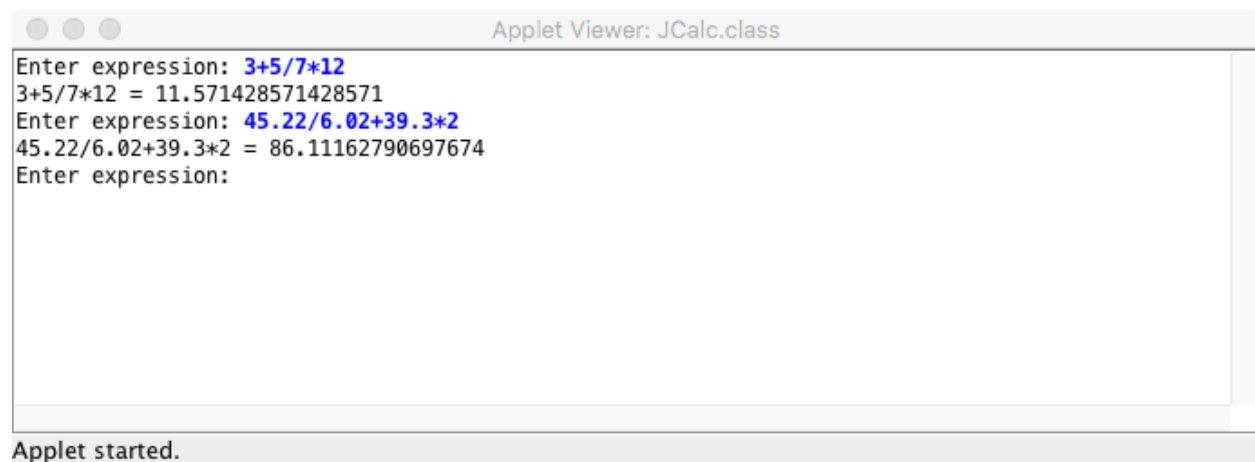


**Department of Electrical and Computer Engineering**  
**ECSE 202 – Introduction to Software Development**  
**Assignment 5 - Adding an Interpreter**

due date November 21, 2017 at 5:00 pm.

### Problem Description

In the previous assignment, you wrote a Java program that prompted the user for an infix expression and converted the result to postfix, displaying the result in the console window. In this assignment, instead of simply displaying the postfix, you will implement a simple interpreter that evaluates the expression, and returns a numeric result: For example:



```
Applet Viewer: JCalc.class
Enter expression: 3+5/7*12
3+5/7*12 = 11.571428571428571
Enter expression: 45.22/6.02+39.3*2
45.22/6.02+39.3*2 = 86.11162790697674
Enter expression:
Applet started.
```

### Method

The simplest approach is to keep your existing implementation for Assignment 4 and add in the additional code to implement the interpreter and print the result, as shown above

Notes:

1. Even though the arguments are integers, the calculations should be implemented using floats and the result displayed to the full precision of the float datatype. (In Assignment 6 we'll ask you to have a chooser that allows the user to set the number of decimal places used).
2. You are not required to use the ACM classes. Any input method will suffice provided that it returns a string containing the infix input.
3. Another bonus will be awarded for correct implementation of parentheses. (15)
4. Finally, a bonus will be awarded for correct implementation of the unary minus (-) . (15)

Max grade on this assignment: 130/100.

### Test Cases:

Enter expression:  $34/5+16*2$

$34/5+16*2 = 38.8$

Enter expression:  $5+9.27/1.4*3+2/3$

$5+9.27/1.4*3+2/3 = 25.53095238095238$

Enter expression:  $1.1-2.2*3.4/5.6$

$1.1-2.2*3.4/5.6 = -0.23571428571428577$

Enter expression:  $6/7+3/4+1/2$

$6/7+3/4+1/2 = 2.107142857142857$

Enter expression:  $9.8+3*6.7/2-4$

$9.8+3*6.7/2-4 = 15.850000000000001$

Enter expression:

### Instructions

Write the program as described in the preceding sections. It should operate interactively and be able to replicate the output from the test cases shown above. To obtain full marks your code must be fully documented and correctly replicate the test cases.

Your submission should consist of 5 files:

1. Stack.java - stack class
2. Queue.java - queue class
3. listNode.java - node object
4. JCalc.java - program
5. JCalc.txt - output

Upload your files to myCourses as indicated.

Note: You may decide to include other classes in your program, but the main class must be called Jcalc.

fpf/November 6, 2017