MIS 768: Advanced Software Concepts

Spring 2024

GUI Application (1)

Purpose

- Understand the concepts of creating graphical user interface using JavaFX.
- Practice the creation of JavaFX controls and event handler.
- See example of setting up customized style for the applications.

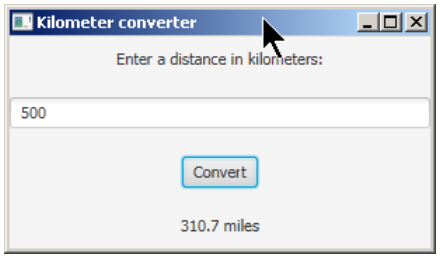
1. Preparation

(1) Launch Eclipse. Create a new package to hold our source file. Name the package as **edu.unlv.mis768.labwork14.**

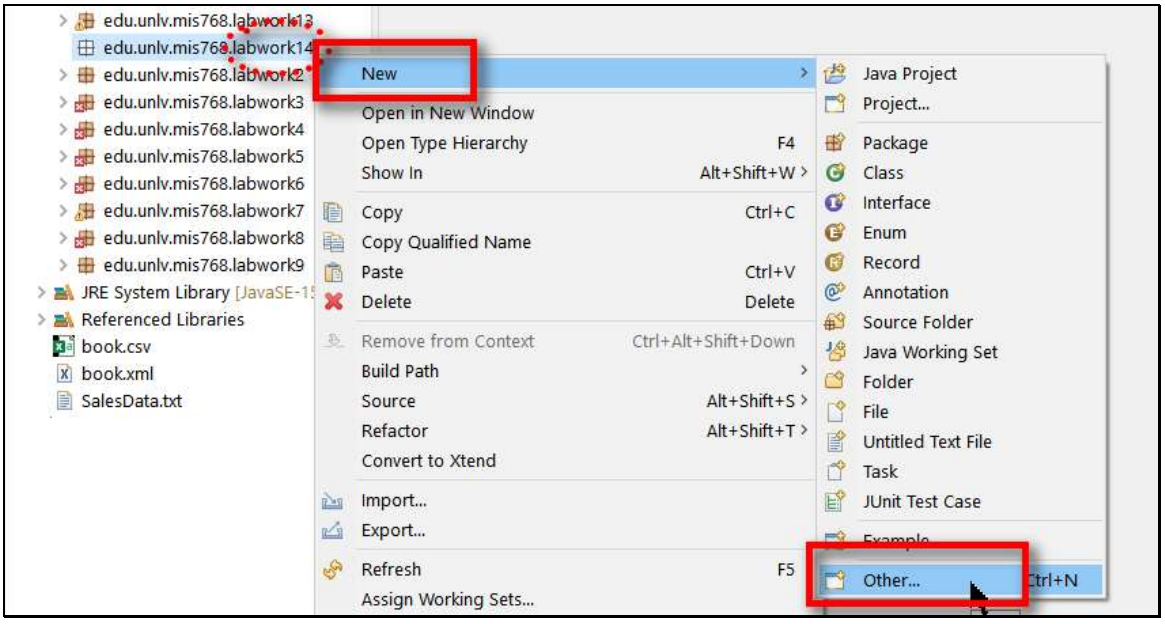
(2) Download **14_lab_files.zip** from WebCampus. Extract the zip file and then import the .java files into the package.

