CSSE2002/7023
JavaFX Installation Guide
Version 1.0

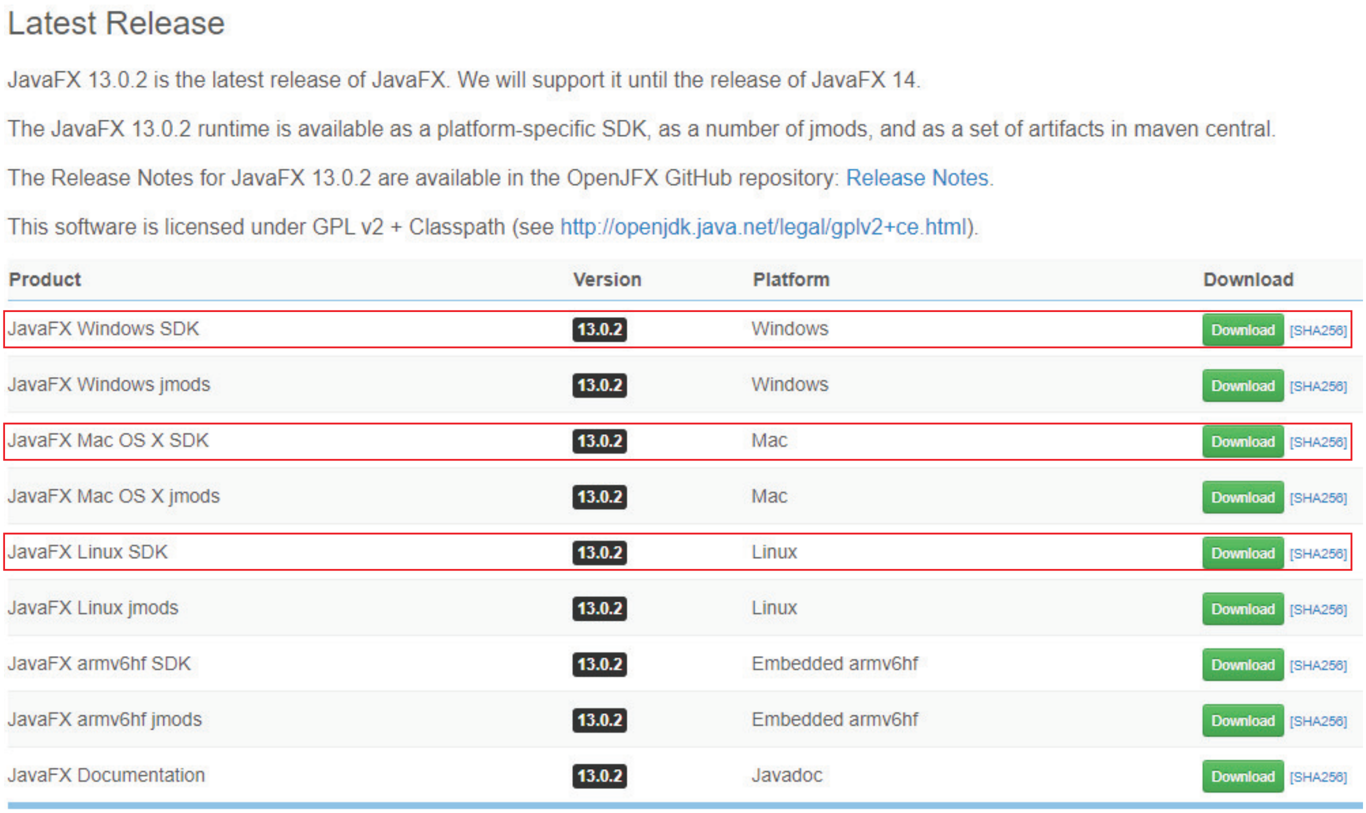
1 JavaFX

JavaFX is a set of libraries that provide a flexible approach to developing sophisticated interactive environments in Java. It is designed to deliver a consistent experience across platforms. These platforms include windowing systems, web interfaces, graphics and multi-media. JavaFX separates appearance from implementation. This allows developers to build the structure of the interface while user experience designers can separately work on the application's style. Customisation of the appearance is done via Cascading Style Sheets.

