MIS 768: Advanced Software Concepts

Spring 2024

GUI Application (2)

Purpose

- Use Scene Builder to create FXML file
- Connect FXML file to a JavaFX application
- Learn to use various controls in JavaFX application

1. Preparation

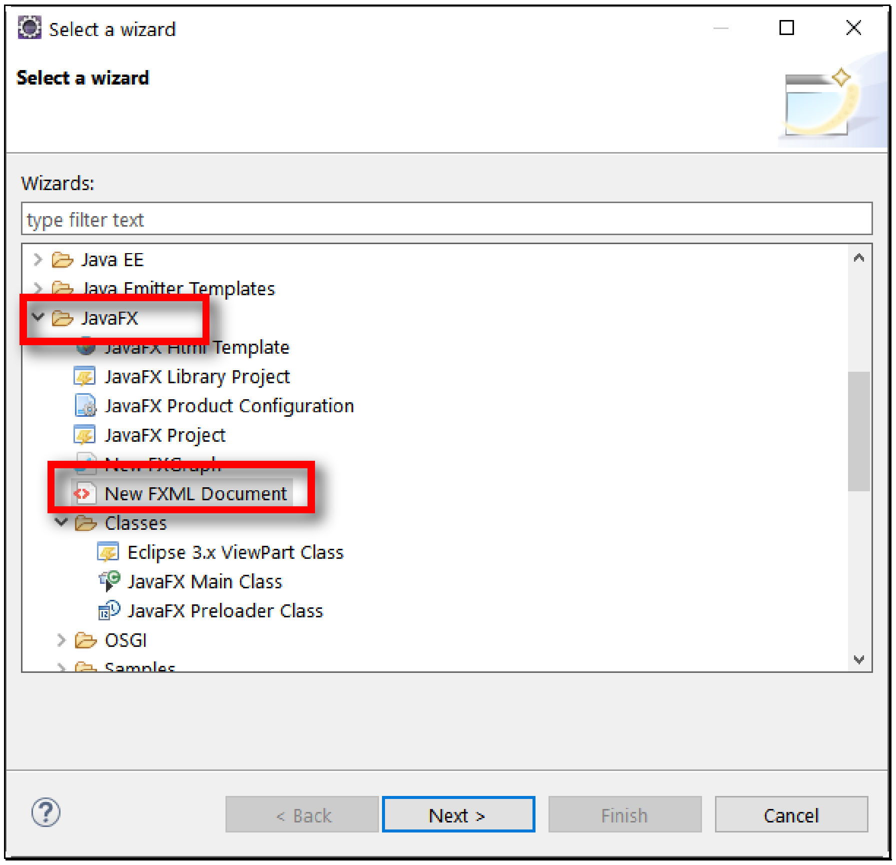
(1) Launch Eclipse. Create a new package to hold our source file. Name the package as **edu.unlv.mis768.labwork15.**

(2) Download **15_lab_files.zip** from WebCampus. Extract the zip file and then import the .java files into the package.

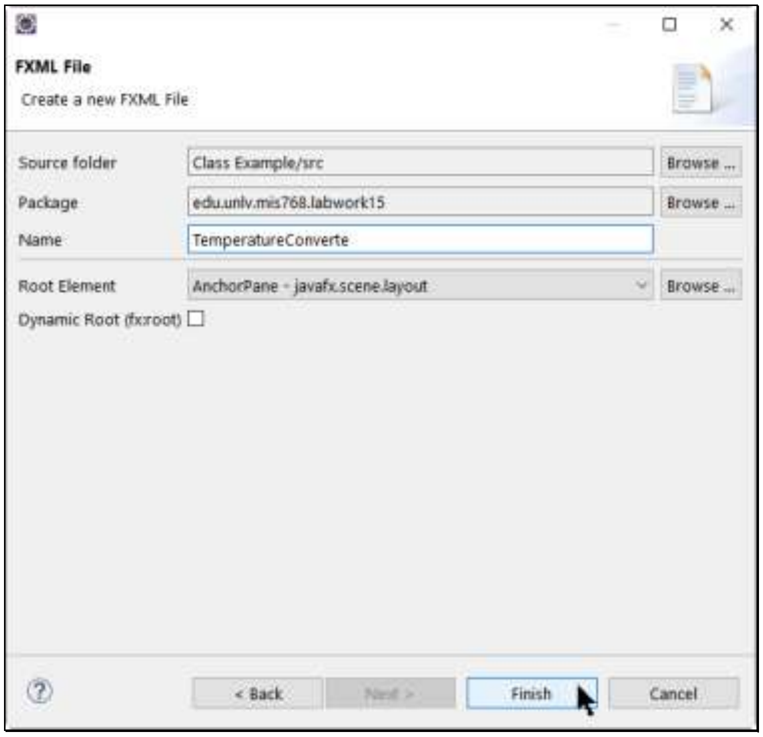
2. Create GUI using Scene Builder

(3) Right click at the package, and select **New \ Other**.

Select **JavaFX \ New FXML Document**



(4) Name the new FXML file as **TemperatureConverter**



FXML File

Create a new FXML File

Source folder: Class Example/src

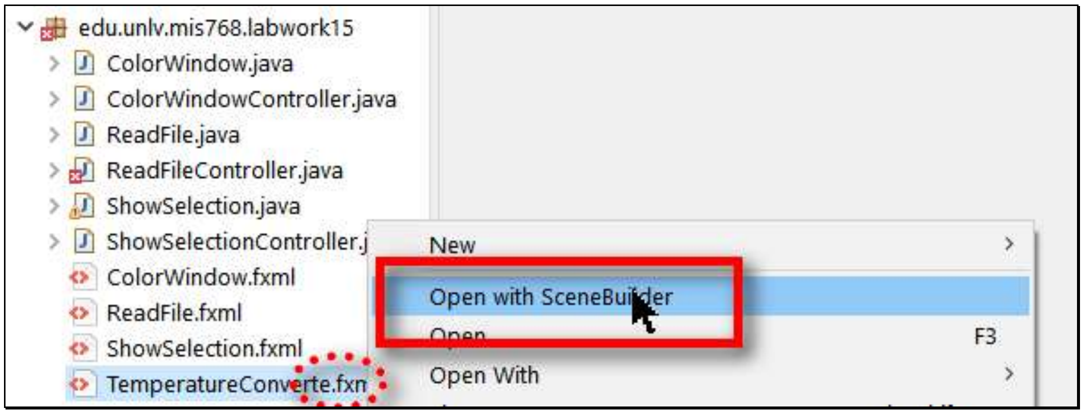
Package: edu.unlv.mis768.labwork15

Name: TemperatureConverte

Root Element: AnchorPane - javafx.scene.layout

Dynamic Root (fx:root): ☐

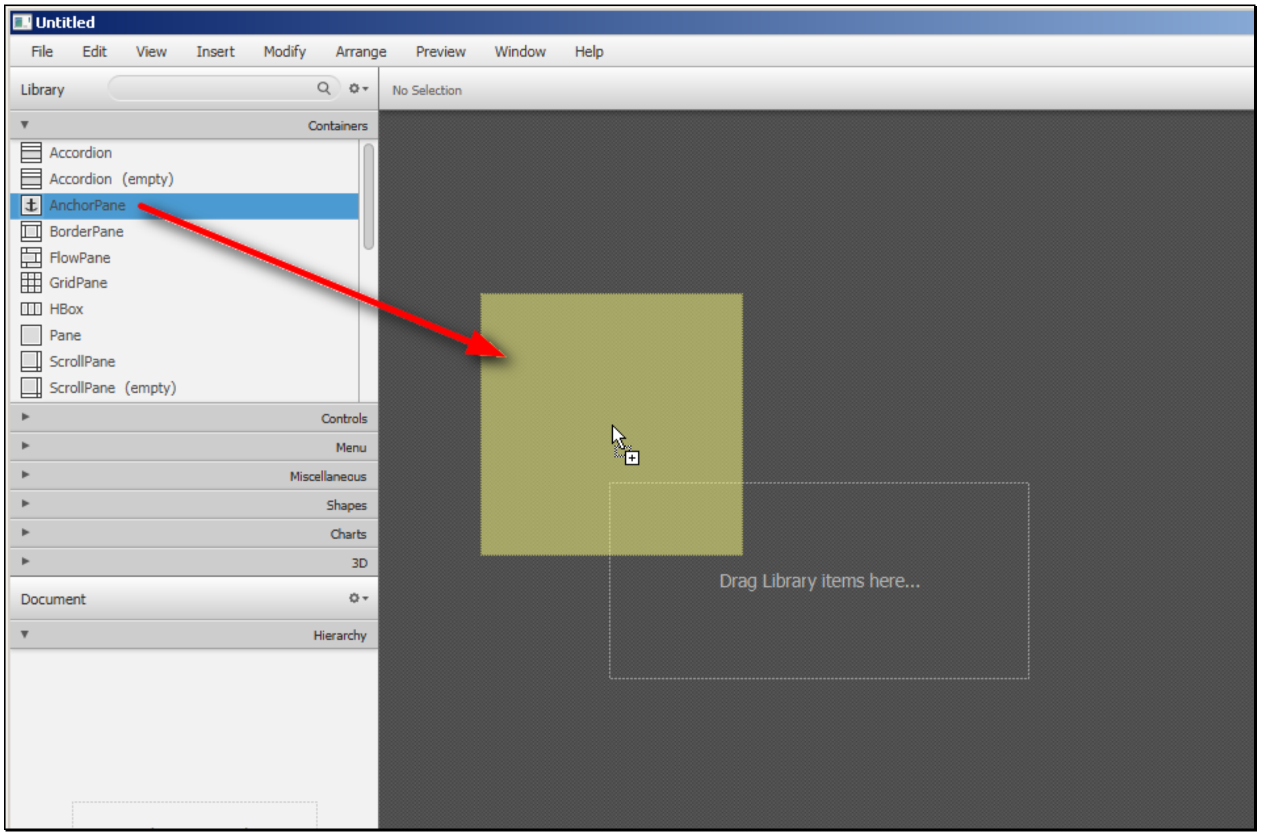
(5) Right click on the file and select Open with SceneBuilder



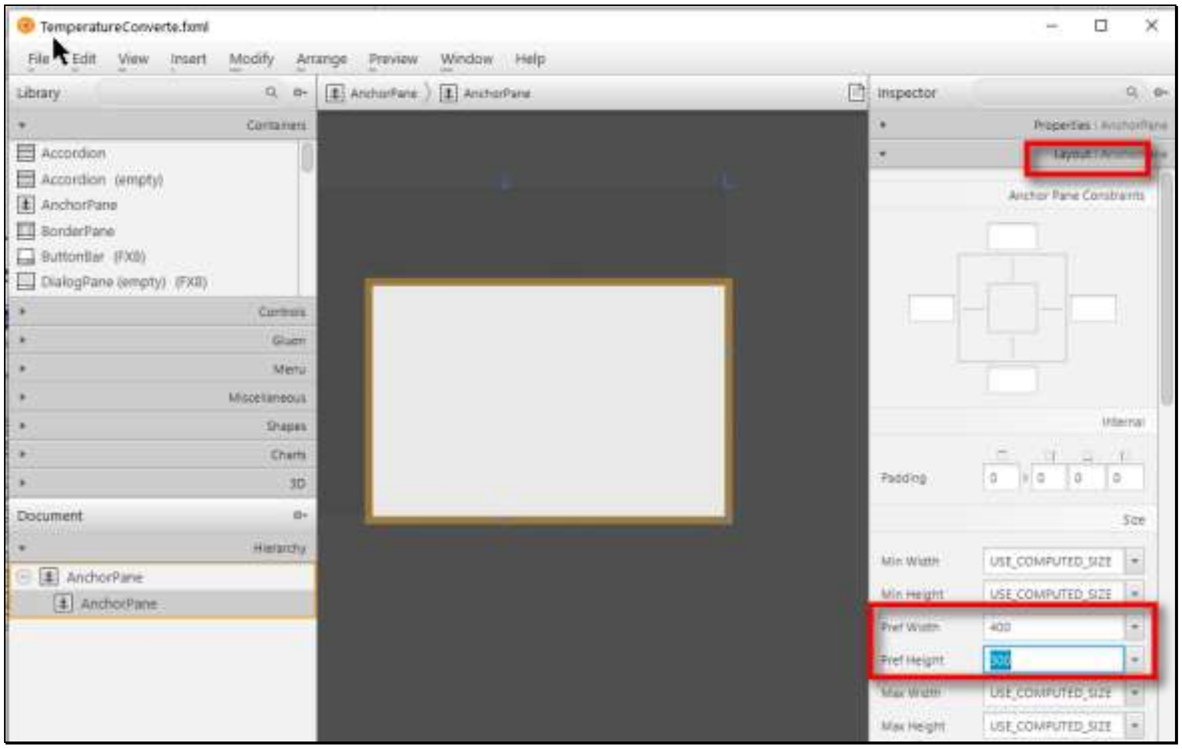
edu.unlv.mis768.labwork15
- ColorWindow.java
- ColorWindowController.java
- ReadFile.java
- ReadFileController.java
- ShowSelection.java
- ShowSelectionController.j
- ColorWindow.fxml
- ReadFile.fxml
- ShowSelection.fxml
- TemperatureConverte.fxn

(6) Start **Scene Builder**.

Drag an **AnchorPane** component from the **Containers** section of the **Library** panel, and drop it into the **Content** panel.

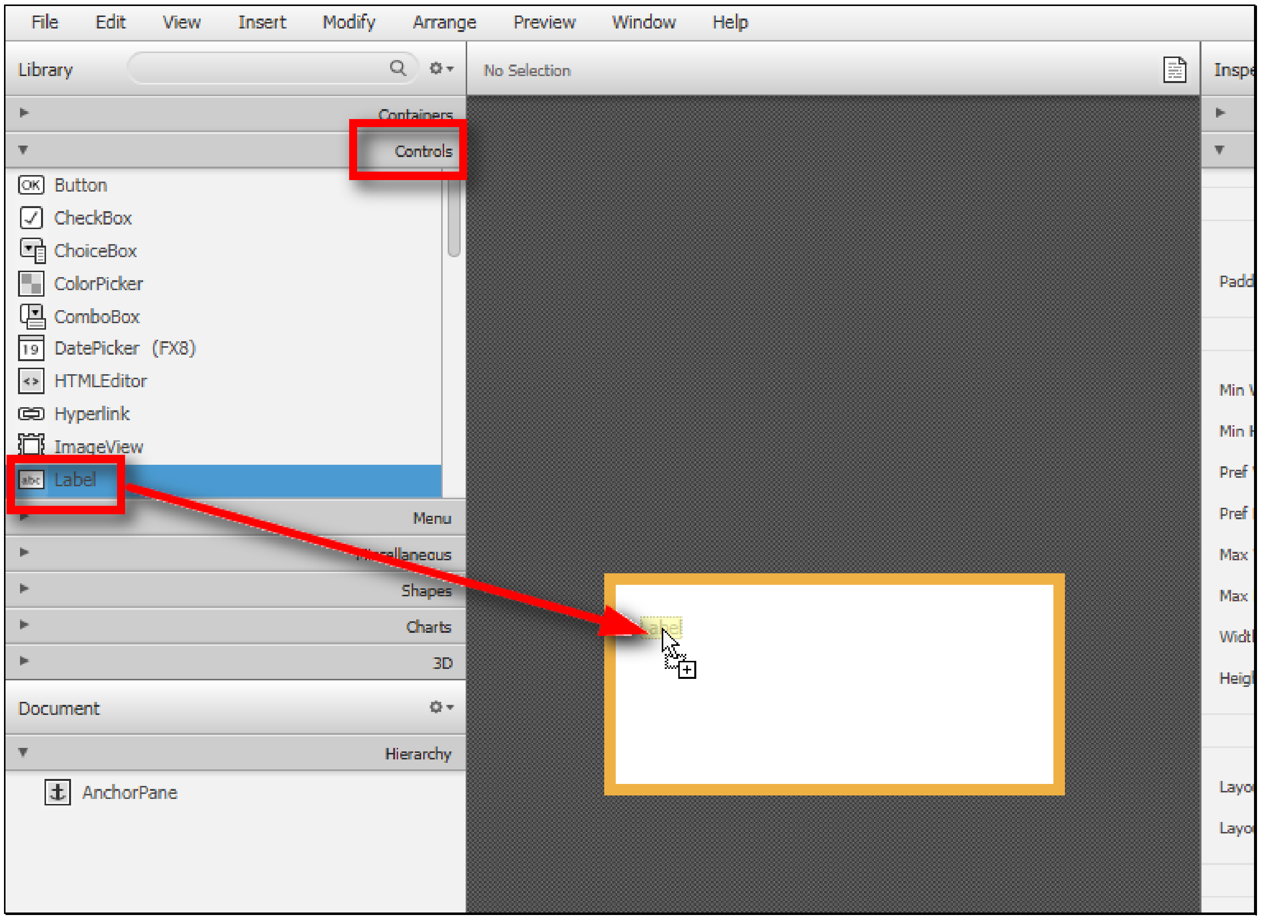


(7) Resize the **AnchorPane** to make it 400 by 300 in the size.

