CSCD 300 Homework 5 and Homework 6 (Project)
Evaluating Numeric Expressions Using Stack

Total 100 points

Turn in: On EWU Canvas, CSCD300→ Assignments→Hw56→Submit.
Please put all your source code (.java) files and output files together into a zip file. Name the zip file with your last name followed by first initial plus hw56.zip. For example, smithjhw56.zip is for John Smith.

If you forget to include your source code in the zip file, you get a zero credit for this homework. If your code shows a compile-time error, you get a zero credit. If you turned in a corrupted file, or a file that cannot be opened, you get a zero.

Problems Description
Based on the materials we discussed in classroom, you are required to implement the pseudo code regarding converting infix expression into postfix expression, then evaluating the postfix expression to produce a single result. Based on the demonstration I did in classroom, you will solve two problems, **the first one (hw5) is required, while the second one (hw6) is optional but considered as bonus.**
Problem 1) →(hw5)

a) Read in the value of the symbols from input file **hw5_input2.txt**, and represent each symbol with a **double** precision value (Java double type).
b) Read in infix expressions from **hw5_input3.txt**. These infix expressions use the symbols defined in hw5_input2.txt. Please do not change either of the input files.
c) Convert each infix expression from step (b) into postfix expression.
d) Evaluate each postfix expression in step (c) using the value of symbols in hw5_input2.txt to yield a single result of **double** type, if the original infix expression has a correct syntax and in correct format. Otherwise, your program has to show an error message. The error message is thrown by your program based on the evaluation result for the current expression, which is shown below at the end of this section. You **cannot** hardcode your error message!
e) Output the results for each expression in step (c) and step (d) into a file, named **hw5_output.txt.** The format and part of the output file **hw5_output.txt** are shown below. Format is "original infix expression - - > corresponding postfix expression - - > evaluated postfix in double type".

```
(((A + B) - (C - D)) / (E - F)) --> A B + C D - - E F - /  --> -2.0
(((A))) --> A  --> 8.0
(A) --> A  --> 8.0
((A --> Parens Not Match Error!
(B --> Parens Not Match Error!
……
((  ))--> Empty infix expression Error!
……
A*B - C ^ C ^ D --> A B * C C D ^ ^ -  --> -9.000810786714514
A B - C ^ C ^ D --> Infix Syntax Error!
……..
(( A - B * C) ^ D ^ E) ^ ( F / G * H + I ) --> A B C * - D E ^ ^ F G / H * I + ^  --> 0.0
……
A * ( B / C) + D ^ A - B --> A B C / * D A ^ + B -  --> 65535.857142857145
```

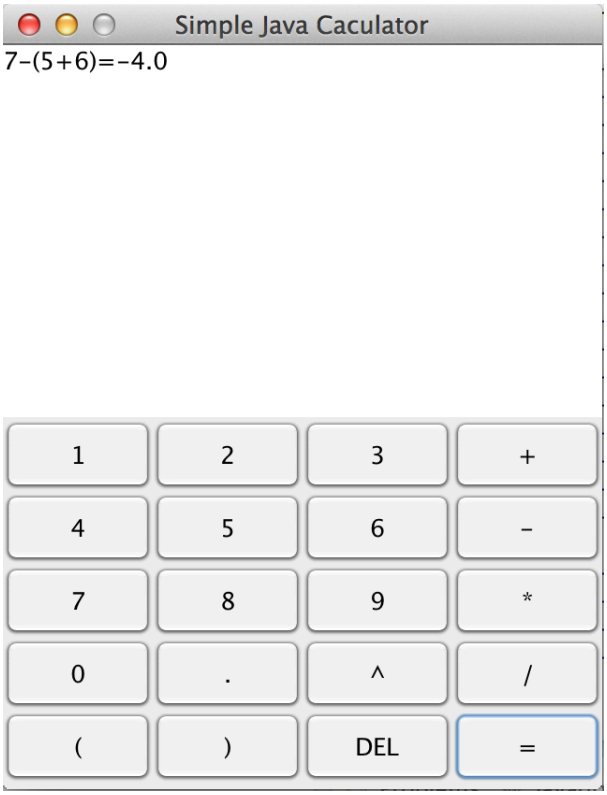
Problem 2) →(hw6) (Extra Credits as Bonus 20 points)

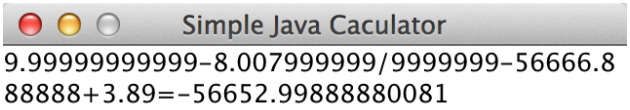
Based on the startup GUI program and demonstration in class, you have to design and implement a calculator with a GUI interface in Java,

1. The calculator supports operators +, -, *, /, ^(exponential), and any combinations of parentheses.
2. Your calculator has to support fractional number input, i.e. provides a decimal point on the panel.
3. Your calculator has to correctly evaluate any infix expressions, producing a result in **double** type or an output of error message if the expression contains error.

