CS 240 Homework 3

Alex Vondrak

DUE: January 27, 2012

An RPN calculator uses a stack upon which operands are pushed before operators like `+` or `-` are invoked. Not only does this make it easier to input expressions (at least for some contingent of RPN enthusiasts; see `http://www.hpmuseum.org/rpn.htm`), but it also makes it easier to implement the calculator: you don't have to worry about parentheses or operator precedence.

In this homework, you'll write an RPN calculator that performs the basic integer arithmetic operations (`+`, `-`, `*`, and `/`) using the `ArrayStack` implementation from class to store **`int`**s.

Input Format

Input consists of any number of whitespace-separated tokens. *Hint:* the `next` method of the `Scanner` class will find and return the next complete token.

Output Format

Starting with an empty stack, you must evaluate each token and output the resulting contents of the stack from bottom to top (see the sample output). To evaluate a token:

- If the token represents an integer, push it onto the stack. *Hint:* use methods of the `Integer` class (`http://docs.oracle.com/javase/6/docs/api/java/lang/Integer.html`).
- If the token represents one of the basic arithmetic operators:
  - If the stack has enough elements, pop them off and push the operator's result back onto the stack. In the event of an attempted division by zero, print `Division by zero` before the stack.
  - Otherwise, print `Not enough operands` before the stack.
- Otherwise, print `Unknown operator` before the stack.

Input Sample

```
1
2
+
3 * 4 -
5 /
8 6 7 five 3 0 / 9
+ + + + +
```

Output Sample

```
1
1 2
3
3 3
9
9 4
5
5 5
1
1 8
1 8 6
1 8 6 7
Unknown operator 1 8 6 7
1 8 6 7 3
1 8 6 7 3 0
Division by zero 1 8 6 7
1 8 6 7 9
1 8 6 16
1 8 22
1 30
31
Not enough operands 31
```