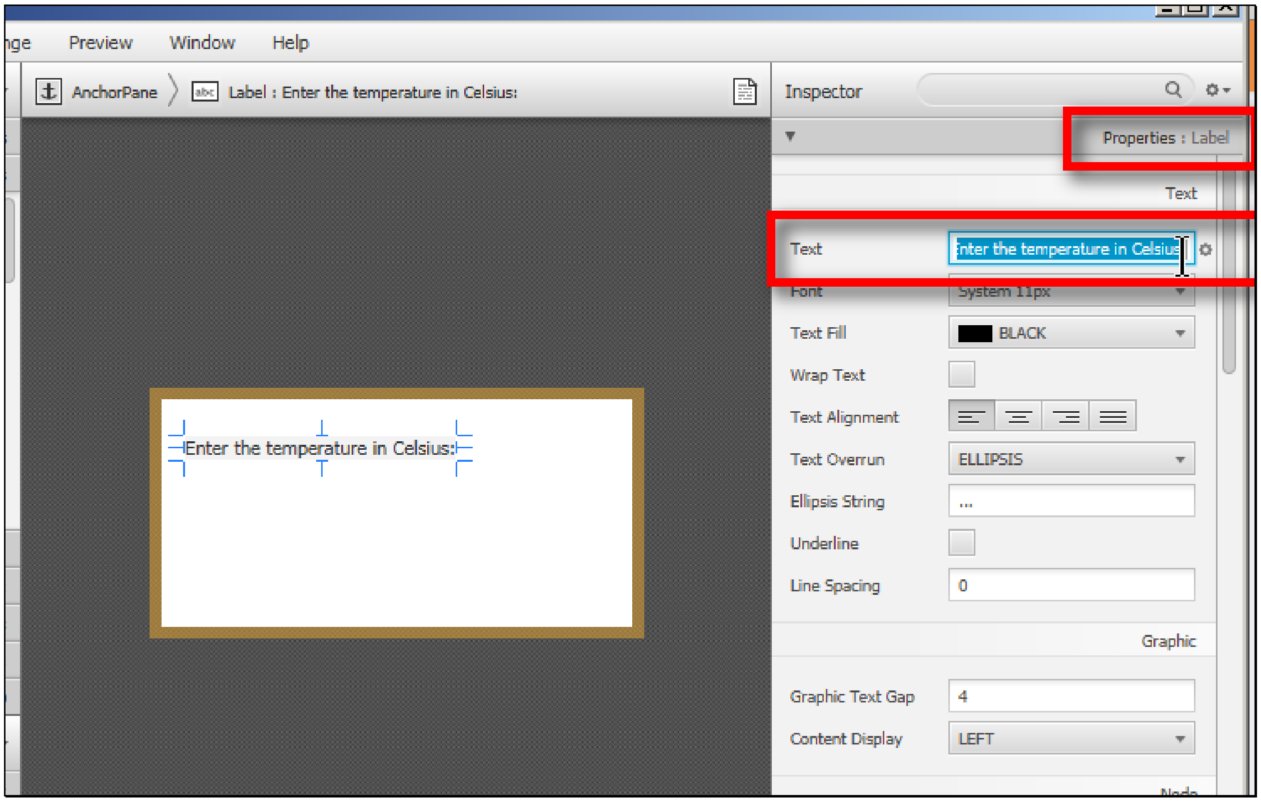


Padding: 0 0 0 0
Min Width: USE_COMPUTED_SIZE
Min Height: USE_COMPUTED_SIZE
Pref Width: 400
Pref Height: 300
Max Width: USE_COMPUTED_SIZE
Max Height: USE_COMPUTED_SIZE

(8) Now add the **Label** component to the **AnchorPane**.



(9) The text that is displayed by a **Label** component is determined by the Label's **Text** property. Change it to **Enter the temperature in Celsius**.


Preview
Window
Help
AnchorPane
Label : Enter the temperature in Celsius:
Inspector
Properties : Label
Enter the temperature in Celsius:

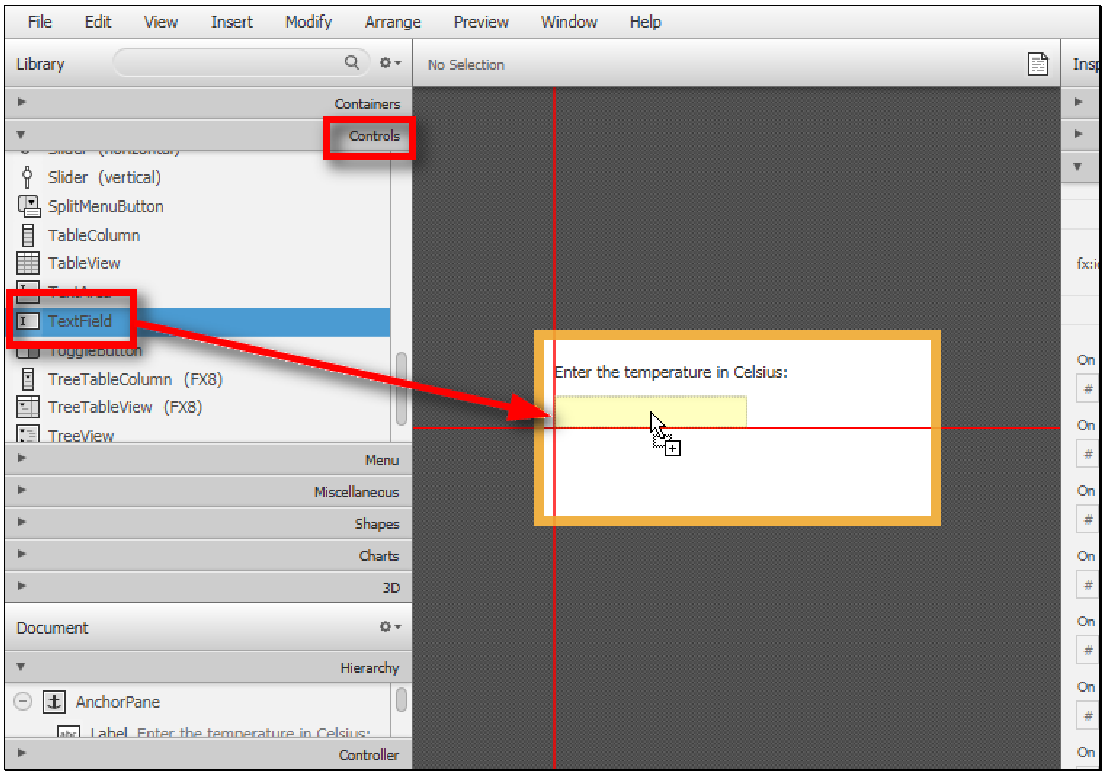
Text

Text	Enter the temperature in Celsius
Font	System 11px
Text Fill	BLACK
Wrap Text	☐
Text Alignment	
Text Overrun	ELLIPSIS
Ellipsis String	...
Underline	☐
Line Spacing	0

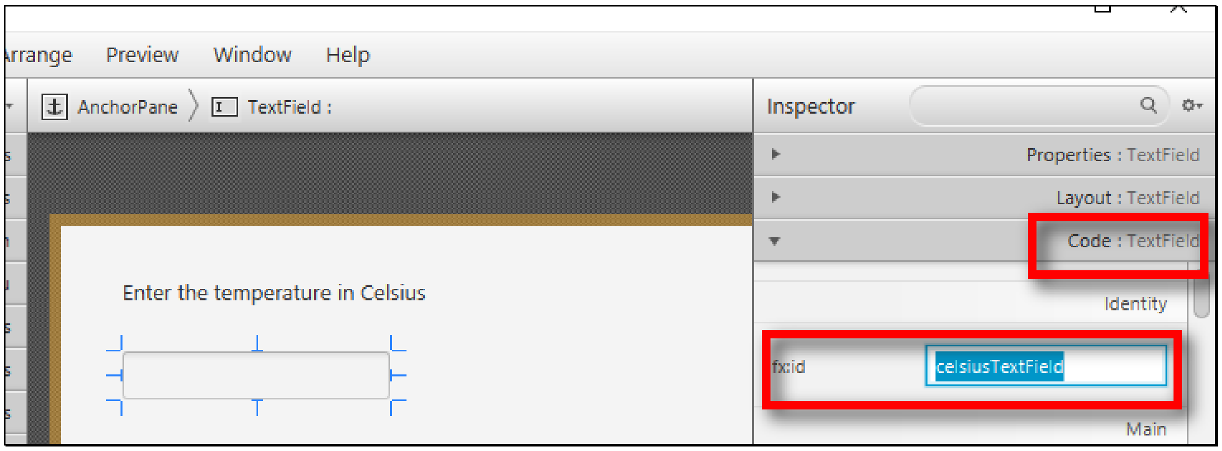
Graphic

Graphic Text Gap	4
Content Display	LEFT

(10) Next, please add a **TextField** component to the **AnchorPane**.

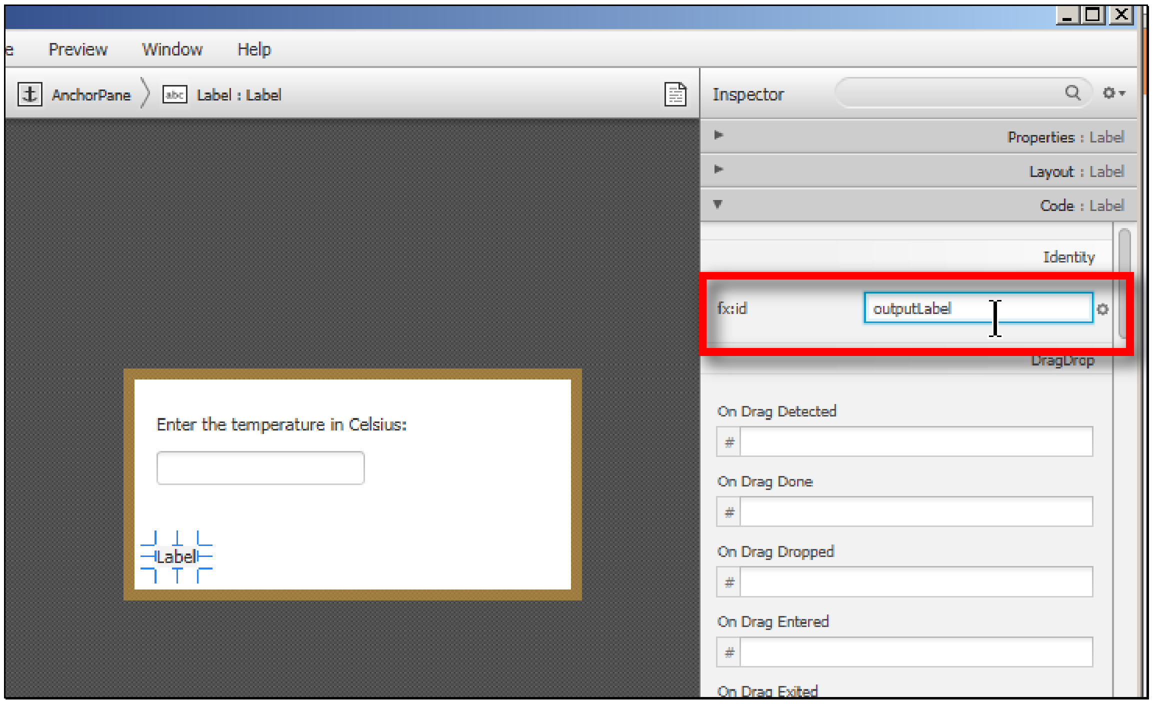


(11) Set the **fx:id** as **celsiusTextField**. **fx:id** is the name that identifies a component in the FXML file.



(12) Next one is a label for displaying the result. Please set the **fx:id** as **outputLabel**.

Set the **Text** property to a null string.



fx:id outputLabel

Enter the temperature in Celsius:

On Drag Detected #

On Drag Done #

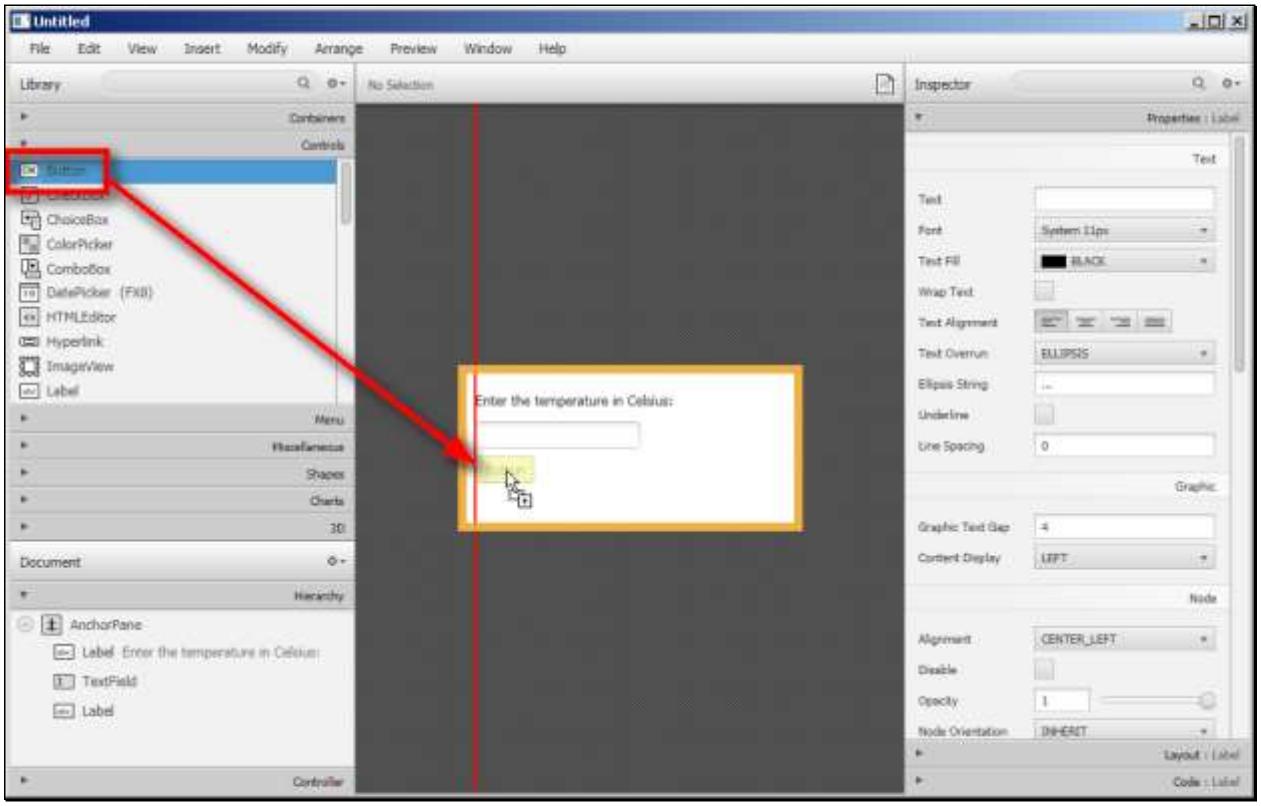
On Drag Dropped #

On Drag Entered #

On Drag Exited

(13) Finally, a **Button** component needs to be added to the **AnchorPane**, for executing the conversion.

Set the **Text** property to **Convert to Fahrenheit** and the **fx:id** as **convertButton**.



Enter the temperature in Celsius:

Text

Font System 11px

Text Fill BLACK

Wrap Text

Text Alignment

Text Overrun ELLIPSIS

Ellipsis String ...

