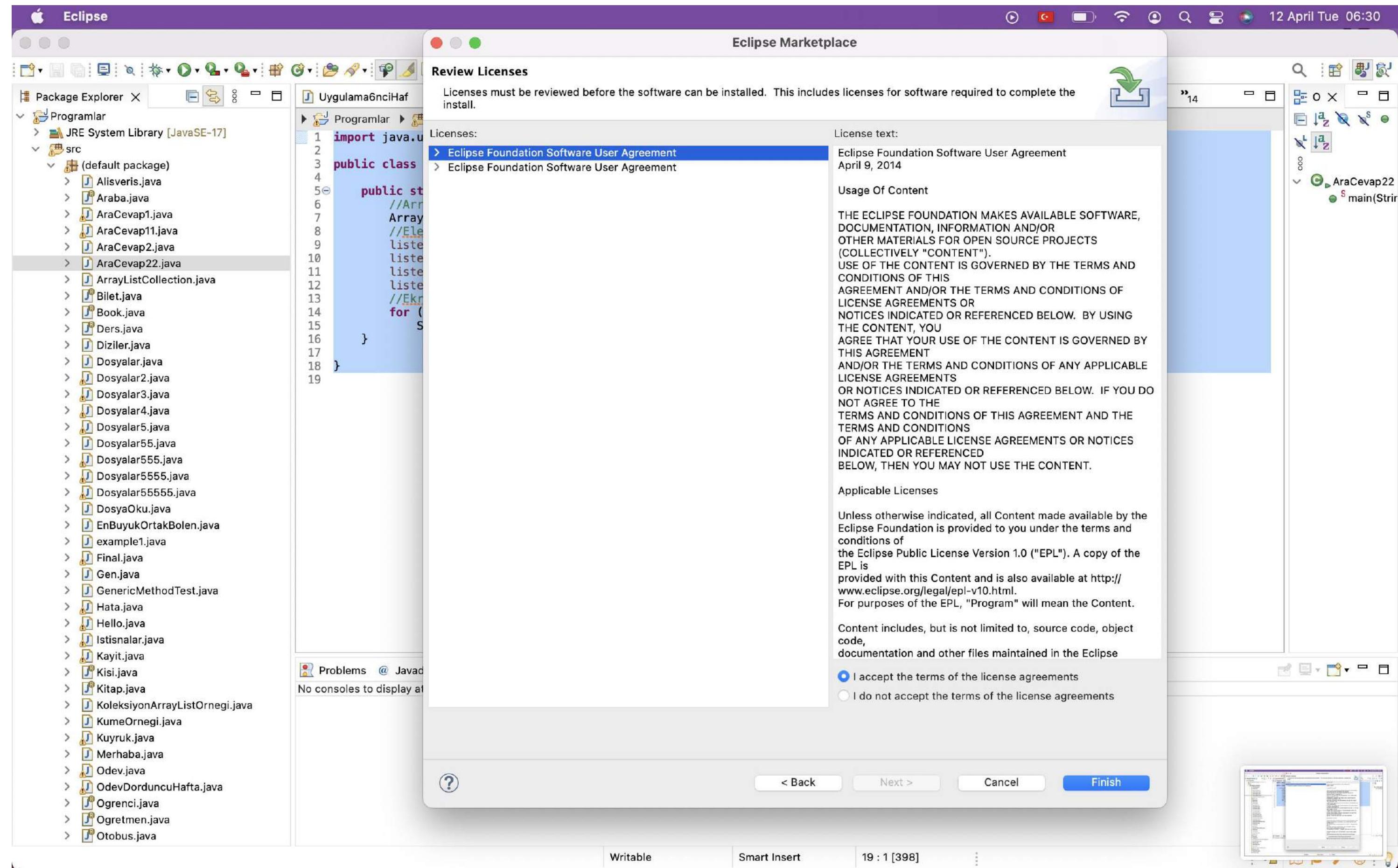
# Installing JavaFX

- Let's install the JavaFX plugin.



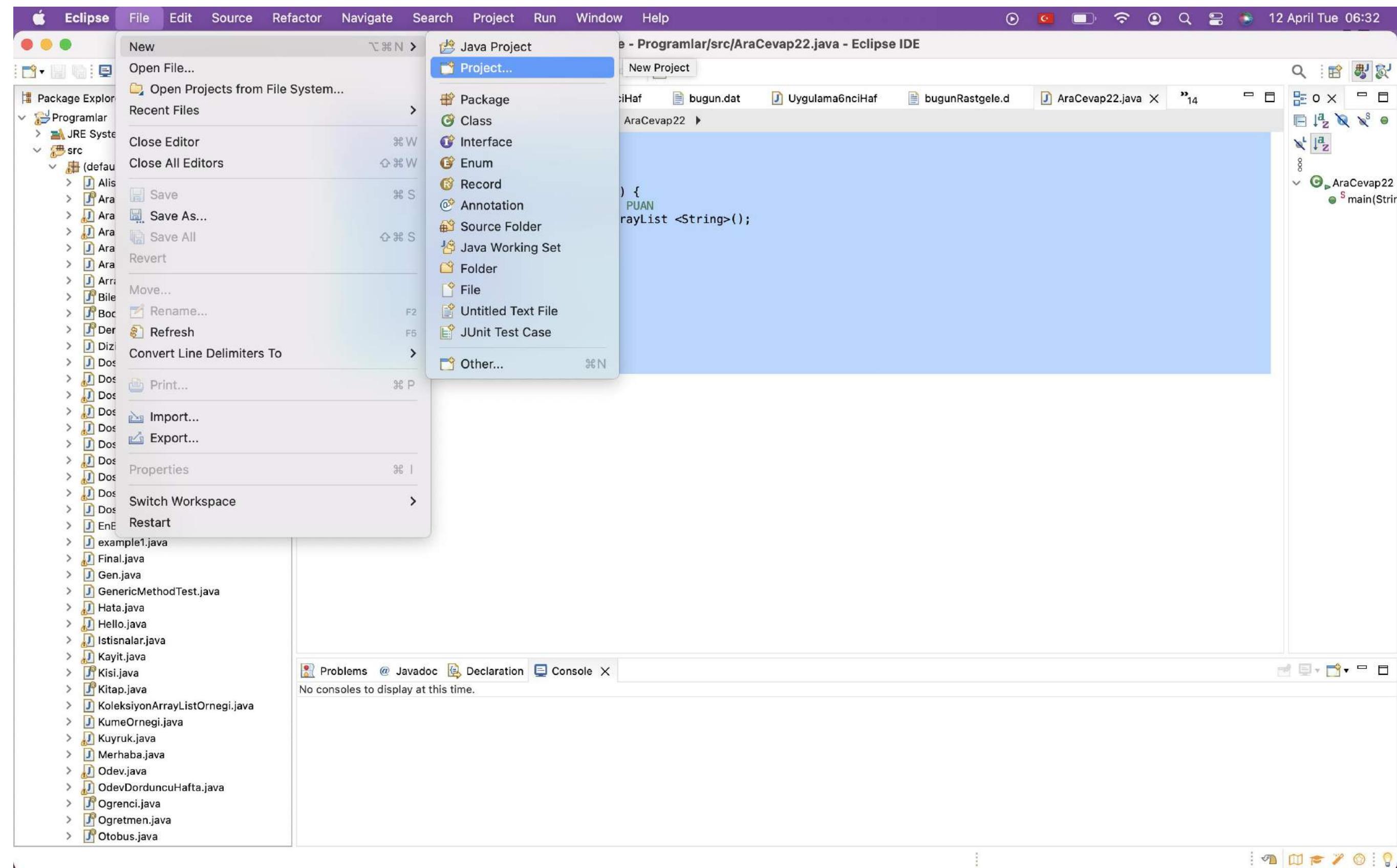
# Installing JavaFX

- Let's install the JavaFX plugin.



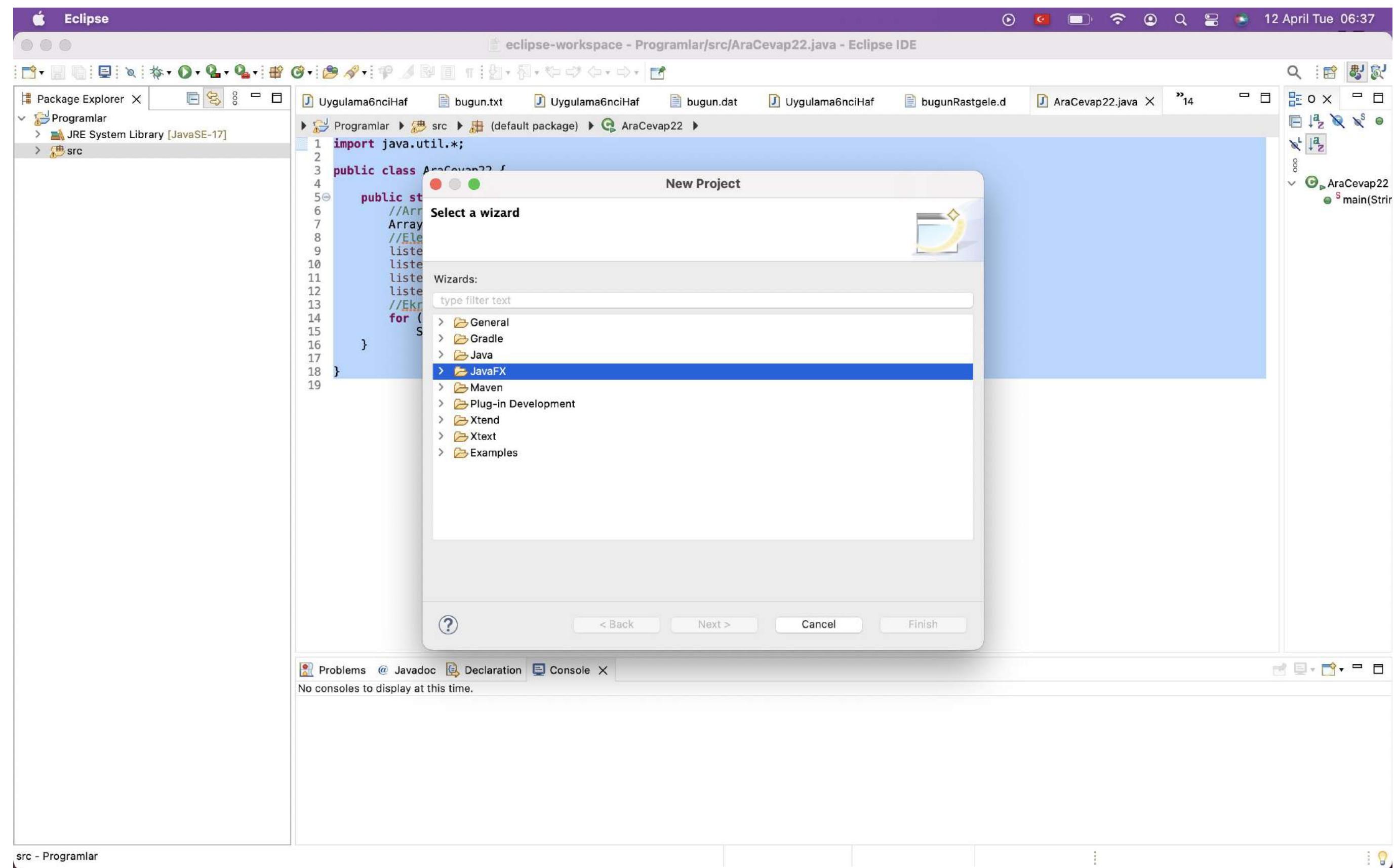
# Installing JavaFX

- Let's create a JavaFX project.



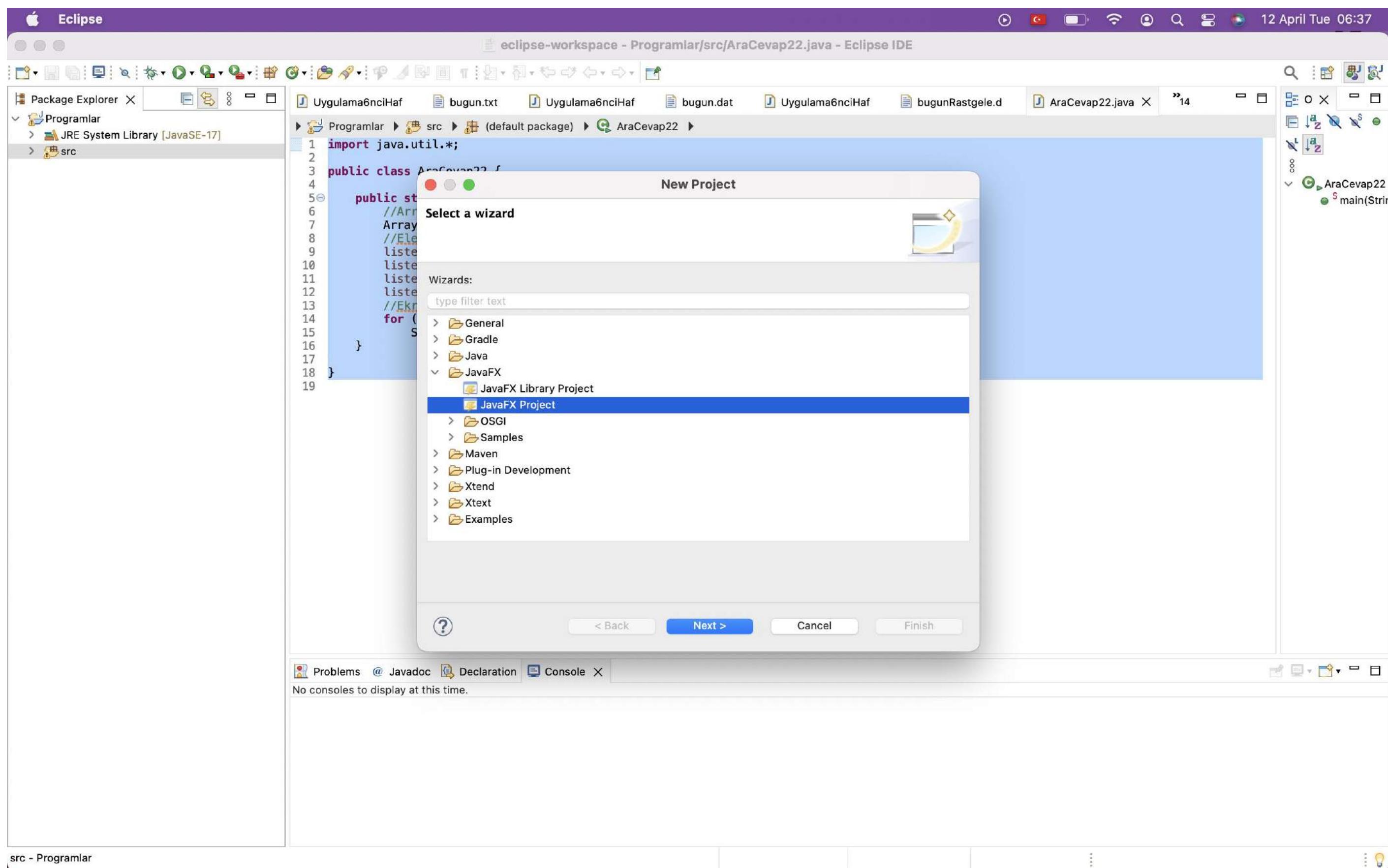
# Installing JavaFX

- Let's create a JavaFX project.



