COMP20050 - Software Engineering Project II

Introduction to JavaFX

Ravi Reddy Manumachu
ravi.manumachu@ucd.ie

UCD School of Computer Science.

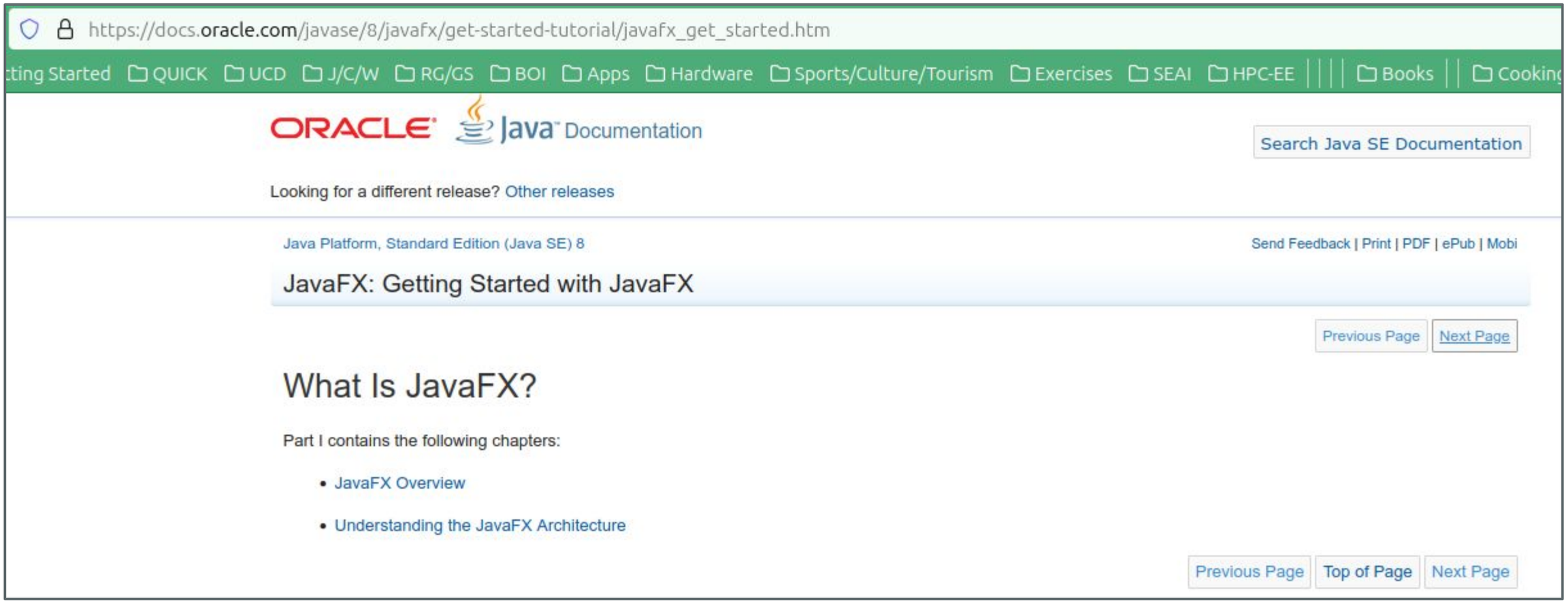
Scoil na Ríomheolaíochta UCD.

Outline (Learning Objectives)

- Become familiar with features in **JavaFX**.
- Become aware of the main components in **JavaFX architecture**.
- **Build** and **run** a sample JavaFX application in IntelliJ IDE.

User Guide and Tutorial

https://docs.oracle.com/javase/8/javafx/get-started-tutorial/index.html



JavaFX Sample Applications

https://docs.oracle.com/javase/8/javafx/sample-apps/index.html

https://docs.oracle.com/javase/8/javafx/sample-apps/index.html

ting Started | QUICK | UCD | J/C/W | RG/GS | BOI | Apps | Hardware | Sports/Culture/Tourism | Exercises | SEAI | HPC-EE | Books | Cooking | NCI

ORACLE Java Documentation

Search Java SE Documentation

Looking for a different release? Other releases

Java Platform, Standard Edition (Java SE) 8

Send Feedback | Print

JavaFX: Sample Applications for Tutorials

JavaFX

Sample Applications for Tutorials

Release 8

E51640-01

March 2014

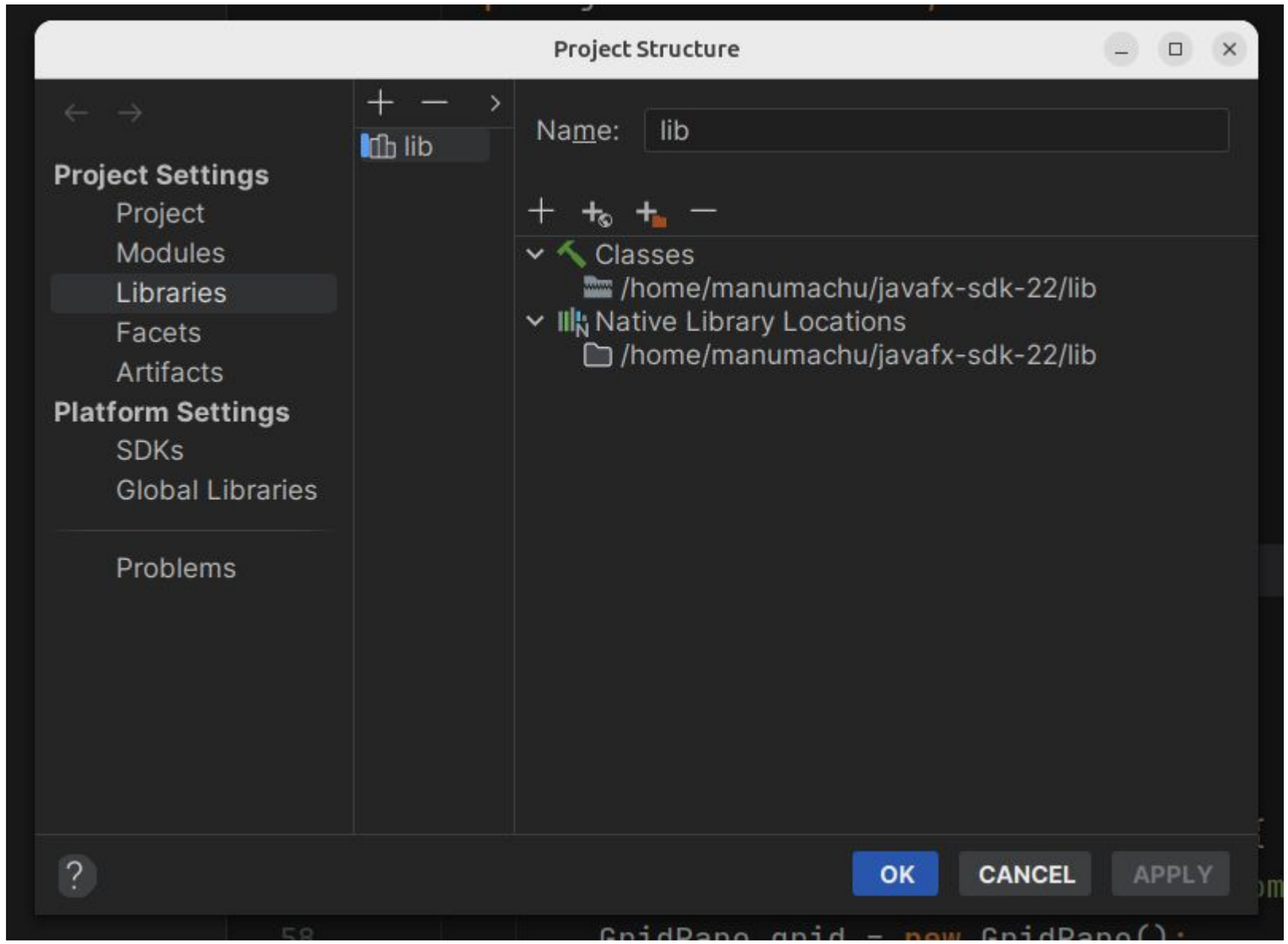
This page contains links to all of the sample applications and images that are described in JavaFX documentation. Clicking a link initiates a download of a zip file, which contains NetBeans project files, source code files, and resource files for that sample application.

The set of applications is described in the following table.

JavaFX Document	Sample Application or Resource File Name
Adding HTML Content to JavaFX Applications	`WebViewSample.zip`
Getting Started with JavaFX	`ColorfulCircles.zip`
Getting Started with JavaFX	`FXMLExample.zip`
Getting Started with JavaFX	`Login.zip`
Getting Started with JavaFX	`LoginCSS.zip`

IntelliJ and JavaFX - Add JavaFX lib

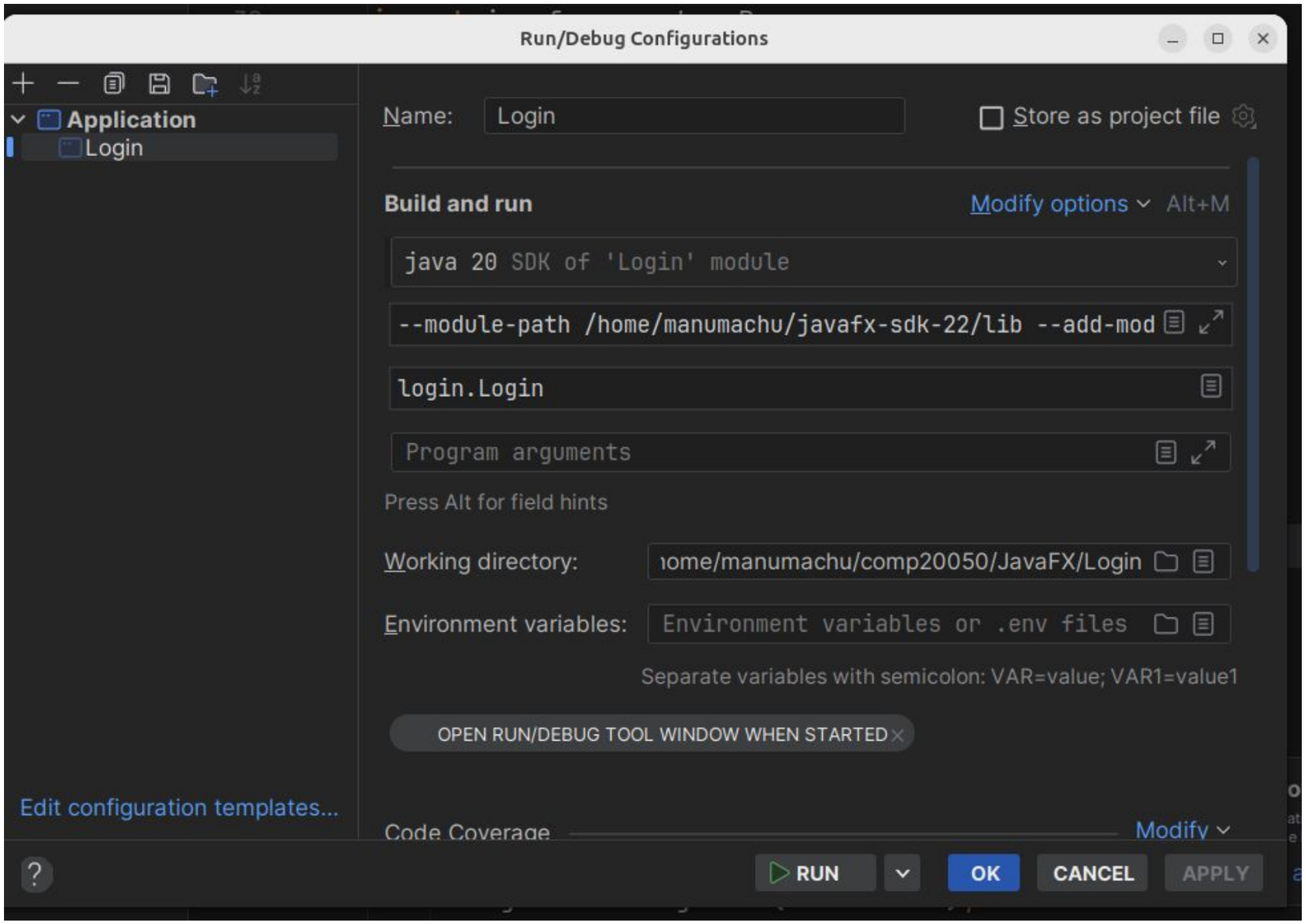
- Click **File -> Project Structure.**
- Click **Project Settings -> Libraries.**
- Click **+** and add the JavaFX lib directory.



