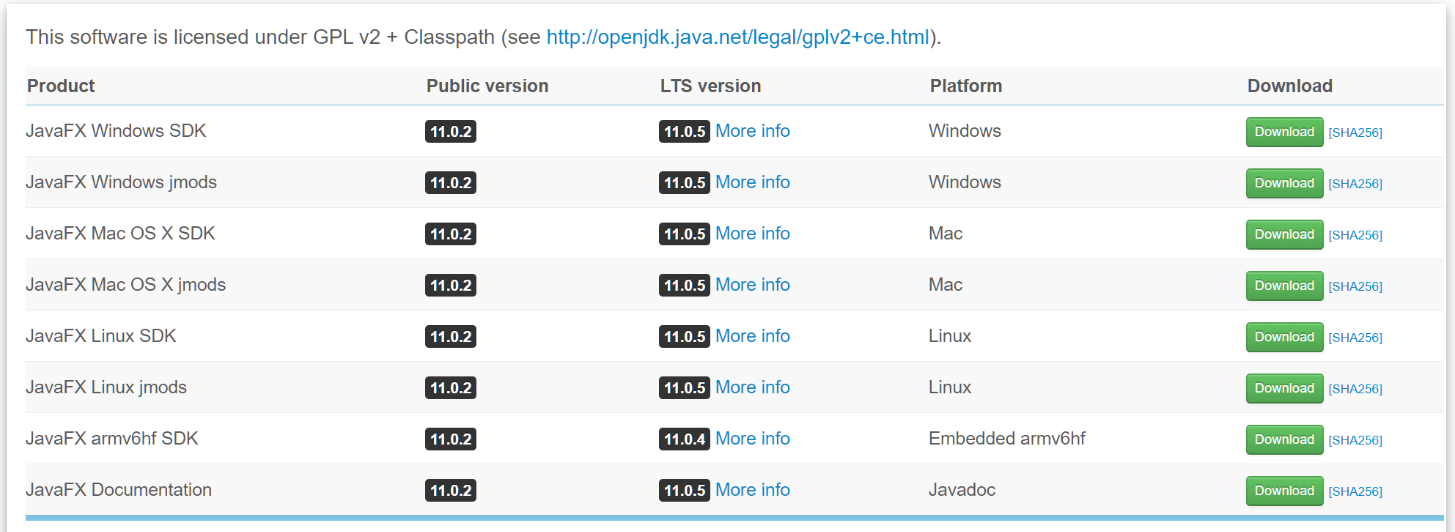
JavaFX does not come bundled with Java anymore - you must add the JavaFX library to each project separately. These instructions will take you through downloading the library and then show you how to configure each Java project to use the JavaFX library.

**Downloading and Installing the Library**

First, you need to obtain the JavaFX library. You can find it at [Gluon JavaFX](https://gluonhq.com/products/javafx/" \t "dl). Note that the version numbers you see in the images may not match the current version exactly, since the library may have been updated after these screen shots were taken.

We will be downloading the latest LTS version of the JavaFX library: Scroll down the page to the first table under the "Long Term Support" heading. Select the download button for the Java FX SDK for your system. Don't worry about the jmods download, you only need the SDK.



Save the file somewhere on your computer so you can find it later.

Earlier in the term you created a directory for this course. For example C:\Java2

In your course directory, add a sub-directory called "libraries" or "library" or "lib" - whatever you prefer. Do not put this inside your /projects directory that contains your NetBeans projects: /libraries and /projects should be siblings and have the same parent directory /prog24178 or /Java 2 or whatever. The /libraries is where you're going to put your JavaFX library, and where you'll add other libraries you'll be given in the course, as well as libraries that you create yourself in this course.

Download javaFX Windows SDK.

Once the download has finished, open the downloaded file in a zip utility of your choice (e.g. WinRAR, IZarc, 7-zip, etc).

Unzip the contents to a directory on your system in your prog24178/libraries (or Java 2/library or whatever you called things).

That's all you need to do: JavaFX is ready to use.

