Setup JavaFX with JDK 13

Downloads

JDK 13   Documentation

JavaFX Windows SDK   SceneBuilder

IntelliJ setup

Download the appropriate JavaFX SDK for your operating system and unzip it to a desired location, for instance

C:\Program Files\Java\javafx-sdk-13

IntelliJ setup

Define the JDK in IntelliJ IDEA

- Open the **Project Structure** dialog (e.g. Ctrl+Shift+Alt+S).
- In the leftmost pane, under Platform **Settings**, click SDKs.
- Above the pane to the right, click + and select **JDK 13**.
- In the dialog that opens, select the installation directory of the **JDK** to be used and click OK (C:\Program Files\Java\jdk-13)

IntelliJ setup

Setup SceneBuilder

- Open the Settings dialog (e.g. Ctrl+Alt+S).
- In the leftmost pane, under Platform **Languages&Frameworks**, click **JavaFX**.
- On the right side locate and set the path to the **SceneBuilder** executable.

By default it is found in

C:\Program Files\SceneBuilder

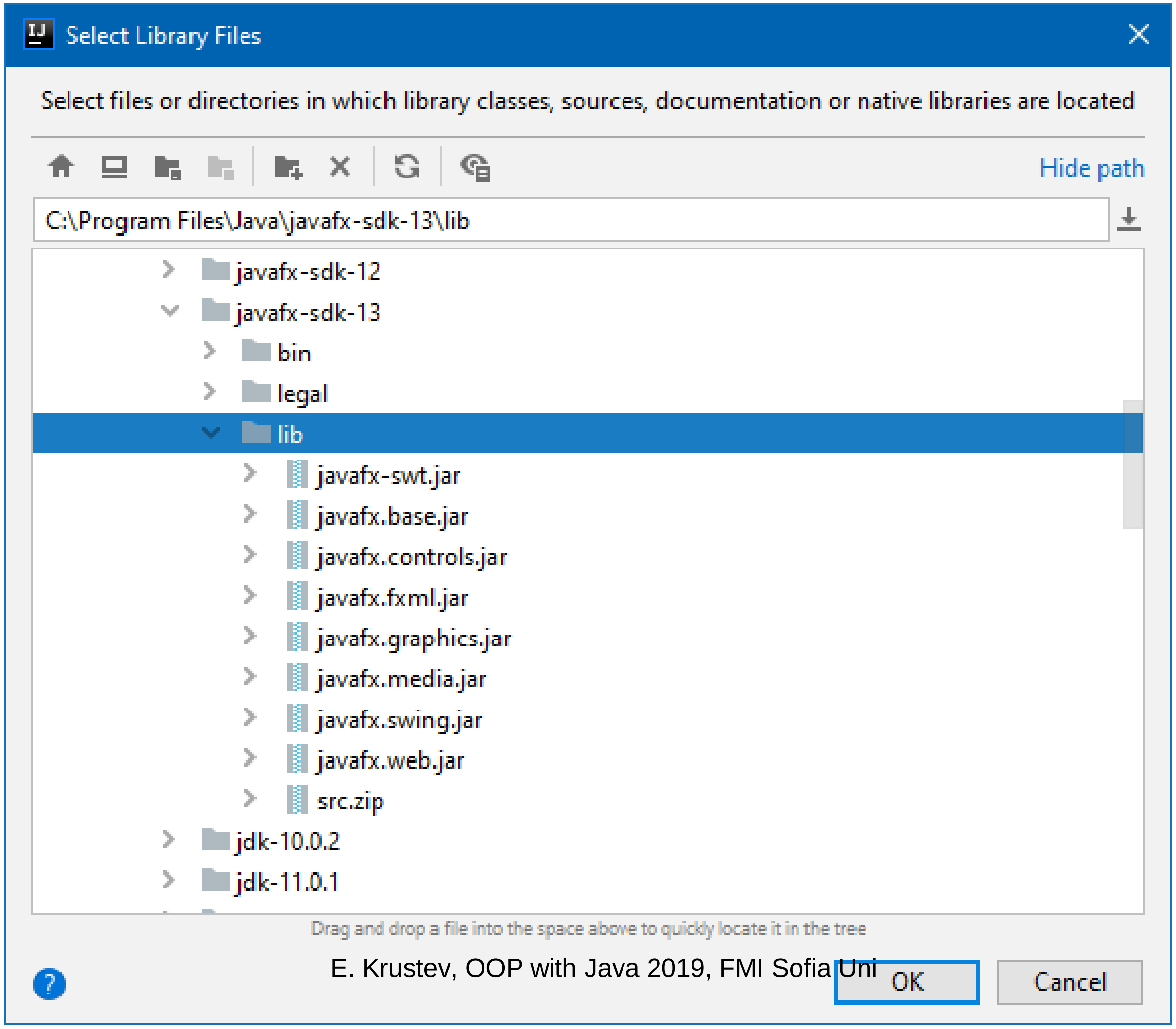
IntelliJ setup

Setup JavaFX with JDK 13 as a **Global library**

- Open the Project Structure dialog (e.g. Ctrl+Shift+Alt+S).
- Select **Global Libraries**
- **Click + to add for Java the location of** the **lib** directory (Library-> Java) where you have unpacked JavaFX (for me,

C:\Program Files\Java\javafx-sdk-13\lib).

