The University of Nottingham

UNITED KINGDOM · CHINA · MALAYSIA

Lecture 03B

Maintainable GUI Development (1/2)

Introduction to GUI Programming in Java

COMP2013: Developing Maintainable Software
Week 4 – 1:00pm Thursday – 19 October 2023

valid for 65 minutes from 12:55pm
generated 2023-10-10 03:14

Horia A. Maior and Marjahan Begum

COMP2013

Developing Maintainable Software

COMP2013: Developing Maintainable Software
Week 4 – 1:00pm Thursday – 19 October 2023

Lecture 03B

Maintainable GUI Development (1/2)

GUI Development in Practice

Horia A. Maior and Marjahan Begum

University of Nottingham
UK | CHINA | MALAYSIA

Topics for this Week

- Lecture 03A
  - Introduction to GUI programming in Java
    - GUIs; Swing vs JavaFX; JavaFX foundations

- Lab 03
  - Practice JavaFX and Swing GUI development (phone app)

- Lecture 03B
  - GUI development in practice
    - Lab reflection;
    - MVC Pattern
    - FXML;
    - multiple windows

COMP2013: Developing Maintainable Software
Week 4 – 1:00pm Thursday – 19 October 2023

COMP2013: Developing Maintainable Software
Week 4 – 1:00pm Thursday – 19 October 2023

valid for 65 minutes from 12:55pm
generated 2023-10-10 03:14

lab reflection

Lab Reflection

LAB 3: SIMPLE GUI DEVELOPMENT WITH JAVAFX and SWING

Aims:

- Ensure that JavaFX and SWING work on your computer
- Gain some GUI development experience by re-implementing Lecture 03A's examples
- Gain some GUI development experience by building a simple phone app in SWING and JAVAFX.

Please do the exercise(s) that you feel are appropriate for you.

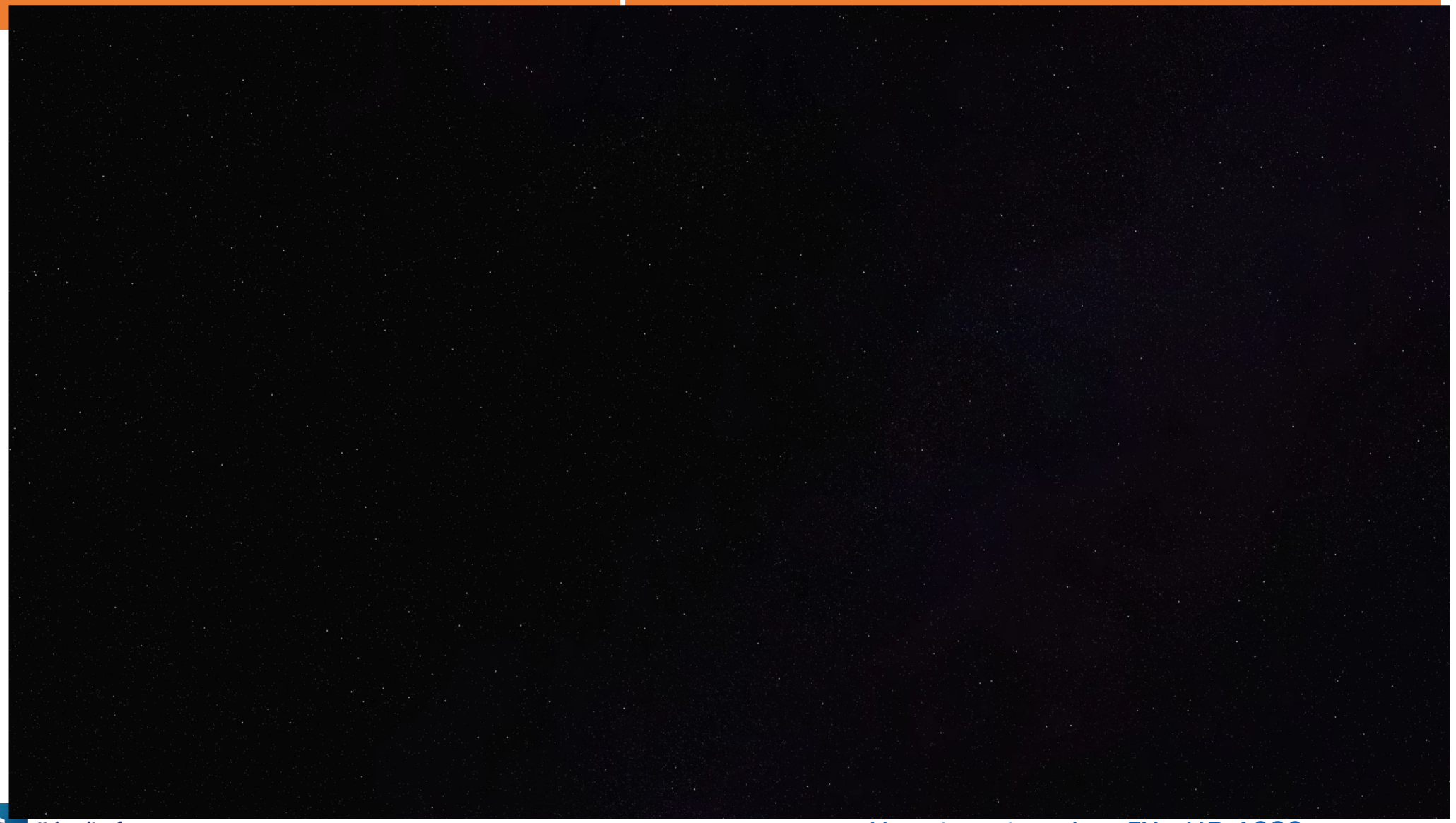
BEGINNER LEVEL: CREATING A JAVAFX PROJECT

To ensure that JavaFX works on your computer, set up some test projects in IntelliJ. Start by doing it manually (as I did in the lecture, using a plain Java project) and then by using the JavaFX project option, which will configure some build script to set up the project and dependencies automatically.

Part 1: RE-IMPLEMENTING and MODIFYING THE LECTURE EXAMPLE in JAVAFX and SWING

Re-implement the HelloJavaFXWorld example presented in Lecture 03A. Then do some modifications to it (e.g. adding components or using different layout panes) and check out the consequences.

Lab Reflection JavaFX Setup



Part 2: TELEPHONE APPLICATION in JavaFX

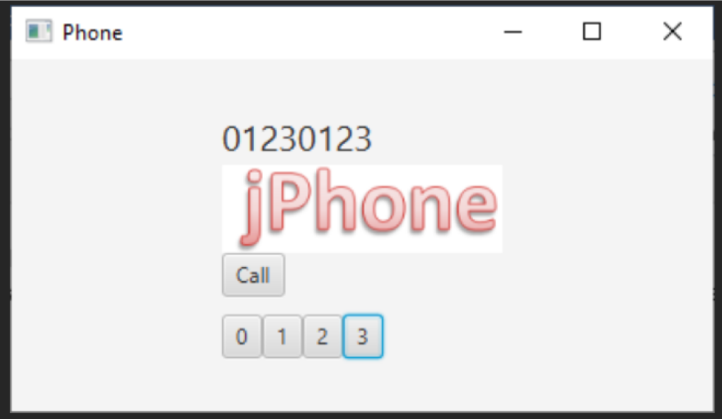
Your main task for the lab is to build a simple telephone application. You can find the spec below.
Specification: You need to build a JavaFX GUI-based app with the following features:

1. Number buttons (0 to 9) to enter the phone number {Tip: A GridPane might be useful here}
2. A display to view the current number
3. A Call button. This will obviously not actually make a call, but should pass the number to a ***makeCall()*** method, which implements an ***Phone*** interface. This class should represent a dummy telephone device, and the method should wait for a random duration (in seconds) and then return the duration of the "call".

```
//Things which can make or fake telephone calls implement this.
public interface Phone {
    public int makeCall(String number);
}
```

   - Note how, by implementing an interface, we can switch out a "dummy" or "real" phone application easily.
   - To wait for some time you could do something like this (note though, that this example snippet does not wait for a random time, and that this might not be the best way to deal with this requirement):

```
duration=3000; // could make this random
try {
    Thread.sleep(duration); // note how this hangs the interface! how can we fix that?
} catch (InterruptedException e) {
    e.printStackTrace();
} // faking phone call time.
```



5. Maybe think on how you could make this look pretty 😊

Things to bear in mind:

- You need to handle telephone numbers starting with a 0, so think about your data type!