All you have to do is add it to any existing or new Java project in which you want to code with JavaFX. This can be a bit annoying (you do have to do these next steps for each project) but it's quite normal and you'll get used to it. You can keep configuration information in a sticky note or other file on your desktop for easy access.

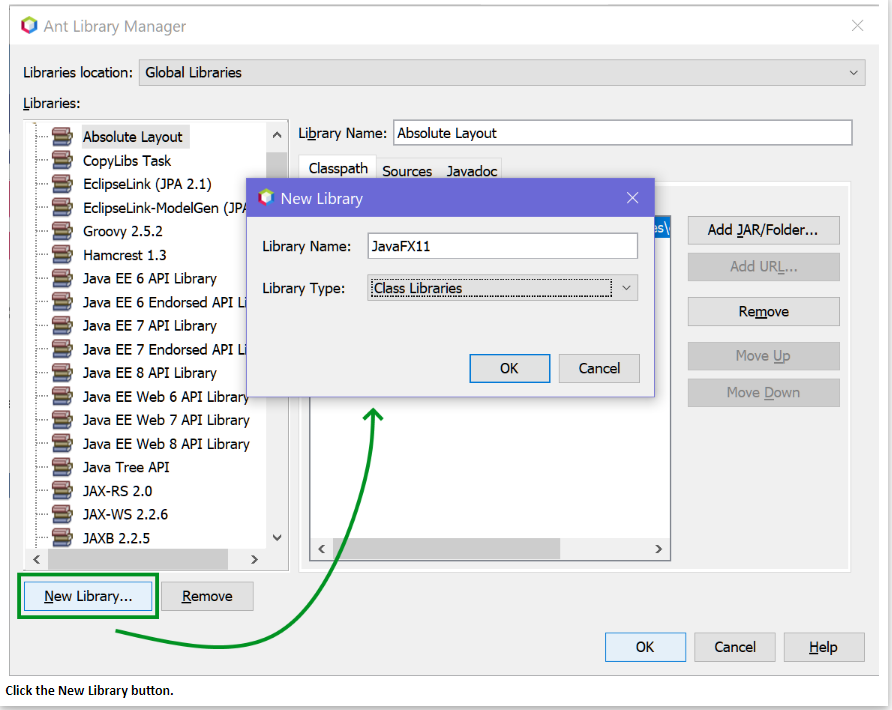
To code with JavaFX, you need to add the JavaFX library to each project you want to create. In order to do this, we need to set up the JavaFX Library as a **global library**, meaning that it's available to any project we create in NetBeans.

## Adding JavaFX Global Library

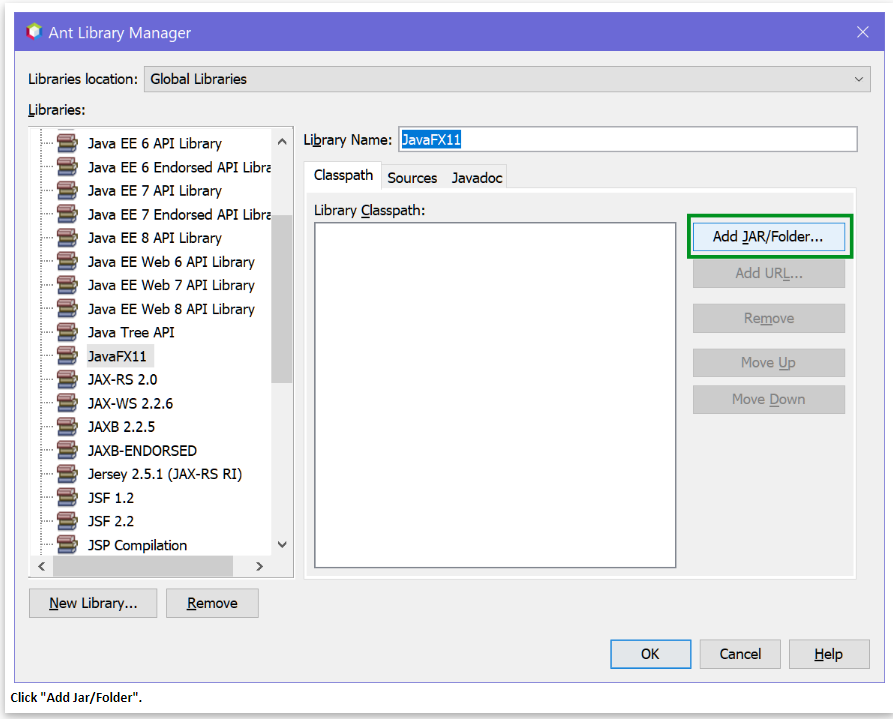
Run NetBeans, if it's not already running.

