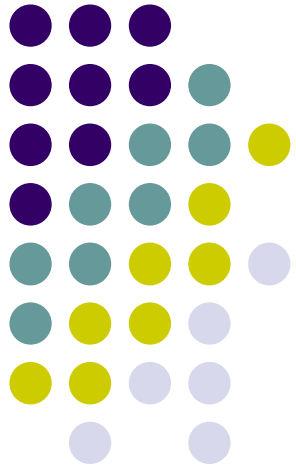
Setup JavaFX

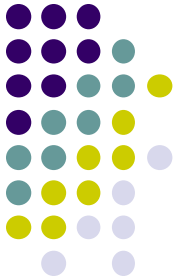
Downloads

JDK Documentation

JavaFX Windows SDK SceneBuilder



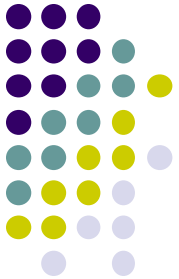
IntelliJ setup



Complete the following tasks before starting any new JavaFX project

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

C:\Program Files\Java\javafx-sdk-21

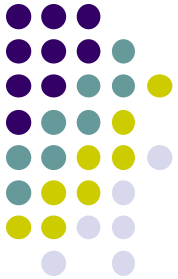


IntelliJ setup

2. Define the JDK in IntelliJ IDEA

- Open the **Project Settings** 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 **Add JDK**
- In the dialog that opens, select the location of the **JDK** to be used and click OK

  (C:\Program Files\Java\jdk-21)



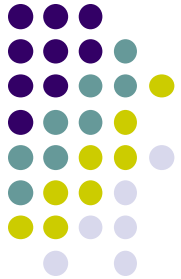
IntelliJ setup

3. 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:\Users\<username>\AppData\Local\SceneBuilder\SceneBuilder.exe



