WIA1002 DATA STRUCTURE
WEEK 5 LAB

1. Implement an Integer stack class named MyStack. It should contain the following methods:

Name	Specification
MyStack	Default constructor
isEmpty	Return whether the generic stack is empty
peek	Return the value of the top element in the generic stack without removing it
push	Add an element to the top of the generic stack
pushMany	Add several elements to the generic stack using comma as the split delimiter
pop	Remove and return an element from the top of the generic stack
popAll	Remove and return all elements from the generic stack
toString	Return a string that displays the content of the stack.

2. Postfix Evaluation

Implement a class named PostfixEvaluation that evaluates postfix expressions using the MyStack class above. Postfix notation is a way of writing expressions without using parentheses. For example, the expression (1 + 2) * 3 would be written as 1 2 + 3 *. A postfix expression is evaluated using a stack. Scan a postfix expression from left to right. A variable or constant is pushed into the stack. When an operator is encountered, apply the operator with the top two operands in the stack and replace the two operands with the result.

Write a method to evaluate postfix expressions. Pass the expression as a string. The method only has to support the operation of + - * / for integers.

public static int evaluatePostfix(String expression)

The following diagram shows how to evaluate 12+3*.

```
public static void main(String[] args) {
    System.out.println("12+3+=" + evaluatePostfix("12+3+"));
}
```

```
12+3+=6
BUILD SUCCESSFUL (total time: 1 second)
```

3. Check a HTML file.

Download the sample.txt, sample2.txt, and sample3.txt files. Write a method to scan the text file and determine if the tags are correctly written. Assume that the first line <!DOCTYPE html> is always free of error.

Here is a sample run for sample.txt:

```
Pushing into stack: html
Current Stack: [html]

Pushing into stack: body
Current Stack: [html, body]

Pushing into stack: h1
Current Stack: [html, body, h1]

Found an ending tag: h1
The ending tag is correct
Current stack: [html, body]

Pushing into stack: p
Current Stack: [html, body, p]

Found an ending tag: p
The ending tag is correct
Current stack: [html, body]

Found an ending tag: body
The ending tag is correct
Current stack: [html]

Found an ending tag: html
The ending tag is correct
Current stack: []

Is the html file correct? true
BUILD SUCCESSFUL (total time: 1 second)
```

Here's a sample run for sample2.txt:

```
Pushing into stack: html
Current Stack: [html]

Pushing into stack: h1
Current Stack: [html, h1]

Found an ending tag: h1
The ending tag is correct
Current stack: [html]

Pushing into stack: p
Current Stack: [html, p]

Found an ending tag: p
The ending tag is correct
Current stack: [html]

Found an ending tag: body
Error: html, body

Is the html file correct? false
BUILD SUCCESSFUL (total time: 2 seconds)
```

Here's a sample run for sample3.txt:

```
Pushing into stack: html
Current Stack: [html]

Pushing into stack: body
Current Stack: [html, body]

Pushing into stack: h1
Current Stack: [html, body, h1]

Pushing into stack: h1
Current Stack: [html, body, h1, h1]

Pushing into stack: p
Current Stack: [html, body, h1, h1, p]

Found an ending tag: p
The ending tag is correct
Current stack: [html, body, h1, h1]

Found an ending tag: body
Error: h1, body

Is the html file correct? false
BUILD SUCCESSFUL (total time: 2 seconds)
```

4. Infix to Postfix

Infix expression: The expression of the form a op b. When an operator is in-between every pair of operands. Postfix expression: The expression of the form a b op. When an operator is followed for every pair of operands.

Write a method to turn a fully parenthesized infix expression into postfix.

public static String toPostfix(String str)

Pass the expression as a string. Try it with this fully parenthesized infix expression:

((((A-B)+((M^N)*(O+P)))-((Q/(R^S))*T))+Z)

Here's a sample run:

```
public static void main(String[] args) {
    String str = "((((A-B)+((M^N)*(O+P)))-((Q/(R^S))*T))+Z)";
    System.out.println("Infix: " + str);
    System.out.println("Postfix: " + toPostfix(str));
}
```

```
Infix: ((((A-B)+((M^N)*(O+P)))-((Q/(R^S))*T))+Z)
Postfix: AB-MN^OP+*+QRS^/T*-Z+
BUILD SUCCESSFUL (total time: 1 second)
```