IntelliJ and JavaFX - Edit Run Configuration

- Open the **Run/Debug Configurations** screen.
- Click **Modify Options** and tick **Add VM options**.
- Add the following to the VM options textbox.

--module-path /path/to/JavaFX/lib --add-modules=javafx.controls



JavaFX Overview

What is JavaFX?

- **JavaFX** is a set of graphics and media packages that enables developers to create and deliver rich desktop and web applications that operate consistently across diverse platforms.

- Since the **JavaFX library** is written as a Java API, JavaFX application code can reference APIs from any Java library.

JavaFX Look and Feel

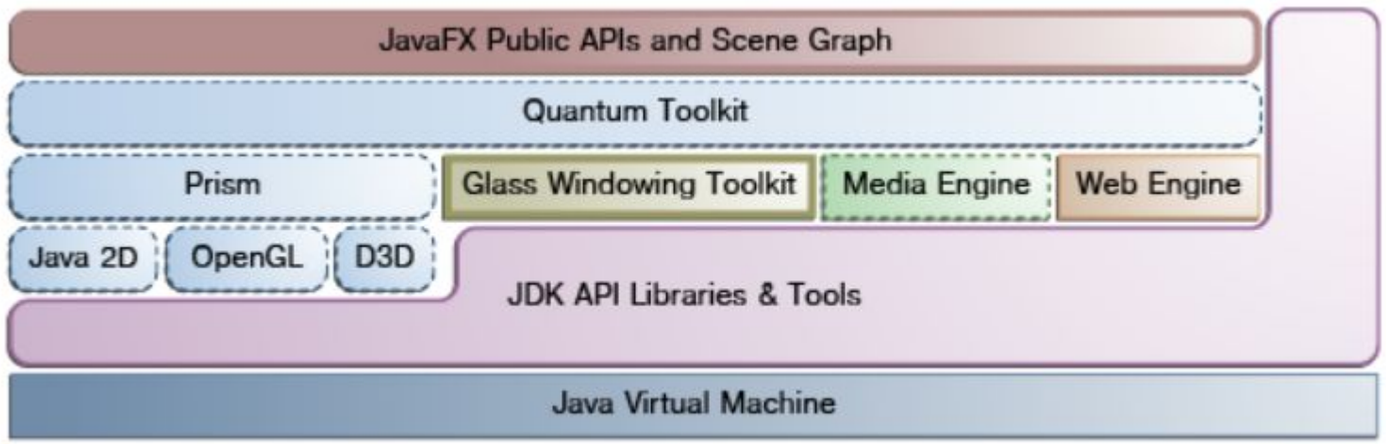
- The look and feel of JavaFX applications can be customized using **Cascading Style Sheets (CSS)**.
- Cascading Style Sheets (CSS) separate **appearance** and **style (UI)** from **implementation** (business logic).

JavaFX FXML and SceneBuilder

- One can develop the **UI** in the **FXML scripting language** and use Java code for the **application logic**.
- One can also design UIs using **JavaFX Scene Builder**.
- As you design the UI, Scene Builder creates **FXML markup** that can be ported to an IDE so that developers can add the business logic.

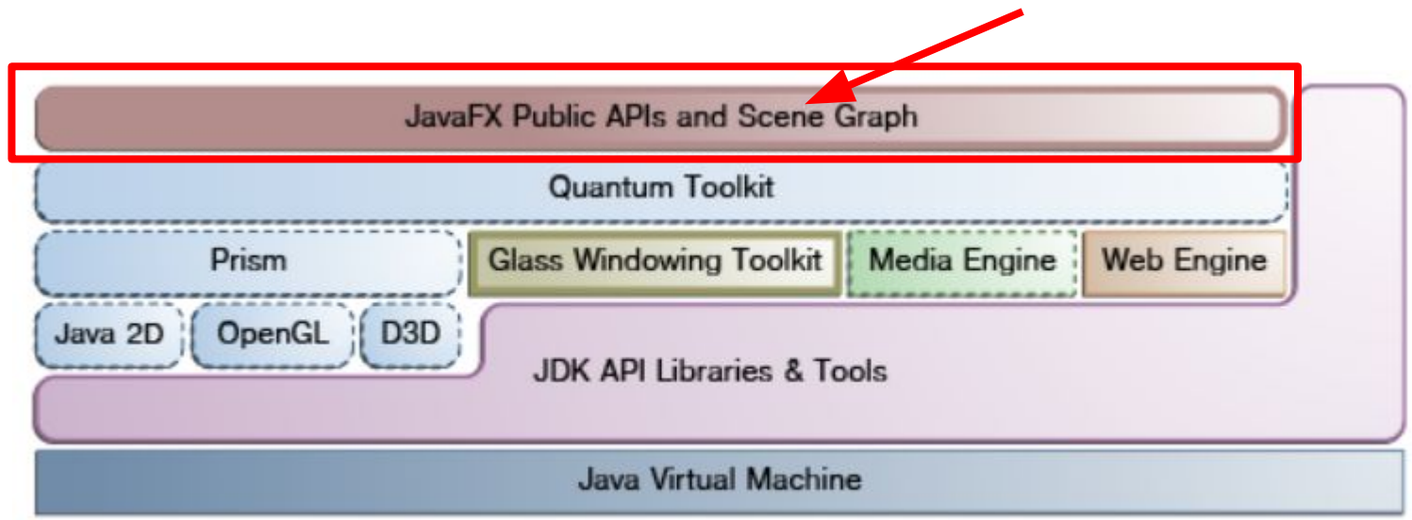
JavaFX Architecture

JavaFX Architectural Components



- Shown above are the prominent **architectural components** of JavaFX.
- You will invoke the **JavaFX public APIs** in your project code.
- The API implementation involves several **internal subcomponents** (**not exposed to the user**):
  - A JavaFX high performance graphics engine, called **Prism**.
  - A small and efficient windowing system, called **Glass**.
  - A media engine, and
  - A web engine.

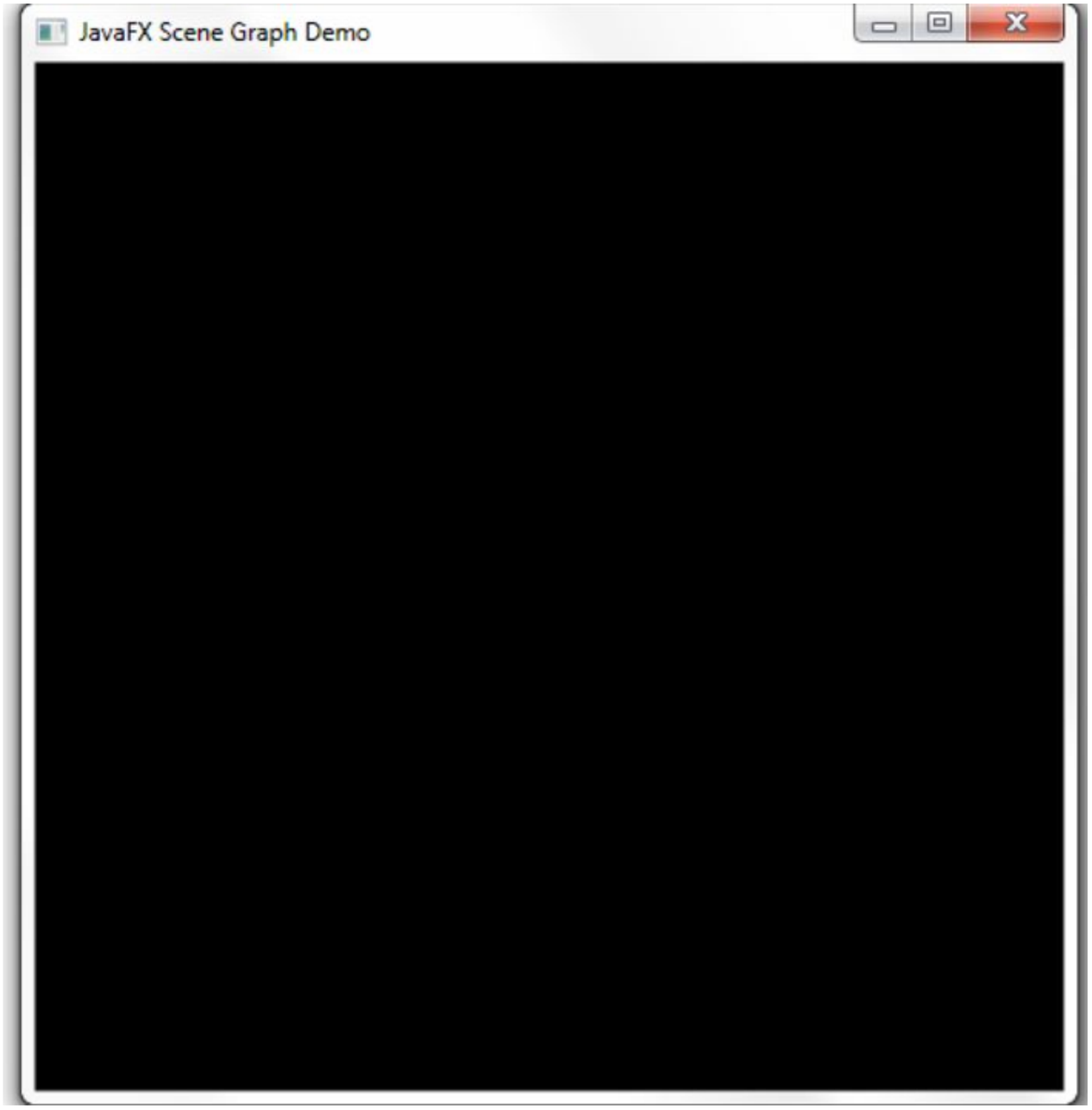
Scene Graph



- The **JavaFX scene graph** is the starting point for constructing a JavaFX application.
- It is a **hierarchical tree of nodes** that represents all of the visual elements of the application's user interface.
- Working with the Scene Graph: https://docs.oracle.com/javase/8/javafx/scene-graph-tutorial/scenegraph.htm#JFXSG107