First, we need to create a new global library for the JavaFX library: This will add a library item for the JavaFX library that you can then add to each Java project in which you want to use JavaFX.

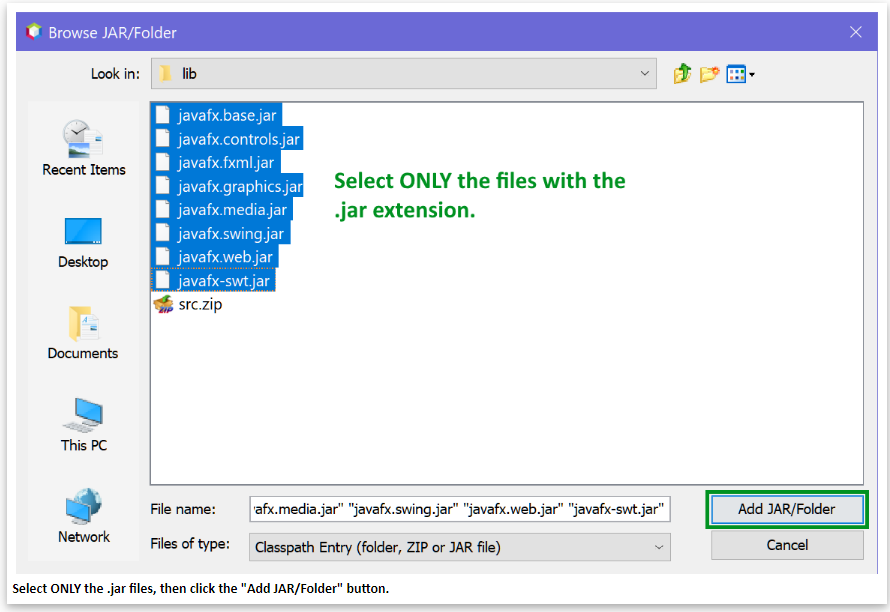
1. From the Tools menu, select Libraries. This shows you the Library Manager. You'll see a list of existing libraries on the left and an area in the middle where you can add other libraries.
2. Below the list of libraries on the left, click the New Library button. This will open a small, simple dialog where you can enter the information for the JavaFX library.



1. For the library name, enter "JavaFX11". You don't have to change the library type (it should already say "Class Libraries").
2. Click the OK button to go back to the Library Manager screen.
3. The JavaFX library should now appear in the Libraries list on the left and currently selected, so that its information appears in the main area of the screen.
4. On the right side of the screen, click the "Add Jar/Folder" button to add the files that make up the JavaFX library.



1. Browse to the location where you unzipped your JavaFX zip file and go into the /lib directory, for example Documents\prog24178\libraries\javafx-sdk-11.0.2\lib
2. Select all the files with the .jar extension. DO NOT select the src.zip or the properties file (if there is one), ONLY select the JAR files.



