WESTERN UNIVERSITY - CANADA
FACULTY OF ENGINEERING
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

SE 2203B – SOFTWARE DESIGN

Laboratory 1: Creating GUI Applications with JavaFX and Scene Builder

Due Date: January 20, 2023 – 11:55PM

For these exercises, no need to physically attend the lab, but If you get stuck on something in the following lab requirements, you should ask your TA in person during the designated lab hours in ACEB 4435. The designated lab hours are mentioned in the Lab schedule document in OWL.

1 Goal

- Be able to create a GUI with Scene Builder and save it to an FXML file
- Be able to edit the JavaFX application's main application class
- Be able to run and test a standalone JavaFX application.

2 Resources

- Rose.gif, Cherry Blossom.gif, Canna.gif, Cherry Blossom.gif, and Rose.gif image files

3 Introduction

JavaFX is a standard Java library for developing rich applications that employ graphics. You can use it to create GUI applications, as well as applications that display 2D and 3D graphics. You can use JavaFX to create standalone graphics applications that run on your local computer, applications that run from a remote server, or applications that are embedded in a Web page.

We will use the IntelliJ IDE along with the Scene Builder library to create the JavaFX-based applications. Once we create a new JavaFX project, the IntelliJ IDE creates three files: Java main class file, FXML layout file, and Java control class file.

1. The FXML layout file.
   - We use the Scene Builder to create the GUI controls by dragging and dropping the relevant components onto a blank window. We visually arrange the components on the window and set various properties to create the appearance that we want for the application. The IDE then generates the corresponding FXML file.
   - Once the FXML file is created, the next step is to write Java code that reads the FXML file and displays the GUI on the screen and handles any events that occur while the application is running. This code should be written in the following two classes:
2. The main application Java class. This class is responsible for building the scene graph and performs the following:
   - Loads the FXML file.
   - Builds the scene graph in memory.
   - Displays the GUI

3. The controller Java class. This class is responsible for handling events that occur while the application is running. The controller class contains the following items:
   - The necessary import statements.
   - Private variables to reference the components that have an fx:id in the scene graph.
   - An optional initialize method that is automatically called after the FXML file is loaded.
   - Event listener methods

4 Directed Lab Work

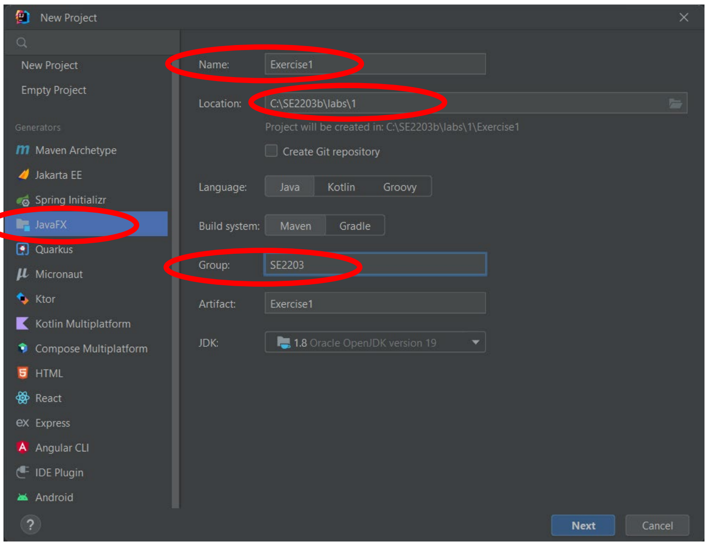
4.1 Setting up IntelliJ IDE and Java FX Scene Builder

- Please watch the posted video for Week 1 "How to install JavaFX and Scene Builder for the IntelliJ IDE", and then perform the same steps in your computer.

4.2 Run Hello application to make sure your JavaFX is properly installed

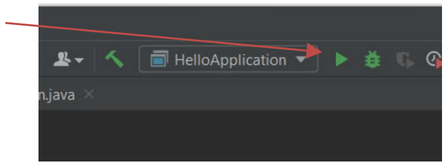
4.2.1 Construct the Application

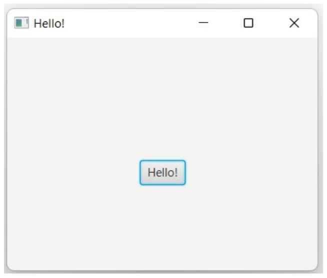
1. From IntelliJ IDE **File** menu, choose **New Project**. In the New Project pop window, select **JavaFX** from the left-hand side menu.
2. Name the project **Exercise1**, setup the other fields as shown and then click Next and then Finish.
3. IntelliJ IDE opens an FXML sample project that includes the code for a basic window titled "Hello!". The application includes three files:
   - **HelloApplication.java**. This file takes care of the standard Java code required for an FXML application.
   - **hello-view.fxml**. This is the FXML source file in which you define the user interface.
   - **HelloController.java**. This is the controller file for handling the mouse and keyboard input.