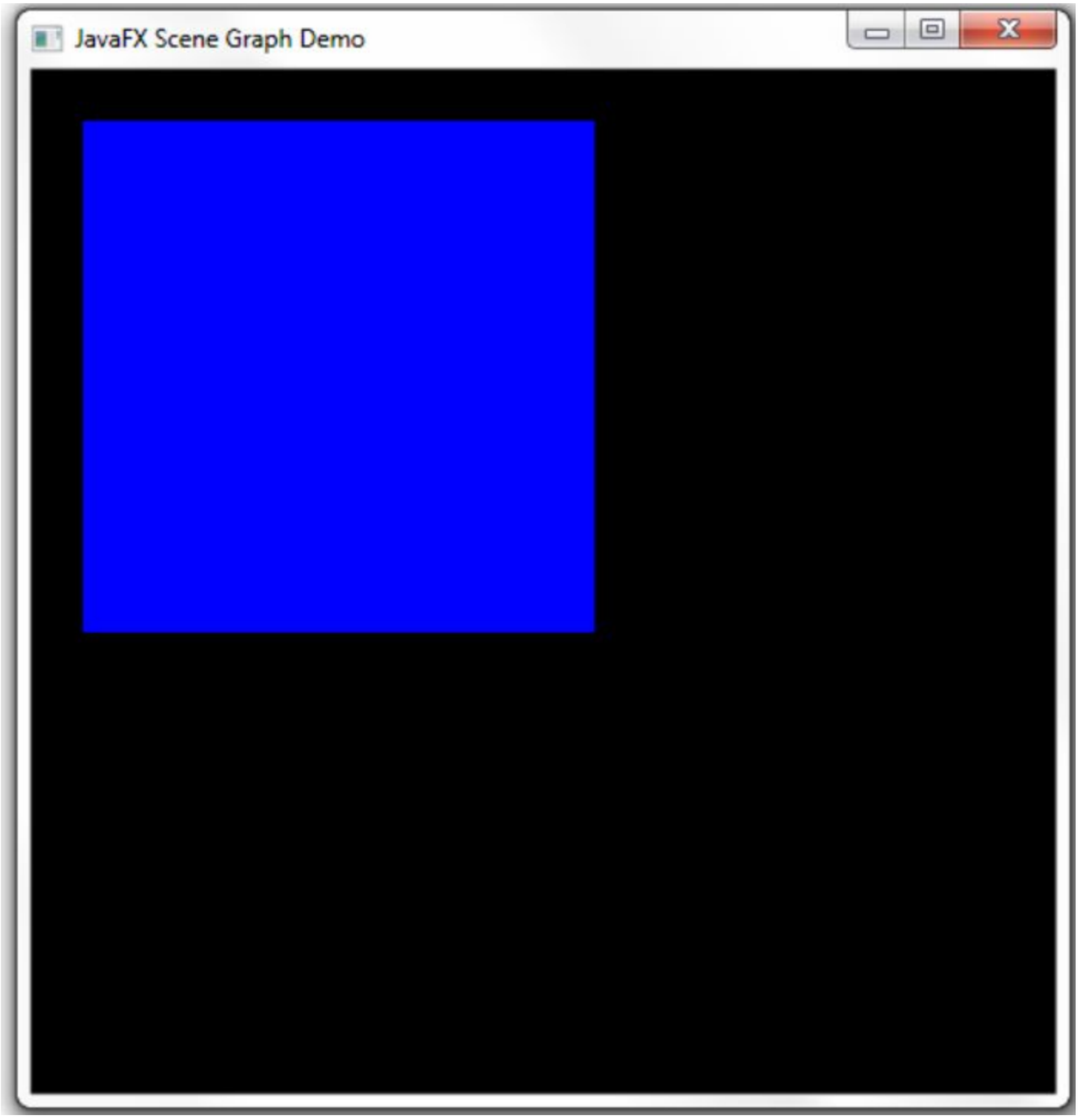
Scene Graph: Only Root Node

```java
package scenegraphdemo;

import javafx.application.Application;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.paint.Color;
import javafx.stage.Stage;

public class Main extends Application {

   @Override
   public void start(Stage stage) {
       Group root = new Group();
       Scene scene = new Scene(root, 500, 500, Color.BLACK);
       stage.setTitle("JavaFX Scene Graph Demo");
       stage.setScene(scene);
       stage.show();
   }

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

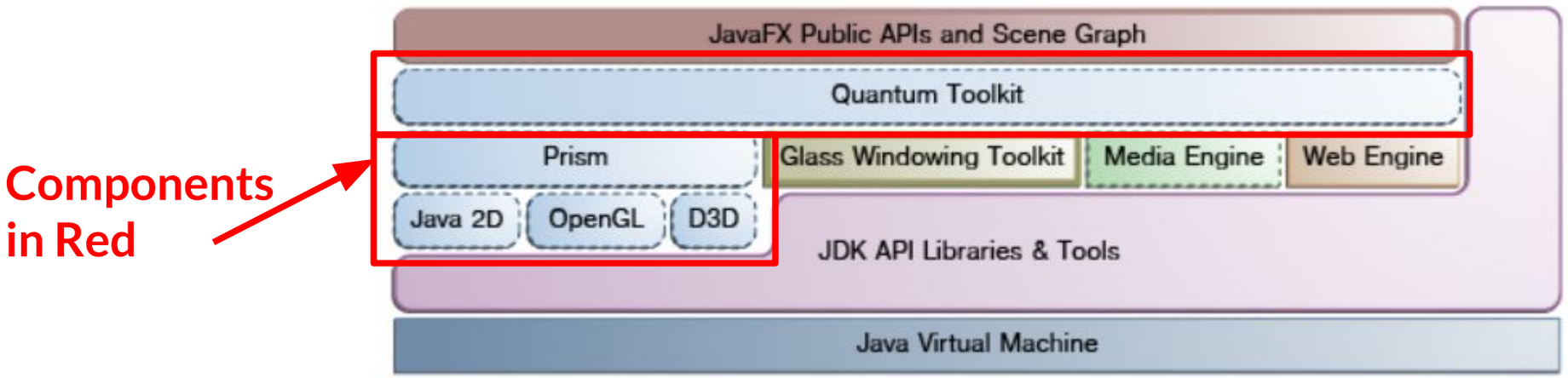


Scene Graph: Root Node + One Leaf Node

```java
package scenegraphdemo;

import javafx.application.Application;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.paint.Color;
import javafx.scene.shape.Rectangle;
import javafx.stage.Stage;

public class Main extends Application {

   @Override
   public void start(Stage stage) {
       Group root = new Group();
       Scene scene = new Scene(root, 500, 500, Color.BLACK);

       Rectangle r = new Rectangle(25,25,250,250);
       r.setFill(Color.BLUE);
       root.getChildren().add(r);

       stage.setTitle("JavaFX Scene Graph Demo");
       stage.setScene(scene);
       stage.show();
   }

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

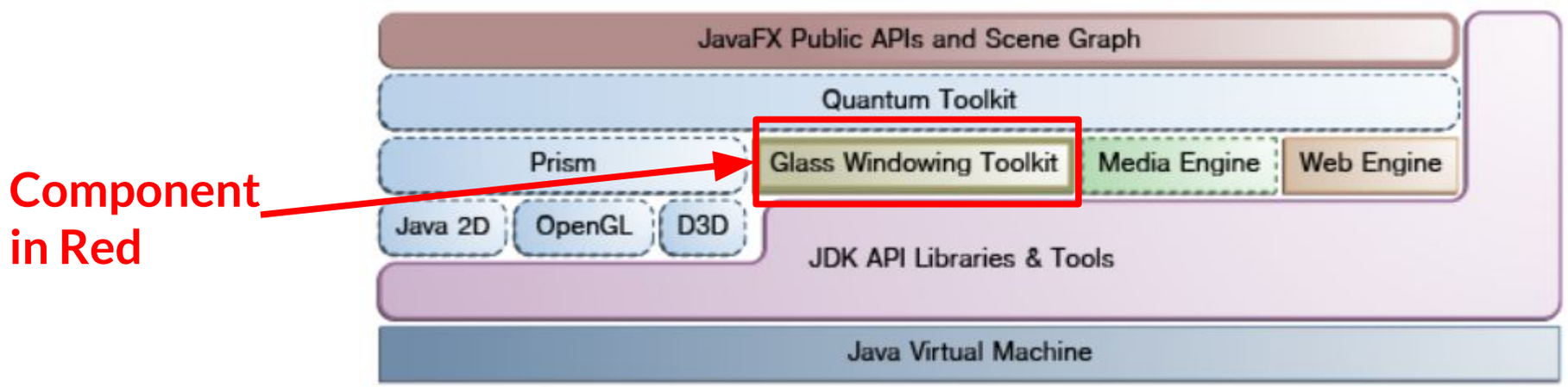


Graphics System



- The JavaFX Graphics System supports both 2-D and 3-D scene graphs.
- **Prism** processes render jobs and is responsible for rasterization and rendering of JavaFX scenes.

Glass Windowing Toolkit



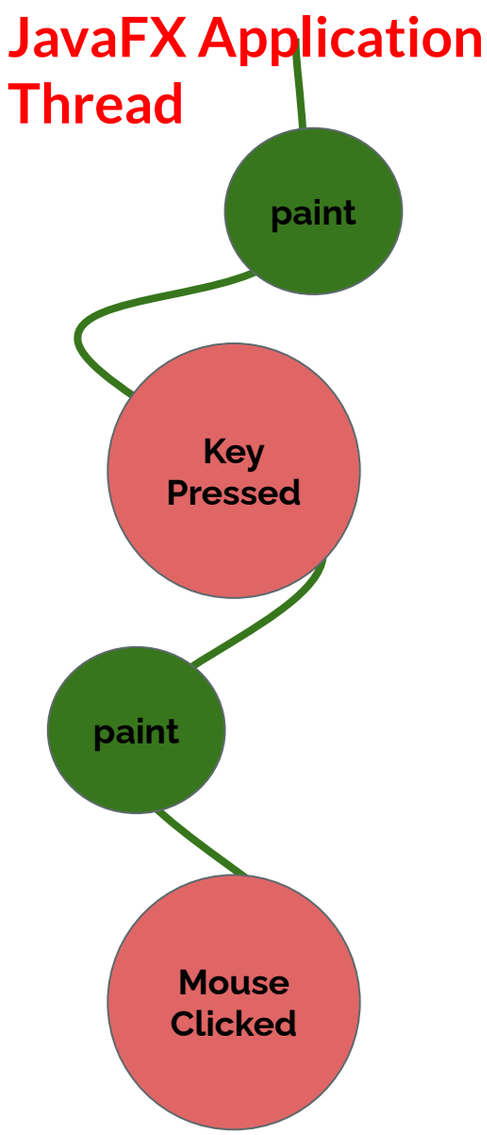
- The **Glass Windowing Toolkit** connects the JavaFX platform to the native operating system and provides native operating services, such as managing the windows, timers, and surfaces.

- The **Glass toolkit** is also responsible for managing the event queue.

Threads (1/2)

- The **JavaFX scene graph** can only be accessed and modified from the **UI thread** also known as the **JavaFX Application thread**.