Part 3: REWRITE THE TELEPHONE APPLICATION in SWING

Now that you have managed to finalise your desing in JavaFX, we want you to build the same program but using Swing. This is related to legacy code; a lot of projects out there are build in Swing, so learning and understanding will be highly beneficial.

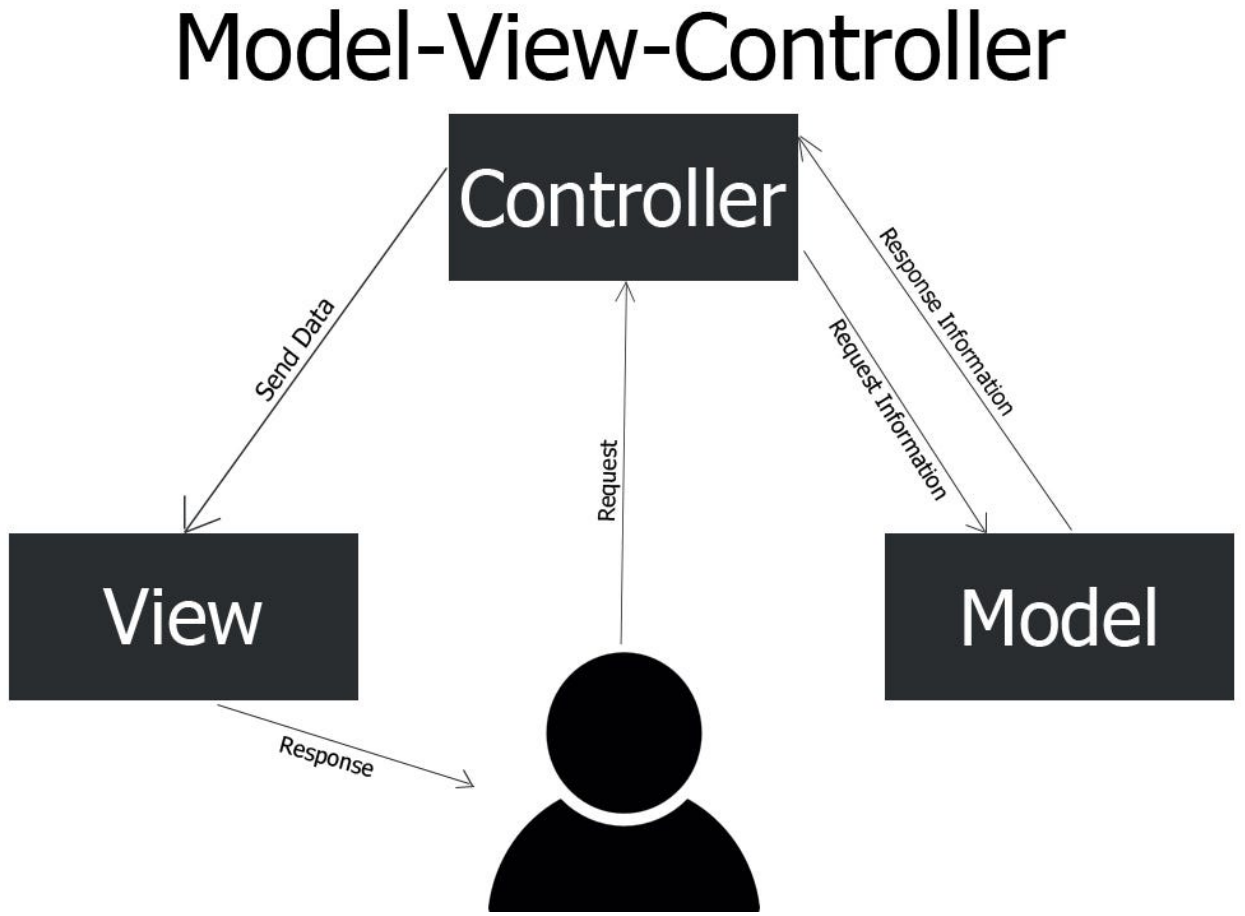
- Start from the Hello World Example in the Monday Lecture, and build on from there. This will require some individual research into how to build different parts.
- For those of you not so familiar with Swing, here are some excellent tutorials to follow in Java Swing

mvc pattern and FXML

Model-View-Controller

MVC Explained

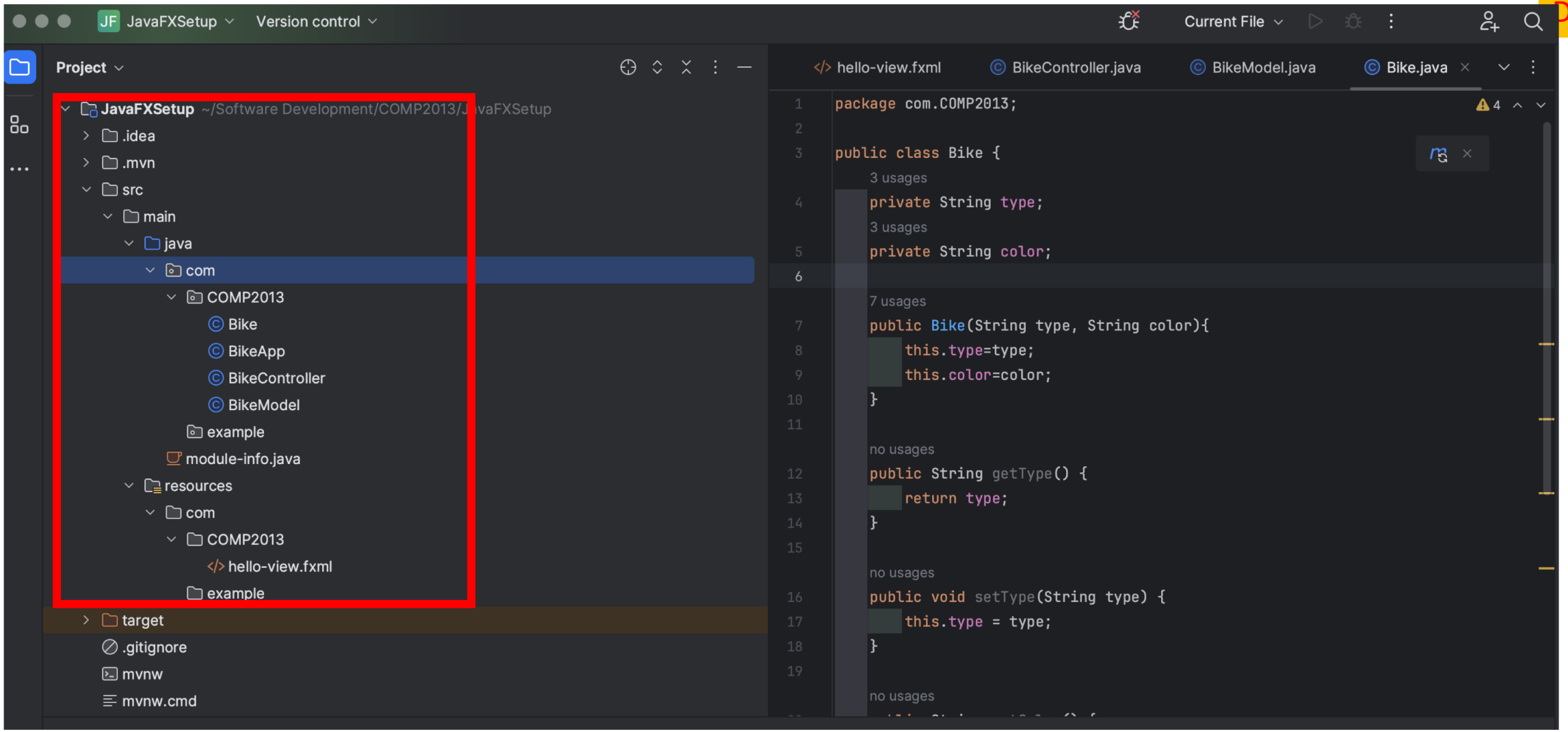
- View tells controller what happened
- Controllers inform the model what to do
- Once the job is finished the model broadcasts notifications; it is not aware of any specific view/controller
- Views catches relevant notifications and updates