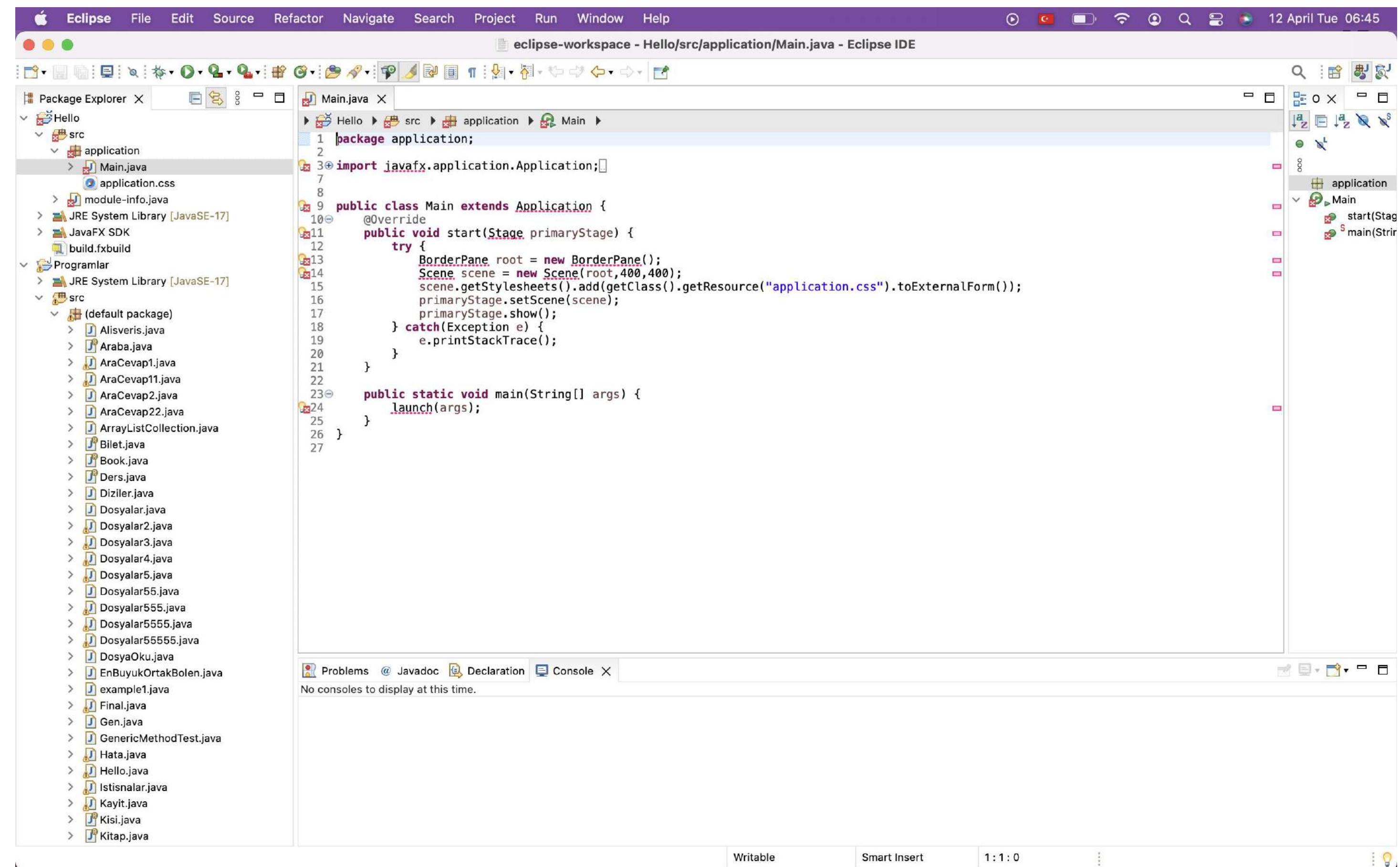
# Installing JavaFX

- Let's create a JavaFX project.



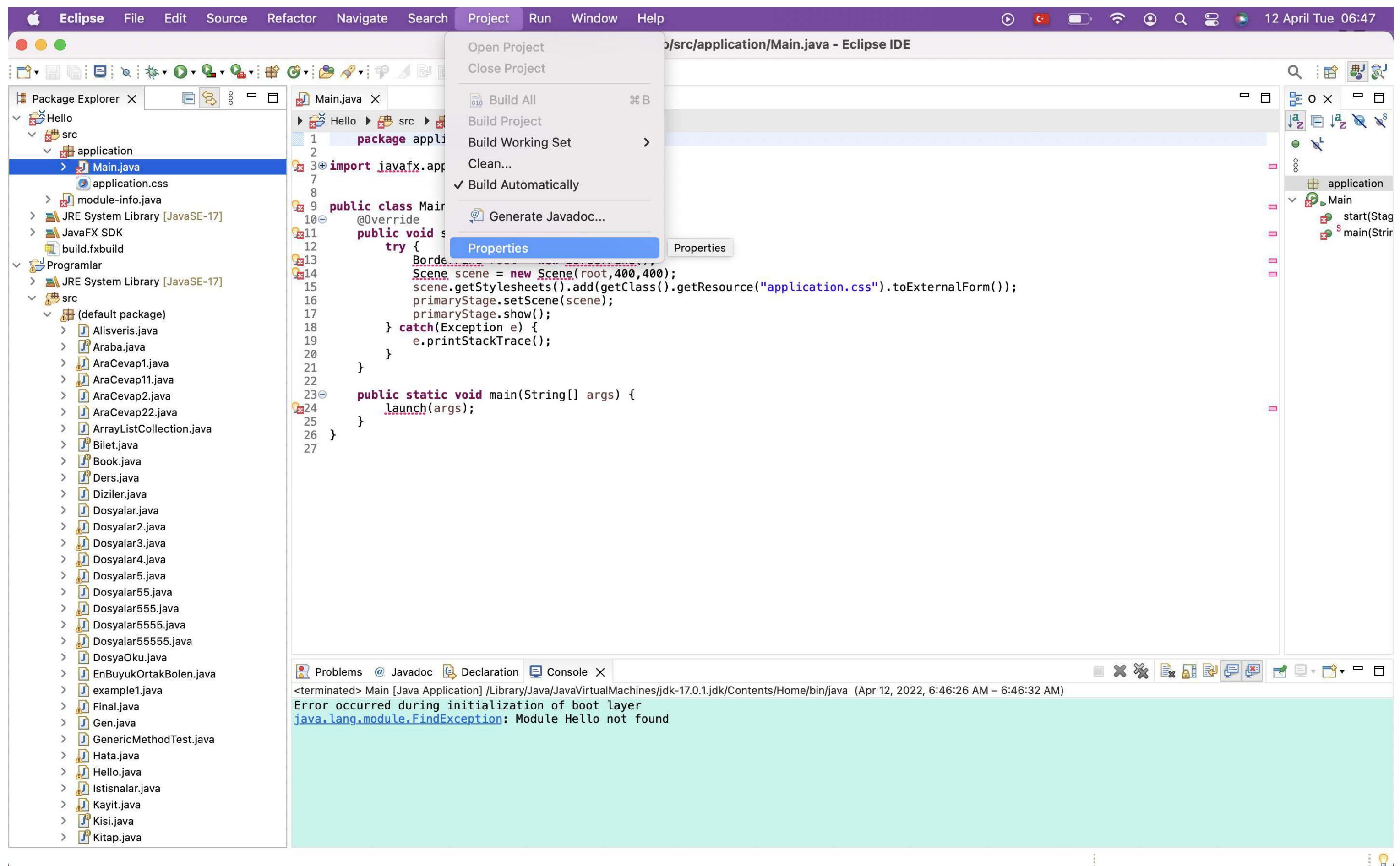
# Installing JavaFX

- Let's create a JavaFX project.



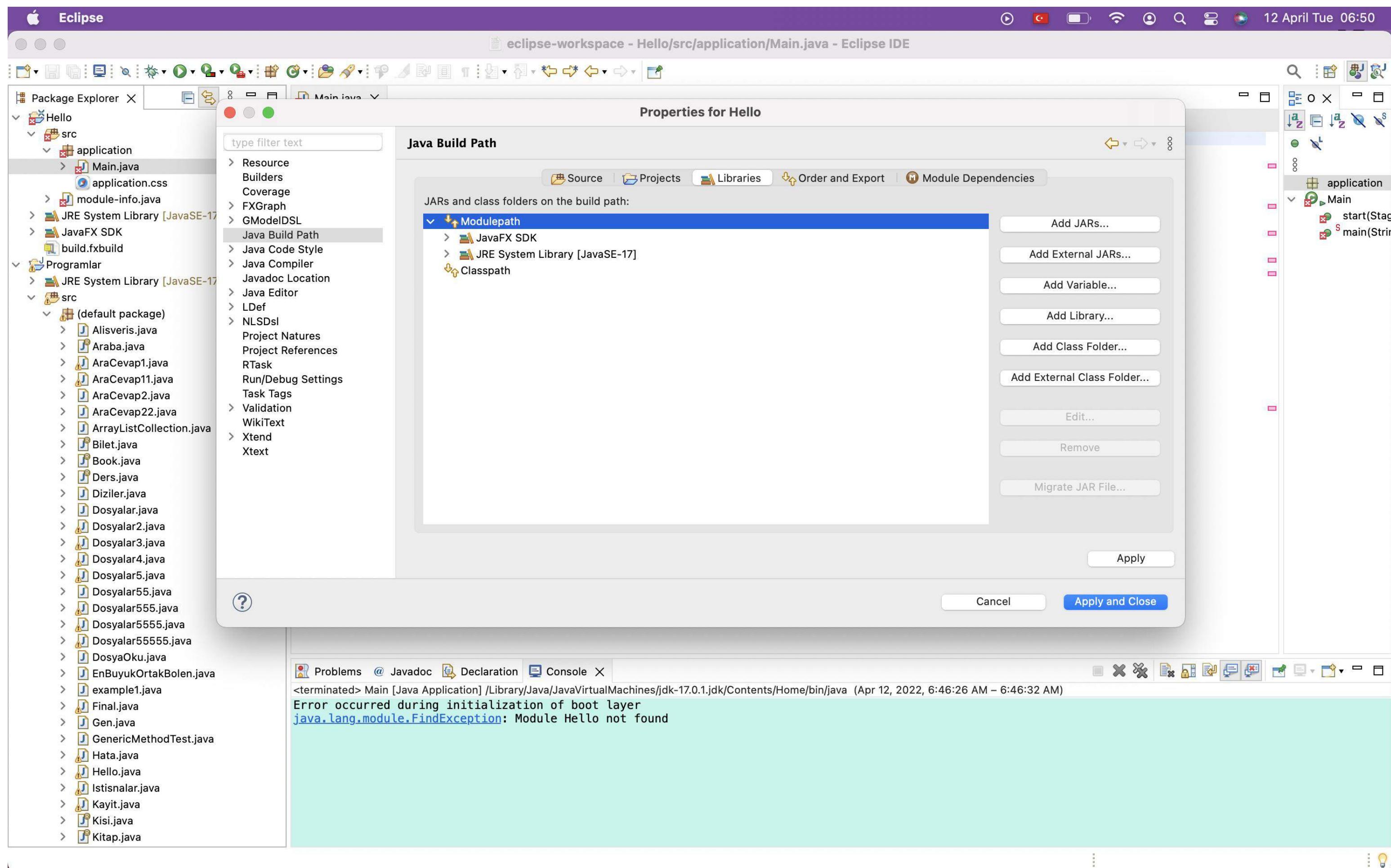
# Installing JavaFX

- Let's add libraries to the JavaFX project.



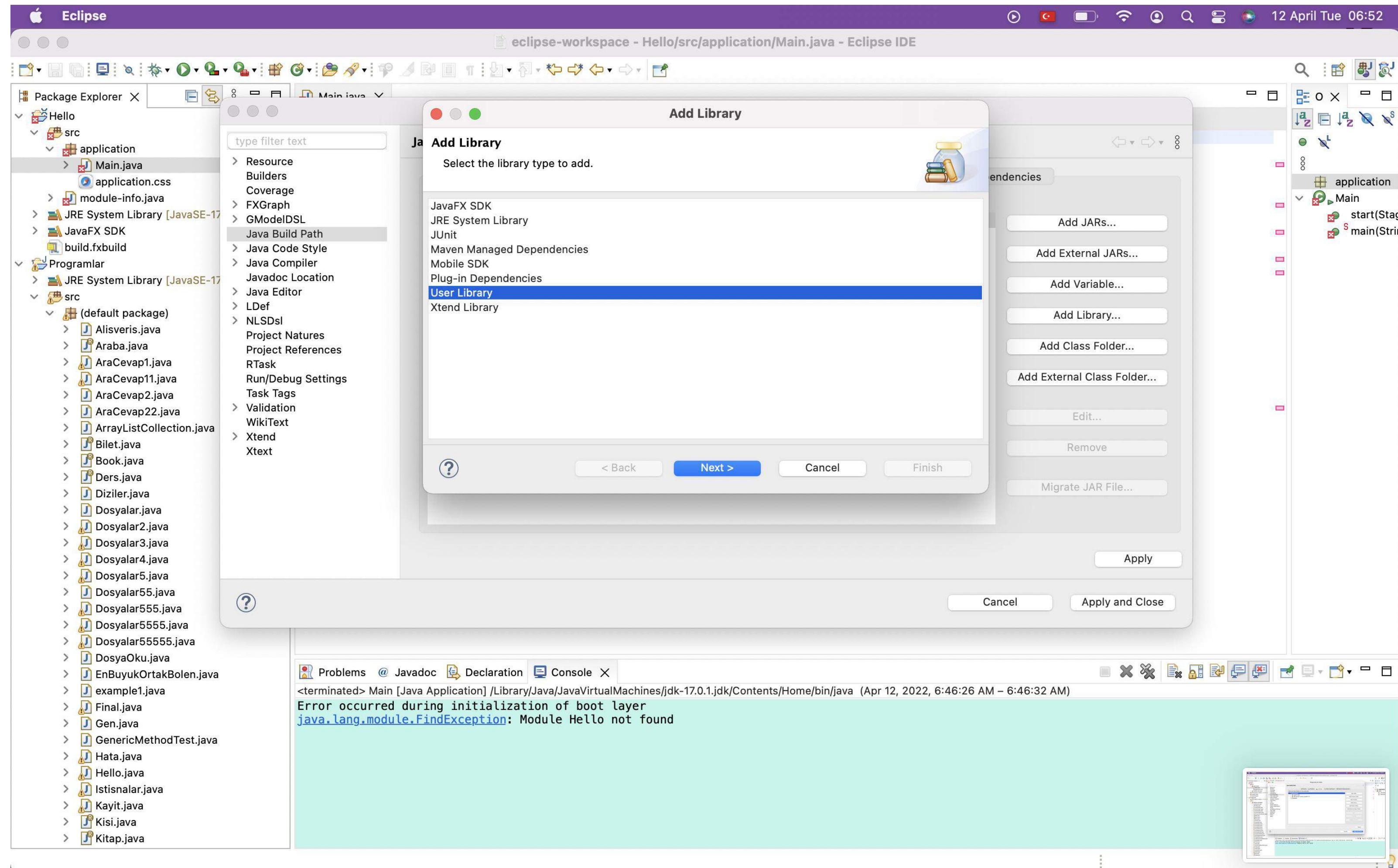
# Installing JavaFX

- Let's add libraries to the JavaFX project.