Underline

Line Spacing 0

Graphic Text Gap 4

Content Display LEFT

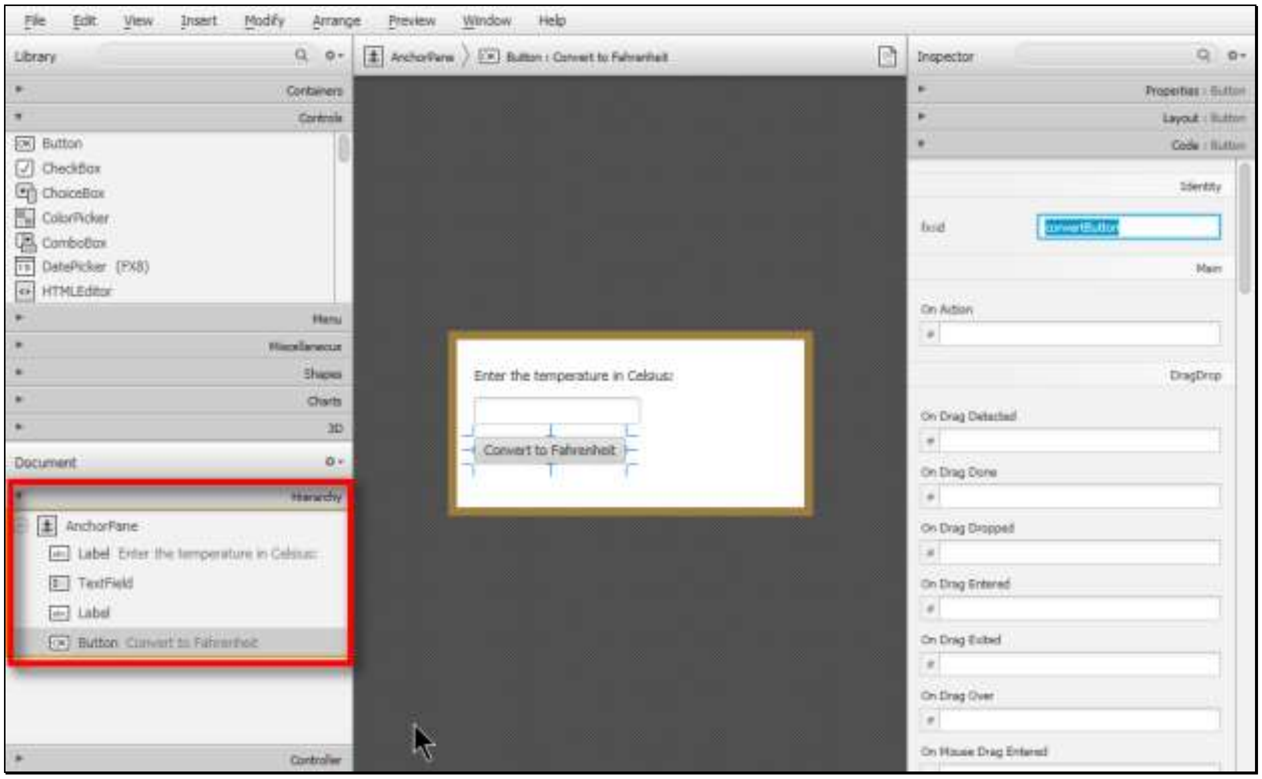
Alignment CENTER_LEFT

Disable

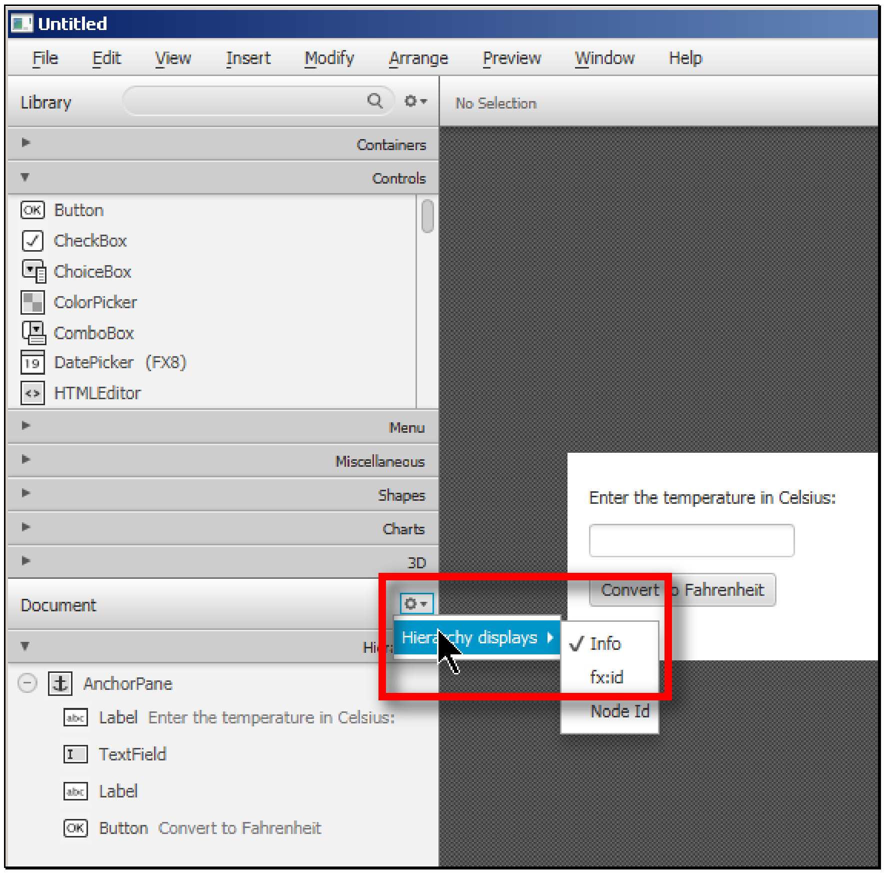
Opacity 1

Node Orientation INHERIT

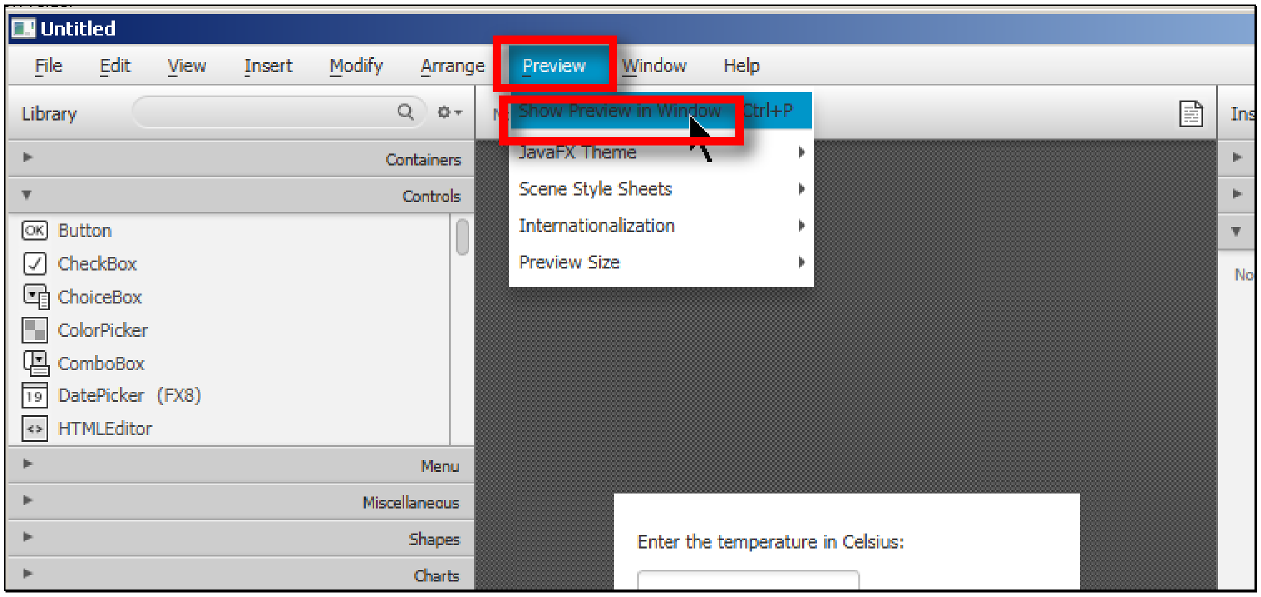
(14) Now the **Hierarchy** panel shows the component we added to the **Content** panel.



(15) You can set the **Hierarchy** panel to display the fx:id, by setting the **Hierarchy display** to **fx:id.**



(16) You can also preview the GUI. Go to **Preview** on the menu bar, and then select **Show Preview in Window**.

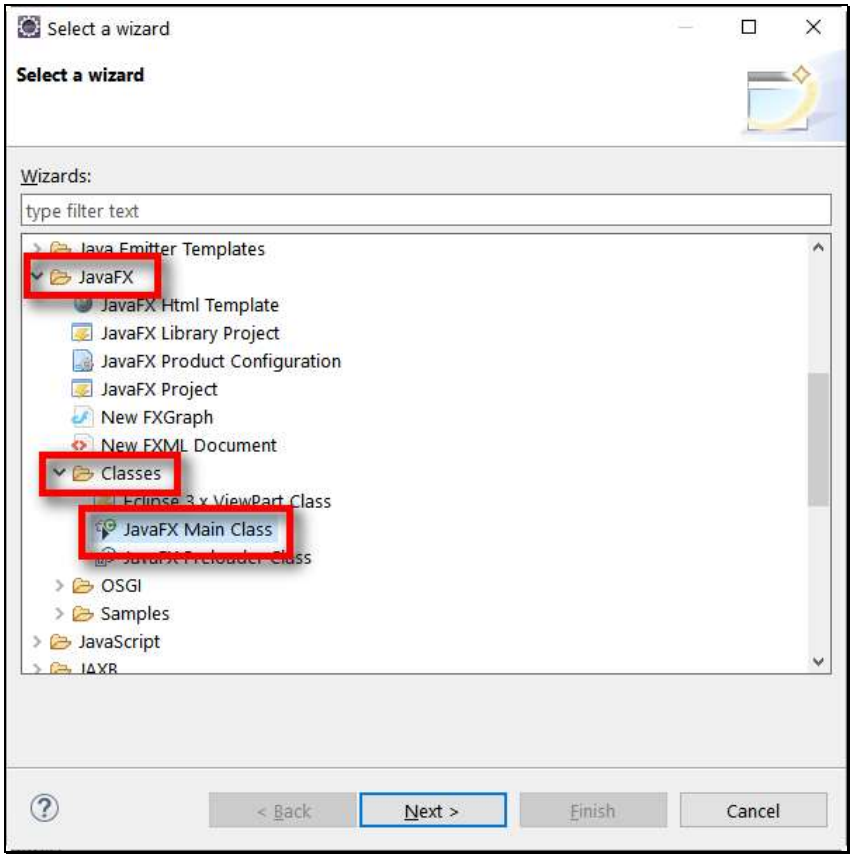


(17) Save the file in the **ScenceBuilder**.

3. Application Code

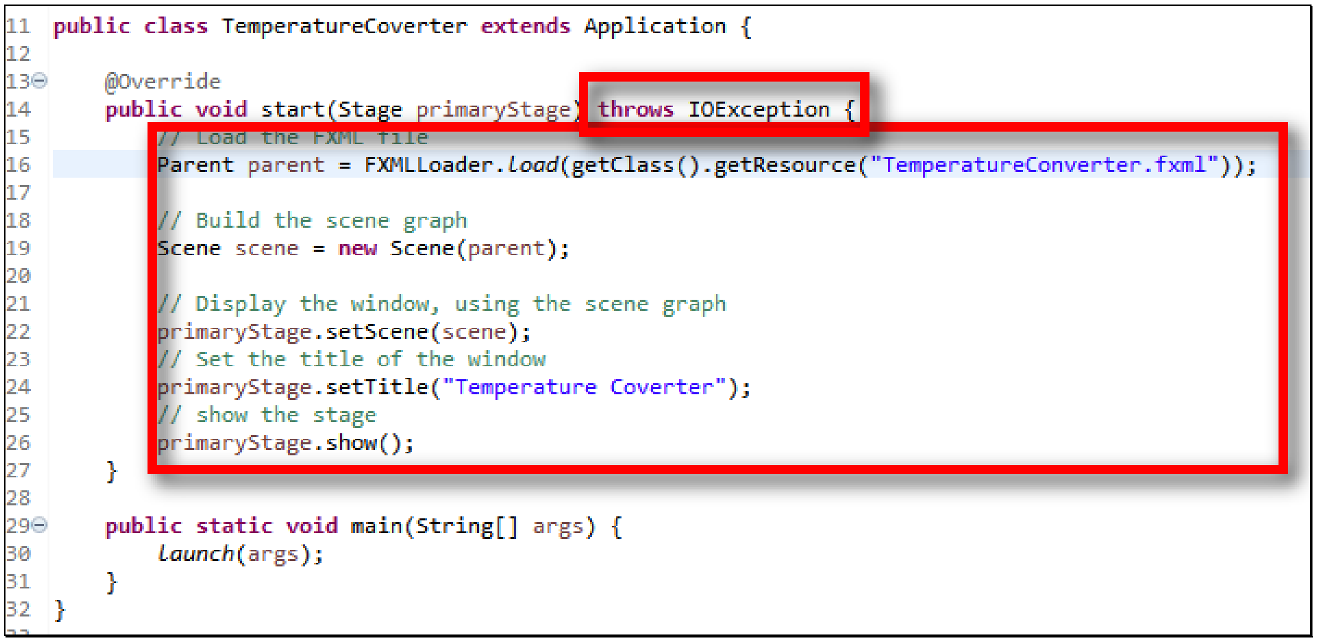
(18) Under the same package, right click and select **New \ Other**