https://medium.com/@joespinelli_6190/mvc-model-view-controller-ef878e2fd6f5

MVC Applied in Java

DEMO



See https://stackoverflow.com/questions/36868391/using-javafx-controller-without-fxml for more explanations

MVC Applied in Java

</>

DEMO

BikeApp.java

Runnable class

```
package com.COMP2013;
import ...

public class BikeApp extends Application {
    @Override
    public void start(Stage stage) throws IOException {
        FXMLLoader fxmlLoader = new FXMLLoader(BikeApp.class.getResource( name: "hello-view.fxml"));
        Scene scene = new Scene(fxmlLoader.load(), v: 500, v1: 500);
        stage.setTitle("AddBikesApp!");
        stage.setScene(scene);
        stage.show();
    }

    public static void main(String[] args) {

        launch();
    }
}
```

UK | CHINA | MALAYSIA

MVC A

BikeApp.java | BikeController.java | hello-view.fxml

```
<?xml version="1.0" encoding="UTF-8"?>
<?import javafx.geometry.Insets?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.ListView?>
<?import javafx.scene.control.TextField?>

<VBox alignment="CENTER" spacing="20.0" xmlns:fx="http://javafx.com/fxml"
      fx:controller="com.COMP2013.BikeController">
    <padding>
        <Insets bottom="20.0" left="20.0" right="20.0" top="20.0"/>
    </padding>
    <Label text="Check out the list of my bike objects below:"/>
    <ListView fx:id="bikeListView" />
    <Label text="Bike Name"></Label>
    <TextField fx:id="bikeNameField"/>
    <Label text="Bike Color"></Label>
    <TextField fx:id="bikeColorField"/>
    <Button text="Add Bike" onAction="#onAddBikeBtnClick" />
</VBox>
```

</>

DEMO

University of Nottingham
UK | CHINA | MALAYSIA

MVC A

BikeApp.java | BikeController.java | hello-view.fxml

```
<?xml version="1.0" encoding="UTF-8"?>
<?import javafx.geometry.Insets?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.ListView?>
<?import javafx.scene.control.TextField?>

<VBox alignment="CENTER" spacing="20.0" xmlns:fx="http://javafx.com/fxml"
      fx:controller="com.COMP2013.BikeController">
    <padding>
        <Insets bottom="20.0" left="20.0" right="20.0" top="20.0"/>
    </padding>
    <Label text="Check out the list of my bike objects below:"/>
    <ListView fx:id="bikeListView" />
    <Label text="Bike Name"></Label>
    <TextField fx:id="bikeNameField"/>
    <Label text="Bike Color"></Label>
    <TextField fx:id="bikeColorField"/>
    <Button text="Add Bike" onAction="#onAddBikeBtnClick" />
</VBox>
```

</>

DEMO

University of Nottingham
UK | CHINA | MALAYSIA

MVC Applied in Java

BikeApp.java | BikeController.java | hello-view.fxml

```
1   package com.COMP2013;
2   import ...
8   public class BikeController {
9       @FXML
10      private ListView<Bike> bikeListView;
11      @FXML
12      private TextField bikeNameField;
13      @FXML
14      private TextField bikeColorField;
        3 usages
15      private BikeModel model;
16
17      public void initialize() {
18          // Set up the ListView to display the list of bikes
19          model = new BikeModel();
20          ObservableList<Bike> bikes = model.getBikes();
21          bikeListView.setItems(bikes);
22      }
23
24      @FXML
25      private void onAddBikeBtnClick() {
26          String name = bikeNameField.getText();
27          String color = bikeColorField.getText();
28          Bike newBike = new Bike(name, color);
29          model.addBike(newBike);
30      }
31  }
```

</>

DEMO

MVC A

BikeApp.java | BikeController.java | BikeModel.java | hello-view.fxml

```java
package com.COMP2013;
import ...

public class BikeModel {
    5 usages
    private ObservableList<Bike> bikes = FXCollections.observableArrayList();
    public BikeModel() {
        // Initialize the model with some sample data
        bikes.add(new Bike( type: "Mountain Bike",  color: "Blue"));
        bikes.add(new Bike( type: "Road Bike",  color: "Red"));
        bikes.add(new Bike( type: "City Bike",  color: "Green"));
    }
    1 usage
    public ObservableList<Bike> getBikes() { return bikes; }

    1 usage
    public void addBike(Bike bike) { bikes.add(bike); }

}
```

BikeApp.java | BikeController.java | BikeModel.java | Bike.java

```java
package com.COMP2013;

public class Bike {
    3 usages
    private String type;
    3 usages
    private String color;

    7 usages
    public Bike(String type, String color){
        this.type=type;
        this.color=color;
    }

    no usages
    public String getType() { return type; }
    no usages
    public void setType(String type) { this.type = type; }
    no usages
    public String getColor() { return color; }
    no usages
    public void setColor(String color) { this.color = color; }
}
```

FXML

Introduction

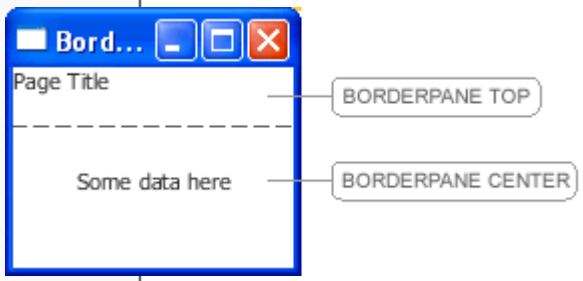
FXML

- So far we have seen how we can build simple GUIs in code
- BUT if the GUI is going to get complicated JavaFX supports describing the layout using FXML (FX Markup Language)
- Supports the idea of separating 'Design' and 'Functionality'
  - Languages like C# in Visual Studio also support this separation of concerns

JavaFX vs FXML

Example 1-1 Java Code for a User Interface

```
BorderPane border = new BorderPane();
Label toppanetext = new Label("Page Title");
border.setTop(toppanetext);
Label centerpanetext = new Label ("Some data here");
border.setCenter(centerpanetext);
```