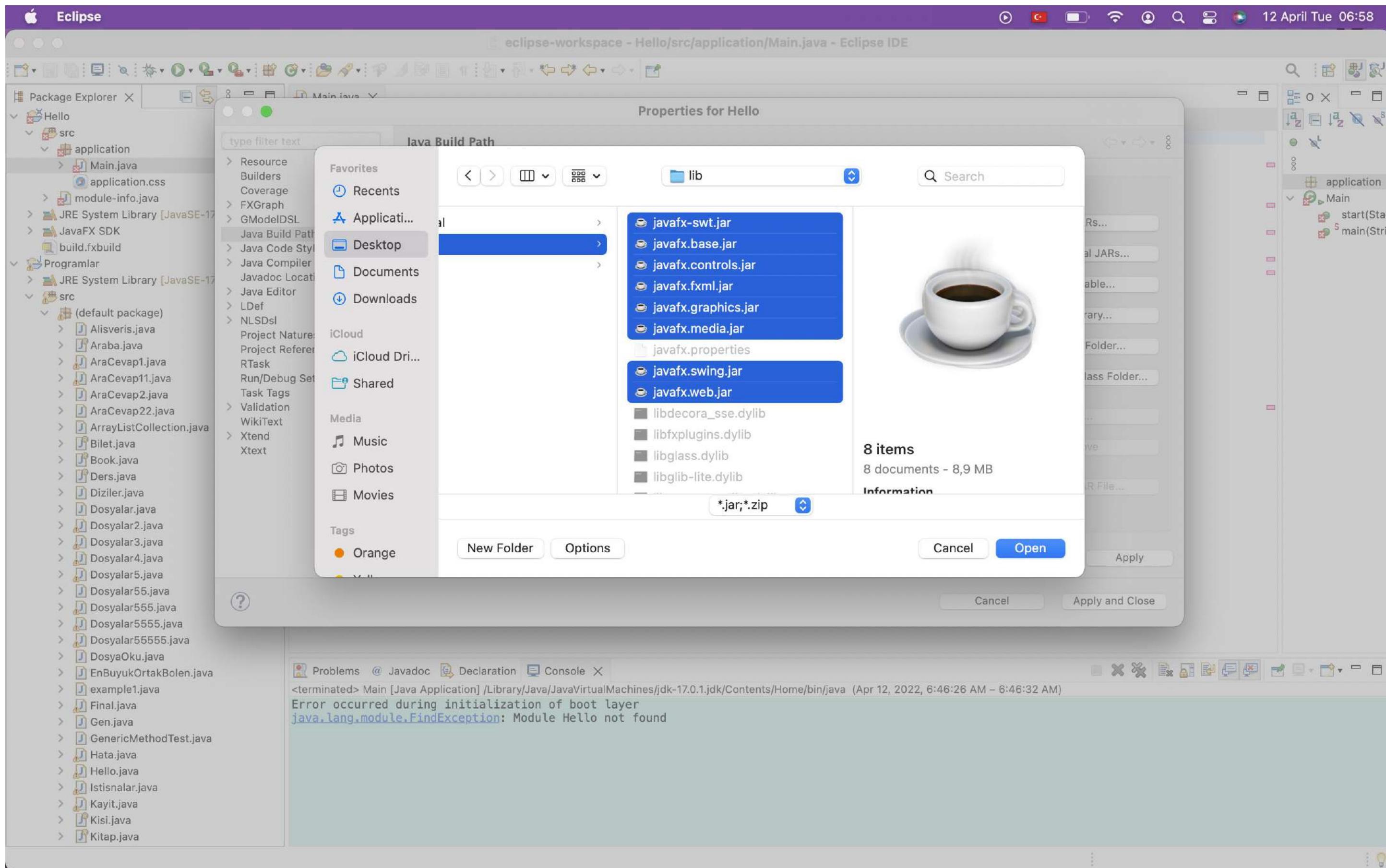
# Installing JavaFX

- Let's add libraries to the JavaFX project.



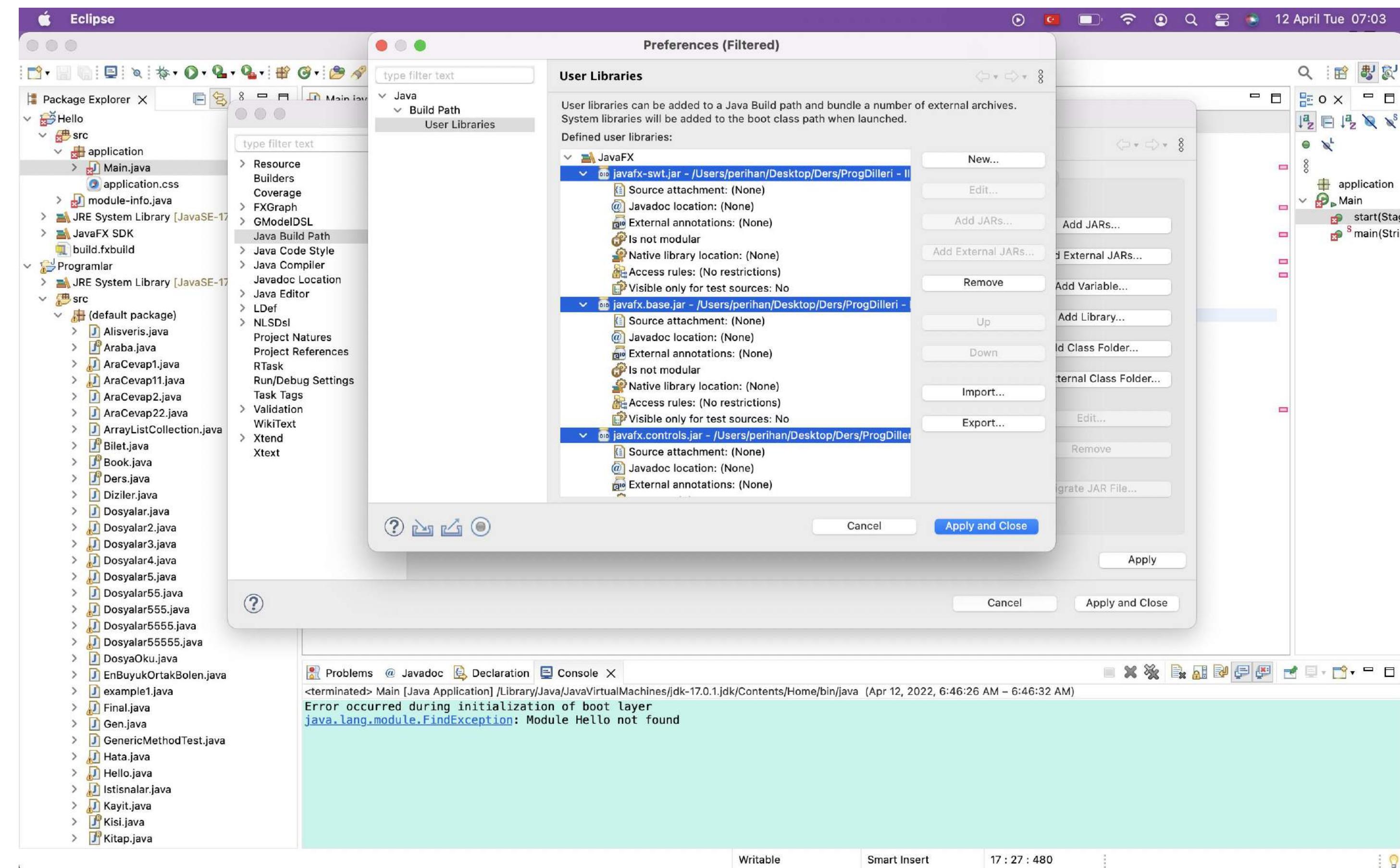
# Installing JavaFX

- Let's add libraries to the JavaFX project.



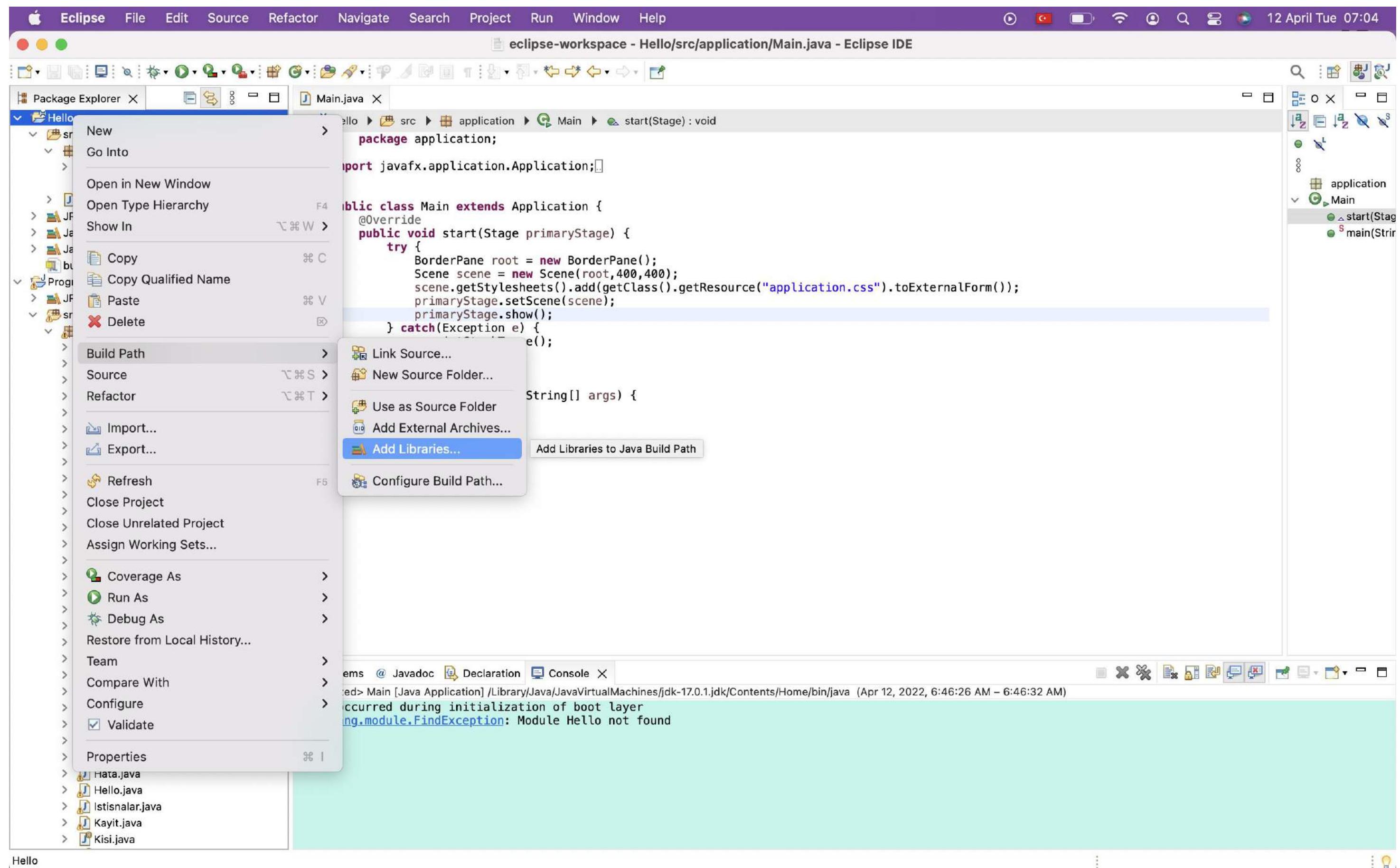
# Installing JavaFX

- Let's add libraries to the JavaFX project.



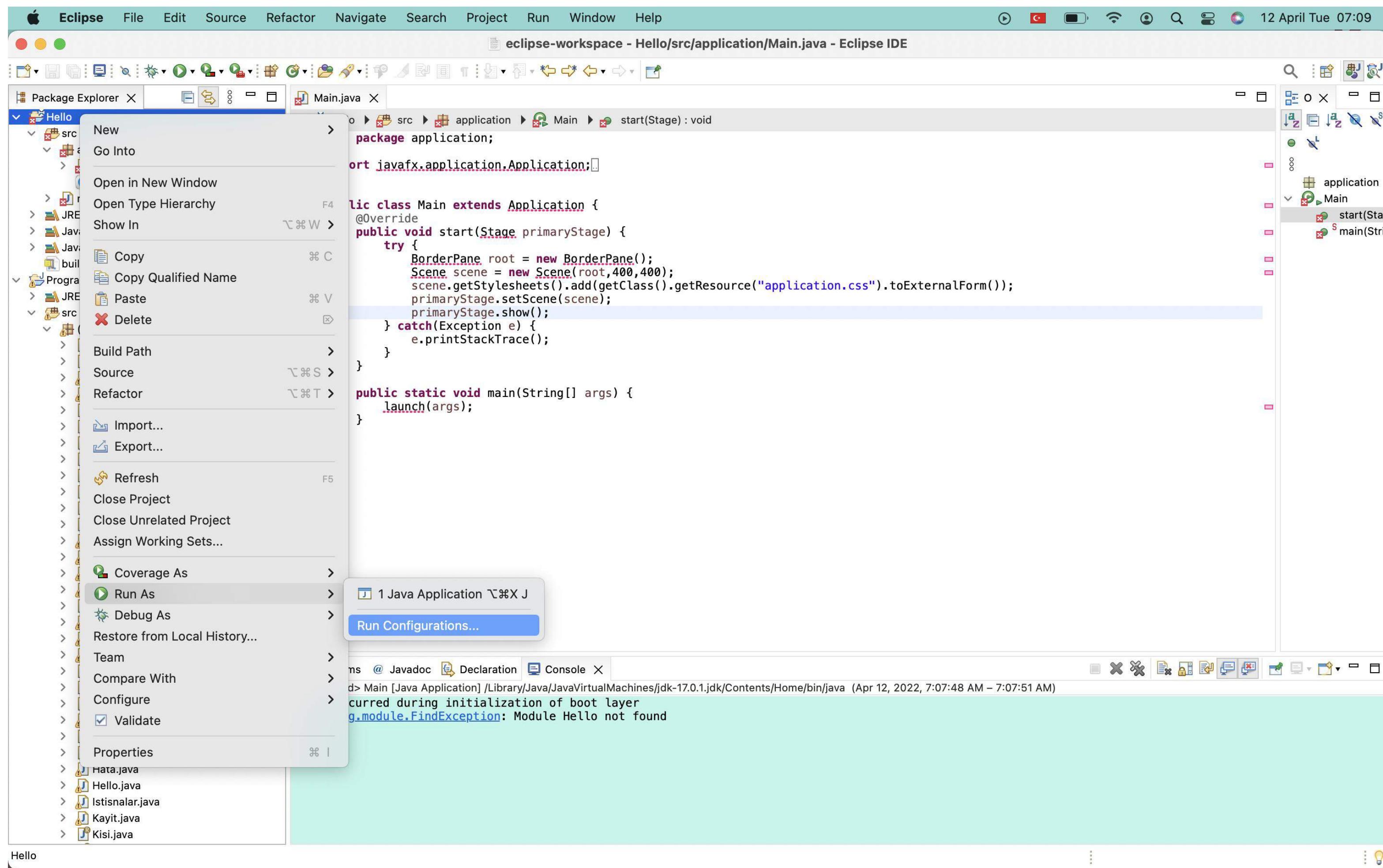
# Installing JavaFX

- Let's add libraries to the JavaFX project.