- Implementing long-running tasks on the JavaFX Application thread inevitably makes an application UI unresponsive.

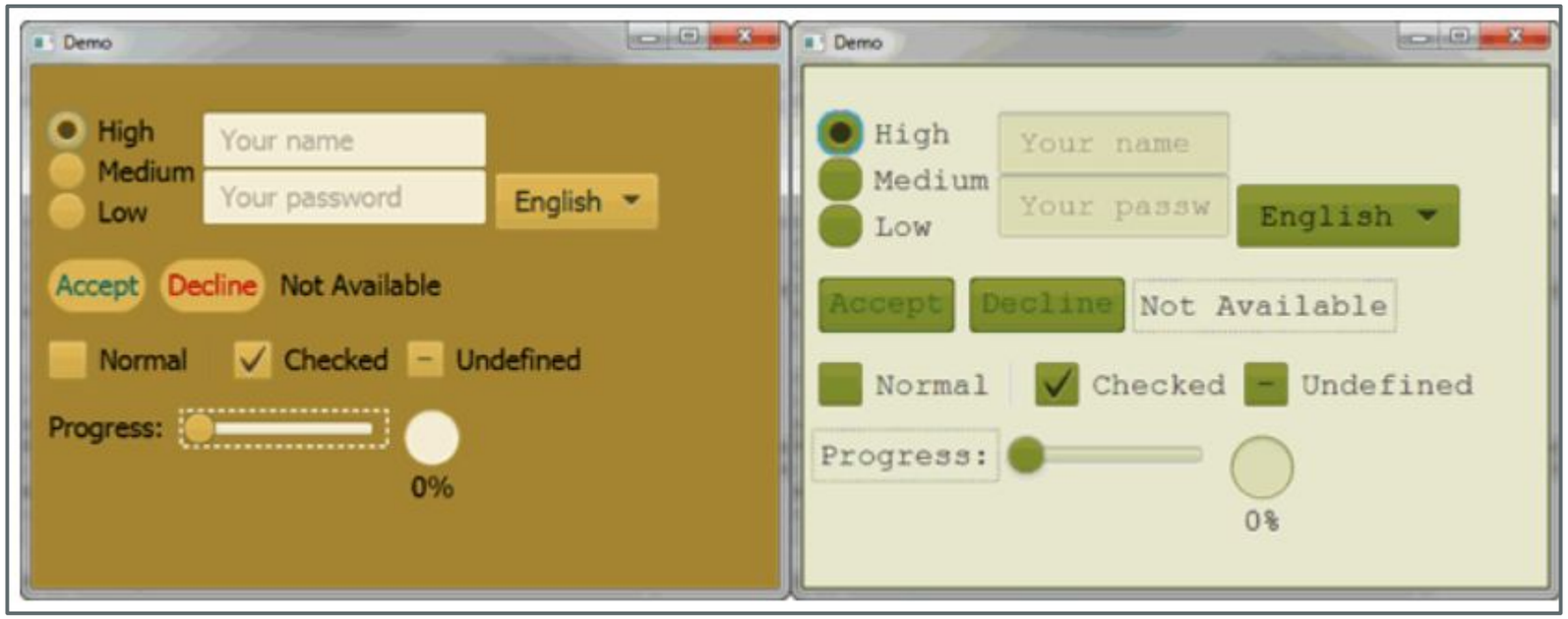


Threads (2/2)

- A best practice is to do the time consuming task in a background thread.
- Let the JavaFX Application thread process user events.
- The task in the background thread can modify the JavaFX scene graph using the **Platform.runLater() technique**.
- **Note multithreading not covered in this module.** Refer the article below about multithreading in JavaFX:

https://docs.oracle.com/javafx/2/threads/jfxpub-threads.htm

CSS



- JavaFX CSS allows applying customized styling to the user interface without changing application's source code.
- CSS can be applied to any node in the JavaFX scene graph.
- CSS styles can also be easily assigned to the scene at runtime, allowing an application's appearance to dynamically change.

https://docs.oracle.com/javase/8/javafx/user-interface-tutorial/css_tutorial.htm#JFXUI733

UI Controls

- The JavaFX UI controls available through the JavaFX API are built by using nodes in the scene graph.
- These controls reside in the javafx.scene.control package.

Layout

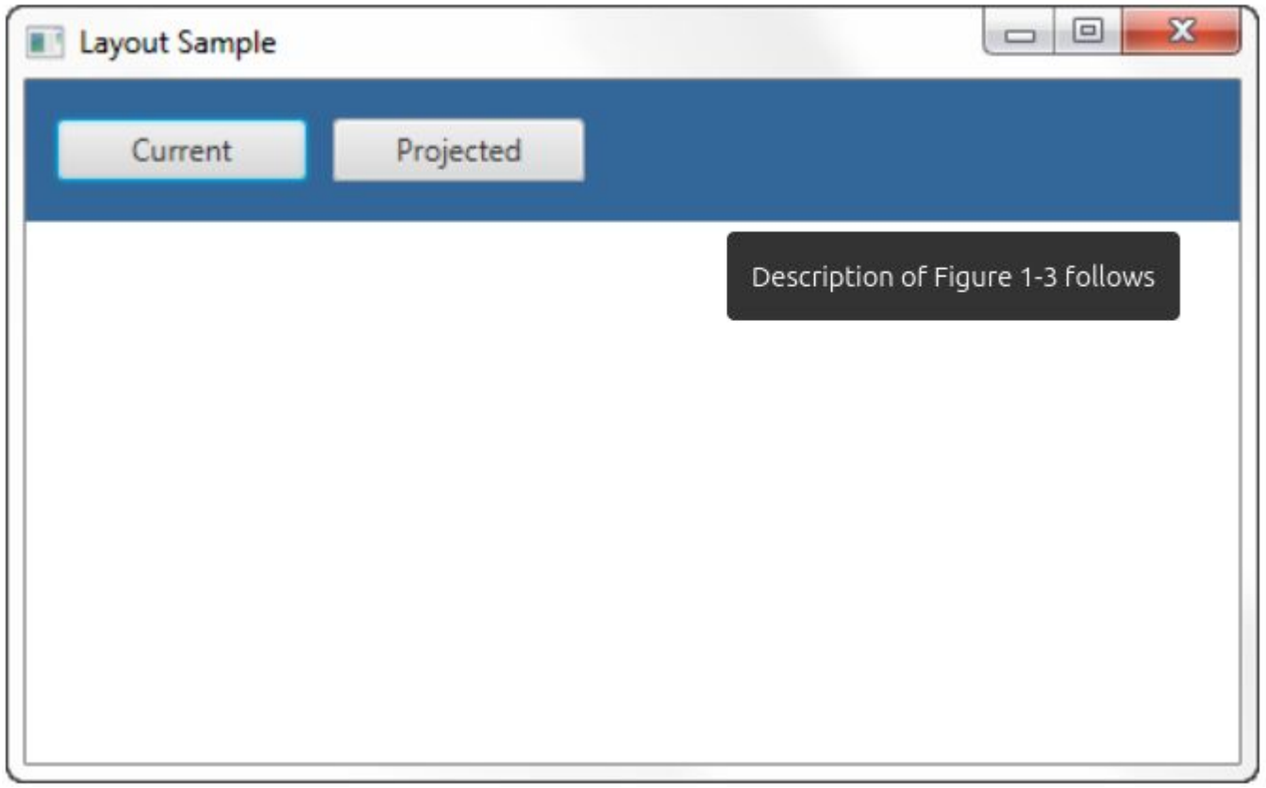
- One can develop the **UI** in the **FXML scripting language** and use Java code for the **application logic**.
- One can also design UIs using **JavaFX Scene Builder**.
- As you design the UI, Scene Builder creates **FXML markup** that can be ported to an IDE so that developers can add the business logic.

For more about layouts,
https://docs.oracle.com/javase/8/javafx/layout-tutorial/index.html

Sample Layouts (1/2)

