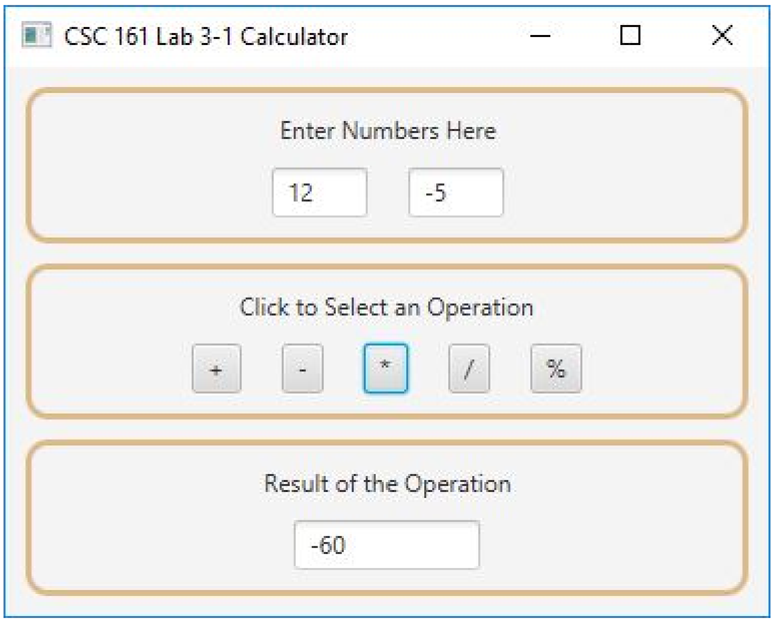
CSC 161 LAB 3-1 JAVA FX CALCULATOR

PROFESSOR GODFREY MUGANDA

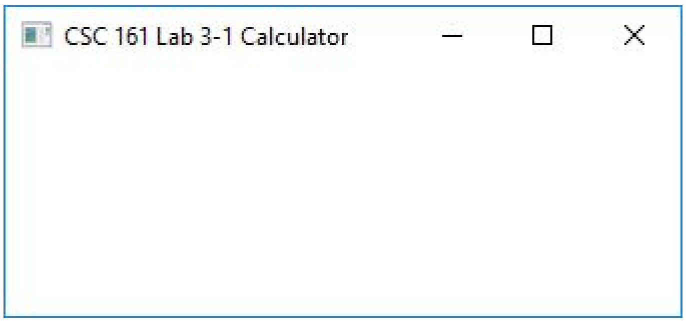
1. Introduction and Overview

In this lab, you are going to use JavaFX to create a calculator that can add, subtract, divide, multiply, and find the remainder of two integers. The user interface of the calculator should look like this:



2. Setting the Stage

Begin by creating a JavaFX application whose start method creates a `VBox` with a vertical gap of 10 pixels, uses it as the root of a scene object, sets the scene on the stage, and shows the stage. The resulting user interface should look like this:



3. Create the Active User Interface Components

The active UI components are those that will actually interact with the user. These are the components that need to accessible to the event handling methods that that respond to user events.

We need three text fields, one each of the x and y inputs, and one for displaying the result of the calculation. We also need an array of `Button` objects to represent the calculator operations:

```
        // Create the textfields
        TextField xTextField = new TextField();
        TextField yTextField = new TextField();
        TextField resultTextField = new TextField();

        // Create the operation buttons
        String[] operations = {"+", "-", "*", "/", "%"};
        Button [] operationButtons = new Button[operations.length];
```

Put this code at the very beginning of the `start()` method, before any of the other code. Add code that does the following:

(1) Set the preferred column count of the x and y input text fields to 3, and set the preferred column count of the result text field to 7.
(2) Make the result text field un-editable.
(3) Uses a loop to create the `Button` objects for the operations as elements of the `operationButtons` array. The button at position k in the `operationsButtons` array should have the text at position k in the `operations` array. Thus, the button at position 0 in the array should have text `"+"`, and the last button in the array will have text `"%"`.

If you are not sure about what you have done, you might want to call the preceptor, or the professor, to look over your code before proceeding to the next step.

4. Building the User Interface

Looking over the user interface, we can tell that we will need three panels. Each panel will have a label object vertically stacked on top of a row of UI components. We can achieve this by using a `VBox` to which we add a label and a `HBox` containing the row of one or more UI components. Note that all components in JavaFX are `Node` objects, so we are talking about a row of one or more `Node` objects.