Select **JavaFX \ Classes \ JavaFX Main Class**



(19) Name the new class as **TemperatureCoverter**

(20) Enter the following code in the **start()** Method.



```
public class TemperatureCoverter extends Application {

    @Override
    public void start(Stage primaryStage) throws IOException {
        // Load the FXML file
        Parent parent = FXMLLoader.load(getClass().getResource("TemperatureConverter.fxml"));

        // Build the scene graph
        Scene scene = new Scene(parent);

        // Display the window, using the scene graph
        primaryStage.setScene(scene);
        // Set the title of the window
        primaryStage.setTitle("Temperature Coverter");
        // show the stage
        primaryStage.show();
    }

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

(21) Now if you run **TemperatureCoverter.java**, you can see the GUI application showing up, but it does nothing when you click the Convert button.

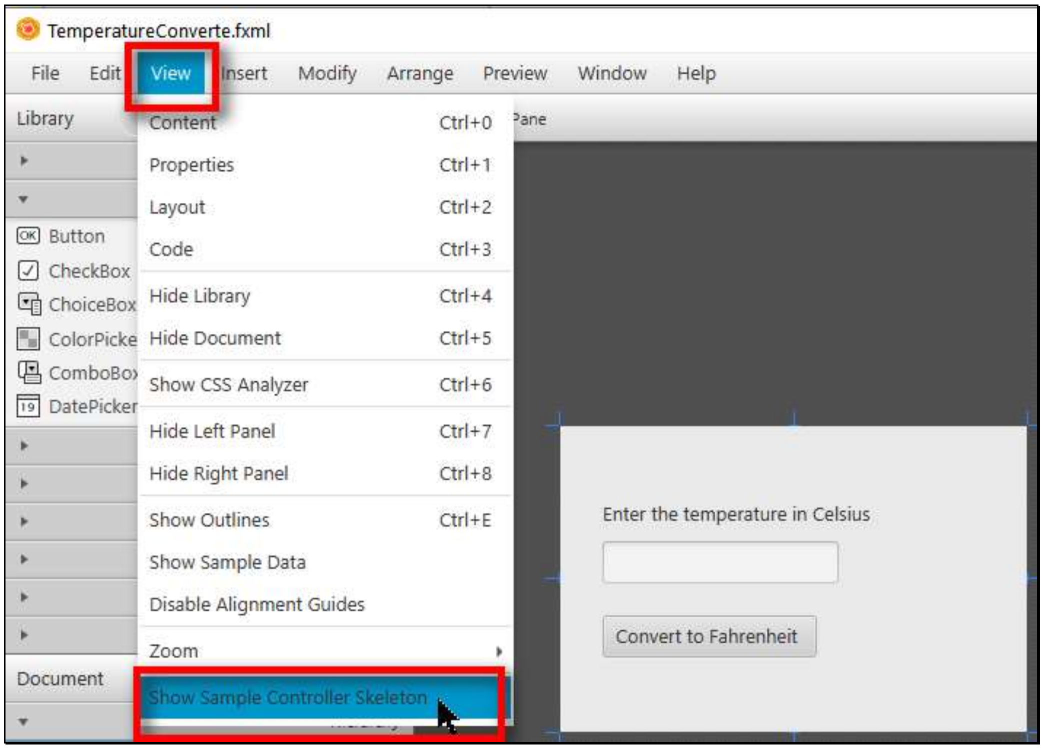
Note: Please do not forget to set the VM arguments in **Run Configurations.**

4. Create the Controller Class

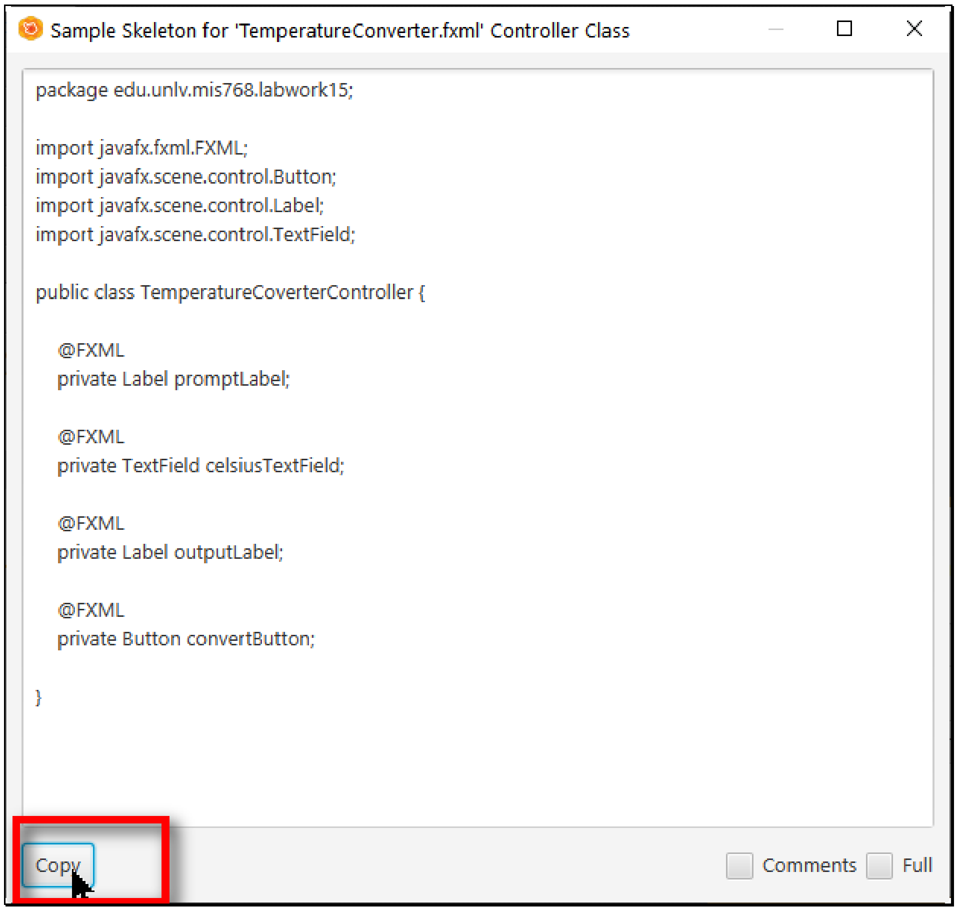
(22) Create a normal class in the same package and name it as **TemperatureCoverterController.**

(23) Switch to **Scene Builder**.

Under the **View** menu, select **Show Sample Controller Skeleton**.

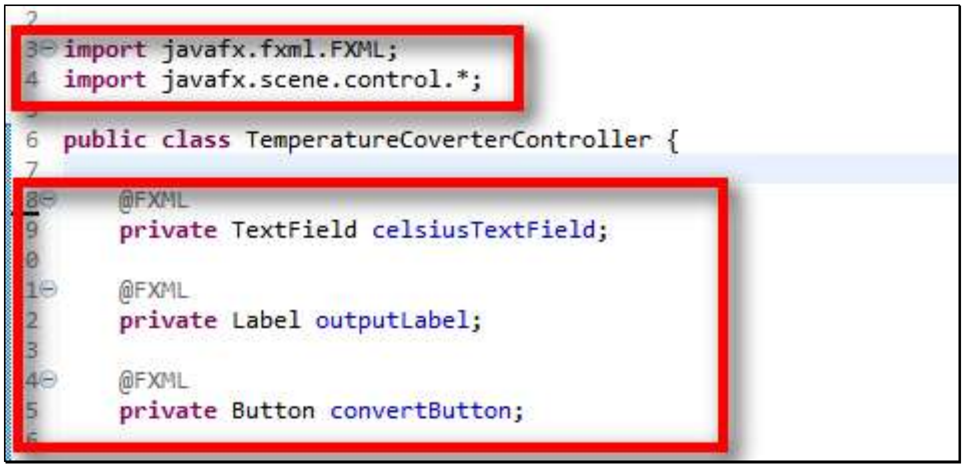


(24) Copy the code in the **Sample Skeleton**.


Sample Skeleton for 'TemperatureConverter.fxml' Controller Class

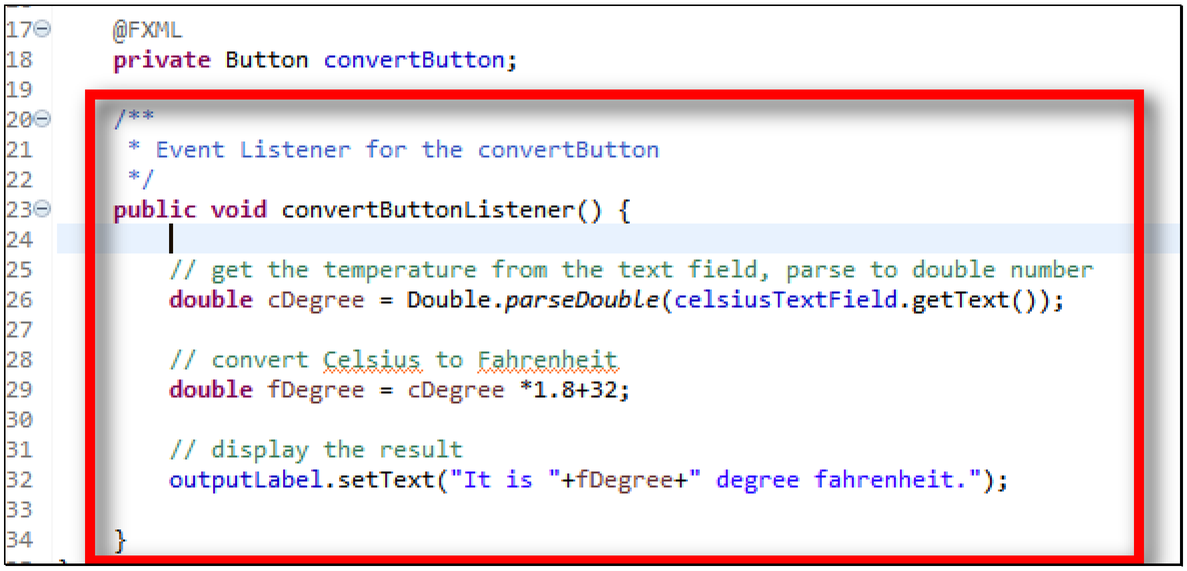
```
package edu.unlv.mis768.labwork15;

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

public class TemperatureCoverterController {

    @FXML
    private Label promptLabel;

    @FXML
    private TextField celsiusTextField;

    @FXML
    private Label outputLabel;

    @FXML
    private Button convertButton;

}
```

(25) Switch back to **Eclipse** and paste the code in **TemperatureCoverterController**



```
import javafx.fxml.FXML;
import javafx.scene.control.*;

public class TemperatureCoverterController {

    @FXML
    private TextField celsiusTextField;

    @FXML
    private Label outputLabel;

    @FXML
    private Button convertButton;
```

(26) Please complete the program as shown below

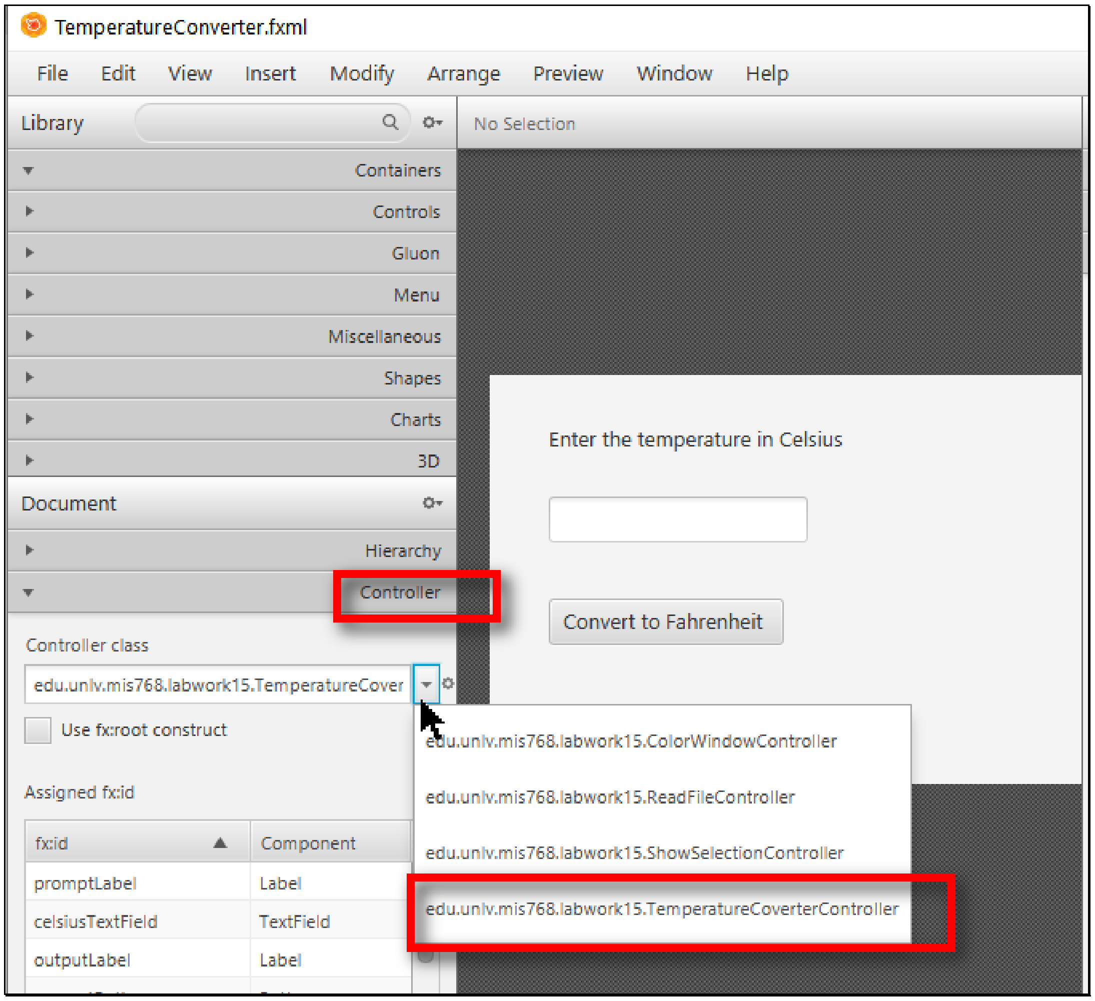


```
    @FXML
    private Button convertButton;

    /**
     * Event Listener for the convertButton
     */
    public void convertButtonListener() {
        
        // get the temperature from the text field, parse to double number
        double cDegree = Double.parseDouble(celsiusTextField.getText());

        // convert Celsius to Fahrenheit
        double fDegree = cDegree *1.8+32;

        // display the result
        outputLabel.setText("It is "+fDegree+" degree fahrenheit.");

