

# COMP3111: Software Engineering

## Java FX

### Learning Outcomes:

- Able to create a simple JavaFX UI using Scene Builder.
- Able to program on a simple GUI program using JavaFX.

### Guided Lab Exercises

*Environment: Eclipse (Version: 2020-12) with Java Development Kit (JDK 15) installed on Windows 10. The steps involved may be slightly different if you are using other versions of Eclipse, or Eclipse on Mac.*

Submission/Demo – Not required.

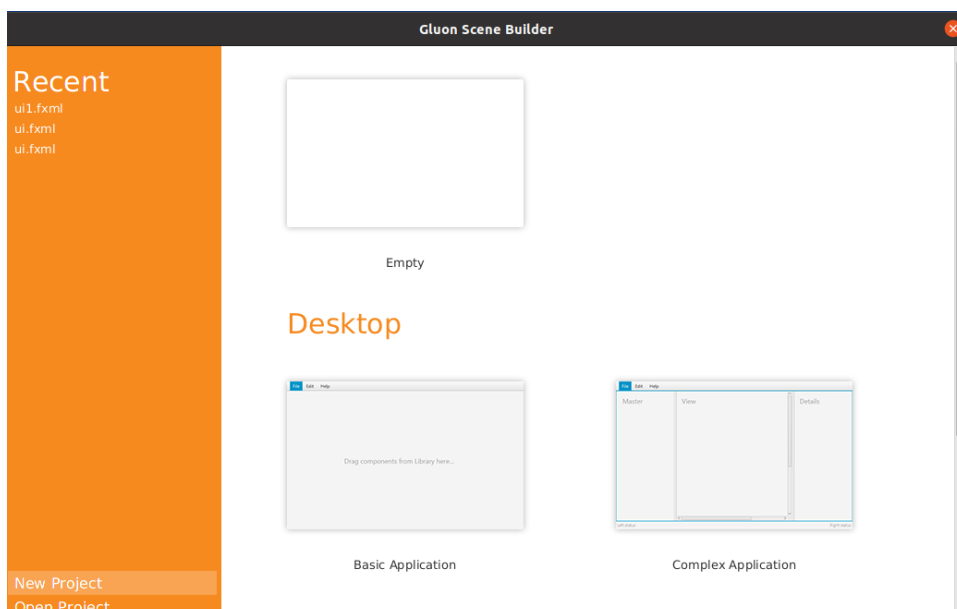
*JavaFX is a technology that helps you create a GUI environment. There are other Java GUI technologies like Swing / AWT. In our project, we select JavaFX.*

*In JavaFX, the UI design is saved in a XML file with the file extension “.fxml”. This file will be loaded to the program in run-time. In this lab we will let you create a fxml file and do some programming on Java to create a simple Java GUI application.*

### Exercise 1: Create a JavaFX UI using Scene Builder

**Step 1.1:** Download and install Scene Builder, a visual layout tool for developing JavaFX application, from <https://gluonhq.com/products/scene-builder/#download> Take note of the install path of Scene Builder.

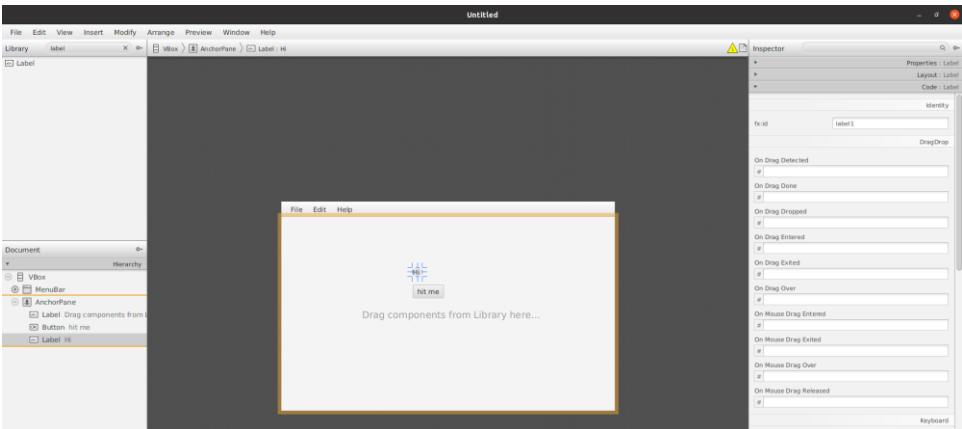
**Step 1.2:** Launch Scene Builder and select the “Basic Application” template to create a New Project



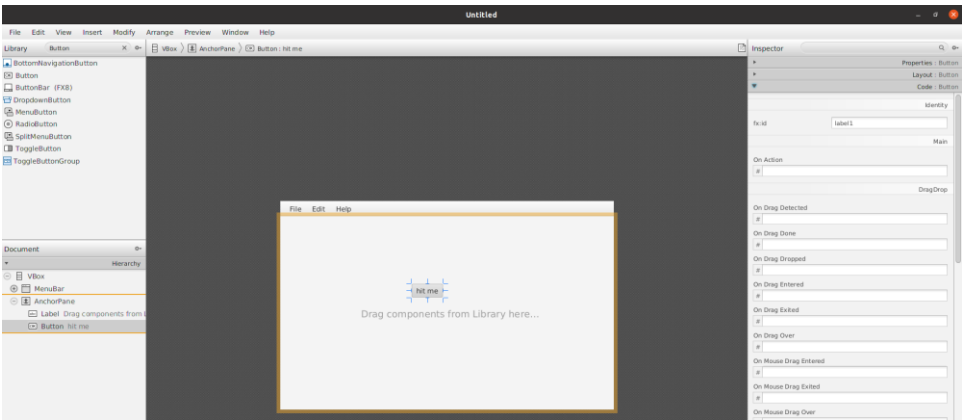
**Step 1.3:** Drag a “Button” control from the Library panel (on the left column) to the empty pane (in the middle). Then drag a “Label” control to the pane as well.



