Assignment 2

CSE 214

Note: Submission guidelines on page 2.

1. Generic Stack Implementation

a. Your stack class must contain a `private static` nested class called `Node`.
b. Your stack must have all instance variables as `private`.
c. It must have the stack operation methods `isEmpty`, `peek`, `pop`, `push` and `size`. Remember to carefully deal with exceptions in these methods (wherever needed).

2. Infix to Postfix conversion

Write a class called `InFixToPostfixConverter`. This class, as its name suggests, will be responsible for converting infix expressions to postfix expressions using your stack implementation.

a. Your class must contain a `public String convert(char[] infix)` method that takes as input an infix expression, and outputs the equivalent postfix expression.

3. Evaluating a Postfix expression

Write a class called `PostfixEvaluator` to evaluate postfix expressions.

a. It must contain a `public int evaluate(char[] postfix)` method.
b. It must contain a `public static void main(String[] args)` method to run your code. This main method should ask the user to input one infix expression per line until the user types "`q`" or "`Q`". After every input, it should print out the equivalent postfix expression, and the final value of the expression, and then ask for the next input.

Important Notes

- **You may not use any Java data structures directly (other than arrays). Using Java's native data structures in this homework will result in a zero score.**
  Note that you will not need to use linked list for this stack implementation.
- There are points in this homework for properly using Java keywords. For instance, don't fill up your code with new instances of the same variable. If you are repeatedly using a particular value, use the suitable Java keywords.
  (**Hint:** Think about the example `public static final PI =` 3.14... we discussed in class.)
- It may be useful to look into the official Java documentation[1] of `StringBuilder` and `Character`.
- You may work under the assumption that your code will be tested only for expressions with single-digit numbers (e.g., "$4 - (5 + 2/1)$", not something like "$23 - 11$").

Points Distribution	**Total: 50 points**
Stack implementation	10 points
Infix to postfix conversion	20 points
Evaluating a postfix expression	10 points
Proper usage of Java keywords	5 points
Main method in `PostfixEvaluator`	5 points

[1] `http://docs.oracle.com/javase/7/docs/api/index.html?overview-summary.html`

Submission Guidelines

1. Submit a single `.zip` file consisting of three `.java` files:
   i. `Stack.java`
   ii. `InFixToPostfixConverter.java`
   iii. `PostfixEvaluator.java`
2. Remember to include the proper documentation in your comments in all three files!
3. The `.zip` file that you submit should be named as `<SBUID>_<NETID>.cse214.hw2.zip`. For example, Mr. John Doe with student ID number 123456789 will submit the file: `123456789_jdoe.cse214.hw2.zip`.
4. Please make sure that your code compiles and can be run from the command line. **Code that does not compile will not be graded**.
5. The `public static void main(String[] args)` method is the *entry point* for a user. If you do not implement it, we cannot run and use your code. **If this method is missing, your code will not be graded**.
6. Assignments will not be regraded if you submit the wrong files. So, **double/triple-check what you are submitting!**

Submission Deadline

The due date for this assignment is midnight of **Friday, Oct 16, 2015**.