# Installing JavaFX

- Let's add the VM to the JavaFX project.



# Installing JavaFX

- Let's add the VM to the JavaFX project.

The screenshot shows a browser window displaying the JavaFX Getting Started with JavaFX page from openjfx.io. The page has a sidebar with links like Introduction, Install Java, Run HelloWorld using JavaFX, etc. A main content area contains a warning message:

**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 17 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.

Below this, there is a section titled "3. Add VM arguments" with instructions:

To solve the issue, click on `Run -> Run Configurations... -> Java Application`, create a new launch configuration for your project named 'hellofx' and add these VM arguments:

**Linux/Mac**   **Windows**

```
--module-path /path/to/javafx-sdk-17.0.1/lib --add-modules javafx.controls,javafx.fxml
```

**Warning:** Make sure the checkbox "Use the -XstartOnFirstThread argument when launching with SWT" is not checked.

At the bottom, there is a screenshot of the IntelliJ IDEA Run Configurations dialog. The "VM arguments" field contains the command: `--module-path /path/to/javafx-sdk-17.0.1/lib --add-modules javafx.controls,javafx.fxml`. The "Use the -XstartOnFirstThread argument when launching with SWT" checkbox is unchecked.

# Example Program

## Example - 1

- Let's write code that adds various shapes to the window.

# Example Program

## Example - 1

```
package com.example.hellofx;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Scene;
import javafx.scene.paint.Color;
import javafx.scene.shape.Circle;
import javafx.scene.shape.Line;
import javafx.scene.shape.Polygon;
import javafx.scene.shape.Rectangle;
import javafx.scene.text.Font;
import javafx.scene.text.Text;
import javafx.stage.Stage;
import javafx.scene.Group;

import java.io.IOException;

public class HelloApplication extends Application {
    @Override
    public void start(Stage primaryStage) throws Exception {
        //FXMLLoader fxmlLoader = new FXMLLoader(HelloApplication.class.getResource("hello-view.fxml"));
        Group root = new Group();

        Scene scene = new Scene(root, 600,600, Color.LIGHTSKYBLUE);
        Stage stage = new Stage();
        stage.setTitle("Hello!");
        stage.setScene(scene);

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

        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);

    }
}
```

# Example Program

## Example - 1

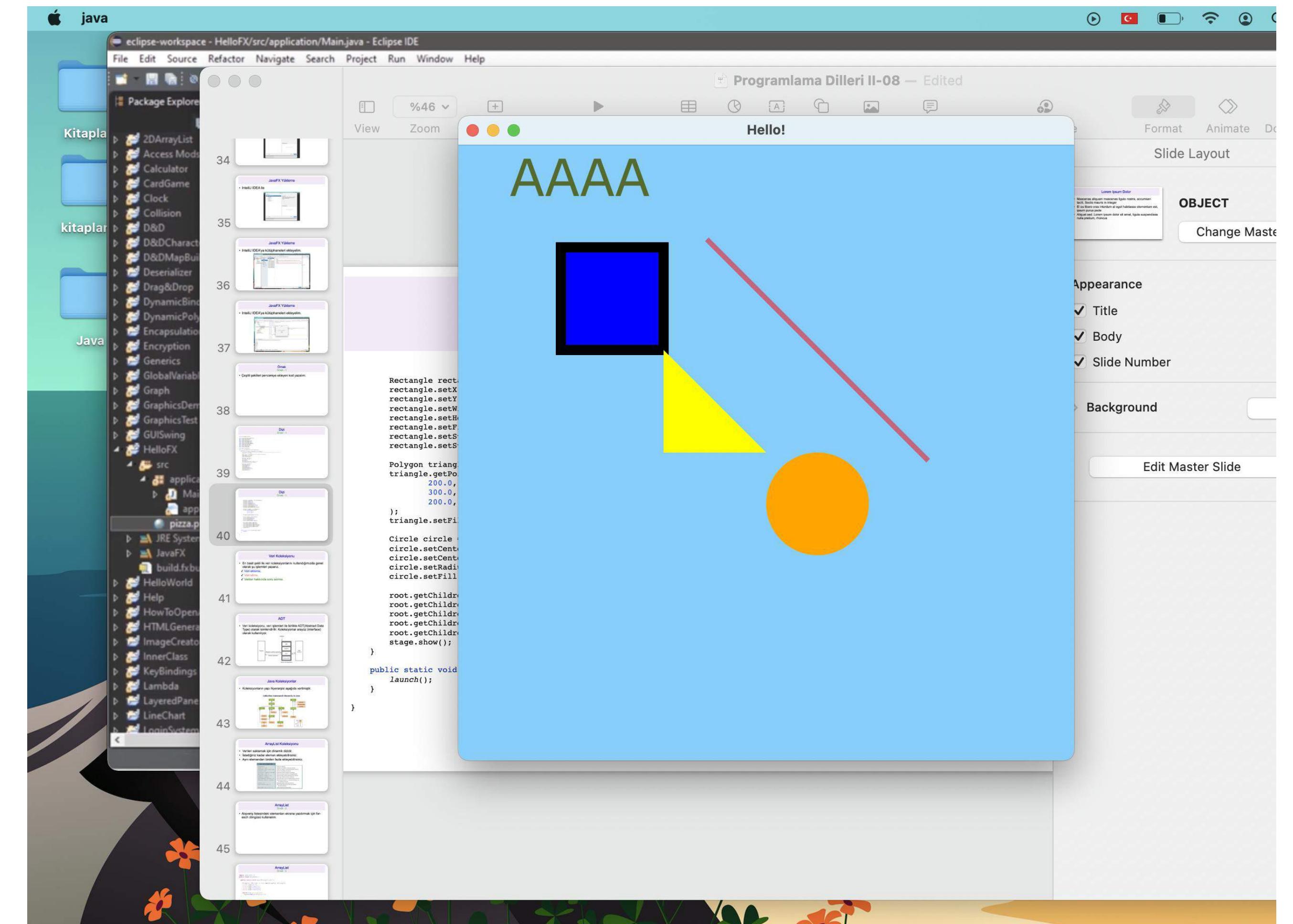
```
Rectangle rectangle = new Rectangle();
rectangle.setX(100);
rectangle.setY(100);
rectangle.setWidth(100);
rectangle.setHeight(100);
rectangle.setFill(Color.BLUE);
rectangle.setStrokeWidth(10);
rectangle.setStroke(Color.BLACK);

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

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);
stage.show();
}

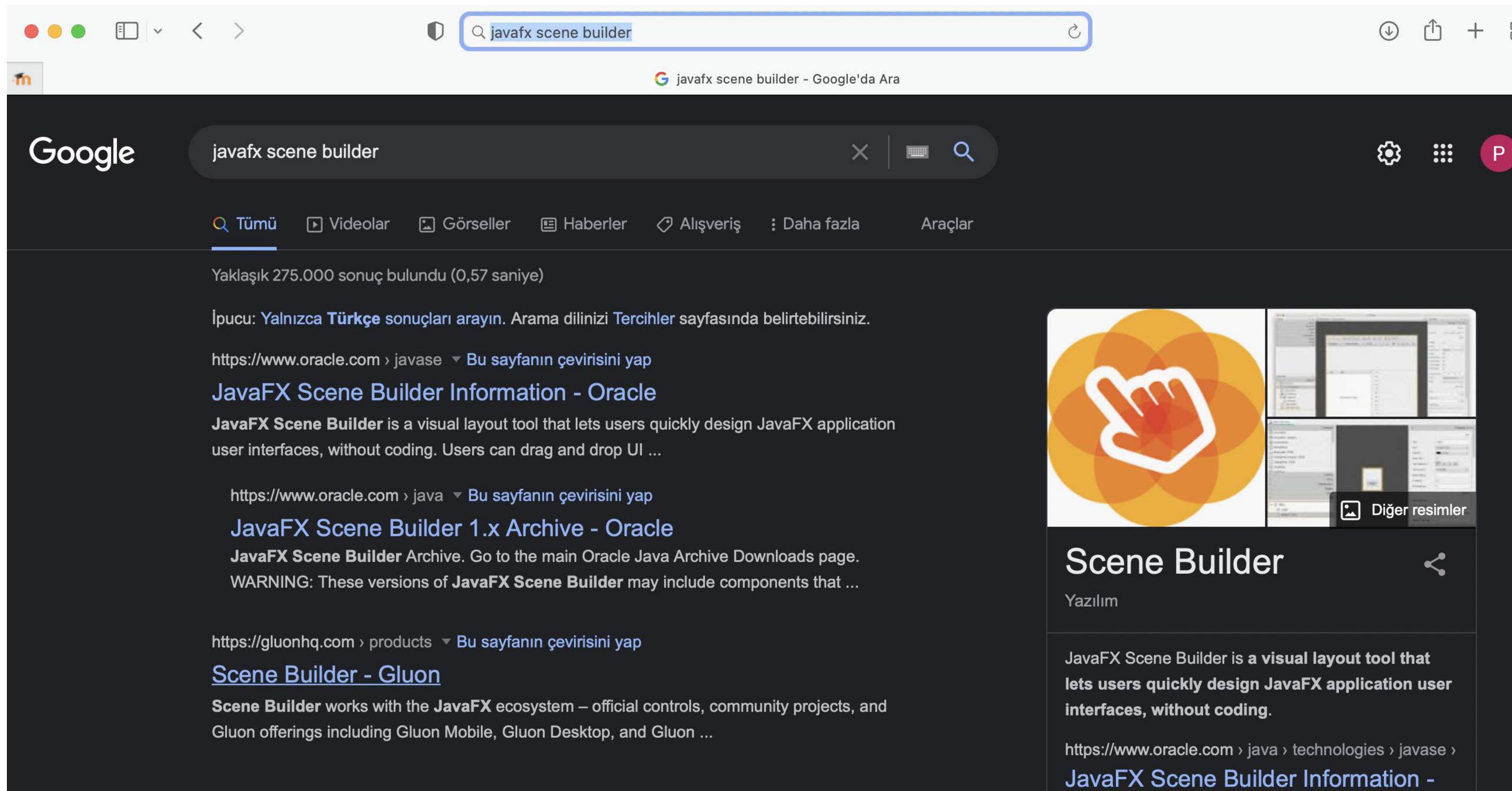
public static void main(String[] args) {
    launch();
}
```



# **JavaFX and Scene Builder**

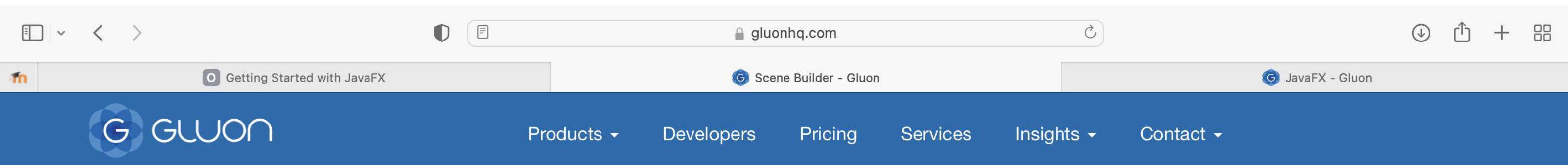
# Installing JavaFX Scene Builder

- Type JavaFX Scene Builder and search Google.



# Installing JavaFX Scene Builder

- Install JavaFX Scene Builder.



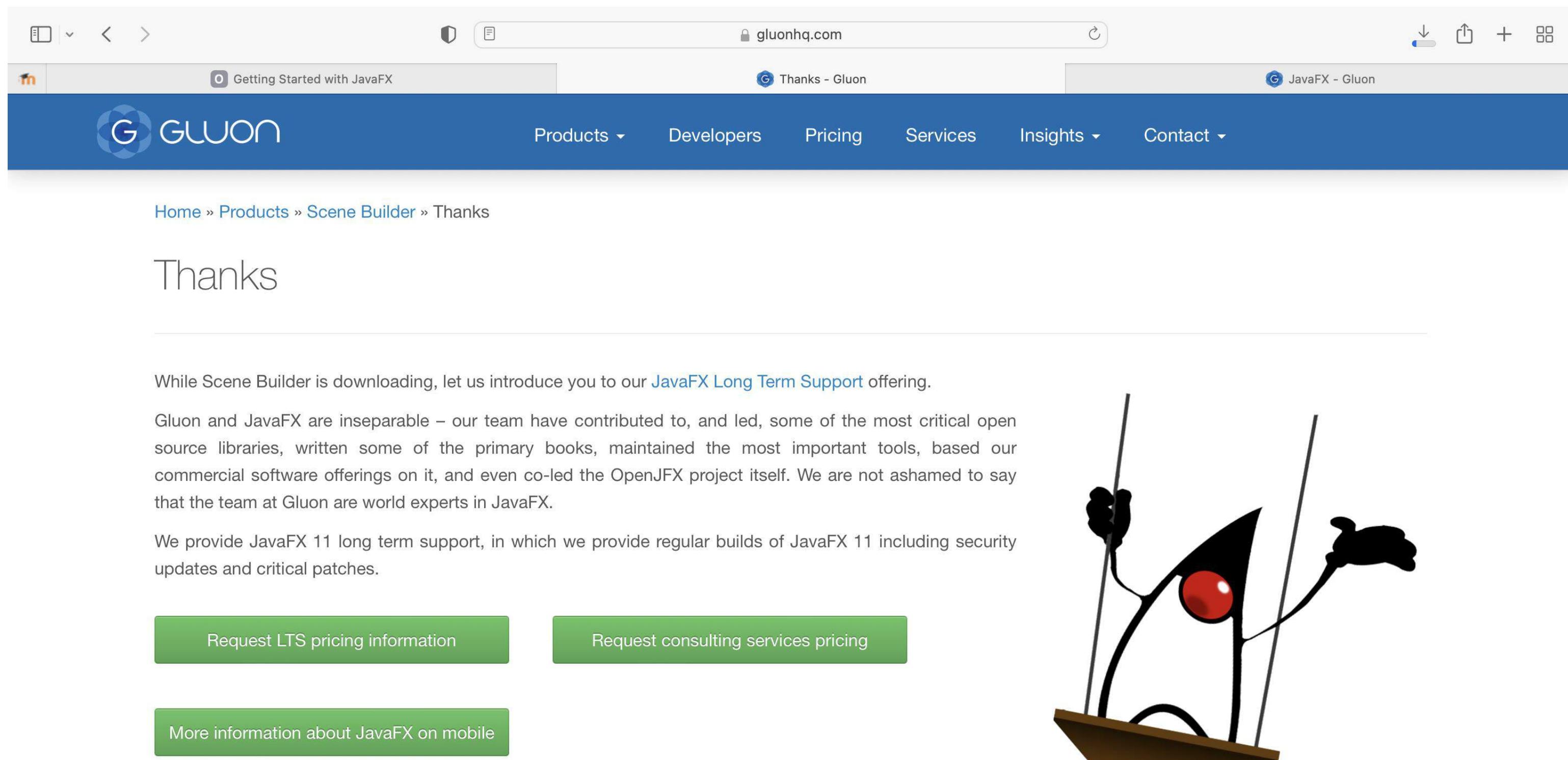
The screenshot shows a web browser displaying the Gluon website at gluonhq.com. The URL bar shows 'gluonhq.com'. The page title is 'Scene Builder - Gluon'. The navigation menu includes 'Products', 'Developers', 'Pricing', 'Services', 'Insights', and 'Contact'. Below the menu, a breadcrumb trail reads 'Home > Products > Scene Builder'. The main content area features a large image of the JavaFX Scene Builder application interface. The application window is titled 'Untitled' and contains a 'BorderPane' layout. Inside the 'BorderPane', there is a 'HTMLEditor' component which displays a table with two columns labeled 'C1' and 'C2'. The 'Inspector' pane on the right shows properties for the 'HTMLEditor' node, including 'Disable' (unchecked), 'Opacity' (set to 1), 'Node Orientation' (set to 'INHERIT'), and 'Visible' (checked). A green button labeled 'Download Now' is located in the bottom right corner of the application window.