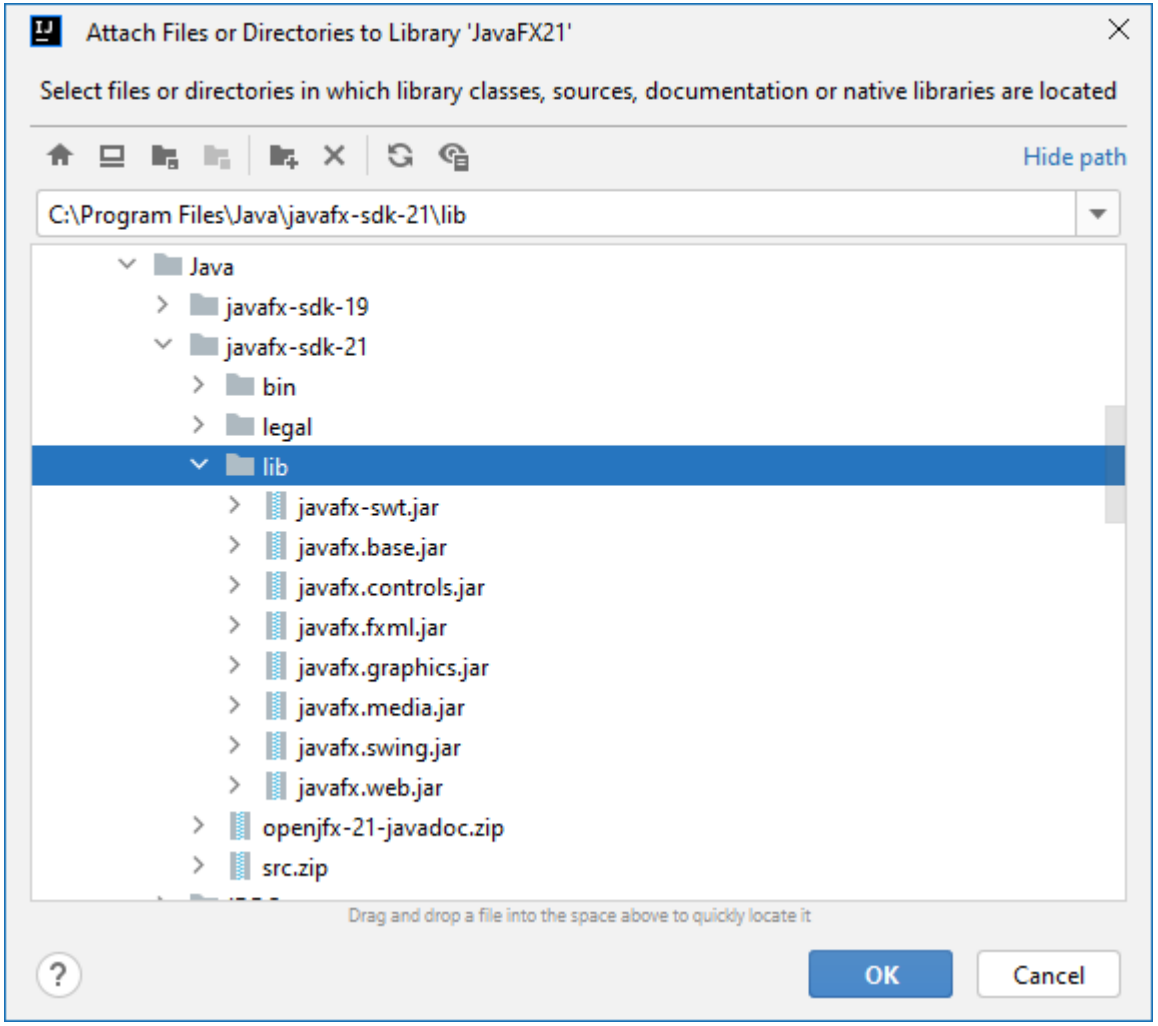
IntelliJ setup

4. Setup JavaFX as a **Global library**

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

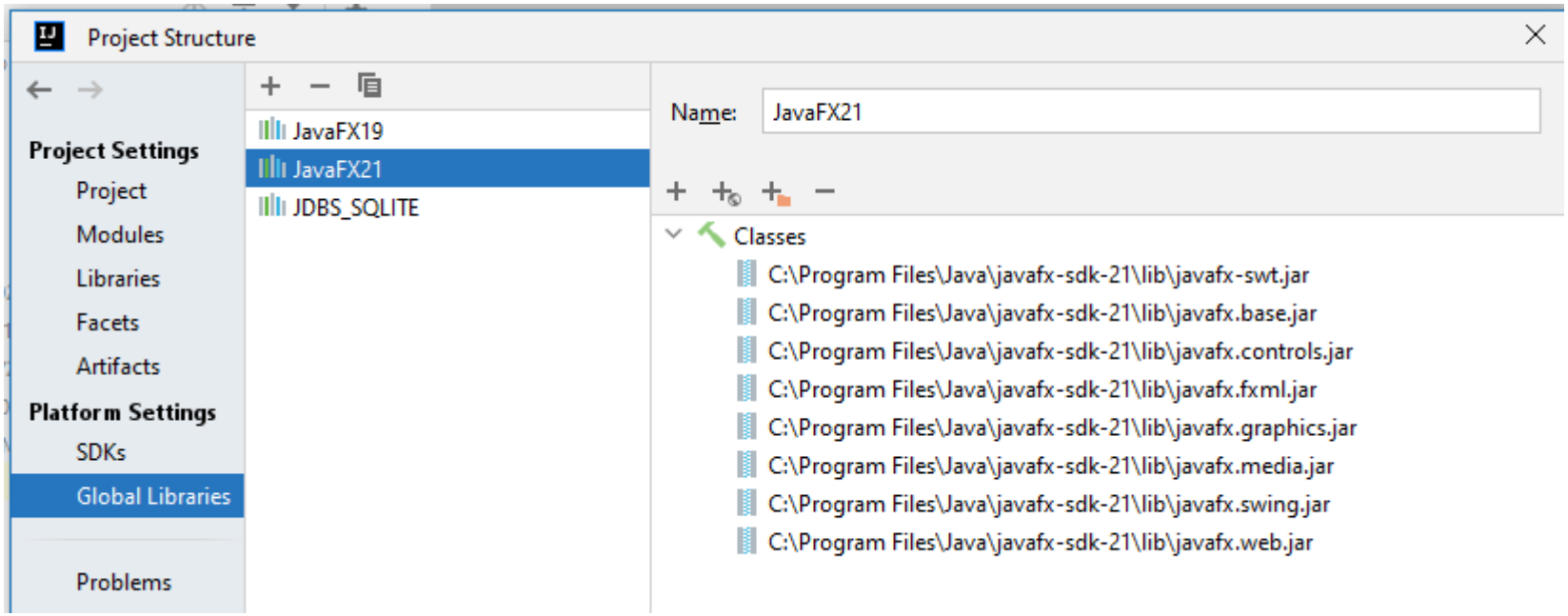
C:\Program Files\Java\javafx-sdk-21\lib).

IntelliJ setup



IntelliJ setup

Assign a descriptive name for the Global library, for example **JavaFX21** in this example



IntelliJ setup

5. Add path variables in Settings

- Open the Settings dialog (e.g. Ctrl+Alt+S).
- Add Path variables (use the + to add these vars)

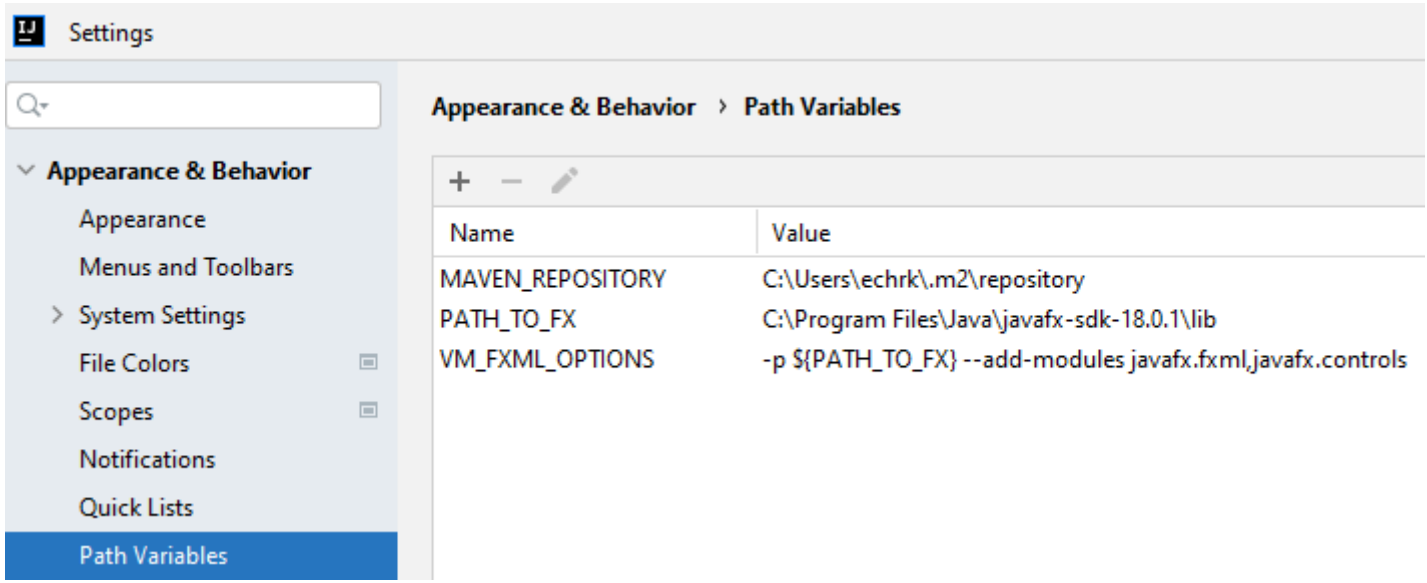
PATH_TO_FX

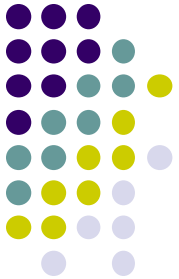
(points to the Lib folder of JavaFX)

VM_FXML_OPTIONS has the following setting

-p ${PATH_TO_FX} --add-modules javafx.fxml,javafx.controls

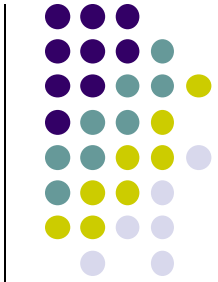
(copy this text and insert it where required)



