Lab 15a – JavaFX Setup in Eclipse

Working in the CS Open Lab (2111 Nevins)

*Last updated: 2.27.2025*

* **This lab is a brief tutorial on how to use JavaFX in the CS Open Lab (2111 Nevins).**
* **If you are working on a personal device, see Lab 15b for additional steps necessary to setup the appropriate libraries.**
* **This Lab is not graded – there is nothing to turn in. You will do this lab in class and show me when you have completed it.**
* **There are two graded Labs that follow this: Lab 15 & Lab 16**

Contents

[1 Introduction 1](#_Toc191982803)

[2 Create a JavaFX Application 2](#_Toc191982804)

[3 Build User Library 3](#_Toc191982805)

[4 Attach JavaFX Libraries to Project 6](#_Toc191982806)

[5 Some Possible Errors & how to Fix them: 8](#_Toc191982807)

[6 Customize the GUI 9](#_Toc191982808)

[7 Attach JavaFX Libraries to Subsequent Projects 11](#_Toc191982809)

[8 Use Downloaded Packages with JavaFX Code 12](#_Toc191982810)

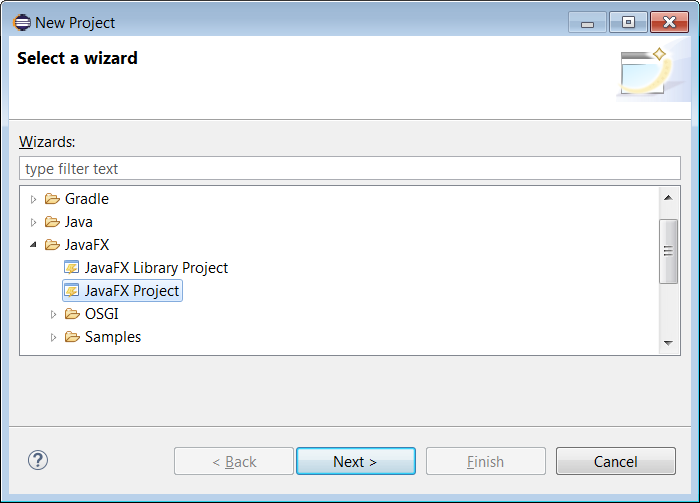
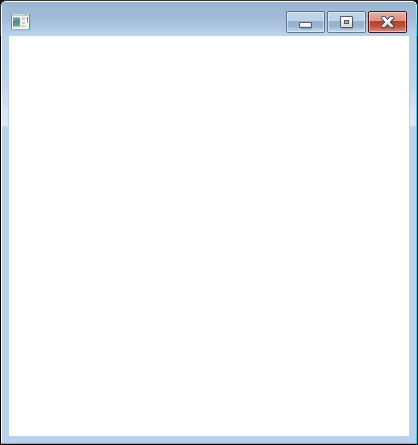
[9 Use Jar File with JavaFX Code 14](#_Toc191982811)

# Introduction

To create Graphical User Interfaces (GUI) in Java, we will use the JavaFX libraries that support this. The computers in the CS Open Lab (2111 Nevins) have the proper libraries installed and plugin configured. The objectives of this lab are:

1. Create a JavaFX project
2. Attach the JavaFX libraries to the project
3. Add preexisting code to a JavaFX project
4. Use a Jar file with a JavaFX project

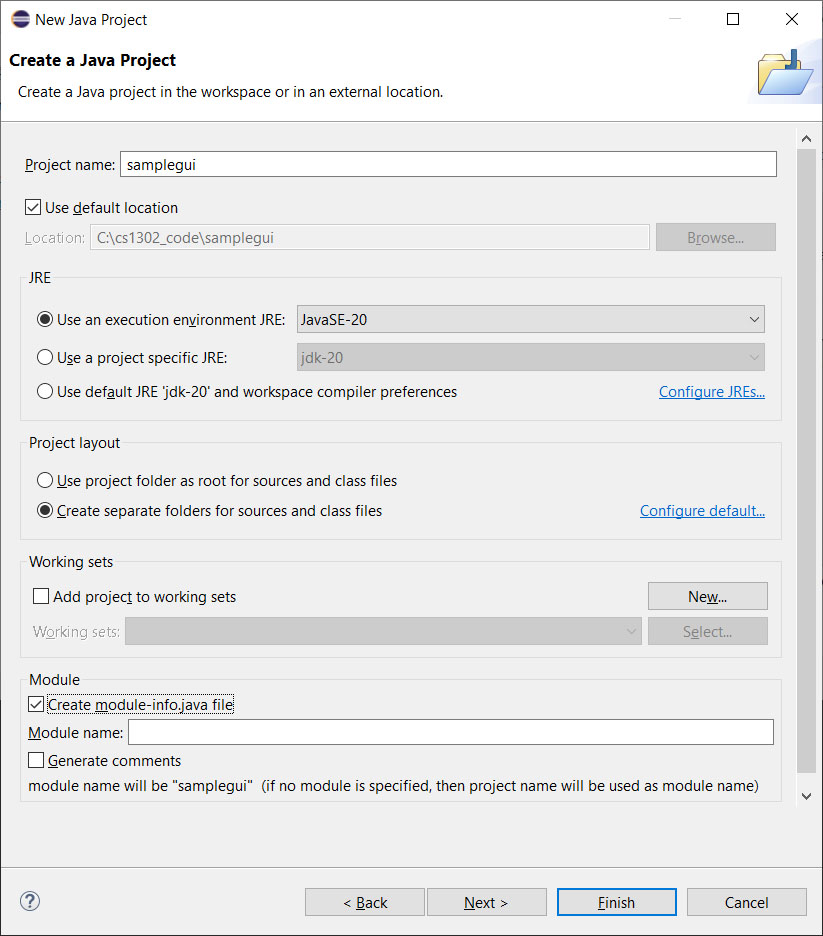
# Create a JavaFX Application

In this section, and continuing through Section 4, we create a JavaFX project which allows us to create a GUI (Graphical User Interface) application. When you create a JavaFX project, it automatically provides sample code that displays a GUI consisting only of an empty window as shown on the right.