Name: Exercise1
Location: C:\SE2203b\labs\1
Project will be created in: C:\SE2203b\labs\1\Exercise1
Create Git repository
Language: Java Kotlin Groovy
Build system: Maven Gradle
Group: SE2203
Artifact: Exercise1
JDK: 1.8 Oracle OpenJDK version 19

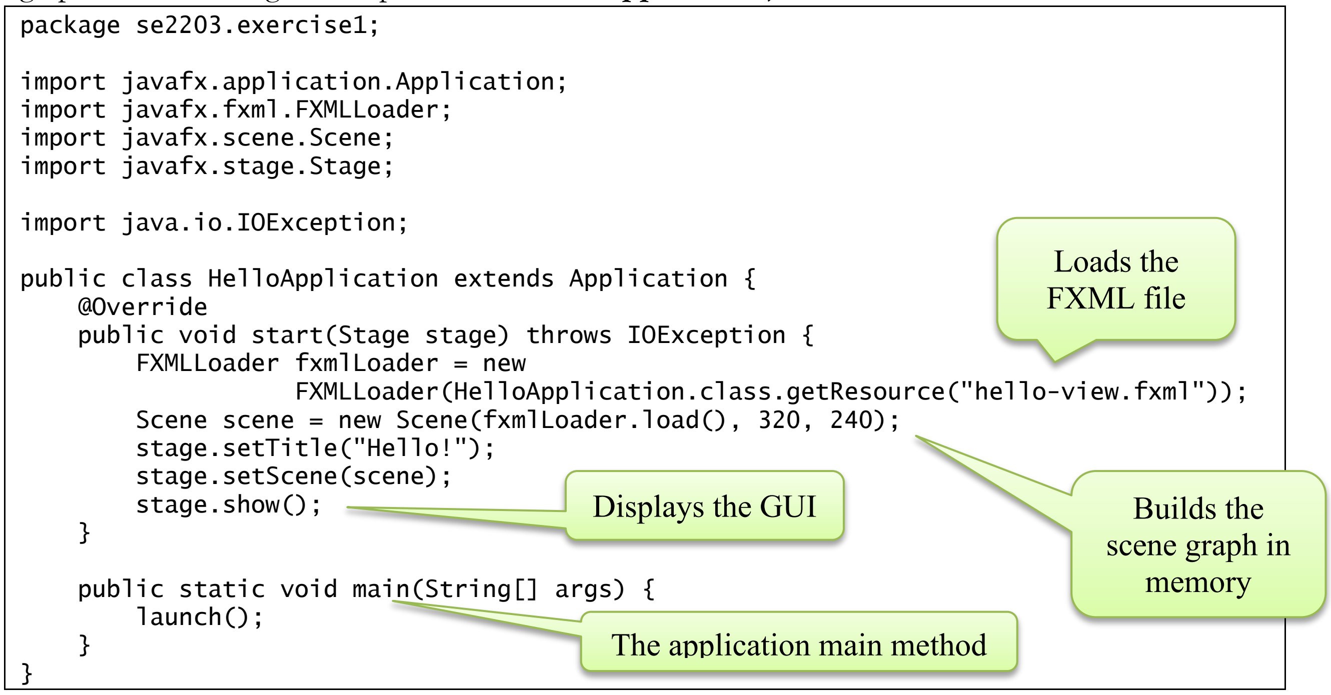
4. Now, run the HelloApplication. Click the green in the top menu option or in front of the Strat method.
5. The application should build and run without error and display the following screen.