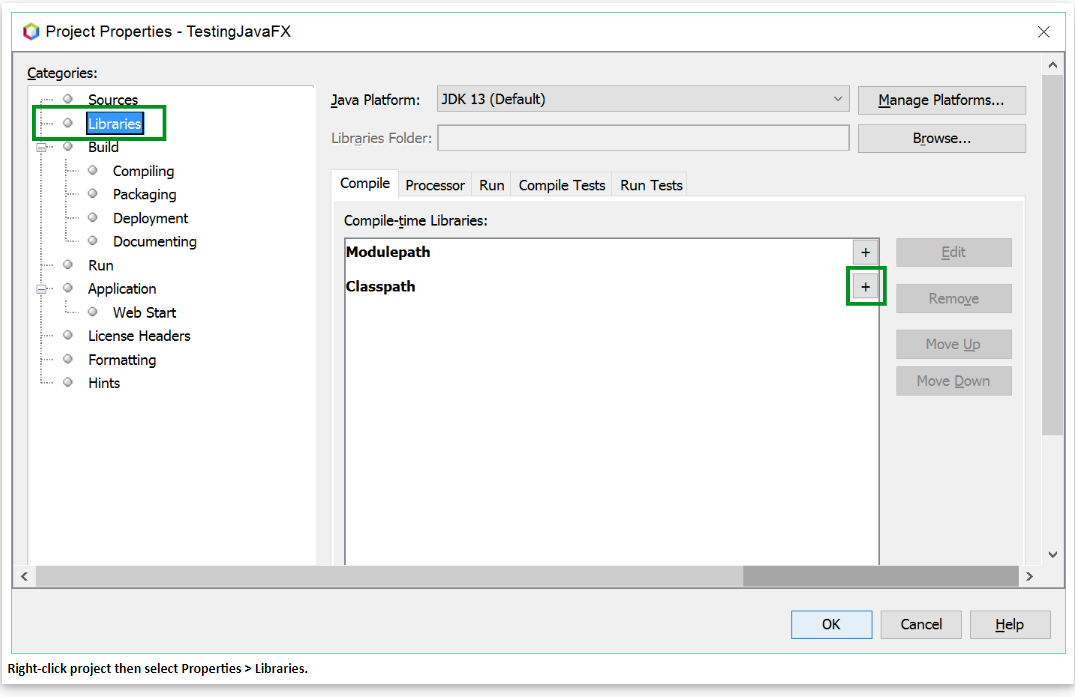
1. In the list in the classpath tab, you should now see absolute paths to each of the JAR files. Click the OK button at the bottom.

Restart NetBeans

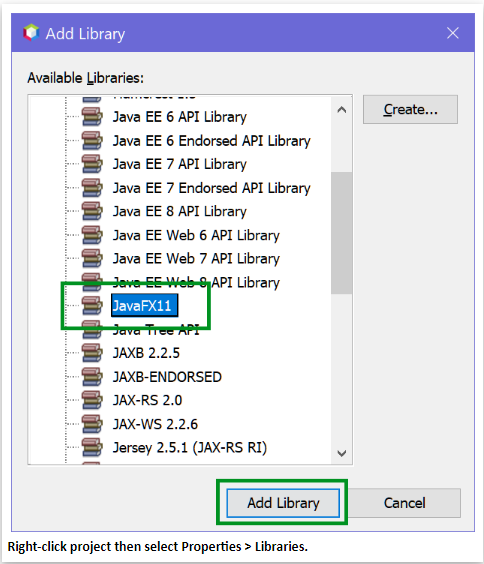
This adds the JavaFX library as a global library so that you can now add that library to any project in which you want to code JavaFX UIs. Let's try it out.

**Creating a JavaFX Project**

1. Create a new project: File > New Project
2. As usual, click Java With Ant > Java Application. DON'T create JavaFX project: this actually doesn't work at this time.
3. Set up your project like you normally would. I used the Project Name "TestingJavaFX" and stored it in the usual /projects directory. For the main class, I used prog24178.fx.TestingJavaFX (I didn't use Main this time, I usually don't for GUI applications, but you can use Main if you prefer).
4. Once your default project has loaded, right-click your TestingJavaFX node in the project explorer and select "Properties".
5. In the Properties dialog, click the Libraries node along the left side (it's the second item in the list).