1. Do the following: in Eclipse, as shown on the right, choose: File, New, Other (or Project), JavaFX, JavaFX Project, Next
2. Do the following:
3. Provide a *Project name*: *samplegui*

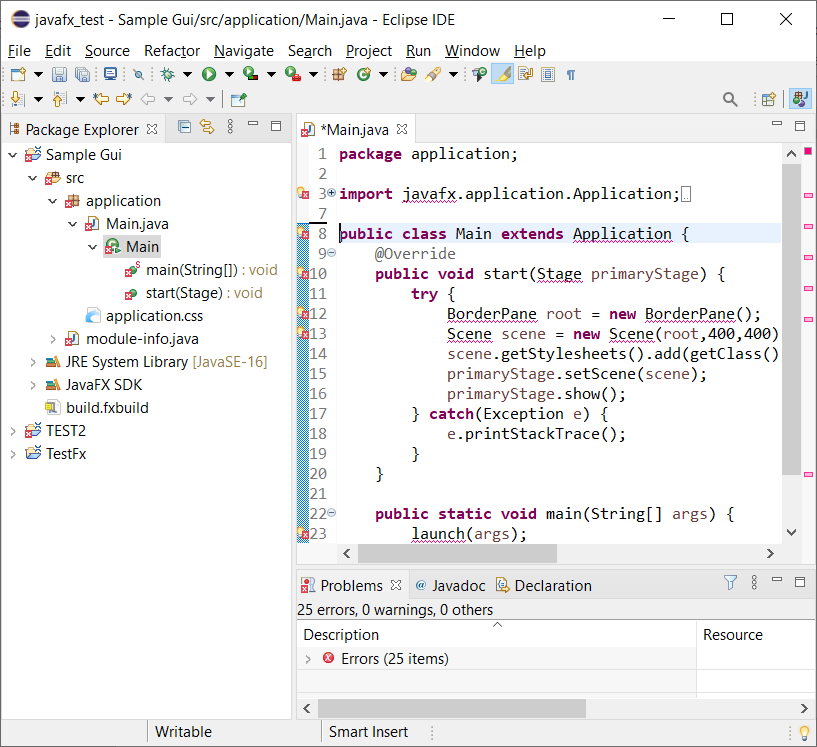
**Warning: The name cannot have any spaces or special characters and should be lower-case**

1. CHECK: *Create module-info.java* at the bottom
2. Choose: Finish



1. Reference the figure below.

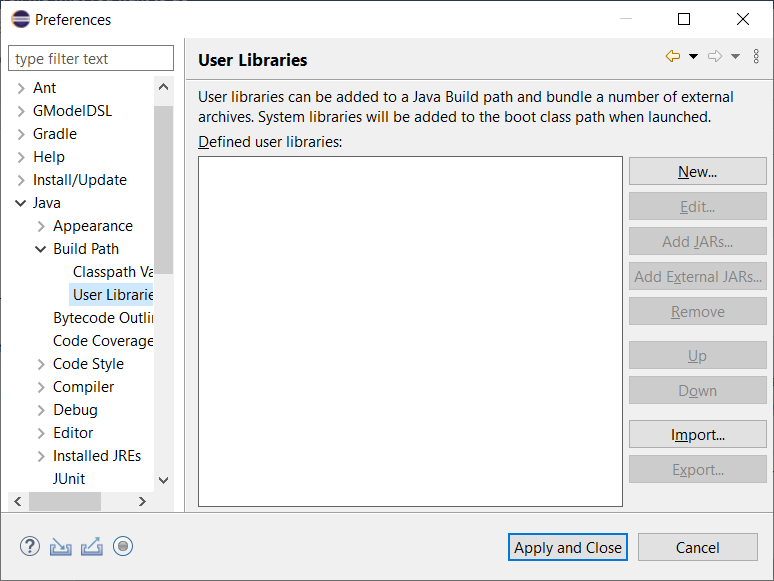
* An *application* package was created. Inside this package is a class, *Main.*java that contains a very simple Gui.
* Double-click *Main.java* to open it as shown below on the right.
* Notice the red “x”’s – that means there are compilation errors. We’ll resolve those next.



# Build User Library

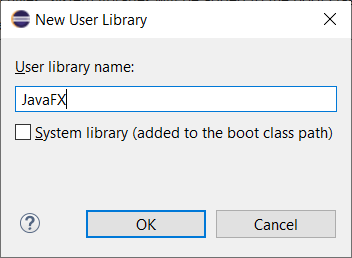
In this section you will build a *User Library* that contains the JavaFX libraries. In the [next section](#_Attach_JavaFX_Libraries), you will attach this User Library to your JavaFX project. You’ll need to create this library each time you use a new workspace. Probably, you’ll just have a single workspace for working on JavaFX projects and they can all use this User Library.