Sample Border Pane

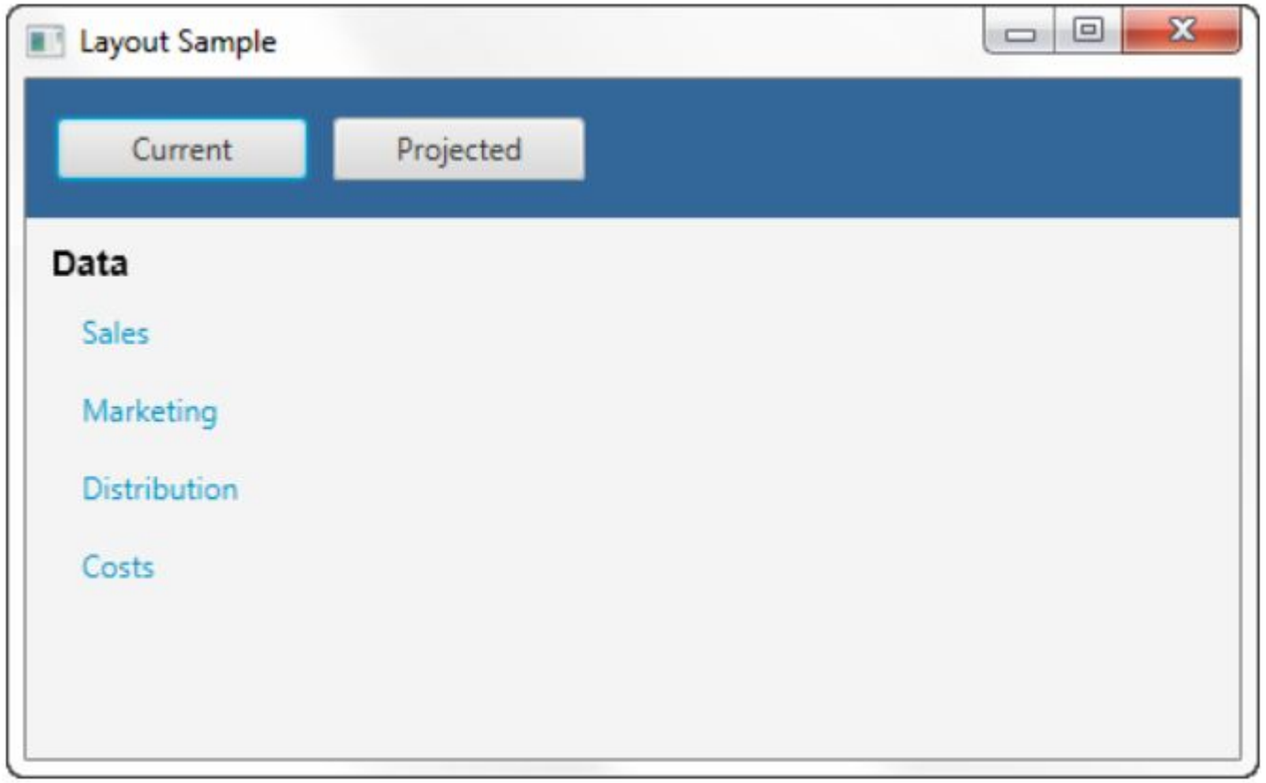
Sample HBox Pane



HBox in Border

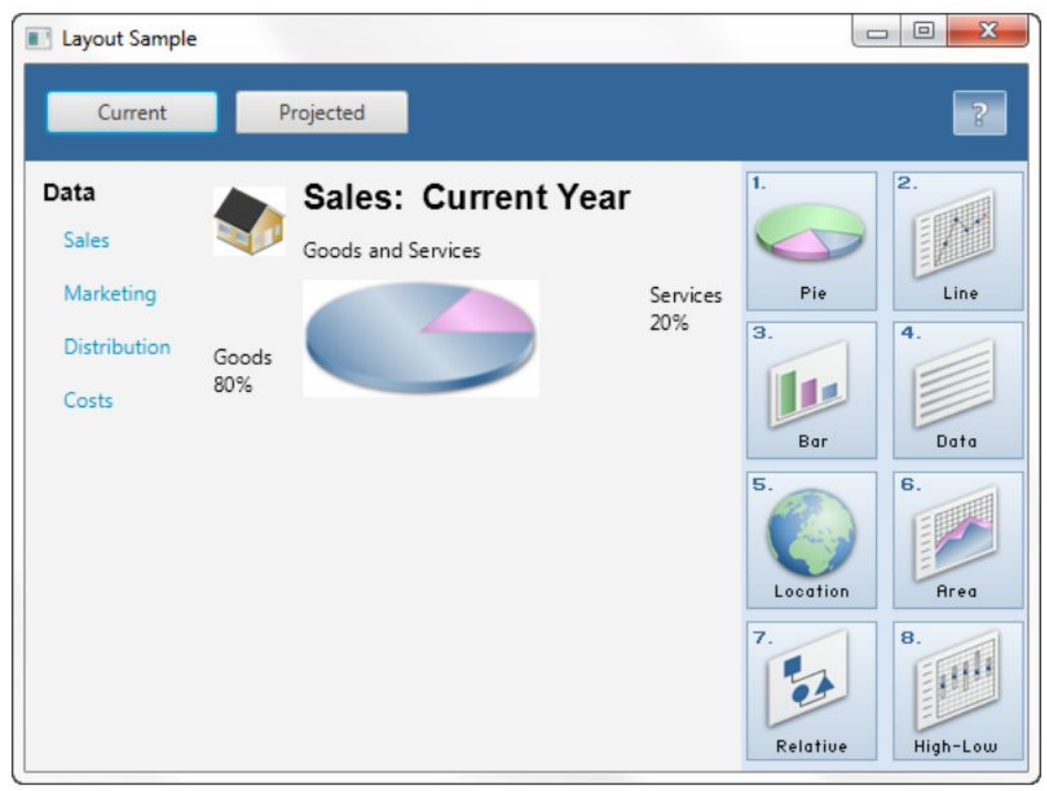
VBox

Sample Layouts (2/2)



VBox in Border Pane

Horizontal Flow Pane

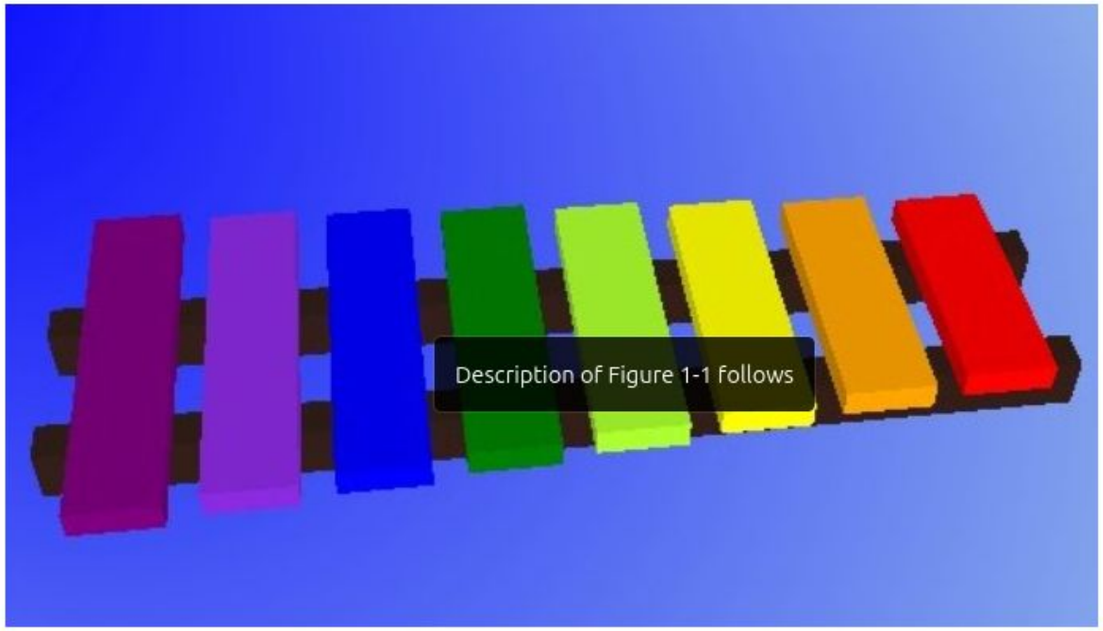


Flow Pane in Border Pane

2D and 3D Transformations

- Each node in the JavaFX scene graph can be transformed in the x-y coordinate using the following **javafx.scene.transform** classes:
  - **translate** - Move a node from one place to another along the x, y, z planes relative to its initial position.
  - **scale** - Resize a node to appear either larger or smaller in the x, y, z planes, depending on the scaling factor.
  - **rotate** - Rotate a node about a specified pivot point of the scene.

For more about transformations, https://docs.oracle.com/javase/8/javafx/visual-effects-tutorial/transforms.htm#JFXTE139



Xylophone transformed

JavaFX Sample Demo

JavaFX FXMLTableView Demo

```
manumachu@system76-pc:~/comp20050/JavaFX$ ls FXMLTableView
build.xml  License.txt  manifest.mf  nbproject  src
manumachu@system76-pc:~/comp20050/JavaFX$
```

https://docs.oracle.com/javase/8/javafx/fxml-tutorial/index.html

- **FXML** is an XML-based language that provides the structure for building a user interface separate from the application logic of your code.
- **FXML** is particularly useful for user interfaces that have large, complex scene graphs, forms, data entry, or complex animation.

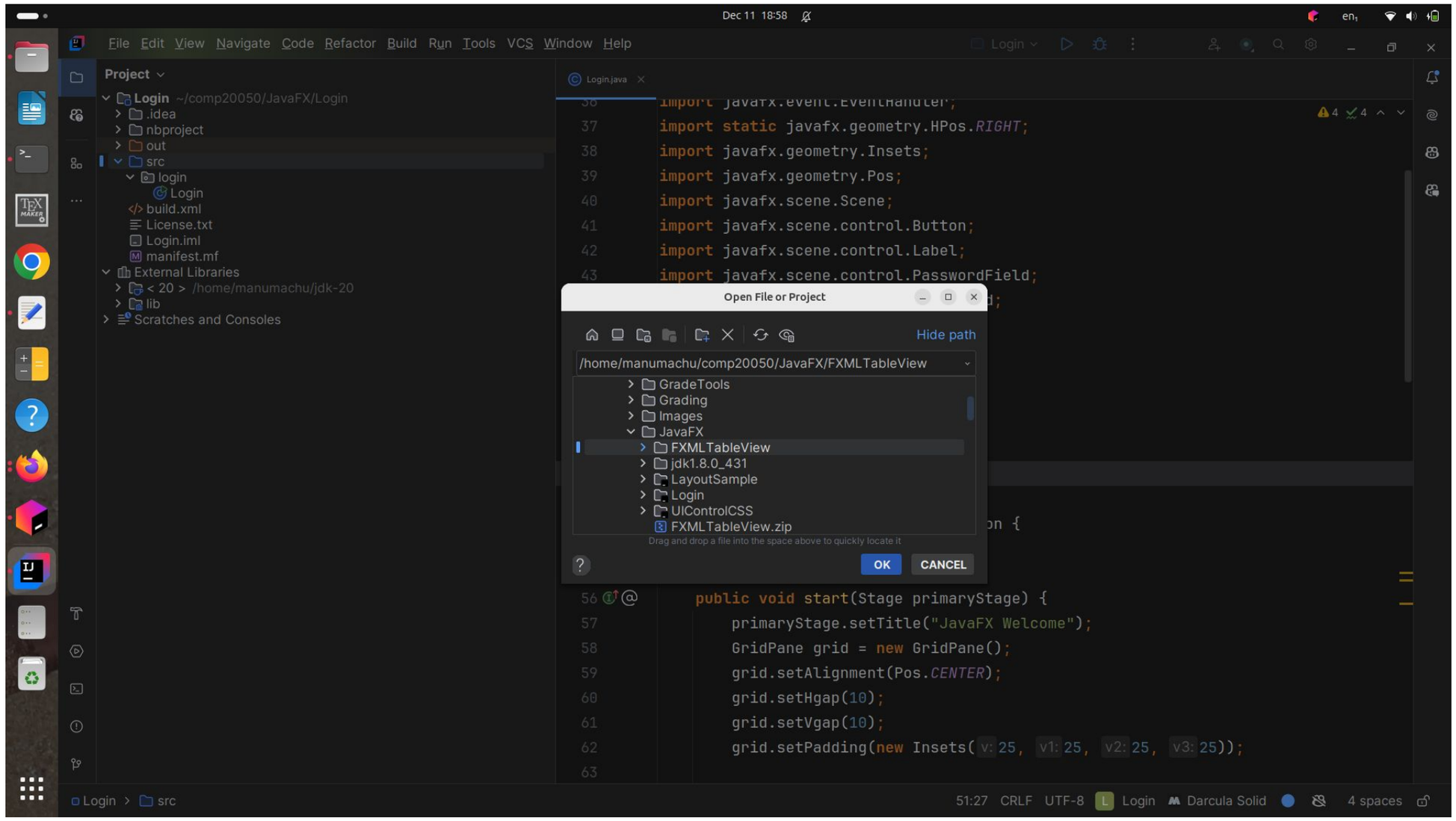
JavaFX FXML and SceneBuilder

```
manumachu@system76-pc:~/comp20050/JavaFX$ ls FXMLTableView
build.xml  License.txt  manifest.mf  nbproject  src
manumachu@system76-pc:~/comp20050/JavaFX$
```