IntelliJ setup


Select Library Files

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

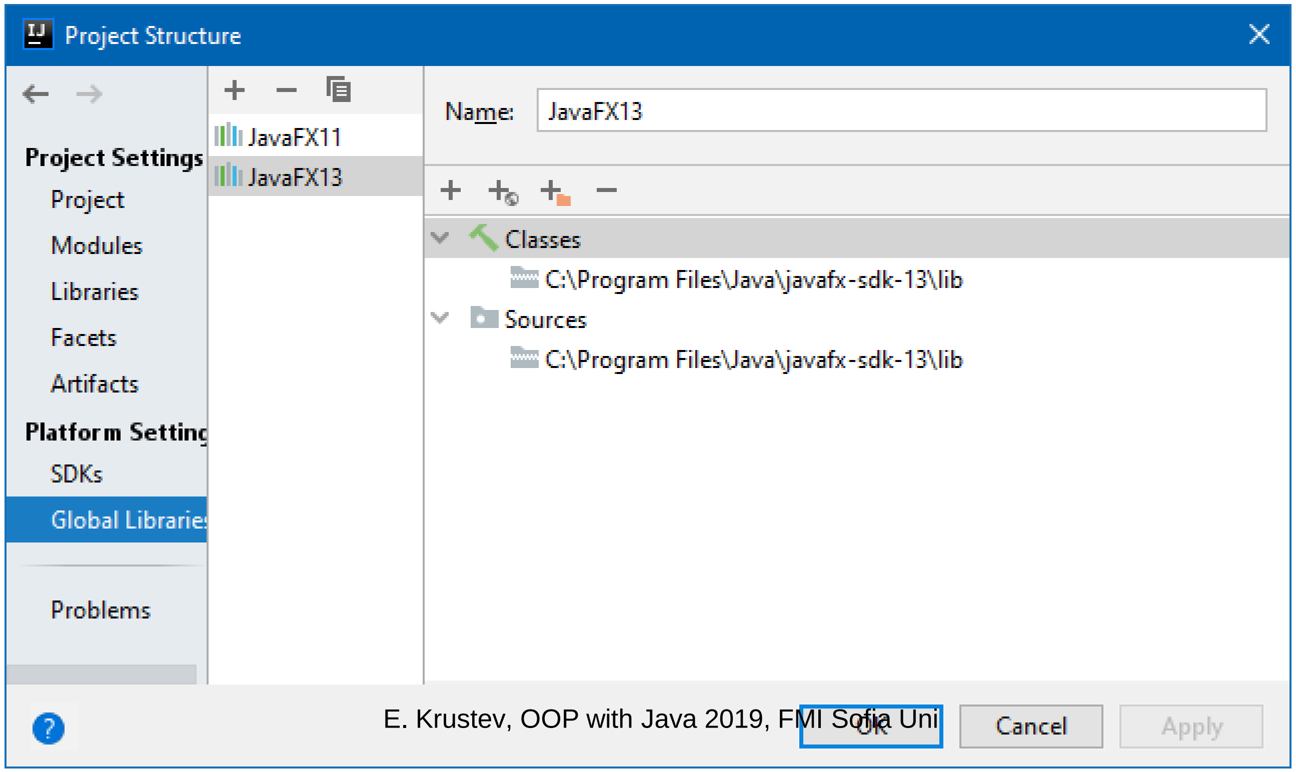
C:\Program Files\Java\javafx-sdk-13\lib

```
javafx-sdk-12
javafx-sdk-13
    bin
    legal
    lib
        javafx-swt.jar
        javafx.base.jar
        javafx.controls.jar
        javafx.fxml.jar
        javafx.graphics.jar
        javafx.media.jar
        javafx.swing.jar
        javafx.web.jar
        src.zip
jdk-10.0.2
jdk-11.0.1
```