1. In Eclipse, choose: Window, Preferences. Then, from the tree on the left, choose: Java, Build Path, User Libraries, and then New (on the right)

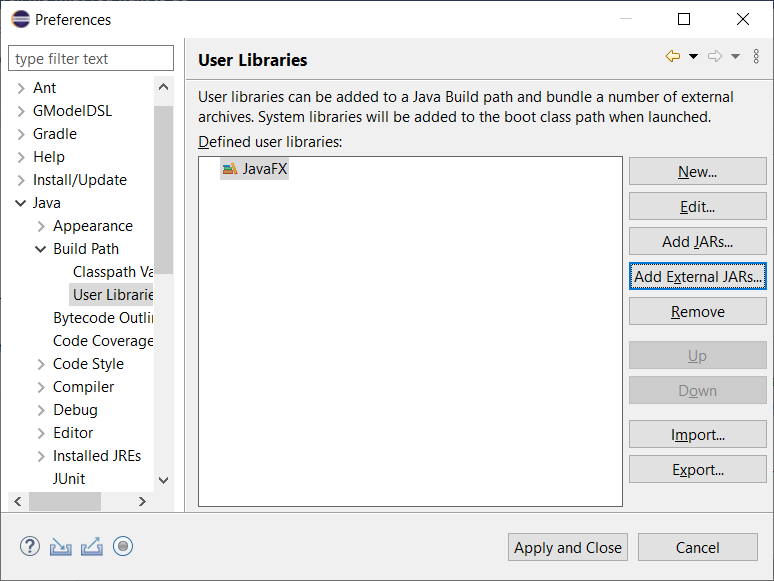


1. Choose: New, and then provide the name: JavaFX2 (name doesn’t matter) and then OK.

***The figure below shows the name as JavaFX – don’t use that, put a 2 or something on the end. There already exists one with the name, JavaFX, but it doesn’t work.***

