SE 317: Lab 8
The Scientific Calculator
Weight: 15%

In this assignment you will build and test a simple GUI calculator using the MVC pattern or any of it's simplified versions. The Observer class in java is a good example.

The GUI side (the observer) will display a graphical image of your interactive calculator

The model side (the observable) will receive input from the user's interaction with the GUI displayed on the screen, performs the selected operations on them, and return the results to the screen to be displayed.

Your calculator will have the following features and functions:
0-9 digit buttons
The decimal point.
4 basic math operations: add, subtract, multiply, divide
2 advanced operations: square and square root (use text or a suitable icon on each button)
Memory functions:
Your calculator will have one memory to store a value and add or subtract the result of a **successfully executed** operation or reuse this value as an operand in a new operation.

M+ to add the output of a successfully executed (and displayed) math operation to the value currently stored in its memory (negative values allowed).
Note: A math operation is executed when you enter one (or two) valid numbers (operands) and a valid operation then press the = sign to execute.
For example, the following operations are valid (operands in blue, operations in red, result in **black:**

1+1 = 2
4.2 * 2 = 8.4
16 (Square Root) = 4
8 (square) = 64

Some operations take two operands (like +, -), other operations take one operand (like square root and square)
Only valid results can be added to memory. i.e. You can only store the results of executed operations. Entering a number (like 3232) then pressing M+ will give an error as below:
3232 M+ → "Error"

M- to subtract the number on the calculator screen from the number stored in its memory, negative values allowed
M-Recall to recall the number in the memory and use it as an input operand to current calculation
M-Clear to delete the memory contents

Delete: will delete the last digit or the decimal point entered by the user. Multiple **Delete** operations will delete multiple digits in reverse order (including the decimal point). For example
If you enter 1234.5 then press the **Delete** key three times, you will delete 5, then the decimal point then the 4, You will end up with 123 as your current operand.
Note: Only the operands and the result are displayed on the calculator screen. The operations are **NOT** displayed
Note: When pressed, each operation buttons on the screen **changes its visual appearance** to show it is pressed and is currently active (they get bold, or change color to show being active).

Each time you press an operation, the first operand remains on screen, and the operation button changes it's visual look to show it is being active. When you start typing the second operand (if the operation needs two operands), the screen refreshes, showing the second operand as you type it. For example if you want to add the two numbers [**111111 + 222222** =], your display will have the following three consecutive values displayed on the screen (each screen will replace the previous one:
111111 (the display will show 111111)

+	(the display will continue to show 111111, and the + button will change its look to indicate being active)
222222	(the display will refresh, showing 222222)
=	(the display will refresh, showing the result 333333 (or an error for invalid operations like division by zero or an out –of-range value)

Output on the screen: 333333

Notes: You don’t need to use colors on the display
In your GUI, each button will have two images, one when in idle state, and one when depressed or active (or both). That includes all buttons on the GUI (digits, operations, etc.)
You only display the digits, the decimal point, and the negative sign on your display. You don’t need to display the active operations on the screen

Clear: will reset the calculator to a blank start, including, clearing the memory.

Deliverables:

1- Source code
   a. Use exception handling and add appropriate comments to your code
   b. Interaction I/O will be done using interactive GUI
   c. Add a Read-me file of how to deploy and use your assignment
   d. Use MVC or the simple Java Observer class for implementation

2- A comprehensive teat plan of your calculator code and interface.

   a. **The Model** will be tested at the code level by simulated function calls using java test code, with screenshots of the output (like all previous assignments)

   b. **The GUI** will be tested by just providing **the test cases** to be run on the interactive GUI with the expected results. You will need to run each UI test as a sequence of GUI inputs (manually) and record the final output” of each test, next to the expected one

For example:

Division by zero test case:
Input sequence:
222 / 0 =
expected result: Error
Actual result: Error

Out of range test case…

Etc.

Additional details and Q&A will be discussed in class.

Good luck,