CST 8221 – JAP, Assignment #1, Part 2

Due Date: prior or on March 16, 2018

Earnings: 7% of your total course mark (plus up to 4% 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 for a Calculator Application. The Calculator you are to build will be a functional replica of your Windows calculator. The assignment is based on the material covered in lectures, lab exercises, and hybrid activities. You will also find Chapter 12 of Textbook 1 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 to the *calculator* package, 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:

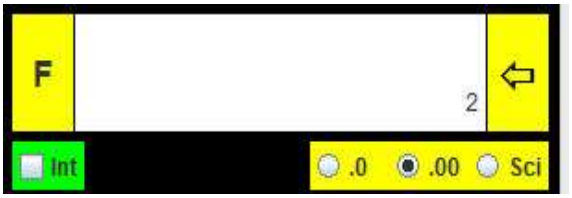
- set methods for setting the operands (both operands must be of String type).
- set methods to set the arithmetic operation and the operational mode (Integer or Float)
- set method to set the floating-point precision.
- get method to return the result from the operation in formatted form based on the current precision. The result should be implemented as a "*virtual filed*".
- set and get method for the *error state* field of the class. If the result of the calculations cannot be determined (illegal operands, operators or mode; result is ***NaN***, ***Infinity***, or out of range) the error state must be set to *true*.
- *a* private *calculate()* method that calculates the result based on the current operands, arithmentic operation and 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.

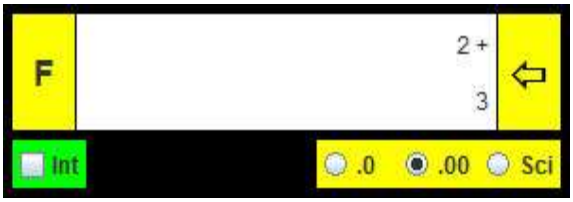
If for some reason you need to handle events different from ***ActionEvent***, you must implement the event handler(s) as private inner classes in the same way as the Controller class. You must name them appropriately: The name of the class must contain the word Controller. For example: ***XYZController***.

The implementation of the event handler(s) must provide behavior identical to the behavior described in the ***Important test checkpoints*** 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 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 calculator display2 must show 2; then when + is clicked display2 must still display 2 but *display1* must show 2 +; then when 3 is clicked *display2* must show 3; and finally when = is clicked *display2* must show the result which in this case is 5.00 in Float (**F**) mode and 5 in Integer (**I**) mode and *display1* must be cleared. It should operate in the same way your Windows calculator operates except for number of precision digits displayed after the decimal point.



2 clicked

+ clicked



3 clicked

= clicked

After = has been clicked if a number is clicked, display2 must display the number. Also, after = has been clicked the backspace button ⇦ is not operational. The backspace button is operational only when numbers (operands) are entered.

8 clicked

8 clicked again

backspace ⇦ has been clicke

Important test checkpoints:

1) When the user clicks on the **[Int]** checkbox the **. button** must be disabled and its background of the button must be changed to new *Color(178,156,250)*. The *mode/error display* label must display a black letter **I** on a green background. The calculator must operate in integer mode. Then if one of the floating-point precision radio buttons is clicked the **. button** must be enabled and blue again. The label must display a black letter **F** on a yellow background. The calculator must operate in Float mode.

2) 22 + 22 = must display 44 in *Integer* (**I**) mode and 44.00 in *Float* (**F**) mode. Do not forget that at launch the Calculator is in **F** mode and the precision is **.00.** If 2 is clicked after =, must display 2 (display2).

3) 2 * 2 = 4.00 (display2) [+] (*display1*: 4.00+, *display2*: 4.00) [5] (*display2*:5) [=] must display 9.00 (*display2*) in **F** mode or 9 in **I** mode.