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

IntelliJ setup

Assign a descriptive name for the Global library, for example **JavaFX13**

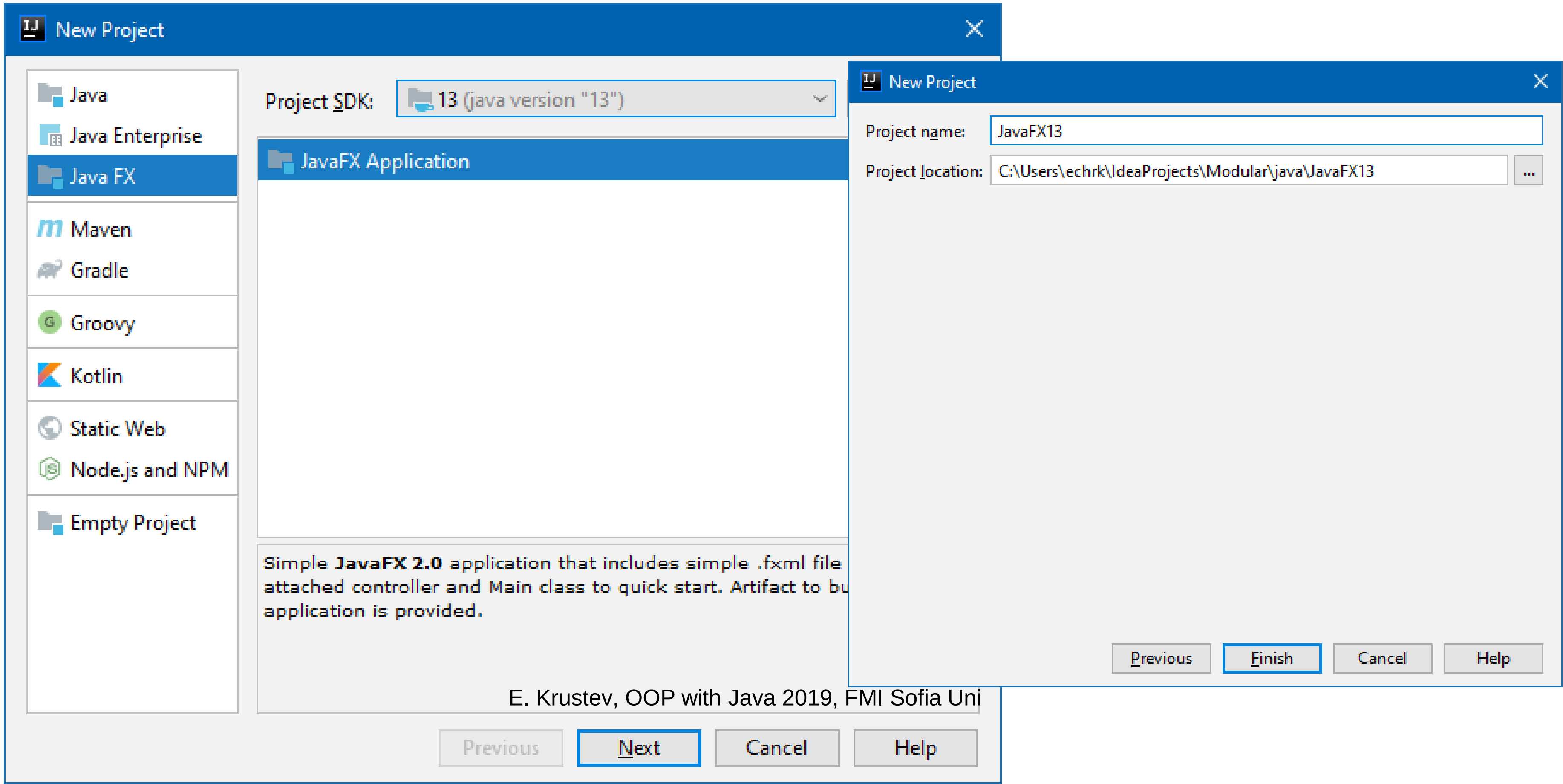


Name: JavaFX13

```
Classes
    C:\Program Files\Java\javafx-sdk-13\lib
Sources
    C:\Program Files\Java\javafx-sdk-13\lib
```

Non-Modular JavaFX project

Create a JavaFX project in IntelliJ in JDK 13. Use JDK 13



Project SDK: 13 (java version "13")

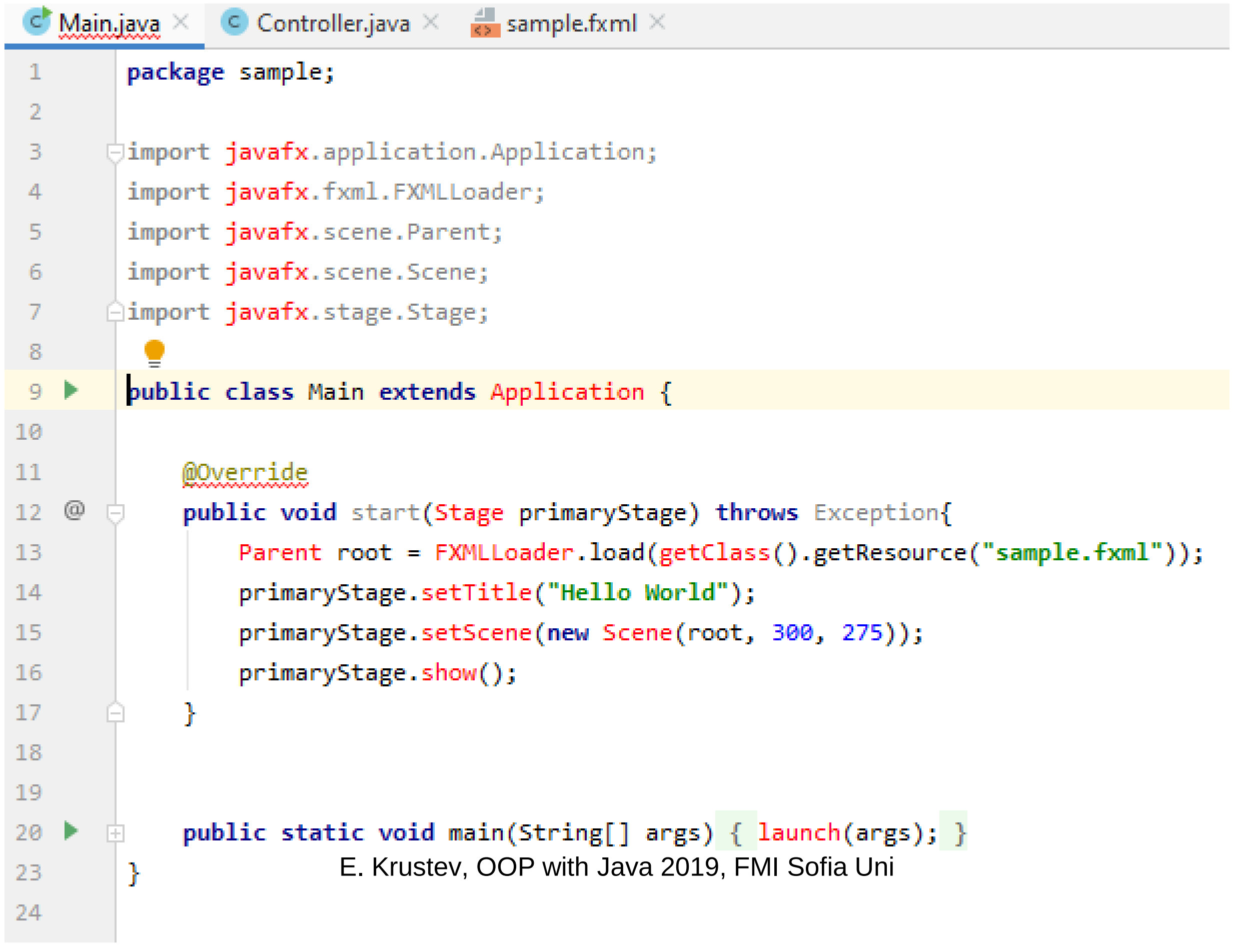
Simple **JavaFX 2.0** application that includes simple .fxml file attached controller and Main class to quick start. Artifact to bu application is provided.