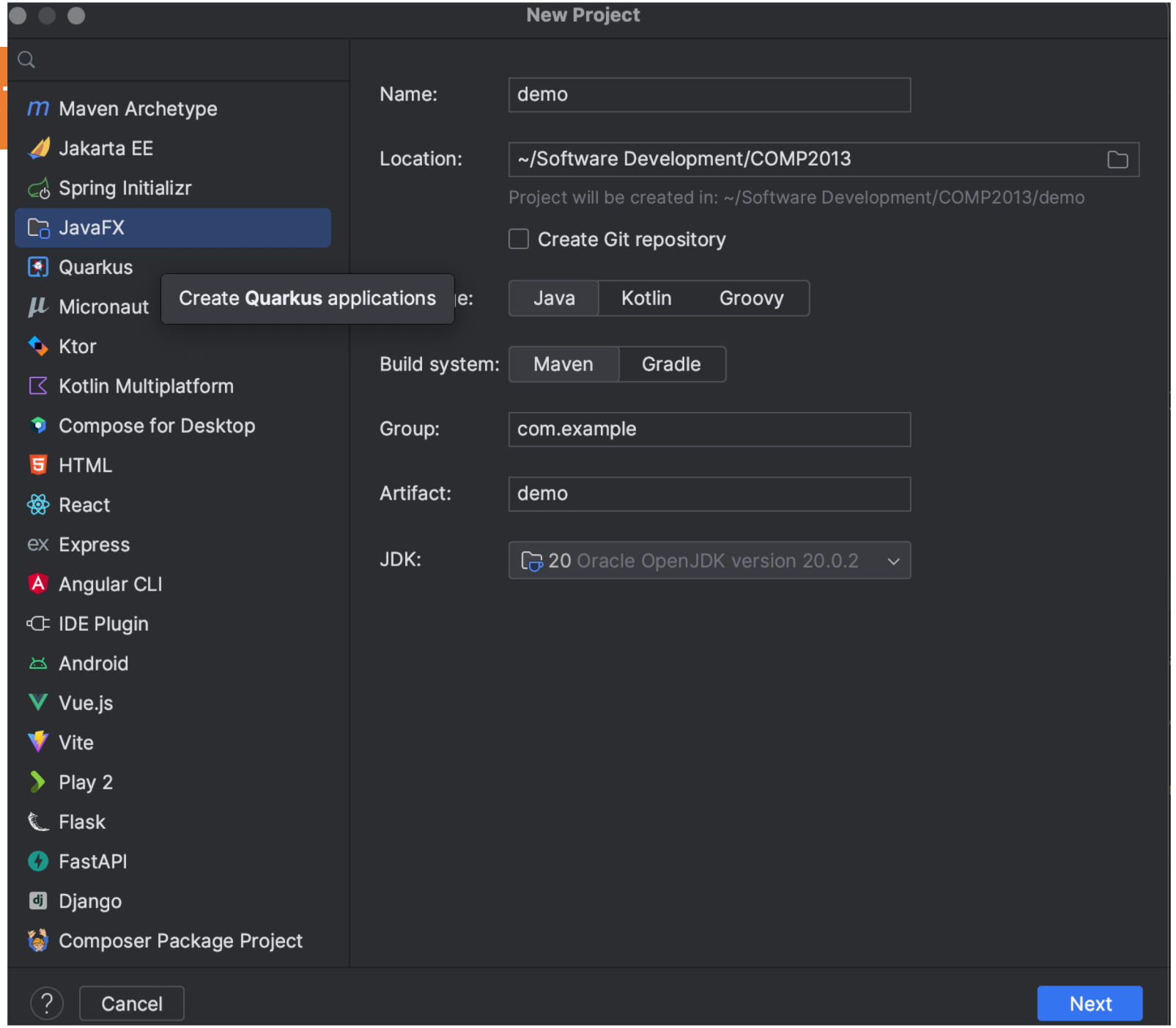


Example 1-2 FXML Markup for a User Interface

```
<BorderPane>
    <top>
        <Label text="Page Title"/>
    </top>
    <center>
        <Label text="Some data here"/>
    </center>
</BorderPane>
```