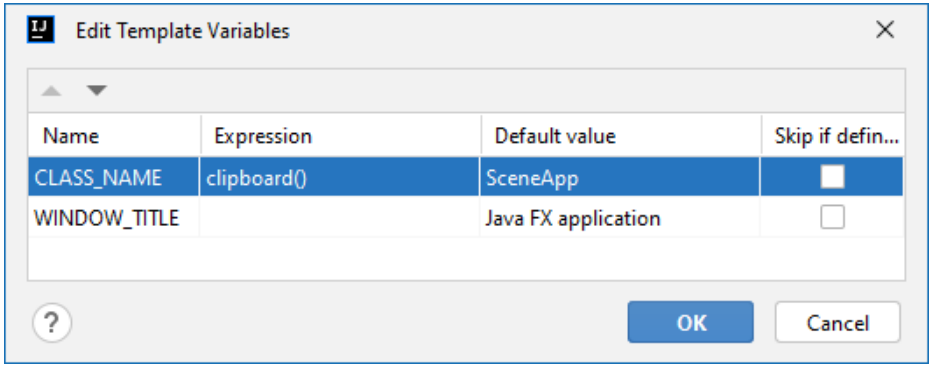


6. Create a Live template with the following text. Name the template fx-app (use the attached LiveTemplateJfx.txt)

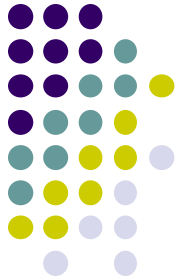
Live Template for simple JFX app



```
import javafx.application.Application;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.paint.Color;
import javafx.scene.shape.Line;
import javafx.stage.Stage;

// !Copy to Clipboard the Classname of the original class before overwriting it with this Template

public class $CLASS_NAME$ extends javafx.application.Application {

       @java.lang.Override
    public void start(javafx.stage.Stage primaryStage) {
        javafx.scene.Group group = new javafx.scene.Group();
        javafx.scene.Scene scene = new javafx.scene.Scene(group, 300, 300);

        // Add Nodes to scene

        primaryStage.setTitle("$WINDOW_TITLE$");
        primaryStage.setScene(scene);
        primaryStage.show();
    }

    /**
     * @param args the command line arguments
     */
    public static void main(java.lang.String[] args) {
        launch(args);
    }

}
```



Non-Modular JavaFX project



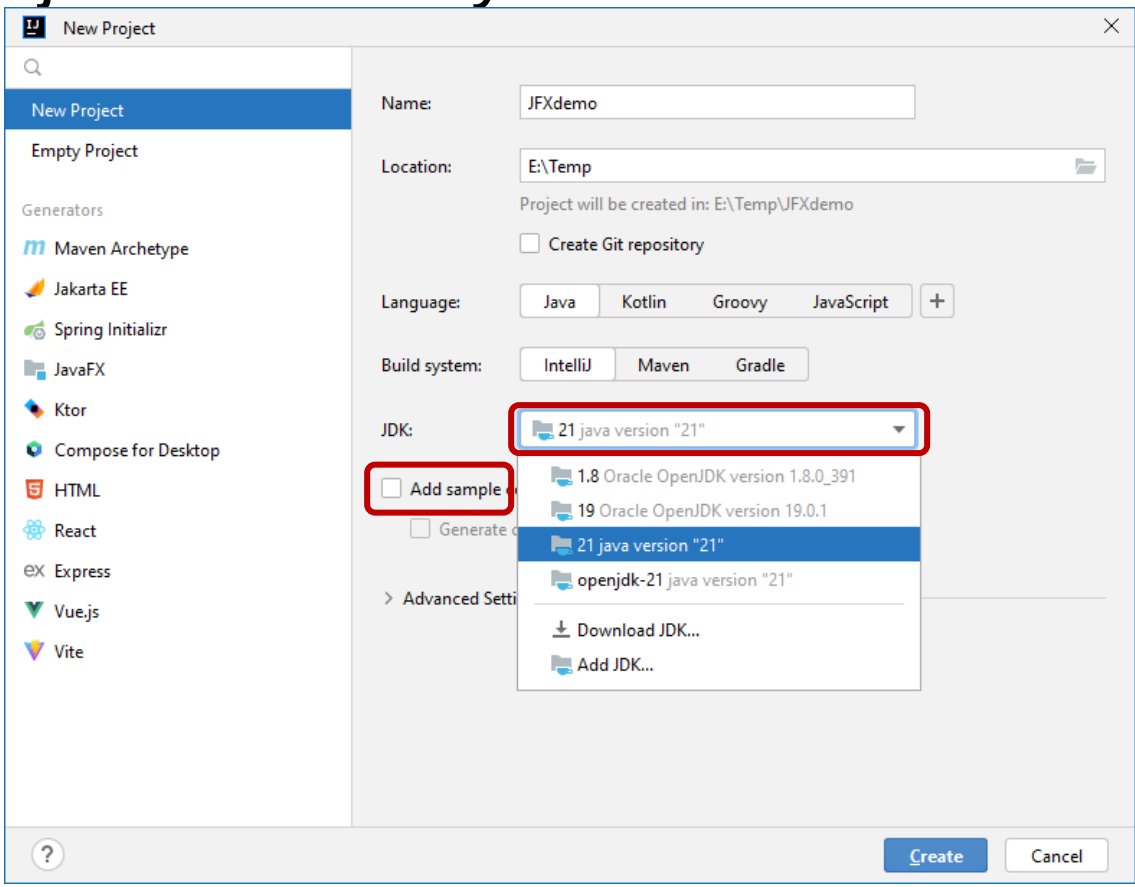
Once you apply the template in the JavaFX app file, update the values for the placeholders denoted as $....$ in the live template with their actual values.

Non-Modular JavaFX project

Create a JavaFX project in IntelliJ

1. Create a New Project as usually

A. Use **descriptive name** for the project
B. Select the **location** for the project
C. Use **defaults for Language** (Java) and Build options (IntelliJ)
D. Don’t use sample code (**uncheck the checkbox**)
E. Press button **Create**