Project name: JavaFX13

Project location: C:\Users\echrk\IdeaProjects\Modular\java\JavaFX13

Non-Modular JavaFX project

Initially JavaFX 13 is not recognized



Main.java × Controller.java × sample.fxml ×

```
package sample;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

public class Main extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception{
        Parent root = FXMLLoader.load(getClass().getResource("sample.fxml"));
        primaryStage.setTitle("Hello World");
        primaryStage.setScene(new Scene(root, 300, 275));
        primaryStage.show();
    }


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

```

Non-Modular JavaFX project

Select **File->Project Structure->Project structure**

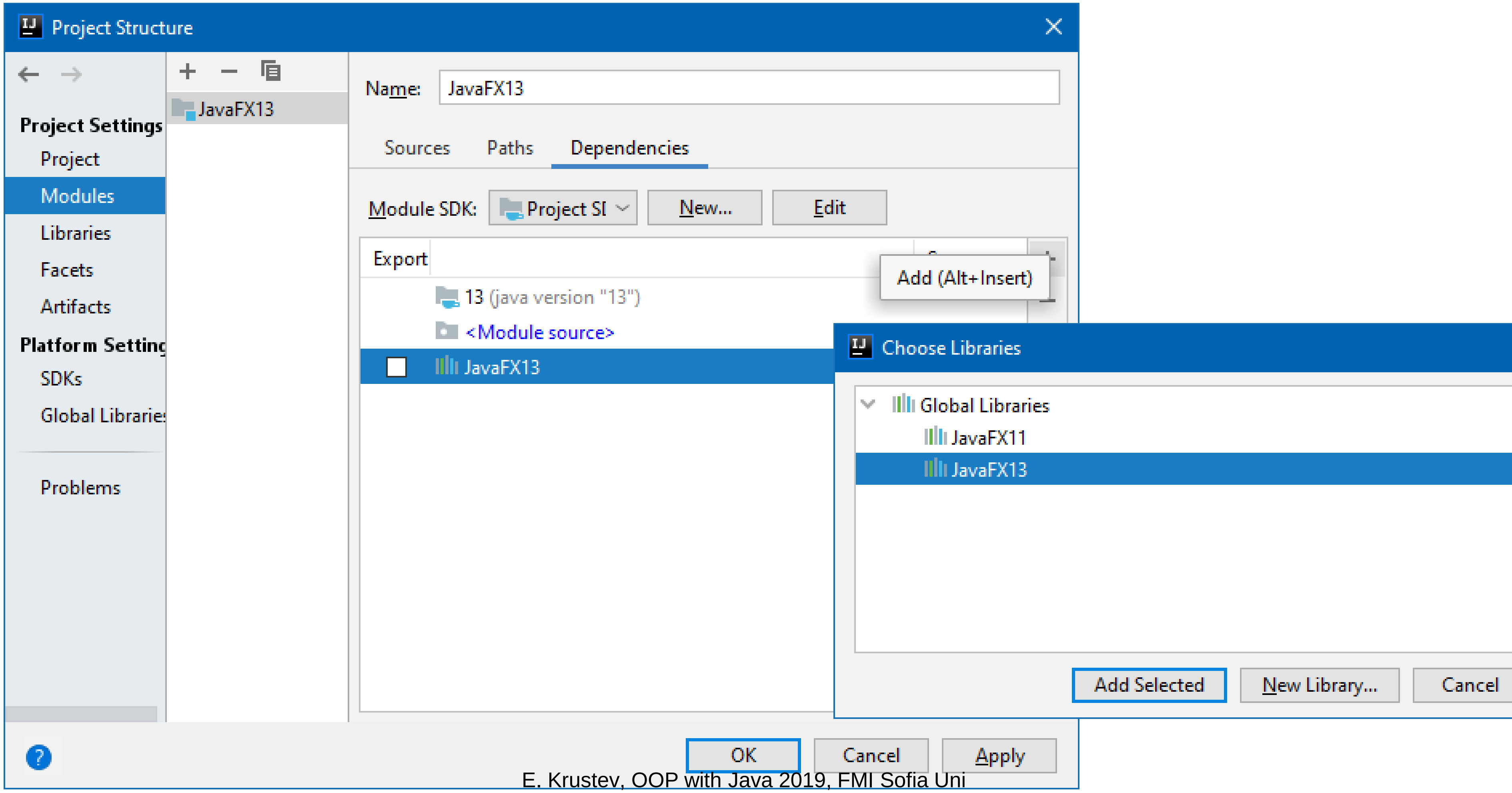
Select **Modules**

In the Dependencies tab click + *(on the rightmost location)* **and Select Library**

Among the Global Libraries select the previously create JavaFX library (click Add selected)

