Project 4, Stacks and ADT's (100 points)

Due Thursday October 26, in class

Program A – Java's Stack Class (20 points). Write a program to read in a series of words terminated by the word DONE (called a sentinel) and to print the words in reverse order of entry.

The program must use the class Stack in the Java library

```
    import java.util.*;
```

to store the words given as input. Again, follow the instructions in "Printed Submission of Projects.doc". You need only use 4 or 5 words in the sample run.

Program B – Expression Evaluation (80 points). A simple infix expression is one with no parentheses, e.g.

```
    5 + 6 * 3
```

Using 2 stacks you can evaluate a simple infix expression in one pass without converting to a postfix form first. This requires use of two stacks, one with integers and one with operators. One way to do this is to use the approach in Program A, and instantiate two separate stacks for use in the main program.

The main program is again separate. The main program must implement the algorithm given below:

```
  1. Create an empty operator stack
  2. Create an empty value stack
  3. Repeat
1.        Get the next token.
0         If the token is:
     A number:
              Push the number onto the value stack.

     An operator (call it thisOp):
       1.     While the operator stack is not empty, and the top item on the operator stack
       2.     has the same or greater precedence as thisOp,
                 Pop the operator from the operator stack.
                 Pop the value stack twice, getting two operands.
                 Apply the operator to the operands, *in the correct order*.
                 Push the result onto the value stack.
              Push thisOp onto the operator stack.
     until at the end of the string
  4. While operator stack is not empty
            Pop the operator from the operator stack.
2.          Pop the value stack twice, getting two operands.
3.          Apply the operator to the operands, *in the correct order*.
            Push the result onto the value stack.
  5. Pop result from value stack (at this point the operator stack should be empty).
```

You need only use integers for the operands, and +, -, and * for operators. You may use any material in the book. You must create your own stack class, either using arrays or your own linked list. The program must give an error message for an incorrectly formed expression. A GUI is not required; a simple console interface will suffice.

In the printed submission of your project you must at least include the 3 sample runs described below:

The following correct expression, 5+5 + 6*3 + 18

The following incorrect expression with an extra operator, 6+5 + * 6*3 + 18

The following incorrect expression with an extra operand, 6+5 * 6 3 + 18