2. Create the first JavaFX program

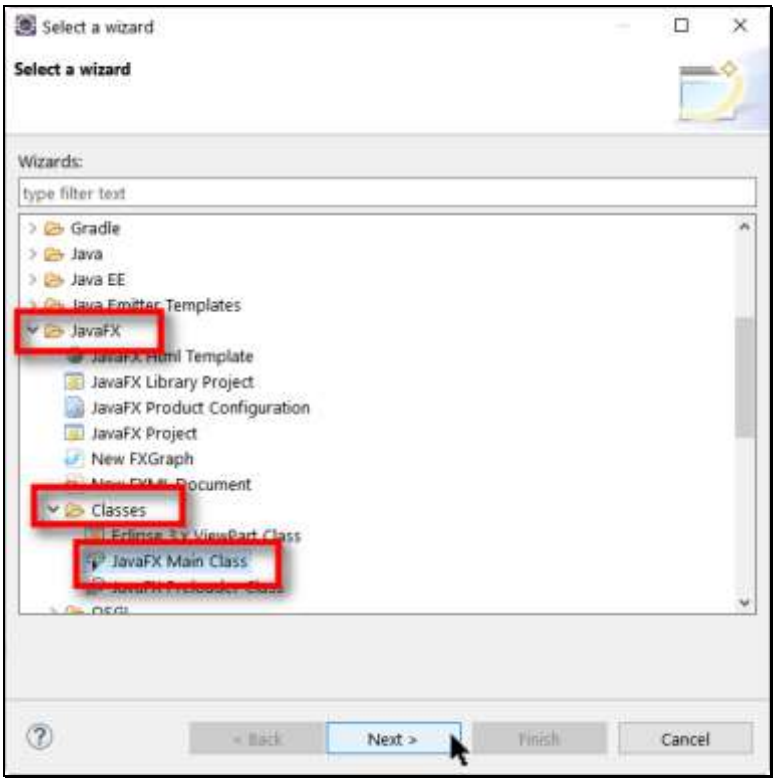
(3) In this example, we will create a simple program for distance conversion.



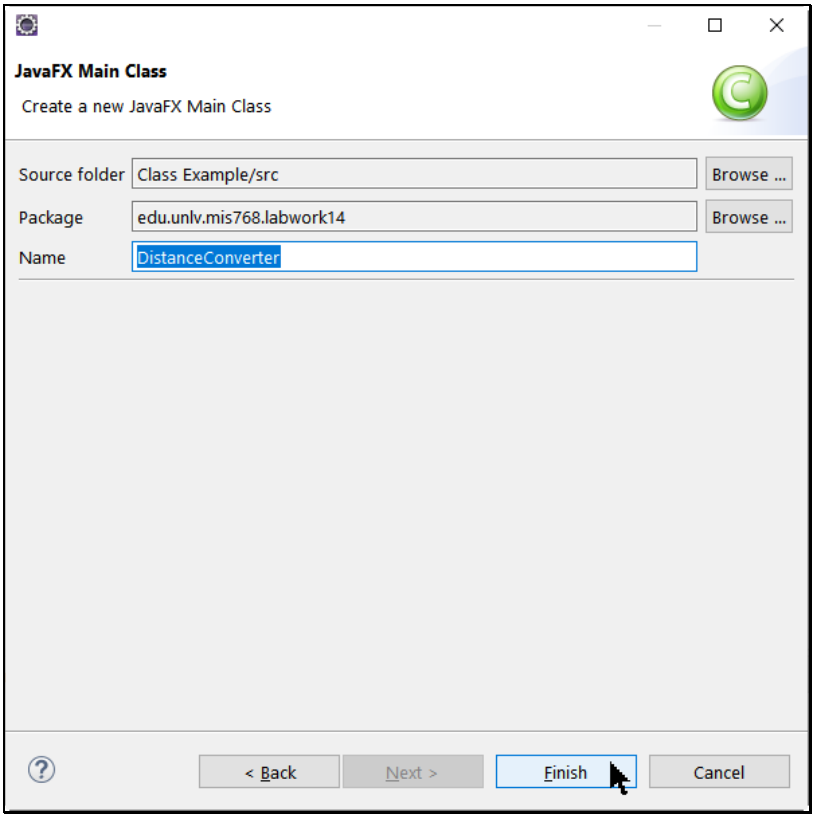
(4) Right click on the package, select **New \ Other**



(5) Find **JavaFX \ Classes \ JavaFX Main Class**. Select and click **Next>** button.



(6) Enter the name as **DistanceConverter** and click **Finish** button.



(7) The template of a JavaFX Application is created.

```
3  import javafx.application.Application;
5  import javafx.stage.Stage;
6
7  public class MyFirstGUI extends Application {
8
9      @Override
10     public void start(Stage primaryStage) {
11
12     }
13
14     public static void main(String[] args) {
15         launch(args);
16     }
17 }
```

3. Create controls, add to the Scene and Stage

(8) Add the import statement to use the different controls in JavaFX.