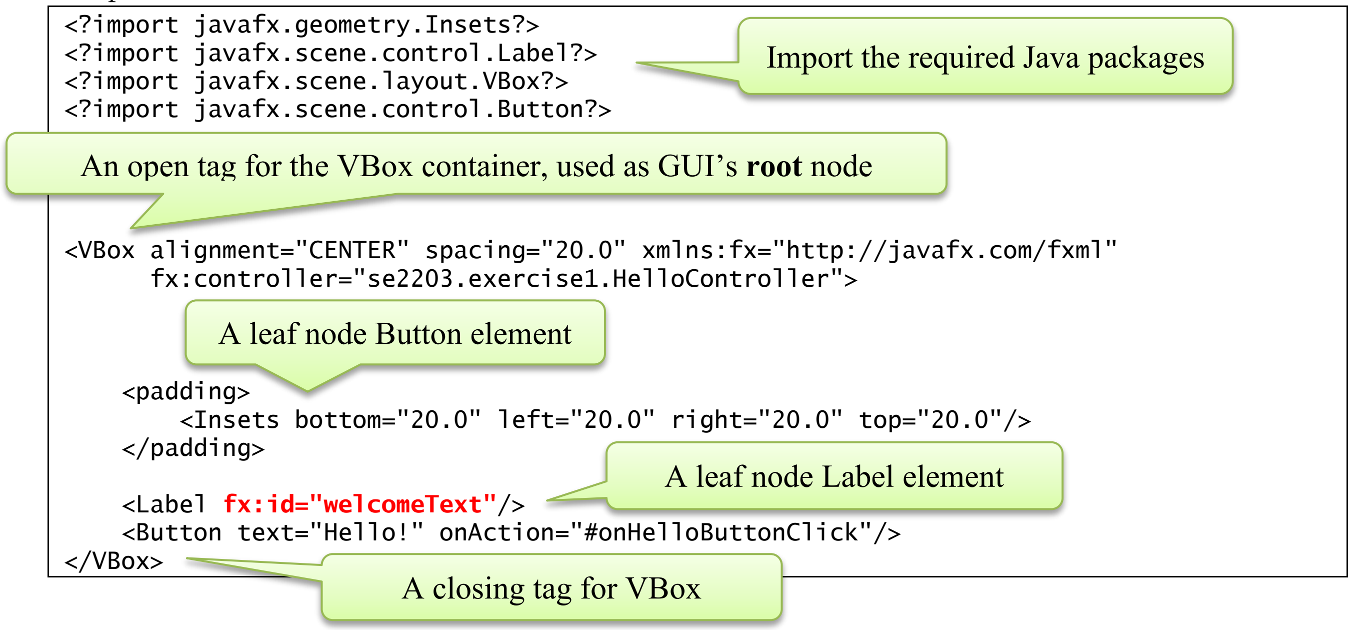
4.2.2 Examine the FXML-based Project Source File

- The first file you examine is the **HelloApplication.java** file. This file includes the code for setting up the application main class and for defining the stage and scene. The file uses the **FXMLLoader class**, which is responsible for loading the FXML source file and returning the resulting object graph. The following code represent the **HelloApplication.java** file:



```
package se2203.exercise1;

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

import java.io.IOException;

public class HelloApplication extends Application {
    @Override
    public void start(Stage stage) throws IOException {
        FXMLLoader fxmlLoader = new
                   FXMLLoader(HelloApplication.class.getResource("hello-view.fxml"));
        Scene scene = new Scene(fxmlLoader.load(), 320, 240);
        stage.setTitle("Hello!");
        stage.setScene(scene);
        stage.show();
    }

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

- The second file you examine is the **hello-view.fxml** file. This file is written (or generated by Scene Builder) to build the user interface for JavaFX applications. FXML is a markup language that describes the components in a JavaFX scene graph. FXML uses tags to organize data, in a manner similar to the way that HTML uses tags to format text in a Web browser. The following code represent the content of the **hello-view.fxml** file:



```
<?import javafx.geometry.Insets?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.control.Button?>


<VBox alignment="CENTER" spacing="20.0" xmlns:fx="http://javafx.com/fxml"
      fx:controller="se2203.exercise1.HelloController">


    <padding>
        <Insets bottom="20.0" left="20.0" right="20.0" top="20.0"/>
    </padding>

    <Label fx:id="welcomeText"/>
    <Button text="Hello!" onAction="#onHelloButtonClick"/>
</VBox>
```

Import the required Java packages
An open tag for the VBox container, used as GUI's **root** node
A leaf node Button element
A leaf node Label element
A closing tag for VBox

- The third file you examine is the HelloController.java file. The following code represent the content of the HelloController.java:

```
package se2203.exercise1;

import javafx.fxml.FXML;
import javafx.scene.control.Label;

public class HelloController {
    @FXML
    private Label welcomeText;