We will write a class that extends `VBox` and does all this for us. Right-click on the package node in the Projects Pane and select the option to add a new class called `CalcPanel`. Use the following skeleton

```
import javafx.scene.Node;
import javafx.scene.layout.VBox;

public class CalcPanel extends VBox
{
    public CalcPanel(String caption, Node ... comps)
    {

    }
}
```

The notation `Node ... comps` in the parameter list for the `CalcPanel` constructor stands for one or more `Node` objects. Inside the constructor method, treat `comps` as though it were an array.

The constructor starts out by calling the superclass constructor to set the vertical spacing, and then sets a border and some padding to make things look nicer:

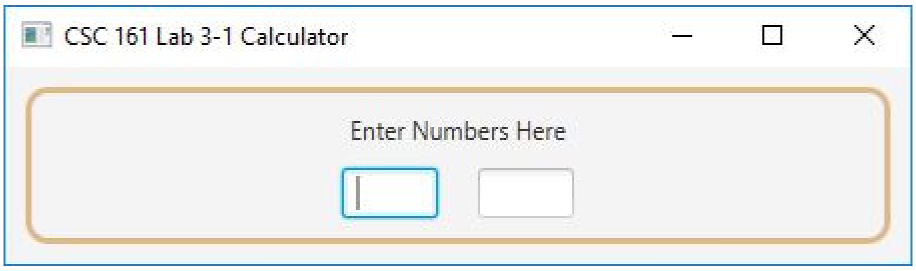
```
    public CalcPanel(String caption, Node ... comps)
    {
        super(10);
        // Set the alignment of this CalcPanel to center,
        // set a border, and set padding to 10 all around
        super.setAlignment(Pos.CENTER);
        BorderStroke borderStroke = new BorderStroke
                                    (
                                      Color.BURLYWOOD,
                                      BorderStrokeStyle.SOLID,
                                      new CornerRadii(10),
                                      new BorderWidths(3)
                                    );
        Border border = new Border(borderStroke);
        super.setBorder(border);
        super.setPadding(new Insets(10, 10, 10, 10));

        // Create a label with the caption.
        // The label should center its contents
        // *** Add code here ****


        // Create a HBox with hgap 20 and add all the comps to
        // the HBox, and set the alignment of the HBox to center
        // *** Add code here ****

        // Add the label and the HBox to this CalcPanel
        // *** Add code here  ****
    }
```

5. Add an Instance of CalcPanel for the Input Text fields

Returning to the `start` method, add the code

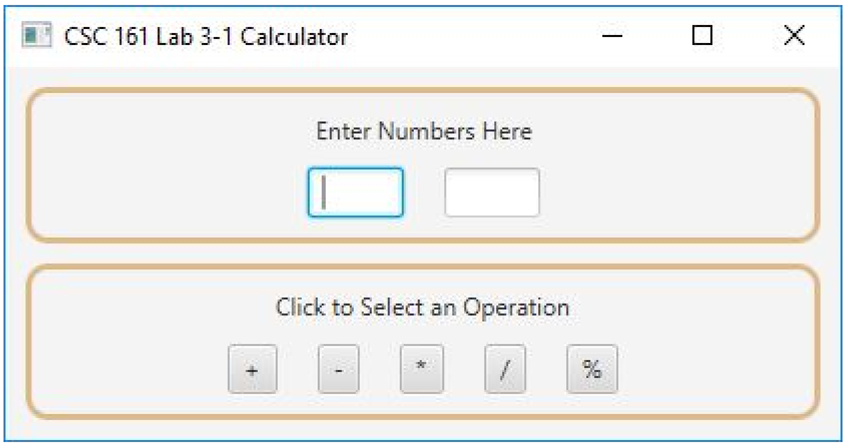
```
   CalcPanel inputsPanel = new CalcPanel("Enter Numbers Here",
                                         xTextField, yTextField);
```

to create a panel to hold the input text fields and their label, and add it to the `topLevelVBox`. Run the program to check up on your progress. You should a user interface like this:



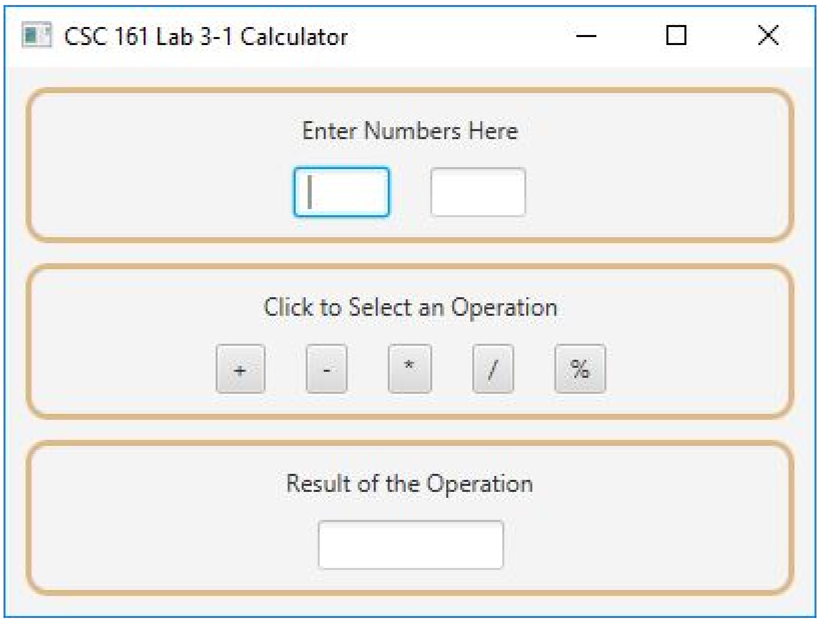
6. Add an instance of CalcPanel for the operation buttons

Add code to create a panel that contains the operation buttons and their label. The first argument will be the string. Pass the array of operation buttons for the second argument. When you run the program, you should get something like this:



7. Add an instance of CalcPanel for the result text field

Add code to add panel to hold the result text field and its label. The result should be as you see here.



8. Add an Event Handler

So our calculator is almost done, and that is cool. The only thing left to do is to create an event handler for `ActionEvent` on the operations buttons. At the very end of `main`, after all the other code you already have, add this code

```
// Add event handler
EventHandler<ActionEvent> operationsHandler = evt ->
{

};
```

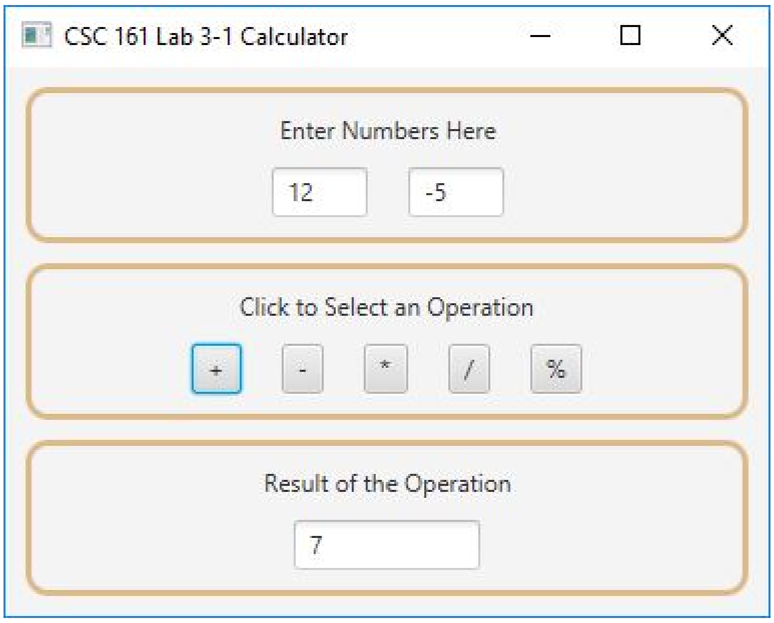
Your event handler should start out with

```
  Button b = (Button)evt.getTarget();
```

to retrieve the button that was clicked. Then, you should use the `getText()` method on the button to determine the operation specified by the button. *You must use a switch statement.* The event handler retrieves the inputs, performs the operation, and sets the result text field.

Finally, use a loop to call `setOnAction(operationsHandler)` on every operation button.

At this point the calculator is done and is fully usable:


CSC 161 Lab 3-1 Calculator
Enter Numbers Here
12
-5
Click to Select an Operation
+
-
*
/
%
Result of the Operation
7