CS 112 Programing II
Lab Session 11: Event-driven Programming

Objectives

I am sure you have been enjoying your experience working on GUI projects (i.e. JavaFX projects)! The experience is not only enjoyable but also *beneficial in helping you absorb some of the object-oriented programming concepts* that you have covered previously. In this lab session, you will add an element of interaction in your JavaFX application through using the concept of *event-driven programming*.

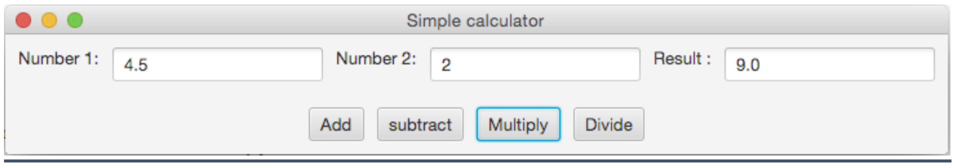
Upon finishing this lab session, you should be able to:

- Create different UI controls such as labels, text fields and buttons.
- Associate source objects with event handlers.
- Define even handlers using lambda expression.

Exercise 1

(Create a simple calculator)

In this exercise, you will be guided through the process of developing a simple calculator that performs addition, subtraction, multiplication and division. A screenshot of the program is shown below: *(exercise 15.4 from your textbook)*



The steps below guide you through your development:

1. Creating the Pane to Hold Labels and Text Fields

You can see that the *labels* as well as the *text fields* are arranged horizontally in a single row. This arrangement can be easily achieved by using **HBox** (**H** for **H**orizontal) layout pane. So, create an object from HBox class and set the spacing (gaps) between nodes to 10px. This task can be achieved by one or two lines of code. *Refer to the class diagram of HBox in pp. 558 of your textbook.*

2. Creating and Setting the Label and Text Field Controls

The first row of the application above contains three labels and three text fields. Create them with the appropriate parameters and add them all to the previously created HBox pane.

Hint: the code to create the 1st label is as follows:

```
Label lblNumber1 = new Label("Number 1:"); //note the chosen name for the object.
```

3. Creating the Pane to Hold the Buttons

Similar to step 1, you can see that the *buttons* are arranged horizontally in a single row. So, create another object from HBox class and set the spacing (gaps) between the nodes to 10px.

4. Creating and Setting the Button Controls

Create four buttons for the four operators with the appropriate text and event handlers and add them all to the previously created HBox pane. *The code below is used to create and set the event handler of the first button (add button):*

```
Button btAdd = new Button("Add");
btAdd.setOnAction(e -> {
     result = Double.parseDouble(tfNumber1.getText()) +
Double.parseDouble(tfNumber2.getText());
     tfResult.setText(result + "");
});
```

Why did we add an empty String to the result variable in the highlighted part above? Can you use another method to achieve the same thing?

5. Creating a BorderPane and Setting its Properties

You have learnt that the **BorderPane** divides its region into five areas: left, right, top, bottom and center. Any of the areas might be null, in which case no space will be allocated for it.

In this exercise, we will use this type of pane to place our hboxes that have been created previously to hold our UI controls.

Create a BorderPane, set its center and bottom nodes to hold the 2 hboxes that you have created previously and then adjust its margins to leave 10px around each hbox. *Margins can be set using the* `public static void setMargin(Node child, Insets value)` *method.*

6. Set the Scene and Show your Stage!

Finally, you have to show what you have done! In order to do that, create a scene, set the scene and the title of the primaryStage and show the primaryStage! *The code below does that:*

```
primaryStage.setScene(new Scene(borderPane));
primaryStage.setTitle("Simple calculator");
primaryStage.show();
```

Reflections: we have arranged our UI controls in HBox panes which have been then placed in a BorderPane. Can you achieve a similar layout of this application using other types of panes? What other features you can add to this simple calculator? *(do this at home! ☺)*