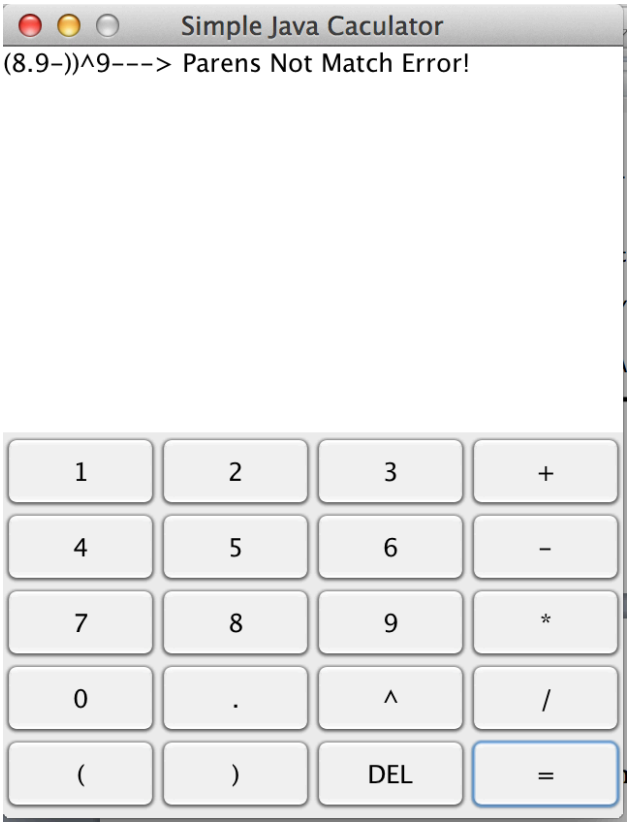
Hint: please carefully design your solution to problem 1, so that you can reuse the code in solution one.

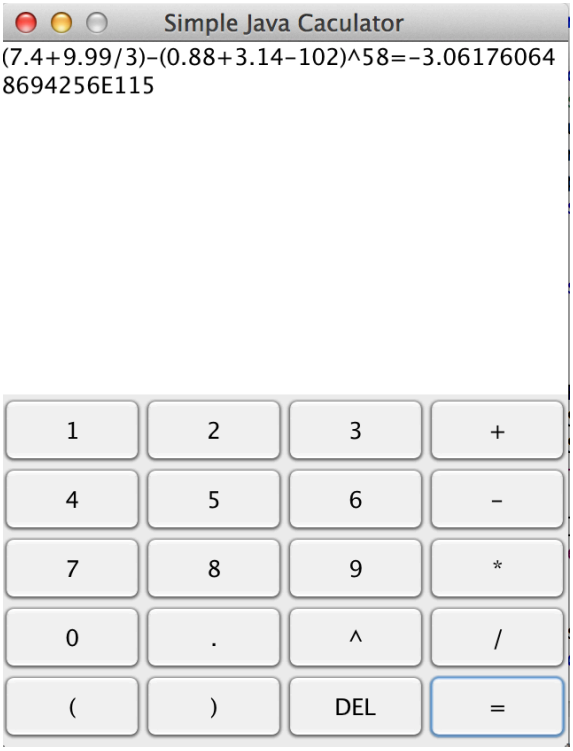
Some test cases are provided below,

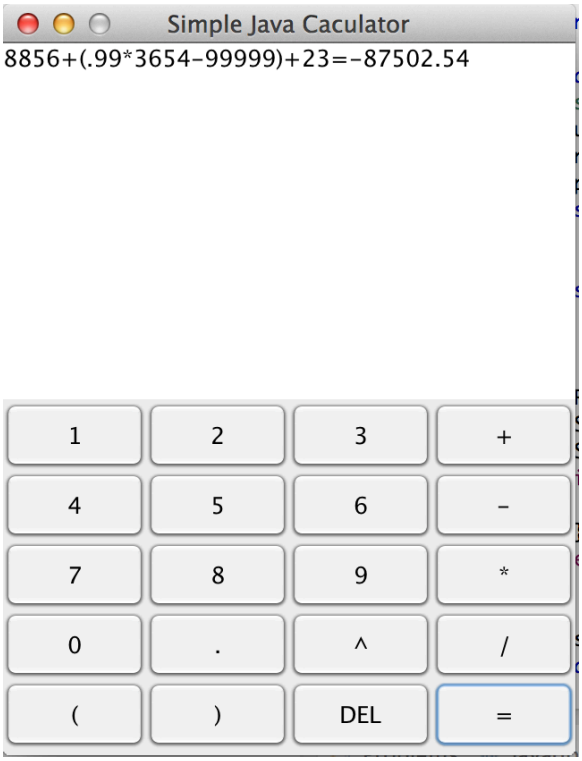



Simple Java Caculator
9.99999999999-8.007999999/9999999-56666.8
88888+3.89=-56652.99888880081


1
2
3
+
4
5
6
-
7
8
9
*
0
.
^
/
(
)
DEL
=


Simple Java Caculator
(8.9-))^9---> Parens Not Match Error!
1
2
3
+
4
5
6
-
7
8
9
*
0
.
^
/
(
)
DEL
=


Simple Java Caculator
(7.4+9.99/3)-(0.88+3.14-102)^58=-3.06176064
8694256E115
1
2
3
+
4
5
6
-
7
8
9
*
0
.
^
/
(
)
DEL
=


Simple Java Caculator
8856+(.99*3654-99999)+23=-87502.54
1
2
3
+
4
5
6
-
7
8
9
*
0
.
^
/
(
)
DEL
=

This infix expression is (), left parens and right parens, not 0.