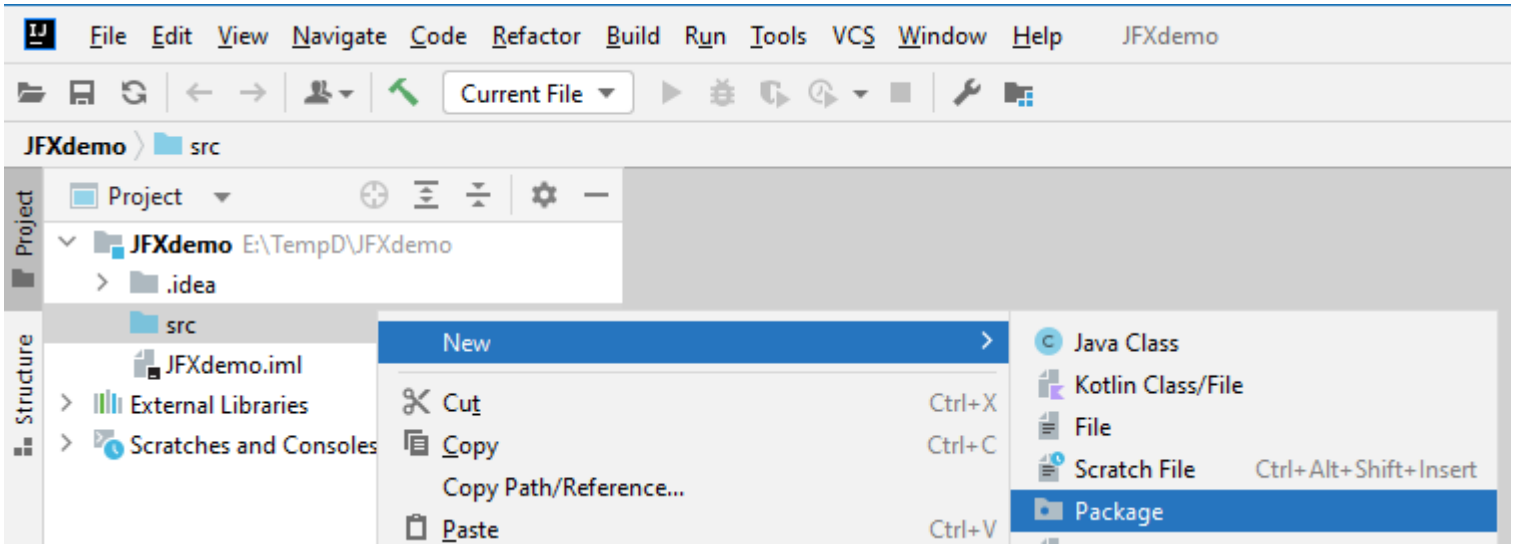


Non-Modular JavaFX project

Create a JavaFX project in IntelliJ

2. Add a **New package** to the src root as usually

Give a name of your choice for the package, for example, name it view

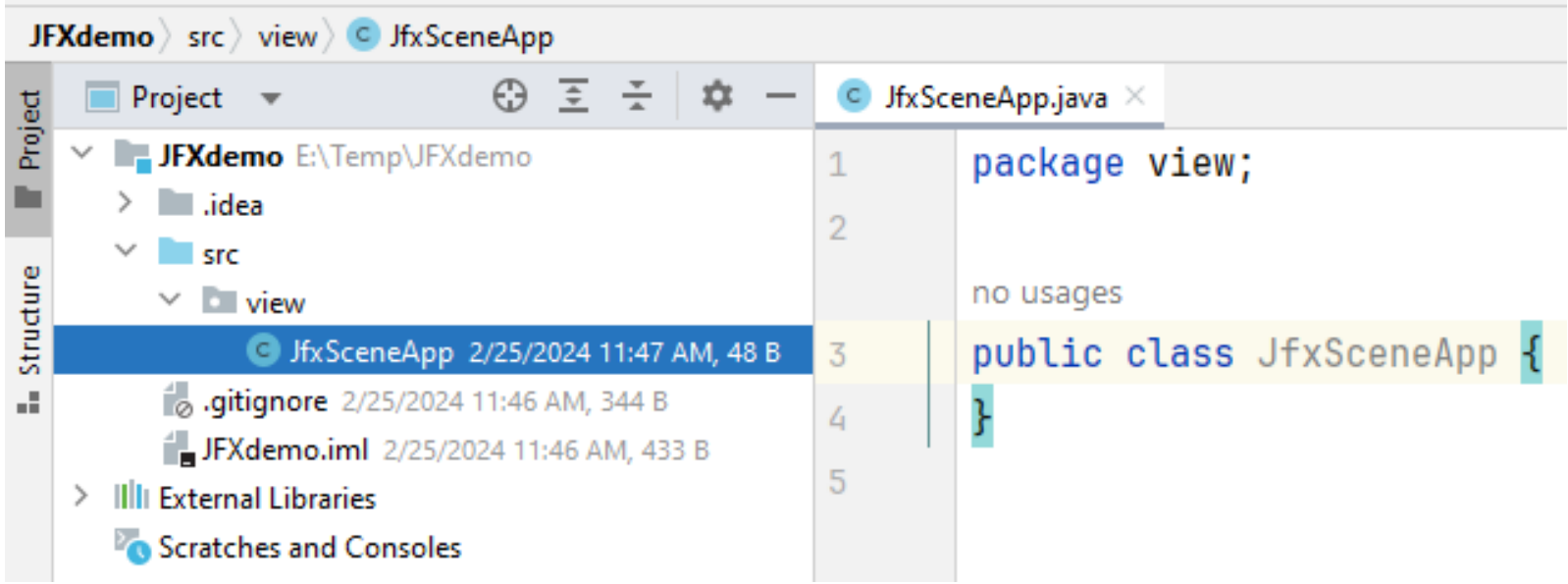


Non-Modular JavaFX project

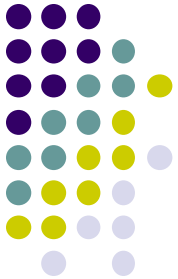
Create a JavaFX project in IntelliJ

3. Add a Java file to this package

A. The file of the application.
It is a Java file, say,
(JfxSceneApp.java)



Non-Modular JavaFX project



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

Select **Modules**

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

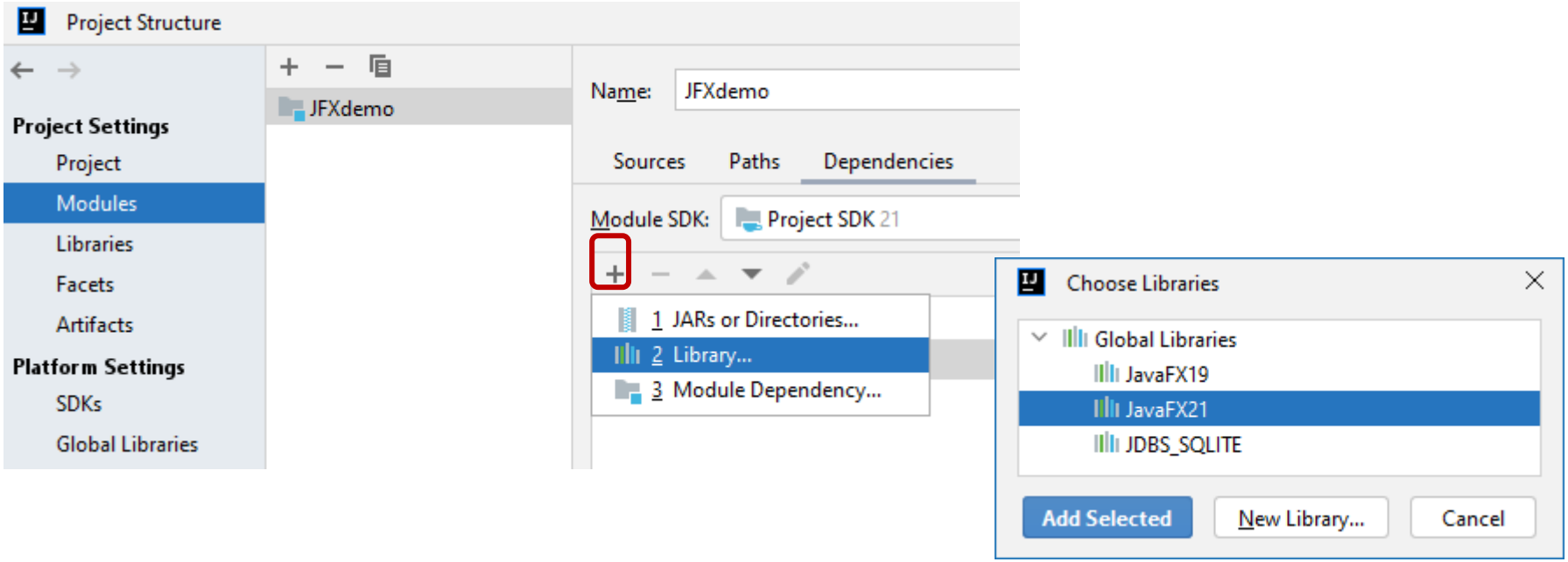
Among the Global Libraries select the previously create JavaFX library (click Add selected)

