CS209 – LAB5

Key Content

- Install JavaFX and Scene Builder.
- Creating and starting a JavaFX project.
- Using Scene Builder to design the user interface.
- Make an application following a tutorial.

1. Install JavaFx and Scene Builder

This tutorial is based on the JDK8 and it is fine to use this (or install multiple versions of Java and switch to this).

Please note that currently - from Java 11 - JavaFX has removed from the Java SDK again so now JavaFX has been detached into its own open source project after previously being added in Java 8. This means that to download JavaFX from Java 11 / JavaFX 11, you can obtain it by downloading from here:
http://openjfx.io

1.1 Eclipse

This section will introduce how to develop and run JavaFX program on Eclipse IDE.

1.1.1 e(fx)eclipse

The Eclipse e(fx)clipse project provides tooling and runtime components that help developers create JavaFX applications. e(fx)eclipse is a set of tools and necessary library. It is going to help you execute JavaFX program, so you need to make sure that you have already download it and installed successfully on your eclipse. If you haven't installed e(fx)eclipse yet.

You can use the following resources to assist you to install it.

- **Chinese**: http://www.yiibai.com/javafx/install-efxclipse-into-eclipse.html
- **English**: http://wiki.eclipse.org/Efxclipse/Tutorials/AddingE(fx)clipse_to_eclipse#Installing_e.28fx.29clipse_IDE.

1.1.2 JavaFX Scene Builder

JavaFX Scene Builder is a visual layout tool for designing JavaFX application user interfaces, with less coding (even no coding). Since it is quite time consuming and tedious to design user interfaces at times, these types of tools are widely used in industry. You can drag and drop UI components to a work area, modify their properties, apply style sheets, and the FXML code for the layout that they are creating is automatically generated in the background. The result is an FXML file that can then be combined with the rest of your Java project (this involves binding the functionality of your program with the generated code from the graphical interface builder).

Chinese:http://www.yiibai.com/javafx/install-javafx-scene-builder-into-eclipse.html

English:https://o7planning.org/en/10621/install-javafx-scene-builder-into-eclipse

1.2 Intellij IDEA

You need to download JavaFx and the JavaFx Scene Builder plugin. See the following instructions.

English:https://www.jetbrains.com/help/idea/preparing-for-javafx-application-development.html

Chinese:https://blog.csdn.net/hst_gogogo/article/details/82530929

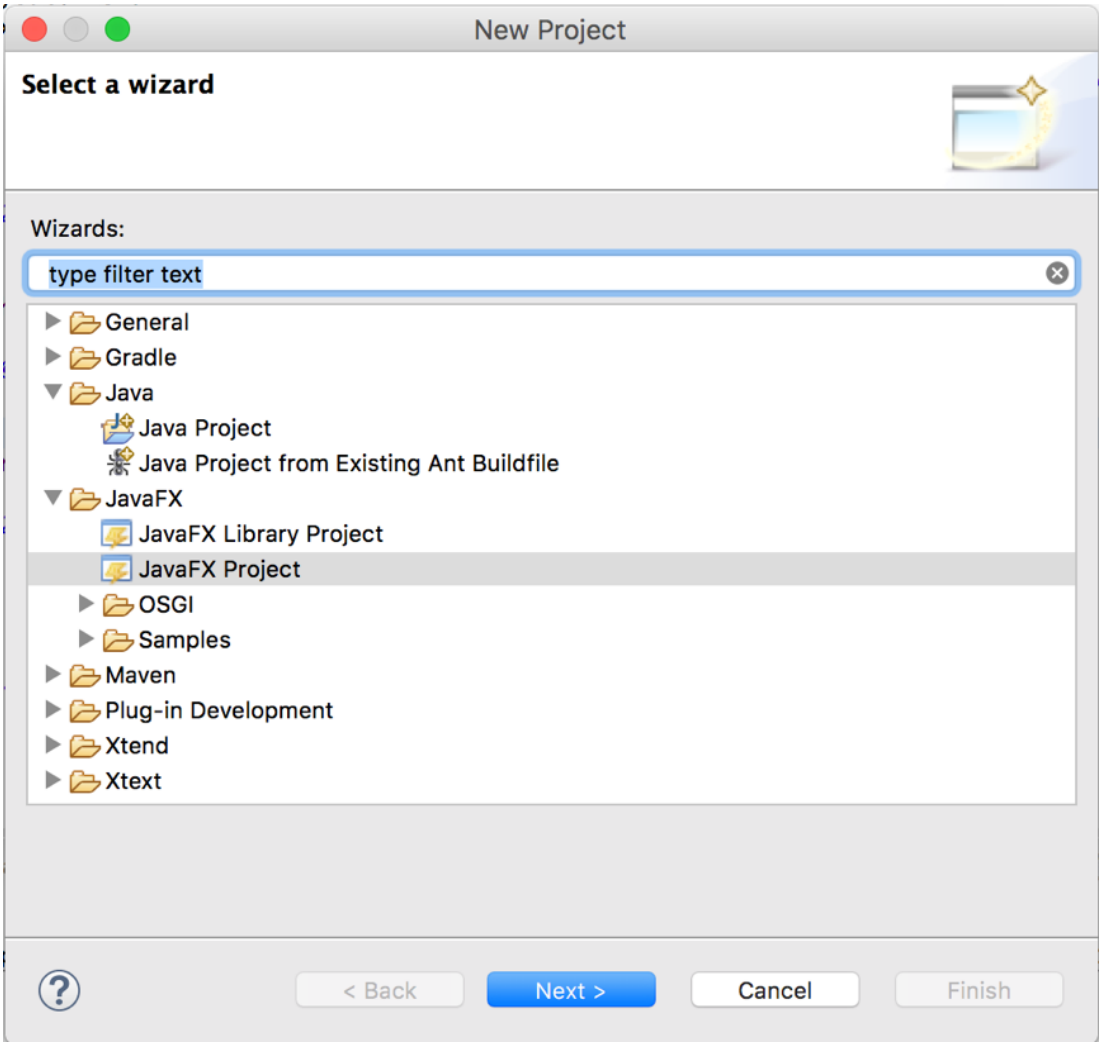
2 Creating and starting a JavaFX project

2.1 Eclipse

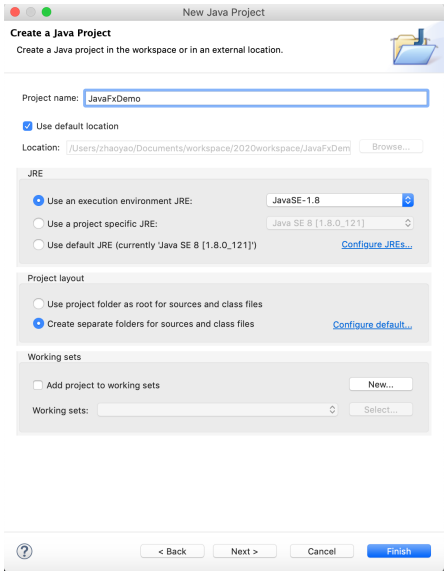
(1) Open Eclipse, Select:

File -> New -> Project

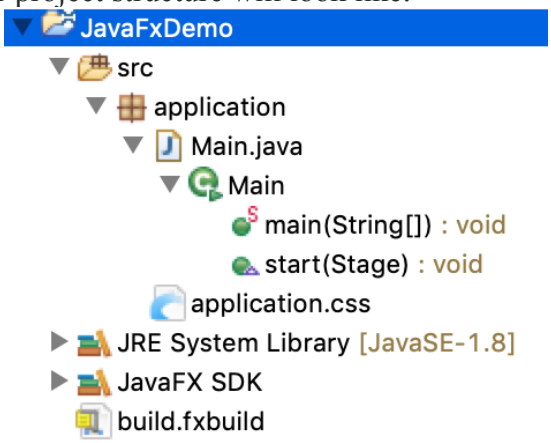
Select JavaFX Project:



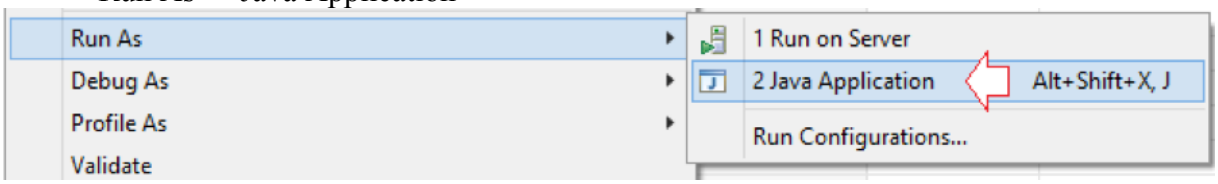
(2) Name your project:



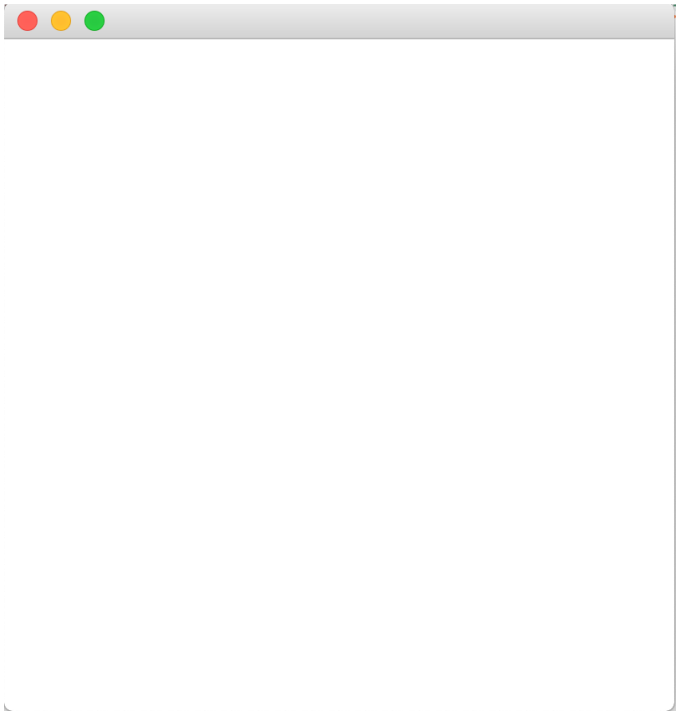
(3) After this your project structure will look like:



(4) Then, let us make sure that JavaFxDemo executes successfully, right click on the main class (Main) and select:
Run As -> Java Application

