# A06

**Due Date:** Tuesday, June 6   
**File to be submitted:** Calculator.java   
**Sample output:** [SampleOutput.html](http://cs.smu.ca/~myoung/csci2341/Assignments/A06/SampleOutput.html)

[SUBMIT](http://cs.smu.ca/~myoung/csci2341/protected/getSubmission.cgi?06)   /   [Check](http://cs.smu.ca/~myoung/csci2341/protected/viewasgn.cgi?A06)

## Postfix Calculator (Stacks)

#### Summary

Implement a postfix calculator using a **Deque** for a stack.

### Details

Your program starts by identifying itself and you, and explaining to the user how to use the program. It then repeatedly prompts the user for a postfix expression, and evaluates each of them until the user enters quit. The postfix expressions include just numbers and operators. Allow double values for the numbers.

A postfix expression is one where the operators follow the numbers they apply to. For example:

15 6 +

is the postfix version of 15 + 6, and

15 6 + 7 \*

is the postfix version of (15 + 6) \* 7. (Note that the postfix version doesn't use any parentheses -- it doesn't need to!)

Sometimes multiple numbers will appear before any operator. For example:

15 6 7 + \*

is the postfix version of 15 \* (6 + 7).

The method for evaluating a postfix expression is always the same:

* Push any numbers onto a stack.
* For any operator, pull the last two numbers off the stack, apply the operator, and then push the result back onto the stack.

So for the expression

15 6 7 + \*

the numbers 15, 6 and 7 get pushed onto the stack (in that order), then the + pops the top two numbers (7 and 6), adds them (to get 13), then pushes 13 onto the stack (so now the stack contains 15 and 13). Then the \* pops the 13 and 15 off the stack, multiplies them (to get 195), which it pushes back onto the stack.

When we get to the end of an expression, the value of the expression is the number left on the stack. (There should be exactly one number left on the stack.)

In addition to the mathematical operators +, -, \*, / and % the calculator also accepts the operators ^ (exponentiation) and // (integer division). (See the sample output for examples of how these work.)

The user enters the postfix expression at the prompt, all on one line, with spaces between the numbers and operators. The program reports each operation as it's carried out.

>>>: **15 6 7 + \***

6.0 + 7.0 = 13.0

15.0 \* 13.0 = 195.0

Result: 195.0

Your program reports any problems that arise. For example, an unknown operator, too many numbers in the expression or not enuf numbers in the expression.

Sometimes the user may forget to put spaces in their input, and the program will get confused. So long as it reports an error that's fine. For example:

>>>: **15 6 7+\***

Error: Unknown operator: 7+\*

### Notes

Use **java.util.ArrayDeque** for the stack. Do not use java.util.Stack, or MyStackInterface, or any other kind of stack. Remember to use only the Stack operations -- push, pop and peek -- and utility operations -- such as size, clear and isEmpty. You will be penalized for using other **Deque** operations.

The integer division operator does normal (double) division then truncates the result to an integer value. Thus 3.5 1.2 // evaluates to 2.0, not 3.0 (as it would if it changed the operands into int values first). Notice that the % operator works similary (3.5 1.2 % evaluates to 1.1), but that you don't need to do anything special for that -- that's the way %works in Java, too.

The main loop of your program will be a sentinel-controlled loop -- looking for the word quit. It should accept any version of that word (QUIT, Quit, quIT, ...).

The body of the loop processes one line (i.e. one expression). If something goes wrong while the expression is being processed, processing ends without a result being printed.

To process one line just read in that line, then create a **Scanner** on it. Keep processing that **Scanner** until it runs out of data (or until an exception occurs). If everything went fine, print the result.

### Grading Outline

* 50% -- Class performs as required
* 30% -- Class shows good design
* 20% -- Submitted material meets the standard requirements as described in the [rules for submissions](http://cs.smu.ca/~myoung/csci2341/Rules.html) and the [style rules](http://cs.smu.ca/~myoung/csci2341/Style.html).

[SUBMIT](http://cs.smu.ca/~myoung/csci2341/protected/getSubmission.cgi?06)   /   [Check](http://cs.smu.ca/~myoung/csci2341/protected/viewasgn.cgi?A06)