https://docs.oracle.com/javase/8/javafx/fxml-tutorial/index.html

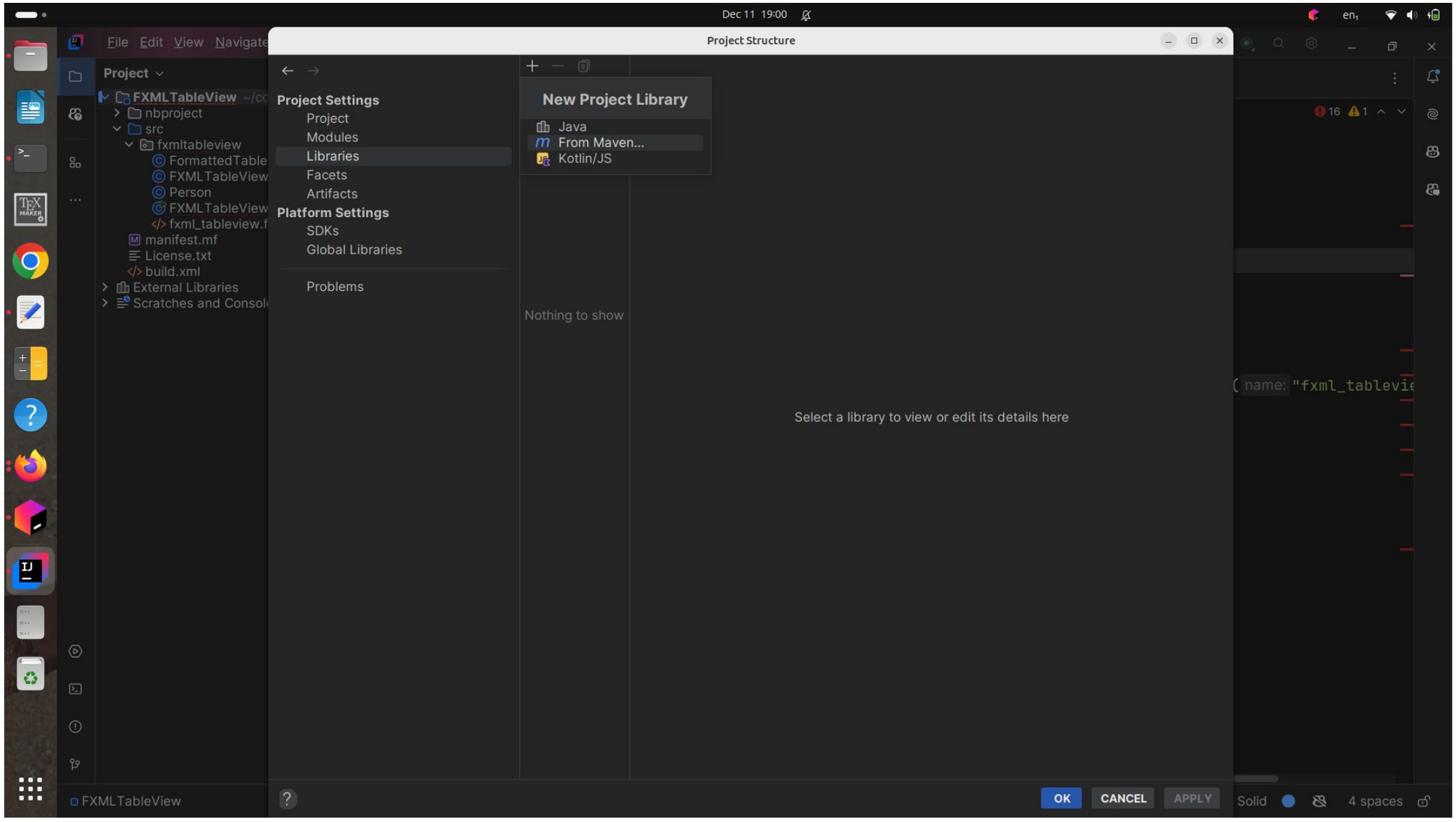
- You can work directly with FXML OR you can use **Scene Builder** (covered later).
- **SceneBuilder** is a design tool that generates the FXML source code as you define the user interface for your application.
- You can further edit FXML files, generated by Scene Builder, in any text or XML editor.

FXMLTableView: IntelliJ Open Project



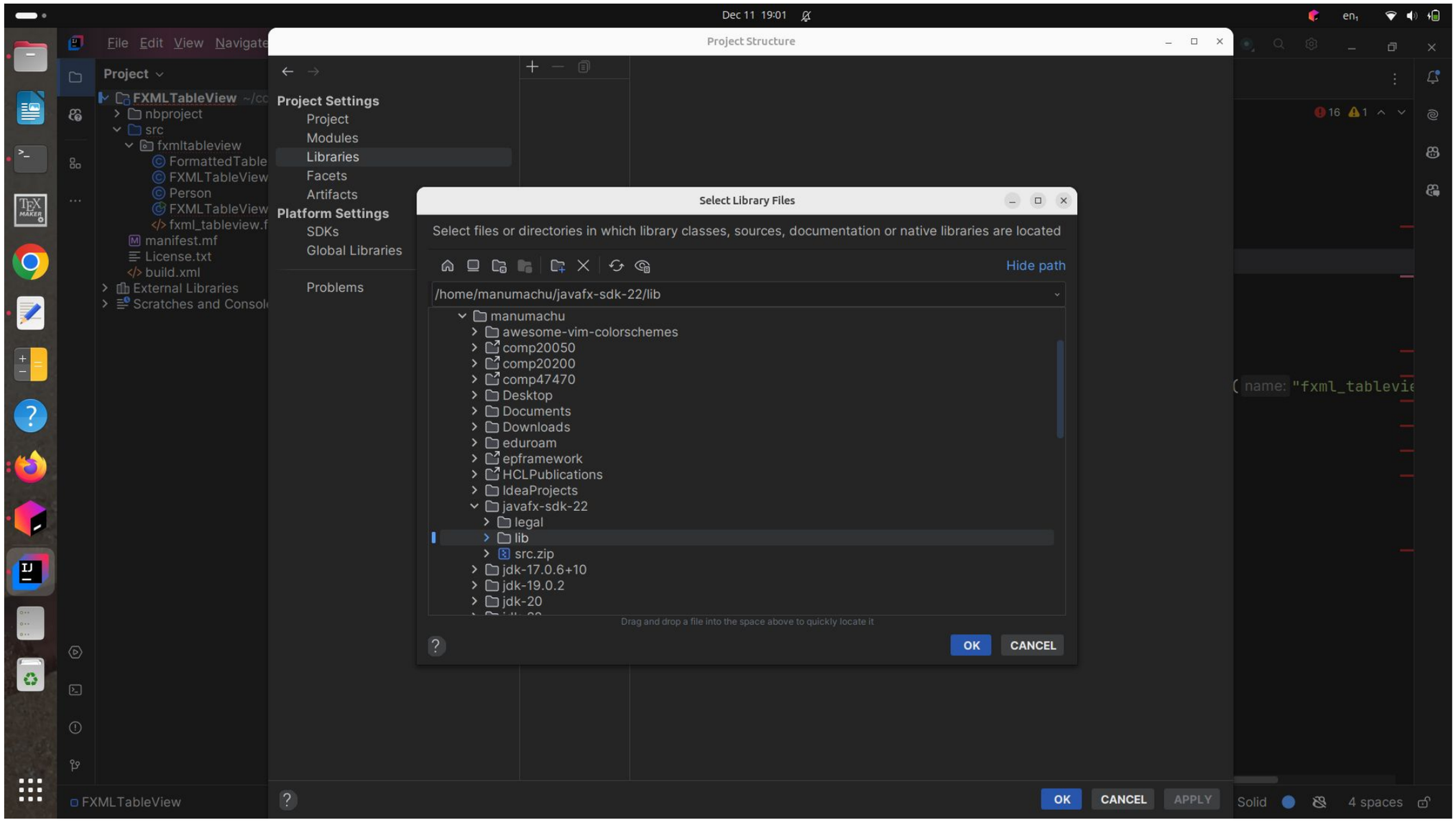
- **File -> Open.**
- Navigate to the directory containing the demo and click **OK**.

FXMLTableView: Add JavaFX Library (1/3)



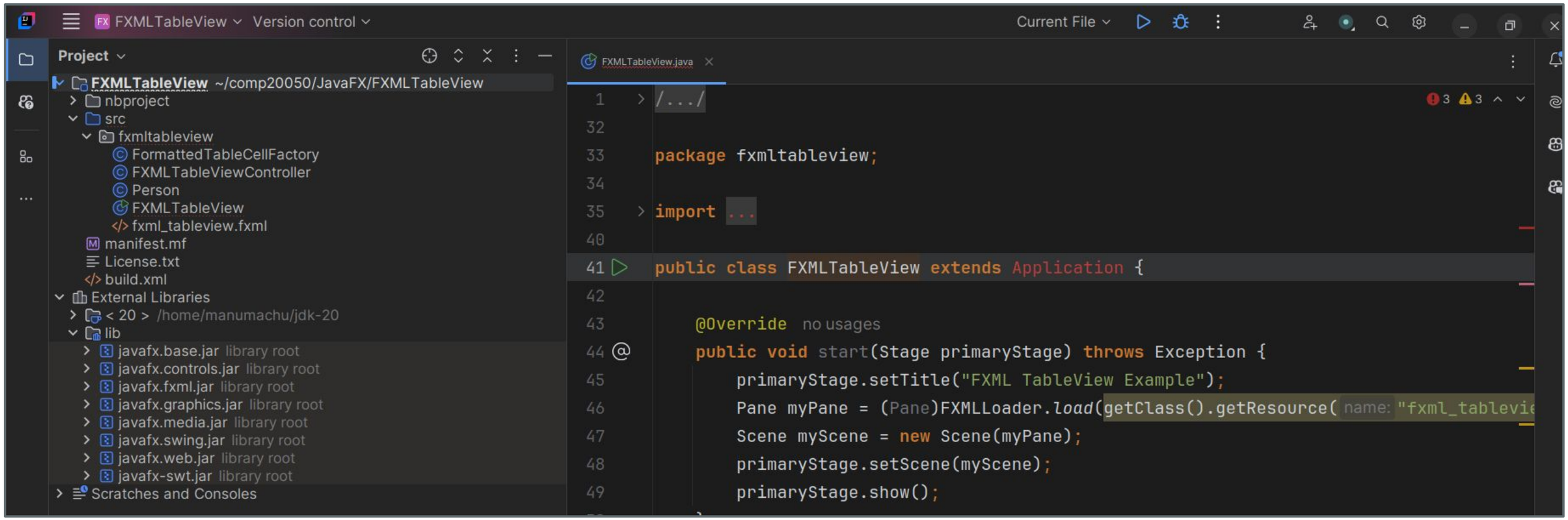
- **File -> Project Structure -> Project Settings -> Libraries.**
- Click **+**.