4) 2+= must display 4 or 4.00 (*display2*) [=] displays 6 or 6.00 [=] displays 8 or 8.0

5) 2*= displays 4 or 4.00 [=] displays 8 or 8.00 [=] displays 16 or 16.00

6) 2*/ (*display1*: 2 /) [=] must display 1or 1.00 (*display2*) [=] displays 0 or 0.50

7) 2/*= (*display1*: 2*) must display 4 or 4.00 (*display2*)

8) 12 [⇦]+2 = must display 3 or 3.00

9) 2 [±] must display -2 (*display2*). If followed by [±], must display 2 (*display2*)

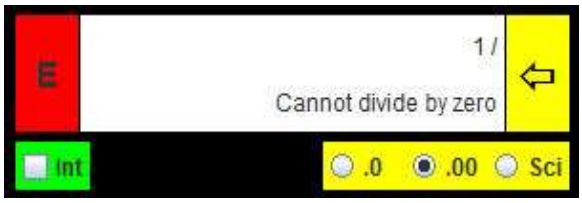
10) -123[⇦][⇦] must display -1

11) -123[⇦][⇦][⇦] must display 0 *(display2*) in **I** mode ([Int] checked)

12) -123.0 [⇦][⇦]] must display -123

13) -123.0 [⇦][⇦][⇦][⇦][⇦]must display 0.0 (*display2*)

14) 1 / 0 = must display the following:



Additionally, after an error, the calculator GUI must not be responsive to any button (buttons must not be disabled) except **C**, **Int**, and the **precisions radio buttons.** The calculator display and the mode/error display must not change until **C** is pressed. Pressing **C** (clear) must display 0 or 0.0 depending on the current mode of operation, and change mode-error display correspondingly to **I** or **F** with the appropriate background color and alignment.

C clicked in mode F

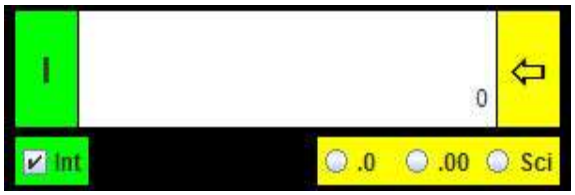
C clicked in mode I

15) 0 / 0 = must display the following:

The calculator must behave in the same way as explained in 14)

16) -4 / 2 = must display -2 or -2.00

17) 2.5 [Int] or 2.5 [+ - / *] [Int] must clear the calculator display



18) [Int]2[radio button] or [Int] 2 [+ - / *] [radio button] must clear the calculator display

19) [.0] 1 / 3= must display 0.3 in **F** mode; [Int]1 / 3 must display 0

20) [.00] 1 / 3 = must display 0.33

21) [Sci] 1 / 3 = must display 3.333333E-01

22) [Sci] 1 / 3 = + 2 = must display 2.333333E+00 in **F** mode;
[Int] 1 / 3 = + 2 = must display 2

23) At any time during operation [C] must clear the calculator display (see 17 and 18)

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 …
The calculator must not display numbers that are longer than the current width of the display.
The calculator must not display incorrect results.
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 (1.0%)

According to the assignment specifications in order to change the operational mode of the calculator from Float (which is the default mode) to integer the user must click on the **[Int] checkbox** once. Then if the user wants to return back to the floating-point mode, the user must click on one of the precision radio buttons. The design specification does not indicate what should happen if the user clicks on the checkbox twice or multiple times. The purpose of this bonus task is to rectify this situation.

Modify your event handling code so that the calculator works as specified before but additionally if the user clicks on the checkbox (the calculator goes into integer mode) and immediately clicks the checkbox again, the checkbox becomes unchecked, the mode returns back to floating-point, and the precision radio **.00 button** becomes selected. If the checkbox is clicked for the third time the calculator returns to integer mode.
Note: Currently only four components are available to the Controller inner class - *JTextField display1 and display2*, *JLabel error* and *JButton dotButton*. You may need to add some more.

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 , and finally, followed by your lab section number (s301, s302). For example, *Ranev007_JAP_A1P2_s301.zip*.

The submission must follow the course submission standards. The ***Assignment Submission Standard*** and the ***Assignment Marking Guide*** are posted on Blackboard in the Getting Started folder. Test Plan is not required for this assignment (except for the bonus tasks).

Enjoy the assignment. And do not forget that:
“2 + 2 is always 4 except early in the morning.” Anonymous Swing Programmer
CST8221 –JAP, 16 February 2018, S^R