Click OK

Non-Modular JavaFX project

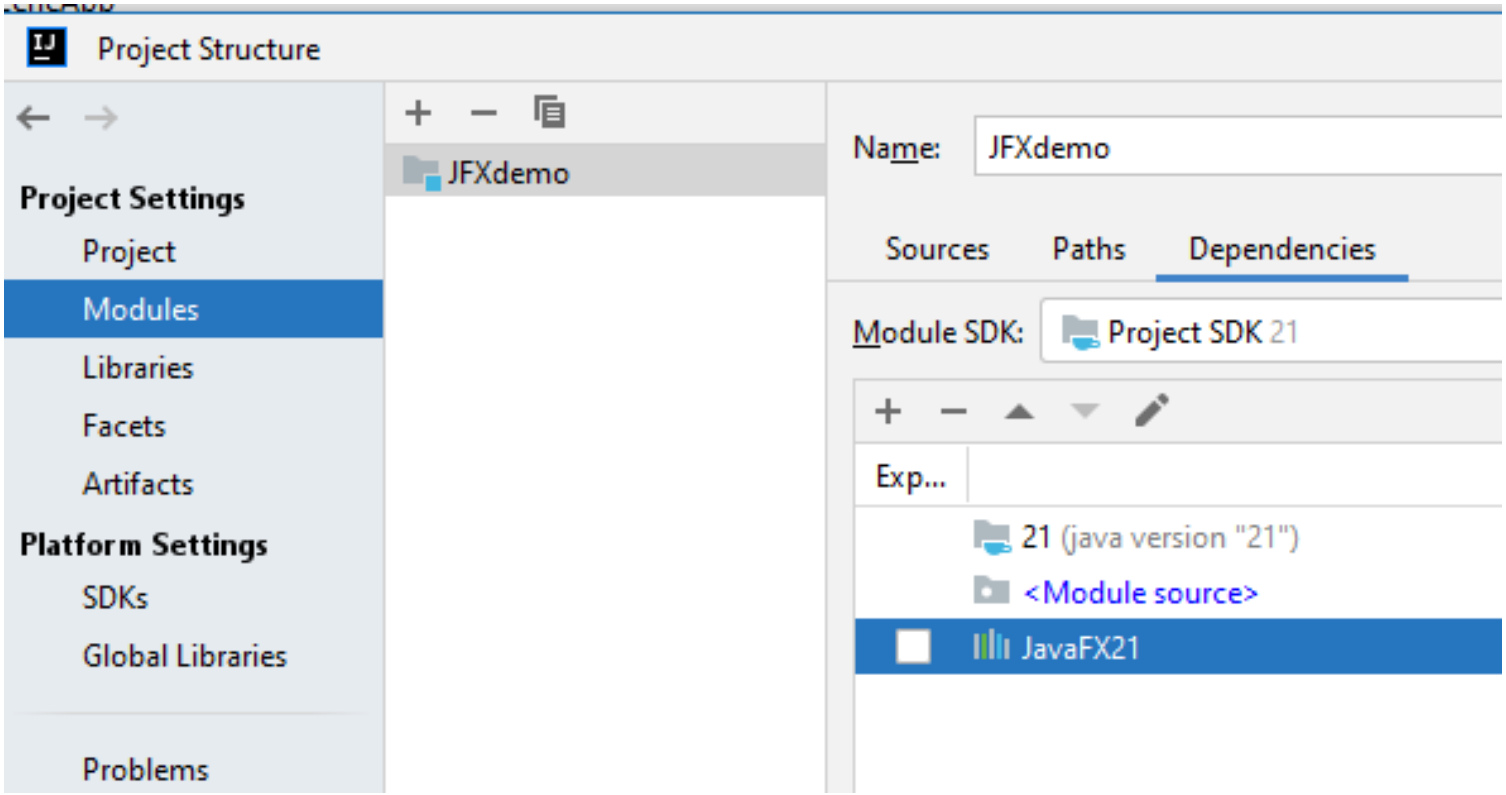
4. Add a Dependency to the JavaFX global library in Project structure (Ctrl-Alt-Shift-S)

Click the + in Dependencies and select Library

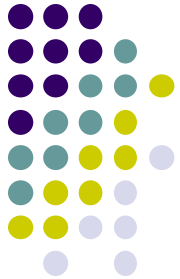


Non-Modular JavaFX project

The selected JFX library is added to Dependencies



Non-Modular JavaFX project



5. Open the file JfxSceneApp.java

- ✓ Copy-Paste the class name (JfxSceneApp)
- ✓ Delete everything below the package instruction
- ✓ Type the live template name fx-app followed by pressing Tab

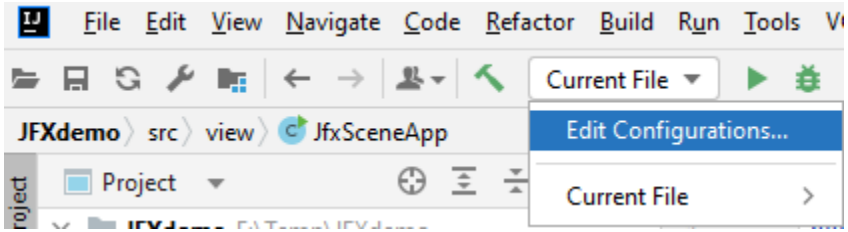
The contents of class JfxSceneApp are updated as shown in the following slide.