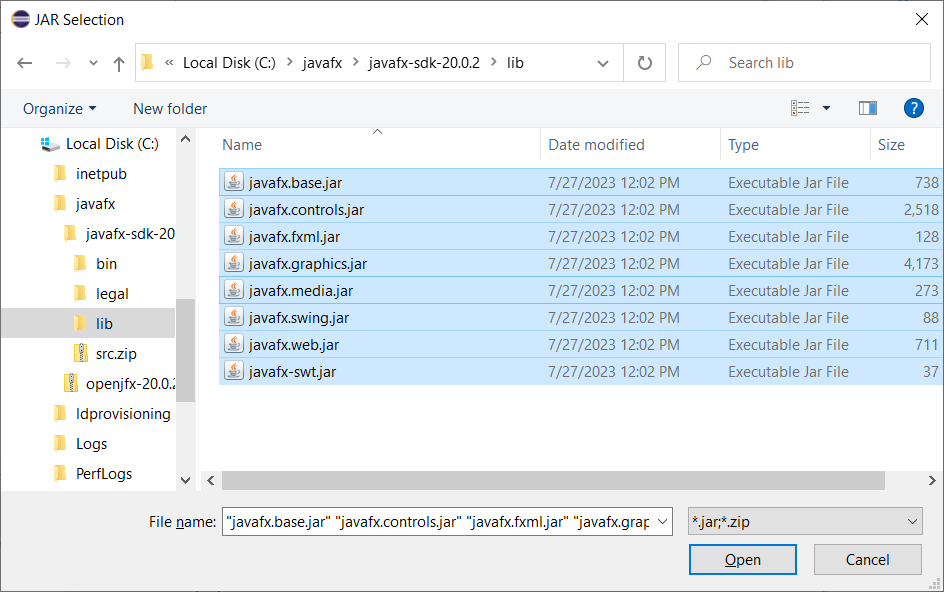


1. Choose: Add External Jars (on the right)

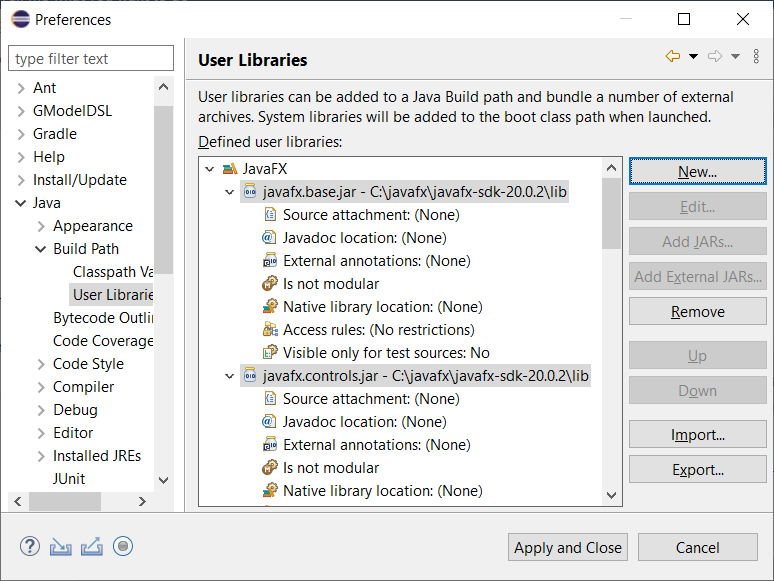


1. Navigate to the JavaFX jar files, select them, and choose: Open

**Note: if working in the CS Open Lab (2111 Nevins), the location of the JAR files is: C:\Eclipse\JavaFX\lib**



1. Choose: Apply and Close

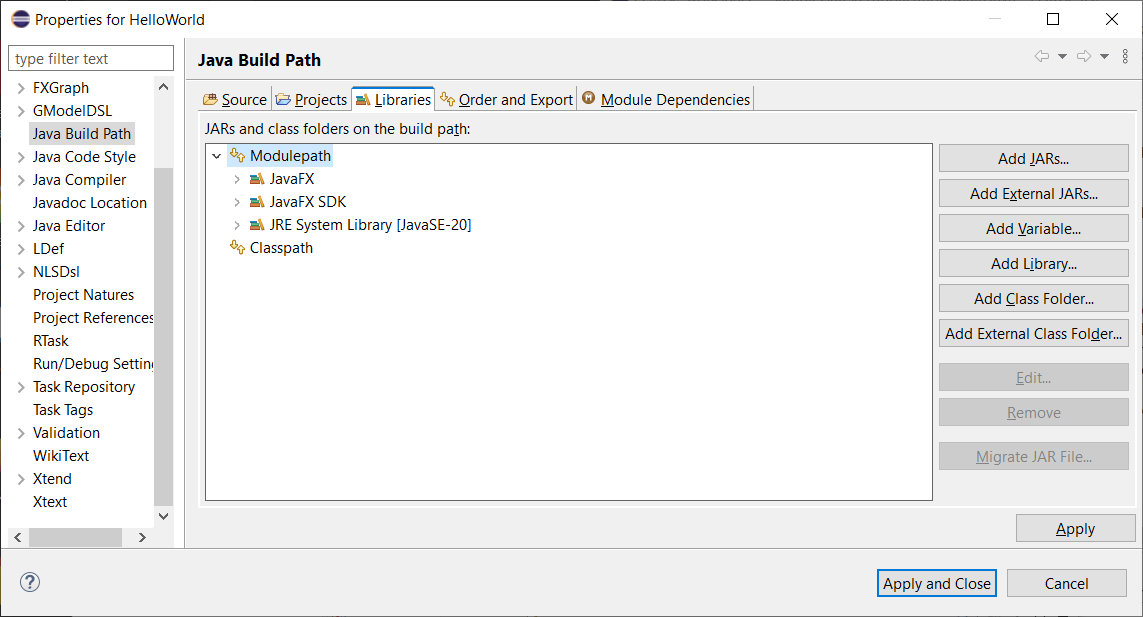


1. The compilation errors are still there. We’ll fix that in the next section by attaching the User Library we just built to the project.

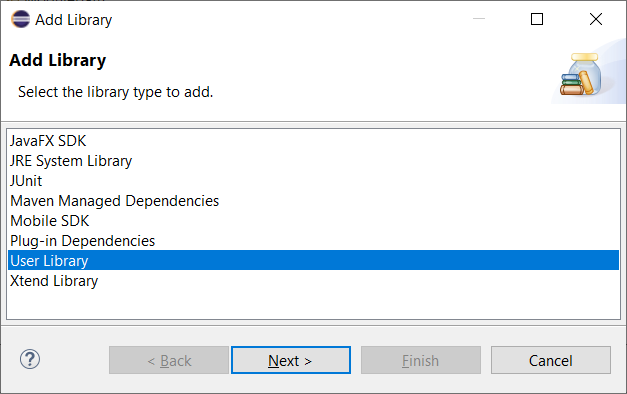
# Attach JavaFX Libraries to Project

In this section you will attach the JavaFX User Library from the previous section to the JavaFX Project you created earlier. At the conclusion, the sample GUI will compile, run, and display (hopefully!).

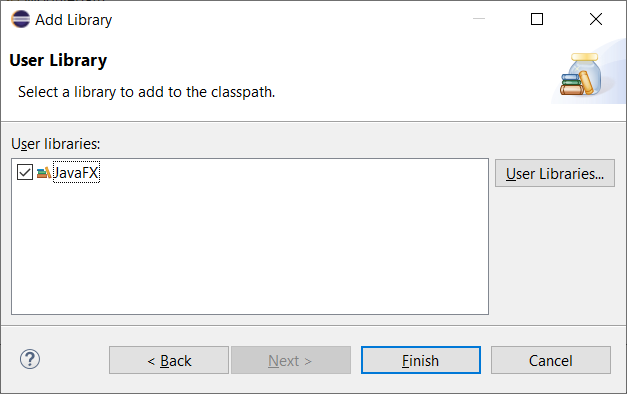
1. Next, we need to attach the JavaFX libraries to our project.
2. Select the *SampleGui* project node, right-click, and choose: Build Path, Configure Build Path
3. Choose the *Libraries* tab and the *Modulepath* node. Then choose: Add Library



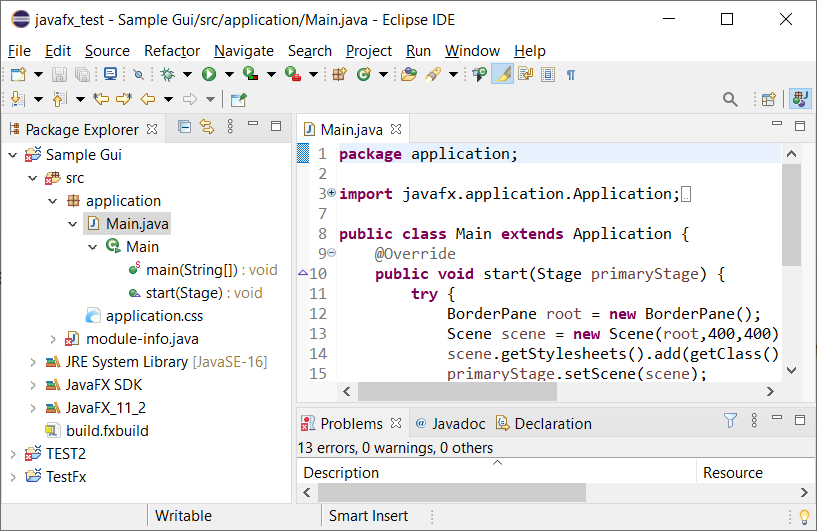
1. Choose: User Library, and then Next

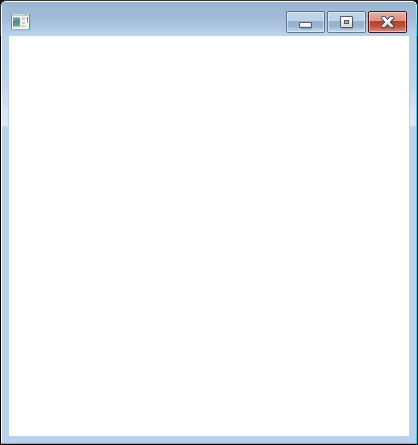


1. Select: JavaFX2 and Finish (the figure shows JavaFX, on your screen, you’ll see: JavaFX (possibly) and JavaFX2 (the one you created)



1. Choose: Apply and Close.
2. Open *Main.java* and the compile errors should be gone now:

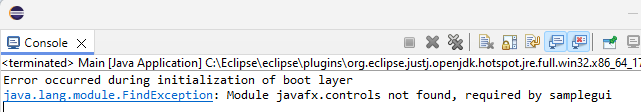


1. In theory, this should run properly now. However, a strange error was occurring while I was testing (2.26.2025). So, run *Main* by choosing: Run, Run (or the Green arrow icon). If things are successful, you’ll see an empty Gui as shown on the right. If you are successful, move on to [section 6](#_Customize_the_GUI).