Drag & Drop,  
Rapid Application  
Development.

Download Now

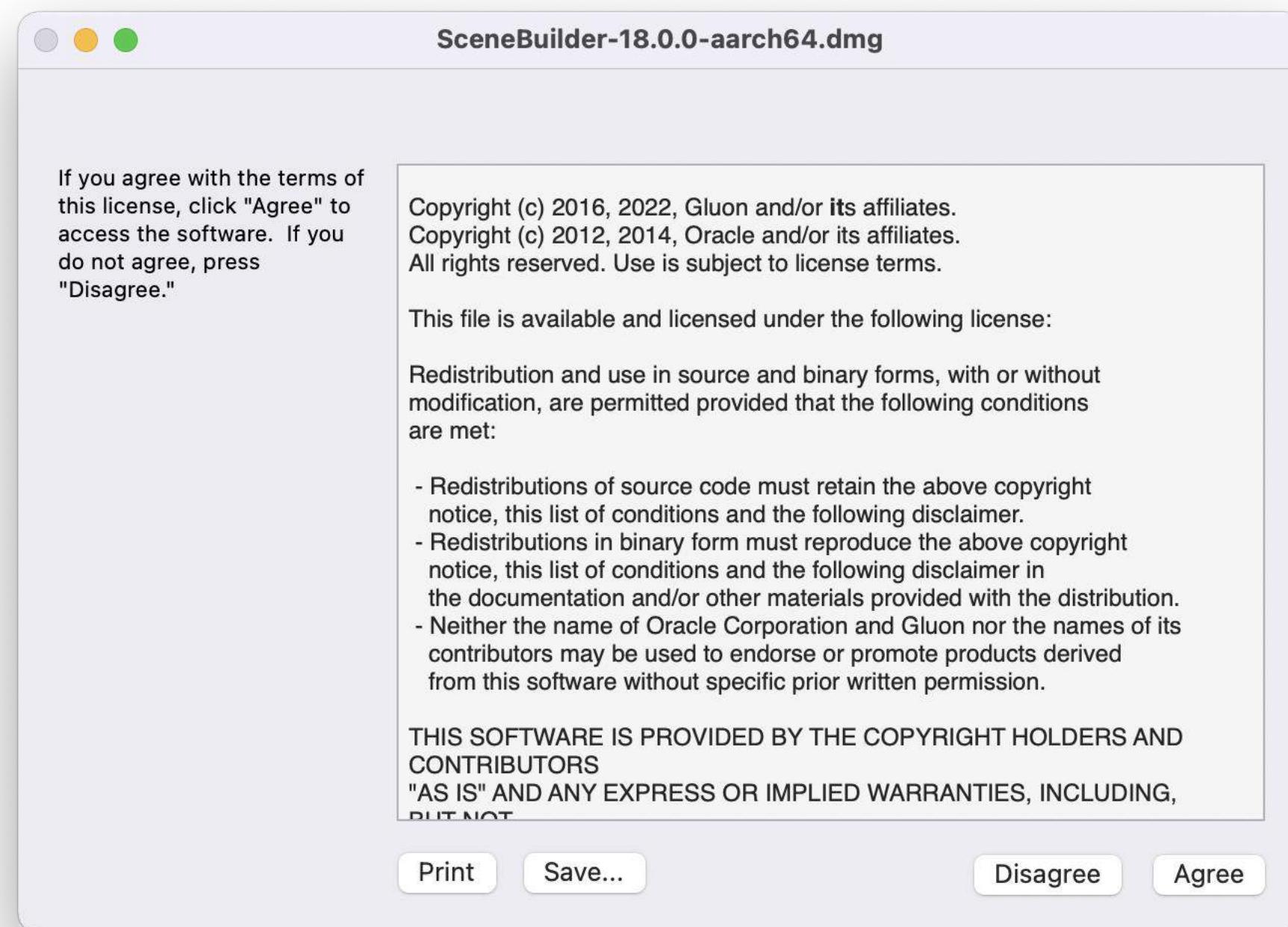
# Installing JavaFX Scene Builder

- Install JavaFX Scene Builder.



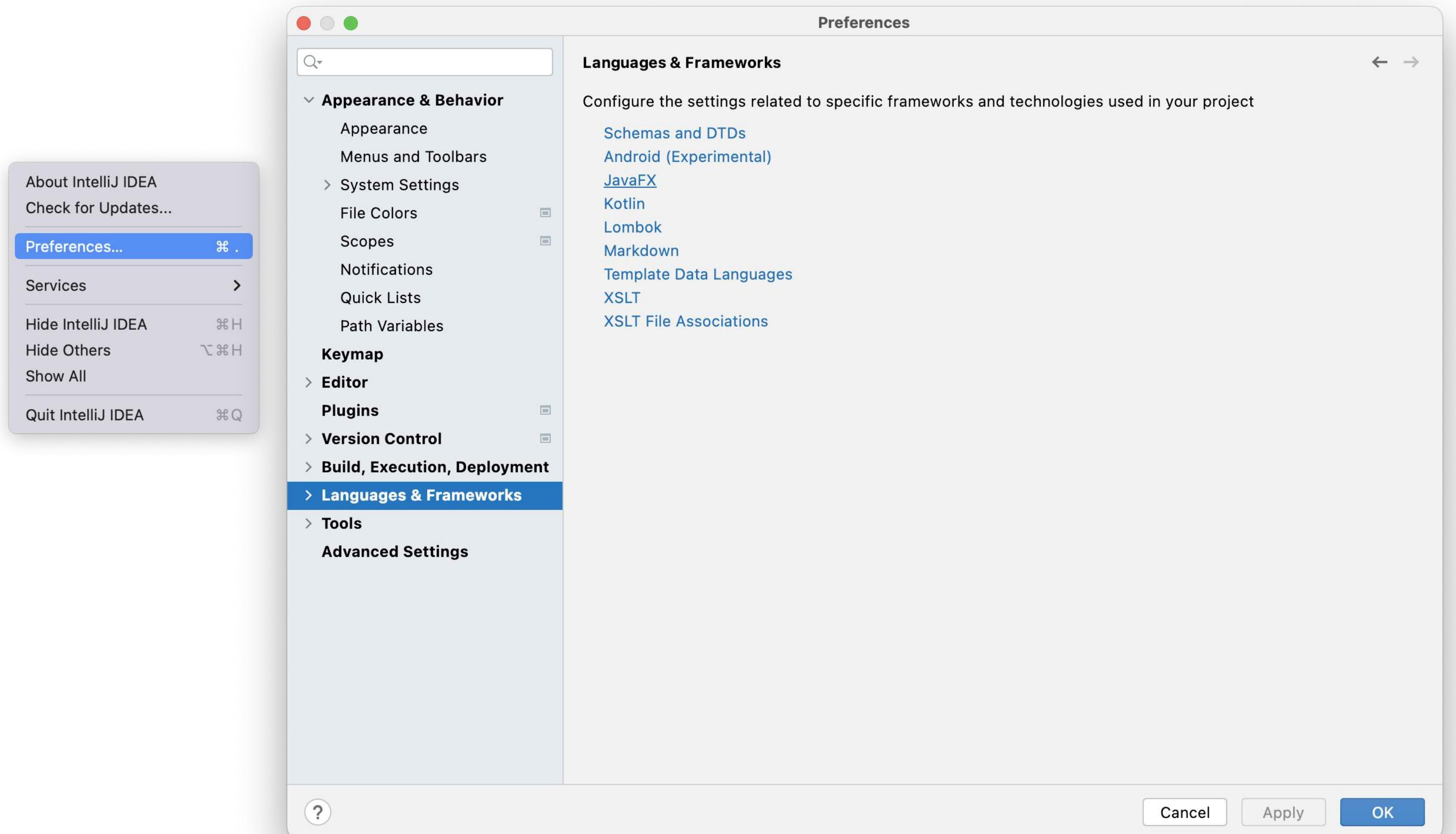
# Installing JavaFX Scene Builder

- Install JavaFX Scene Builder.



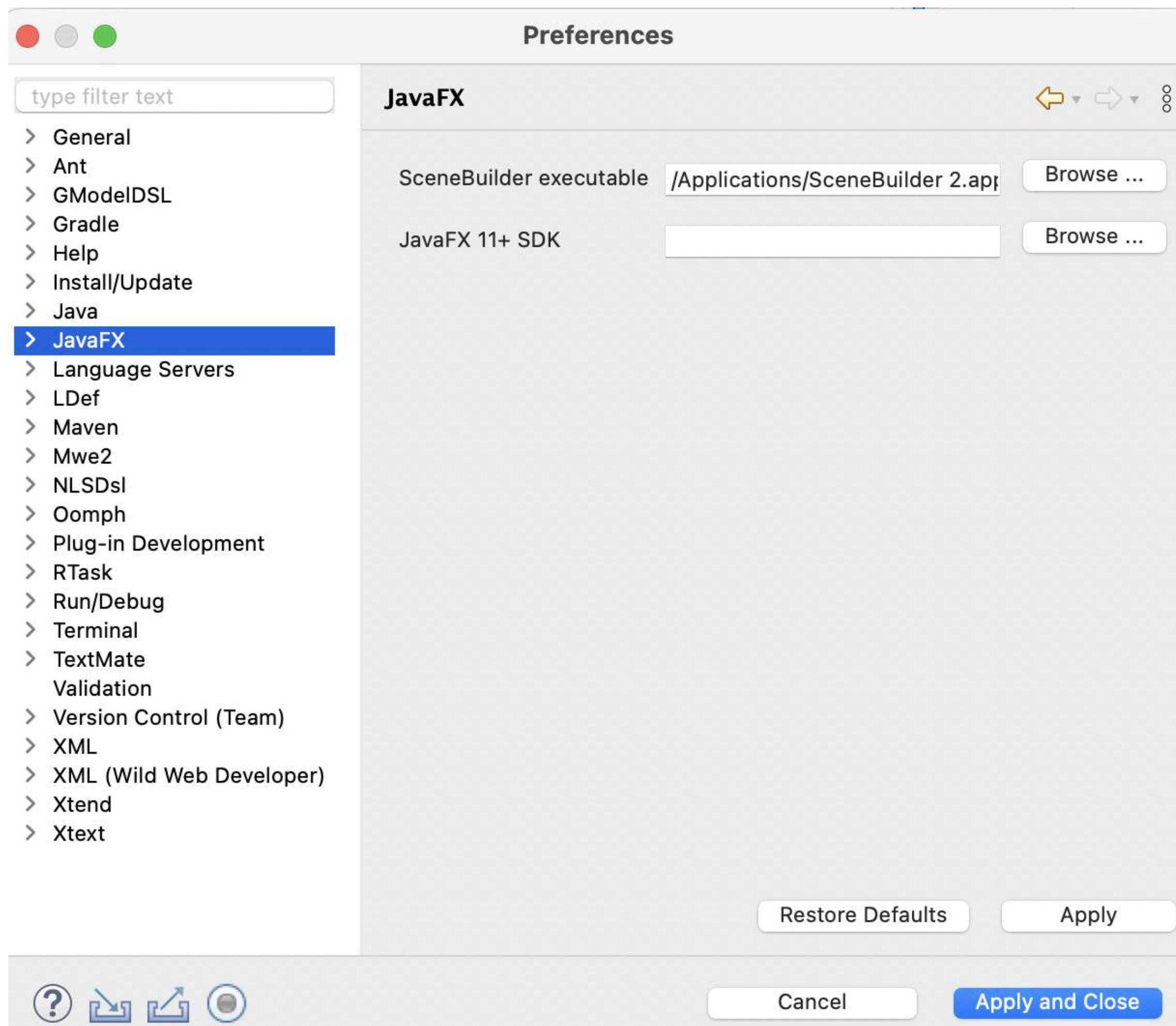
# Installing JavaFX Scene Builder

- Install JavaFX Scene Builder.