FXMLTableView: Add JavaFX Library (2/3)



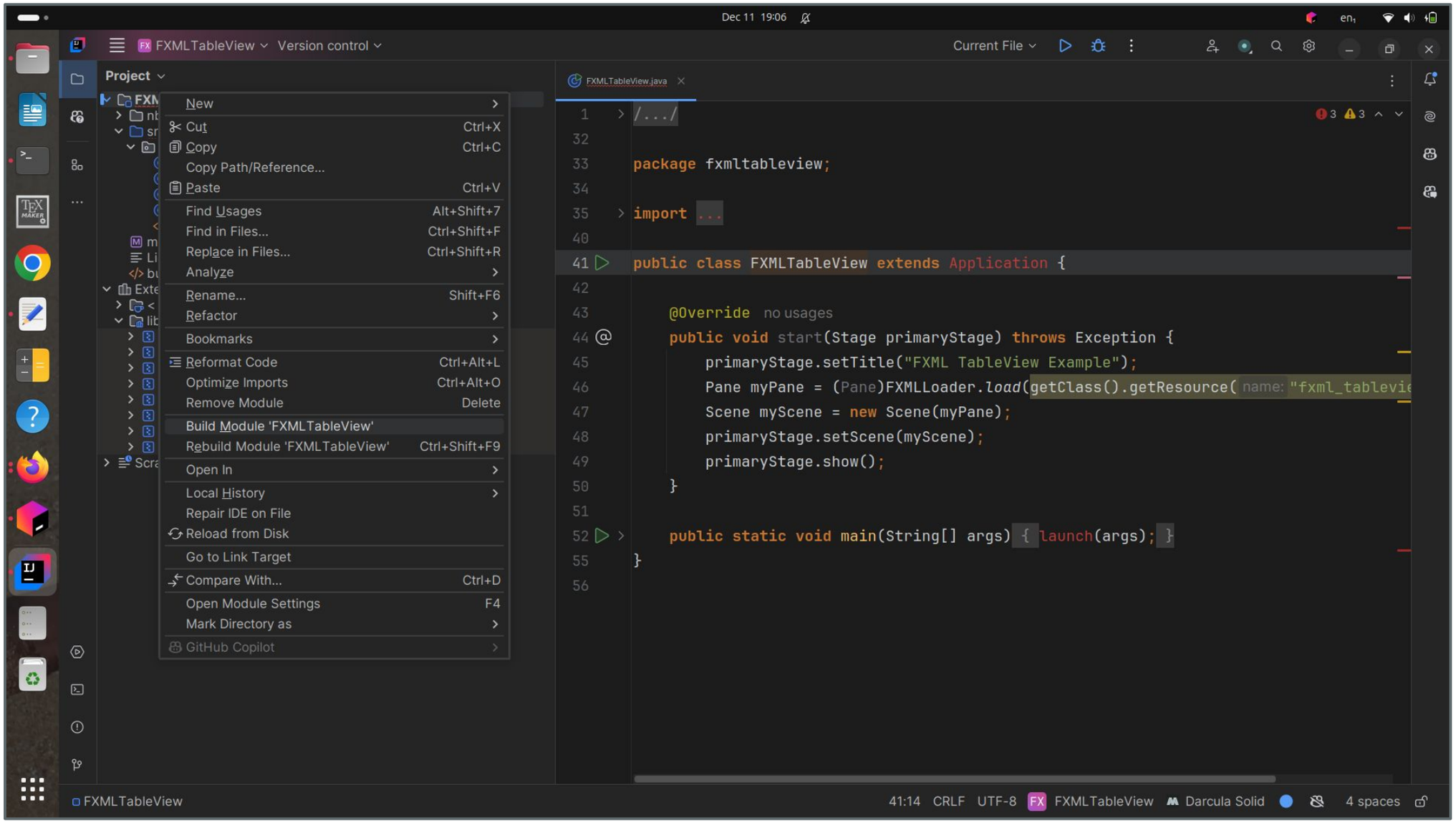
- Navigate to the javafx library installation folder and select by clicking **OK**.

FXMLTableView: Add JavaFX Library (3/3)



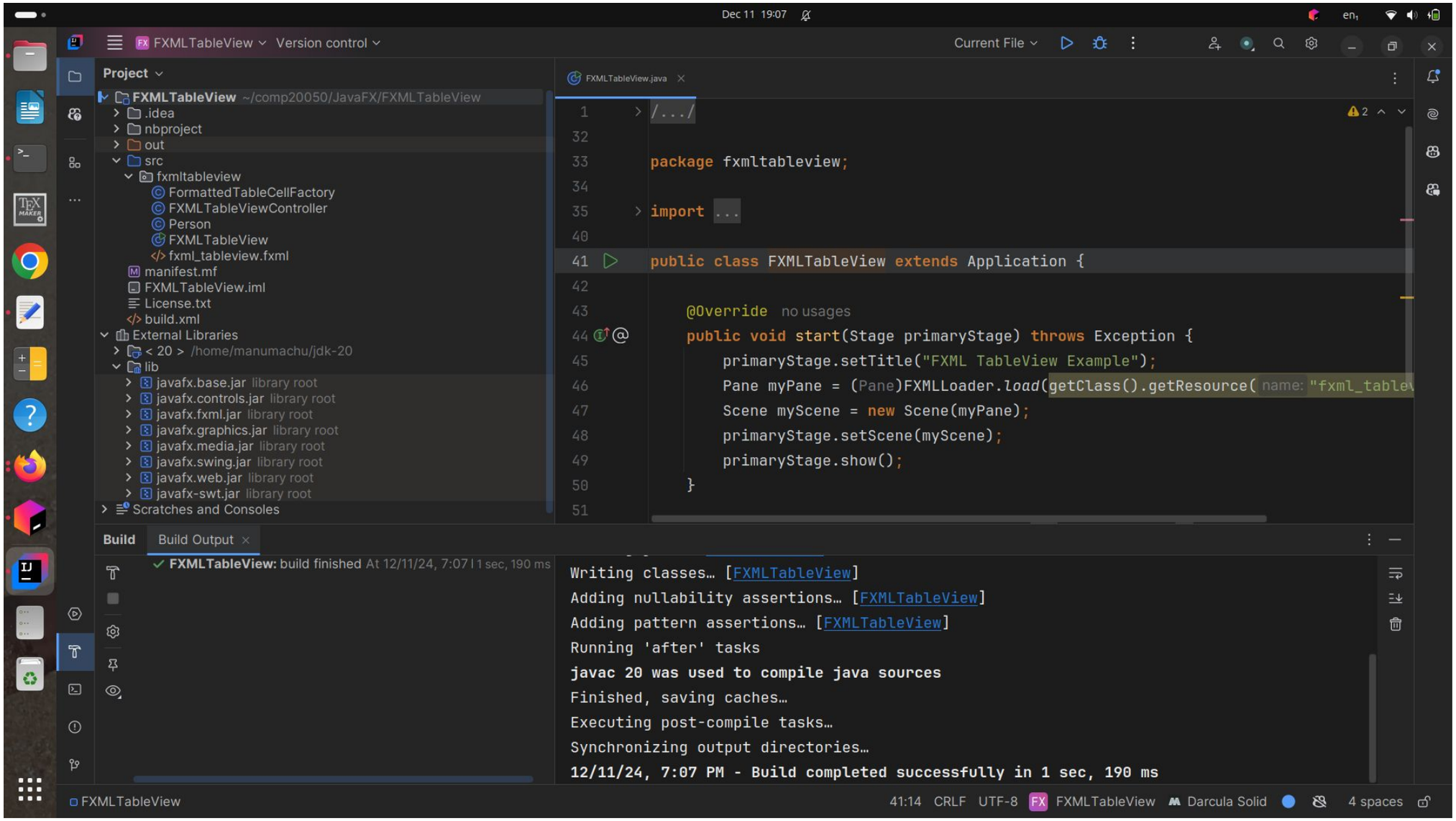
- You will see the JavaFX libraries under the **External Libraries** in the Project folder.

FXMLTableView: Build Project (1/2)



- Right click **FXMLTableView** and select **Build Module FXMLTableView**.

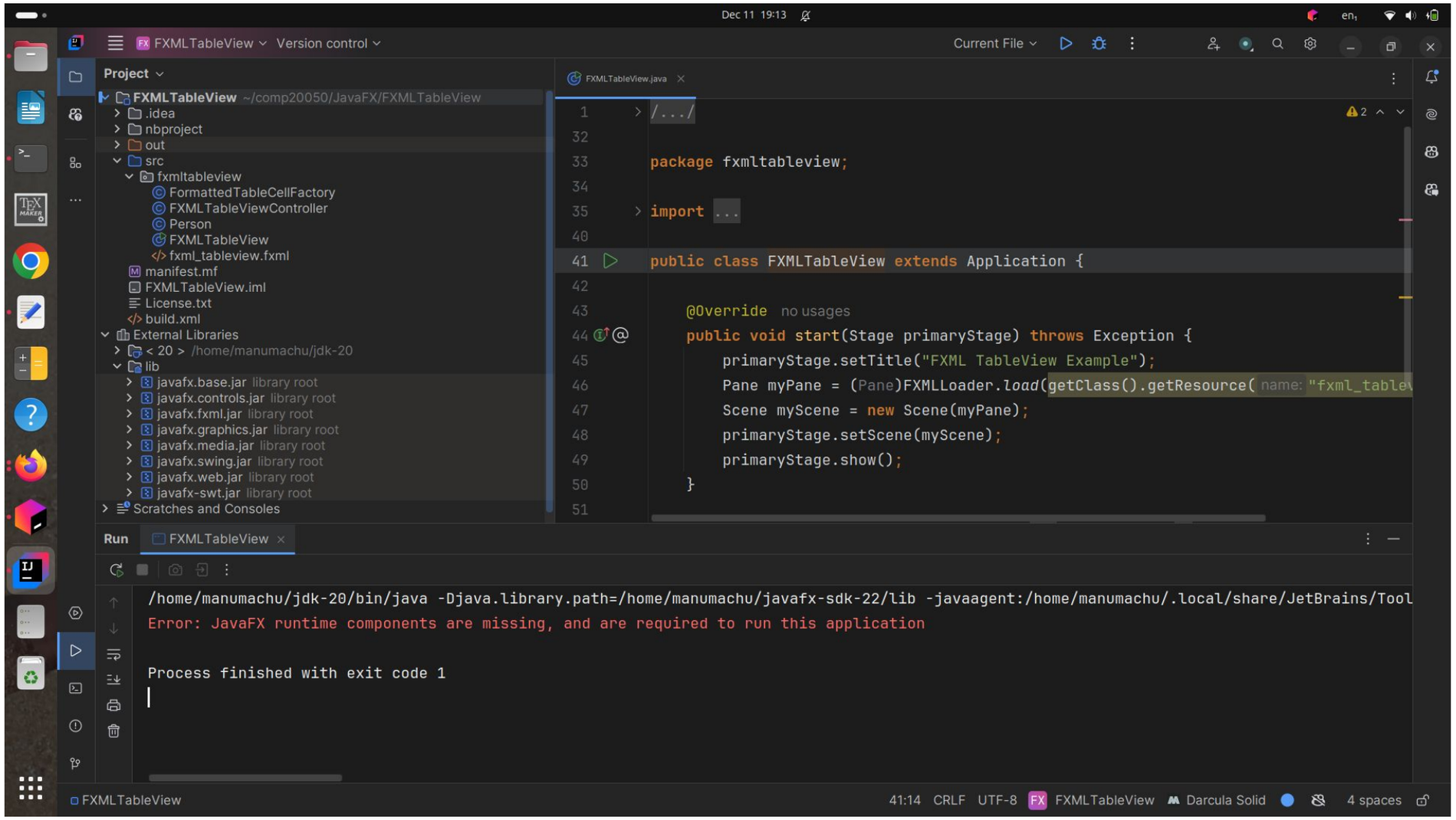
FXMLTableView: Build Project (2/2)



- In Tool Windows -> Build, you will see the build output.