There remains to update the title of the window, for example, **JFX demo window**

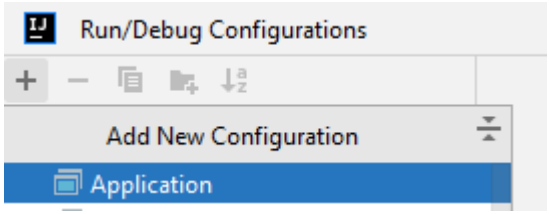
```java
package view;
import javafx.application.Application;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.stage.Stage;
// Abbreviaton key:       fx-drawing-main
// Template description: JavaFX App class for drawing
// Variables: CLASS_NAME must be assigned clipboard() expression
// 1. Create a Java class
// 2. Copy the class name in the Clipboard (^C)
// 3, Overwrite all  the class contents by running this Live template
// 4. Right-click the class name (should be the same as in the originally created class)
// 5. Select Show Content actions and execute Set package name to ...<your package name>

public class JfxSceneApp extends Application {

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

    @Override
    public void start(Stage stage) throws Exception {
        Group group = new Group();
        Scene scene = new Scene(group, 300, 250);
        // TODO  Type code for Java FX drawing objects


        // end TODO
        stage.setTitle("JFX demo window"); // Update Title as required
        stage.sizeToScene();
        stage.resizableProperty().setValue(Boolean.FALSE);
        stage.setScene(scene);
        stage.show();
    }
}
```

