CST 8221 – JAP, Assignment #1, Part 2

Due Date: prior or on October 30th, 2015

Earnings: 7% of your total course mark (plus up to 5% Bonus)

Purpose: Developing GUI based Calculator Application

The purpose of this simple yet comprehensive assignment is to give you the enjoyment of exercising your programming skills and the delight of applying your knowledge of how to build a relatively simple GUI based Calculator Application. The assignment is based on the material covered in lectures, lab exercises, and hybrid activities. You will find Chapter 14 of the textbook to be very helpful.

Requirements Specification:

In this part of Assignment #1 you are to complete the implementation of the Calculator Application. Your Calculator Application must exhibit the behavior described below.

Tasks:

In this part of Assignment#1 you must add a ***CalculatorModel*** class, and complete the implementation of the event handling in the ***Controller*** class or classes.

Class *CalculatorModel*

The class ***CalculatorModel*** class is responsible for performing the calculations. The CalculatorModel class code must provide at least the following:

- set methods for setting the string operands.
- set methods to set the operation and the operational mode (Integer or Float)
- set method to set the floating-point precision.
- get methods to return the results from the operation in formatted form.
- set and get method for the error state. If the result of the calculations is ***NaN***, ***Infinity***, or out of range the error state must be set to *true*.
- *performCalculations()* method that performs perform the calculations and sets the result depending on the current operational mode (Integer or Float)

Controller class(es)

In this part of the assignment you must complete the implementation of the event handling. All components used in the Calculator GUI generate an action event. You can use only one instance of the ***Controller*** class (see Assignment 1 Part 1) to handle all action events in one place, or you can use more than one instance to handle different action events in different places.

If for some reason you need to handle events different from ***ActionEvent***, you must implement the event handler(s) as private inner classes in the same way as the Controller class. You must name them appropriately: The name of the class must contain the word Controller. For example: ***XYZController***.

The implementation of the event handler(s) must provide behavior identical to the behavior described in the ***Important checkpoint*** section. No calculations must be performed in the *Controller* class(es). All calculations must be carried out by the *CalculatorModel* class.
In the description of the checkpoints the notation [ ] means that a specific non-numeric or non-arithmetic operation button must be clicked. For example, [Int] means that the Int checkbox must be clicked.
The sequence 2 + 3 = means that the following buttons have been clicked in the specified order: button 2, button +, button 2, button =. When 2 is clicked the text display must show 2; then when + is clicked the display must still display 2; then when 3 is clicked the display must show 3; and finally when = is clicked the display must show the result which in this case is 5.00 in Float (**F**) mode and 5 in Integer (**I**) mode.

Important checkpoints:

1) When [Int] is checked the **.** button must be disabled and its background must be changed to gray. The label must display a black letter **I** on a green background. When [Int] is unchecked the **.** button must be enabled and the label must display a black letter **F** on a yellow background.
2) 2 + 2 = must display 4 in *Integer* (**I**) mode and 4.00 in *Float* (**F**) mode. Do not forget that at launch the Calculator is in **F** mode and the precision is **.00**
3) 2 * 2 = + 4 = must display 8 in **I** mode or 8.00 in **F** mode
4) 2*= must display 4 or 4.00
5) 2*/= must display 1or 1.00
6) 2/*= must display 4 or 4.00
7) 2 + 2 = must display 4 or 4.00. If 2 is clicked after =, must display 2
8) 12 [←]+2 = must display 3 or 3.00
9) 2 [±] must display -2. If followed by [±], must display 2
10) -123[←][←] must display -1
11) -123[←][←][←] must display 0 in **I** mode ([Int] checked)
12) -123.0 [←][←] must display -123
13) -123.0 [←][←][←][←][←] must display 0.0
14) 1 / 0 must display -- (two minus signs) in the text display and the label (mode-error display) must display a black **E** (aligned in the center of the label) on a red background. After error, the calculator GUI must not respond to any button except **C**, **Int**, and the precisions radio buttons**.** The test display and the mode-error display must not change until C is pressed. Pressing C must display 0 or 0.0 depending on the current mode of operation, and change the label correspondingly to **I** or **F** with the appropriate background color and alignment.
15) 0 / 0 = must display error (see 14)..
16) -4 / 2 = must display -2 or -2.00
17) 2.5 [Int] must display 2
18) 2.5 [Int] + 2 = must display 4
19) 2 [Int] must display 2
20) [Int] 2 [C] must display 0
21) [Int] unchecked (**F** mode displayed) 2 [C] must display 0.0
22) [.0] 1 / 3 = must display 0.3 in **F** mode and 0 in **I** mode
23) [.00] 1 / 3 = must display 0.33 in **F** mode and 0 in **I** mode
24) [Sci] 1 / 3 = must display 3.333333E-01 in **F** mode and 0 in **I** mode
25) [Sci] 1 / 3 = + 2 =must display 2.333333E+00 in **F** mode and 2 in **I** mode

You should try other calculations and combinations and make sure that they work properly.
The calculator must not accept improperly formatted real numbers like 0..0, . , ..5, or …
In Integer mode the **.** button must be disabled (must turn gray and must not respond to clicks).
In Float mode the **.** button must be enabled.
The Calculator must not print on the console, and must not crash or generate exceptions on input or calculations.

Bonus Tasks

Bonus 1 (1.0%)

Incorporate a progress bar in the Calculator splash screen. The progress bar should initially read "Loading Calculator. Please wait…", and then show growing color bar. See HybridAct#6 for details of how to work with progress bars. Create an internal delay in your splash screen class.

Bonus 2 (1.5%)

Make the calculator accept expression-like calculations. Example:

2+2+2= must display 6 (or 6.00)

Note: 2+2+ (displays 4) 2 = displays 6

Bonus 3 (2.0%)

Make the calculator accept keyboard input alongside with the mouse input. The text field must stay disabled i.e. it must not accept keyboard directly. The get full credit for this bonus you must use Input and Action maps.

NOTE*: In order to receive credit for the bonus tasks your calculator must operate according to the specifications and your bonus tasks must be reflected in the test plan.*

What to Submit:

Paper submission:
No paper submission is required for this assignment

Code submission:

Compress in one **.zip** file all **.java** files, **.class** files, and **image** file(s)**.** Use the **Assignment 1 Part 2** upload link in the **Assignment** folder and upload the submission ***zip*** file on Blackboard prior to or on the due date. The name of the zip file must have the following structure: Student's family name followed by the last three digits of the student ID number followed by _JAP_A1P2. For example, *Ranev123_JAP_A1P2.zip*.

The submission must follow the course submission standards. The ***Assignment Submission Standard*** and the ***Assignment Marking Guide*** are posted on Blackboard in the Course Information folder.

Enjoy the assignment. And do not forget that:

"2 + 2 is always 4 except early in the morning." Anonymous Swing Programmer