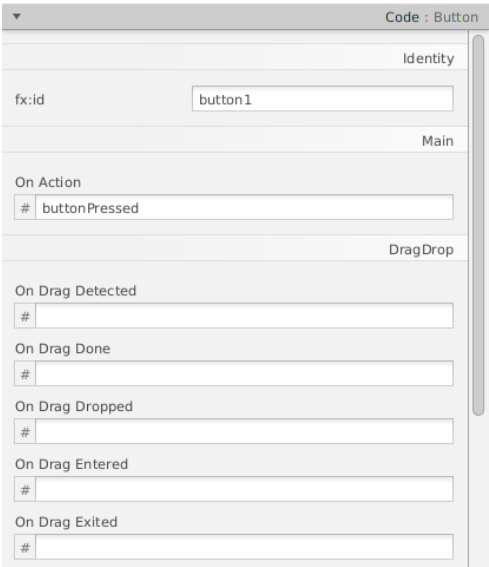
**Step 1.4:** From the Inspector panel (on the right column), you can change the attributes of the controls. Set the text on the button control and the label control to “hit me” and “Hi”, respectively.



**Step 1.5:** Under “Inspector > Code”, change the identity of the label as “label1”  
*Note: we will refer the label using the variable name “label1” in our Java code.*



**Step 1.6:** For the button, we change the identity of it as “button1” and also add an event “buttonPressed” under “Main > On Action”.



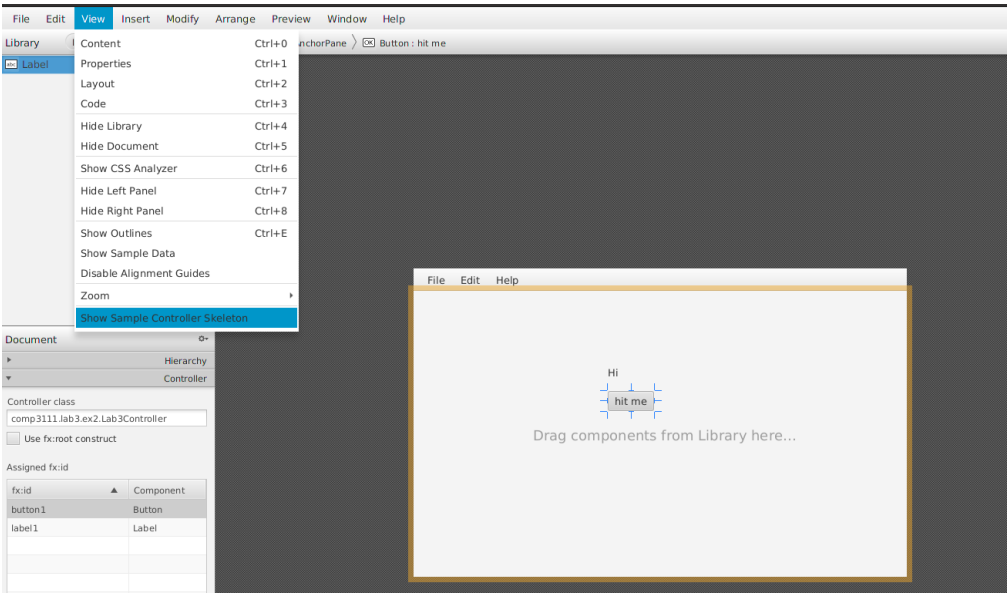
*Note: we will refer the button using the variable name “button1” and when the button is clicked, the function “buttonPressed” will be called.*