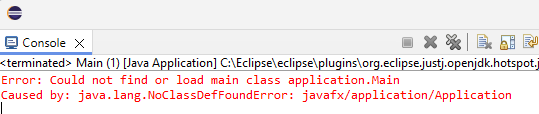
# Some Possible Errors & how to Fix them:

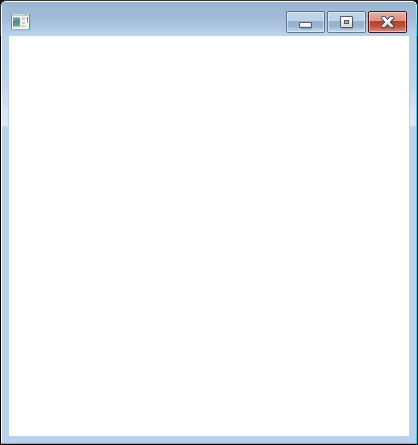
These are some possible errors you got from the previous step and their solution.

1. **Error 1**

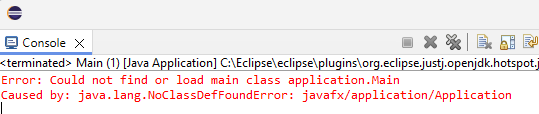


1. DELETE *module-info.java* from the Package Explorer.
2. Note: you are going to “undo” this delete a future step. On a Windows machine, I press: “Del” and it asks to confirm the delete. Then, in the future step, I use: Ctrl+z to undo the delete.
3. Rerun the program and you will probably get this error:



* 1. Press: Ctrl+z (undo) and *module-info.java* should appear back in the Package Explorer.
  2. Rerun *Main* by choosing: Run, Run (or the Green arrow icon) and it will display an empty Gui as shown on the right.

1. **Error 2 –** This is probably related to the above error; however, I’ve seen this one without the earlier error above. In this case, I simply created a new JavaFX project (in the same workspace, perhaps named: samplegui2, built the user library, attached it, and things worked.



1. **Error 3 –** The error message in the console refers to “Css2Bin”. It will look similar to:

Exception in thread "main" java.lang.IllegalArgumentException: expected file name as argument

at com.sun.javafx.css.parser.Css2Bin.main(Css2Bin.java:44)

The solution is to: right-click the project in the Package Explorer and choose: Run/Debug Settings. Then, select “Css2Bin” and choose: Delete (on the right).

**Remember to try creating a new JavaFX project if things go wrong.**

# Customize the GUI

1. Next, we add some controls to the GUI as a very brief example of how to build a GUI. Do the following
2. Close the empty GUI window that you launched above.
3. Identify this line in the code window:

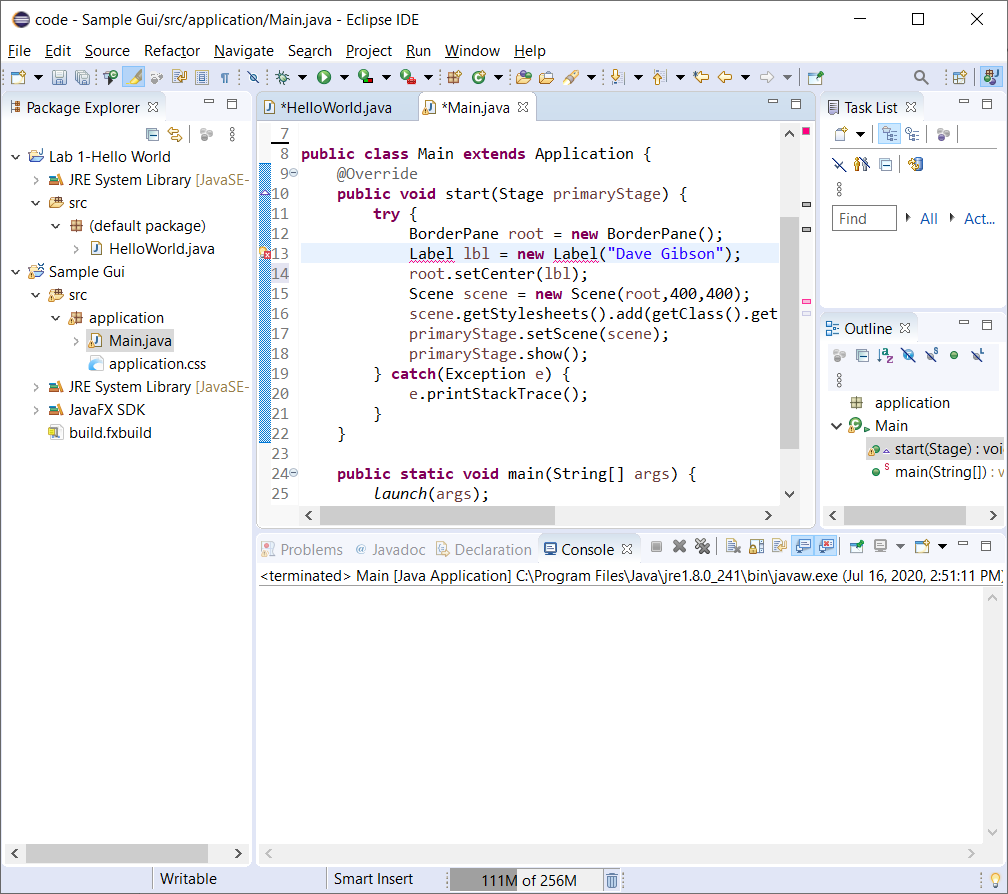
BorderPane root = **new** BorderPane();