Simpler, less code, and a more intuitive organisation

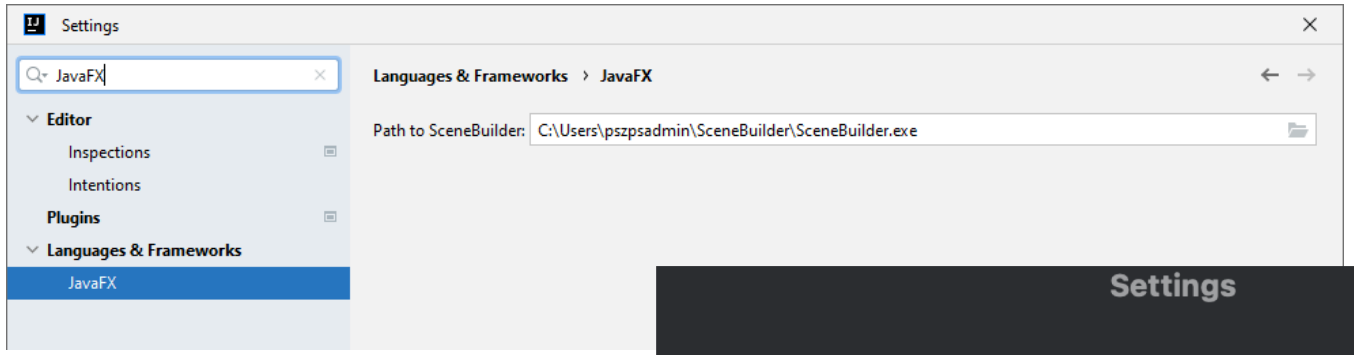
Creating FXML Projec

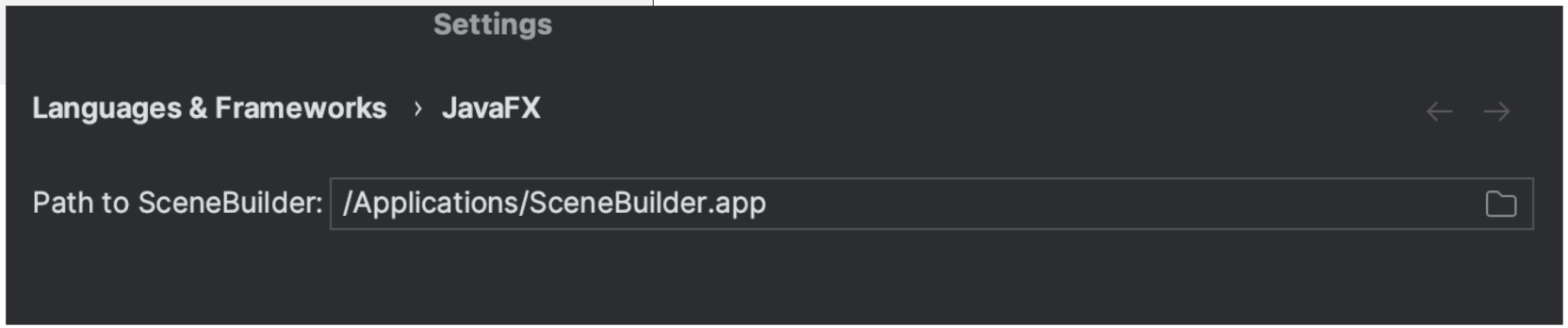


University of Nottingham
UK | CHINA | MALAYSIA

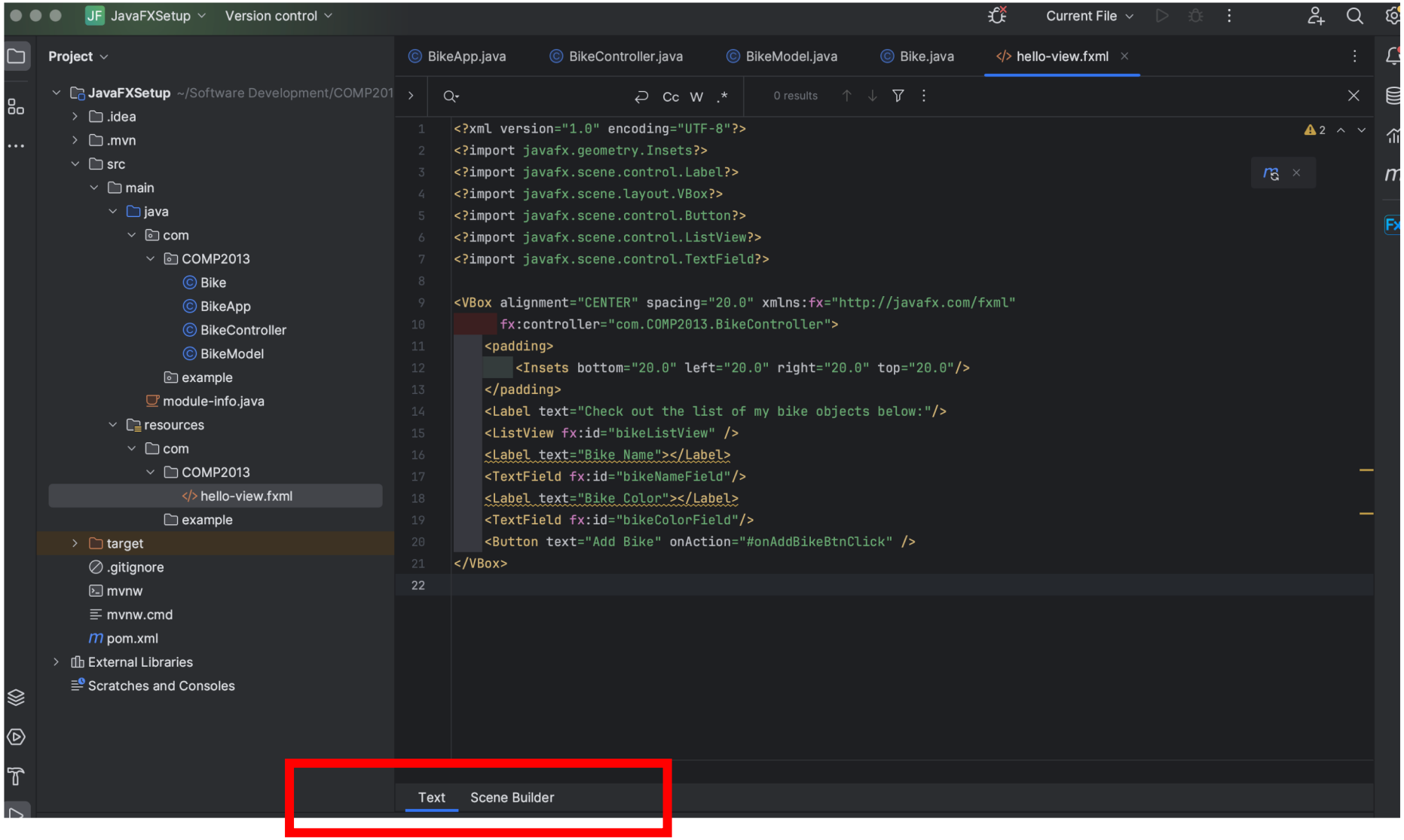
Preparing FXML Editor

- Scene Builder:
  - A design tool for created FXML files graphically
  - Download from https://gluonhq.com/products/scene-builder/ and install
  - You need to tell the IDE where to find the Gluon Scene Builder executable
  - https://www.jetbrains.com/help/idea/opening-fxml-files-in-javafx-scene-builder.html#open-in-scene-builder