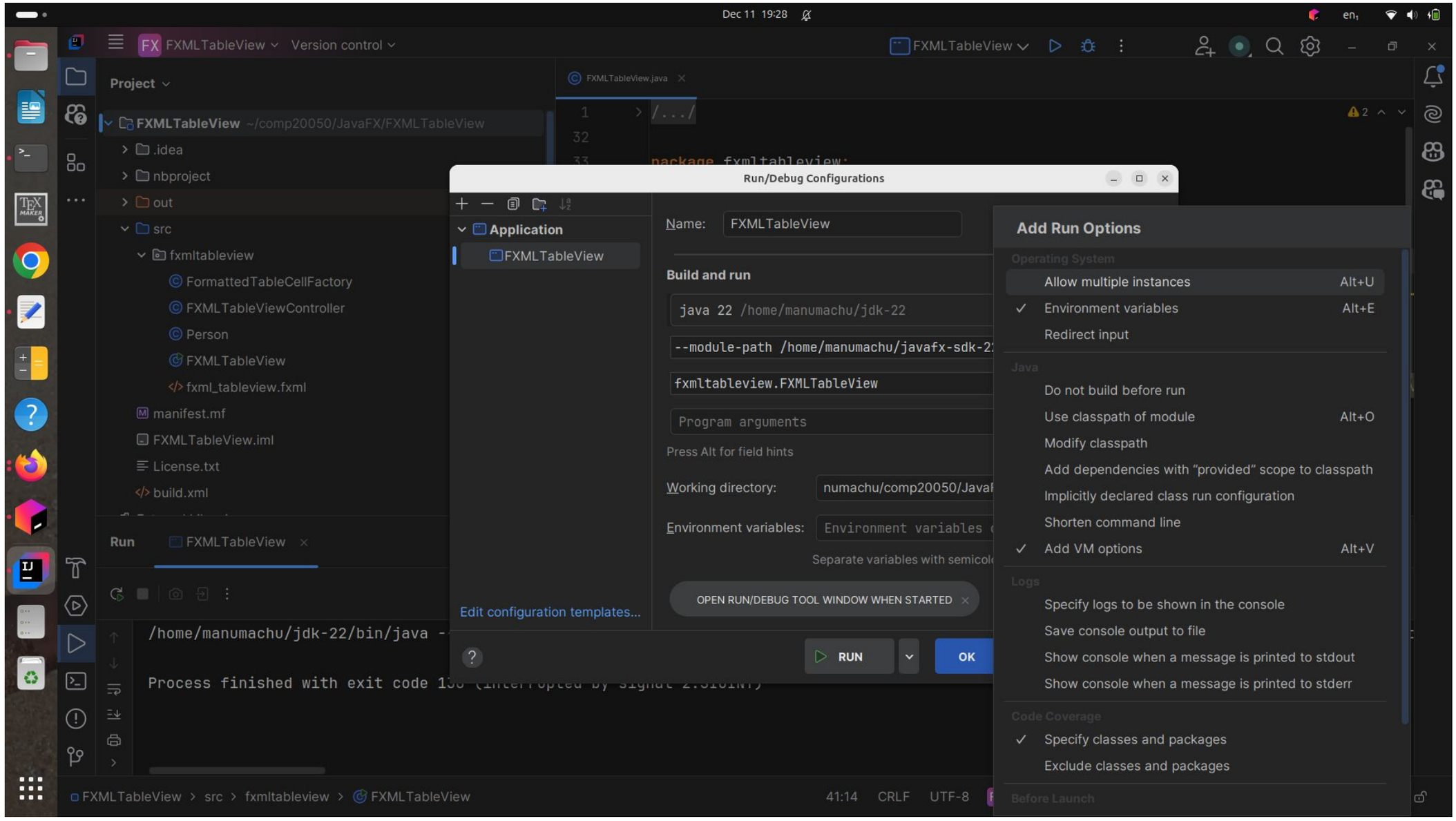
FXMLTableView: Run Project (1/4)

Run Button



- When you click **Run**, you may face the above error.
- You need to provide extra parameters.

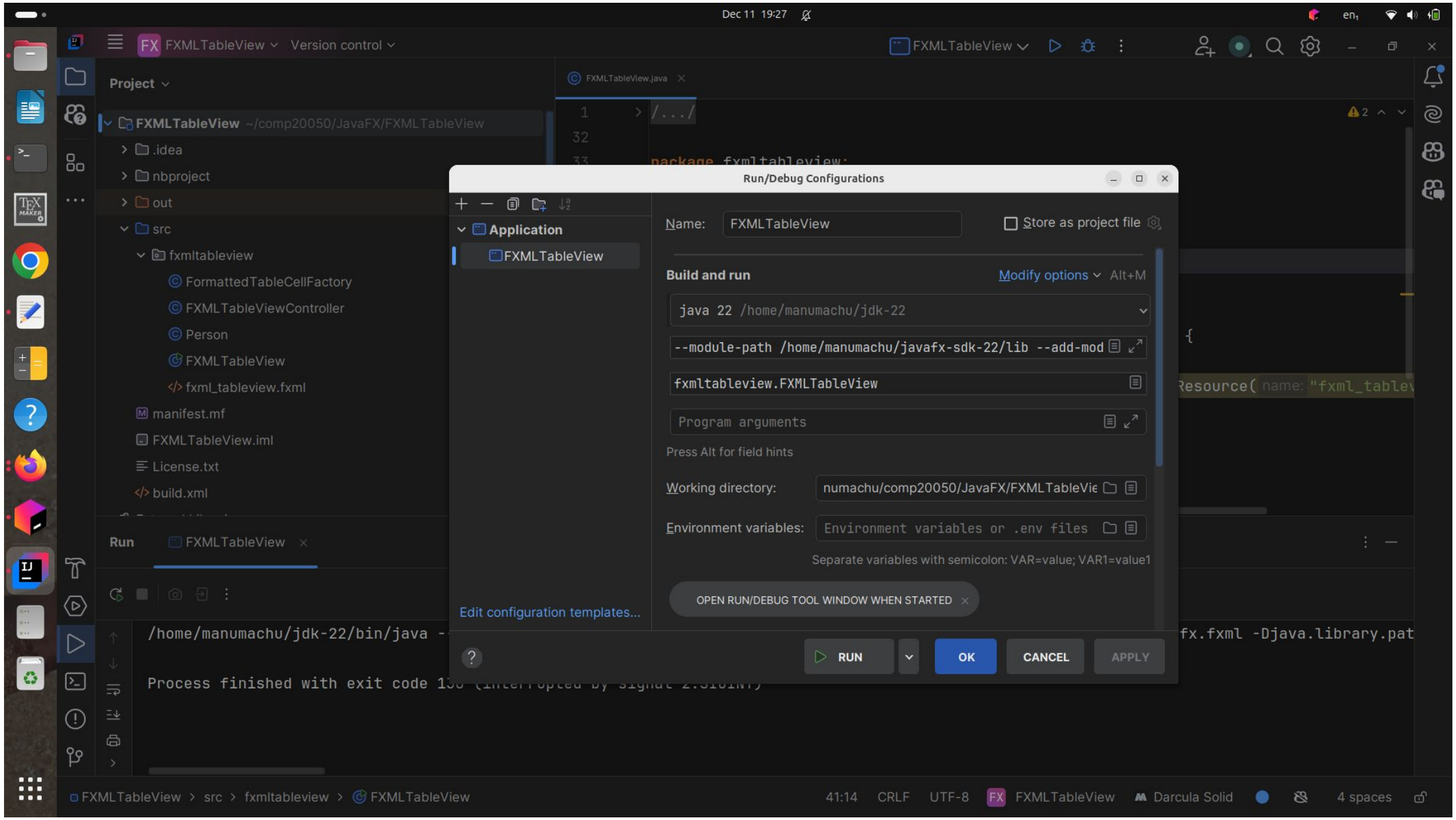
FXMLTableView: Run Project (2/4)



- Click **Add VM Options**. Provide the following in the box provided.

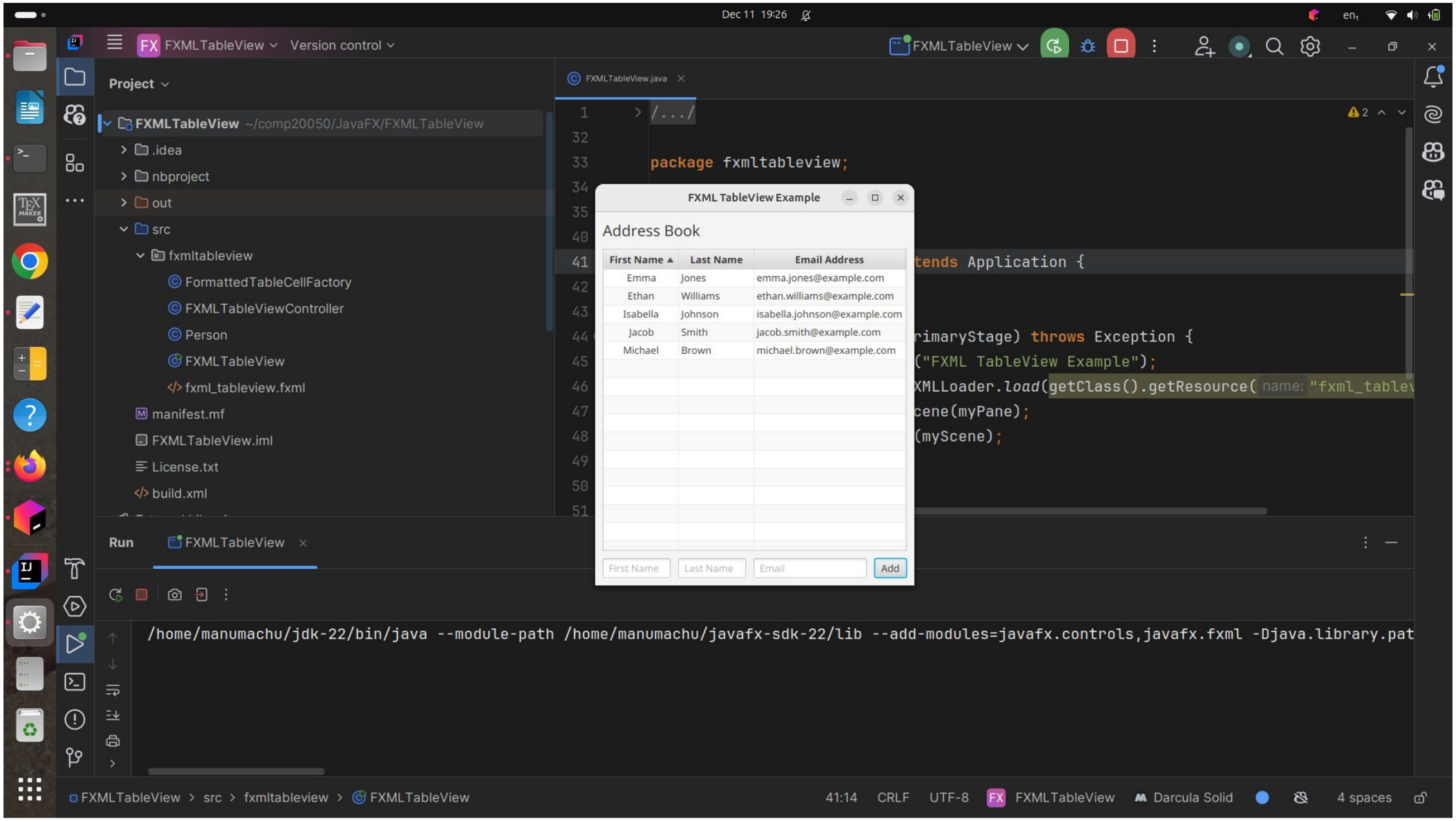
--module-path
/home/manumachu/javafx-sdk-22/lib
--add-modules=javafx.controls,**javafx.fxml**

FXMLTableView: Run Project (3/4)



- Click **Apply** and **Run**.

FXMLTableView: Run Project (4/4)



Q&A

To follow...

JavaFX Scene Builder