Programming Assignment 4

CSCD300 Data Structure

Instructor: Dr. Bojian Xu
Eastern Washington University, Cheney, Washington

Due: 11:59pm, July 13, 2014 (Sunday)

Please follow these rules strictly:

1. Verbal discussions with classmates are encouraged, but each student must independently write his/her own work, without referring to anybody else's solution.
2. No one should give his/her code to anyone else.
3. The deadline is sharp. Late submissions will **NOT** be accepted (it is set on the Blackboard system). Send in whatever you have by the deadline.
4. Every source code file must have the author's name on the top.
5. All source code should be commented reasonably well.
6. Sharing any content of this assignment and its keys in any way with anyone who is not in this class of this quarter is NOT permitted.

Computing the value of a postfix expression using a stack [1].

Definition 1. *The postfix* [2] *of a single operand is itself. The postfix of a two-operand arithmetic expression* $E = \alpha\ op\ \beta$ *is:*

$$\textit{postfix(E)} = \textit{postfix}(\alpha)\ \textit{postfix}(\beta)\ op$$

where each operand of α *and* β *can be recursively another arithmetic expression.*

Example 1. *The postfix expression of* $E = 5 + ((1+2) \times 4) - 3$ *is*

$$\begin{aligned} \textit{postfix}(E) &= \textit{postfix}(5 + ((1+2)\times 4)) \quad 3 \quad - \\ &= 5 \quad \textit{postfix}((1+2)\times 4) \quad + \quad 3 \quad - \\ &= 5 \quad \textit{postfix}(1+2) \quad 4 \quad \times \quad + \quad 3 \quad - \\ &= 5 \quad 1 \quad 2 \quad + \quad 4 \quad \times \quad + \quad 3 \quad - \end{aligned}$$

It can be verified that any arithmetic expression can be unambiguously rewritten using the postfix expression without using pairs of parentheses.

[1] Here for simplicity, we only consider arithmetic expressions that only have two-operand operators.

[2] http://en.wikipedia.org/wiki/Reverse_Polish_notation

An algorithmic idea for calculating the value of a postfix expression. Given a postfix expression, the following idea can be used to calculate the value of the expression. We simply scan over the expression element by element, where an element can be an operator or an operand. (1) If the current element is an operand, we save it in our record and move on unless it's the end of the expression; Otherwise, (2.1) we will recall the two most recent operands that we have in our record and combine them using the operator being scanned. (2.2) We will discard those two operands from our record and save the result from the combining into our record. (2.2) We move on unless it's the end of the expression.

Specification of your program

1. Your program should be named as: `Test_Postfix.java`

2. The input and output of your program.

   The input is a text file that contains a postfix expression, where each element occupies one line of the file. The file name should be supplied to your program as a command line parameter. **You can assume that we will only use two-operand operators:** $+, -, \times, /$ (the four Java opeators: +-*/).

   The output will be the value of the given postfix expression.

An example

Suppose we supply to your program with the file named `data.txt` that has the following content:

```
5
1
2
+
4
*
+
3
-
```

Then, the command line you should type would be:

```
$java Test_Postfix data.txt
```

Then your program will calculate the value of the above postfix using the above algorithmic idea with the aid from the stack data structure. The output should be:

```
$14
```

Implementation of your program

You need to figure out how to use the stack data structure to perfectly implement the process shown in the above algorithmic idea. **You must use a singly linked list based stack** data structure to implement the algorithmic idea for calculating the value of the given postfix expression.

Submission

- All your work files must be saved in one folder, named: **firstname_lastname_EWUID_cscd300_prog4**
  (1) We use the underline ‘_’ not the dash ‘-’.
  (2) All letters are in the lower case including your name’s initial letters.
  (3) If you have middle name(s), you don’t have to put them into the submission’s filename.
  (4) If your name contains the dash symbol ‘-’, you can keep them.

- You need to include a pure ascii text file in the above folder, which contains the description of your implementation of the singly linked list based stack data structure and how to use it for calculating the value of a postfix expression.

- You then compress the above whole folder into a .zip file.

- Submit .zip file onto the Canvas system by the deadline.