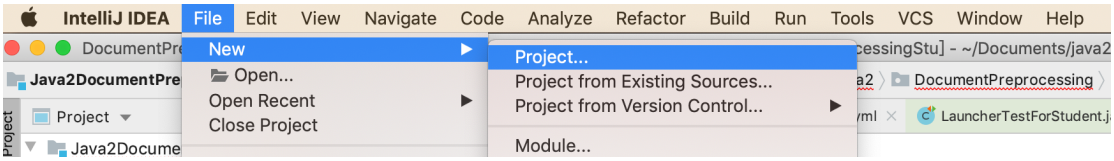


(5) JavaFxDemo application is running, and the result is blank pane (we will put some things there later ☺):

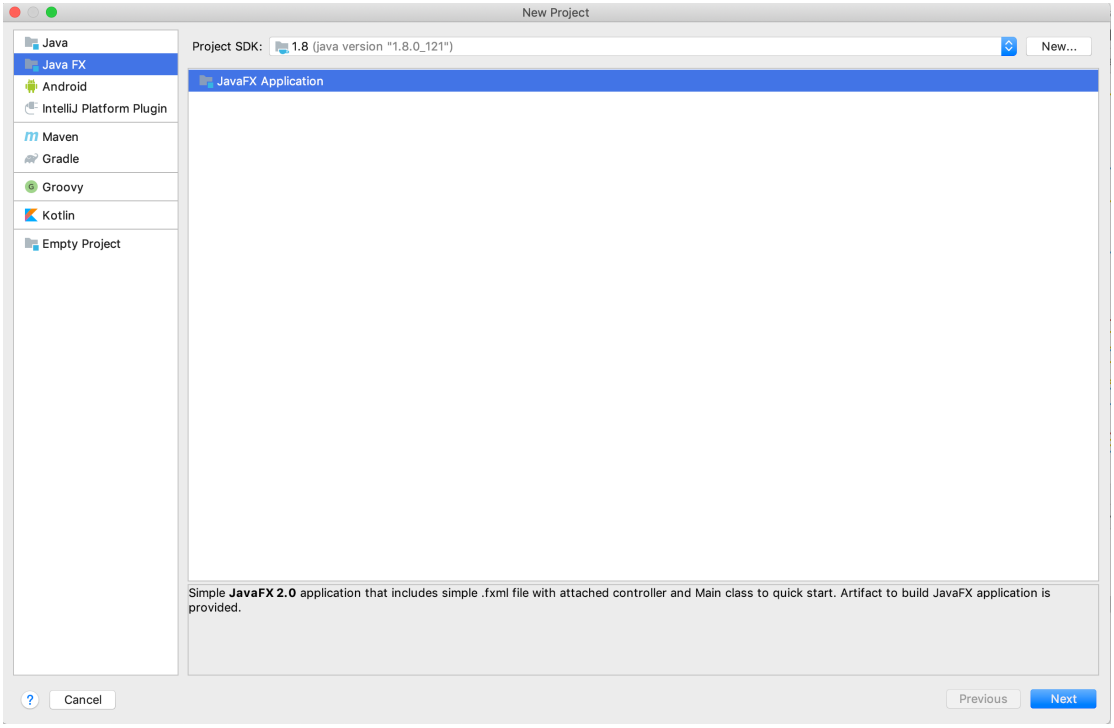


2.2 Intellij IDEA

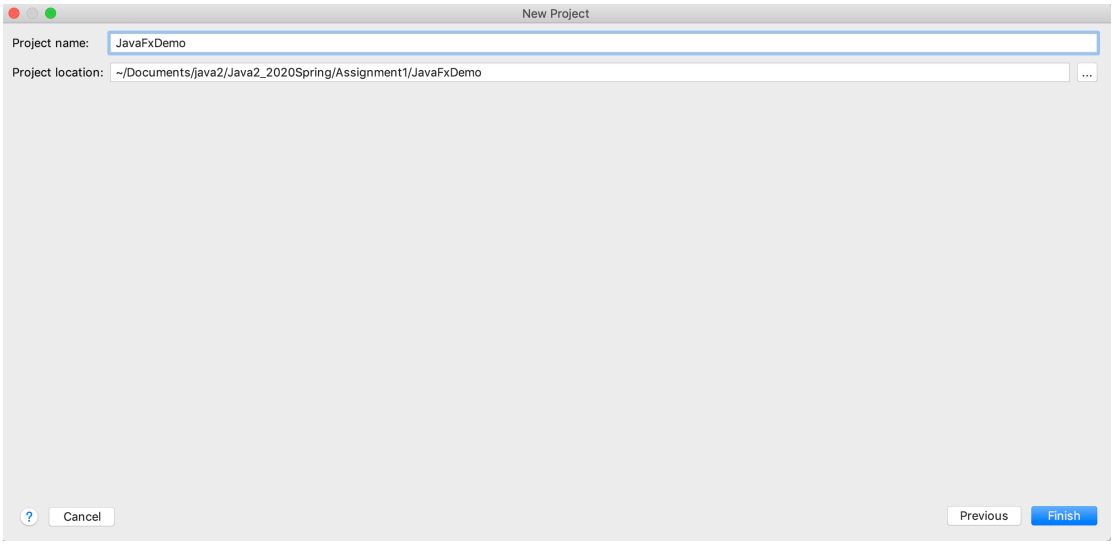
(1) File -> New -> Project



(2) Select JavaFX on the left

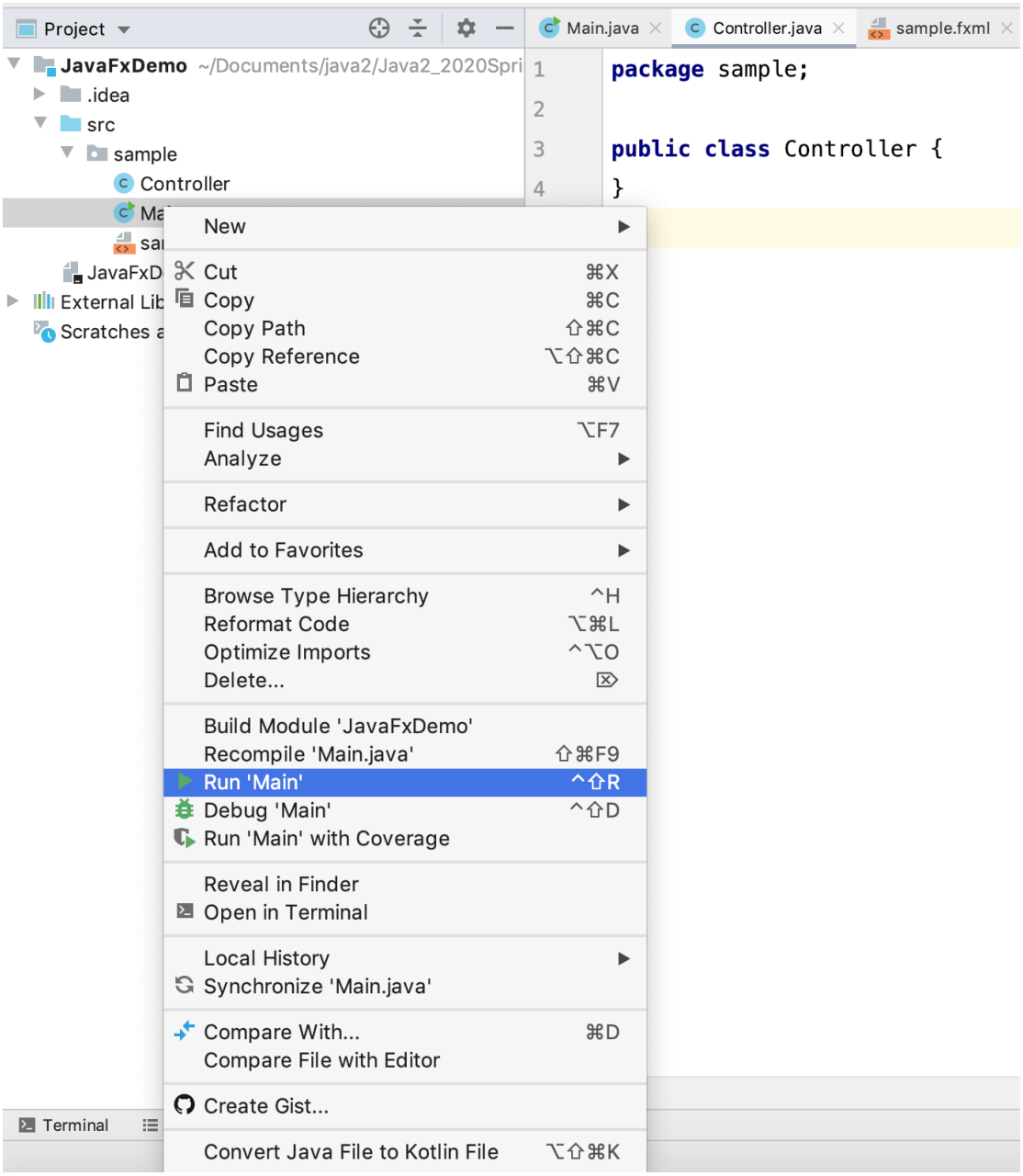


(3) Next -> type name -> Finish



(4) Right click on the main class (Main) and select:

Run ‘Main’

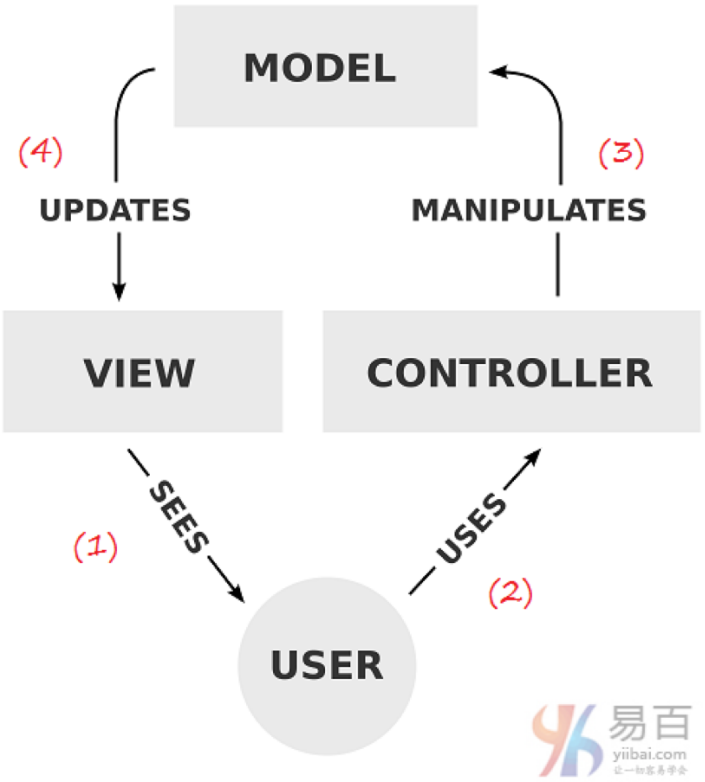


(5) JavaFxDemo application is running, and the result is a blank window with title “Hello World” (nice ☺)

2. Using Scene Builder to design a user interface

In this prac we will now apply scene builder to construct a user interface. See also the lectures for additional information about what we are trying to do.