    }
```

(27) If you now run the application, the button still would not work.

5. Registering the Controller Class with the GUI.

(28) Switch to **Scene Builder**. Open the **Controller** section of the **Document** panel. Select the controller class.



Controller class: edu.unlv.mis768.labwork15.TemperatureCover

Use fx:root construct

edu.unlv.mis768.labwork15.ColorWindowController
edu.unlv.mis768.labwork15.ReadFileController
edu.unlv.mis768.labwork15.ShowSelectionController
edu.unlv.mis768.labwork15.TemperatureCoverterController

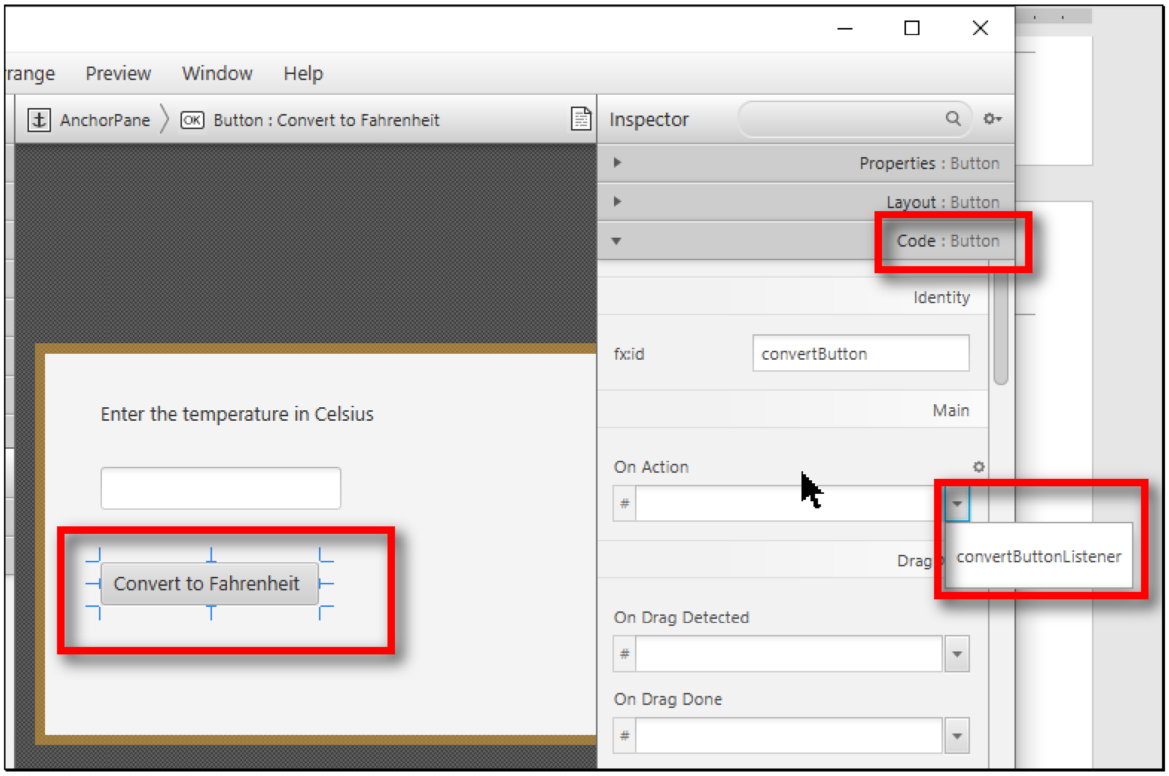
Assigned fx:id

fx:id	Component
promptLabel	Label
celsiusTextField	TextField
outputLabel	Label

(29) Next, we need to connect the event listener to the button.

Select the button, then open the **Code** panel on the right-hand side.

Under **On Action**, use the drop down list to select **convertButtonListener**.

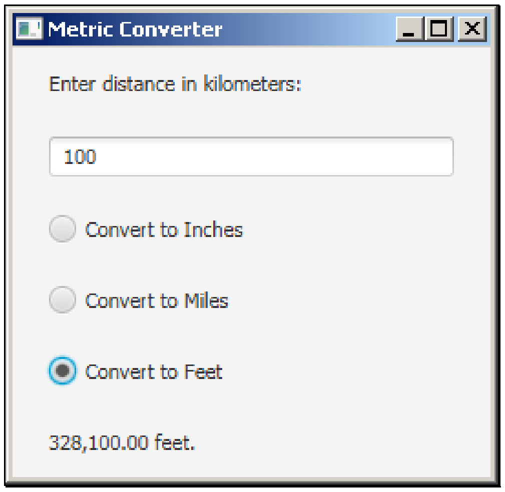


(30) You can save the file and run **TemperatureCoverter.java** to execute the program.

6. Radio Button in JavaFX

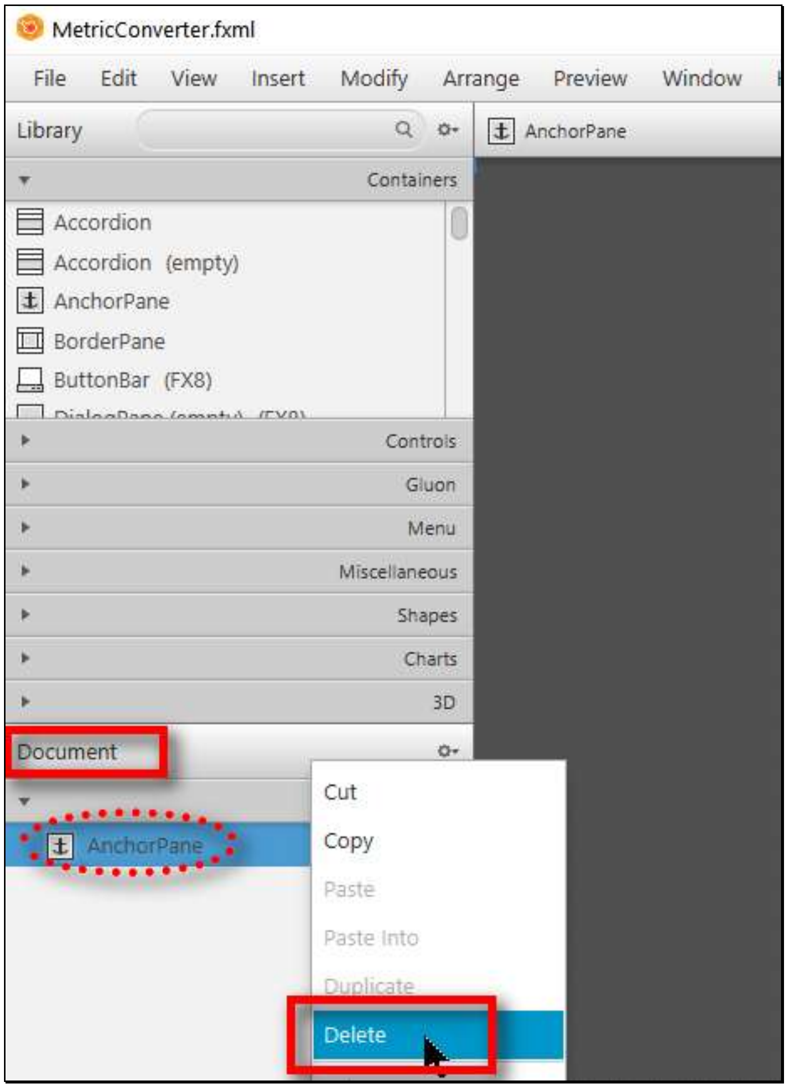
(31) In this example, we will create an application that converts a distance in kilometer to different units.

Upon the user clicks on a radio button, the program executes the conversion.



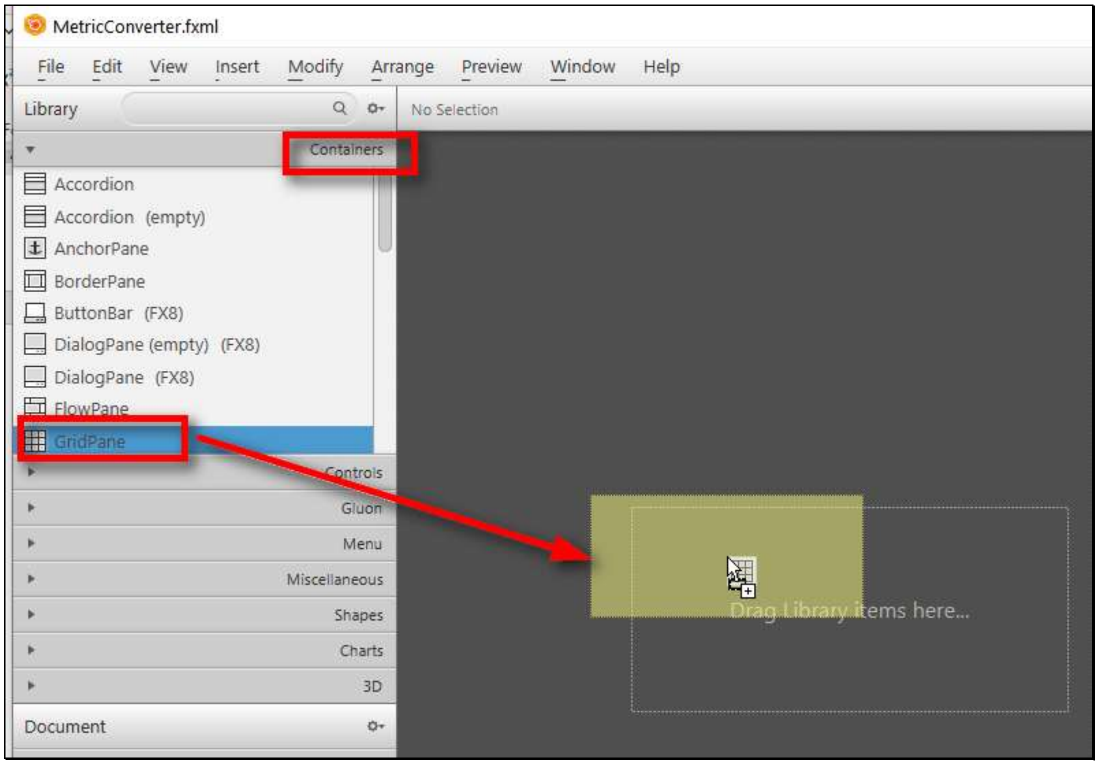
(32) Create a new FXML file, name it as **MetricConverter.fxml**. Open it in Scene Builder.

(33) In the **Document** tab, select the default **Anchor Pane**, and delete it.

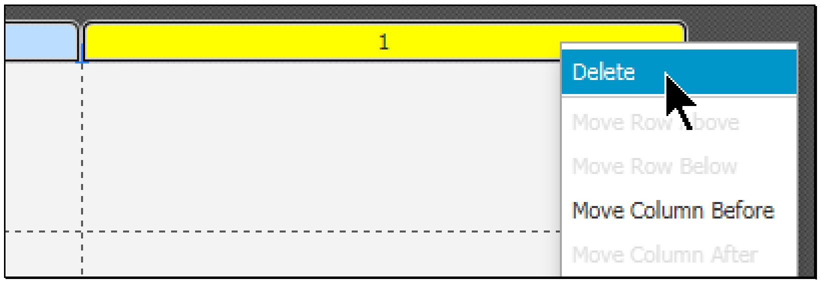


(34) From **Containers**, select **GridPane** and drag it to the design canvas.

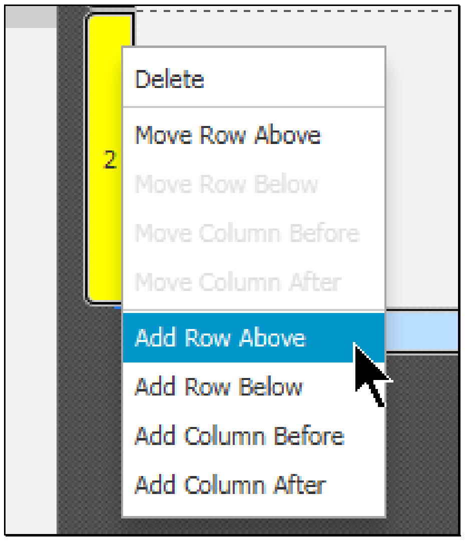
By doing so, we add a **GridPane** as the root node



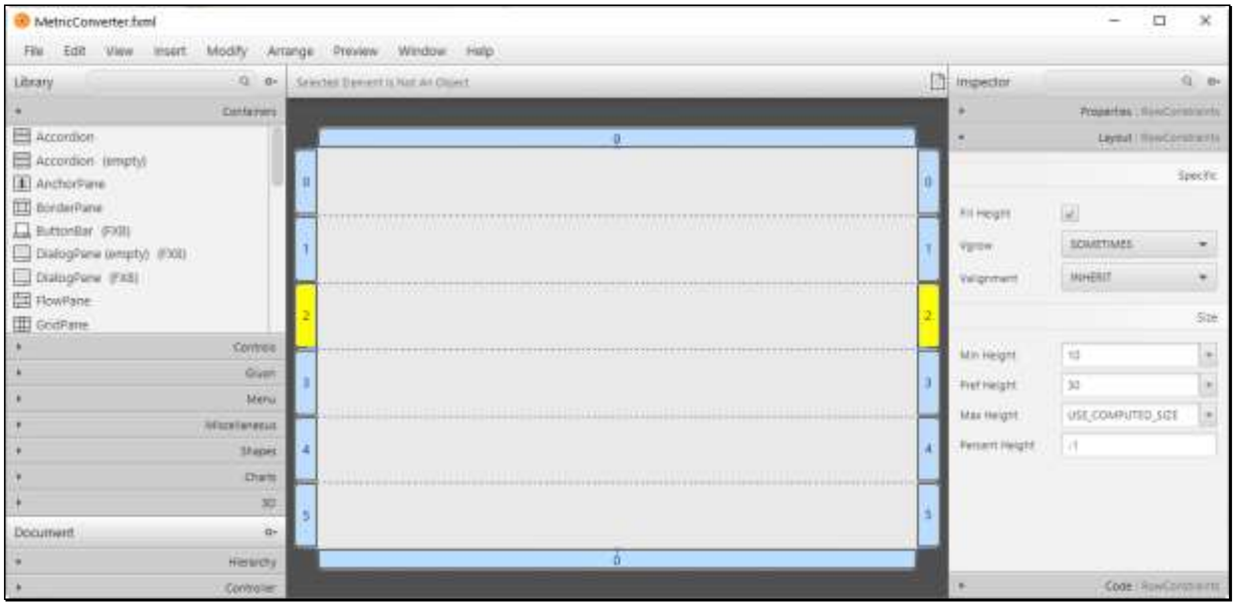
(35) Select **Column 1** and then right click, select **Delete**.



(36) Select **Row 2**, right click, and then select **Add Row Above**.



(37) Repeat the above step to get a **1 column and 6 rows** table.



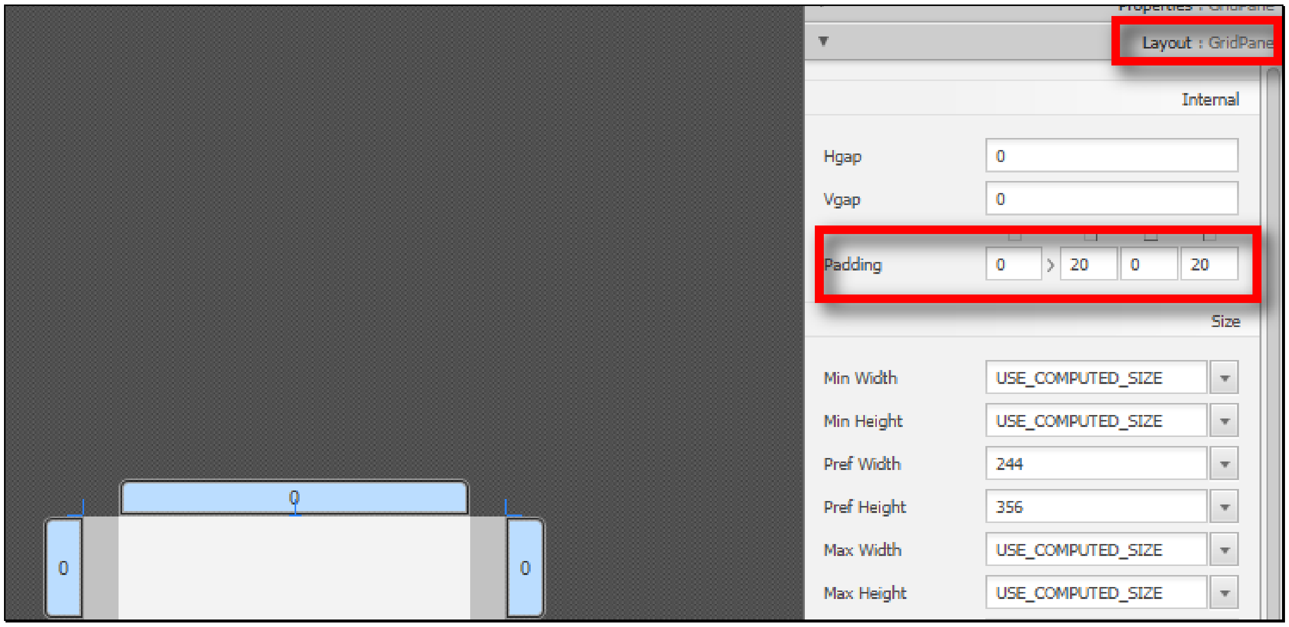
Specific

Fill Height	✓
Vgrow	SOMETIMES
Valignment	INHERIT

Size

Min Height	10
Pref Height	30
Max Height	USE_COMPUTED_SIZE
Percent Height	-1

(38) Set the padding of right and left to **20**



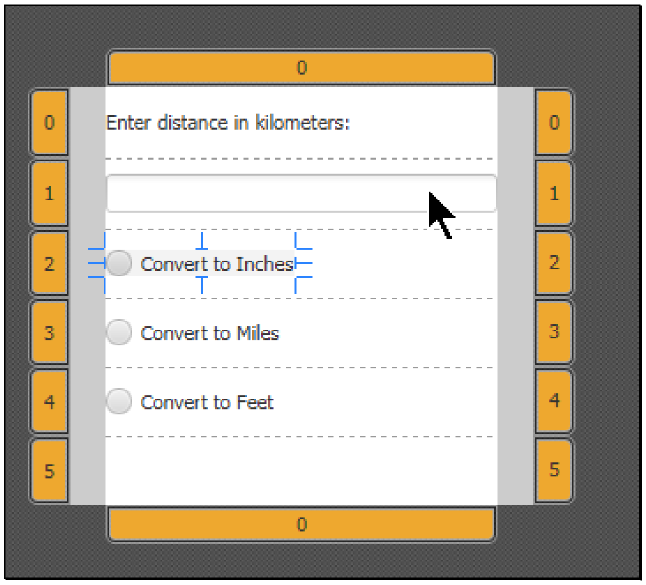
Internal

Hgap	0			
Vgap	0			
Padding	0	20	0	20

Size

Min Width	USE_COMPUTED_SIZE
Min Height	USE_COMPUTED_SIZE
Pref Width	244
Pref Height	356
Max Width	USE_COMPUTED_SIZE
Max Height	USE_COMPUTED_SIZE

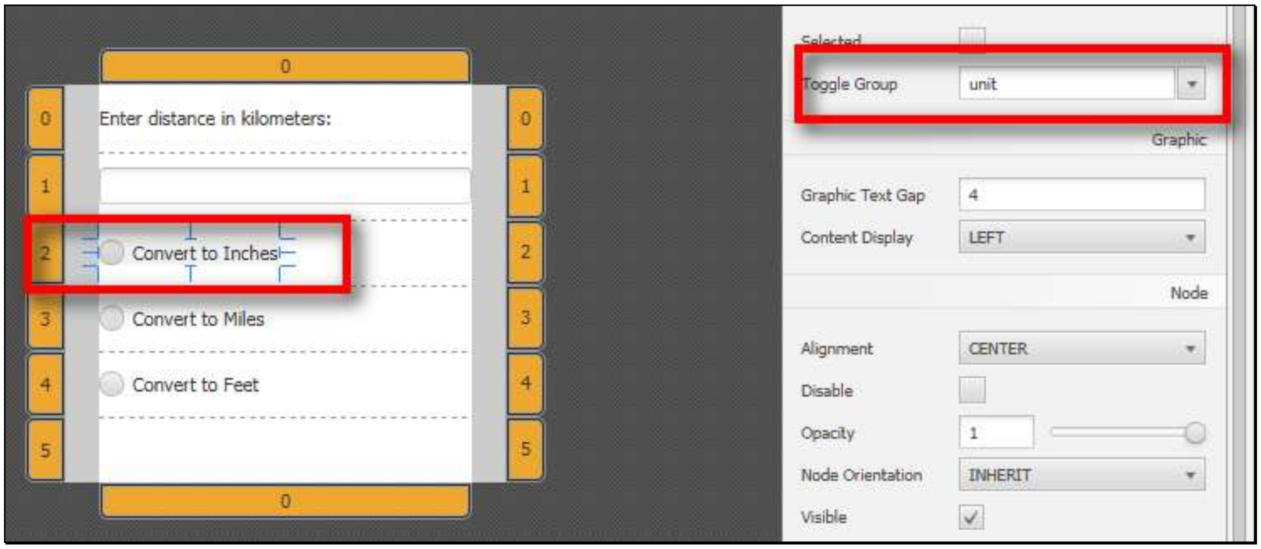
(39) Add the following components to each of the grid:



Component Type	Text	fx:id
Label	Enter distance in kilometers:	
TextField		kiloTextField
RadioButton	Convert to Inches	inchesRadioButton
RadioButton	Convert to Miles	milesRadioButton
RadioButton	Convert to Feet	feetRadioButton
Label		resultLabel

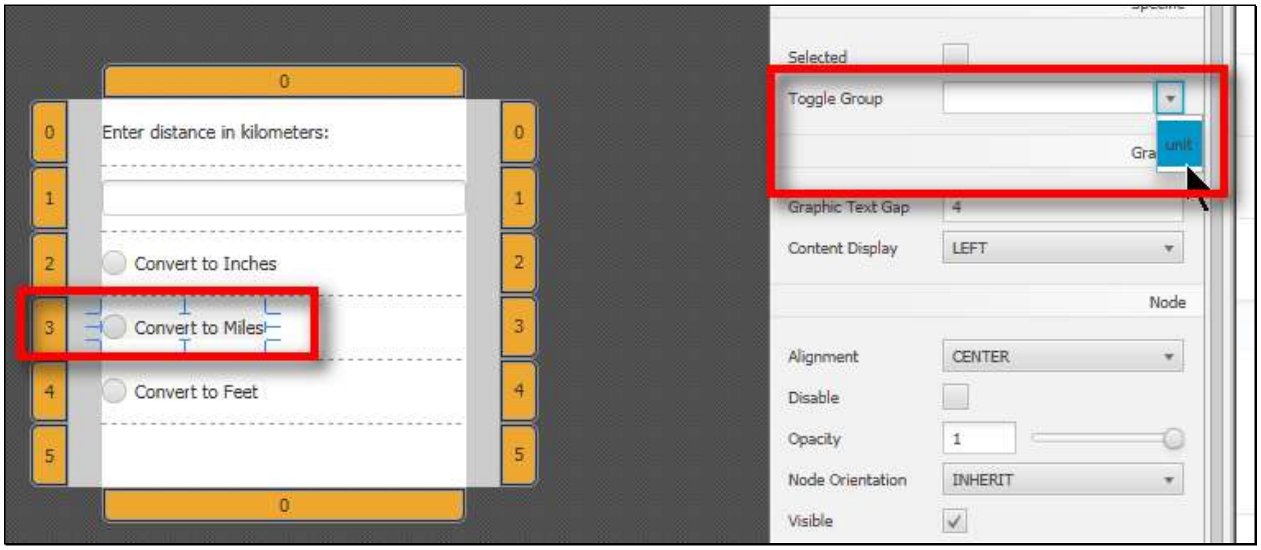
(40) The radio buttons need to be set in a group.

Click on **inchesRadioButton**, on **Properties** panel, set the **Toggle Group** as **unit**.



Selected
Toggle Group: unit
Graphic Text Gap: 4
Content Display: LEFT
Alignment: CENTER
Disable
Opacity: 1
Node Orientation: INHERIT
Visible: ✓

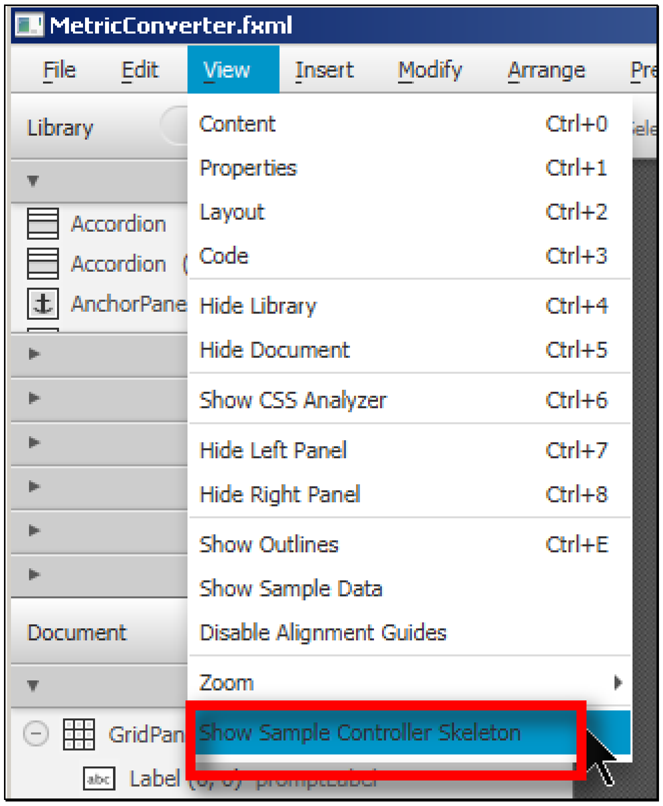
(41) Click **milesRadioButton**, set the **Toggle Group** by selecting **unit** from the drop-down list.



(42) Do the same for **feetRadioButton**.

(43) Save the fxml file.

(44) Click on **View \ Show Sample Controller Skeleton**, and copy the code.

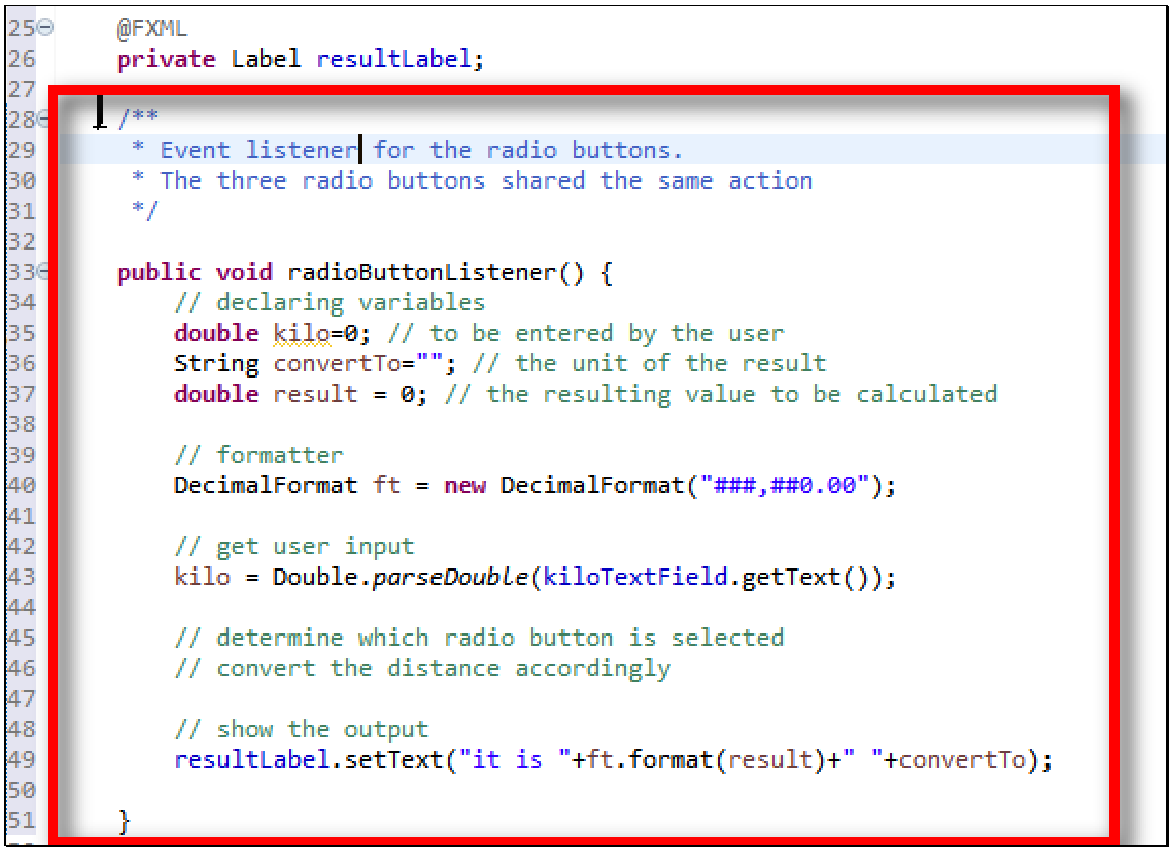


(45) Close the **Scence Builder**.

(46) Please open **MetricConverterController.java.**

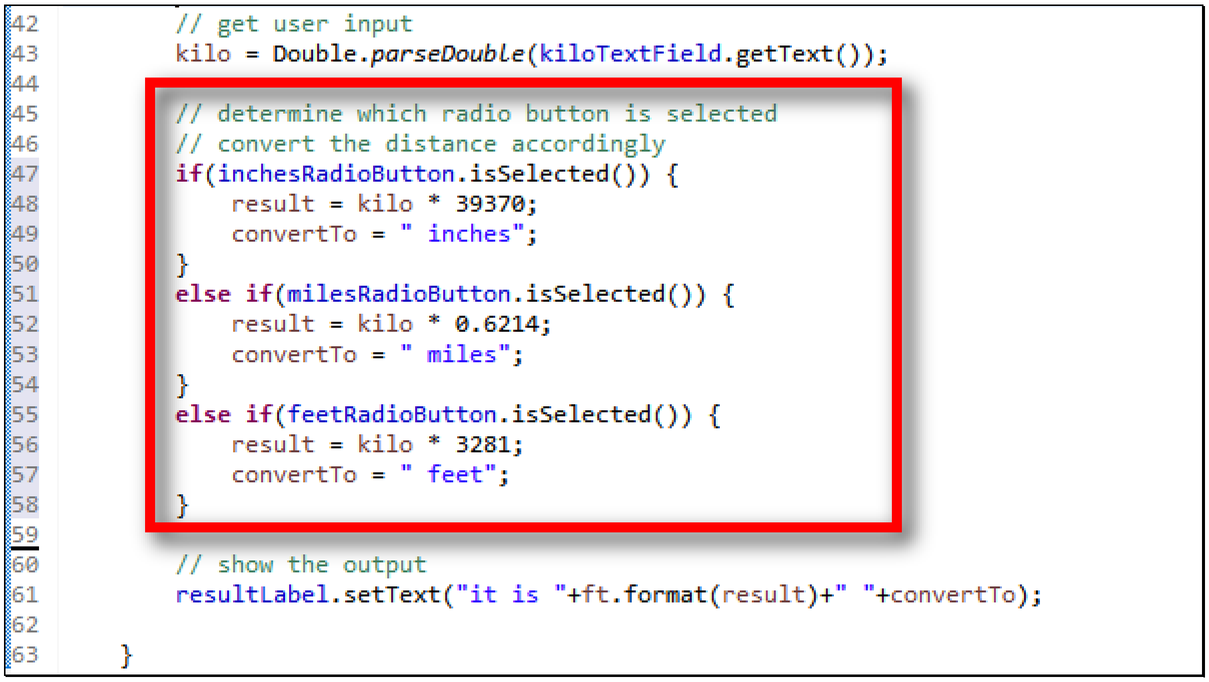
The program already has **radioButtonListener**() created to serve as the event listener.

Note: if you name the controls different in Scene Builder, you will need to change the controls declaration on lines 10-26.