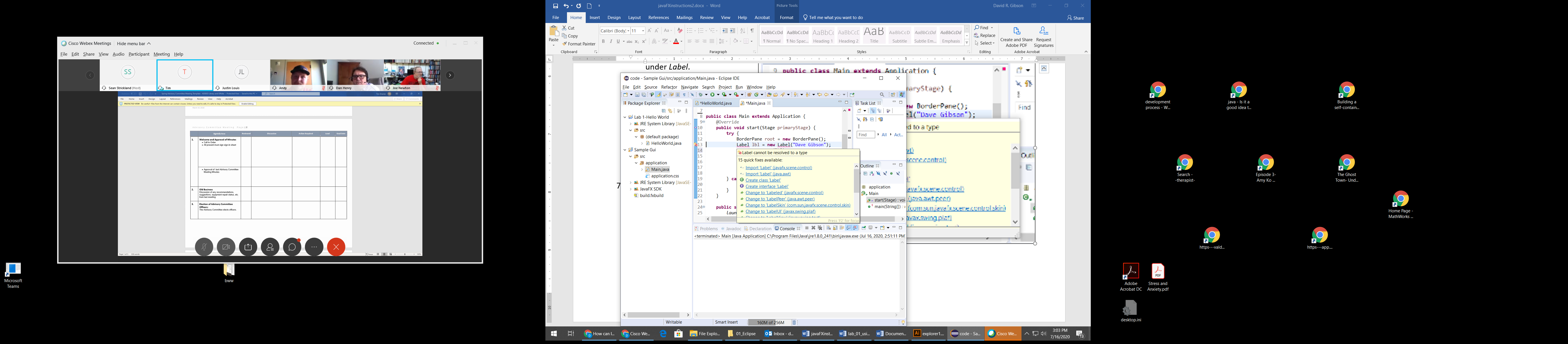
1. Immediately after the line above, add these two lines (**use your name**):

Label lbl = **new** Label("Dave Gibson");

root.setCenter(lbl);

The resulting code will appear as shown below:



1. You’ll notice a red X in the left margin that indicates a compile error. You also see the red squiggly under the *Label*. This indicates where the compile error is. To resolve this compile error, we need to add in *import* statement. You can do this in one of three ways. Do one of the following:
2. Hover your mouse under the Label and a context menu will appear as shown below (if necessary, click away from that line, and then move your mouse to hover over the Label). Then, choose:

“Import ‘Label’ (javafx.scene.control)”

Note: In the figure on the right, it is the first item in the list, **but might not be on your computer. Make sure you pick the correct one.**

1. Or, similar to *ii* above, click the red X in the margin and then, choose:

“Import ‘Label’ (javafx.scene.control)”

1. Add this in the import section at the top of the file:

**import** javafx.scene.control.Label;

1. Run *Main* by choosing: Run, Run (or the Green arrow icon) and it will display the Gui as shown below.

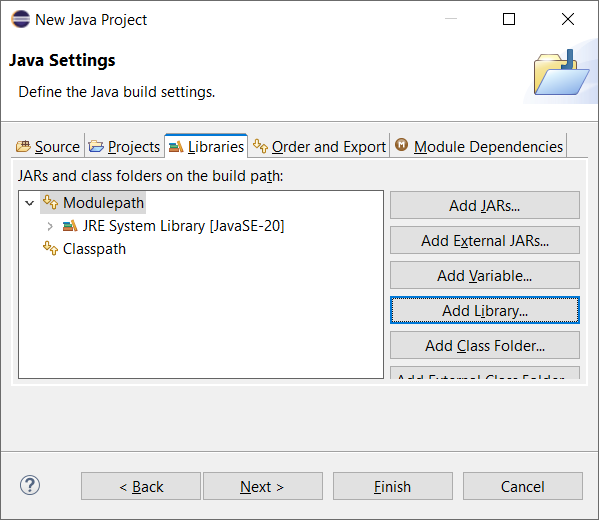


# Attach JavaFX Libraries to Subsequent Projects

**(Read, no action required)**

The User Library lives in the workspace. In other words, if you create another JavaFX project **in this workspace**, all you need to do is attach the User Library as described in [Section 4](#_Attach_JavaFX_Libraries) above. However, if you create a new workspace, or use one without the JavaFX User Library, then you’ll need to first create the library as described in a [Section 3](#_Build_User_Library) above, and then attach it to the project.

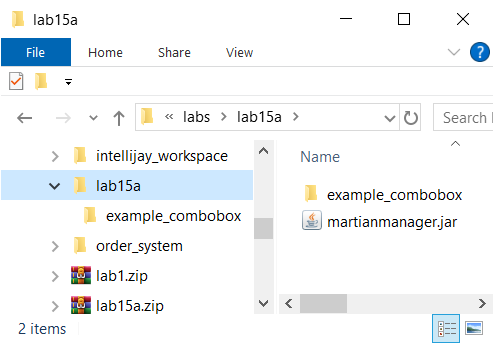
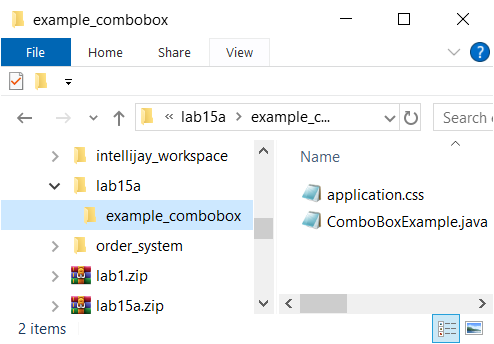
Note that when you first create a new JavaFX Project, you can attach the User Library immediately, instead of adding it after the project is created. On the “New JavaFX Project” dialog, after you supply the project name, instead of choosing: Finish, choose: Next. Then follow the steps (see image below) to attach as described in a [Section 4](#_Attach_JavaFX_Libraries) above, starting at step 1b.