Please enter the code to the start method to create the objects: Label, TextField, Botton, and another Label.

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

public class MyFirstGUI extends Application {

    @Override
    public void start(Stage primaryStage) {
        // create the controls
        Label promptLabel = new Label("Enter a distance in kilometers:");
        TextField kiloTextField = new TextField();
        Button calcButton = new Button("Convert");
        Label resultLabel = new Label("");

    }
```

(9) Next is to create the layout container. In this example, we used VBox, and set the spacing to 20. Add the controls in the order they should be displayed.

We can also set the alignment of the components:

```
        // create a vertical box, set the spacing between controls to 20
        //   and add the controls to the vbox
        VBox vbox = new VBox(20, promptLabel, kiloTextField, calcButton, resultLabel);

        // center-align the VBox
        vbox.setAlignment(Pos.CENTER);
```

(10) Make the Vbox as the root node in the scent.

Then set the scent to the stage.

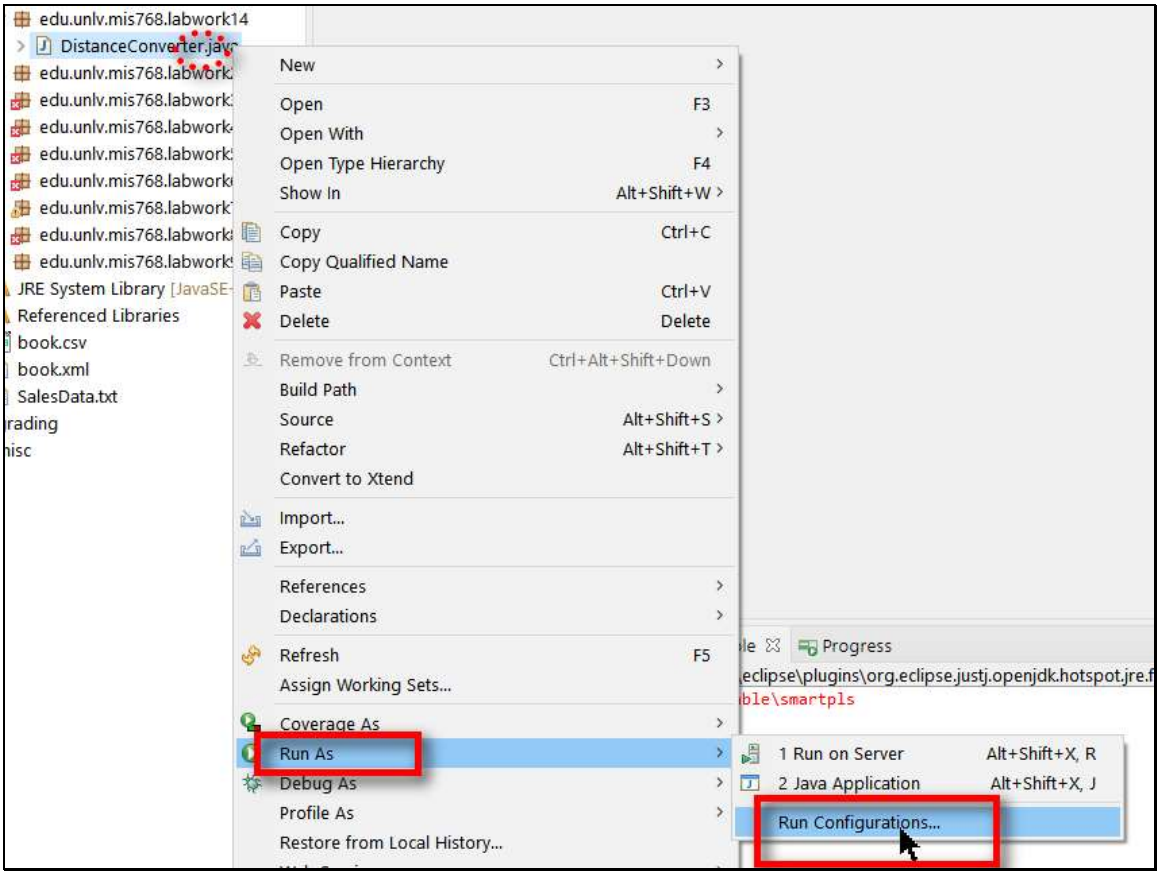
```
31          // Make the VBox the root node in the scene, with width and height
32          Scene scene = new Scene(vbox , 300, 150);
33
34          // Set the scene to the stage
35          primaryStage.setScene(scene);
36
```

(11) Before showing the stage, we can set the title