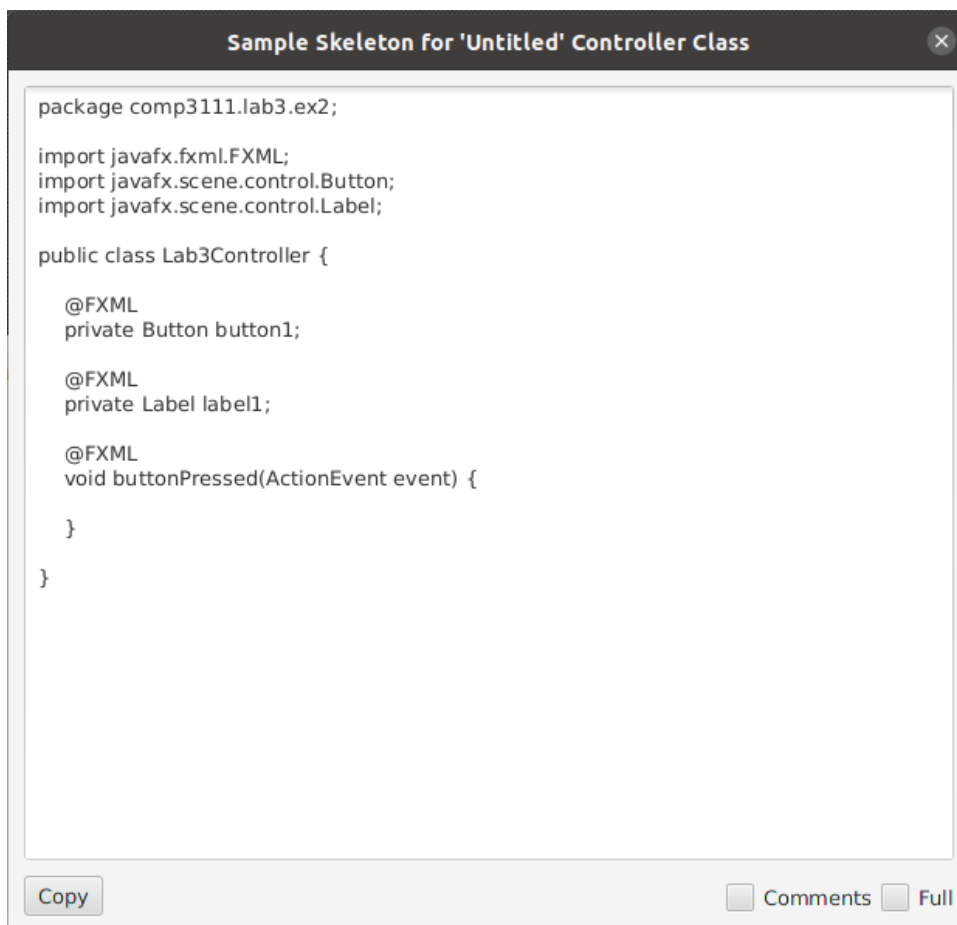
**Step 1.7:** On the Panel “Document” add “comp3111.lab3.ex2.Lab3Controller” to the field Controller.



*Note: we will use the class “Lab3Controller” to control the UI and to accept input from the user.*

**Step 1.8:** From the menu bar click “View > Show Sample Controller Skeleton”. Copy the code and we will paste it in our eclipse later.





```
package comp3111.lab3.ex2;

import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.Label;

public class Lab3Controller {

    @FXML
    private Button button1;

    @FXML
    private Label label1;

    @FXML
    void buttonPressed(ActionEvent event) {

    }

}
```

Copy Comments Full

**Step 1.9:** From the menu bar, click “File > Save” to save your work on your work folder or the Desktop. Name it as “ui.fxml”. We shall add this file to our Eclipse project in Exercise 2.

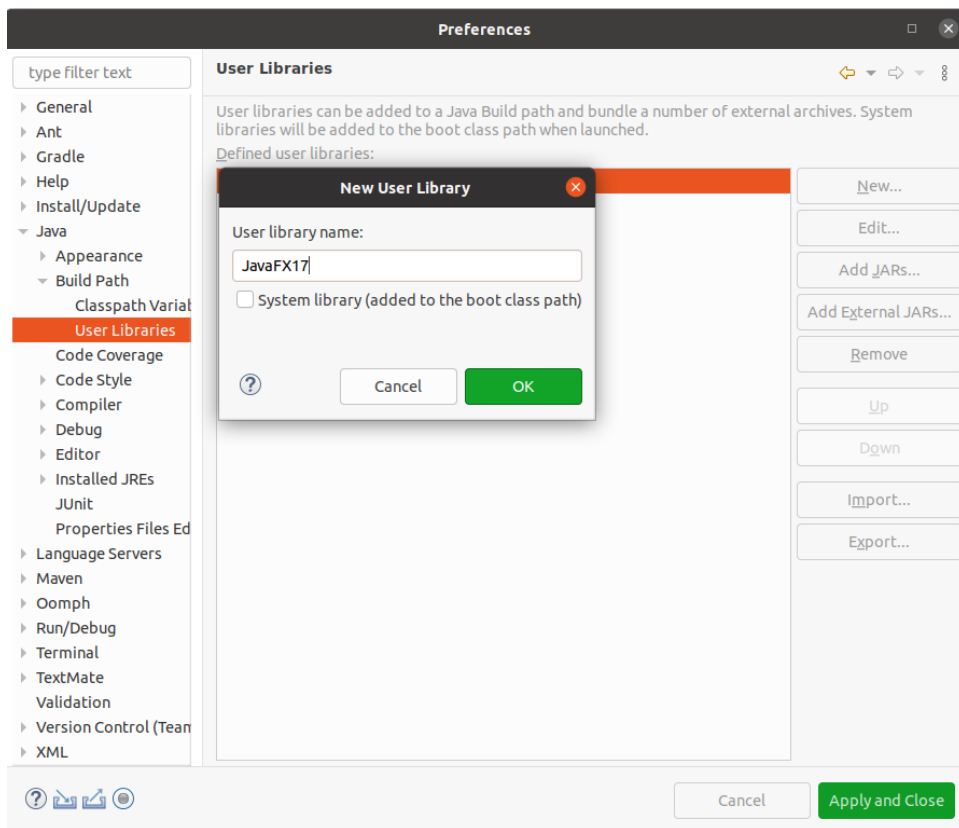
## Exercise 2: Develop JavaFX with Eclipse IDE

**Step 2.1:** Download JavaFX SDK from <https://gluonhq.com/products/javafx/> (version 11/15/17 or above), unzip the file, and put it inside a local folder, such as, “/home/user/OpenJFX/javafx-sdk-17.0.0.1”<sup>1</sup>.

