COMP2396B - Assignment 1

Due: 17 Feb, 2020 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 a **postfix to infix 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.

You are also required to write **JavaDoc** for all non-private classes and non-private class members. **Programs without JavaDoc will not be marked.**

Requirements

You will be provided a skeleton file, `PostfixReader.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 Postfix from input** (using the provided `readPostfix()` method), convert that to **infix** and **print** it out. In addition, **print** the **result** of the input equation to **the next line**.

To support the **postfix to infix conversion**, 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 Assignment1_Background Slides for the basic operations of a stack.

Your program should read in an arithmetic expression in **postfix** form, and output the same expression in **infix** form. Numbers and operators including parenthesis in the expression are separated by **at least one single space**. For example:

```
Input postfix: 12 23 +
Infix: ( 12 + 23 )
Result: 35
Input postfix: 34 56 *
Infix: ( 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 postfix: 1 2 + 3 *
Infix: ( ( 1 + 2 ) * 3 )
Result: 9
Input postfix: 8 64 4 2 ^ / -
Infix: ( 8 - ( 64 / ( 4 ^ 2 ) ) )
Result: 4
```

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

```
Input postfix: 5 -2 2 * +
Infix: ( 5 + ( -2 * 2 ) )
Result: 1
```

Your program should also detect the validity of postfix expressions. For example:

```
Input postfix: 2 + 3 *
Error: Invalid postfix
Input postfix: 1 3 5 * + 7 / -
Error: Invalid postfix
```

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 **space** delimited 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. JavaDoc for the main method may be omitted.

Submission:

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

- *PostfixReader.java*