```
38          // Set the stage title
39          primaryStage.setTitle("Kilometer converter");
40
41          //Show the stage (display it)
42          primaryStage.show();
```

4. Setup VM arguments

(12) Right click on the file, then select **Run As \ Run Configurations…**



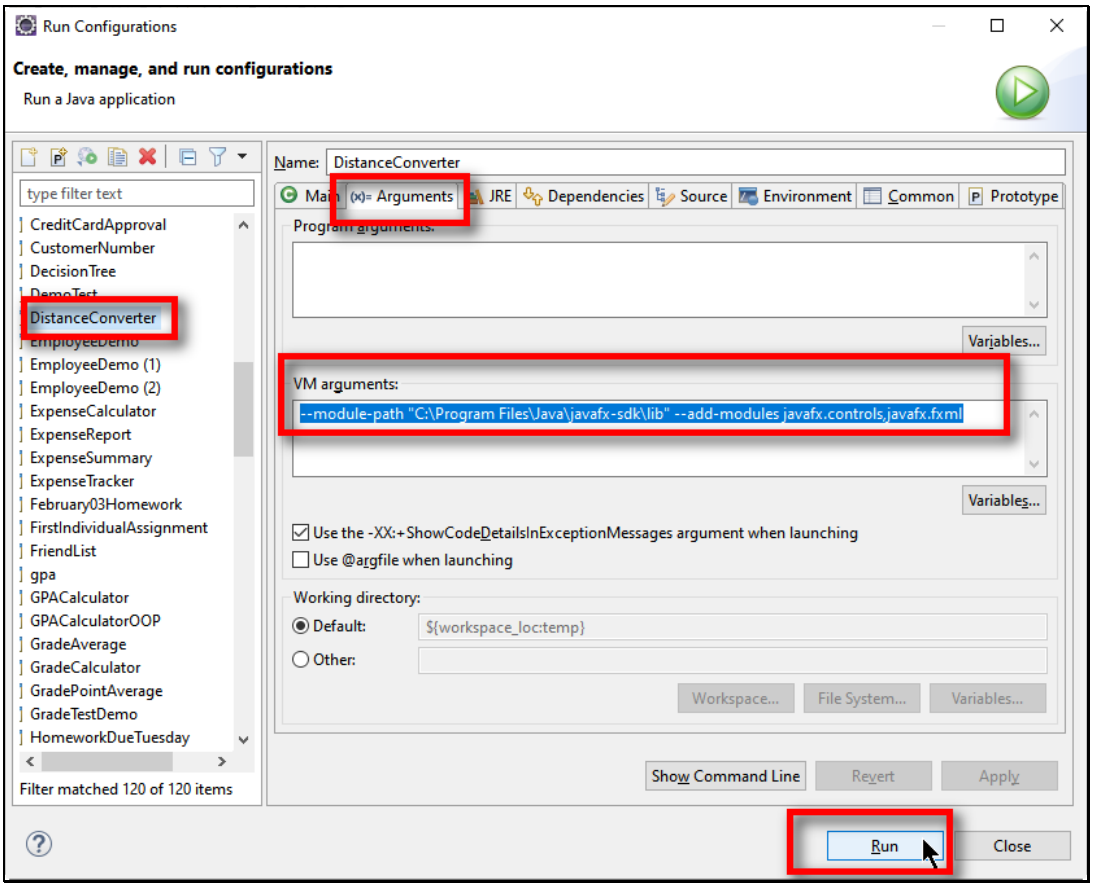
(13) Click **Arguments** tab, under **VM argements:**, enter the following

--module-path "C:\Program Files\Java\javafx-sdk\lib" --add-modules javafx.controls,javafx.fxml

Or

--module-path "/user/xxxx/download/javafx-sdk/lib" --add-modules javafx.controls,javafx.fxml

Please use the path you saved the JavaFX SDK.

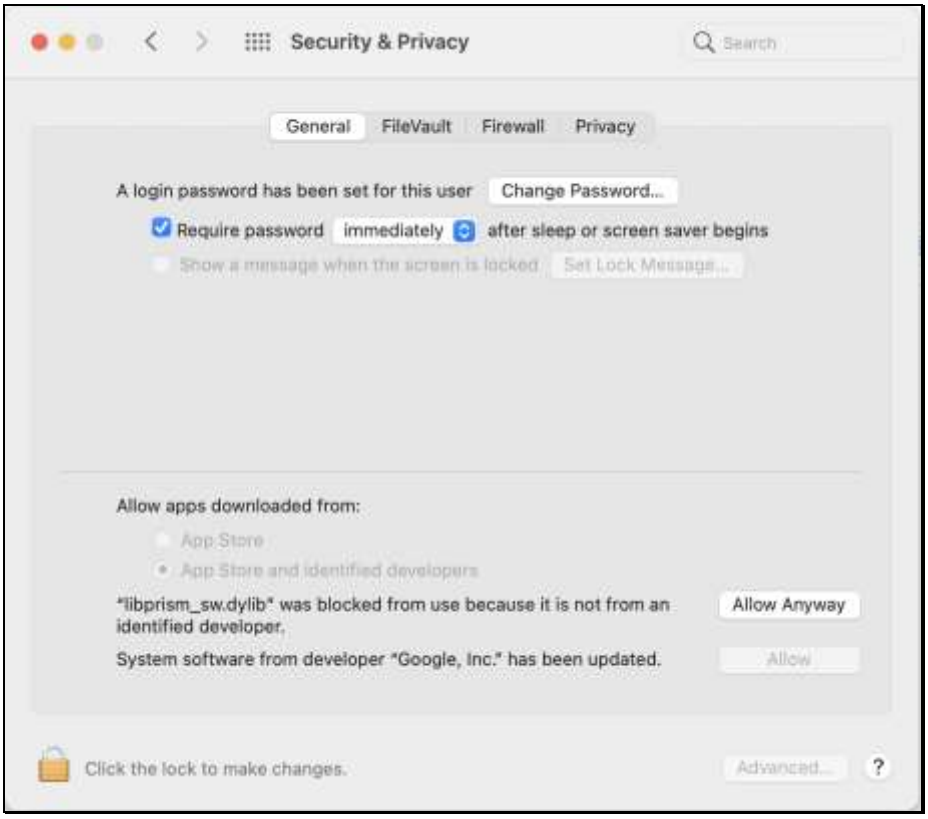


(14) Click **Run** button to test the application.