```
25      @FXML
26      private Label resultLabel;
27
28      /**
29       * Event listener for the radio buttons.
30       * The three radio buttons shared the same action
31       */
32
33      public void radioButtonListener() {
34          // declaring variables
35          double kilo=0; // to be entered by the user
36          String convertTo=""; // the unit of the result
37          double result = 0; // the resulting value to be calculated
38
39          // formatter
40          DecimalFormat ft = new DecimalFormat("###,##0.00");
41
42          // get user input
43          kilo = Double.parseDouble(kiloTextField.getText());
44
45          // determine which radio button is selected
46          // convert the distance accordingly
47
48          // show the output
49          resultLabel.setText("it is "+ft.format(result)+" "+convertTo);
50
51      }
```

(47) For each radio button, check whether it is selected. If yes, convert the input accordingly.



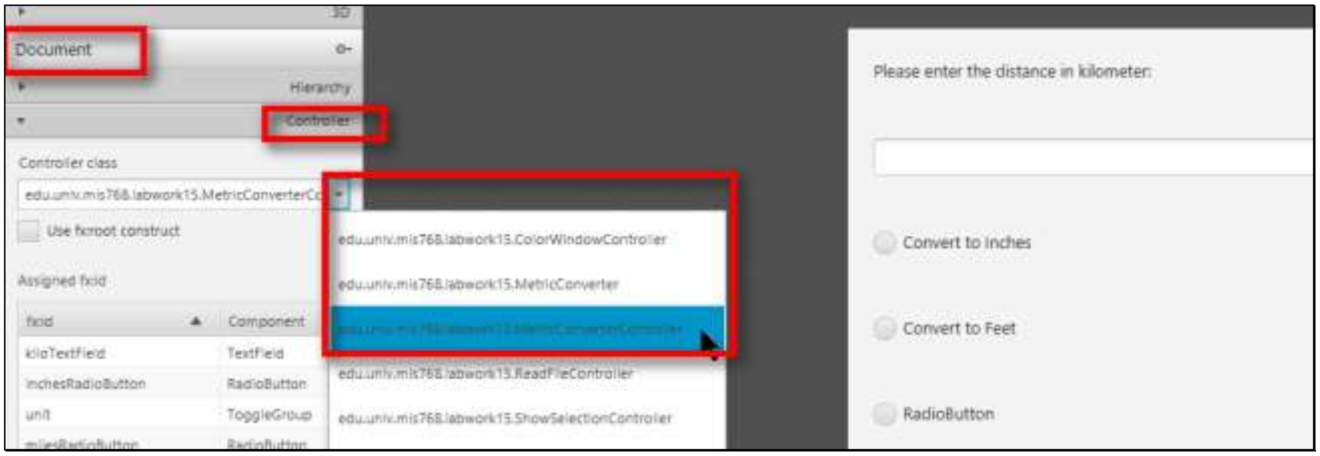
```
42          // get user input
43          kilo = Double.parseDouble(kiloTextField.getText());
44
45          // determine which radio button is selected
46          // convert the distance accordingly
47          if(inchesRadioButton.isSelected()) {
48              result = kilo * 39370;
49              convertTo = " inches";
50          }
51          else if(milesRadioButton.isSelected()) {
52              result = kilo * 0.6214;
53              convertTo = " miles";
54          }
55          else if(feetRadioButton.isSelected()) {
56              result = kilo * 3281;
57              convertTo = " feet";
58          }
59
60          // show the output
61          resultLabel.setText("it is "+ft.format(result)+" "+convertTo);
62
63      }
```

(48) Save and close the file.

(49) Open **MatricConverter.fxml** in Scence Builder.

(50) Select **Controller** panel of rhe **Docement**.

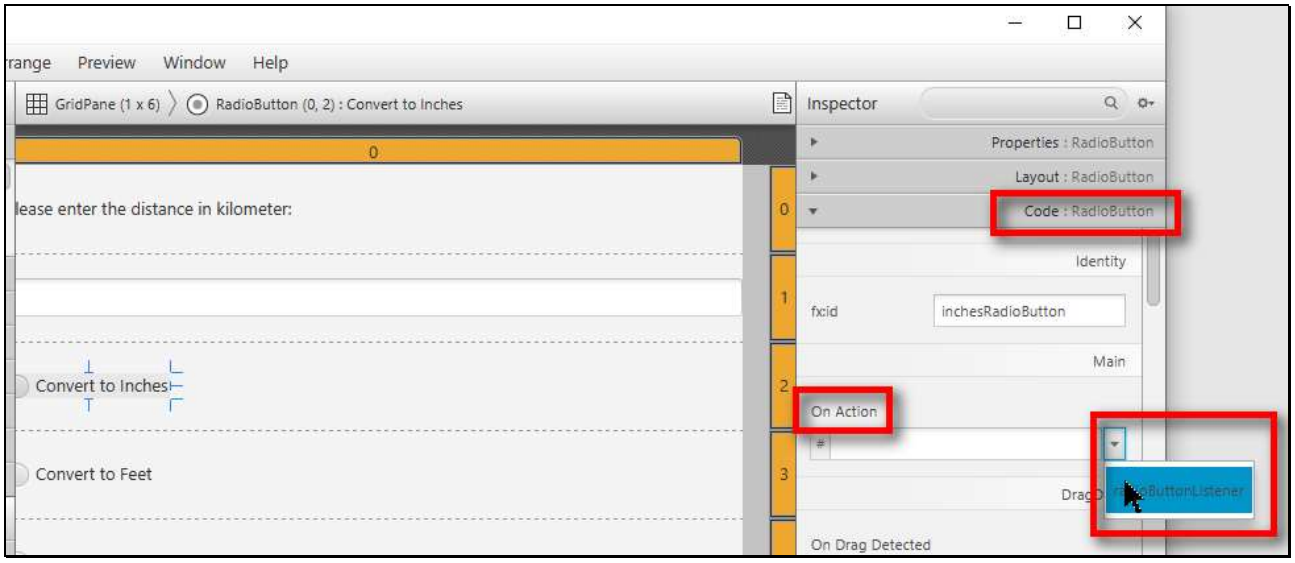
To select the **Controller class**, use the drop down list to choose **MetricConverterController**



Document
Hierarchy
Controller
Controller class
edu.univ.mis768.labwork15.MetricConverterC
Use fx:root construct
Assigned fx:id
fx:id
Component
kiloTextField
TextField
inchesRadioButton
RadioButton
unit
ToggleGroup
milesRadioButton
RadioButton
edu.univ.mis768.labwork15.ColorWindowController
edu.univ.mis768.labwork15.MetricConverter
edu.univ.mis768.labwork15.MetricConverterController
edu.univ.mis768.labwork15.ReadFileController
edu.univ.mis768.labwork15.ShowSelectionController
Please enter the distance in kilometer:
Convert to Inches
Convert to Feet
RadioButton

(51) Select the **inchesRadioButtonn**, and open the **Code** panel.

under **On Action,** use the drop down list to select **radioButtonListener**().

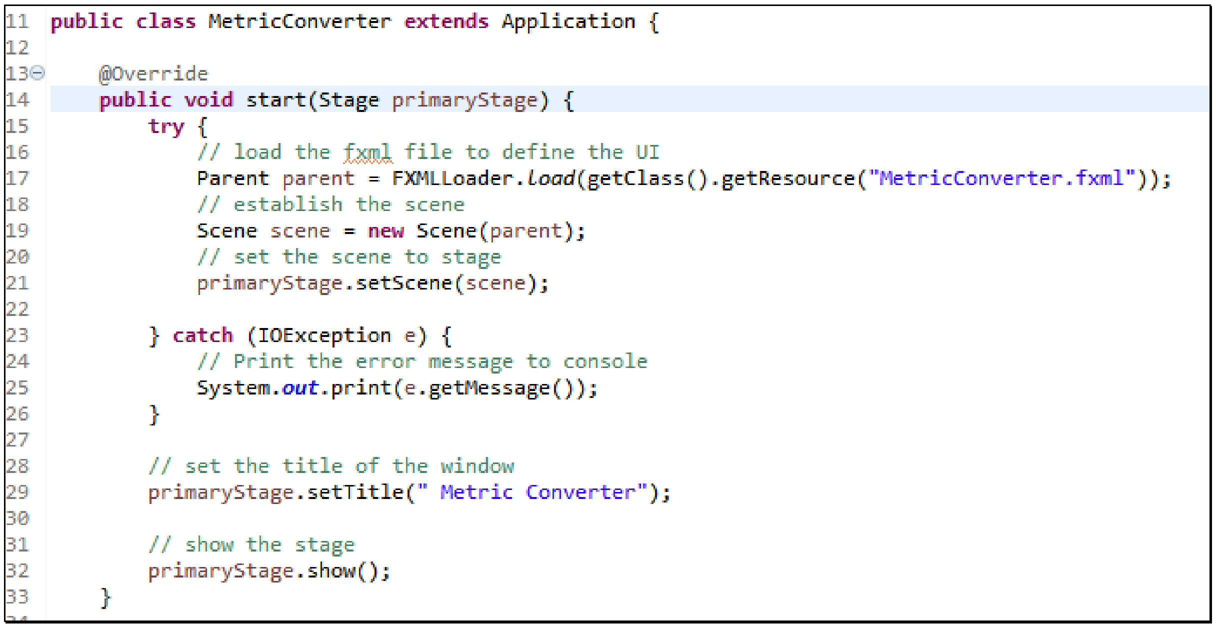


(52) Do the same for the other two radio buttonw. By doing so, we associate on listener to three components.

(53) Save and close Scene Builder.

(54) Please open **MetricCoverter**. The start() method has been implemented.

Please note that it uses the try-catch clause rather than throws exceptions at method header.

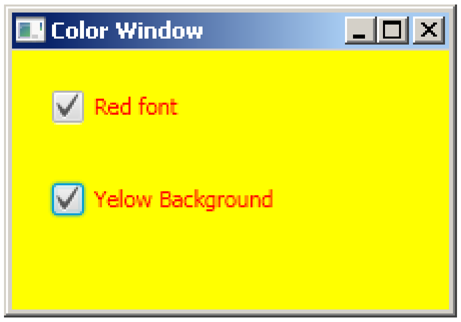