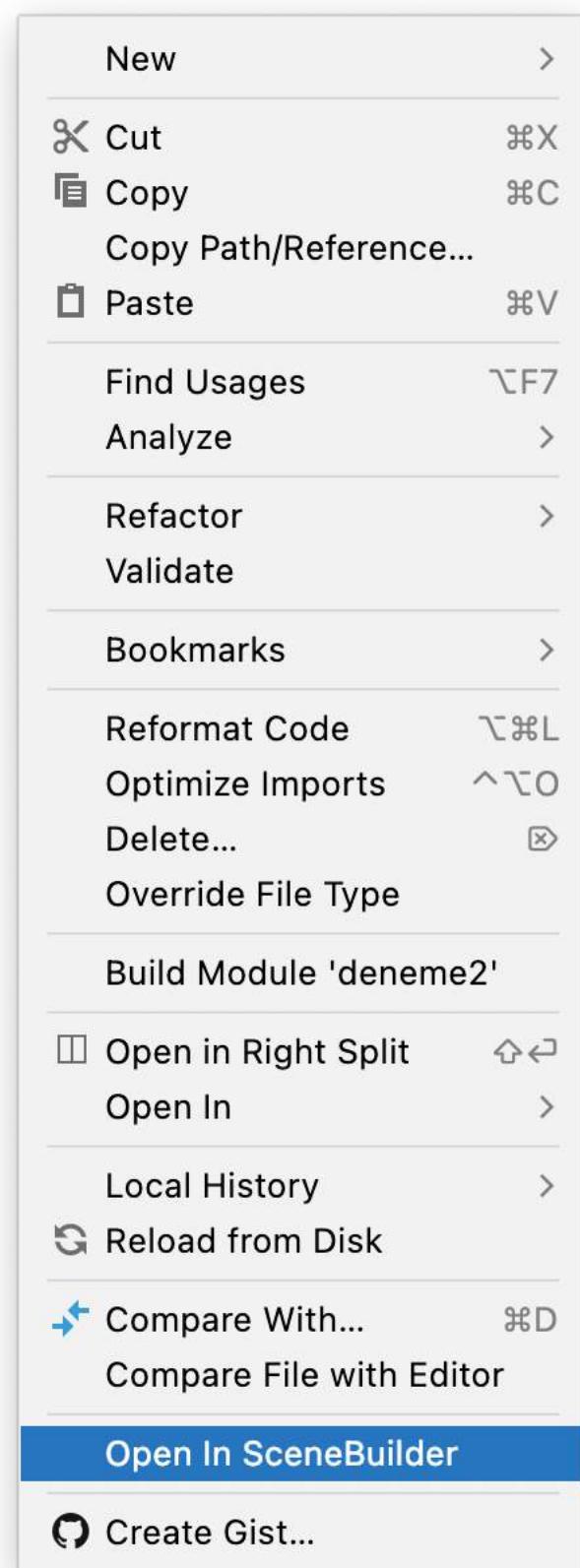
# Installing JavaFX Scene Builder

- Install JavaFX Scene Builder.



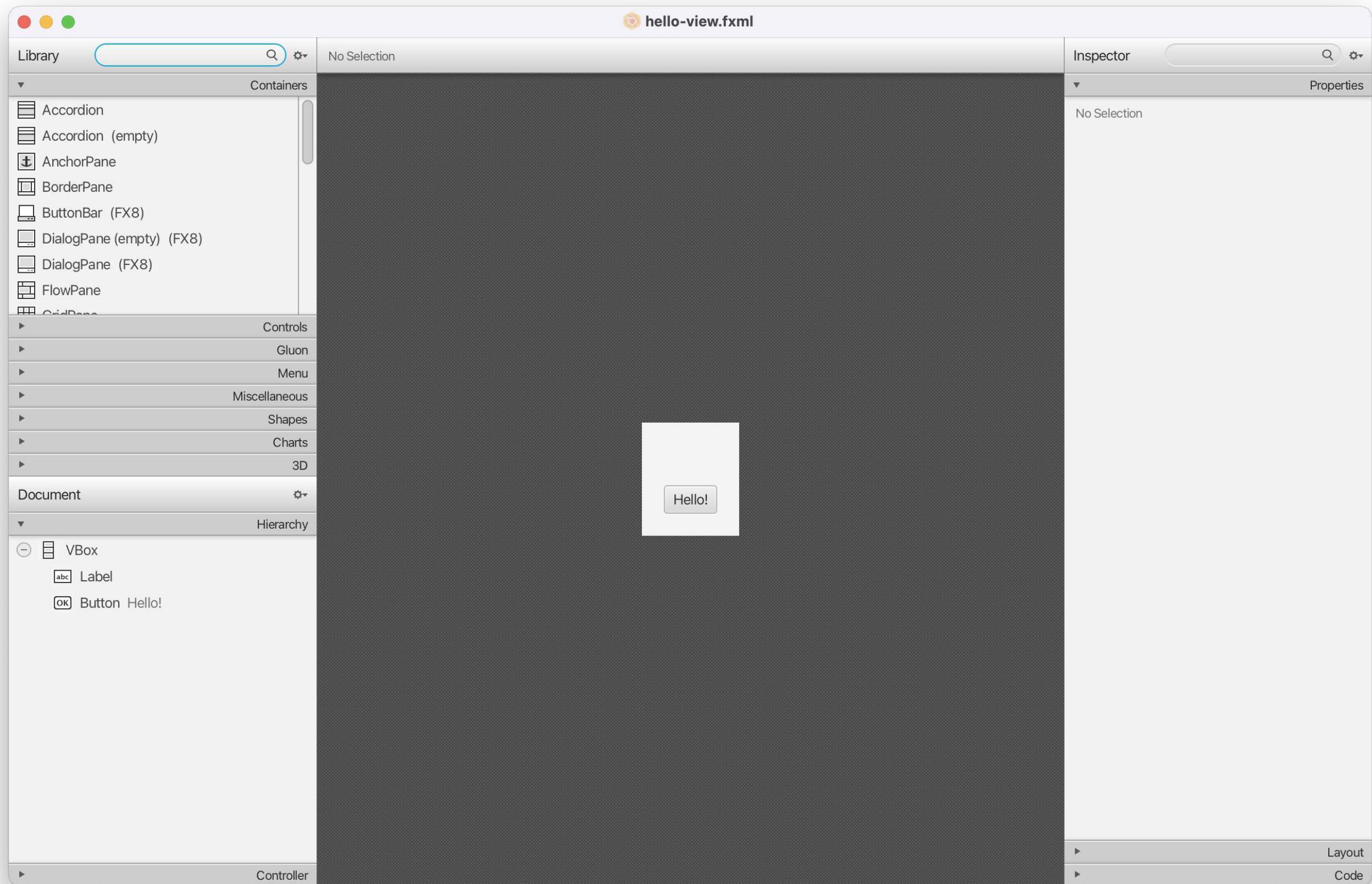
# Installing JavaFX Scene Builder

- Install JavaFX Scene Builder.



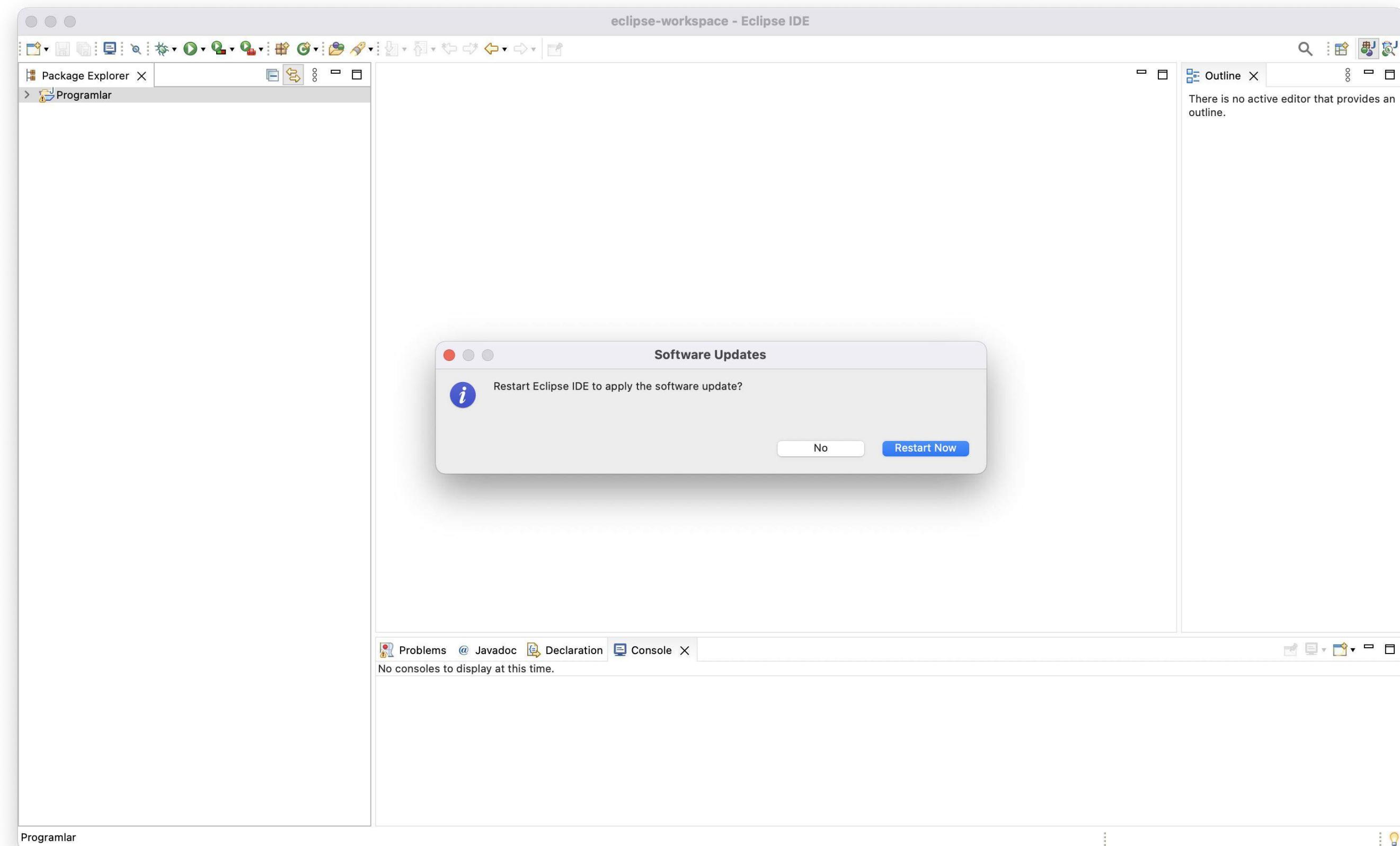
# Installing JavaFX Scene Builder

- Install JavaFX Scene Builder.



# Installing JavaFX Scene Builder

- Install JavaFX Scene Builder.



# Example Program

## Example - 2

- Let's implement the shape application we created with Scene Builder.

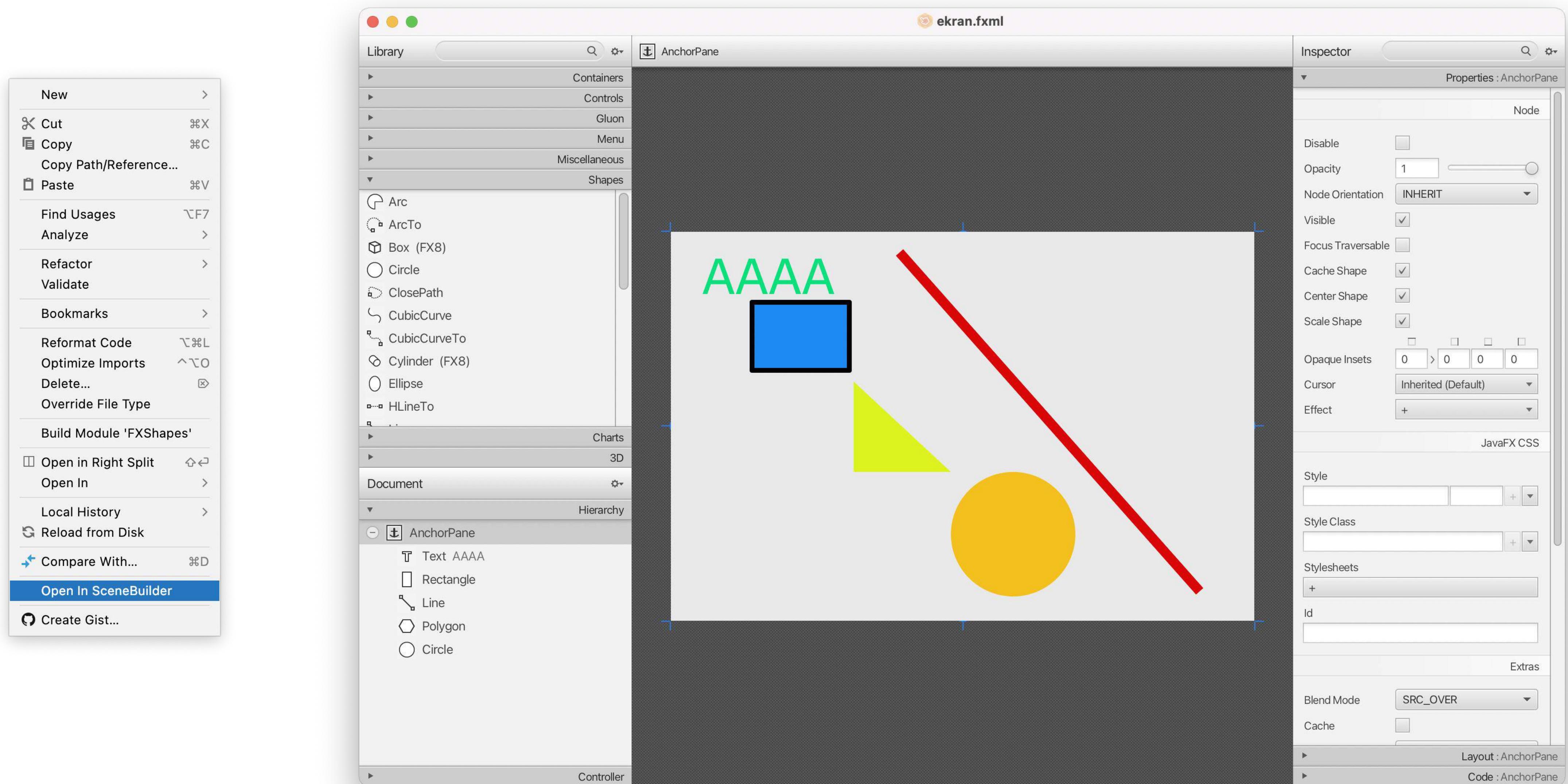
# Shapes

## Example - 2

```
public class HelloApplication extends Application {  
    @Override  
    public void start(Stage stage) throws IOException {  
  
        FXMLLoader fxmlLoader = new FXMLLoader(HelloApplication.class.getResource("ekran.fxml"));  
  
        Scene scene = new Scene(fxmlLoader.load(), Color.LIGHTSKYBLUE);  
        stage.setTitle("Hello!");  
        stage.setScene(scene);  
        stage.show();  
    }  
  
    public static void main(String[] args) {  
        launch();  
    }  
}
```

# Shapes

## Example - 2



## Shapes

### Example - 2

- Let's develop an application that moves a shape up, down, right, left.
- We will connect with the OnAction events of the buttons.
- We will call the .fxml file we created with the following code.

```
Parent root = FXMLLoader.load(getClass().getResource("Main.fxml"));
```

# Shapes

## Example - 2

```
package com.example.deneme4;

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

import java.io.IOException;

public class HelloApplication extends Application {
    @Override
    public void start(Stage stage) throws Exception {
        try {
            Parent root = FXMLLoader.load(getClass().getResource("Main.fxml"));
            Scene scene = new Scene(root);
            stage.setTitle("MERHABA ARKADAŞLAR!");
            stage.setScene(scene);
            stage.show();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

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

# Shapes

## Example - 2

```
package com.example.deneme4;

import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.shape.Circle;