    @FXML
    protected void onHelloButtonClick() {
        welcomeText.setText("Welcome to JavaFX Application!");
    }
}
```

The necessary import statements

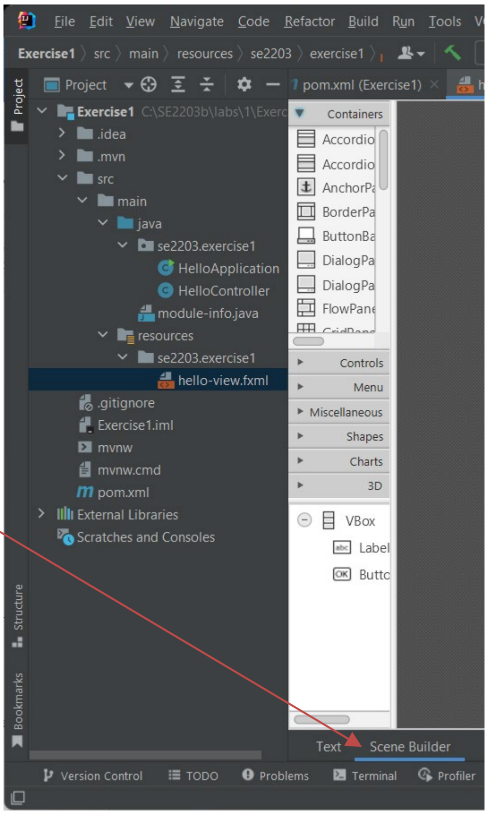
Private variable to reference the components that have a **fx:id** in the FXML file. See id written in red in the above code list.

Event listener method

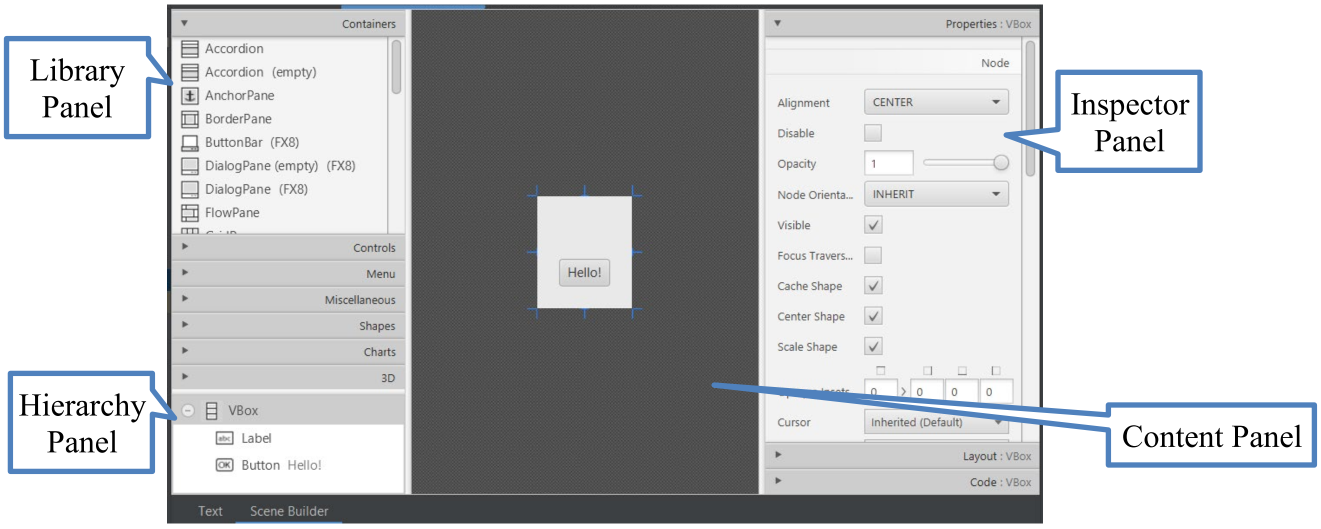
4.2.3 Using Scene Builder tool to generate the FXML sense file

JavaFX Scene Builder is a visual layout tool that lets users quickly design JavaFX application user interfaces, without coding. Users 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 (e.g. hello-view.fxml) that can then be combined with a Java project by binding the UI to the application's logic.

- Double-click click the hello-view.fxml file and then select the "Scene Builder" tab to start edit this file.
- The main window for the JavaFX Scene Builder tool appears showing the correct dessin of the Hello scene. See the Figure below.
- Here is a brief summary of each part of the main window.
  - **Menu Bar**: Scene Builder's commands are located on the menus that access the menu bar at the top of the main window.
  - **Library Panel**: The Library Panel provides a list of JavaFX components that you can use in an application. To place a component in a GUI, you simply drag it from the Library Panel, and drop it into the Content Panel.
  - **Content Panel**: The Content Panel is where you visually design an application's GUI. You can create other components in the GUI by dragging them from the Library Panel and dropping them onto the root node component.

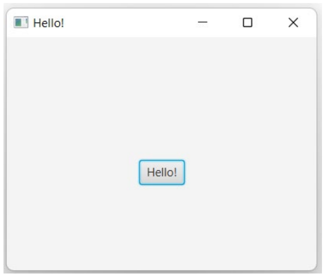


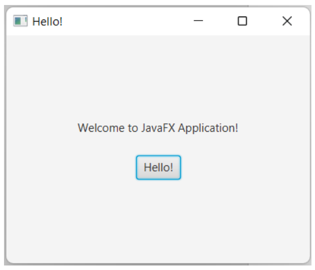