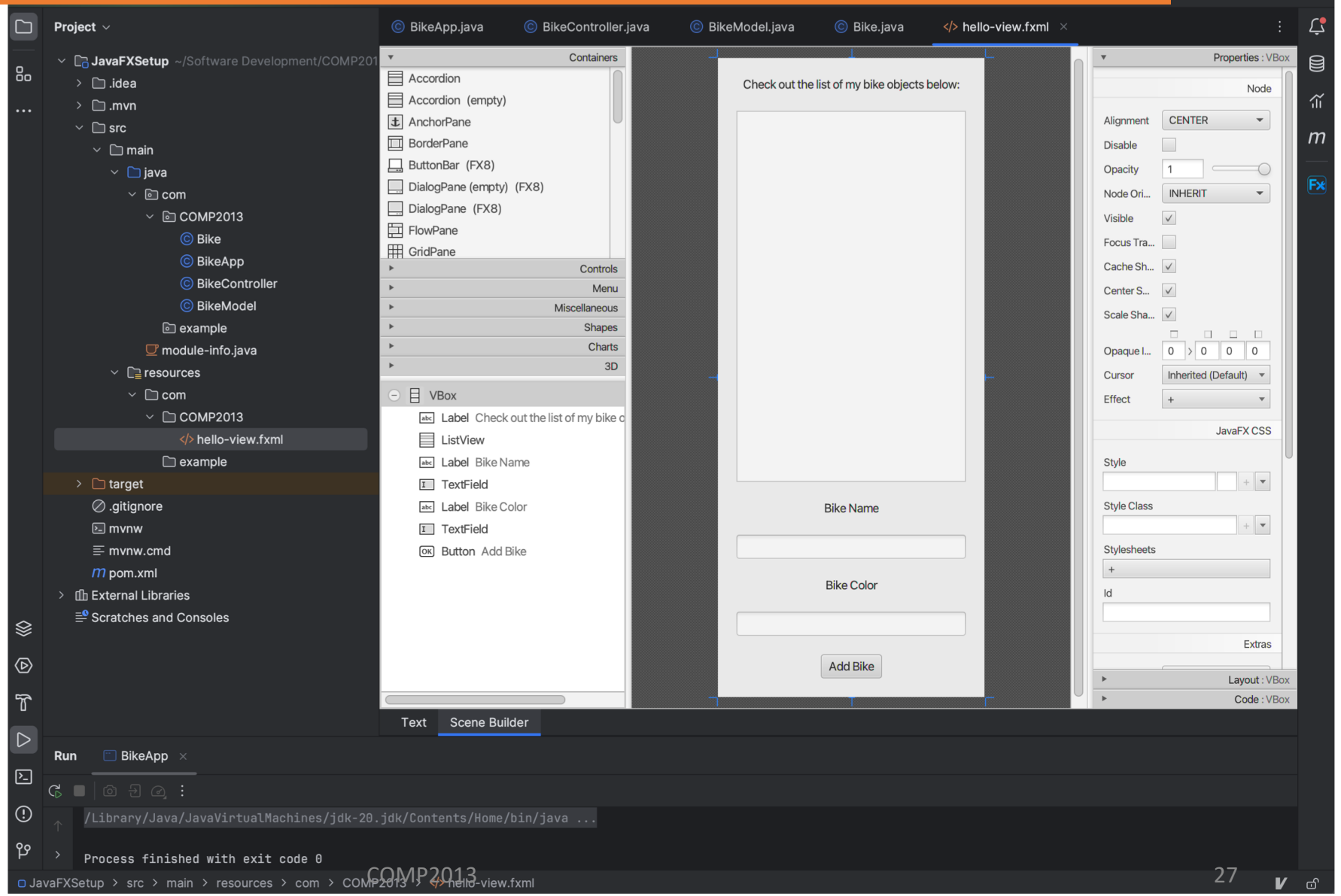


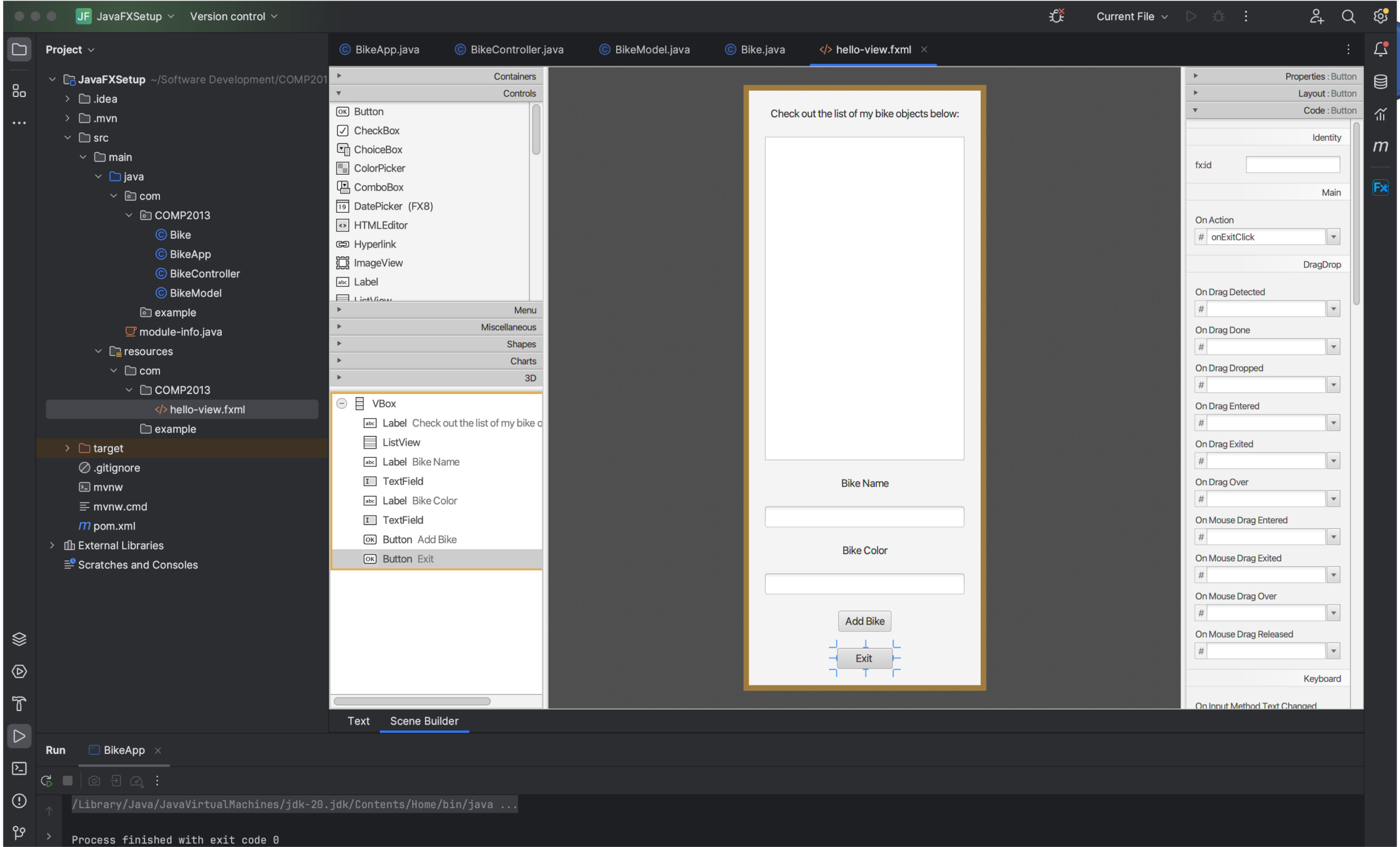


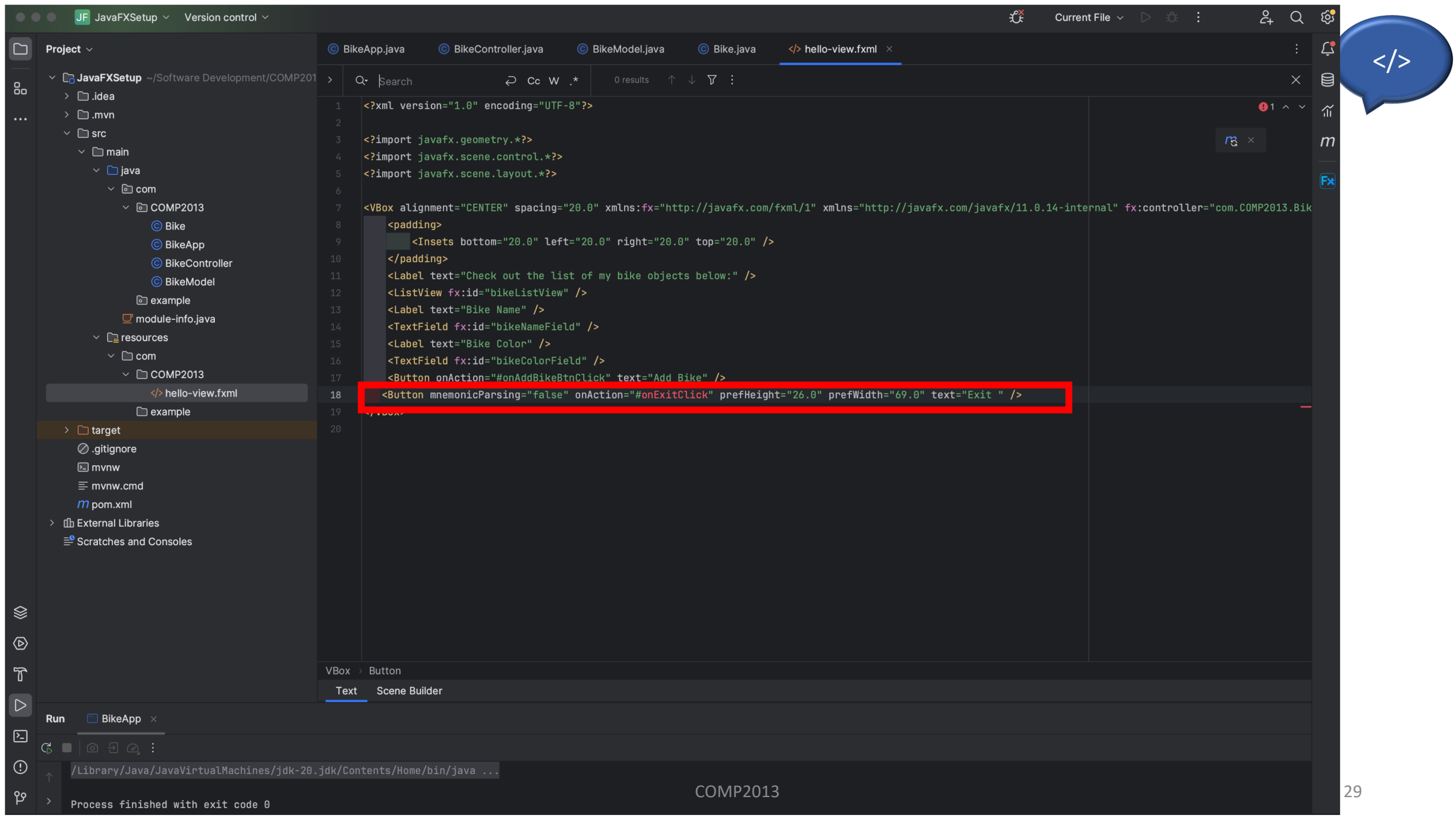
Open FXML Editor



Open FXML Editor




JavaFXSetup
Version control
Current File
Project
JavaFXSetup ~/Software Development/COMP201
.idea
.mvn
src
main
java
com
COMP2013
Bike
BikeApp
BikeController
BikeModel
example
module-info.java
resources
com
COMP2013
hello-view.fxml
example
target
.gitignore
mvnw
mvnw.cmd
pom.xml
External Libraries
Scratches and Consoles
BikeApp.java
BikeController.java
BikeModel.java
Bike.java
hello-view.fxml
Containers
Controls
Button
CheckBox
ChoiceBox
ColorPicker
ComboBox
DatePicker (FX8)
HTMLEditor
Hyperlink
ImageView
Label
Menu
Miscellaneous
Shapes
Charts
3D
VBox
Label Check out the list of my bike o
ListView
Label Bike Name
TextField
Label Bike Color
TextField
Button Add Bike
Button Exit
Check out the list of my bike objects below:
Bike Name
Bike Color
Add Bike
Exit
Properties : Button
Layout : Button
Code : Button
Identity
fx:id
Main
On Action
onExitClick
DragDrop
On Drag Detected
On Drag Done
On Drag Dropped
On Drag Entered
On Drag Exited
On Drag Over
On Mouse Drag Entered
On Mouse Drag Exited
On Mouse Drag Over
On Mouse Drag Released
Keyboard
Text
Scene Builder
Run
BikeApp
/Library/Java/JavaVirtualMachines/jdk-20.jdk/Contents/Home/bin/java ...
Process finished with exit code 0


JavaFXSetup
Version control
Current File
Project
JavaFXSetup ~/Software Development/COMP201
.idea
.mvn
src
main
java
com
COMP2013
Bike
BikeApp
BikeController
BikeModel
example
module-info.java
resources
com
COMP2013
hello-view.fxml
example
target
.gitignore
mvnw
mvnw.cmd
pom.xml
External Libraries
Scratches and Consoles
BikeApp.java
BikeController.java
BikeModel.java