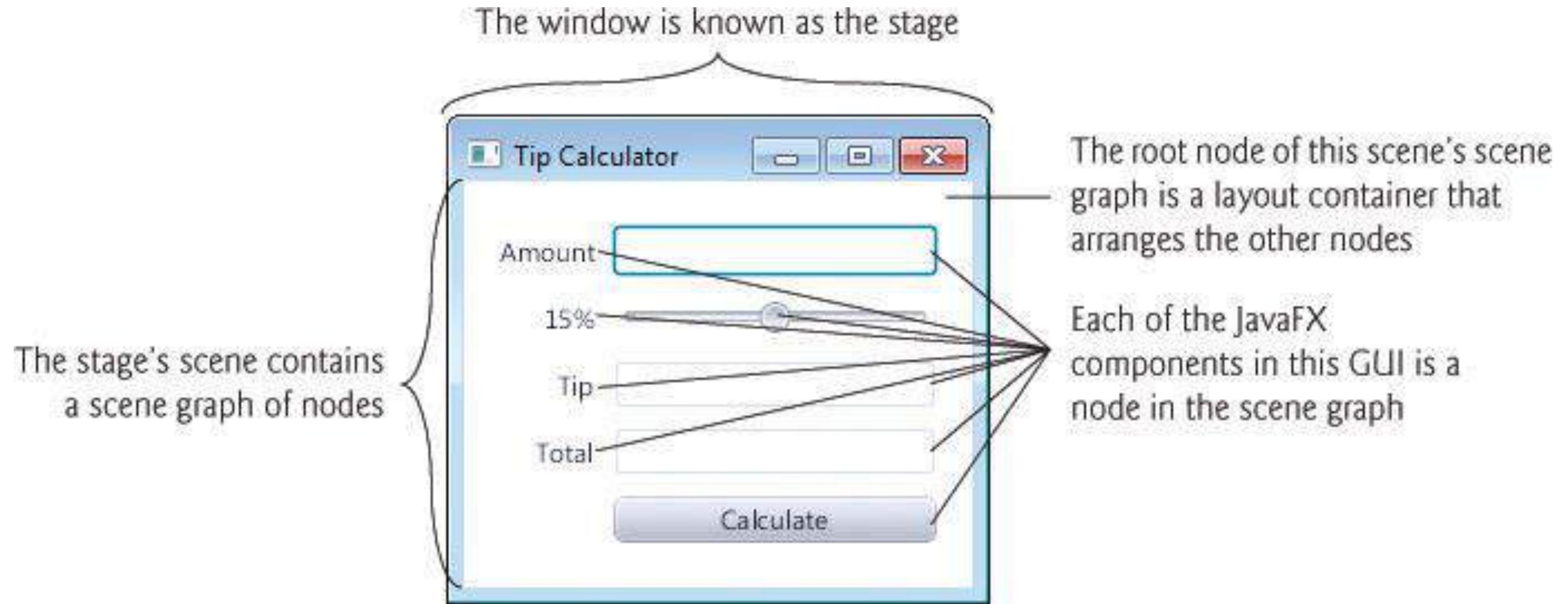
public class Controller {
    @FXML
    private Circle daire;
    private double x;
    private double y;