2 JavaFX Setup

Before Java 11, JavaFX was included with the JDK; however, now it must be installed as a standalone library. This comes with some additional steps in order to get up and running:

1. Download JavaFX from `https://gluonhq.com/products/javafx/`. Ensure you download the correct version for your JDK and your operating system. (You may also download the JavaDoc for JavaFX if you would like easy access to it. It is the last link shown in the screenshot below.)



Latest Release

JavaFX 13.0.2 is the latest release of JavaFX. We will support it until the release of JavaFX 14.

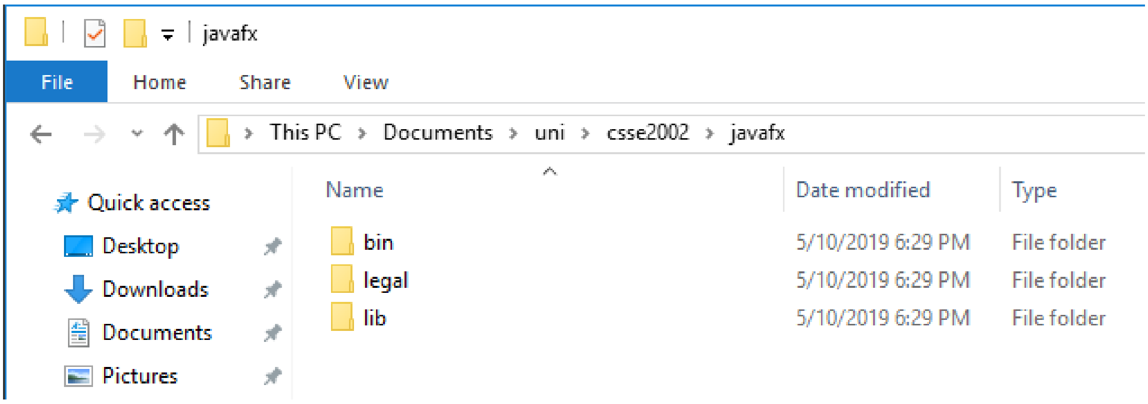
The JavaFX 13.0.2 runtime is available as a platform-specific SDK, as a number of jmods, and as a set of artifacts in maven central.

The Release Notes for JavaFX 13.0.2 are available in the OpenJFX GitHub repository: Release Notes.

This software is licensed under GPL v2 + Classpath (see http://openjdk.java.net/legal/gplv2+ce.html).

Product	Version	Platform	Download
JavaFX Windows SDK	13.0.2	Windows	Download [SHA256]
JavaFX Windows jmods	13.0.2	Windows	Download [SHA256]
JavaFX Mac OS X SDK	13.0.2	Mac	Download [SHA256]
JavaFX Mac OS X jmods	13.0.2	Mac	Download [SHA256]
JavaFX Linux SDK	13.0.2	Linux	Download [SHA256]
JavaFX Linux jmods	13.0.2	Linux	Download [SHA256]
JavaFX armv6hf SDK	13.0.2	Embedded armv6hf	Download [SHA256]
JavaFX armv6hf jmods	13.0.2	Embedded armv6hf	Download [SHA256]
JavaFX Documentation	13.0.2	Javadoc	Download [SHA256]

2. Once the download is complete, extract the SDK to a memorable location on your computer (On EAIT lab computers, put this in your H drive).