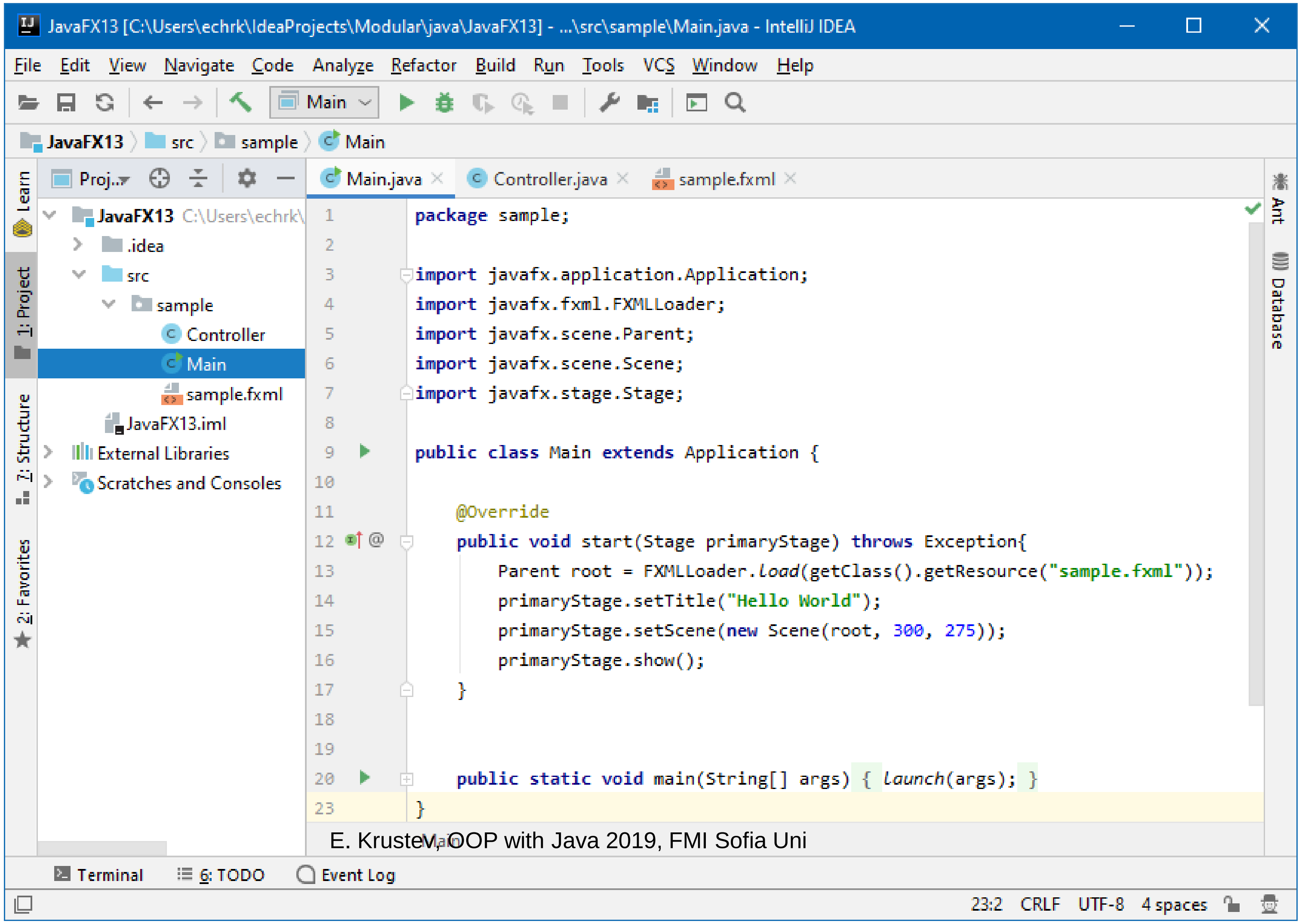
Click OK

Non-Modular JavaFX project



Non-Modular JavaFX project

Now you can compile JavaFX 13 source with and JDK 13


JavaFX13 [C:\Users\echrk\IdeaProjects\Modular\java\JavaFX13] - ...\src\sample\Main.java - IntelliJ IDEA
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
JavaFX13 src sample Main
Main.java Controller.java sample.fxml
JavaFX13 C:\Users\echrk\
.idea
src
sample
Controller
Main
sample.fxml
JavaFX13.iml
External Libraries
Scratches and Consoles
Terminal
6: TODO
Event Log
23:2 CRLF UTF-8 4 spaces