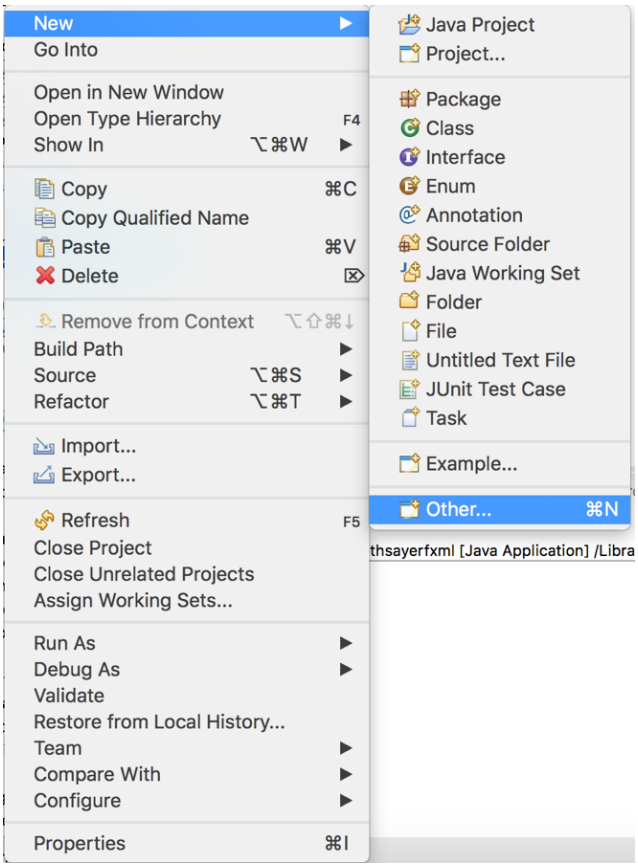
The approach we use in this project is based on the Model View Controller (MVC) paradigm:

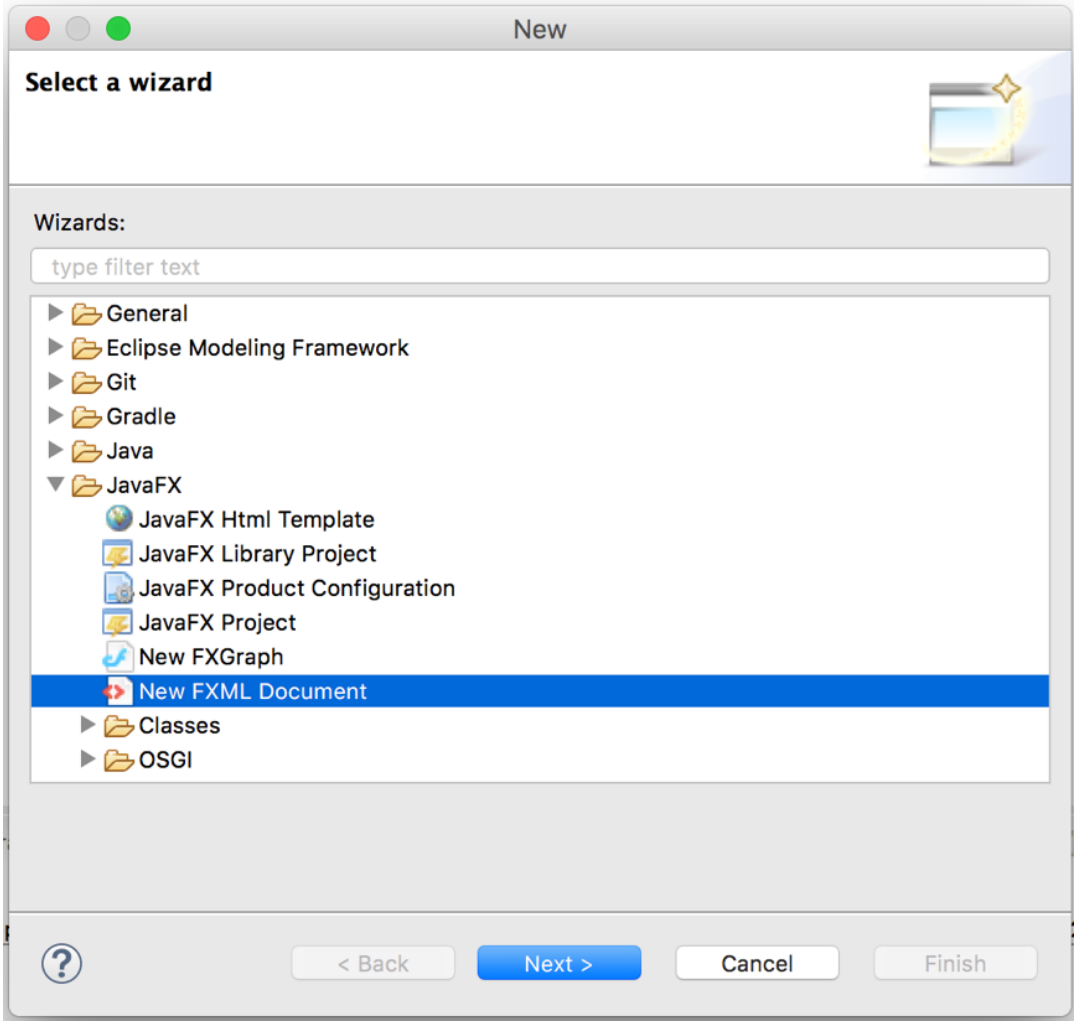


(1) Display on VIEW
(2) User use CONTROLLER
(3) Operate data(Create, update, delete,..) on Model
(4) Display data from MODEL on VIEW

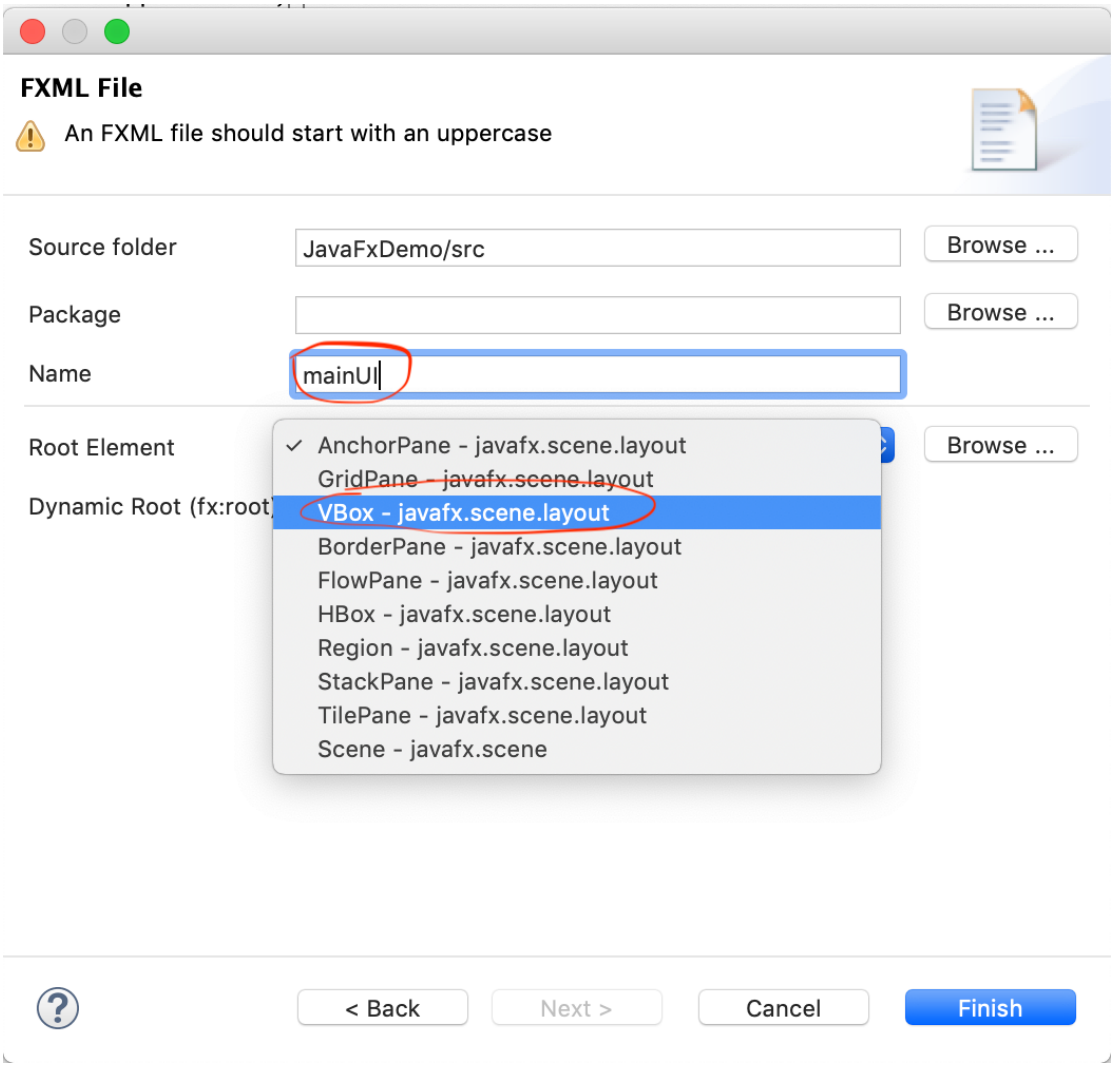
This is a small example of using Scene Builder to design an application interface.

