COMP2396B - Assignment 1

Due: 12 Feb, 2019 23:55

Introduction

This assignment tests your basic programming skill in Java and refreshes the programming skills that you should have learnt in the first programming course.

You are asked to write an **infix to postfix convertor**. Although program design will not be evaluated in this assignment, you are encouraged to apply the object-oriented programming technique that you have learnt, in particular, the principles of abstraction and encapsulation.

You are also required to write **JavaDoc** for all non-private classes and non-private class members. (In this assignment, you don't need to write JavaDoc for readInfix() method although it is non-private) **Programs without JavaDoc will not be marked.**

Requirements

You will be provided a skeleton file, `InfixReader.java`, which consists of a simple `main()` method that **controls the basic program flow**; and a method that **reads a line from input** which is implemented for you. You will need to implement the `doConversion()` method that **read an infix from input** (using the provided `readInfix()` method), convert that to **postfix** and **print** it out. You also need to implement the `evalPostfix()` method that will evaluate the postfix representation of the input arithmetic expression, and then **print** the **result** of the evaluation of the expression on **the next line**.

To support the **infix to postfix conversion** and the evaluation of postfix expression, you are required to implement the class **Stack**. You are **not** allowed to use any class provided in the packages in `java.util.*` or any other Java classes that provide the implementation of a stack from the Internet. Please refer to the tutorial for the basic operations of a stack.

Your program should read in an integer arithmetic expression in **infix** form, and output the same expression in **postfix** form. Numbers and operators in the expression are separated by **at least one single space**. (**NOTE: ALL NUMBERS IN THE EXPRESSIONS ARE INTEGERS**)

For example:

```
Input infix: 12 + 23
Postfix: 12 23 +
Result: 35
Input infix: 34 * 56
Postfix: 34 56 *
Result: 1904
```

Your program should support the **five arithmetic operators**, ^, +, −, * and /. The ^ is the exponential operator and has the highest precedence. The precedence of operators * and / are higher than that of operators + and −. For example:

```
Input infix: 1 + 2 * 3
Postfix: 1 2 3 * +
Result: 7
Input infix: 8 - 64 / 4 ^ 2
Postfix: 8 64 4 2 ^ / -
Result: 4
```

Your program should also support the use of **parenthesis**. For example:

```
Input infix: ( 1 + 2 ) * 3
Postfix: 1 2 + 3 *
Result: 9
```

*** Note the spacing before and after the parenthesis.**

Your program should support **negative values**. There is no space between the negative sign and the number in negative values. For example:

```
Input infix: 5 + -2 * 2
Postfix: 5 -2 2 * +
Result: 1
```

Marking

- **60% marks** are given to the **functionality** of your program.
  - You may add **additional classes, instant variables and methods** to the class.
  - You may assume that the input is always a **valid**, **space** delimited **infix** expression.
  - Your program output must be **identical** to what is described in this document, with the exception of the trailing spaces at the end of each line of output.
- **40% marks** are given to your **JavaDoc**. A complete JavaDoc includes documentation of every classes, member fields and methods that are not private. In this assignment, you don't need to write JavaDoc for readInfix() method although it is non-private. JavaDoc for the main method may be omitted.

Submission:

Please submit the following files to Moodle. **Late submission is not allowed**.

- *InfixReader.java*