Bike.java
hello-view.fxml
Search
0 results
1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <?import javafx.geometry.*?>
4 <?import javafx.scene.control.*?>
5 <?import javafx.scene.layout.*?>
6
7 <VBox alignment="CENTER" spacing="20.0" xmlns:fx="http://javafx.com/fxml/1" xmlns="http://javafx.com/javafx/11.0.14-internal" fx:controller="com.COMP2013.Bik
8 <padding>
9 <Insets bottom="20.0" left="20.0" right="20.0" top="20.0" />
10 </padding>
11 <Label text="Check out the list of my bike objects below:" />
12 <ListView fx:id="bikeListView" />
13 <Label text="Bike Name" />
14 <TextField fx:id="bikeNameField" />
15 <Label text="Bike Color" />
16 <TextField fx:id="bikeColorField" />
17 <Button onAction="#onAddBikeBtnClick" text="Add Bike" />
18 <Button mnemonicParsing="false" onAction="#onExitClick" prefHeight="26.0" prefWidth="69.0" text="Exit " />
19
20
VBox › Button
Text
Scene Builder
Run
BikeApp
/Library/Java/JavaVirtualMachines/jdk-20.jdk/Contents/Home/bin/java ...
Process finished with exit code 0

Why use FXML?

- Why use FXML? What do people think are the benefits?
  - UI designers might not be programmers
  - The designers can use external software (such as Scene Builder) to design the look of the UI whilst the programmers can build the functionality
    - FXML glues the two aspects together
  - Building GUIs visually rather than programmatically makes intuitive sense
  - Event handling is simplified
  - We can test/fix application logic without touching the GUI design or vice versa

JavaFX advanced topics

Multiple Windows

Multiple Windows

- Simple approach

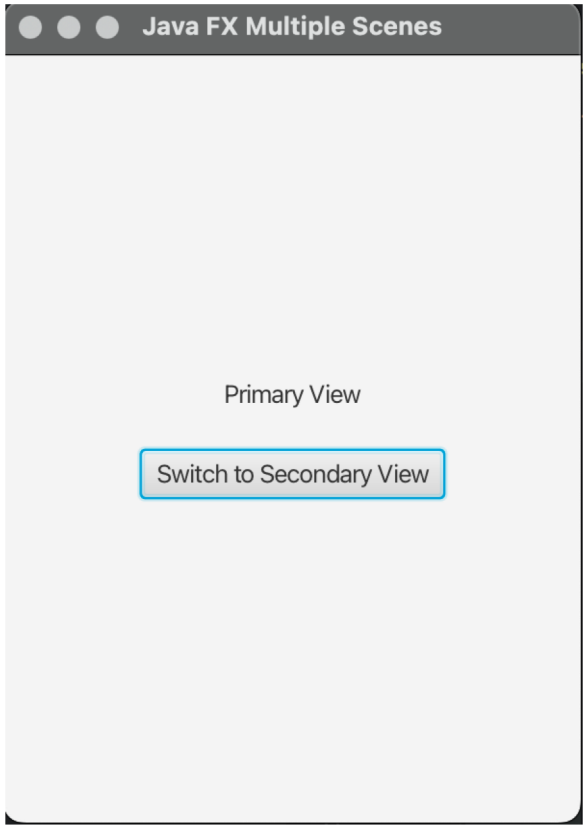
pom.xml (JavaFXMulti-Scenes) | hello-view.fxml | HelloController.java | HelloApplication.java