(15) For Mac users, if you see and error indication the file cannot be opened. You need to do an additional step and change the setting in **Security & Privacy**.

(16) Open **Security & Privacy** and click **Allow Anyway** button to enable it. Then move back to Eclipses to run the program again.

You would need to do the same for several files.



5. Implement an event handler and register it with the object

(17) Add the import statement of **javafx.event.*;**

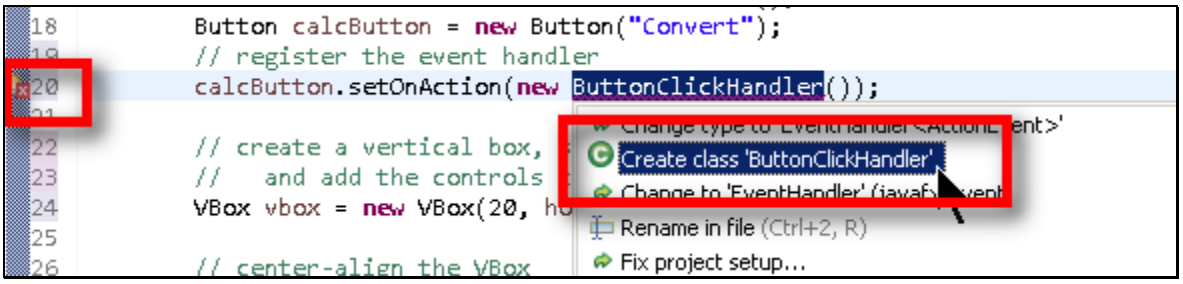
```
import javafx.application.Application;
import javafx.event.*;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.*;
```

(18) In the **start** method, add the following statement to set handler for **OnAction** event.

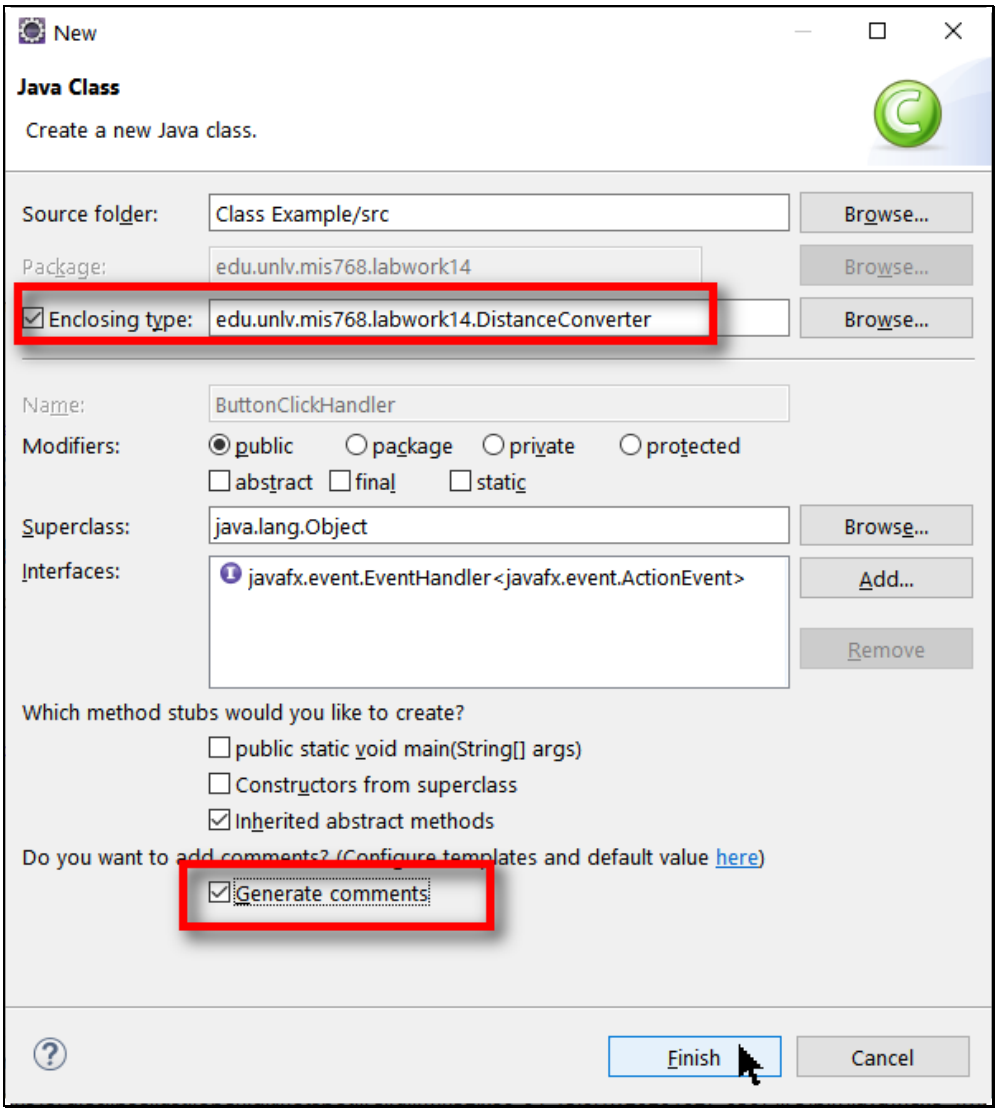
```
    public void start(Stage primaryStage) {
        // create the controls
        Label hoursLabel = new Label("Enter a distance in kilometers:");
        TextField hoursTextField = new TextField();
        Button calcButton = new Button("Convert");
        // register the event handler
        calcButton.setOnAction(new ButtonClickHandler());

```

(19) Click on the error, and select Create class '**ButtonClickHandler**'



(20) In the new Class wizard, select the option "**Enclosing type**." By doing so, we create an inner class (i.e., a class inside a class). Please also check the option "**Generate comments"**.



(21) If the method is generated on top of the program, please move it to the spot below the **start** method.

(22) We willl later comeback and implement the **handle** method.