```
public class MetricConverter extends Application {

    @Override
    public void start(Stage primaryStage) {
        try {
            // load the fxml file to define the UI
            Parent parent = FXMLLoader.load(getClass().getResource("MetricConverter.fxml"));
            // establish the scene
            Scene scene = new Scene(parent);
            // set the scene to stage
            primaryStage.setScene(scene);

        } catch (IOException e) {
            // Print the error message to console
            System.out.print(e.getMessage());
        }

        // set the title of the window
        primaryStage.setTitle(" Metric Converter");

        // show the stage
        primaryStage.show();
    }
```

(55) Set the VM argument and run **MetricCoverter** to see the result.

7. Check Box

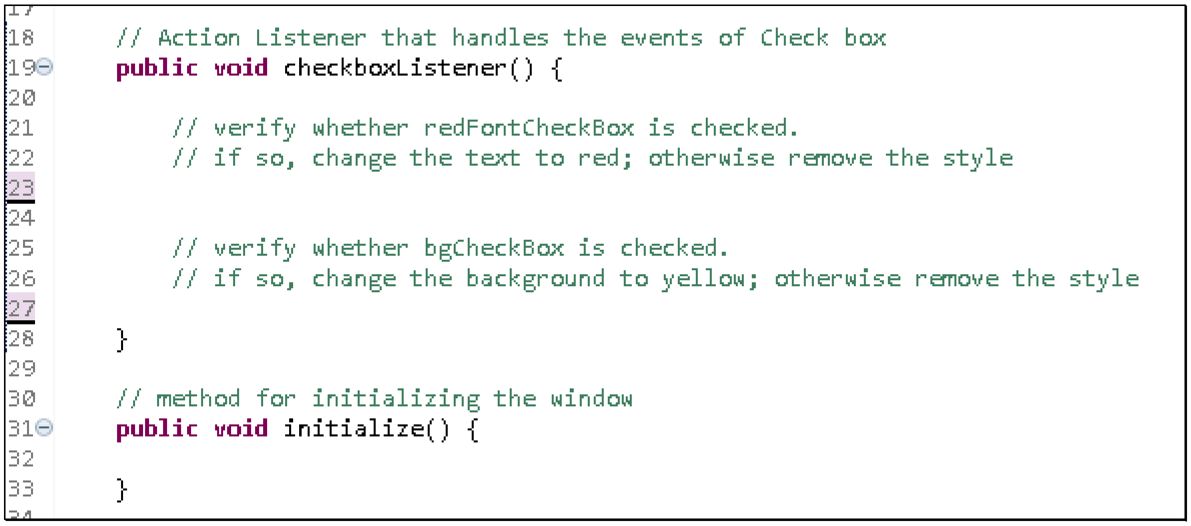
(56) In the application, we change the font color and background color per the action on the check boxes.



(57) Please open **ColorWindow.fxml** in Scene Builder. The components are added to a **VBox** with **fx:id** assigned.

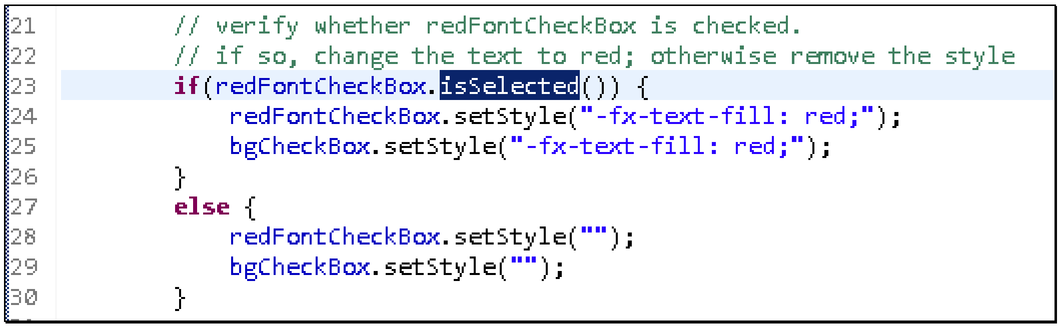
(58) Open **ColorWindowController.**

The **checkboxListener**() method have been create.



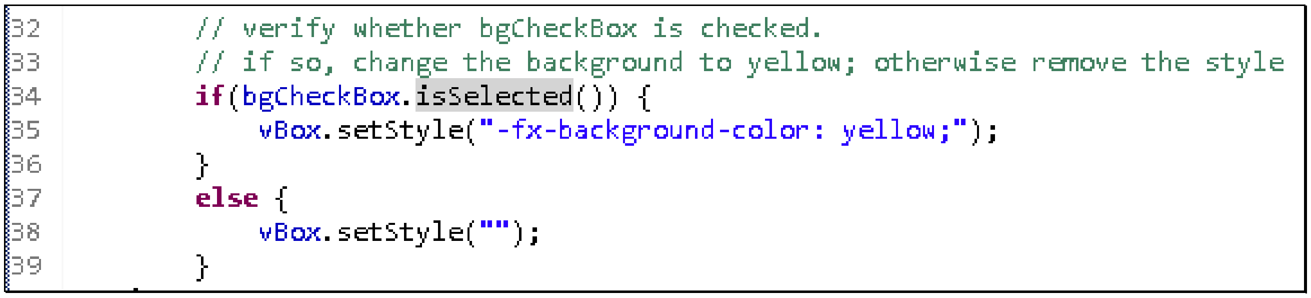
```
    // Action Listener that handles the events of Check box
    public void checkboxListener() {

        // verify whether redFontCheckBox is checked.
        // if so, change the text to red; otherwise remove the style


        // verify whether bgCheckBox is checked.
        // if so, change the background to yellow; otherwise remove the style

    }

    // method for initializing the window
    public void initialize() {

    }
```

(59) Use the **isSelected**() method of check box to verify the status and set the style of the text



```
        // verify whether redFontCheckBox is checked.
        // if so, change the text to red; otherwise remove the style
        if(redFontCheckBox.isSelected()) {
            redFontCheckBox.setStyle("-fx-text-fill: red;");
            bgCheckBox.setStyle("-fx-text-fill: red;");
        }
        else {
            redFontCheckBox.setStyle("");
            bgCheckBox.setStyle("");
        }
```

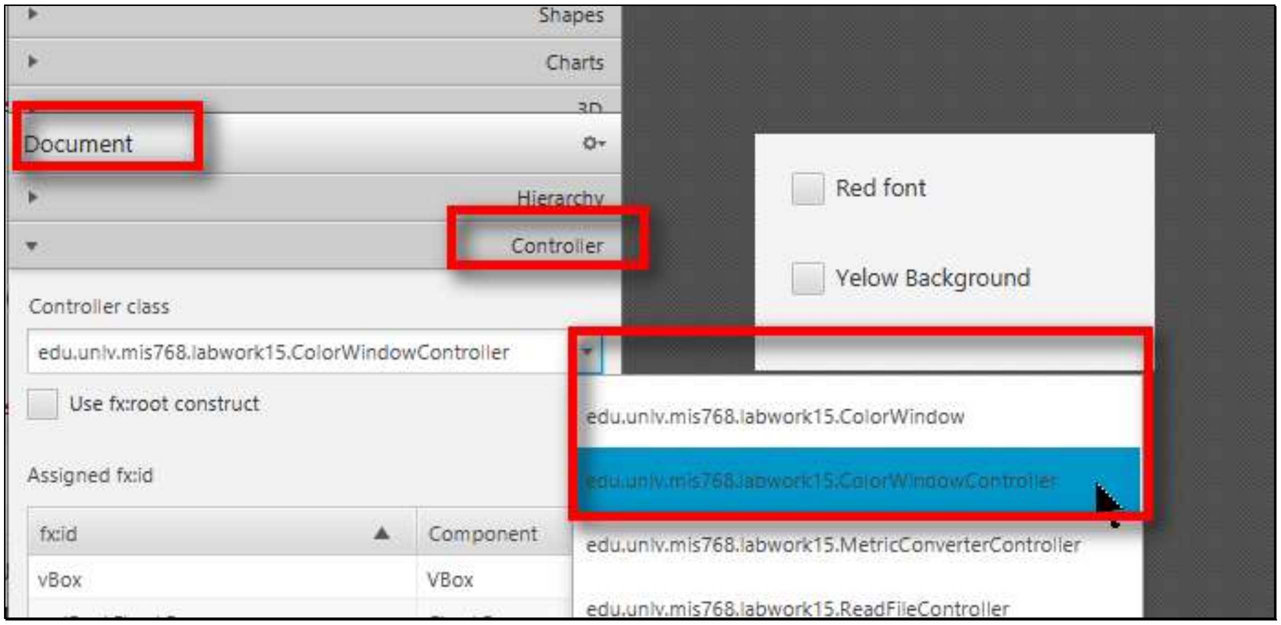
(60) For changing the background, we need to set the style for vBox.