1. You'll notice in the main part of the dialog that your Java platform is set to whatever you installed last, but if you have multiple Java versions, you can change this here for this particular project, if that ever becomes necessary.
2. Near the middle of the screen, the Compile tab is selected and shows an empty list of compile time libraries: there are no extra libraries added to this project, yet.
3. In the list of compile time libraries, you'll see two headings: Modulepath and Classpath. Beside each, to the right is a plus sign + button. Click the one beside Classpath and select "Add Library" from the drop-down list.
4. You'll now see a list of available libraries, including the JavaFX11 library you added earlier. Select it and then click the "Add Library" button.



1. Back at the Project Properties window, you should see that JavaFX11 has now been added to the Classpath for this project. Click the OK button.

You've added the JavaFX11 library to this current project. If you look in the project explorer at your project's libraries node, you'll see all the JavaFX 11 jar files in there.

There are two more project properties you'll need to change for JavaFX to work:

1. Turn off "Compile On Save" (CoS): Go to the project properties > Build > Compiling and uncheck "Compile on Save".
2. Add the module path for two of the jar files in the JavaFX Library: Go to the project properties > Run and in the VM Options field, copy and paste the following:

--module-path "\path\to\javafx\lib" --add-modules javafx.controls,javafx.fxml

Except, replace **\path\to\javafx\lib** with the absolute path to the \lib directory

of your JavaFX library. For example, if you put your library in prog24178\libraries, your command might look like this:

--module-path "C:\Users\parul\Documents\prog24178\libraries\javafx-sdk-11.0.2\lib" --add-modules javafx.controls,javafx.fxml

Make sure that is all enclosed inside the "" quotes or it won't work.

Note  
Modules are a new structure level in Java. A class contains fields and methods. A package contains classes. A module contains packages. Java has become too large for smaller devices, so modules allow a developer to create **dependencies** between classes without having to include a whole JAR file of stuff they don't need. Instead, they can create modules of packages and only include the modules they need, instead of the whole set of Java packages.

Unfortunately, you'll have to do those 3 steps for each JavaFX project you create (even if we could do it globally for all projects, it would be a waste to have those options set for projects that aren't JavaFX projects.

I recommend creating a sticky note on your desktop with the three steps and any values you need, so you don't forget. Mine looks like this:

JavaFX Project:  
1. properties > Libraries > add JavaFX11 library  
2. properties > Build > Compiling > uncheck Compile on Save  
3. properties > Run > VM Options paste this:  
--module-path "C:\Users\Parul\Documents\prog24178\libraries\javafx-sdk-11.0.2\lib" --add-modules javafx.controls,javafx.fxml

To test it everything, we need to add a couple more components to our project.

First, go to your Main class and delete everything under the package statement. Paste this under the package statement (if you changed the class name, you'll have to edit it).

package fxtest1;

import javafx.application.Application;

import javafx.fxml.FXMLLoader;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.stage.Stage;

/\*\*

\*

\* @author Rakesh Kantaria

\*/

public class Fxtest1 extends Application{

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

launch(args);

}

@Override

public void start(Stage stage) throws Exception {

Parent root = FXMLLoader.load(getClass().getResource("FXMLtest1.fxml"));

stage.setTitle("Testing");

stage.setScene(new Scene(root));

stage.show();