Run the JavaFX project

Update the Application template for this application

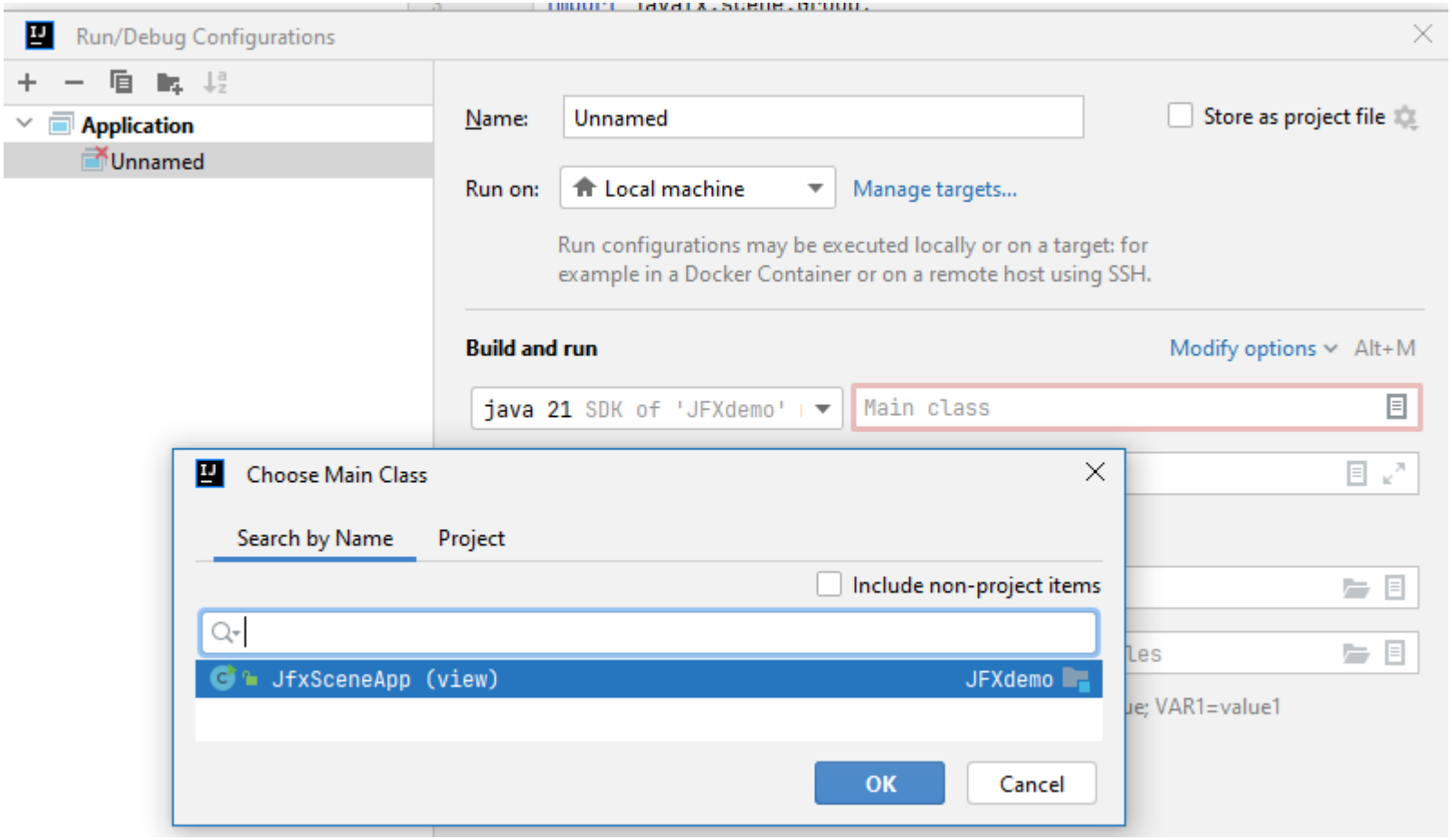


Click the + and add an Application template



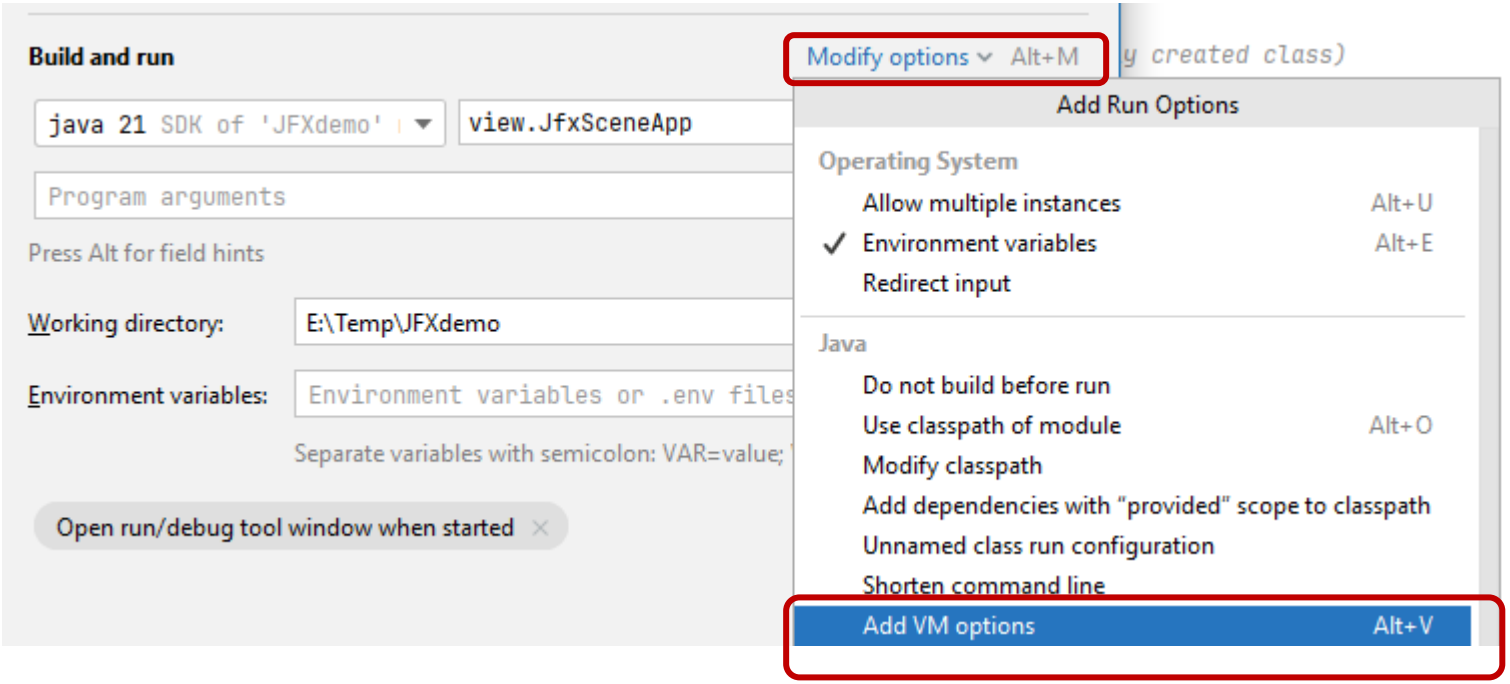
Update the Main class by pressing the “folder” icon as shown in the following slide