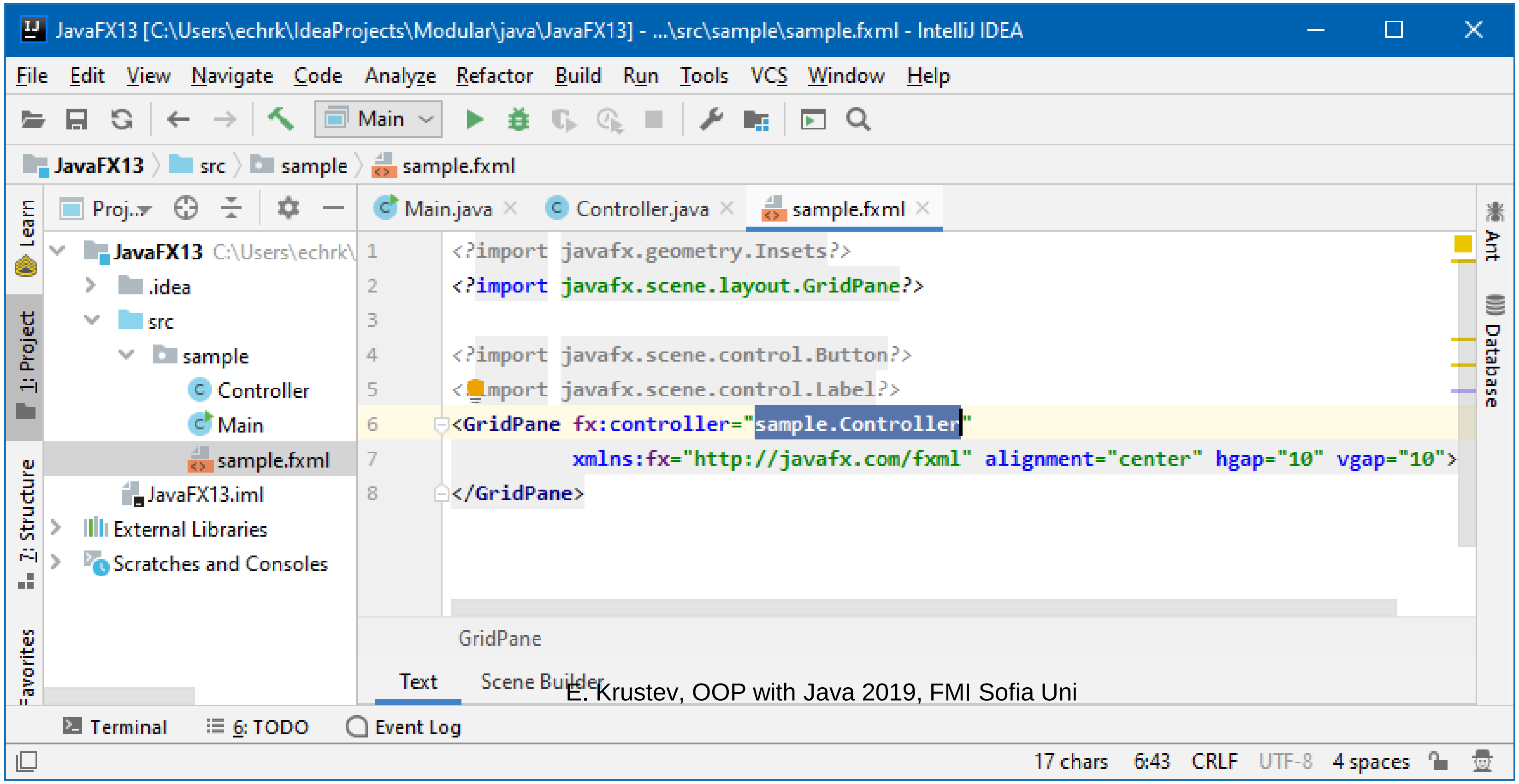
```java
package sample;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

public class Main extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception{
        Parent root = FXMLLoader.load(getClass().getResource("sample.fxml"));
        primaryStage.setTitle("Hello World");
        primaryStage.setScene(new Scene(root, 300, 275));
        primaryStage.show();
    }


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

Non-Modular JavaFX project

Select the file (FXML) of the Scene and click the Tab **SceneBuilder** to edit the Scene with SceneBuilder

