CSCI 2540 Assignment 7

100 points

Due date: Tuesday, March 28 by 11:59pm

In this assignment, you will implement the application discussed in class (and on the textbook) on evaluating infix expression by converting infix expression to postfix form and then evaluating the postfix expression. For example, if the input is an infix expression such as (23+10)*2 – 5, it will first be converted to the postfix expression which is 23 10 + 2 * 5 -, and then calculate the result from the postfix expression, which is 61.

You need to write two classes. One class is the Calculator class and the other class is the CalculatorDemo class.

The Calculator class provides methods for converting infix expression to postfix expression and evaluating the postfix expression.

The CalculatorDemo class includes the main method. It should ask user to enter an infix expression and the program will convert to postfix form and then calculate the result. **Print the postfix expression after conversion and then print the result after evaluation.** Your program should allow user to continue to evaluate another infix expression if they choose to.

The infix expression consists of integer operands, binary operators including +, -, *, /, and may include parentheses. An operand may have one or more digits, for example, 5 or 215. Assume unary operators are illegal. **The input may include spacing between operands, operators, and parentheses, but are not always required to including spacing.**

Use integer division instead of floating-point division during calculation. For example, 6 / 5 should return 1. If you want, your operands can also be floating-point numbers (with decimal point), such as 2.54 *3 + 5. In that case, you will use floating-point division instead of integer division. But you are not required to do so, i.e., you can limit the operands to multi-digit integers.

You need to use wrapper class (Integer/Double and Character) for stack element type. Primitive types are not allowed to be used for generic data type.

The Calculator class should include the following methods:

```
Class Calculator {

  public Calculator(String exp) {}     // Initializes infix expression
  public String toString() {}          // Returns infix expression
  private boolean convertPostfix() {} // Creates postfix expression.
                                       // Returns true if successful.

  // The following methods should throw IllegalStateException if they
  // are called before convertPostfix.
  // Returns the resulting postfix expression
  public String getPostfix() throws IllegalStateException {}

  // Evaluates the expression
  public int evaluate() throws IllegalStateException {}
}
```

Note that if the methods `evaluate` and `getPostfix` are called before the `covertPostfix` method, then the exception `IllegalStateException` should be thrown by these methods. In other words, if the expression is not successfully converted, a call to `evaluate` or `getPostfix` should throw `IllegalStateException`. `IllegalStateException` is already defined in Java.

Your program should also be able to handle the situation when there are unmatched number of paratheses. For example, if the input is (2+3*4 or 2+3)*4, the `covertPostfix` method will return false and the `getPostfix` method will throw the exception. The exception will be handled in the main method.

Psuedocode from the textbook on converting from infix to postfix expression is posted on Canvas (under Textbook pseudocode for Stack Applications). The Psuedocode on the textbook assumes that the input is correct. Therefore, you will need to modify the code to handle invalid input with unbalanced paratheses.

If your program can only handle single digit numbers, you will get partial credit. Please print a message to indicate it if your program only supports single digit numbers instead of multi-digit numbers as expected.

Javadoc comments are required for this assignment and all future assignments (for all public methods or constants, as well as comments at the beginning of each file).

<u>Submission instructions:</u>

To submit your programs, you need to submit your programs electronically on Canvas.

Please **use a named package for each of your assignment.** For example, for assignment 7, create a new package and **name your package as assg7_yourPirateId** (use lower case for your pirate id), such as assg7_smithj19. You also need to include a statement such as "package assg7_smithj19;" at the beginning of each of your .java file. **Please follow this naming convention exactly for all future assignments. You will be deducted points for not doing so.**

When you submit your files to Canvas, please submit a zip file with your package folder inside the zip file. The package folder should include only .java files (make sure you include .java files, not .class files). The name of the folder should match with your package name.