- **Hierarchy Panel**: The Hierarchy Panel shows the GUI's scene graph. As you add components to the Content Panel, nodes appear in the Hierarchy Panel. You can use the Hierarchy Panel to select nodes that you want to edit.
- **Selection Bar**: This area of the screen shows the hierarchical path of the currently selected node in the scene graph.
- **Inspector Panel**: The Inspector Panel is divided into three sections:
  - *Properties*: allows you to view and edit the values of the selected component's properties, which are values that determine the component's appearance
  - *Layout*: lets you specify values that control the way the component is displayed when the GUI's window is resized
  - *Code*: allows you to assign an fx:id to a component, which is similar to assigning a variable name to the component. also allows you to designate event handlers to execute when specific events happen to the selected component



4.2.4 Run your application and export the project file

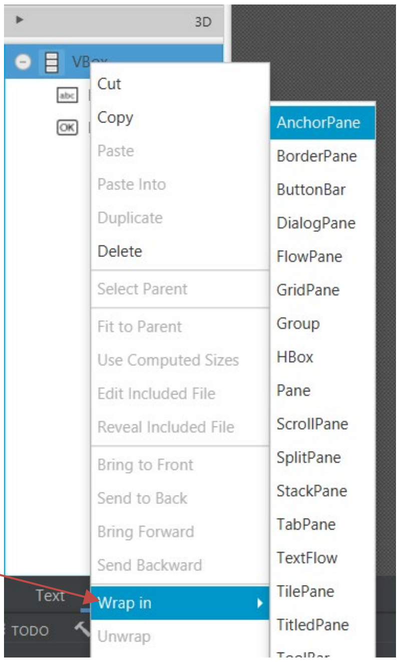
1. Now run your application, click the green arrowhead in the top main menu.
2. Click the "Hello!" button.
3. Verify that the text "Welcome to JavaFX Application!" is displayed above the button.
4. From the IntelliJ main menu, select File → Export to Zip File…, the Save as pop-up window appears, click OK to save your project as Exercise1.zip file.
5. Now you are ready for the next Exercise.



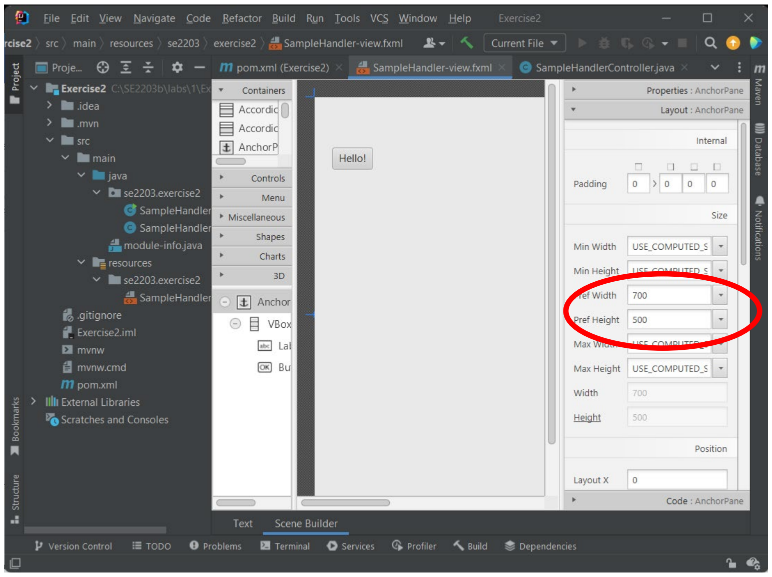


4.3 Creating an Event Handler for a Control