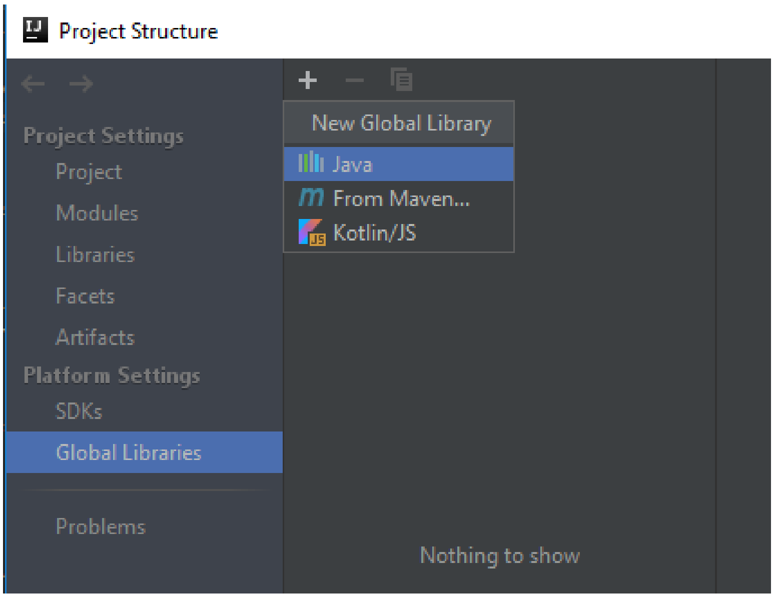
Name	Date modified	Type
bin	5/10/2019 6:29 PM	File folder
legal	5/10/2019 6:29 PM	File folder
lib	5/10/2019 6:29 PM	File folder

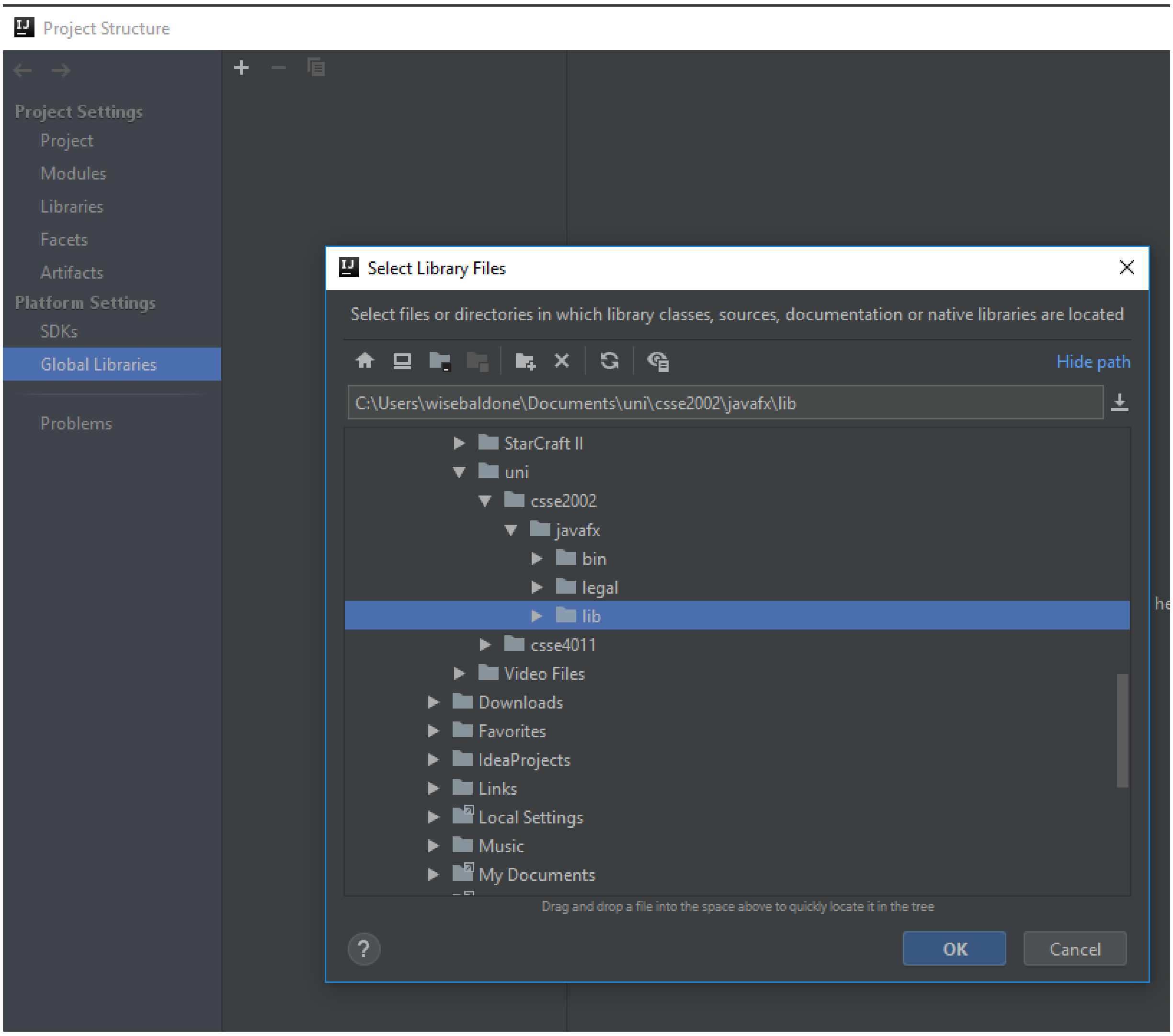
3. Create a new normal Java project in IntelliJ and create a class called `HelloFX` that contains the following code below:

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;

public class HelloFX extends Application {

    @Override
    public void start(Stage stage) {
        String javaVersion = System.getProperty("java.version");
        String javafxVersion = System.getProperty("javafx.version");
        Label l = new Label("Hello, JavaFX " +
            javafxVersion + ", running on Java " +
            javaVersion + ".");
        Scene scene = new Scene(new StackPane(l), 640, 480);
        stage.setScene(scene);
        stage.show();
    }

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

4. Add the libraries you extracted before as a global library for your new project by going to File -> Project Structure -> Global Libraries.

5. Add the library by clicking the '+' symbol and selecting Java as the New Global Library navigate to the `lib` folder within the OpenJFX SDK you extracted earlier.





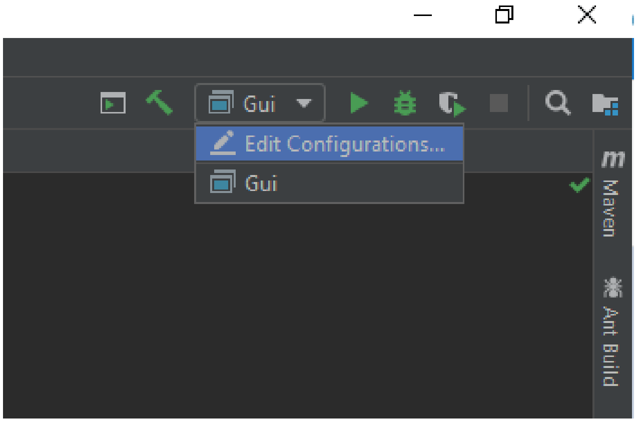
Select files or directories in which library classes, sources, documentation or native libraries are located