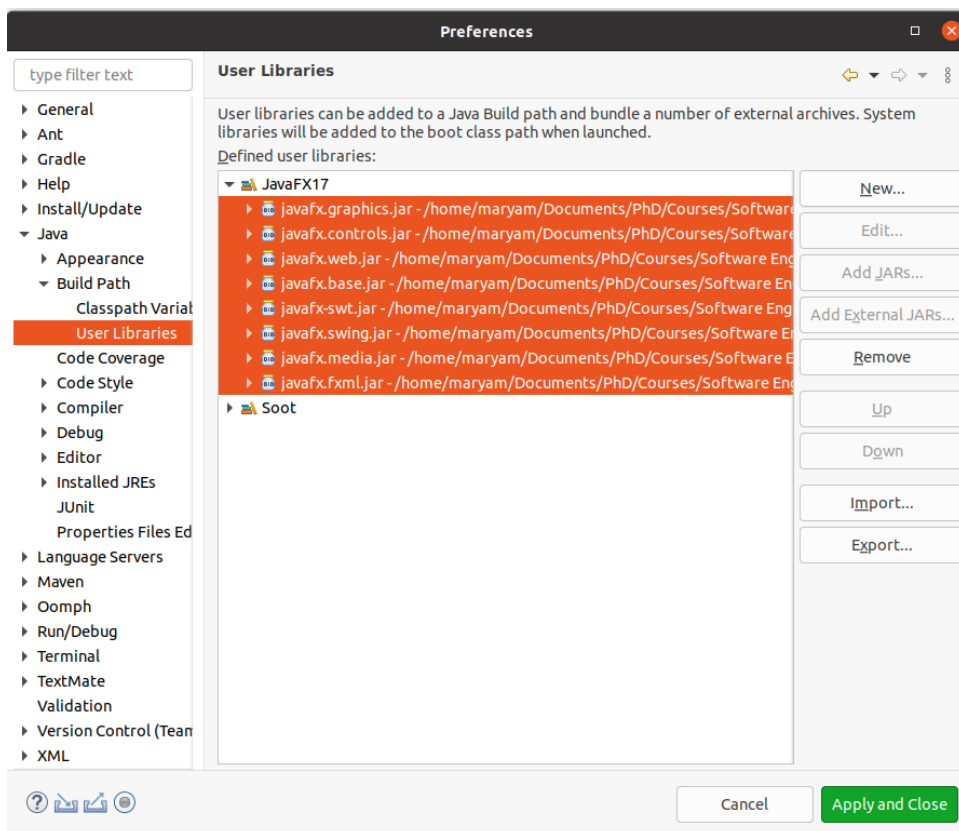
**Step 2.2:** Create a new User Library under  
Eclipse > Window > Preferences > Java > Build Path > User Libraries > New

---

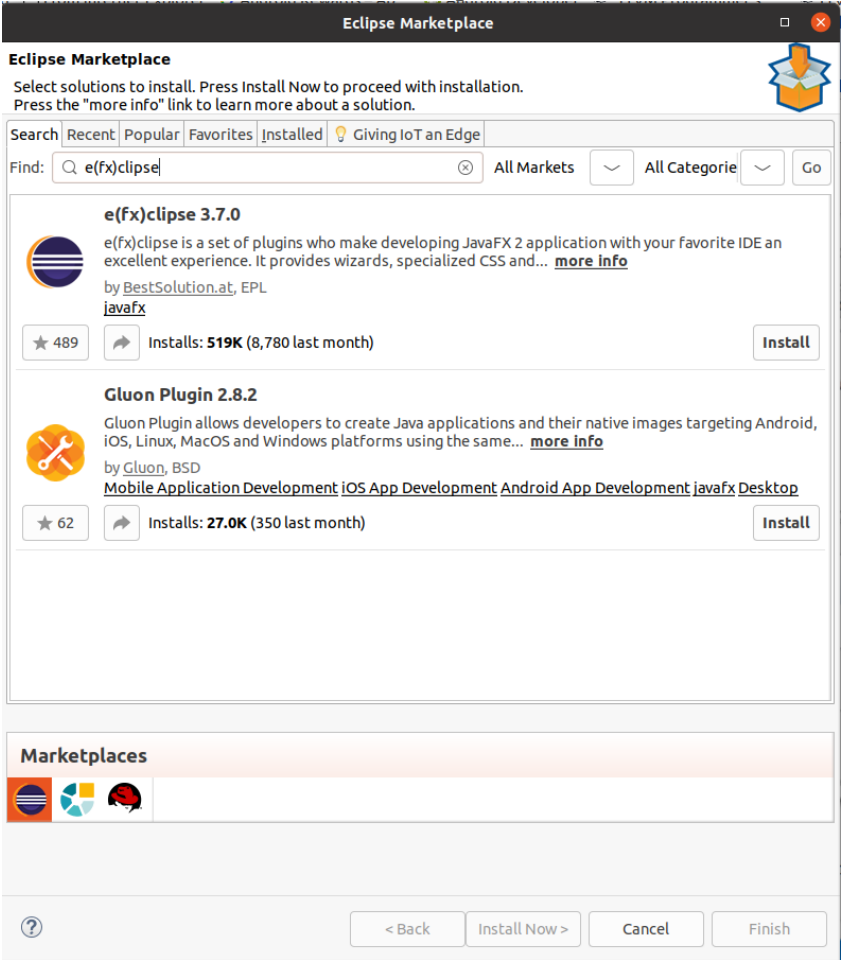
<sup>1</sup> I setup the eclipse in Linux. You can install it on every Operating System but keep in mind not to put javafx file in the Desktop.



