Assignment 3

Objectives

- Review command line input and string to integer conversion
- Practice using Exceptions
- Exposure to postfix notation
- Practice using an ADT to solve a problem

Introduction

In this assignment you will implement a program that evaluates expressions written using postfix notation. For example, the result of evaluating `3 7 9 + +` is 19.

The program will accept a postfix expression on the command line and print the result of evaluating the expression using the algorithm below:

```
try
      while there is more input
            if  next token is an operand
                  push value on the stack
            if next token is an operator
                  pop two values from the stack
                  apply the operator to the two values just popped
                  push the result of applying the operator on the stack

      if one element left on stack
            pop value and display it
      else
            invalid expression
catch EmptyStack or NumberFormatException
      invalid expression
```

This assignment has two parts:

1. implement the Stack ADT using an internal linked data structure. (You may not use any of the Java classes that implement the java.util.List interface.)
2. implement a program that uses a Stack to evaluate postfix expressions

An important concept:

A completed array-based Stack is provided. Note that:

1. It also implements the Stack interface, making all the *array-based* implementation hidden from a user, who can only see public method headers. Therefore, a *user* need not care which version of the stack is used. The user, in this case, is the calculator class.
2. In the calculator, the Stack object can be instantiated as either an ArrayStack or as a LLStack. There is only one line in the source code that would be different. For the ArrayStack, it is
   ```
   Stack stack = new ArrayStack();
   ```
   For the LLStack, it is
   ```
   Stack stack = new LLStack();
   ```

Part I – Implement the stack interface using a linked list

Create a class called `LLStack` in a file named `LLStack.java`. The class `LLStack` must implement the `Stack` interface specified in `Stack.java` using a student-created linked list structure to contain its elements.

Create an appropriate `Node` class for your linked list implementation in a file named `Node.java`

Modify `StackTester.java` so that it tests your implementation of the `LLStack` class.

Part II – Implement the calculator

Implement a program in a file called `Calc.java` that accepts postfix expressions on the command line and outputs the result of evaluating the expression to the console. Your program must also handle invalid expressions gracefully.

You should break down your Calc program into at least two other methods besides main. Solutions that have all the code in the main method will lose marks for poor style.

Your calculator only needs to support integer operands for convenience, but note that integer division with integers will yield a different answer

Your calculator should support the following binary operators:

+	addition
-	subtraction
/	division
*	multiplication

Note that we are using the lower case letter x to represent multiplication, not the *. The reason for this is that the '*' character on the command line will be misinterpreted as a wildcard by the operating system.

The table below shows some of the test cases we will use and the exact output your program must produce for those inputs.

Command line	Output
`java Calc 5 4 +`	`9`
`java Calc 5 4 -`	`1`
`java Calc 5 4 x`	`20`
`java Calc 10 2 /`	`5`
`java Calc 1 2 3 4 5 + + + +`	`15`
`java Calc 4 +`	`Invalid expression.`
`java Calc 4 5 + 6 3 / x`	`18`
`java Calc 1 2 + + +`	`Invalid expression.`
`java Calc what is this`	`Invalid expression.`

Submission

Submit your `LLStack.java, Node.java and Calc.java` using Connex.

Please be sure you submit your assignment; don't just save a draft.

A reminder that it is OK to talk about your assignment with your classmates, and you are encouraged to design solutions together, but each student must implement their own solution.

We will be using plagiarism detection software on your assignment submissions.

Grading

You will be graded on the following:

- Good style as per the coding_conventions.pdf file on conneX and use of methods to do the work.
- Following the exact specifications in this assignment.