or Right click it to select **Open in Scene Builder (better!)**

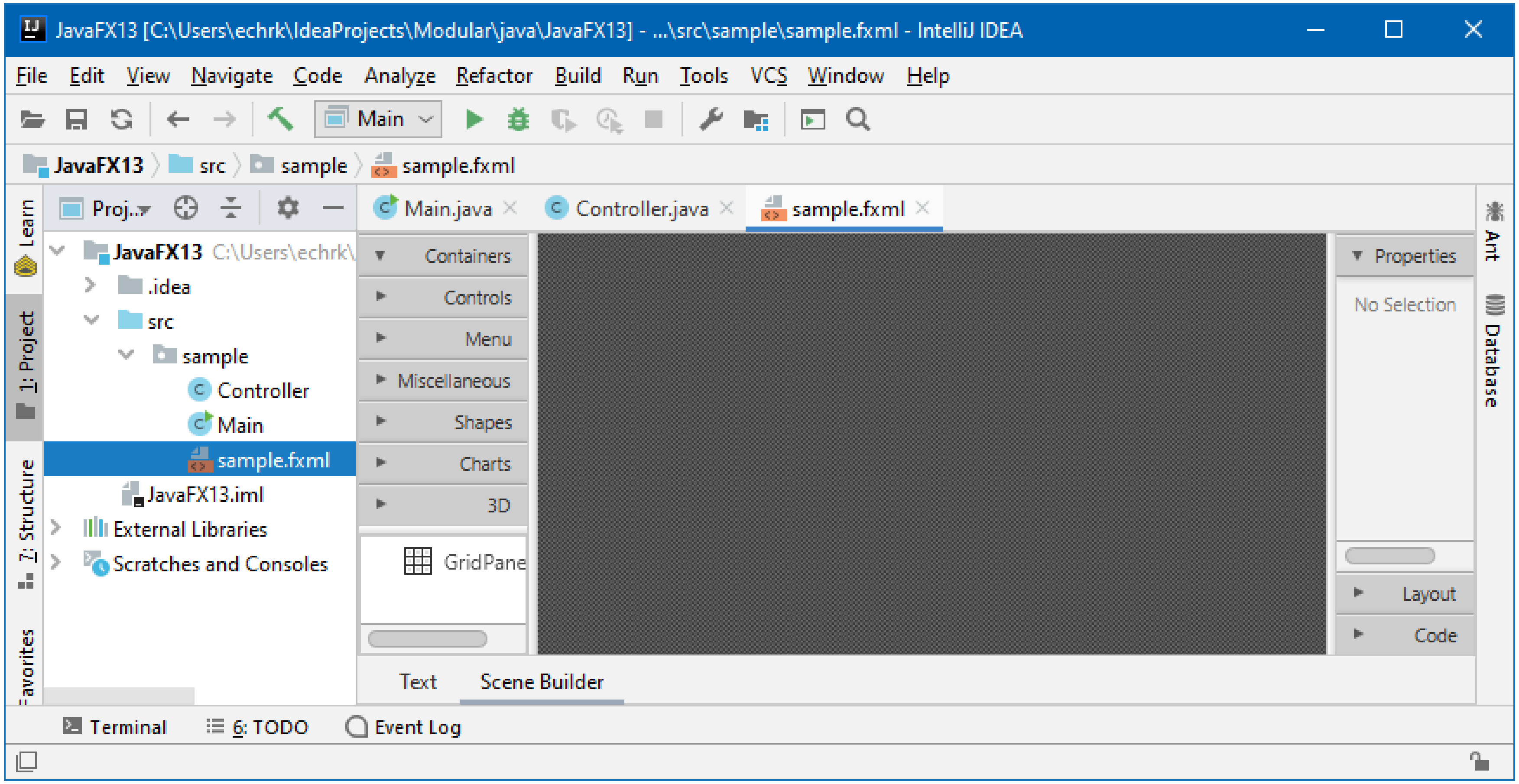
Note: `fx:controller` must be the name of `sample.Controller.java` (incl. package name)


JavaFX13 [C:\Users\echrk\IdeaProjects\Modular\java\JavaFX13] - ...\src\sample\sample.fxml - IntelliJ IDEA
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Main
JavaFX13 src sample sample.fxml
Proj...
JavaFX13 C:\Users\echrk\
.idea
src
sample
Controller
Main
sample.fxml
JavaFX13.iml
External Libraries
Scratches and Consoles
Main.java
Controller.java
sample.fxml
GridPane
Text
Scene Builder
Terminal
6: TODO
Event Log
17 chars 6:43 CRLF UTF-8 4 spaces