C:\Users\wisebaldone\Documents\uni\csse2002\javafx\lib

Drag and drop a file into the space above to quickly locate it in the tree

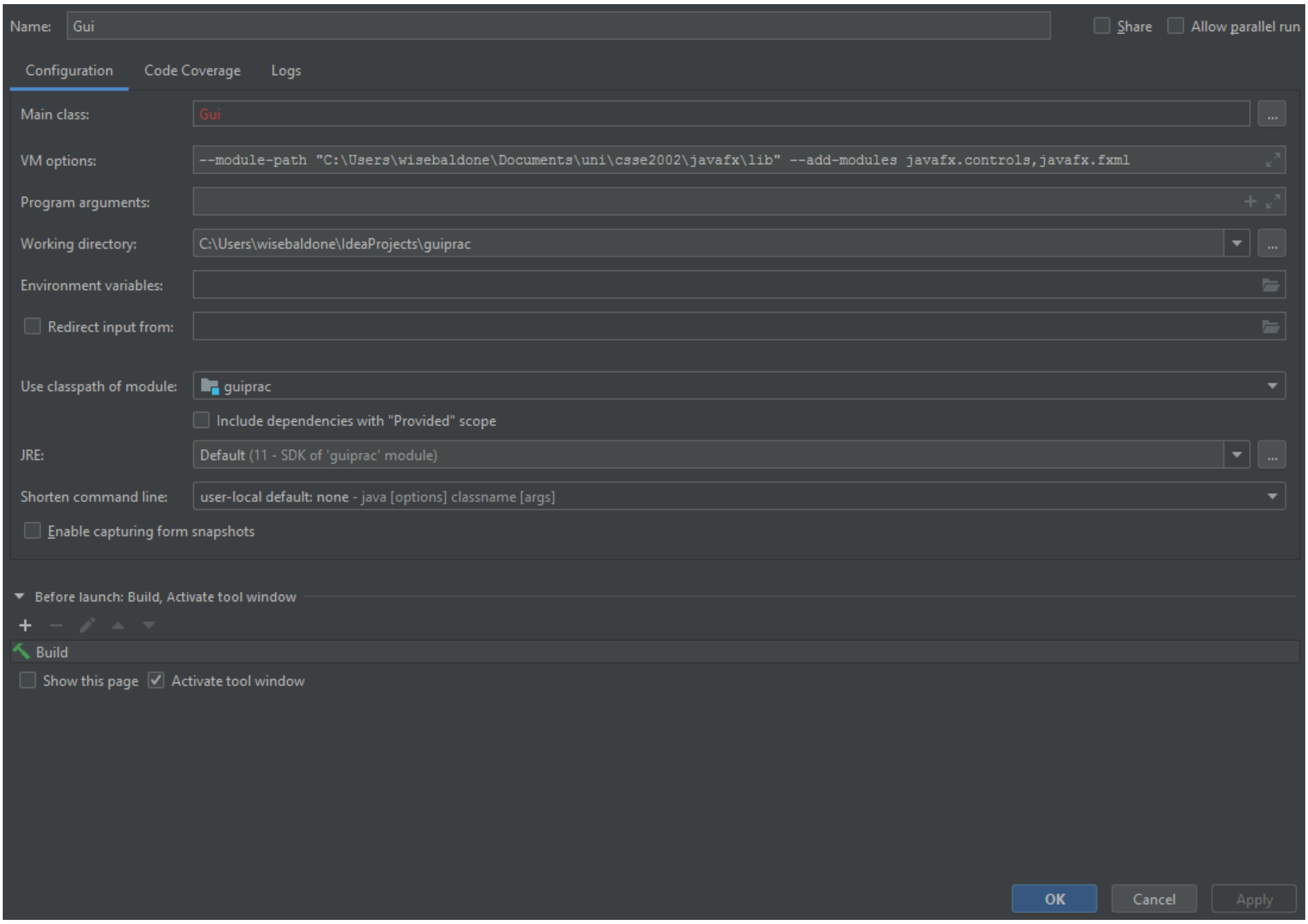
6. Click the play button for the `Launcher` class. This should produce an error.

7. To fix this we need to edit the configuration of the program by selecting it from the "Run" dropdown menu and selecting "Edit Configurations...".