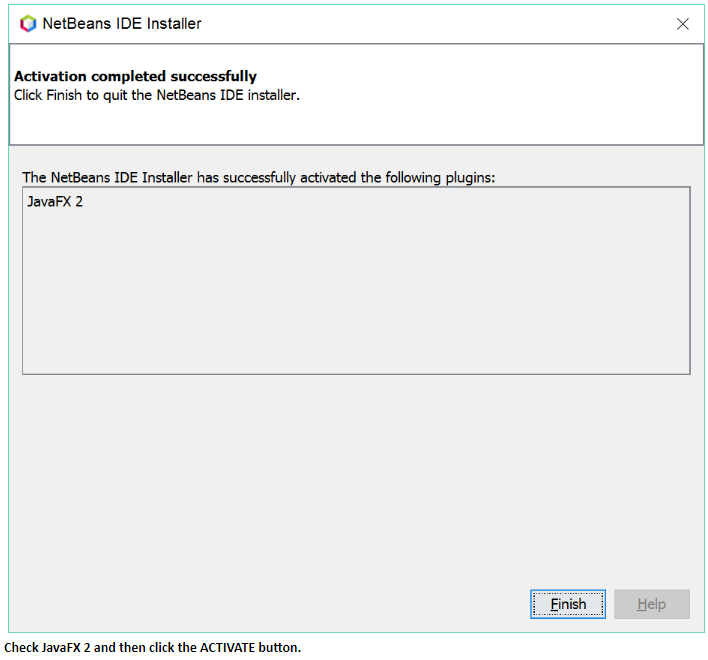
}

}

Save your changes.

Now we'll add two more files: a Controller and an FXML file.

1. Right-click the fxtest1 package in your project and select New > Other.
2. Select the item "JavaFX" in the Categories list and in the list on the right, select "Empty FXML". Then click NEXT. **If you don't see the JavaFX category**, press the CANCEL button and then do the following:
   * Go to Tools > Plugins
   * Click on the "Installed" tab
   * Locate the item "JavaFX 2" and check it.
   * At the bottom of the screen, click the ACTIVATE button.



* + You will then see a screen with the message "Activation completed successfully. Click the FINISH button at the bottom.
  + Restart NetBeans.
  + Now if you try Step 2 again, you should have the JavaFX category.

1. On the next screen, enter the FXML Name as "FXMLtest1". Make sure your fxml file will go in the right package, and then click the NEXT button.
2. On the third screen, check the box "Use Java Controller". For the Controller Name field, enter **TestingController**.
3. Click the FINISH button.

You should now have 2 new items in your fxtest1 package: the FXMLtest1.fxml and a FXMLtest1Controller.java class.

Open the FXMLTesting.fxml in your editor and paste the following code in between the <AnchorPane> </AnchorPane> tags:

<?xml version="1.0" encoding="UTF-8"?>

<?import java.lang.\*?>

<?import java.util.\*?>

<?import javafx.scene.\*?>

<?import javafx.scene.control.\*?>

<?import javafx.scene.layout.\*?>

<AnchorPane id="AnchorPane" prefHeight="400.0" prefWidth="600.0" xmlns:fx="http://javafx.com/fxml/1" fx:controller="fxtest1.FXMLtest1Controller">

<Button fx:id="btnClick" onAction="#click" AnchorPane.topAnchor="10.0" AnchorPane.leftAnchor="10.0" text="\_Click Me!" />

<Label fx:id="lblOutput" AnchorPane.topAnchor="50.0" AnchorPane.leftAnchor="10.0" />

</AnchorPane>

Now open the FXMLtest1Controller.java class and paste the following code inside the class, above the initialize() method:

package fxtest1;

import java.net.URL;

import java.util.ResourceBundle;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.fxml.Initializable;

import javafx.scene.control.Label;

/\*\*

\* FXML Controller class

\*

\* @author Rakesh Kantaria

\*/

public class FXMLtest1Controller implements Initializable {

/\*\*

\* Initializes the controller class.

\*/

@FXML

private Label lblOutput;

@FXML

public void click(ActionEvent event)

{

lblOutput.setText("Hello, World!");

}

@Override

public void initialize(URL url, ResourceBundle rb) {

// TODO

}

}

you'll be prompted to do some imports: Label is in javafx.scene.control and ActionEvent is in javafx.event. The @FXML annotation requires the FXML class in javafx.fxml.

Don't worry too much right now about the code you're pasting. We'll go over all of these things in the coming lessons.

Once you've added all the code, you can run the program in one of the following ways:

* While any of your project's files are in focus in the editor, press F6 or click the RUn button in the toolbar.
* While the Main class is open in the editor, press Shift-F6.
* Right-click the Main class in the project explorer and select "Run File"

You should see a screen with a button that you can click to display "Hello, World!" below.