(1) Create a new mainUI.xml file: File -> New -> Other…
Then select “New FXML Document”

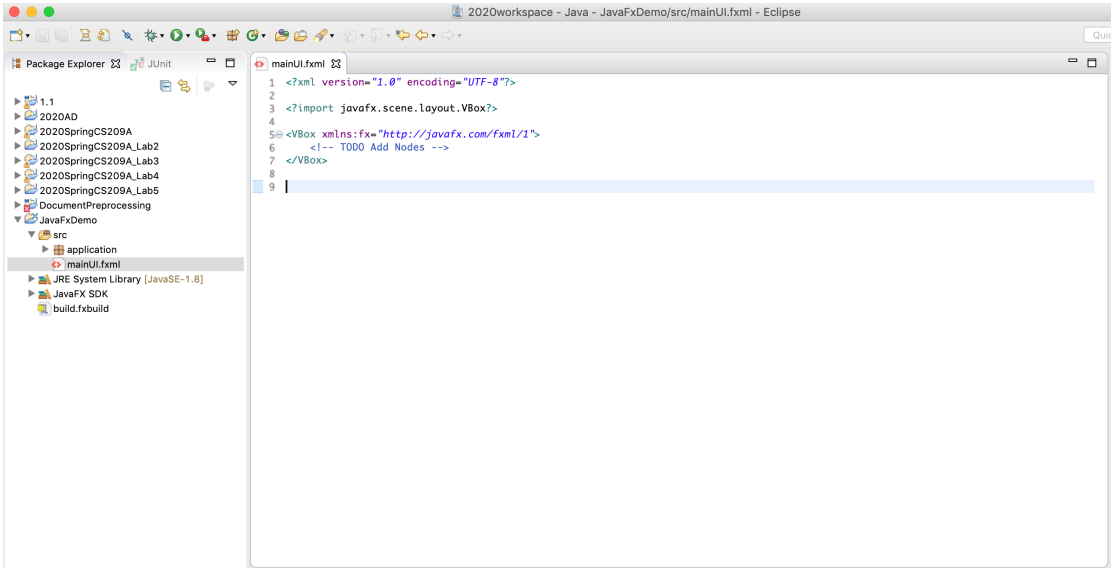

New
Go Into
Open in New Window
Open Type Hierarchy F4
Show In ⌥⌘W
Copy ⌘C
Copy Qualified Name
Paste ⌘V
Delete
Remove from Context ⌥⇧⌘↓
Build Path
Source ⌥⌘S
Refactor ⌥⌘T
Import...
Export...
Refresh F5
Close Project
Close Unrelated Projects
Assign Working Sets...
Run As
Debug As
Validate
Restore from Local History...
Team
Compare With
Configure
Properties ⌘I
Java Project
Project...
Package
Class
Interface
Enum
Annotation
Source Folder
Java Working Set
Folder
File
Untitled Text File
JUnit Test Case
Task
Example...
Other... ⌘N
thsayerfxml [Java Application] /Libra



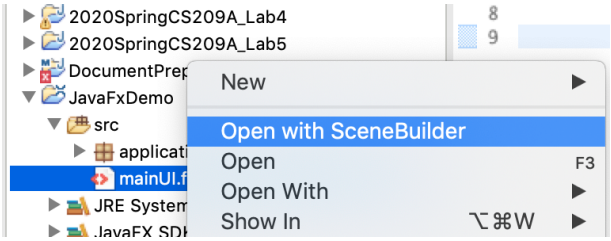
(2) Type the name of file: "mainUI" or other name.(Root element should be VBox)



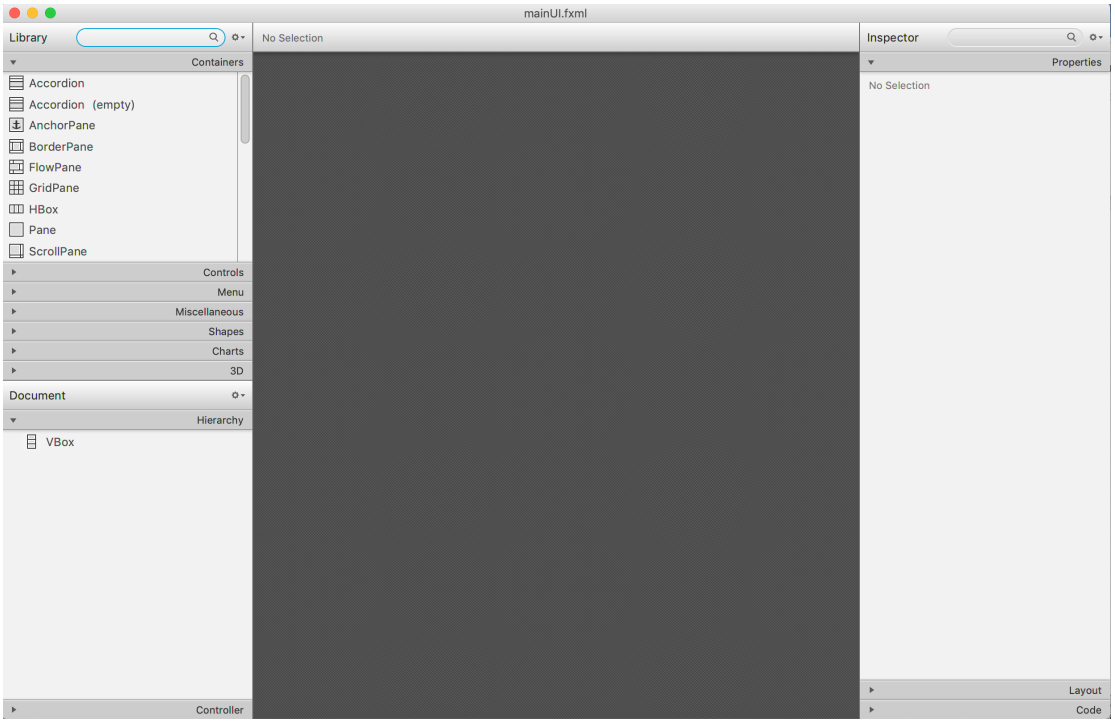
The result should be below:



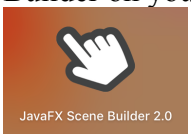
(3) Use Scene Builder to open fxml format file:



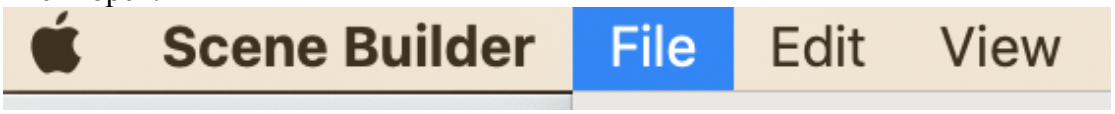
mainUI.fxml design should look like:



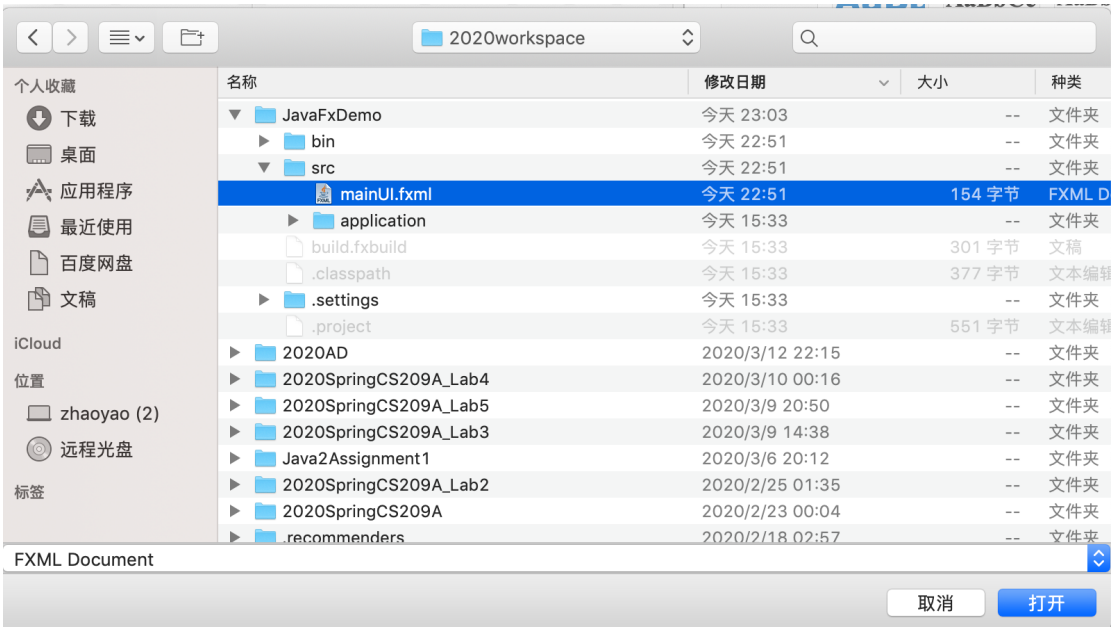
If you eclipse can't associate the fxml with the Scene Builder, you can click the Scene Builder on your desktop.