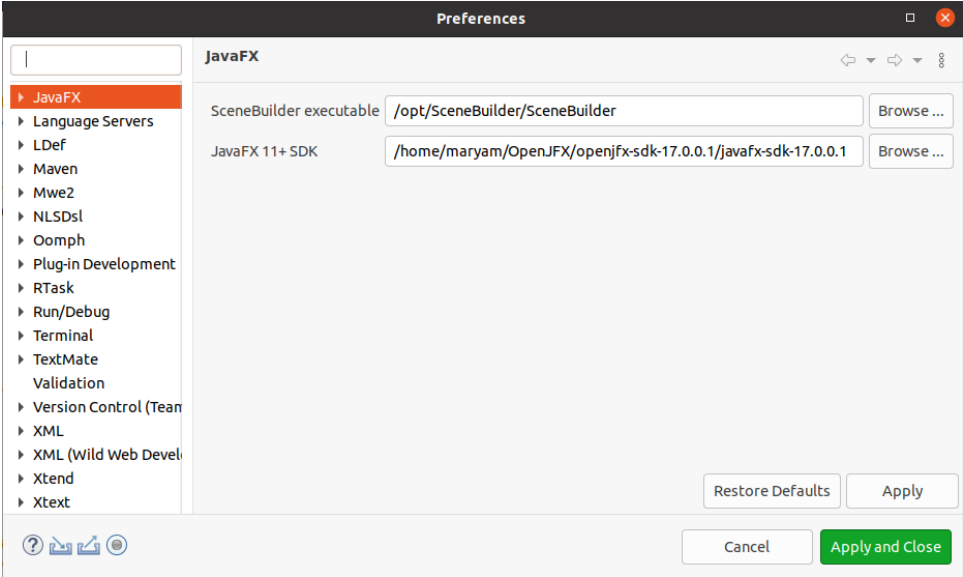
**Step 2.3:** Name it JavaFX17 and include all the jar files under the lib folder of JavaFX SDK by choosing” Add External JARs” button.



**Step 2.4:** Locate and install **e(fx)clipse** plugin from Eclipse > Help > Eclipse Marketplace. Wait until the installation ends and then restart Eclipse.



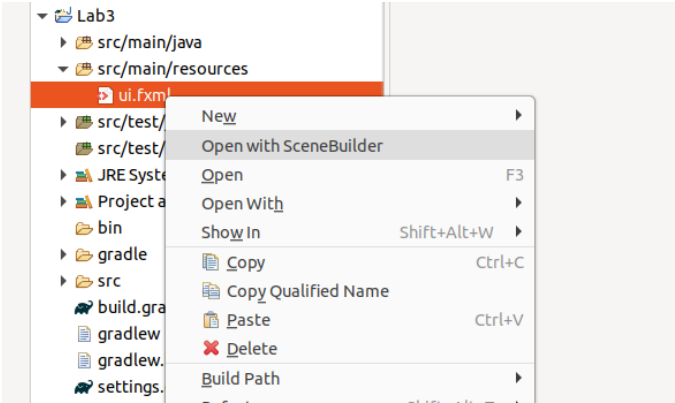
**Step 2.5:** Provide the information for “SceneBuilder executable” and “JavaFX 11 + SDK” under Window > Preferences > JavaFX



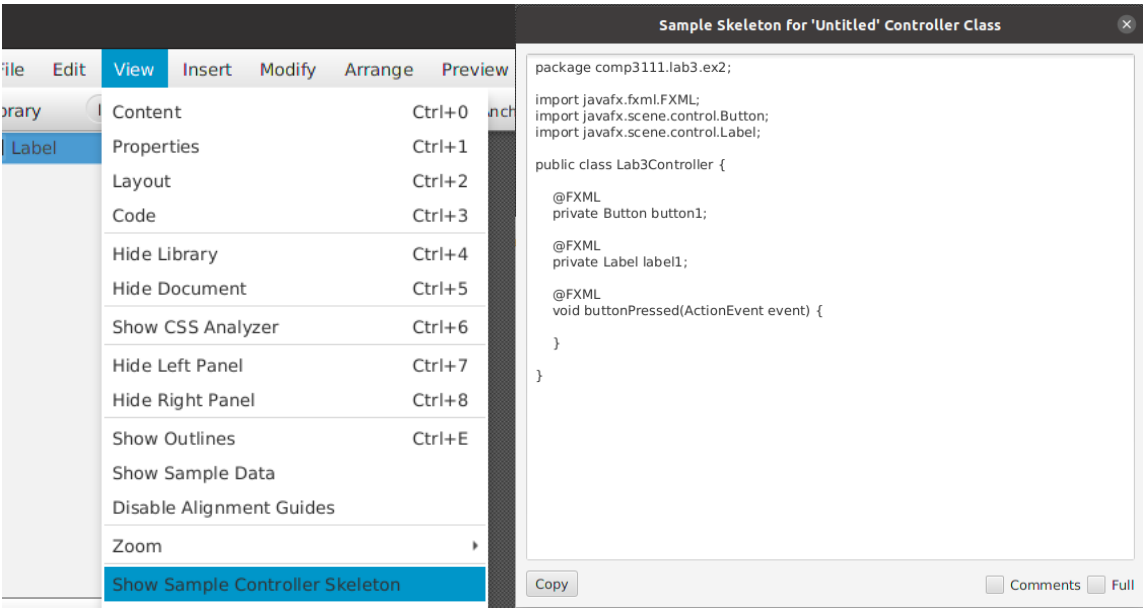
**Step 2.6:** Refer to the worksheet of Lab1 (Step 1.1 to Step 1.4) to create a new Gradle project and name it as Lab3 instead of Lab1

