Golisano College of Computing and Information Sciences
School of Information

ISTE-121
Lab 02: Handling Events, Menus & Text Areas

Target CLO1: Implement interactive graphical user interfaces using the event model

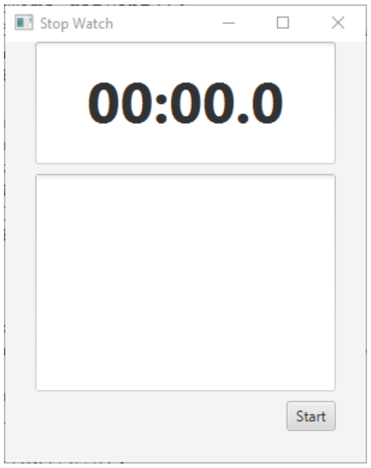
Overview

The purpose of this lab is to handle events from buttons in a number of ways.

Exercise 1 - Stop Watch Revisited (2 Points)

Part 1:

Starting with StopWatch.java (LAB's today's downloads) create a new GUI that looks like:

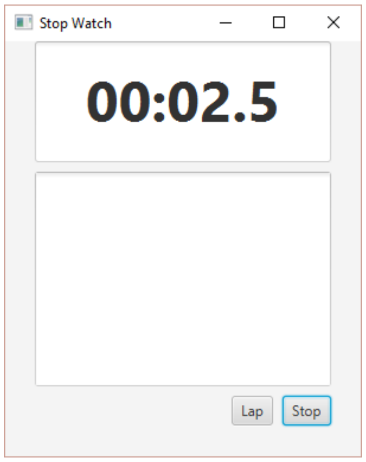


My window is now 300x350. The root layout is VBox with three FlowPanes in it - fpTop, fpCenter and fpBot.

The TextField and TextArea are the same width (250 pixels) set with the setPrefWidth method of each.

The top two FlowPanes are centered (setAlignment(Pos.CENTER)).

The Bottom FlowPane is right aligned (setAlignment(Pos.BOTTOM_RIGHT)). In this FlowPane are the two buttons (one is invisible - see below) plus a dummy Label of 5 spaces to keep the buttons from being crammed up against the right border.



Now, when the Start button is clicked, the window will look something like this:

Note, a Lap button has appeared (initially it is there, but not visible - see the Javadocs). When you click Lap, the current time is posted to the TextArea. If you click several times, you get a list of lap times in the TextArea.

Clicking Stop leaves the final time in the counter and the lap times in the TextArea. Also, the Lap button disappears (again, see the Button method setVisible).

Clicking Start again will clear the counter and the TextArea for another run.

Part 2:

Now, change the handler for the Lap button to an anonymous inner class. This means that **handle** in the StopWatch Application class can be changed to remove the case for "Lap".

Submit your code (the .java file(s) only) to the Lab02 Assignment folder when Exercise 1 is working correctly.

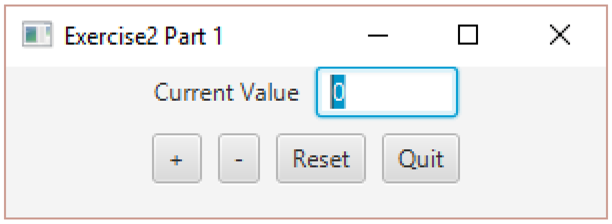
Exercise 2 - Handling Events (3 Points)

Overview

The purpose of Exercise 2 is to use a separate listener class (Part 2).

Part 1: (Counter.java)

Create a class, Counter.java, that will draw the following GUI:



Rather obviously, clicking '+' should add 1 to the value, clicking '-' should subtract 1 from the value. Clicking 'Reset' should set the value to 0. Clicking 'Quit' should terminate the program.

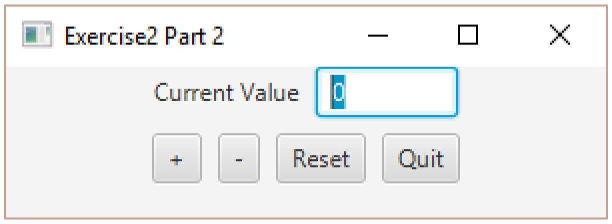
For each button, include the line:

```
    button.setOnAction(this);
```

and include an ActionEvent handler in your program to provide the required functionality.

Note: you may need to resize horizontally to have the complete title displayed.

Part 2: (Counter2.java)



Modify your solution from Part 1 to implement the action listener for the **Reset** button in a separate class name **ResetListener** so that the functionality of the program is the same as Exercise 2 Part 1.

What are the parameters to the constructor for this class?

Place your answer as a comment at the bottom of your Part 2 code (submitted below).

Hint: what components will you need to access in your new class? For each component, you will need to declare an instance variable in your new class to store a reference to that component. The constructor is the only way to pass the references to the required components into your class.

If ResetListener needs to access or mutate any other attributes of the Counter2 class, then those accessors/mutators will have to be written and the Counter2 object reference will have to be passed to ResetListener so that it can call those methods through the reference. Is this the case in your program? It may or may not be, depending on your design.