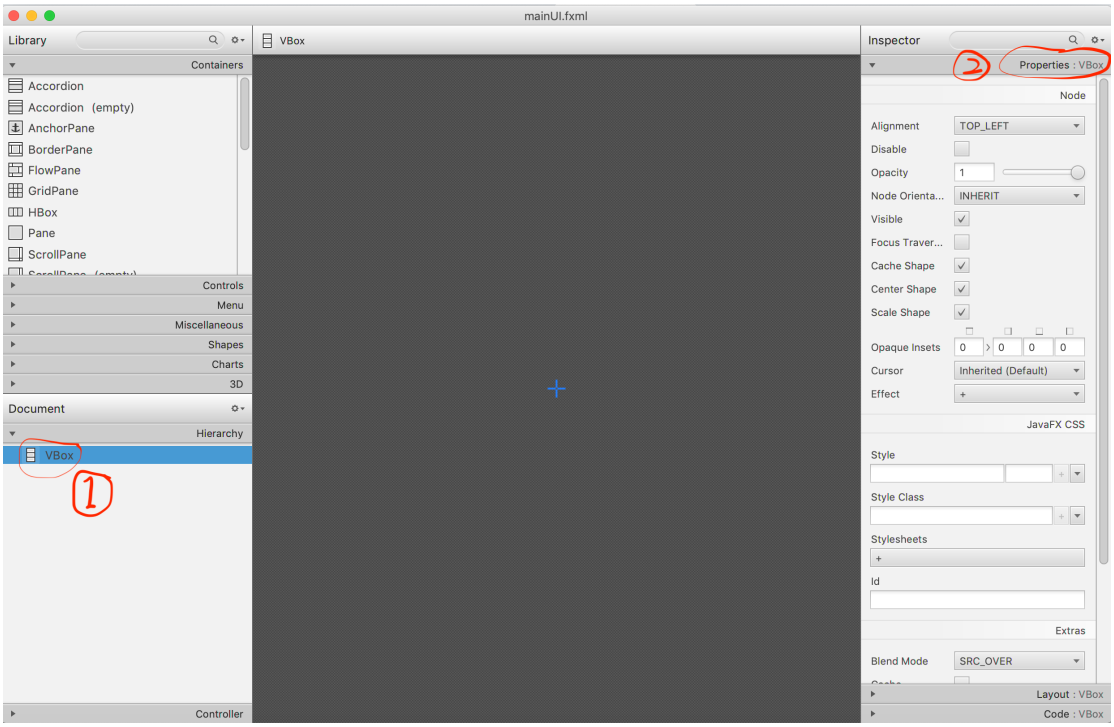
File ->open:



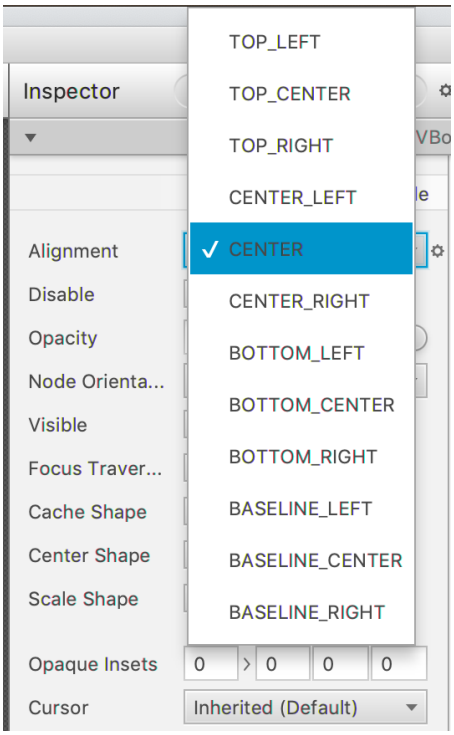
Find your .fxml file and open:



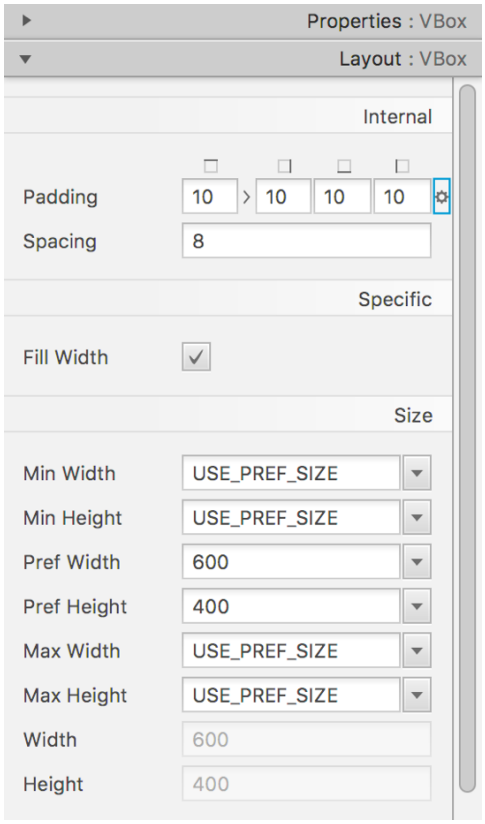
(4) Set the properties of the VBox:



Set “Alignment” to “CENTER”:

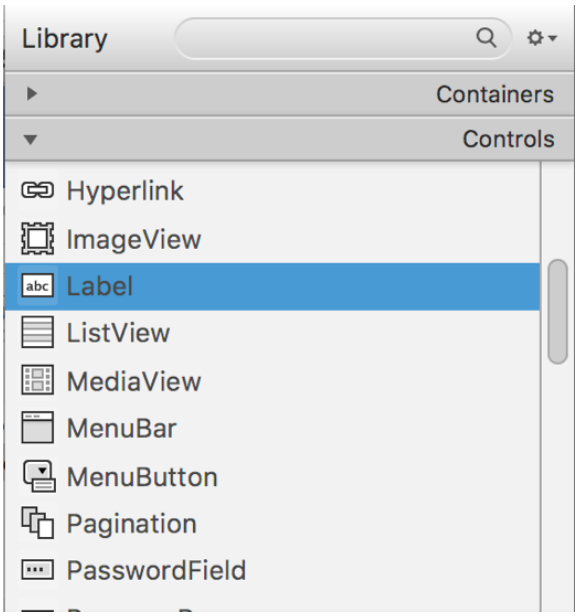


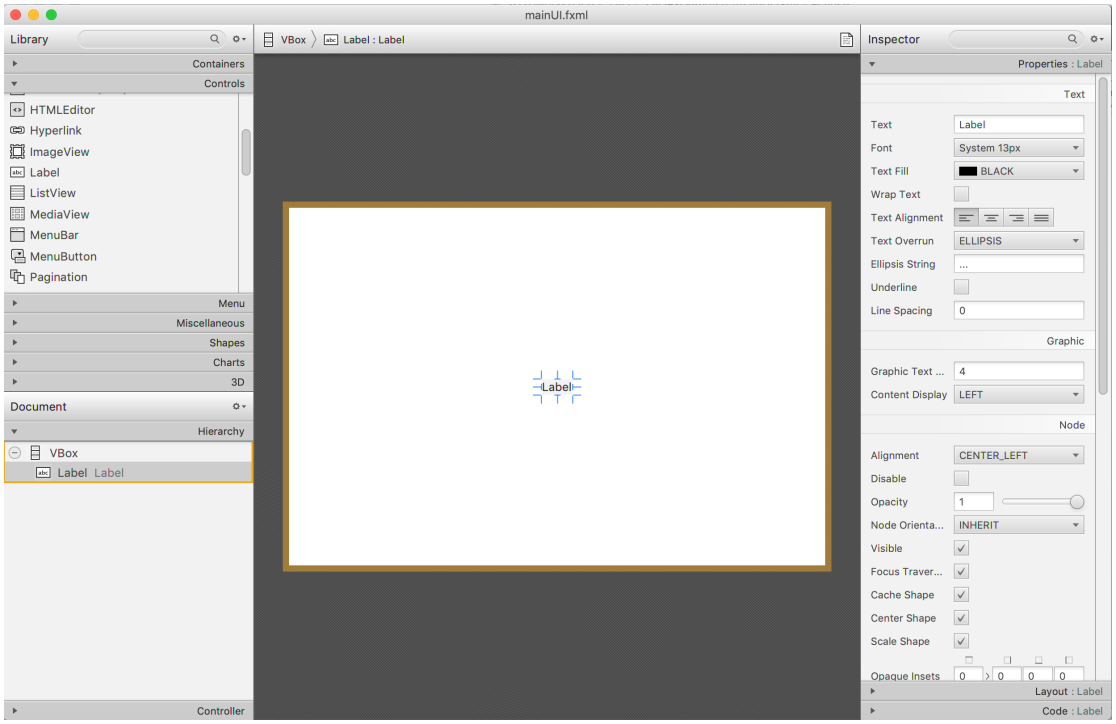
Set Layout:VBox ->Internal and Size, like the following picture:



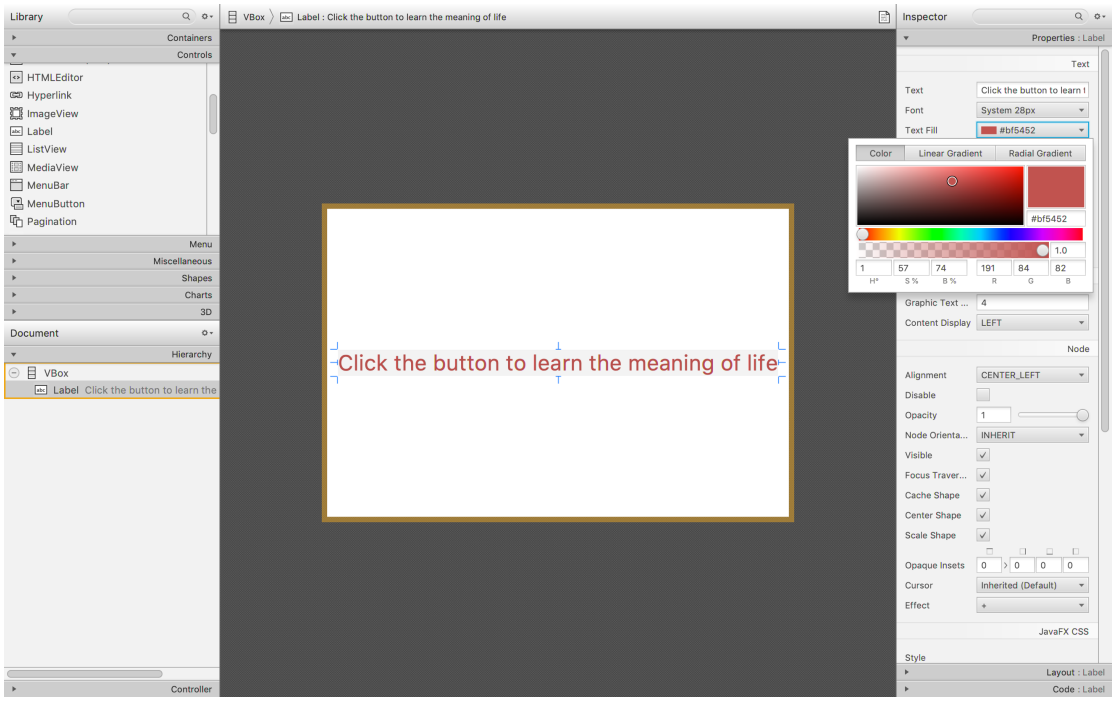
(5) Add a label:

Drag and drop a label into the VBox:



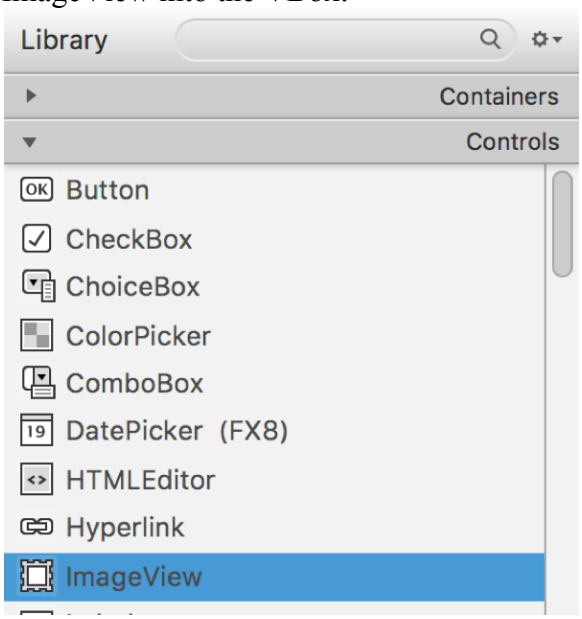


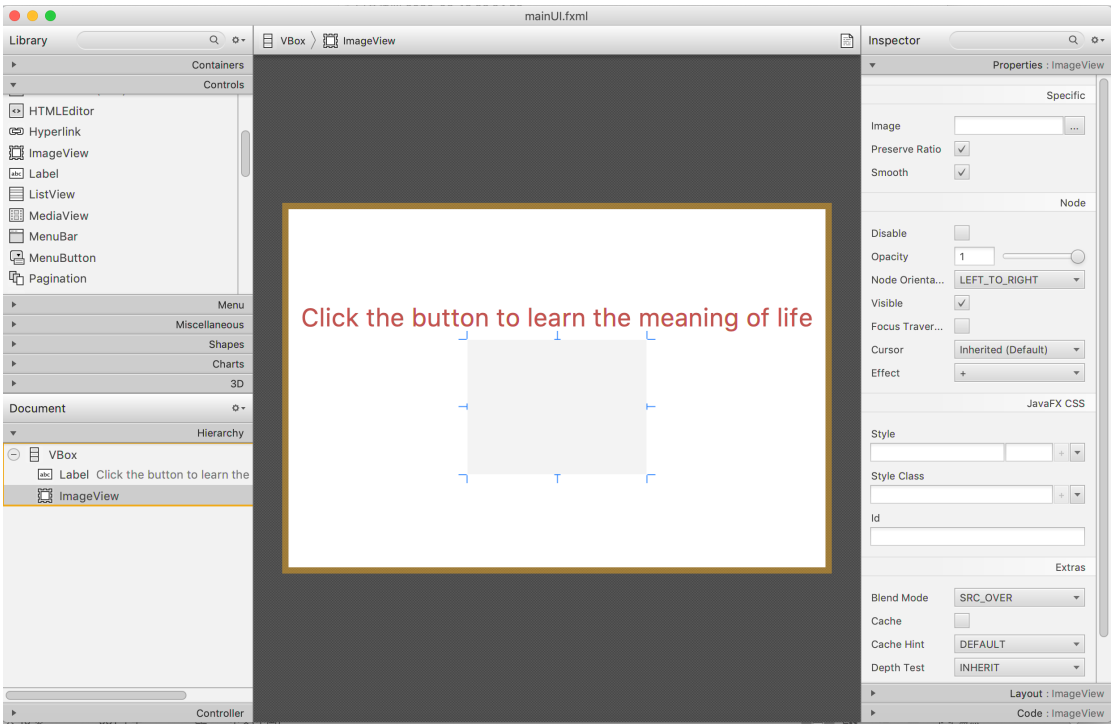
Set its Text to some information to display (For example: "Click the button to learn the meaning of life"), and set the font and text color.



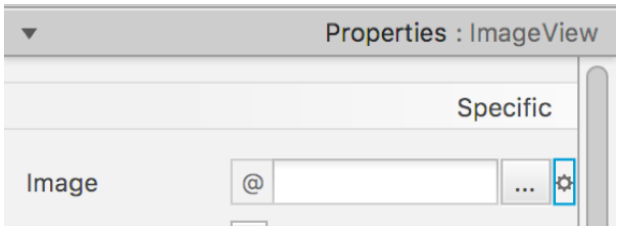
(6) Add a ImageView:

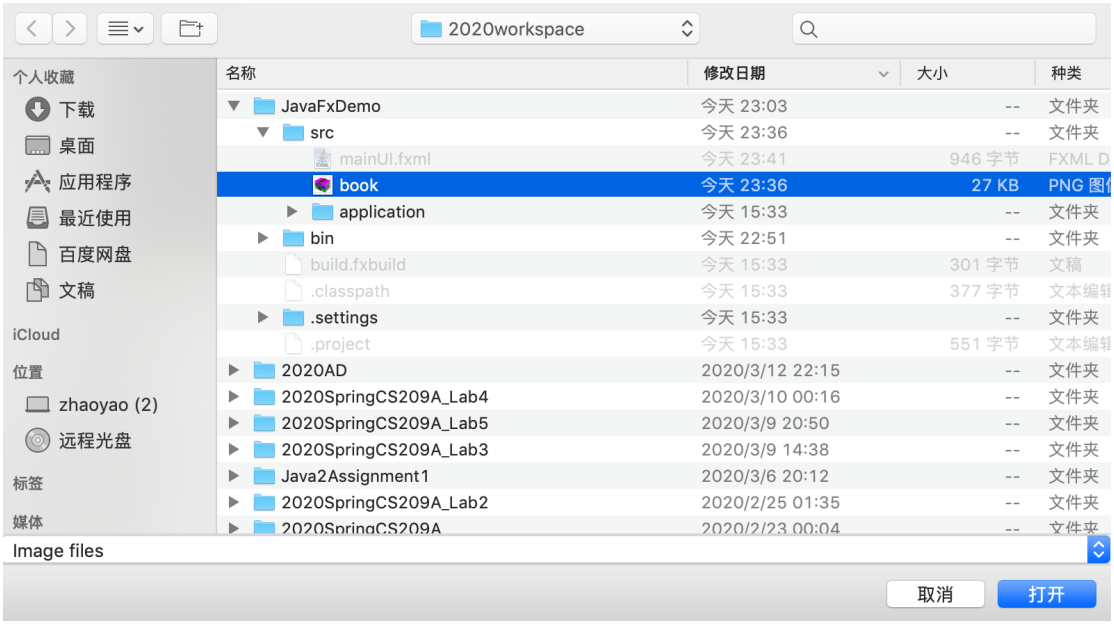
Drag and drop a ImageView into the VBox:

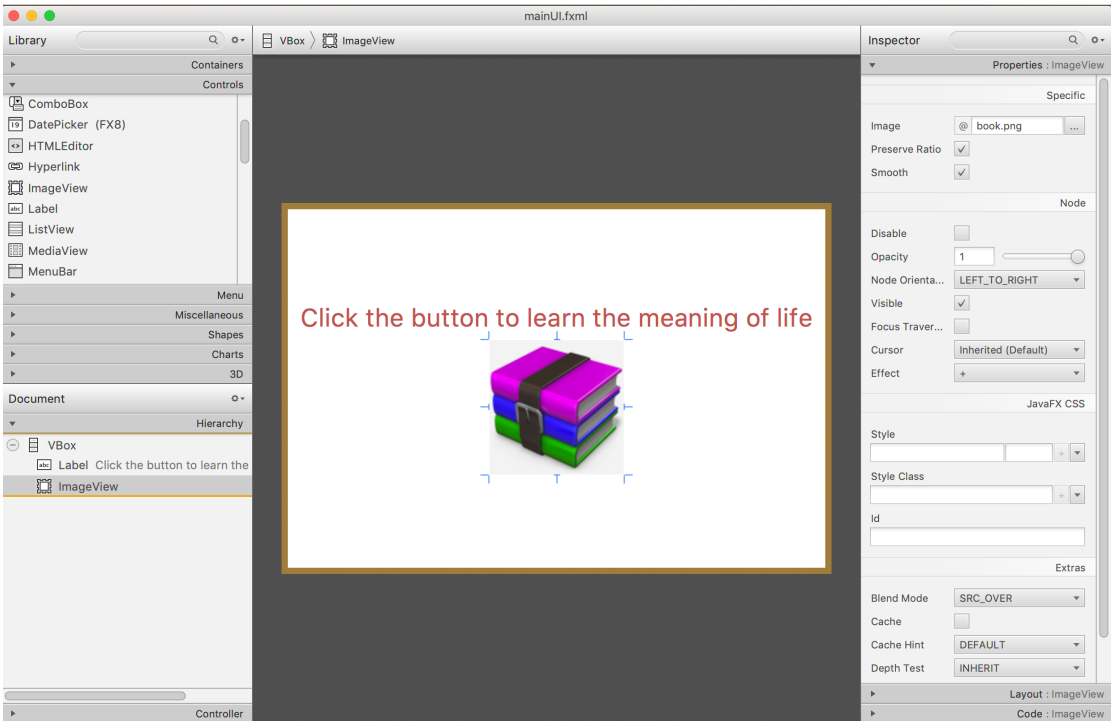




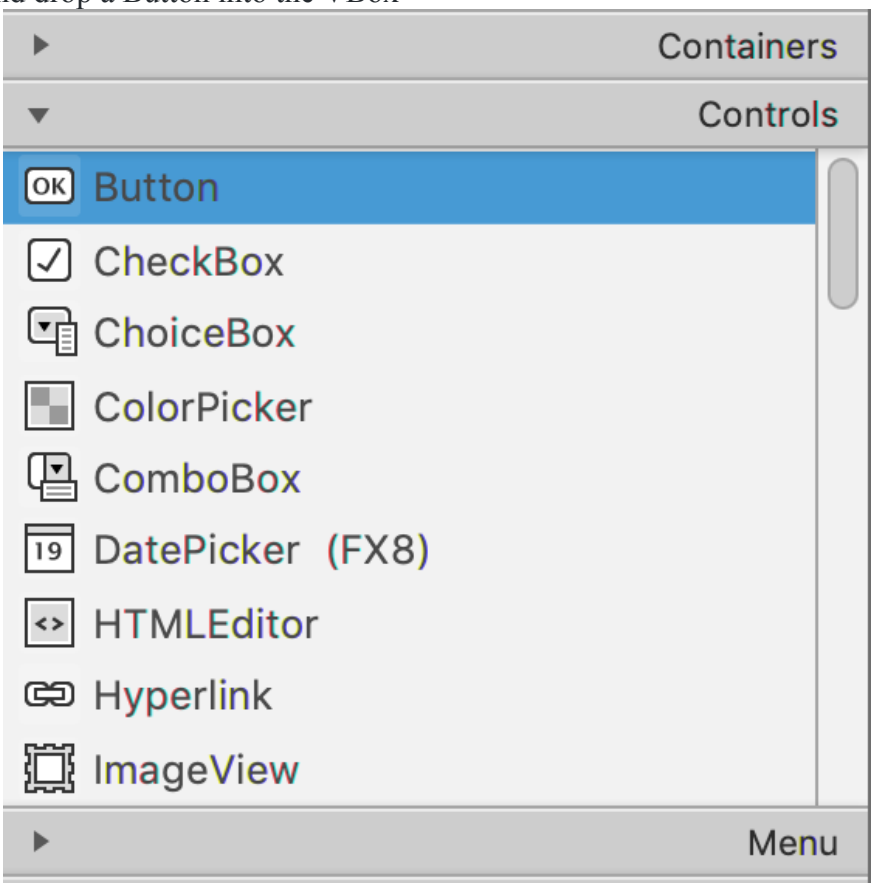
Click "..." select a photo:



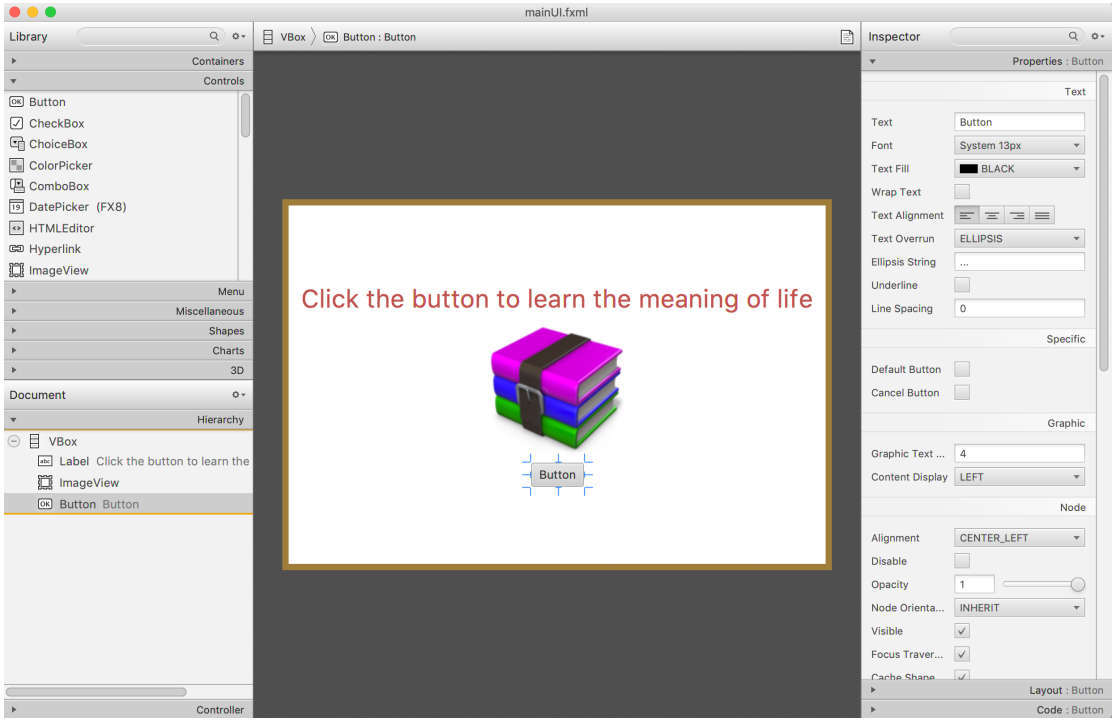


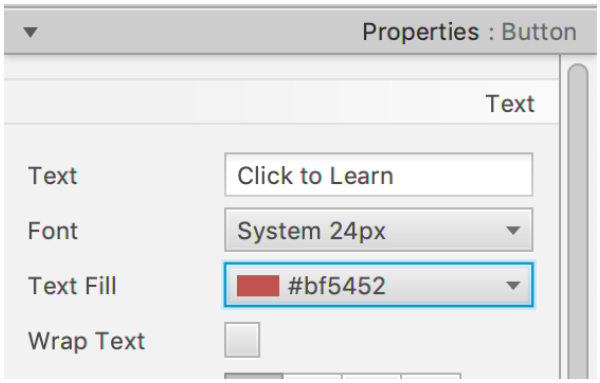


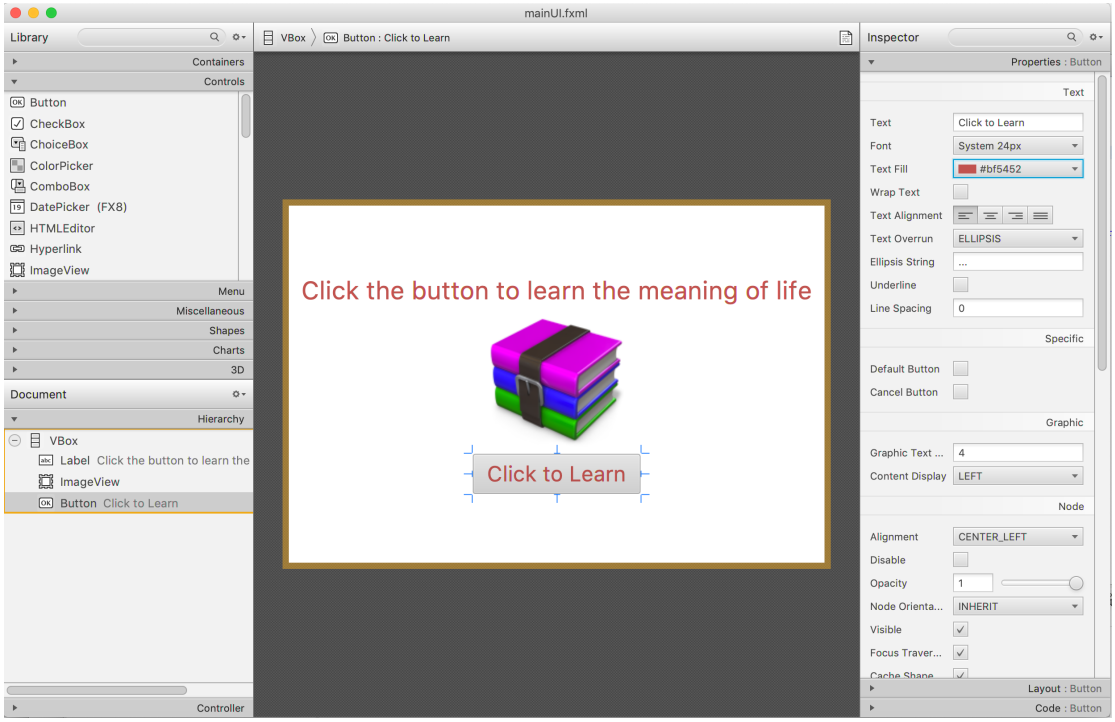
(7) Drag and drop a Button into the VBox



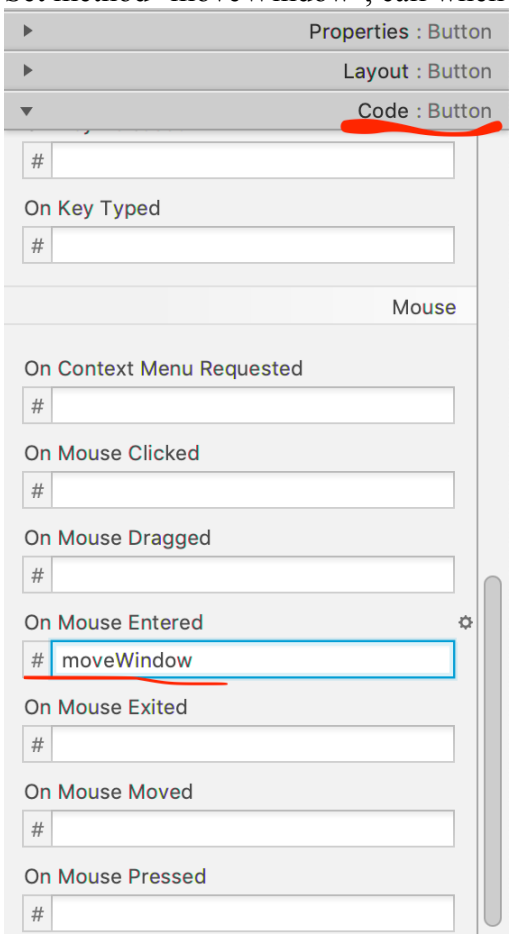
Set it's Text to some information to display(For example:"Click to Learn"),and set the font and text color.


mainUI.fxml
Library
Containers
Controls
Button
CheckBox
ChoiceBox
ColorPicker
ComboBox
DatePicker (FX8)
HTMLEditor
Hyperlink
ImageView
Menu
Miscellaneous
Shapes
Charts
3D
Document
Hierarchy
VBox
Label Click the button to learn the
ImageView
Button Button
Controller
VBox
Button : Button
Click the button to learn the meaning of life
Button
Inspector
Properties : Button
Text
Text
Button
Font
System 13px
Text Fill
BLACK
Wrap Text
Text Alignment
Text Overrun
ELLIPSIS
Ellipsis String
...
Underline
Line Spacing
0
Specific
Default Button
Cancel Button
Graphic
Graphic Text ...
4
Content Display
LEFT
Node
Alignment
CENTER_LEFT
Disable
Opacity
1
Node Orienta...
INHERIT
Visible
Focus Traver...
Layout : Button
Code : Button


Properties : Button
Text
Text
Click to Learn
Font
System 24px
Text Fill
#bf5452
Wrap Text

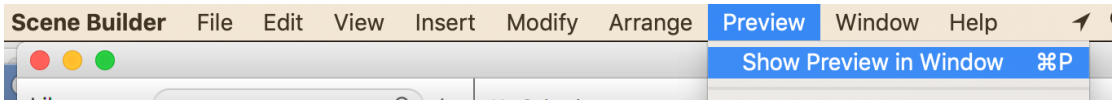


Set method "moveWindow", call when mouse entered.