8. Add the below arguments to “VM Options”:
`--module-path "[location of SDK]" --add-modules javafx.controls,javafx.fxml`

   You will need to change the module path to the actual location of your JavaFX SDK. See the below screenshot for an example for Windows:



Name: Gui □ Share □ Allow parallel run

Configuration | Code Coverage | Logs

Main class: Gui

VM options: `--module-path "C:\Users\wisebaldone\Documents\uni\csse2002\javafx\lib" --add-modules javafx.controls,javafx.fxml`

Program arguments:

Working directory: C:\Users\wisebaldone\IdeaProjects\guiprac

Environment variables:

□ Redirect input from:

Use classpath of module: guiprac

□ Include dependencies with "Provided" scope

JRE: Default (11 - SDK of 'guiprac' module)

Shorten command line: user-local default: none - java [options] classname [args]

□ Enable capturing form snapshots

Before launch: Build, Activate tool window

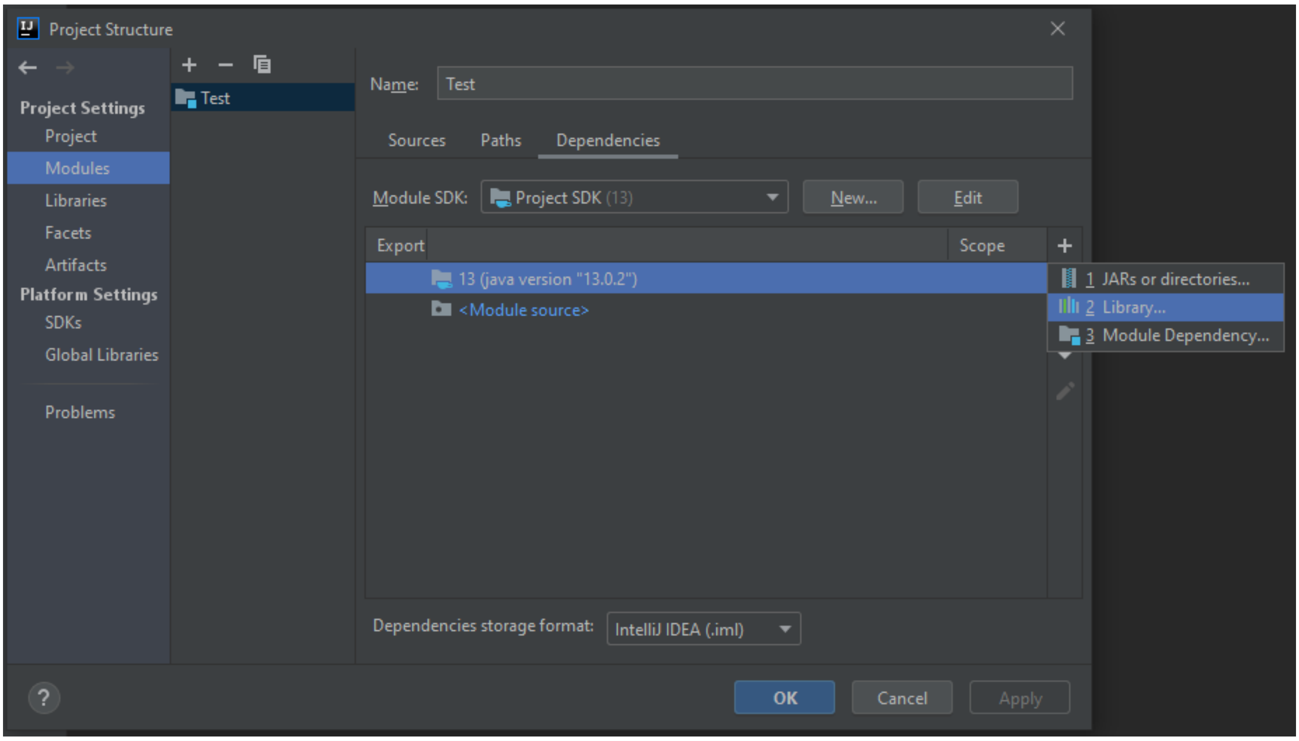
Build

□ Show this page ☑ Activate tool window

9. You should now be able to run the application and a window should display successfully.

You can follow most of these same steps to set up another project to use the JavaFX libraries, for example your assignment two project. Steps 4 and 5 are different, as you have already added JavaFX as a global library. You now have to tell your project to use this library.