**Step 2.7:** Add the FXML document “ui.fxml”, which we have created in Exercise 1, to the project by drag-and-drop it onto the “src/main/resources” folder under Lab3

**Step 2.8:** Try right-click on “ui.fxml” and choose “Open the Scene Builder” to view/edit the FXML document

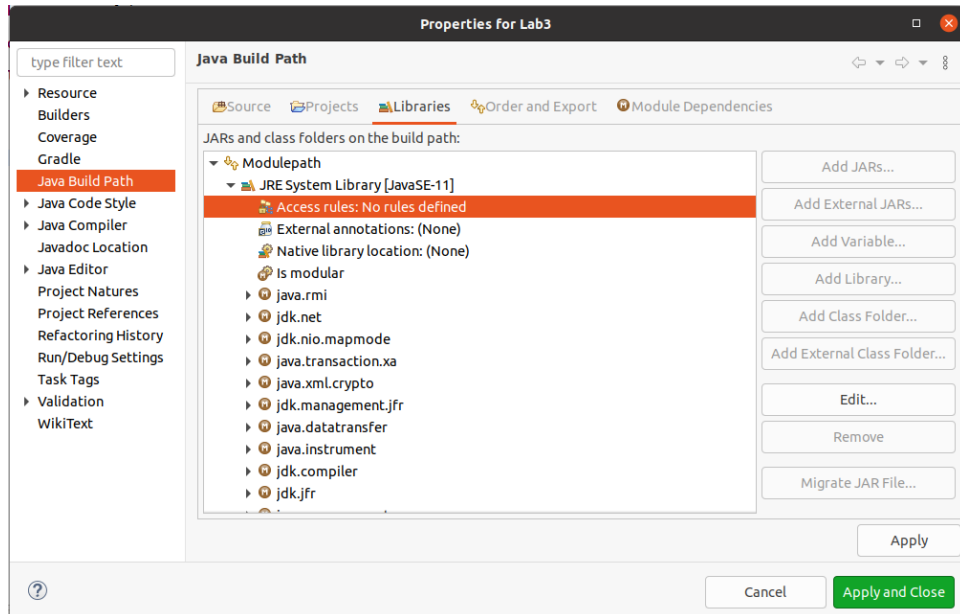


**Step 2.9:** From the menu bar at the top, click “View > Show Sample Controller Skeleton”. Copy the code and we will paste it in our eclipse project later.

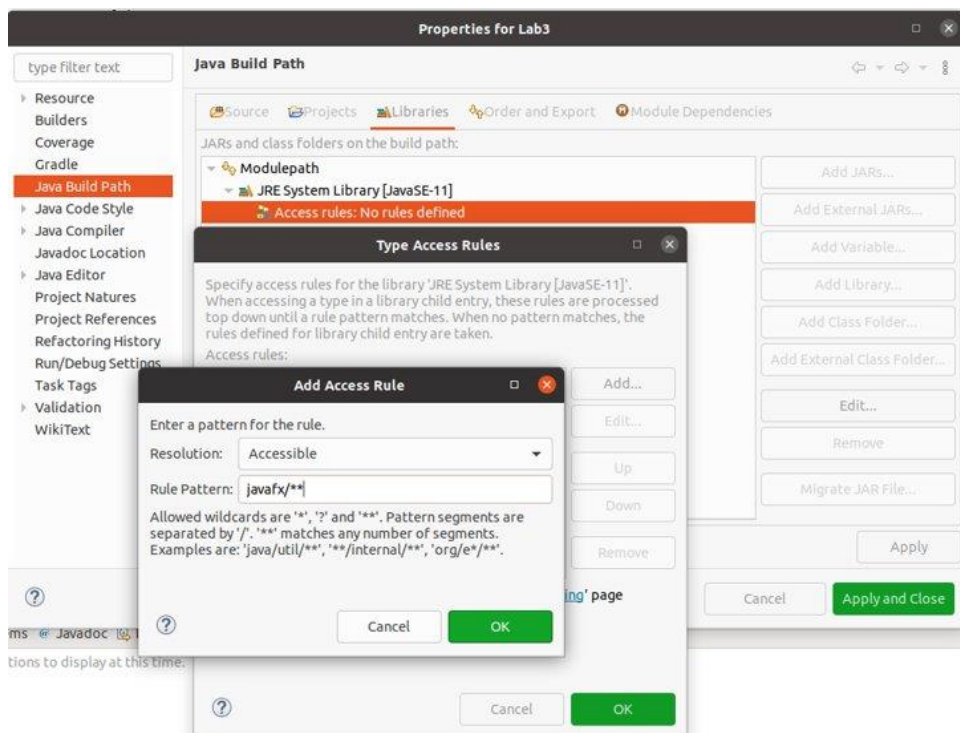


### Exercise 3: Create a simple JavaFX application.

**Step 3.1:** Right Click the project and select “Properties” > “Java Build Path” > “JRE System Libraries” > “Access Rule” and select “Edit”