    public void yukariya(ActionEvent e){
        daire.setCenterY(y=y-10);
        //System.out.println("Yukariya");
    }
    public void asagiya(ActionEvent e){
        daire.setCenterY(y=y+10);
        //System.out.println("Aşağıya");
    }
    public void sola(ActionEvent e){
        daire.setCenterX(x=x-10);
        //System.out.println("Sola");
    }
    public void saga(ActionEvent e){
        daire.setCenterX((x=x+10));
        //System.out.println("Sağa");
    }
}
```

# JavaFX Program Structure

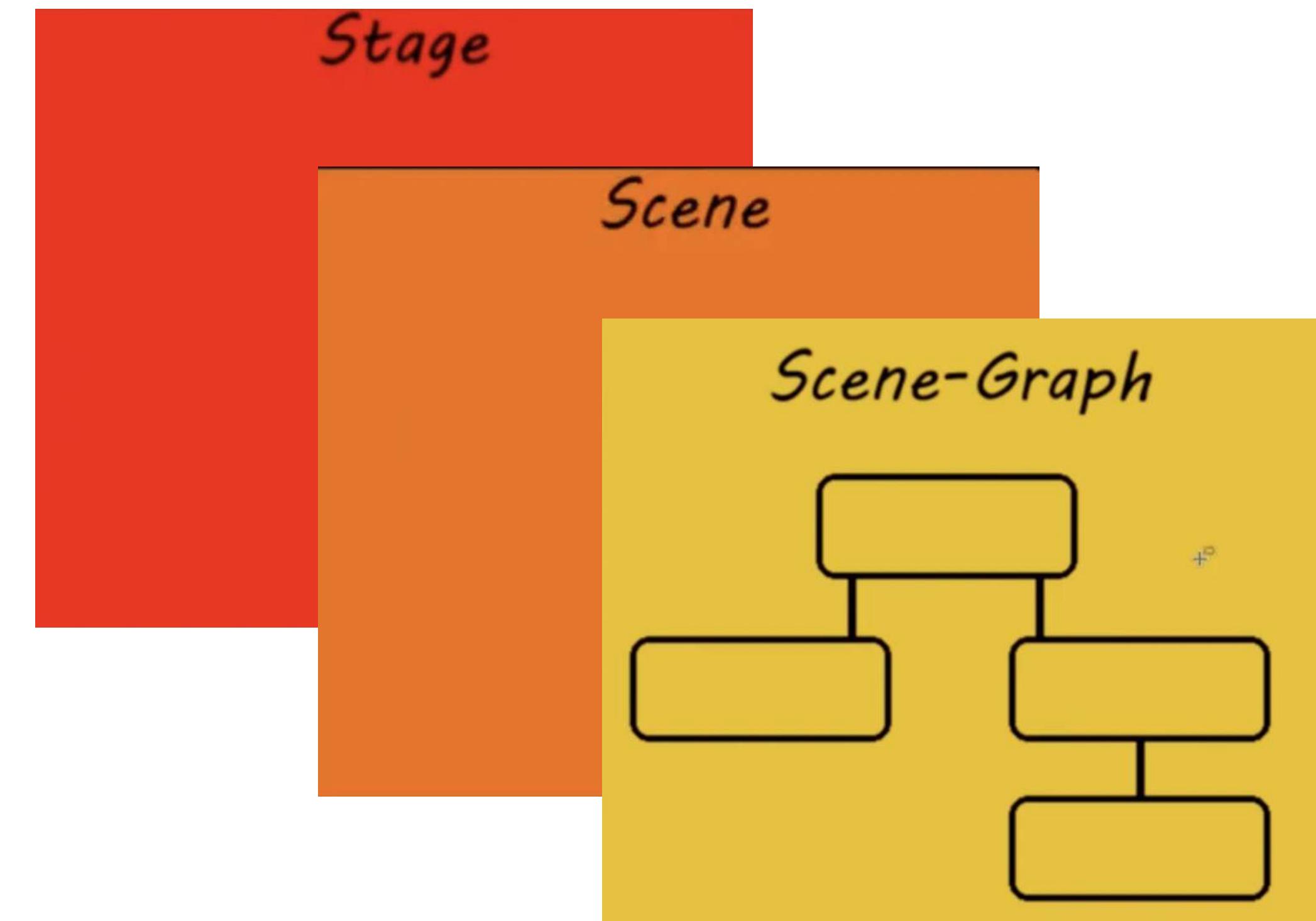
# A JavaFX Program

- It consists of Stage, Scene and nodes.



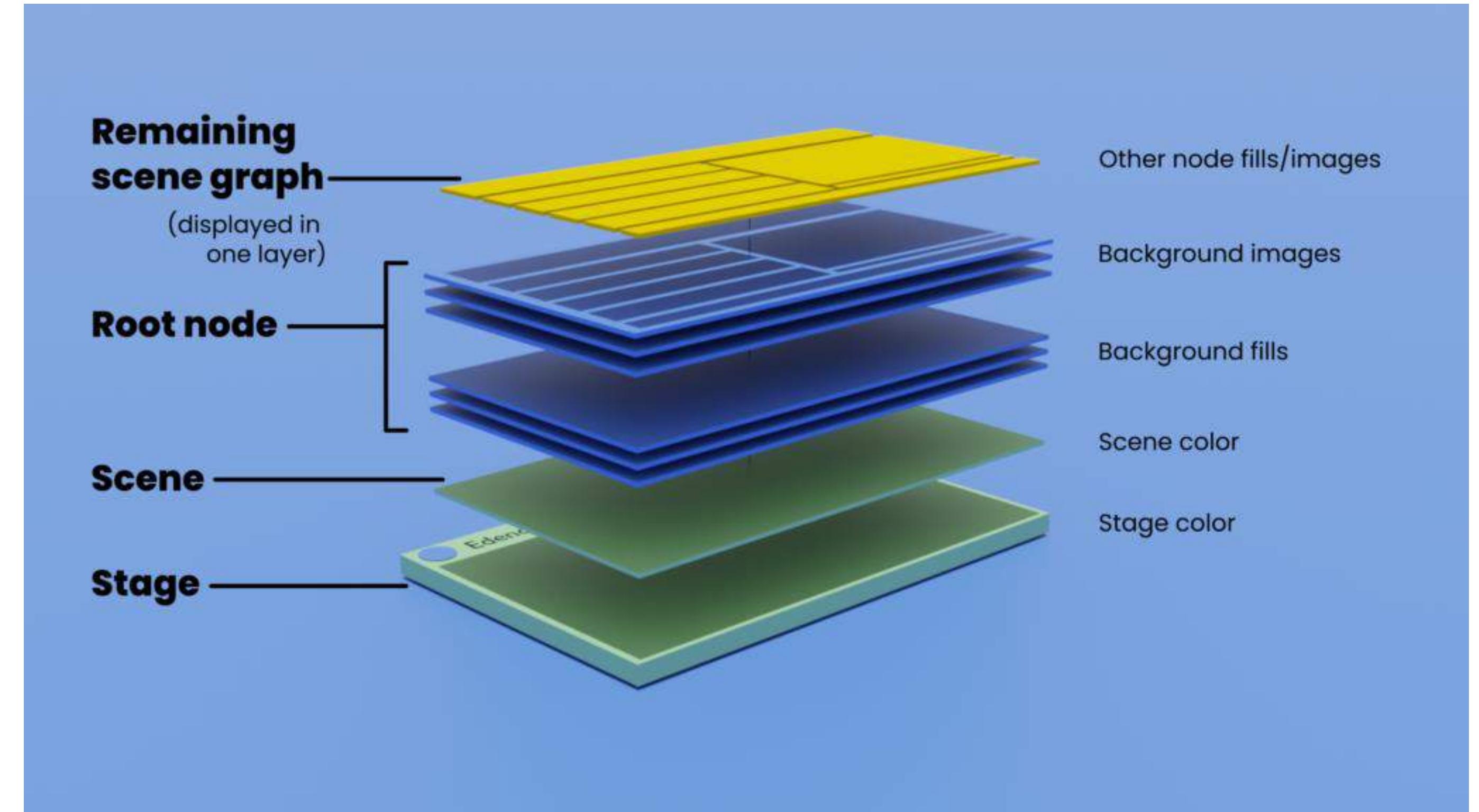
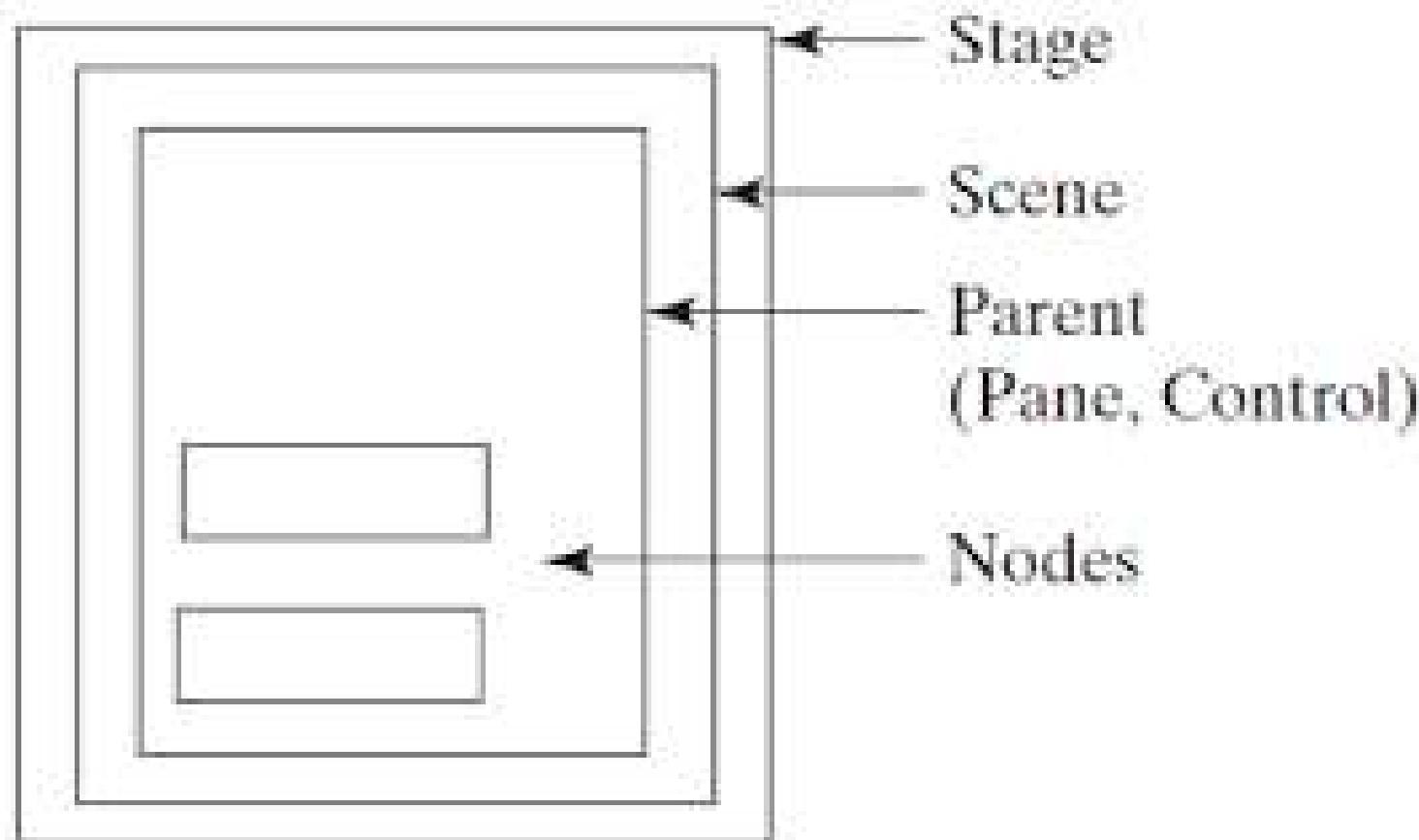
# A JavaFX Program

- It consists of Stage, Scene and nodes.



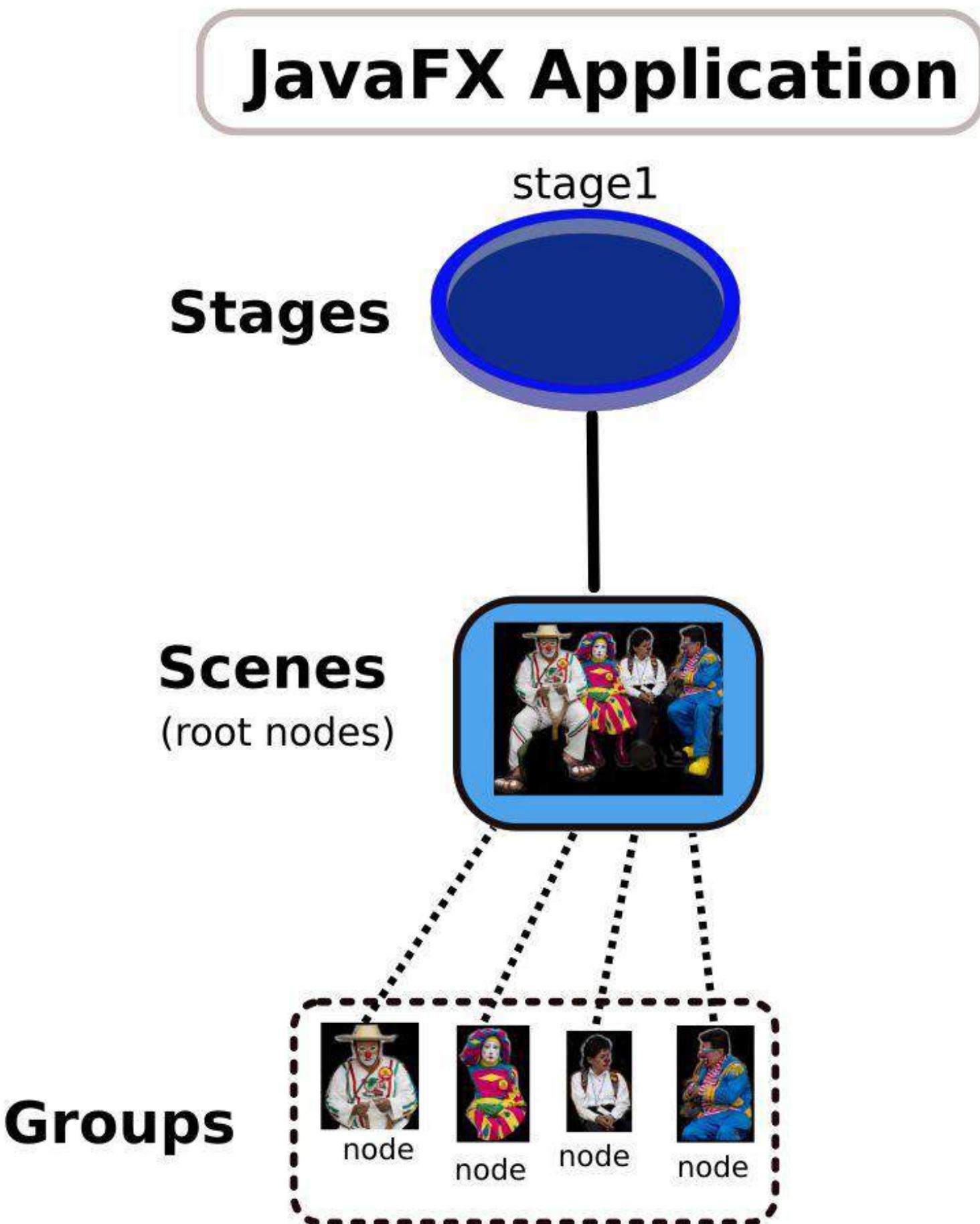
# A JavaFX Program

- It consists of Stage, Scene and nodes.



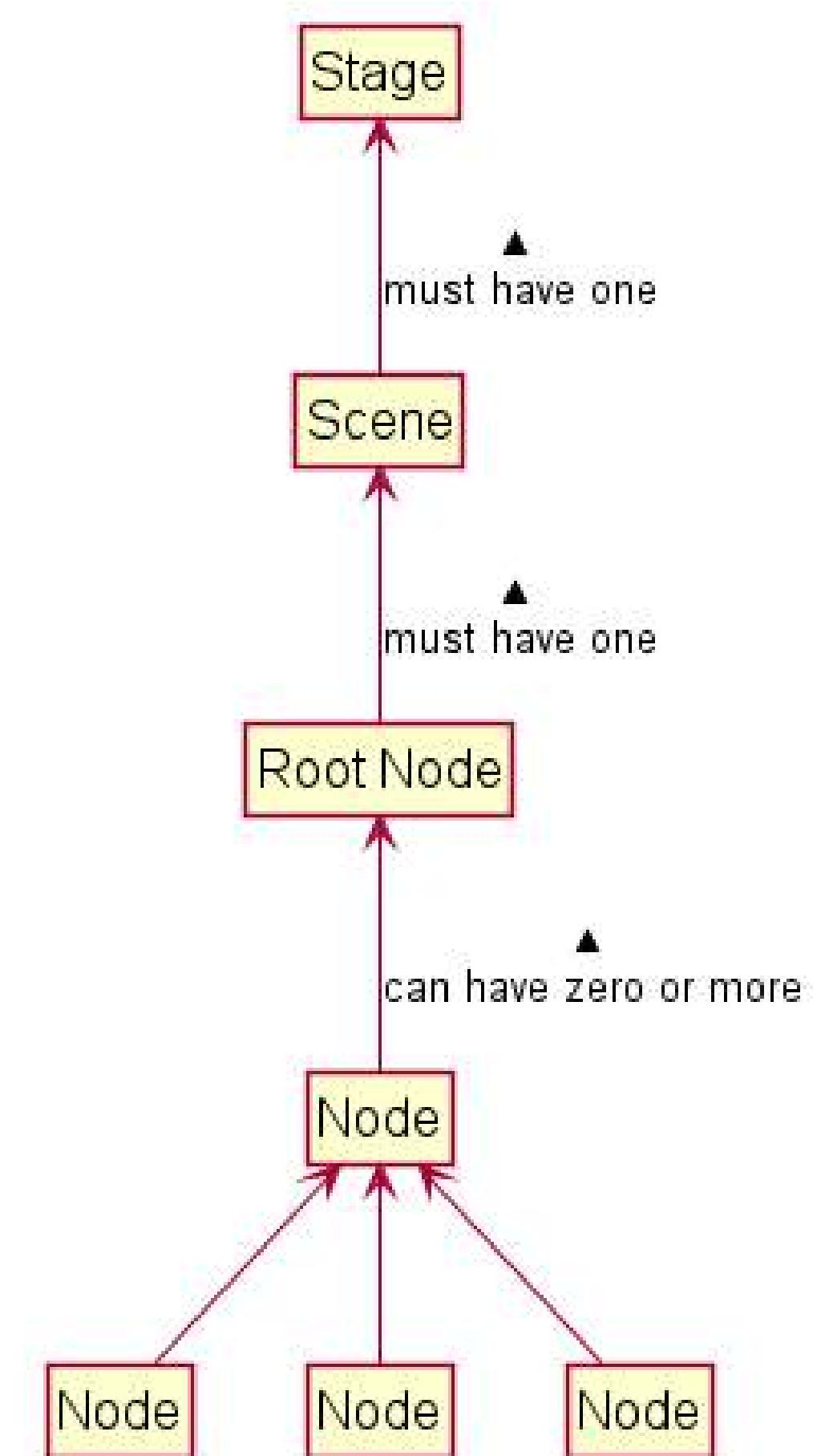
# A JavaFX Program

- It consists of Stage, Scene and nodes.



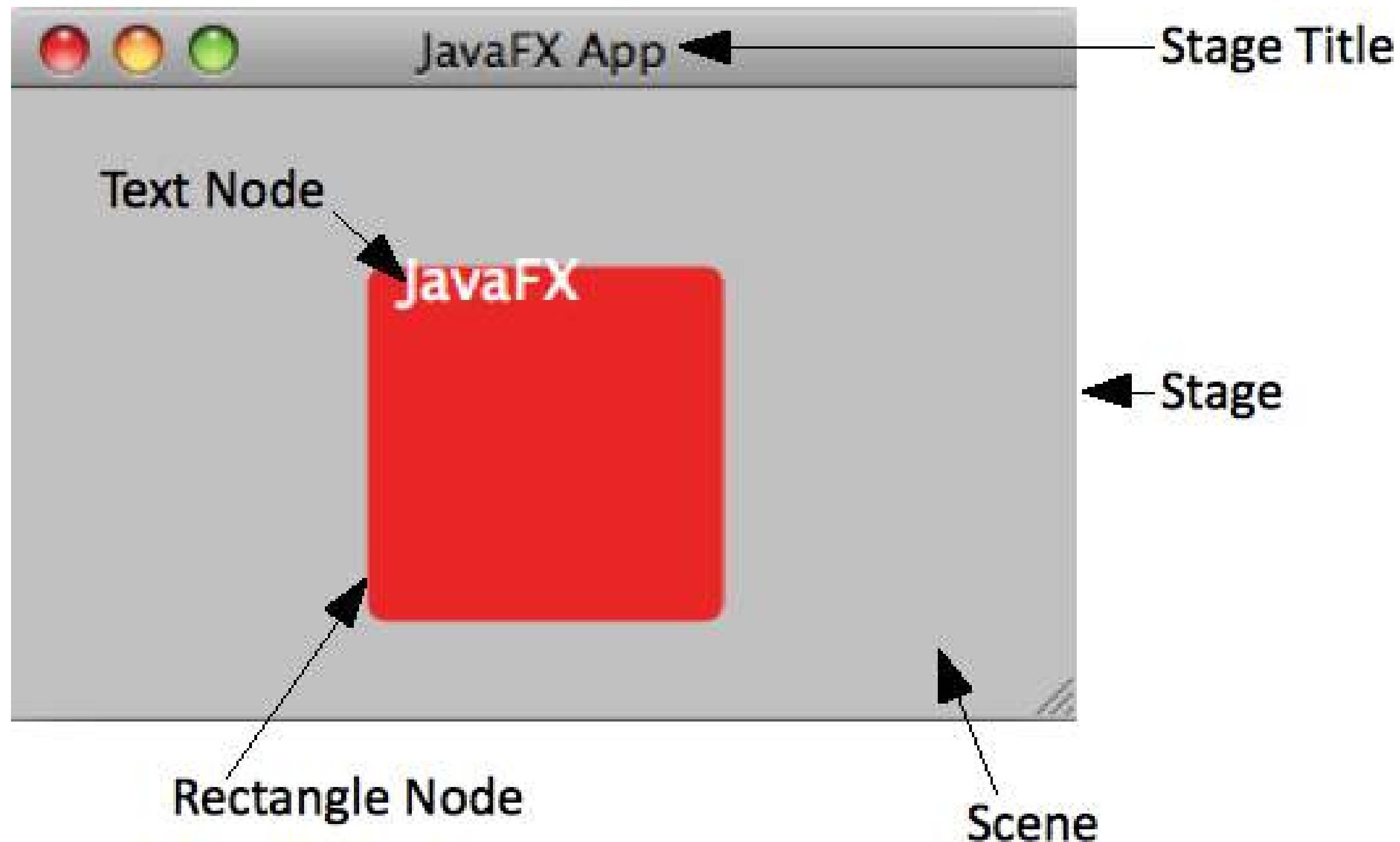
# A JavaFX Program

- It consists of Stage, Scene and nodes.



# A JavaFX Program

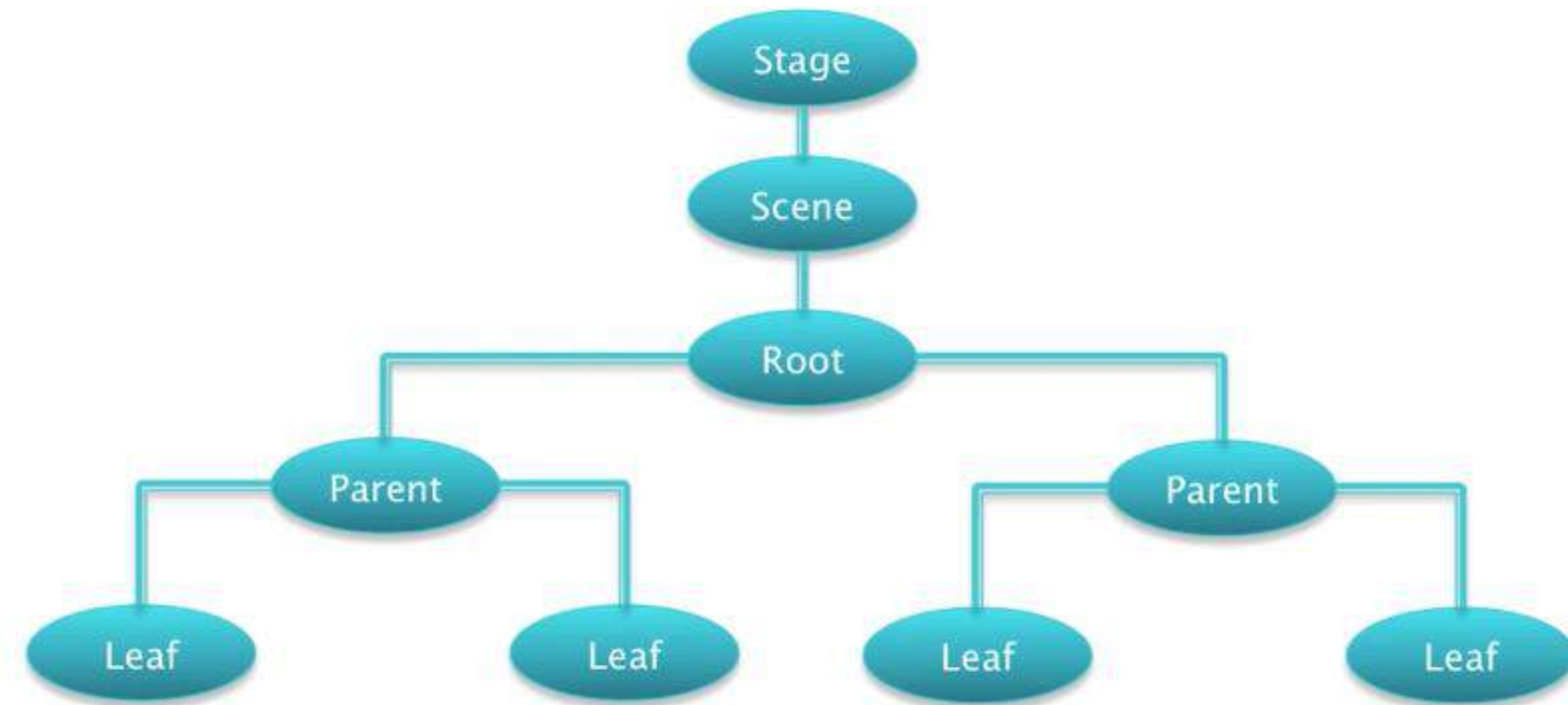
- It consists of Stage, Scene and nodes.



# A JavaFX Program

- It consists of Stage, Scene and nodes.

## Screen Graph Nodes



# Homework - 1

- Work on your own sample shapes program.



## Next Lecture

- We will continue to the topic.

# Visual Programming

- There are too many examples in open sources on the subject. You can review it.

# Programming Session

- Java FX

# Summary

- As a result, this week we learned Java FX basic programming.