# Use Downloaded Packages with JavaFX Code

This section shows how to setup and run a JavaFX Project that has been provided to you in a zip file. The process is similar to a [Section](#_Attach_JavaFX_Libraries) above, except that there is one additional step. This technique is important so that you can run sample code I provide to you for examples, for a homework, or a project.

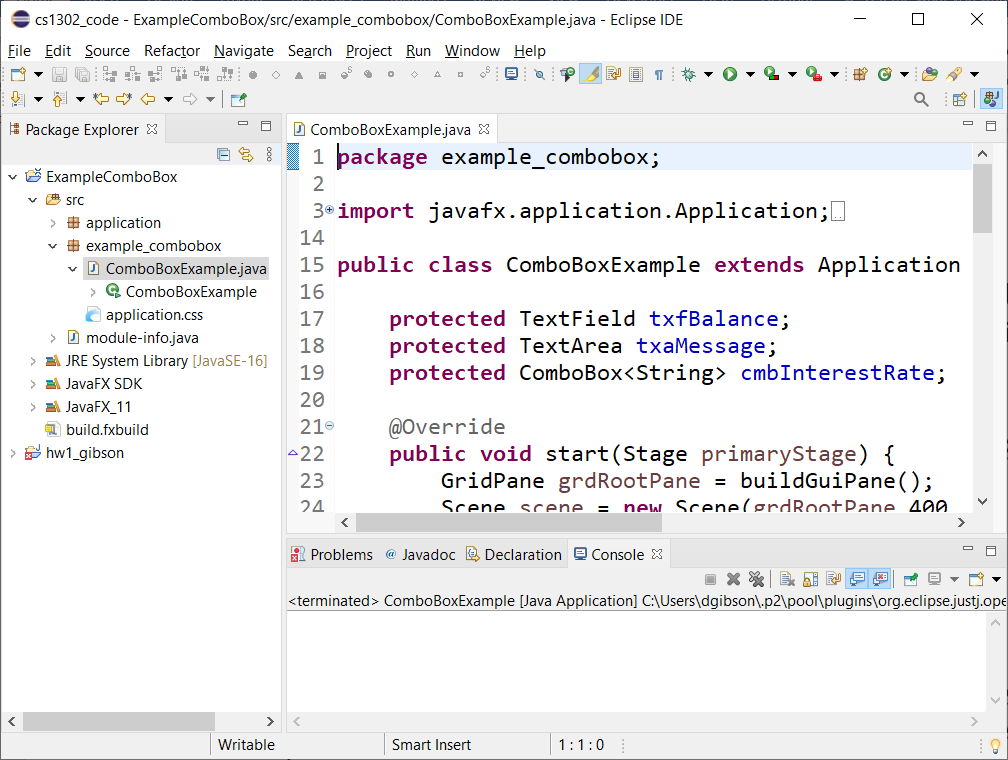
1. Download: *lab15a.zip* from the Lab 15a “code” link on the Labs page. Unzip somewhere (doesn’t matter where). Once unzipped, you will find: (a) *martianmanager.jar* which we will use in Section 8 below (figure on the left, below), (b) an *example\_combobox* folder with 2 files inside (figure on the right, below)

1. In the existing workspace, create a new JavaFX project by doing the following:
2. Provide a *Project name*: *examplecombobox*

**Warning: The name cannot have any spaces or special characters and should be lower-case**

1. CHECK: *Create module-info.java* at the bottom
2. Choose: Finish
3. Attach the JavaFX libraries following directions from this [Section 4](#_Attach_JavaFX_Libraries). Now, *Main.java* should run correctly. Verify this.
4. Copy (or drag) the *example\_combobox* folder into *src* node. When prompted, choose: *Copy files and folders*. Open *ComboBoxExample.java* as shown below:



1. *ComboBoxExample.java* compiles, but will not run. Try it and note the error:

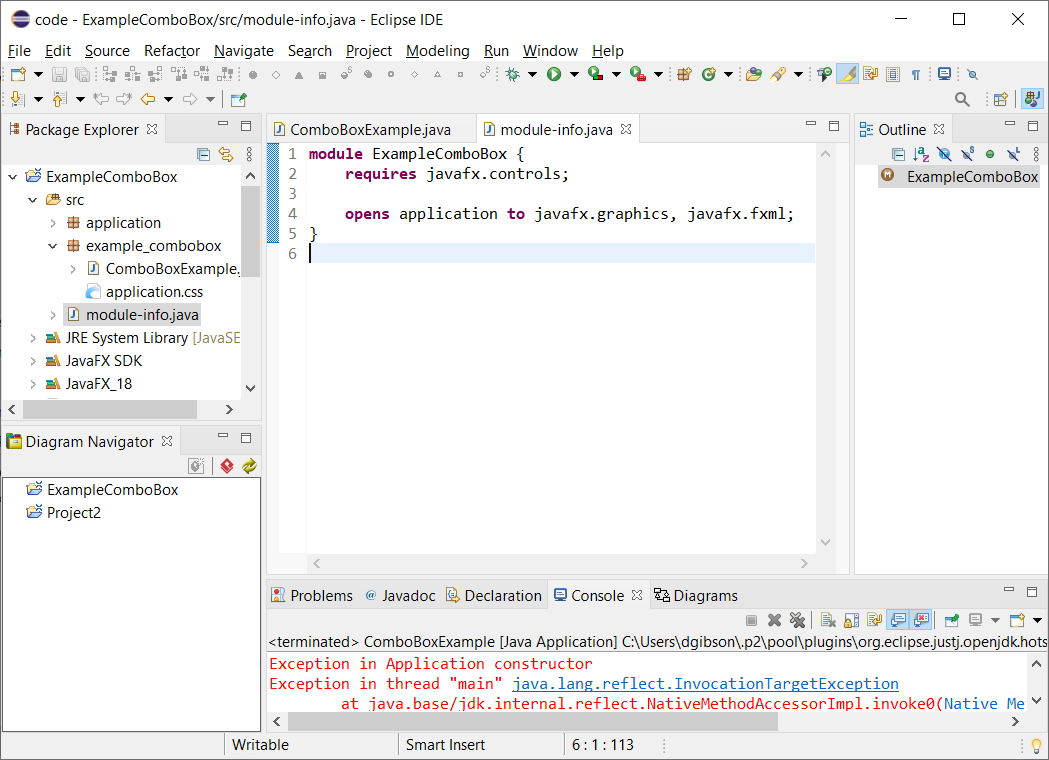
Exception in Application constructor

Exception in thread "main" java.lang.reflect.InvocationTargetException

at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

...

1. To make it run, we need to modify *module-info.java.* Open *module-info.java* (in the Package Explorer under the *src* node). The result is shown below:



1. Add this line to *module-info.java*:

**opens** example\_combobox **to** javafx.graphics, javafx.fxml;

So that the file now reads:

**module** examplecombobox {

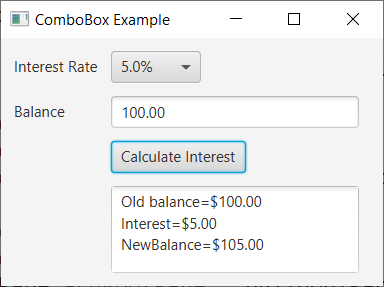
**requires** javafx.controls;

**opens** application **to** javafx.graphics, javafx.fxml;

**opens** example\_combobox **to** javafx.graphics, javafx.fxml;

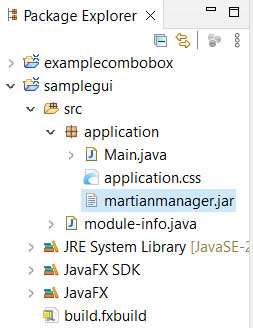
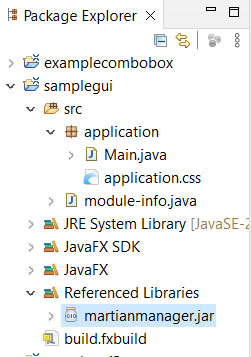
}

1. Save and run. Now, *ComboBoxExample.java* should run and function properly as shown below.



# Use Jar File with JavaFX Code

Jar files are like zip files and are used for distributing Java code. This section explains how to add a Jar file to a JavaFX project.

1. Find *martianmanager.jar* in the download from [Section 8](#_Use_Downloaded_Packages), step 1 above.
2. Return to your *samplegui* project in the Package Explorer. Drag the jar file into the *src* node into the *application* folder in the Package Explorer as shown on the right (choose: Copy files and OK on the dialog that appears).
3. Right-click the jar file in the Package Explorer and choose: Build Path, Add to build path. The file will “move” from the package folder to the Referenced Libraries folder as shown below.
4. Open *module-info.java* and add this line: “requires martianmanager;” Thus, the entire file might look like this:

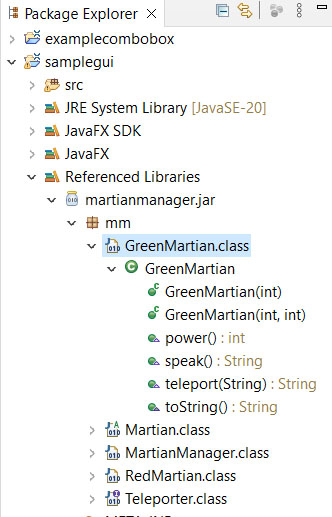
**module** samplegui {

**requires** javafx.controls;

**requires** martianmanager;

**opens** application **to** javafx.graphics, javafx.fxml;

}

1. Expand *martianmanager.jar* in the Referenced Libraries node. As shown on the right, for this example, the available classes are shown (*GreenMartian, Martian, etc.*), as well as the methods in each class and their signature (*e.g. GreenMartian* has a constructor that accepts an integer, and another that accepts 2 integers, *etc.*). In addition, it shows that these classes are in the *mm* package, this will be important next.
2. Open *Main.java* (the one in the *samplegui* project). Add the line below the import statements that are already there.

**import** mm.\*;

Note: to use any of the classes in the JAR file, you must first import them. The statement above imports everything (\*) in the JAR file. Note that you must include the package (*mm* for this example) in the path.

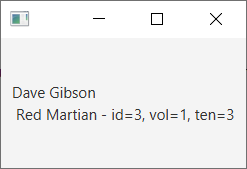
1. In the *start* method, replace this line:

Label lbl = **new** Label("Dave Gibson");

With these two lines (use your name):

RedMartian rm = **new** RedMartian(3,3);

Label lbl = **new** Label("Dave Gibson\n " + rm.toString());

1. Verify that *Main.java* runs correctly. The display will look similar to the figure on the right.

1. The lab is now complete. There is nothing to turn in. However, please report to me in class your success (or failure!).