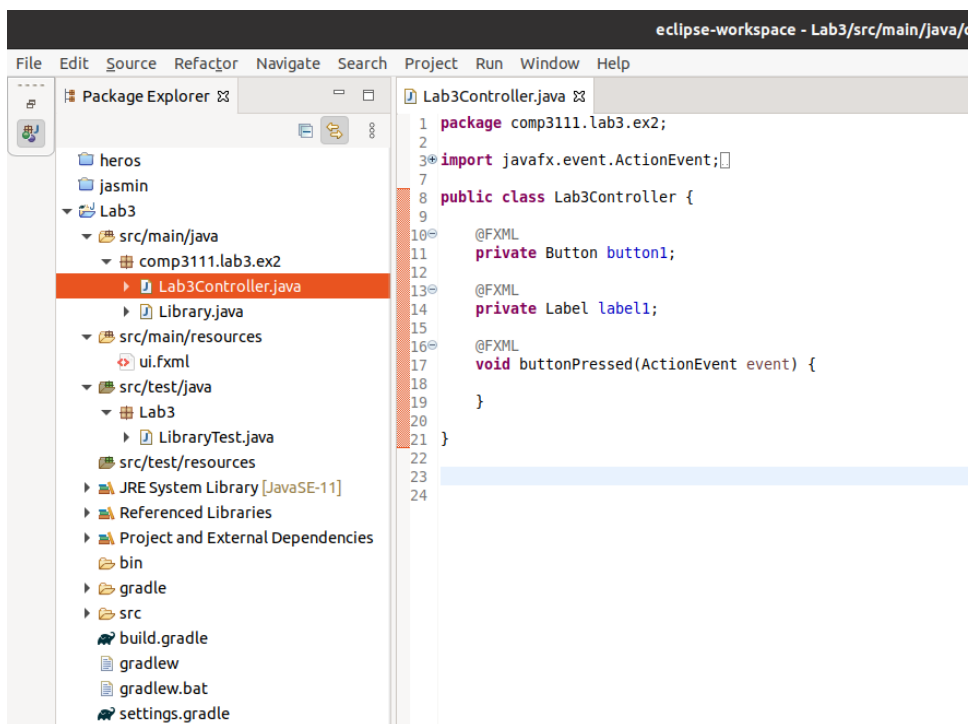
**Step 3.2:** Add a new rule with Pattern “javafx/\*\*” and make it accessible. This will suppress all warnings related to permission problem of JavaFX.



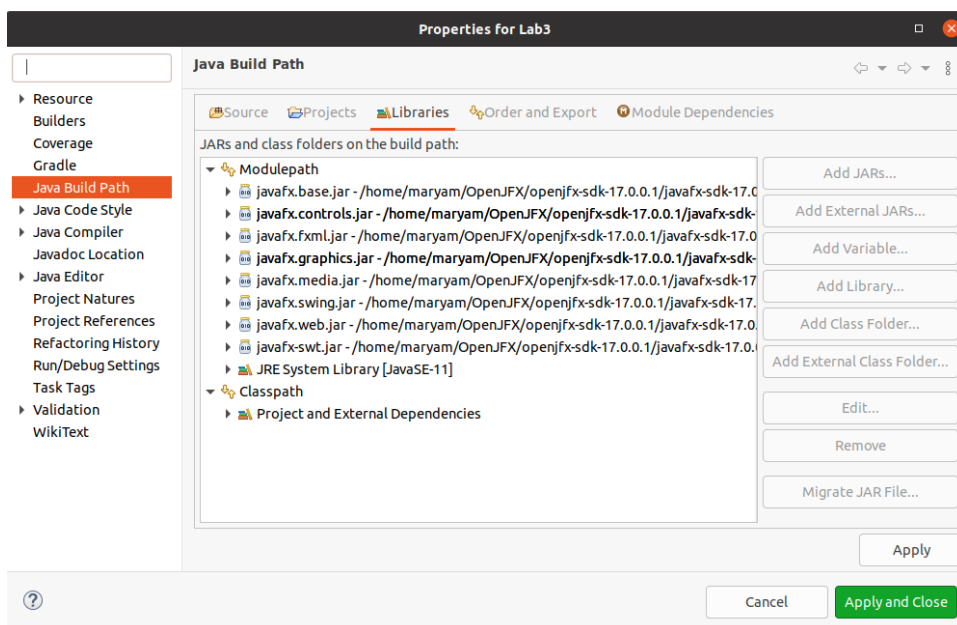
*Note: the program will still **compile** if you don't do this step. However your eclipse may prompt you a lot of errors.*

**Step 3.3:** Rename the package name to “comp3111.lab3.ex2”. Add a new class “Lab3Controller” under the package “comp3111.lab3.ex2”. Paste the content that you copy from the Scene Builder earlier. **Add** the line “import javafx.event.ActionEvent;” to the file.





Note that if you see lots of error due to not detecting javaFX, you can simply right click the project and select “Properties” > “Java Build Path” > “Module Path”. Then, choose “Add External JARs” and simply select all jar files under “javafx-sdk-17.0.0.1/lib” folder.