```
        // verify whether bgCheckBox is checked.
        // if so, change the background to yellow; otherwise remove the style
        if(bgCheckBox.isSelected()) {
            vBox.setStyle("-fx-background-color: yellow;");
        }
        else {
            vBox.setStyle("");
        }
```

(61) Switch back to **ColorWindow.fxml** in Scene Builder.

Set the Controller class.



Controller class

edu.unlv.mis768.labwork15.ColorWindowController

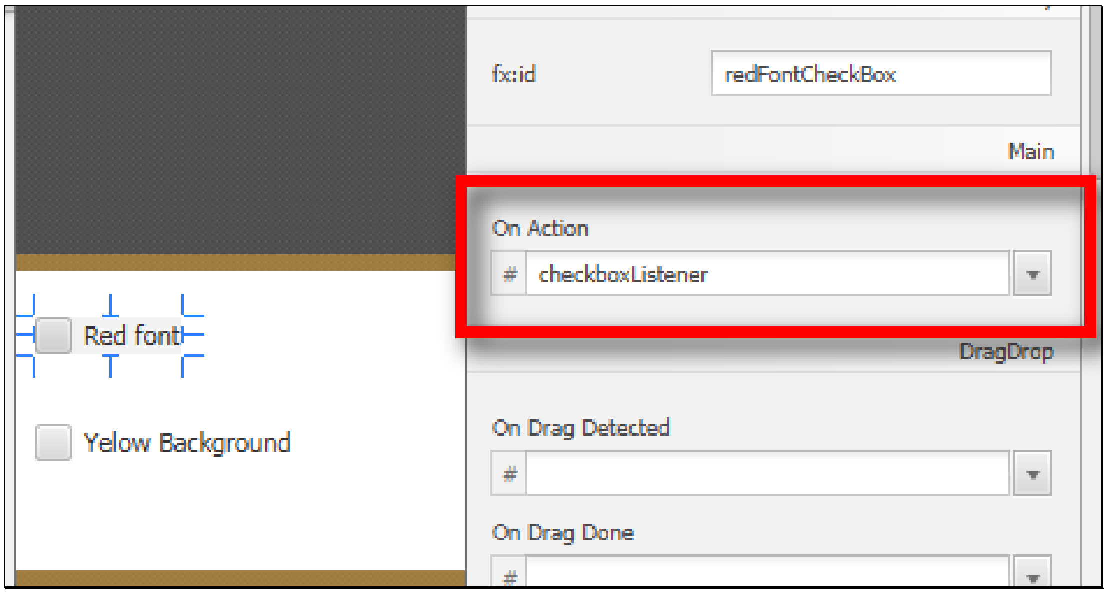
edu.unlv.mis768.labwork15.ColorWindow

edu.unlv.mis768.labwork15.ColorWindowController

edu.unlv.mis768.labwork15.MetricConverterController

edu.unlv.mis768.labwork15.ReadFileController

(62) For both check boxes, set **On Action** to **checkboxListener()**



fx:id redFontCheckBox

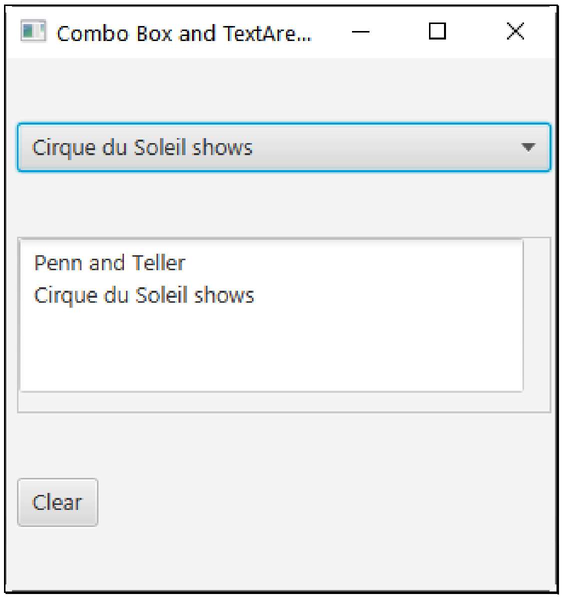
On Action # checkboxListener

(63) Save the file and close the **Scene Builder.**

(64) Open **ColorWindow.java** and run the application.

8. ComboBox and TextArea

(65) In this application, when the user selects an item in the combo box, the context will be added to the text area. The user can also click the **Clear** button to empty the text area.

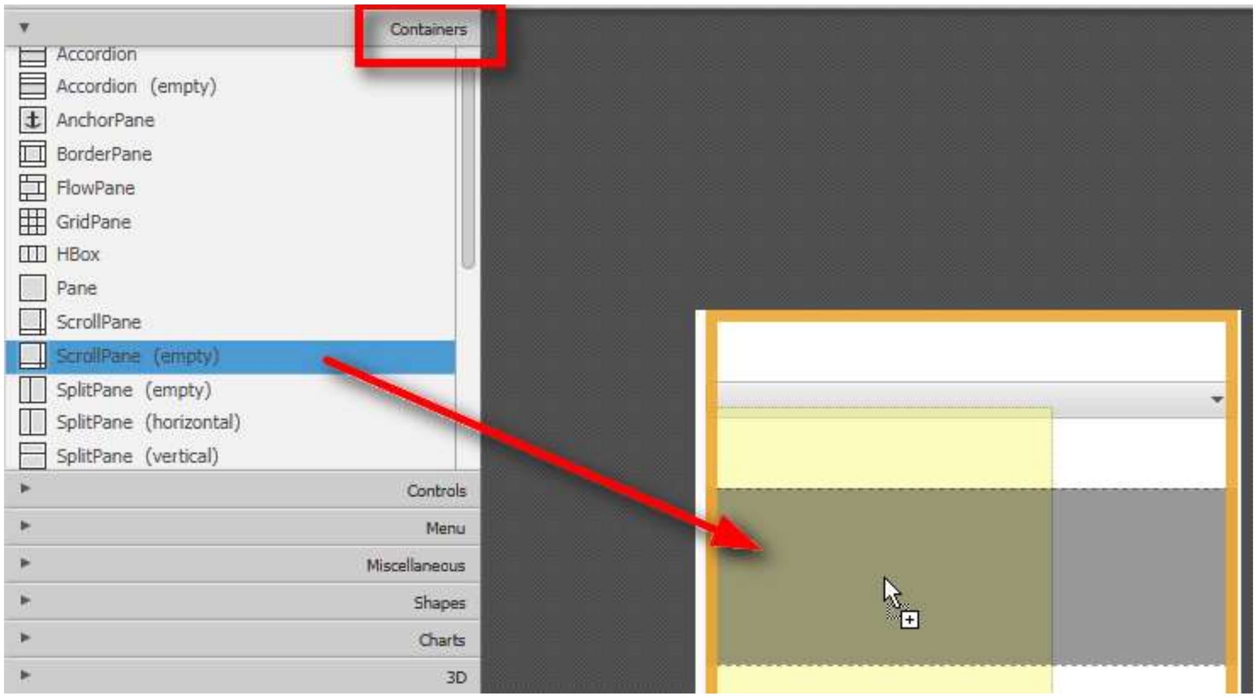


Penn and Teller
Cirque du Soleil shows

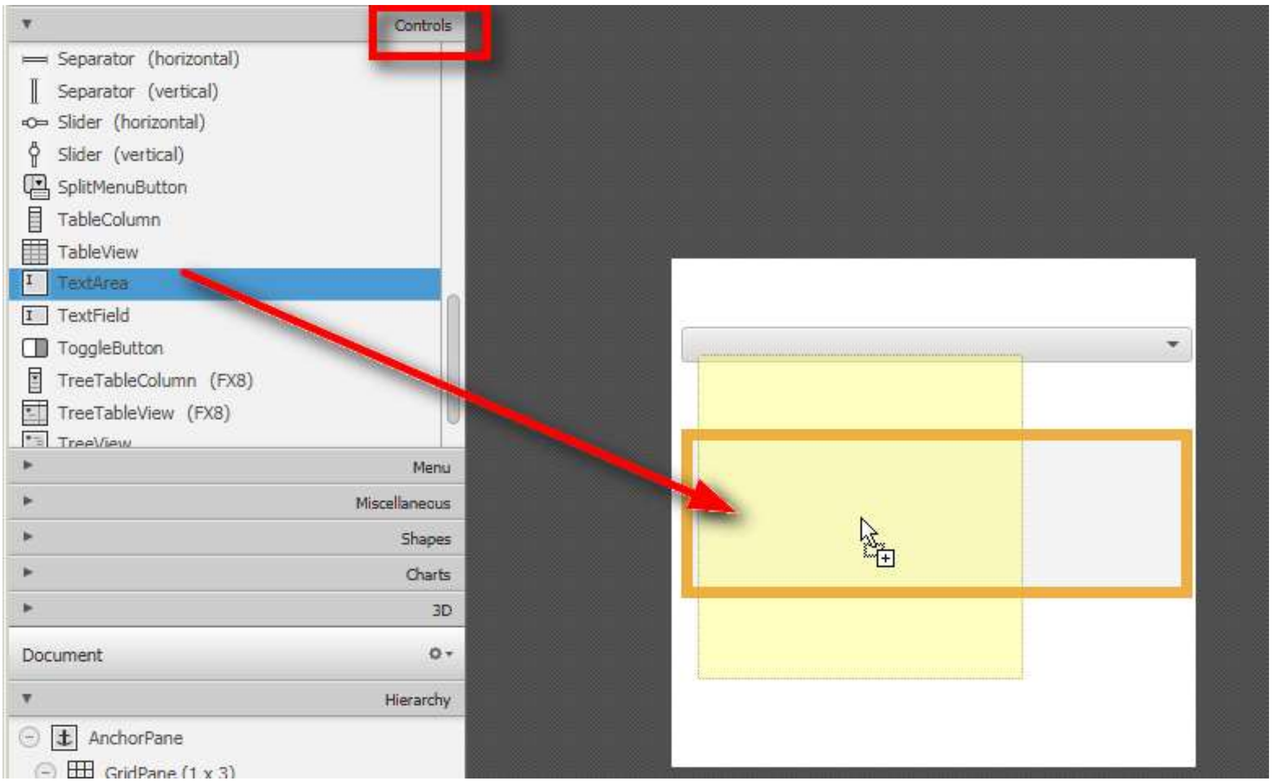
(66) Open **ShowSelection.fxml** in Scene Builder. The ComboBox has been added.

(67) We need the Text Area to be scrollable; therefore we need to add a ScrollPane before adding the Text Area.

Please drag an **ScrollPane (empty)** to the second row of the GridPane.



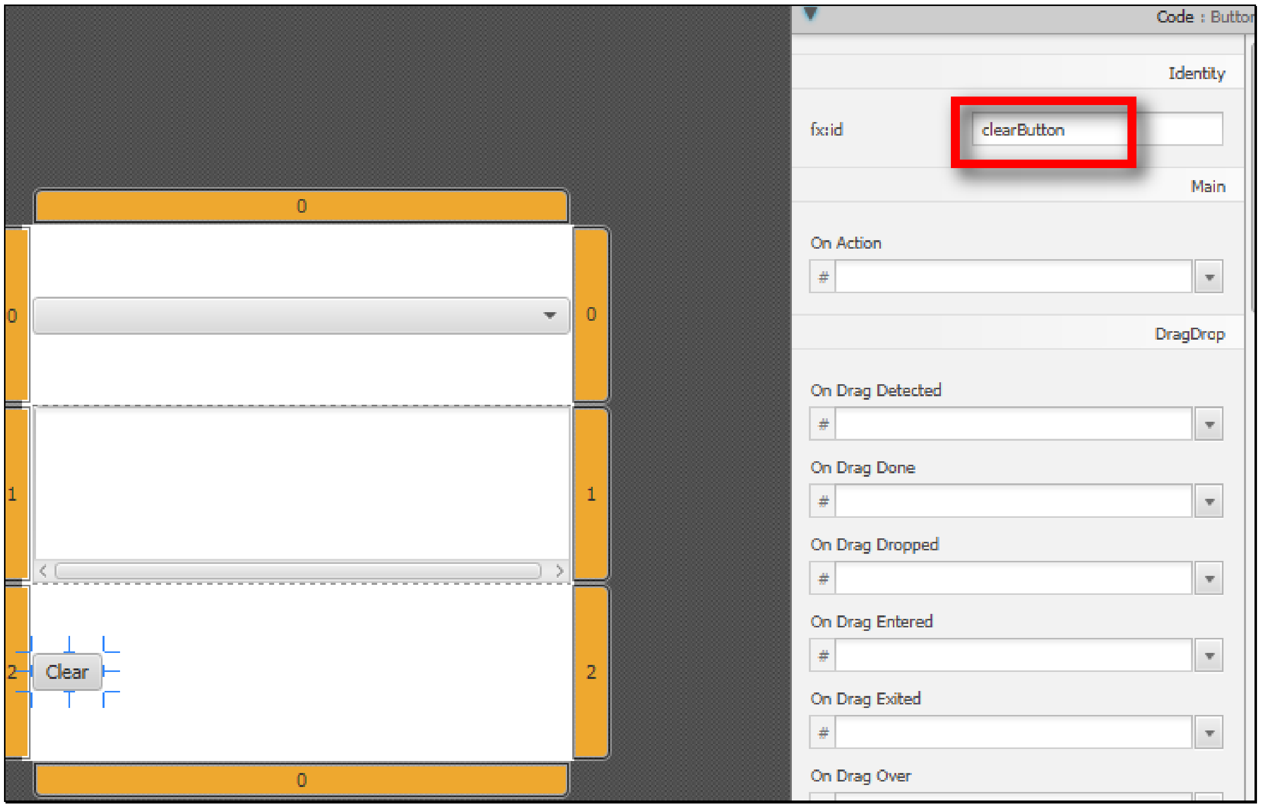
(68) Then add a **Text Area** to the **ScrollPane**. Set the **fx:id** as **contentTextArea**



(69) The default size of the Text Area does not fit the ScrollPane. Please resize it as needed.

(70) Add a Button to the third row. Set the **fx:id** as **clearButton**.

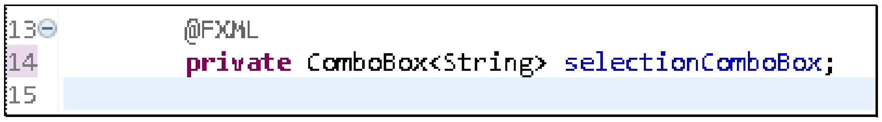
Set the **Text** as **Clear**



(71) Save and exit Scene Builder.

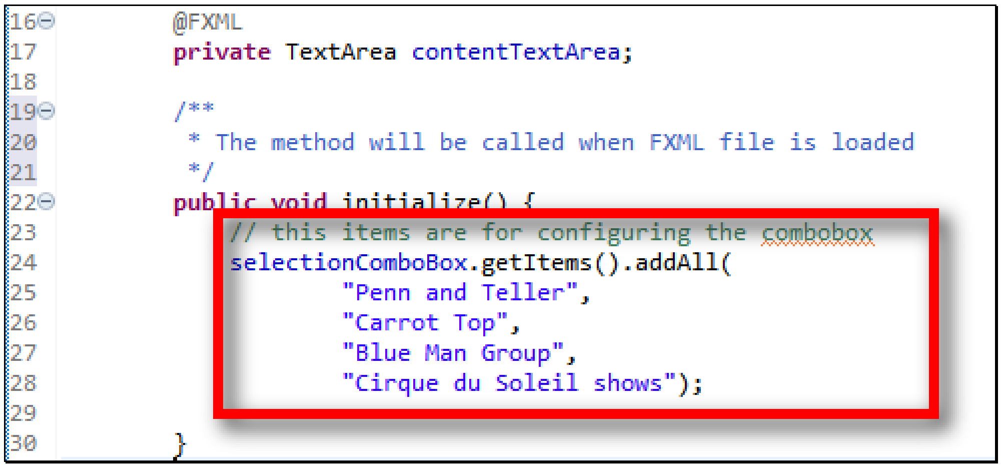
(72) Please open **ShowSelectionController.java** in Eclipse.

(73) The data type for the **ComboBox** by default is <?>. Please change it to String.



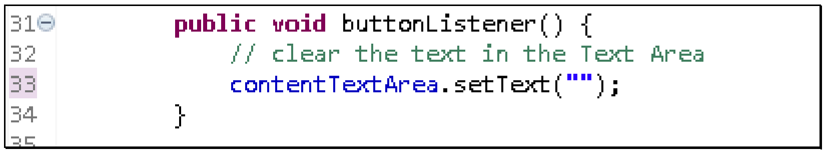
```
    @FXML
    private ComboBox<String> selectionComboBox;

```

(74) In the **initialize()** method, we need to set the items of the ComboBox.



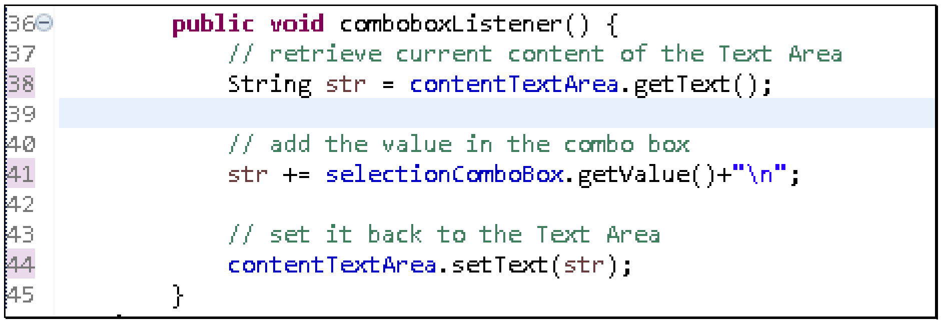
```
    @FXML
    private TextArea contentTextArea;

    /**
     * The method will be called when FXML file is loaded
     */
    public void initialize() {
        // this items are for configuring the combobox
        selectionComboBox.getItems().addAll(
                "Penn and Teller",
                "Carrot Top",
                "Blue Man Group",
                "Cirque du Soleil shows");

    }
```

(75) The **buttonListener**() is used to clear the text in the Text Area.



```
    public void buttonListener() {
        // clear the text in the Text Area
        contentTextArea.setText("");
    }
```

(76) Please implement **comboboxListener**() to add the value of the Combo Box to the Text Area.



```
    public void comboboxListener() {
        // retrieve current content of the Text Area
        String str = contentTextArea.getText();

        // add the value in the combo box
        str += selectionComboBox.getValue()+"\n";

        // set it back to the Text Area
        contentTextArea.setText(str);
    }
```

(77) Open **ShowSelection.fxml** in Scene Builder.

Please set the **Document** \ **Controller**.

Also set **On Action** for **clearButton** and **selectionComboBox**, respectively.

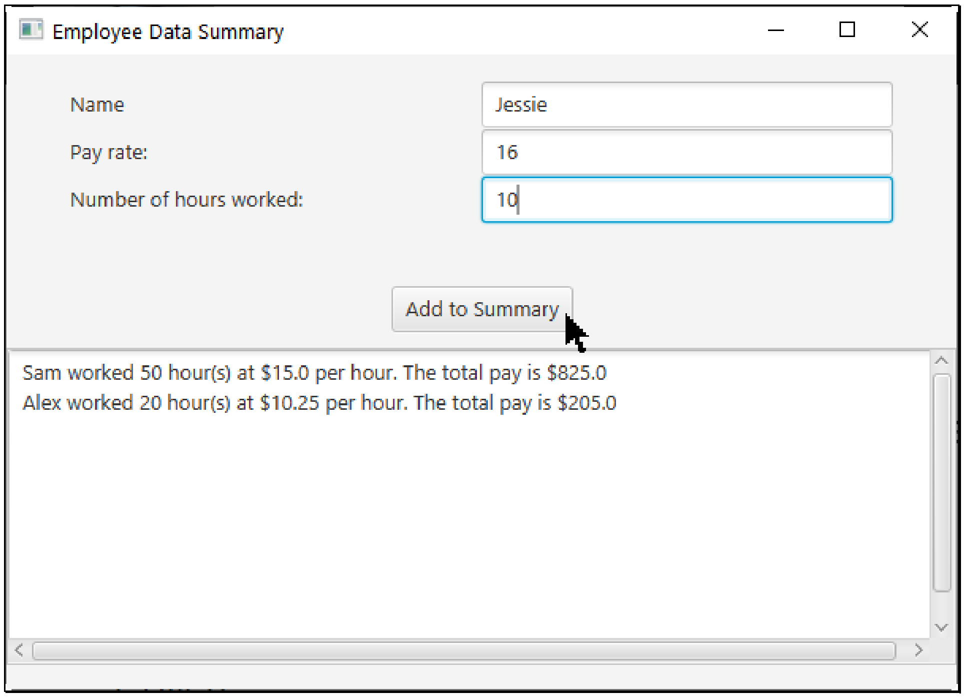
(78) Run **ShowSelection.java** to see the result.

9. Exercise: Employee Data

(79) Please create a JavaFX application to allow entry of employee name, pay rate, and working hours. Then add the employee details into a text area showing the details with total pay.

Payroll (included in the lab files) is a model class with three fields and some methods.

At the action listener of the button, please instantiate Payroll objects and use its methods to complete the program.



Jessie

16

10

```
Sam worked 50 hour(s) at $15.0 per hour. The total pay is $825.0
Alex worked 20 hour(s) at $10.25 per hour. The total pay is $205.0
```