Note: you may need to resize horizontally to have the complete title displayed.

Submit your code (the .java file(s) only) to the Lab02 Assignment folder when Exercise 2 is working correctly.

RIT | Golisano College of Computing and Information Sciences
School of Information

Exercise 3 – Menus & Text Area (5 Points)

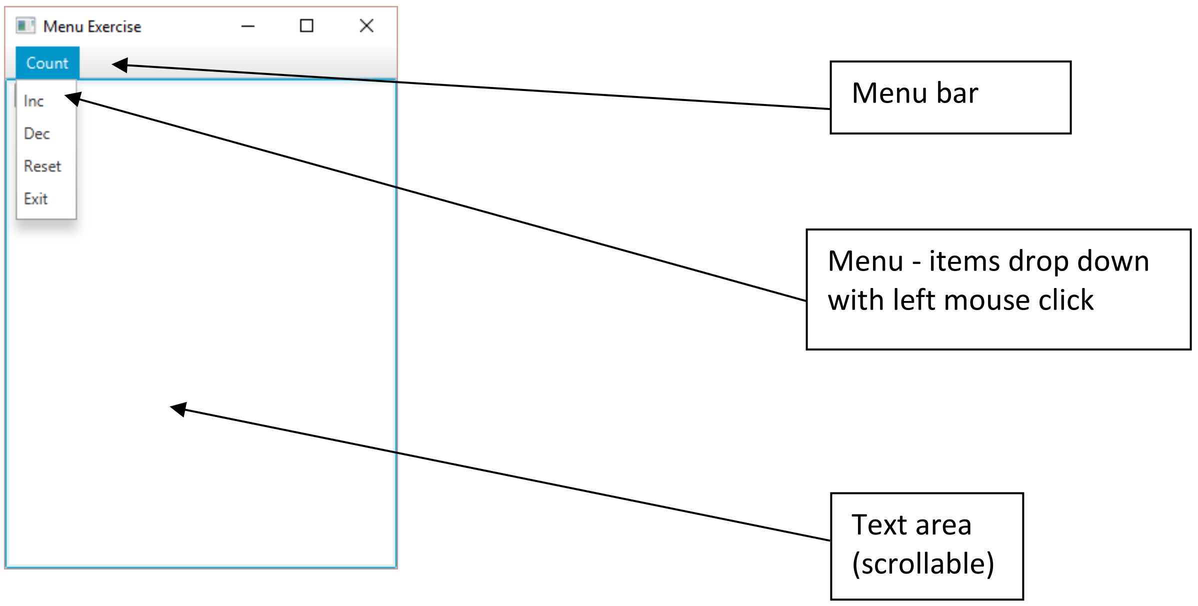
<u>If you do not complete this exercise during the lab period, you need to complete the work outside of the lab period and submit the completed work prior to the lab due date.</u>

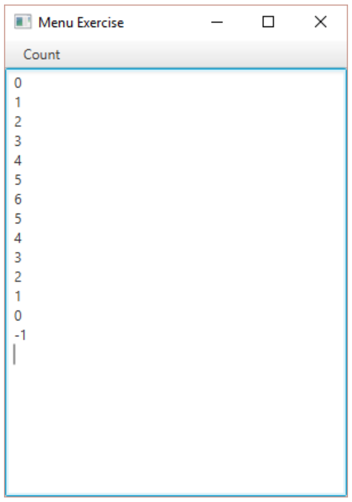
Overview

The purpose of this exercise is to create an interactive Java GUI application which requires the use of a separate class to handle the events of the menu items.

MenuExercise.java

This application will behave in a similar fashion to the previous practice exercise, except that there are no buttons and the options are given in the menu as shown below.





```
0
1
2
3
4
5
6
5
4
3
2
1
0
-1
```

Step 1: Create a text area
Create a class named MenuExercise which extends Application to build the framework for this program. Use a 300x400 window and place the TextArea directly in it (that is, **not** in a Pane).

Set the PrefHeight of the TextArea to 400.

Step 2: Create a menu bar
Create a menu bar and add it to the root layout.

Step 3: Create a menu
Create one menu named **Count** as shown above.

Compile and run your program to see that frame appears as above. At this point, the menu will not respond to a left click.

Step 4: Add menu items to the menu
Add four menu items shown by bold text, to the menu in the following order:

- **Inc** - increment the count
- **Dec** - decrement the count
- **Reset** - set count back to zero
- **Exit** - terminate the program

Step 5: Create an ActionEvent handler

This may be the Application itself, or another class.

Step 6: Set the OnAction handlers
Write the statements to add your ActionEvent handler to each menu item.

Step 7: Implementing handle
In handle, add a case for each menu item. Start with Inc, only, and see that it is working. When Inc is working, go on to the other menu items.

What happens when all of the lines of the text area are full and another line is written to the text area?

Place your answer as a comment at the bottom of your Part 3 code (submitted below).

Submit your code (the .java file(s) only) to the Lab02 Assignment folder when Exercise 3 is working correctly.