**Step 3.4:** Add a new class “UIApplication” under the package “comp3111.lab3.ex2”.

```

Lab3Controller.java  UIApplication.java  ✖
1  package comp3111.lab3.ex2;
2
3  import javafx.application.Application;
4  import javafx.fxml.FXMLLoader;
5  import javafx.scene.Scene;
6  import javafx.scene.layout.VBox;
7  import javafx.stage.Stage;
8
9  public class UIApplication extends Application {
10
11     @Override
12     public void start(Stage stage) throws Exception {
13         // TODO Auto-generated method stub
14
15         FXMLLoader loader = new FXMLLoader();
16         loader.setLocation(getClass().getResource("/ui.fxml"));
17         VBox root = (VBox) loader.load();
18         Scene scene = new Scene(root);
19         stage.setScene(scene);
20         stage.setTitle("Lab 3");
21         stage.show();
22     }
23
24     public static void run( String arg[]) {
25         Application.launch(arg);
26     }
27
28
29
30 }
31

```

**Step 3.5:** Change the driver Program Library.java to:

```

Lab3Controller.java  UIApplication.java  Library.java  ✖
2  * This Java source file was generated by the Gradle 'init' task.
4  package comp3111.lab3.ex2;
5
6  public class Library {
7
8     public static void main(String[] args) {
9         UIApplication.run(args);
10     }
11
12 }
13

```

**Step 3.6:** Configure the build.gradle as:

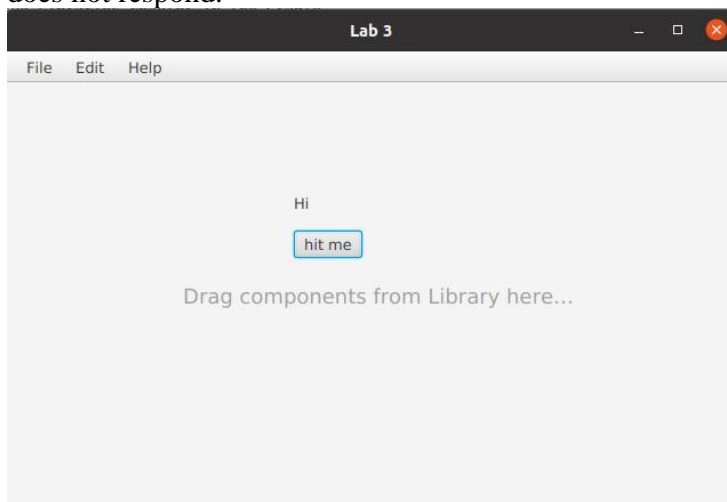
```

Lab3Controller.java  UIApplication.java  Library.java  *build.gradle
1/*
2 * This file was generated by the Gradle 'init' task.
3 *
4 * This generated file contains a sample Java Library project to get you started.
5 * For more details take a look at the Java Libraries chapter in the Gradle
6 * User Manual available at https://docs.gradle.org/6.6/userguide/java_library_plugin.html
7 */
8
9plugins {
10    // Apply the java-library plugin to add support for Java Library
11    id 'java'
12    id 'application'
13    id 'org.openjfx.javafxplugin' version '0.0.8'
14}
15
16repositories {
17    // Use jcenter for resolving dependencies.
18    // You can declare any Maven/Ivy/file repository here.
19    mavenCentral()
20}
21
22javafx {
23    version = "17.0.0.1"
24    modules = [ 'javafx.controls', 'javafx.fxml' ]
25}
26
27mainClassName = 'comp3111.lab3.ex2.Library'
28
29

```

Note : You can delete the folder “src/test/java” under Lab3 as we do not have any testing in this lab session.

**Step 3.7:** Try to execute the Gradle task “Run” and you should see a GUI while the button can be clicked but does not respond.

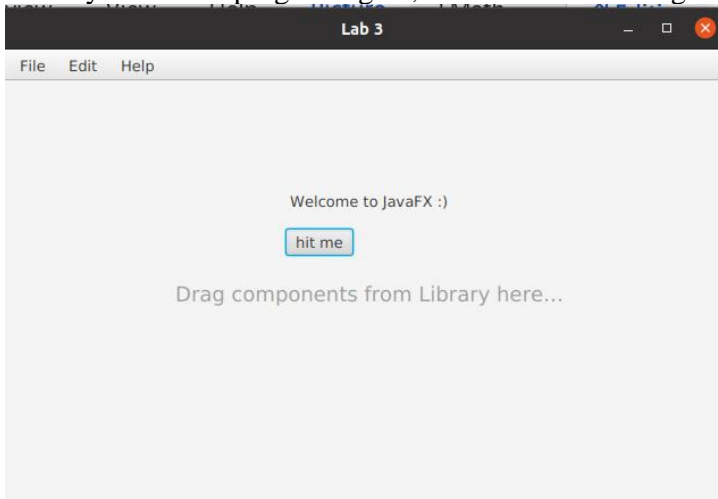


**Step 3.8:** Go to Lab3Controller.java and add “label1.setText(“Welcome to JavaFX”);” to the function “buttonPressed”

```
@FXML
void buttonPressed(ActionEvent event) {

    label1.setText(" Welcome to JavaFX :) ");
}
```

When you run the program again, the button will change the text of label1.



---

## Lab Activity and Assessment

### Lab Activity

- Complete Exercise 1, 2, and 3
- Modify the program in the following way: Add a TextField control to the GUI so that when the button1 is pressed, the label will display the text typed in the textfield.

### Assessment

There is no assessment task in this lab session.