Run the JavaFX project



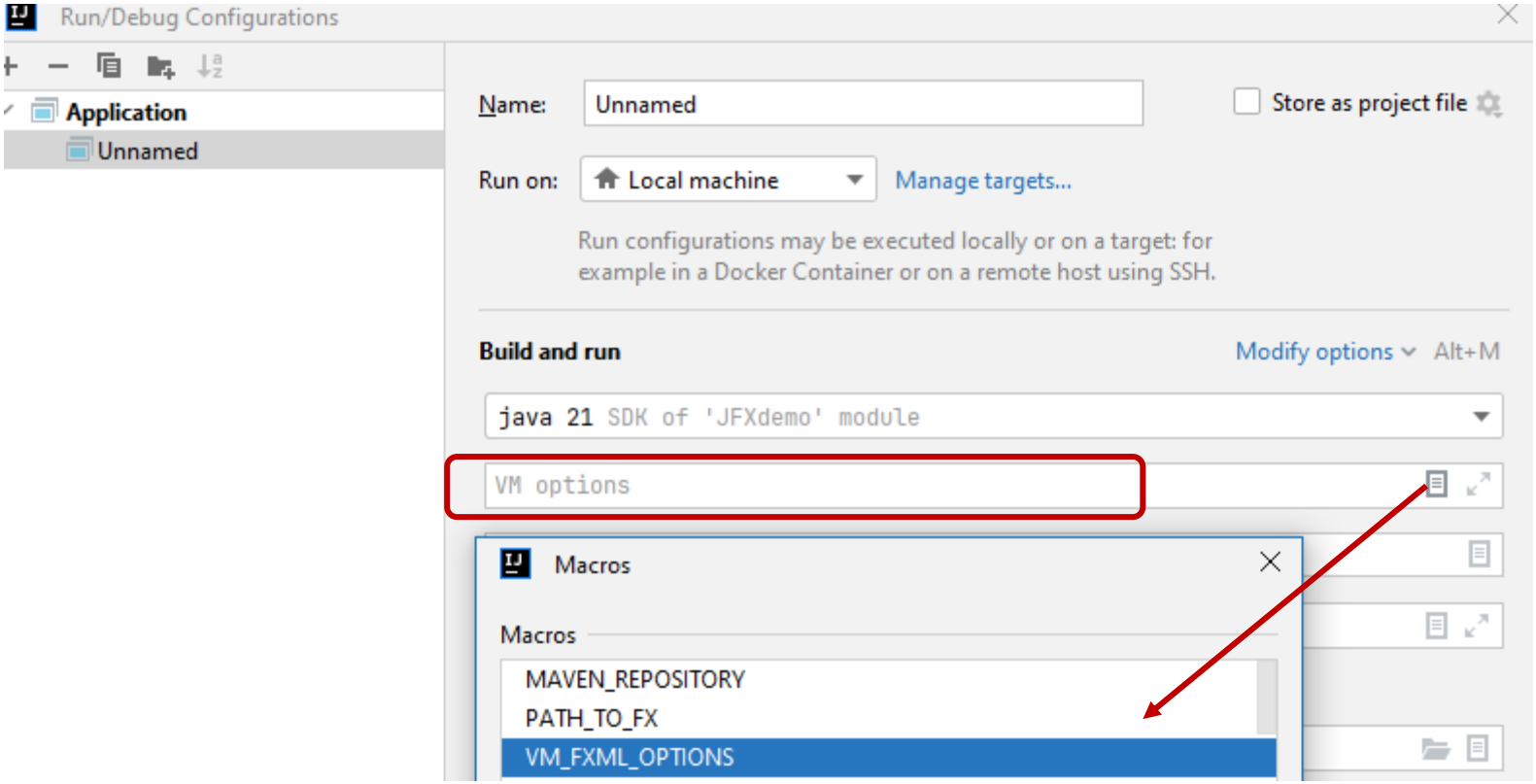
Run the JavaFX project

4. Add VM options to the Run Application template to resolve the problem



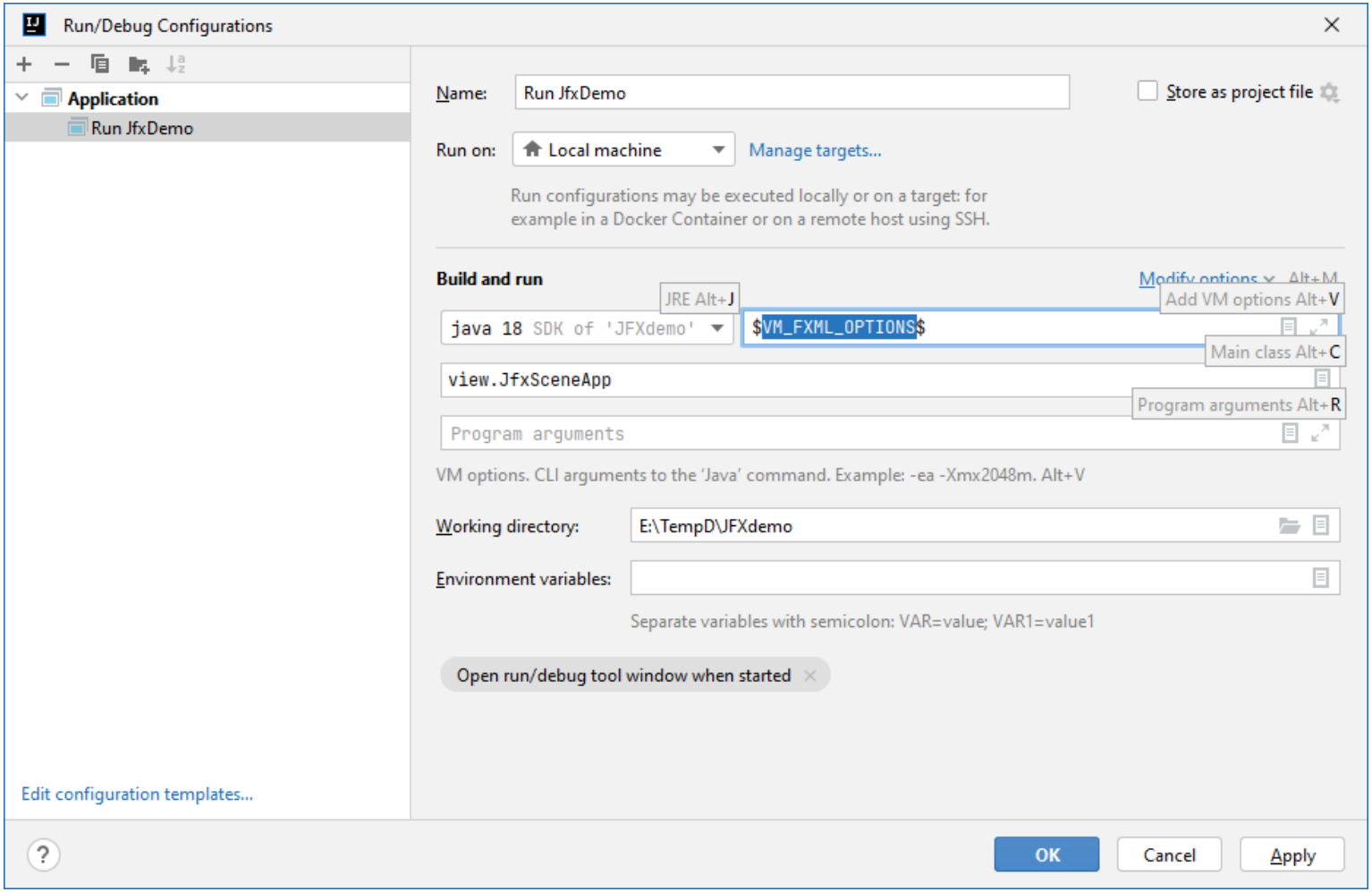
Run the JavaFX project

… where for VM options select the Path variable VM_FXML_OPTIONS (previously created)



Non-Modular JavaFX project

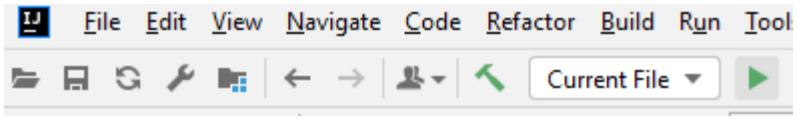
The Application template should be as follows



Run the JavaFX project

To run the application, use one of the following:

Click the green triangle



or

Right-click the Java file with the main() method i.e. JfxSceneApp

Remaining tasks:

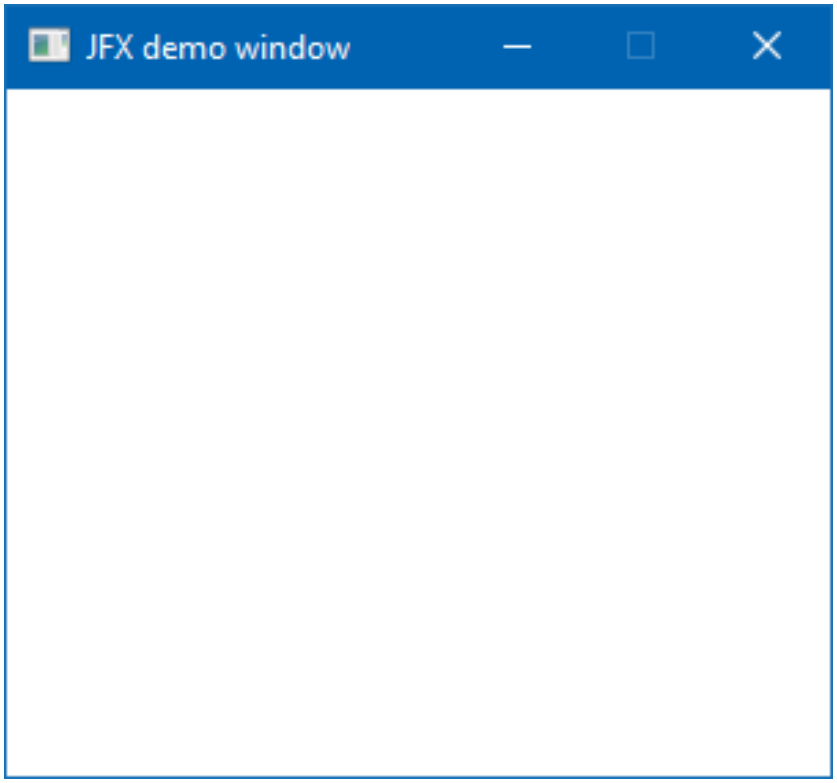
Create nodes and JavaFX content below the comments

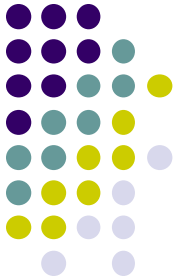
```
// TODO  Type code for Java FX drawing objects
```

Run the JavaFX project

In case the VM options are not setup you get

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





Happy
Object Oriented Programming
with
JavaFX