Assignment 8: Java

You have already seen Object-Oriented Programming in CSC211/212 using C++. For this assignment we are using Java, if you need help with Java, let us know.

Although the program might look overwhelming, you only have to add less than 10 (more like 5) lines of code to make it work.

Do Chapter 17, Exercise 3 only:

Modify the **`CalcParser`** class from the previous exercise(refers to the downloadable JavaCalc.zip code from Attachment). Instead of evaluating the expression, make it produce a string containing commands to evaluate the expression on a stack machine. You should be able to test it with the same main method as before. For example, if you give the command **`java CalcParser 1+2*3`**, the output should be a sequence of stack commands such as this:

```
push 1.0

push 2.0

push 3.0

multiply

add
```

For the command **`java CalcParser (1+2)*3`**, on the other hand, it should print a sequence like this:

```
push 1.0

push 2.0

add

push 3.0
```

```
multiply
```

Create a new project in Eclipse(Java IDE). Place the files from the attached .zip in your project's src folder. Modify only CalcParser.java file.

You should ***not*** **modify CalcLexer** -- this is the routine that reads in and tokenizes the expression.

SUBMIT

ScreenShots: Submit screenshots of outputs from your console showing proper translation of examples including those given in the problem. Use System.out.println, not AWT graphics as in the author's demo.

Program file: your modified CalcParser.java file.

Extra Credit Opportunities

Option 1: Build a tree. Up to +10% points if your modified CalcParser first builds a parse tree and then generates the stack syntax by doing a depth-first traversal of that tree. You will need additional classes to do this; a Tree class and a Node class.

Option 2: Exceptional error handling. Up to +10% points if you use exceptions to handle error conditions. These should be looking for errors in the user input and generate exceptions by printing an error message to the console.