Programs that operate in a GUI environment must be event-driven. An event is an action that takes place within a program, such as the clicking of a button. Part of writing a GUI application is creating event listeners. An event listener is a method that automatically executes when a specific event occurs. In this exercise, you will create a **label** control and an event handler for **onMouseEntered** event.

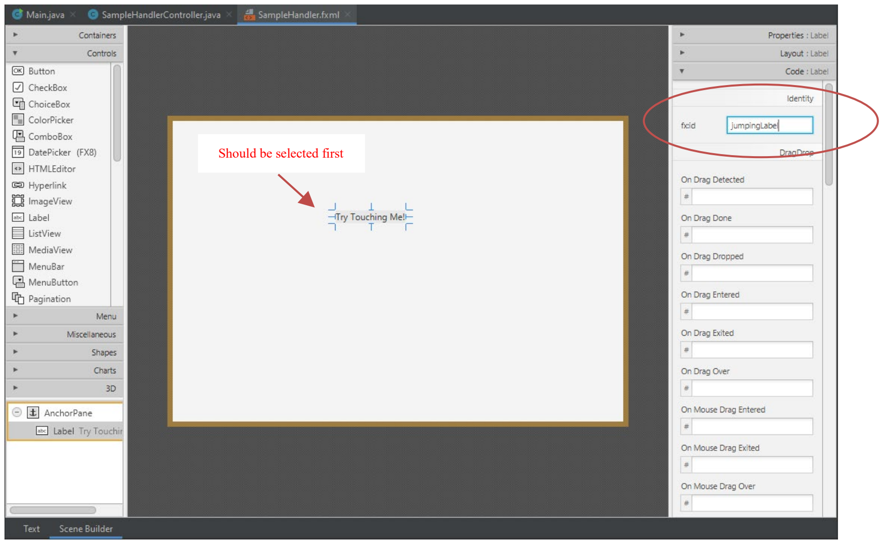


1. In IntelliJ **File** menu, choose **New Project**.
2. Select **JavaFX** project type. Click **Next**.
3. Name the project **Exercise2**, setup the other fields as you did above click Next and then Finish.
4. Using Refactor, change the name of the FXML file to **SampleHandler-view**
5. Using Refactor, change the name of the Controller file to **SampleHandlerController**.
6. Using Refactor, change the name of the Application fille to **SampleHandlerApplication**
7. Open **SampleHandler-view.fxml** in the Scene builder view tab.
8. In the hierarchy panel, Right-Click the "VBox" and select Wrap In → AnchorPane. As shown in the figure.
9. Set the Pref Width and the Pref Height in the Layout of each AchorePane to 700 and 500.



10. Right-Click again the "VBox" and select **Delete**. This will make the AnchorPane container the root scene node.
11. Add (drag and drop) a Label control from the library panel to the generated AnchorPane.
12. In the Inspector Panel Properties, change the name (the Text field) of the label to "**Try Touching Me!**".

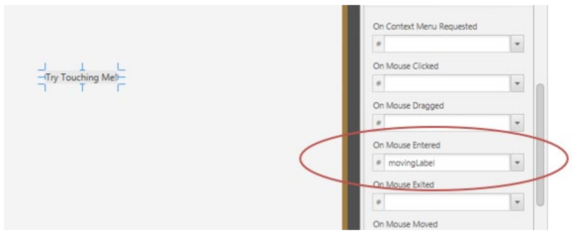
13. In the Inspector Panel Code, set the value of **fxcid** field to **jumpingLabel.** See the Figure below.



14. Now we need to develop the control class and write a handler to response to ***onMouseEntered*** event.
15. Open **SampleHandlerController.java** file and replace the body of the SampleHandlerController class by the following code.