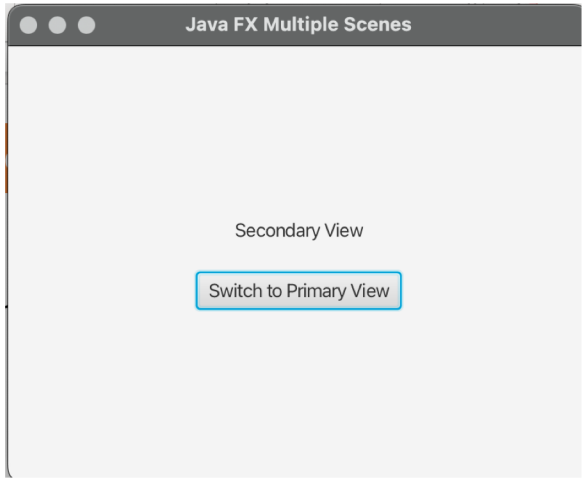
```
public class HelloApplication extends Application {
    3 usages
    private Scene scene1, scene2;

    @Override
    public void start(Stage primaryStage) {
        primaryStage.setTitle("Java FX Multiple Scenes");
        Label label1= new Label( s: "Primary View");
        Button button1= new Button( s: "Switch to Secondary View");
        button1.setOnAction(new EventHandler<ActionEvent>() {
            @Override
            public void handle(ActionEvent actionEvent) {
                primaryStage.setScene(scene2);
            }
        });
        VBox layout1 = new VBox( v: 20);
        layout1.setAlignment(Pos.CENTER);
        layout1.getChildren().addAll(label1, button1);
        scene1= new Scene(layout1,  v: 300,  v1: 400);
        Label label2= new Label( s: "Secondary View");
        Button button2= new Button( s: "Switch to Primary View");
        button2.setOnAction(new EventHandler<ActionEvent>() {
            @Override
            public void handle(ActionEvent actionEvent) {
                primaryStage.setScene(scene1);
            }
        });
        VBox layout2= new VBox( v: 20);
        layout2.setAlignment(Pos.CENTER);
        layout2.getChildren().addAll(label2, button2);
        scene2= new Scene(layout2, v: 400, v1: 300);
        primaryStage.setScene(scene1);
        primaryStage.show();
    }

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

</> DEMO

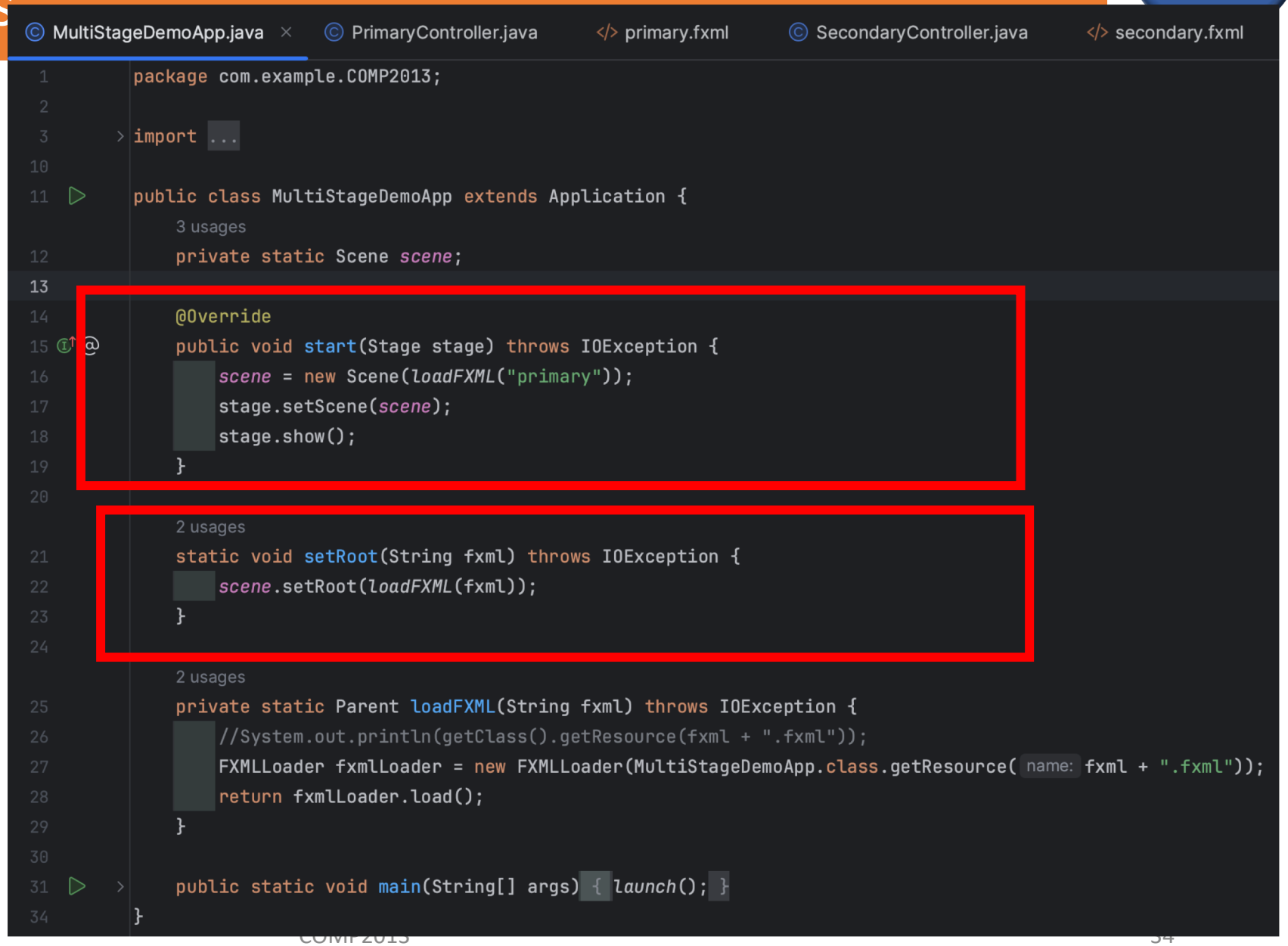
Multiple Windows

- The result





Multiple Windows



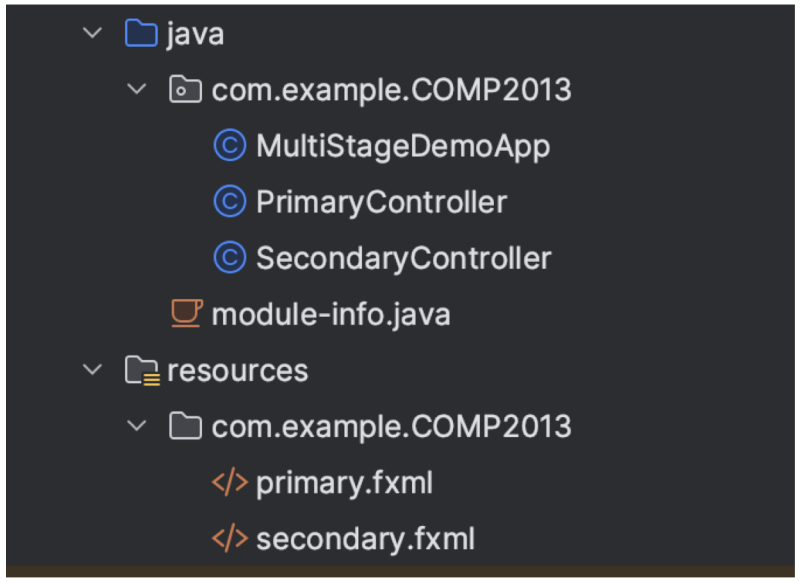

```java
package com.example.COMP2013;

import javafx.fxml.FXML;

import java.io.IOException;

public class PrimaryController {

    @FXML
    private void switchToSecondary() throws IOException {
        MultiStageDemoApp.setRoot("secondary");
    }
}
```

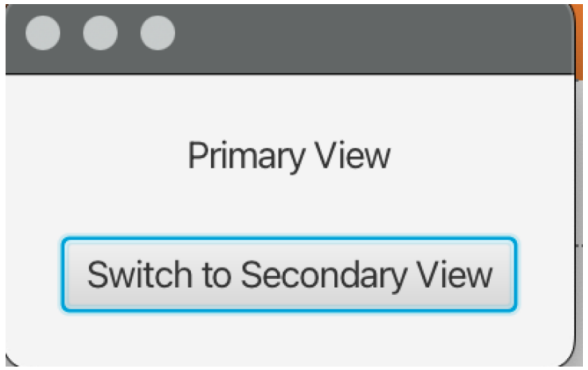
~/Software Development/COMP2013/M

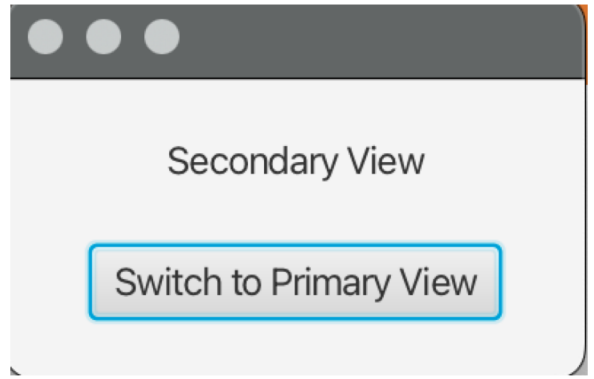
```java
package com.example.COMP2013;

import ...

public class SecondaryController {
    no usages
    @FXML
    private Label welcomeText;

    @FXML
    private void switchToPrimary() throws IOException {
        MultiStageDemoApp.setRoot("primary");
    }
}
```

Multiple Windows

- The result







https://stackoverflow.com/questions/10134856/javafx-2-0-how-to-application-getparameters-in-a-controller-java-file/10136403#10136403

Some final remarks …