```
45      /**
46       * Event Handler class for calcButton
47       * @author huh4
48       *
49       */
50
51      public class ButtonClickHandler implements EventHandler<ActionEvent> {
52          @Override
53          public void handle(ActionEvent event) {
54
55
56          }
57
58      }
```

(23) In order to use the **kiloTextField** and **resultLabel** in the **ButtonClickHandler** class, we need to make them as the fields of the **DistanceConverter** class.

Please declare them on top of **start**() method, and remove the declaration portion at line 20 and 22.

```
11  public class MyFirstGUI extends Application {
12
13      // Field
14      private TextField kiloTextField;
15      private Label resultLabel;
16      @Override
17      public void start(Stage primaryStage) {
18          // create the controls
19          Label promptLabel = new Label("Enter a distance in kilometers:");
20          kiloTextField = new TextField();
21          Button calcButton = new Button("Convert");
22          resultLabel = new Label("");
23          // register the event handler
24          calcButton.setOnAction(new ButtonClickHandler());
25
```

(24) In the **handle**() method, we will retrieve the value entered by the user, convert it to miles, and display the result in the label.

```
41      /**
42       * Event Handler for calcButton
43       * @author HH
44       *
45       */
46      public class ButtonClickHandler implements EventHandler<ActionEvent> {
47
48          @Override
49          public void handle(ActionEvent arg0) {
50              // Retrieve the value in kiloTextField, convert it to double
51              double kilo = Double.parseDouble(kiloTextField.getText());
52              // calculate the miles
53              double miles = kilo * 0.6214;
54              // show the result in the resultLabel
55              resultLabel.setText(miles + " miles.");
56
57          }
58
59      }
```

6. Layout Panes

(25) Please open **BorderPaneExample**.java. It shows an example of layout with **BoarderPane**.

In this example, five buttons are created. A few **HBox** and **VBox** panes are also created, with a button added to each container.

```
13      public void start(Stage primaryStage) {
14            // Create some buttons.
15            Button centerButton = new Button("This is Center");
16            Button topButton = new Button("This is Top");
17            Button bottomButton = new Button("This is Bottom");
18            Button leftButton = new Button("This is Left");
19            Button rightButton = new Button("This is Right");
20
21            // Add each button to its own layout container.
22            HBox centerHBox = new HBox(centerButton);
23            HBox topHBox = new HBox(topButton);
24            HBox bottomHBox = new HBox(bottomButton);
25            VBox leftVBox = new VBox(leftButton);
26            VBox rightVBox = new VBox(rightButton);
27
```

(26) HBox and VBox can be used to set alignment.

```
28            // Set the alignment for the top and bottom.
29            topHBox.setAlignment(Pos.CENTER);
30            bottomHBox.setAlignment(Pos.CENTER);
31
```

(27) Then the VBox or HBox containers are dded to a region in the **BorderPane**.

```
35            // Add the buttons to the BorderPane's regions.
36            borderPane.setCenter(centerHBox);
37            borderPane.setTop(topHBox);
38            borderPane.setBottom(bottomHBox);
39            borderPane.setLeft(leftVBox);
40            borderPane.setRight(rightVBox);
```

(28) To run this program, do not forget to set the VM argument as shown at step (13).