```
    @FXML
    private Label jumpingLabel;

    public void movingLabel(){
        Random randomPos = new Random();
        jumpingLabel.setLayoutX(randomPos.nextInt(500));
        jumpingLabel.setLayoutY(randomPos.nextInt(300));
    }
```

16. Do not forget to import add Java Random library
17. We need now to link the handler method to the FXML scene.
18. Open **SampleHandler-view.fxml** in the Scene builder tab view and click the label control. Set the the onMouseEntered field as shown in the Figure.

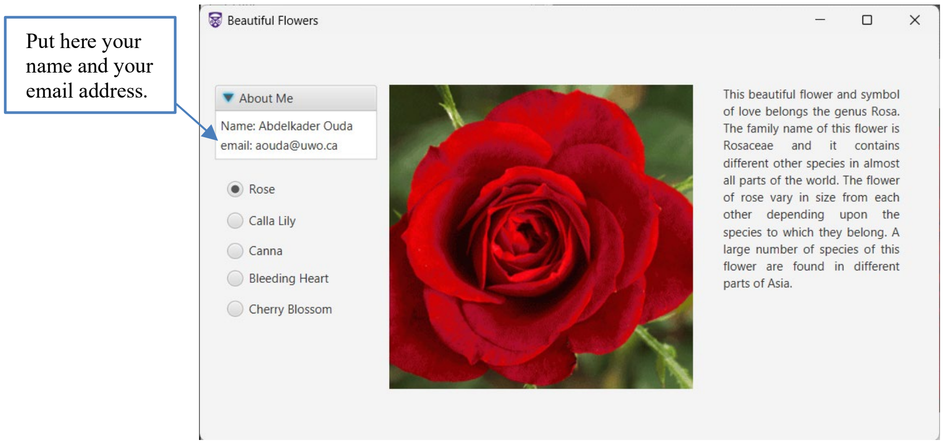


19. Open the SampleHandlerApplication java class, change the stage title to "SampleHandler" and the Scene dimension from 300x275 to 700x500.
20. Run your program and try to click the message and see what will happen!.
21. From the IntelliJ main menu, select File → Export to Zip File…, the Save as pop-up window appears, click OK to save your project as Exercise2.zip file.

- **Notice:** If you are unable to get this exercise run successfully, you should talk to your TAs during their announced office hours (the lab hours).

4.4 Working with more Controls

In this third exercise you will create a standalone GUI application using JavaFX and Scene Builder. The application will display information about the five most bountiful flowers in Canada. The application will display an image of a flower along with one fact note. The image and text are changing depending on which window tab is selected. Your Final GUI should look like this figure. Watch the provided video to see how it works.

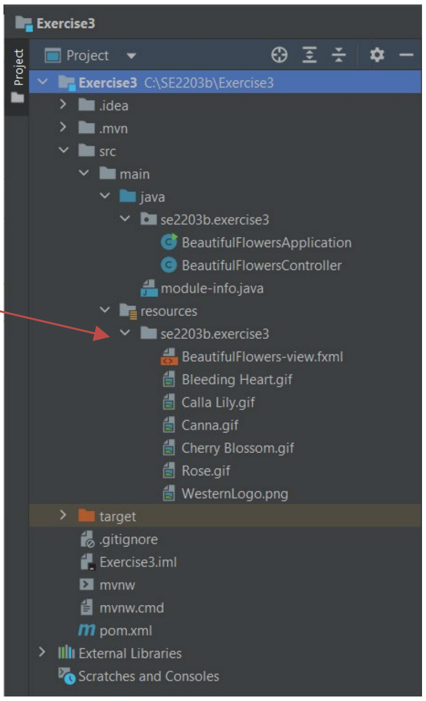


Name: Abdelkader Ouda
email: aouda@uwo.ca

This beautiful flower and symbol of love belongs the genus Rosa. The family name of this flower is Rosaceae and it contains different other species in almost all parts of the world. The flower of rose vary in size from each other depending upon the species to which they belong. A large number of species of this flower are found in different parts of Asia.

4.4.1 Task 1: Creating the GUI's Scene Graph with Scene Builder

1. Using the IntelliJ IDE, create a new JavaFX project and named it **Exercise3**.
2. Using Refactor, change the name of the FXML file to **BeautifulFlowers**-view.
3. Using Refactor, change the name of the Application file to **BeautifulFlowersApplication.**
4. Using Refactor, change the name of the Controller file to **BeautifulFlowersController.**
5. Copy the following files from OWL into the resources folder.
   - Rose.gif
   - Calla Lily.gif
   - Canna.gif
   - Bleeding Heart.gif
   - Cherry Blossom.gif
6. Open **BeautifulFlowers-view.fxml** file and delete its contents.
7. Right-Click the **BeautifulFlowers-view.fxml** file and select Open In SceneBuilder.
8. Add an AnchorPane component to the scene.
9. With the AnchorPane selected,
   - Set the "**Pref Width**" field in the Inspector Panel Layout to **650**
   - Set the "**Pref Height**" field in the Inspector Panel Layout to **400**
   - Set the Controller Class name to "**BeautifulFlowersController**"
10. Add TitledPane container and add two labels in its AnchorPane container to hold your name and email address as shown in the solution video.
11. Add five RadioButton components, arranged in a single column, to the left side of the AnchorPane.
    - Set the Text property of each RadioButton to the following:
      - Rose



  - Calla Lily
  - Canna
  - Bleeding Heart
  - Cherry Blossom
- Set the fx:id field of each RadioButton to the following:
  - roseRadioButton
  - callaLilyRadioButton
  - cannaRadioButton
  - bleedingHeartRadioButton
  - cherryBlossomRadioButton
- To register the event listeners, set the On Action property of each RadioButton to the following:
  - roseRadioButtonListener
  - callaLilyRadioButtonListener
  - cannaRadioButtonListener
  - bleedingHeartRadioButtonListener
  - cherryBlossomRadioButtonListener
- Set the Toggle Group property of each RadioButton to the following:
  - flowersToggleGroup
- Enable the Selected property of the **roseRadioButton** RadioButton component.

12. Add an ImageView component to the right of the five RadioButton components.
    - Resize the component to 300 pixels wide by 300 pixels high.
    - Set the fx:id field to flowersImageView.
    - Set the Image property to Rose.gif
13. Add a Label component, as a single column of width: 140 and hight: 300, to the right of the ImageView component. See the figure above.
    - Set the Text property of this Label to the following:
      "This beautiful flower and symbol of love belongs the genus Rosa. The family name of this flower is Rosaceae and it contains different other species in almost all parts of the world. The flower of rose vary in size from each other depending upon the species to which they belong. A large number of species of this flower are found in different parts of Asia."
    - Set the fx:id field of this Label to the following:
      - flowersNote
14. Make sure the GUI components are arranged to resemble the figure above.
15. Save the changes and close Scene Builder.

4.4.2 Task 2: Editing the Main Application Class

1. Open the BeautifulFlowersApplication.java file.
2. Set the title of the stage as **Beautiful Flowers**.
3. Set the icon of the stage as an image object, the image is "WesternLogo.png" (given within the lab files)
4. Remove the X and Y arguments values from the Scene constructor call.
5. We will edit the logic in the next task.

4.4.3 Task 3: Editing the Controller Class

1. Open the BeautifulFlowersController.java file, and copy the following code template to the controller file:

```java
package se2203b.exercise3;

import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Label;
import javafx.scene.control.RadioButton;
import javafx.scene.control.TitledPane;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import java.net.URL;
import java.util.ResourceBundle;

public class BeautifulFlowersController implements Initializable {
    @FXML
    private ImageView flowersImageView;
    @FXML
    private Label flowersNote;
    @FXML
    private RadioButton roseRadioButton;
    @FXML
    private RadioButton callaLilyRadioButton;
    @FXML
    private RadioButton cannaRadioButton;
    @FXML
    private RadioButton bleedingHeartRadioButton;
    @FXML
    private RadioButton cherryBlossomRadioButton;
    @FXML
    private TitledPane aboutMe;

    // ADD LINES FOR TASK #3 HERE
    // Declare private fields for the images

    // Initialize method
    @Override
    public void initialize(URL url, ResourceBundle rb) {

      // ADD LINES FOR TASK #3 HERE
      // Load the image files in the intialize method

    }
    public void roseRadioButtonListener() {

      // ADD THE LINES FOR TASK #3 HERE
      // If this radio button is selected,
      // display image and data for Rose.

    }

    // Repeat and modify the above handler for the rest of the other flowers.

}
```

2. Edit the BeautifulFlowersController class so that it will load the images when the application starts.
   - Declare private fields for each of the five images.
   - Edit the initialize method to load the images.
3. Edit the event listener method for each radio button so it will display the appropriate data when selected.
   - Use the information provided in the table below.
4. Save and compile the controller class, correcting any errors.

Image	Note
Rose	This beautiful flower and symbol of love belongs the genus Rosa. The family name of this flower is Rosaceae and it contains different other species in almost all parts of the world. The flower of rose vary in size from each other depending upon the species to which they belong. A large number of species of this flower are found in different parts of Asia.
Calla Lily	One simple and common name of this beautiful flower is arum lily and this flowering plant belongs to the family known as Araceae. This flowering plant is grown well in areas which have reasonable rainfall and moderate temperatures.
Canna	This beautiful flowering plant belongs to a genus that contain around 10 species and its family is known as Cannaceae. This flower plant also provides large quantity of starch which is further used for different purposes. This flower is mostly found in tropical regions of the world. The flowers of this plant have three sepals and three petals.
Bleeding Heart	The bleeding heart flower belongs to the family known as Papaveraceae. This heart shaped flower is famous all over the world due to its unique beauty and is found in great numbers in Siberia and China. Blooming season of this flower starts in spring when there spread beautiful pink heart-shaped flowers in gardens. Lady-in-a-bath is also a common name of this flower.
Cherry Blossom	Cherry blossom, a beautiful flowering plant is found in different parts of the world including the Northern Hemisphere. It is found in those areas which have moderate climate. All species of this flowering plant do not produce cherries as there are special species of this flower that produce edible cherries.

4.4.4 Task 4: Run, Test, and then complete your application

1. Run the application. Test the window tabs and determine that the output is correct as shown in the solution video.
2. Once you finished, select File → Export to Zip File…, the Save as pop-up window appears, click OK to save your project as Exercise3.zip file.

5 Hand In

- Group the following three Zip files into one Zip file and name it *yourUwoId*_ Lab1.zip. For example, if your email address is aouda@uwo.ca, then name the archive file as aouda_Lab2.zip.
  - Exercise1.zip, Exercise2.zip, and Exercise3.zip
- Submit your final zip file through OWL on the due date mentioned above, to be graded out of 20.