```xml
<?import javafx.geometry.Insets?>
<?import javafx.scene.layout.GridPane?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<GridPane fx:controller="sample.Controller"
          xmlns:fx="http://javafx.com/fxml" alignment="center" hgap="10" vgap="10">
</GridPane>
```

Non-Modular JavaFX project

Edit the Scene with SceneBuilder



Non-Modular JavaFX project

Warning: If you run now the project it will compile but you will get this error:

Error: JavaFX runtime components are missing, and are required to run this application

This error is shown since the **Java 13** launcher checks if the `main` class extends **`javafx.application.Application`**. If that is the case, it is required to have to **add** the **`javafx.graphics`** module on the **module-path**.

4. Add VM options to resolve the problem

```
--module-path "C:\Program Files\Java\javafx-sdk-13\lib"
```

or

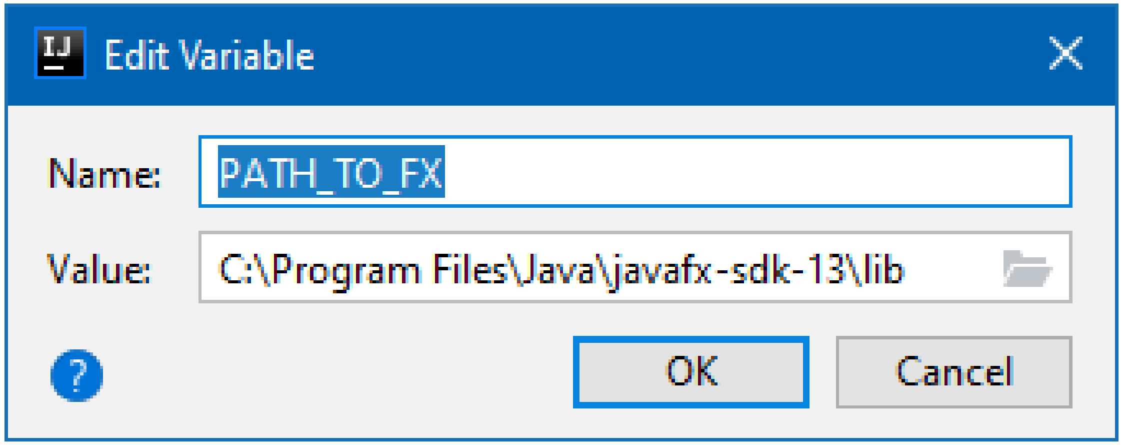
```
-p "C:\Program Files\Java\javafx-sdk-13\lib"
--add-modules javafx.controls,javafx.fxml
```

Note that the default project created by IntelliJ uses FXML, so **`javafx.fxml`** is required along with **`javafx.controls`**. If your project uses other modules, you will need to add them

Non-Modular JavaFX project

Alternatively, you can define a **global variable** that can be used in future projects.

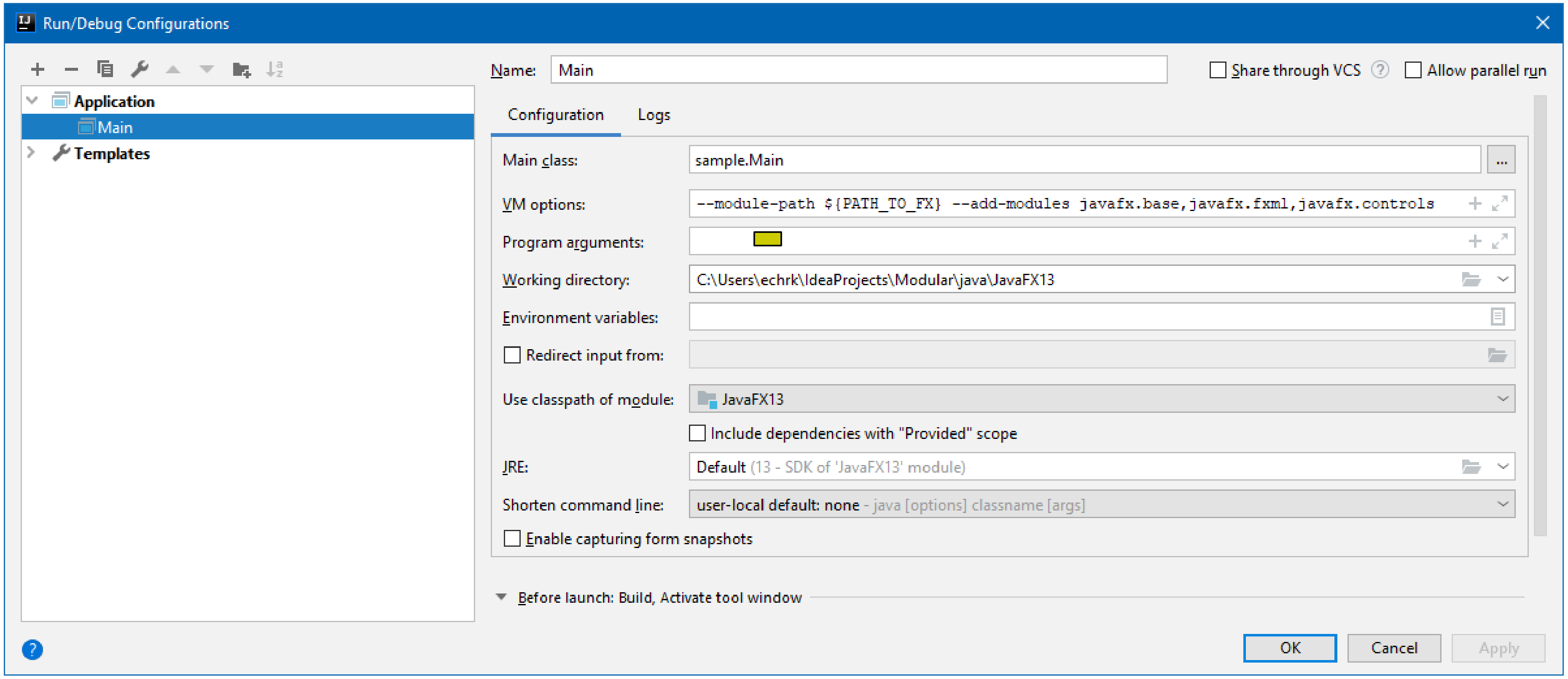
Go to **File -> Settings** -> **Appearance & Behavior** -> **Path Variables**, and define the name of the variable as **PATH_TO_FX**, and browse to the **lib** folder of the JavaFX SDK to set its value, and click **Apply**


Edit Variable

Name: PATH_TO_FX

Value: C:\Program Files\Java\javafx-sdk-13\lib

Non-Modular JavaFX project



Run/Debug Configurations

- Application
  - Main
- Templates

Name: Main

☐ Share through VCS ☐ Allow parallel run

Configuration | Logs

Main class: sample.Main

VM options: `--module-path ${PATH_TO_FX} --add-modules javafx.base,javafx.fxml,javafx.controls`

Program arguments:

Working directory: C:\Users\echrk\IdeaProjects\Modular\java\JavaFX13

Environment variables:

☐ Redirect input from:

Use classpath of module: JavaFX13

☐ Include dependencies with "Provided" scope

JRE: Default (13 - SDK of 'JavaFX13' module)

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

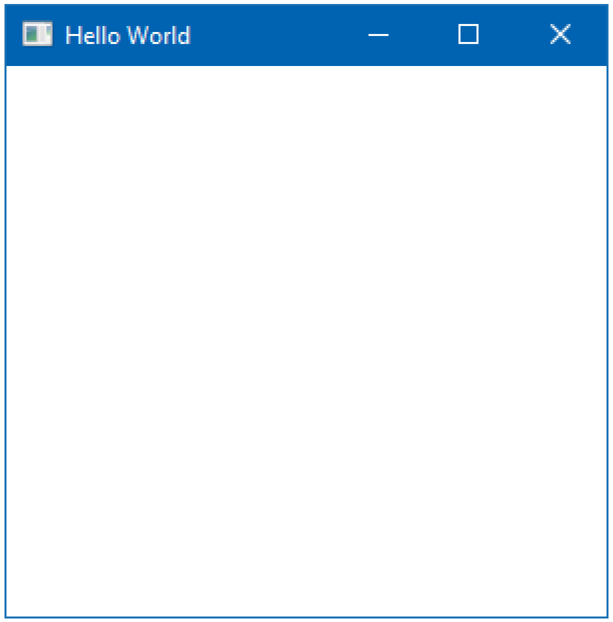
☐ Enable capturing form snapshots

Before launch: Build, Activate tool window

OK | Cancel | Apply

Non-Modular JavaFX project

Now, Run the JavaFX 13 application and see the default window



Happy Object Oriented Programming with

JavaFX 11+