4. Add the JavaFX global library to your new project by going to File -> Project Structure -> Modules, and selecting the Dependencies tab. Click the '+' button on the right of the Modules Dependencies window and select 2 Library....



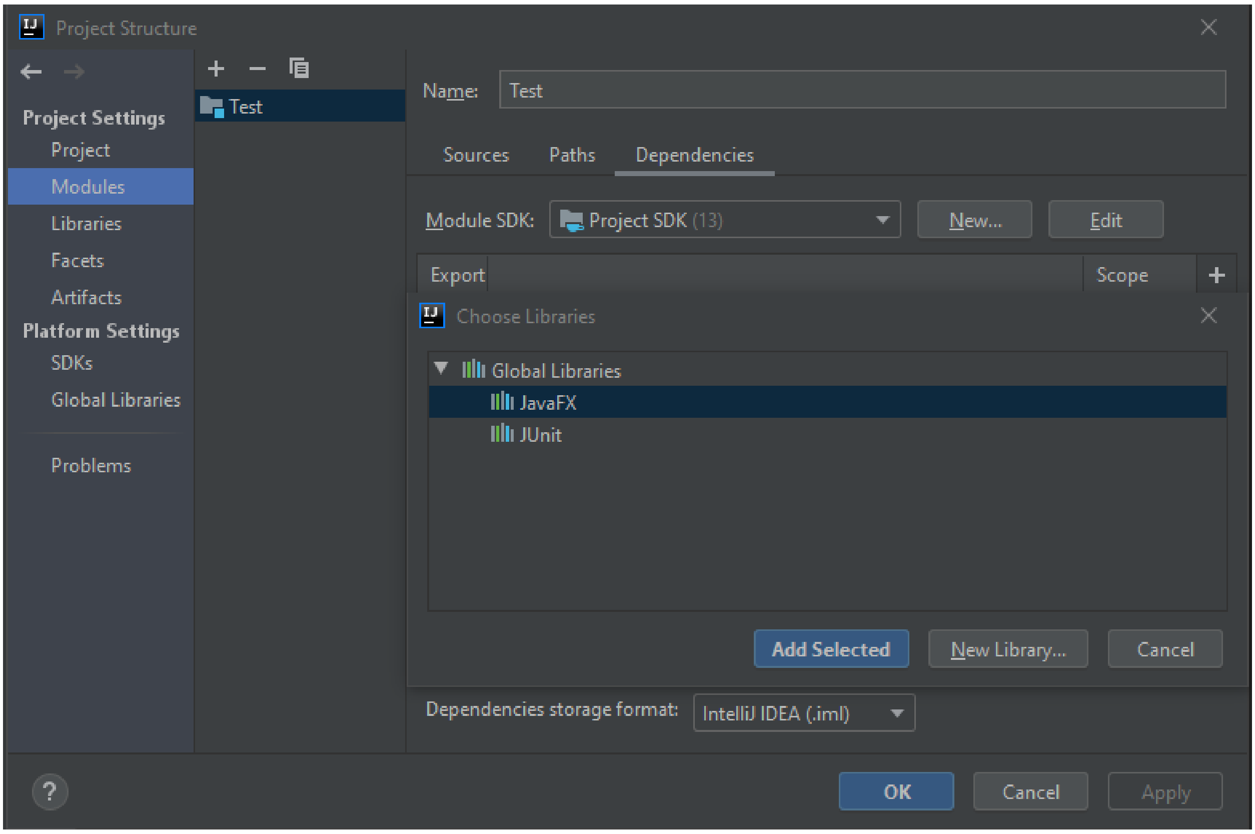
Name: Test

Module SDK: Project SDK (13)

13 (java version "13.0.2")
<Module source>

Dependencies storage format: IntelliJ IDEA (.iml)

5. Select JavaFX from the Global Libraries list and click on Add Selected.



Name: Test

Module SDK: Project SDK (13)

Dependencies storage format: IntelliJ IDEA (.iml)