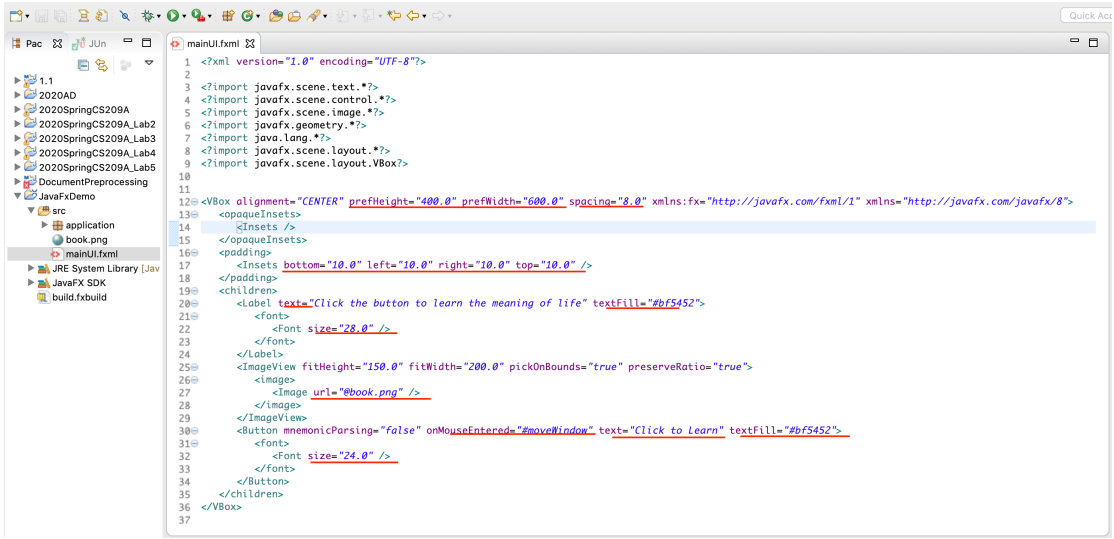
Select "File" -> "Save"

Select "Preview" -> "Show Preview in Window" to preview your design



After all, your design should be like this:

(8) Close JAVAFX Scene Builder and refresh project on eclipse, you can view your code in mainUI.fxml.



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

<?import javafx.scene.text.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.image.*?>
<?import javafx.geometry.*?>
<?import java.lang.*?>
<?import javafx.scene.layout.*?>
<?import javafx.scene.layout.VBox?>


<VBox alignment="CENTER" prefHeight="400.0" prefWidth="600.0" spacing="8.0" xmlns:fx="http://javafx.com/fxml/1" xmlns="http://javafx.com/javafx/8">
   <opaqueInsets>
      <Insets />
   </opaqueInsets>
   <padding>
      <Insets bottom="10.0" left="10.0" right="10.0" top="10.0" />
   </padding>
   <children>
      <Label text="Click the button to learn the meaning of life" textFill="#bf5452">
         <font>
            <Font size="28.0" />
         </font>
      </Label>
      <ImageView fitHeight="150.0" fitWidth="200.0" pickOnBounds="true" preserveRatio="true">
         <image>
            <Image url="@book.png" />
         </image>
      </ImageView>
      <Button mnemonicParsing="false" onMouseEntered="#moveWindow" text="Click to Learn" textFill="#bf5452">
         <font>
            <Font size="24.0" />
         </font>
      </Button>
   </children>
</VBox>
```

Add the property fx:controller to VBox, which will have a reference to a control inside the VBox.

Add the property fx:id ="myButton" to Button, which will have a reference to a control inside the Button.

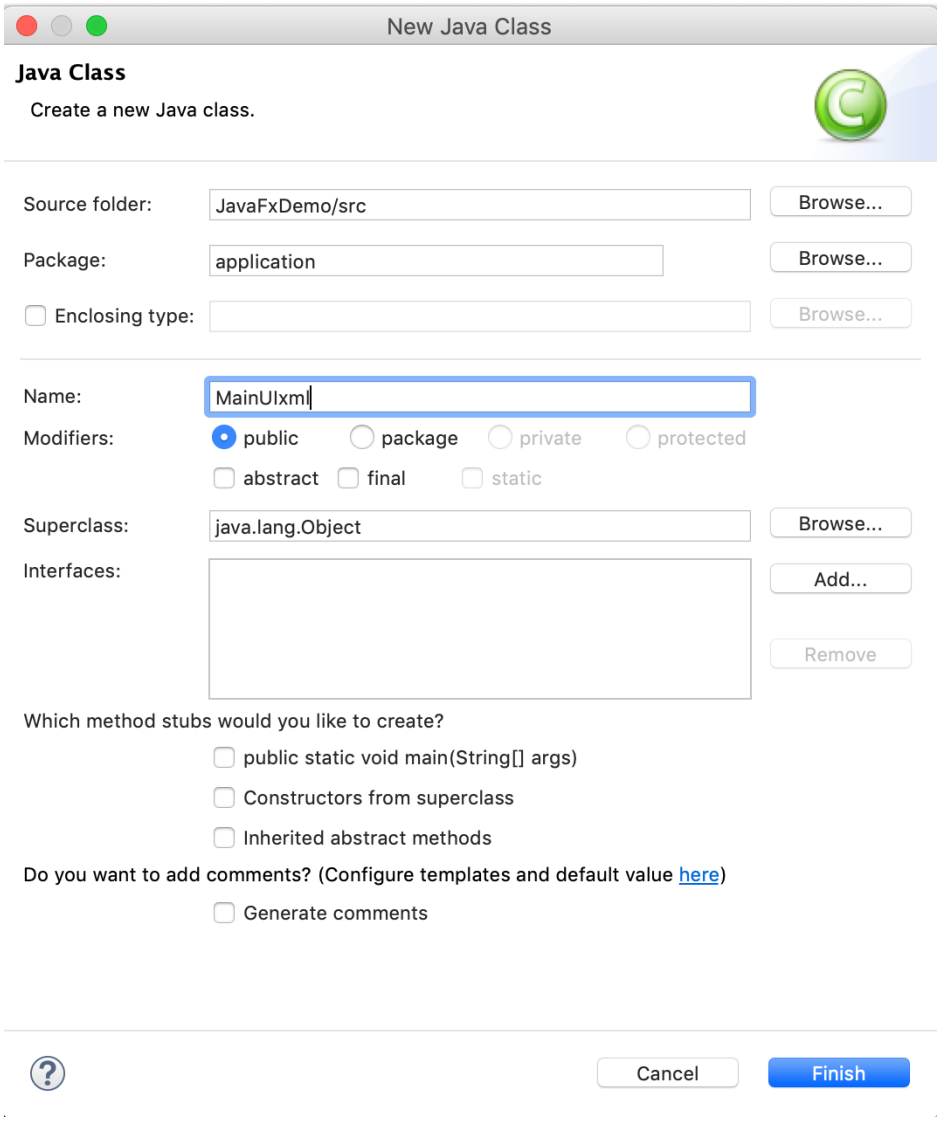
mainUI.fxml

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

<?import javafx.scene.text.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.image.*?>
<?import javafx.geometry.*?>
<?import java.lang.*?>
<?import javafx.scene.layout.*?>
<?import javafx.scene.layout.VBox?>


<VBox alignment="CENTER" prefHeight="400.0" prefWidth="600.0" spacing="8.0" xmlns:fx="http://javafx.com/fxml/1"
 xmlns="http://javafx.com/javafx/8" fx:controller="application.MainUIxml">
   <opaqueInsets>
      <Insets />
   </opaqueInsets>
   <padding>
      <Insets bottom="10.0" left="10.0" right="10.0" top="10.0" />
   </padding>
   <children>
      <Label text="Click the button to learn the meaning of life" textFill="#bf5452">
         <font>
            <Font size="28.0" />
         </font>
      </Label>
      <ImageView fitHeight="150.0" fitWidth="200.0" pickOnBounds="true" preserveRatio="true">
         <image>
            <Image url="@book.png" />
         </image>
      </ImageView>
      <Button fx:id ="myButton" mnemonicParsing="false" onMouseEntered="#moveWindow" text="Click to Learn" textFill="#bf5452">
         <font>
            <Font size="24.0" />
         </font>
      </Button>
   </children>
</VBox>
```

(9) At this point, we create a new class in the application package, named MainUIxml, remember that the name of this class should be the same as the one set in fxml.



(10) Copy the following code to MainUIxml.java:

```
package application;

import java.net.URL;
import java.util.Random;
import java.util.ResourceBundle;

import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.geometry.Rectangle2D;
import javafx.scene.Node;
import javafx.scene.control.Button;
import javafx.scene.input.MouseEvent;
import javafx.stage.Screen;
import javafx.stage.Stage;

public class MainUIxml implements Initializable {
    // Screen size
    private       Rectangle2D screenBounds
                = Screen.getPrimary().getVisualBounds();
    // The random generator MUST be static
    private static Random    rand_generator ;
    // Keep track of the current location of the window
```

```
    private double    x;
    private double    y;


    @Override
    public void initialize(URL location, ResourceBundle resources) {
        // TODO Auto-generated method stub
        rand_generator = new Random();
    }

    public void moveWindow(MouseEvent me) {
        Node node = (Node) me.getSource();
        // Returns a reference to the button
        Stage stage = (Stage) node.getScene().getWindow();
        double height = screenBounds.getHeight();
        double width = screenBounds.getWidth();

        // Add a fixed value to make sure that the Window
        // moves far enough
        double x_move = width / 10 + rand_generator.nextDouble() * width /
2;
        double y_move = height / 10 + rand_generator.nextDouble() * height
/ 2;
        // As x and y represent the upper left corner, we
        // take care of not having part of most of the
        // window outside the screen
        this.x = (double) ((long) (this.x + x_move) % (long) (width -
stage.getWidth()));
        this.y = (double) ((long) (this.y + y_move) % (long) (height -
stage.getHeight()));
        stage.setX(this.x);
        stage.setY(this.y);
    }
}
```

(11)Copy the following code to Main.java:

```
package application;

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

public class Main extends Application {
    @Override
    public void start(Stage primaryStage) {
        try {
            Parent root =
FXMLLoader.load(getClass().getResource("../mainUI.fxml"));

            primaryStage.setTitle("My Application");
            primaryStage.setScene(new Scene(root));
            primaryStage.show();
        } catch (Exception e) {
            e.printStackTrace();
        }
```

```
    }

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

(12) Run the application.

5. Exercise

You should work through the first Parts before Mar. 24, and submit a short video to demonstrate that you did (and also allow others to see how you have set up your system and software in the peer review).

6. Extension

https://code.makery.ch/library/javafx-tutorial/

If you find the tutorial easy, please also start to follow the above tutorial step by step as it provides additional coverage that we will go through in the next week.