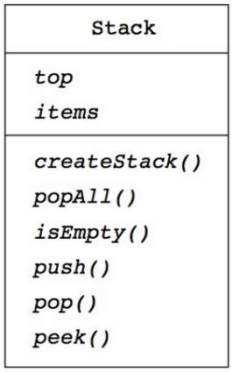
SYSC 2100 Algorithms and Data Structures
Summer 2017
Assignment 3: Stacks
Due: July 28, 2017 at 6 pm

(20 marks)
Stack is an abstract data type with last-in, first-out behavior. Implement a reference-based stack class named `StackReferenceBased` **without using any classes or interfaces from the Java Collections Framework**. This stack implementation should be capable of performing the operations shown in the following UML diagram.



Design a class named *`Infix2PostfixConverter`* to implement an infix calculator using the class *`StackReferenceBased`*. The *`Infix2PostfixConverter`* **must** accept infix expressions from the user, covert the infix expression to postfix expression (method `convertPostfix`), and evaluate the resulting postfix expression (method `getPostfix`). Use only the operators +, -, *, and /. You can assume that the infix expression is syntactically correct and that the unary operators are illegal. However, the infix expression should

a. allow for any type of spacing between operands, operators, and parentheses
b. allow for multi-digit integer operands

The output for some example infix operations should appear as follows:

```
infix: (10 + 3 * 4 / 6)
postfix:  10 3 4 * 6 / +
result: 12

infix: 12*3 - 4 + (18 / 6)
postfix:  12 3 * 4 - 18 6 / +
result: 35

infix:  35 - 42* 17 /2 + 10
postfix:  35 42 17 * 2 / - 10 +
result: -312

infix: 3 * (4 + 5)
postfix:  3 4 5 + *
result: 27

infix: 3 * ( 17 - (5+2))/(2+3)
postfix:  3 17 5 2 + - * 2 3 + /
result: 6
```

For evaluation purposes, it is sufficient to hardcode the infix expressions in the main function.

Optional Question (No additional marks):
Enhance *`Infix2PostfixConverter`* to check whether the infix expression is syntactically correct. If the expression has one of the errors mentioned, print out an appropriate error message, and where possible, indicate where the error occurred in the expression. If the expression is syntactically correct, evaluate the expression.

c. Watch for errors in the infix expression. Here are some examples.
   a. a + 4 (Illegal character a)
   b. 4 + 5 3 (Space between 5 and 3, a missing operator)
   c. 4 + * - 2 (Missing operand)
   d.) 2 + 3 (Improperly nested parenthesis)
   e. (2 + 3) * 5) (Mismatched parentheses-right to left)

Submission Requirements: Submit your assignment (the source files) using ***cuLearn***. Your program should compile and run as is in the default lab environment, and the code should be well documented. Submit all files without using any archive or compression as separate files. The main program should be called **TestInfix2PostfixConverter.java**, if you need to define additional classes etc., you are free to name them according to your own needs. But the TA(s) should be able to run your application by entering **`java TestInfix2PostfixConverter`** on a command-line. Marks will be deducted if the files are submitted in zip or jar format.

Marks for Problem 2 will be based on:

	25%	50%	75%	100%
Program (14 marks)	Compiles	Runs	Program uses JCF stack and evaluates infix operations correctly.	Program uses user-defined stack and evaluates infix operations correctly.
Following good coding style (2 marks)	Hard to decipher	Can follow flow of control	Easy to read, meaningful variable and function names	Good error handling such as gracefully catching invalid input (no stack traces).
Comments (2 marks)	Occasional	Lots, but extraneous	Few, but meaningful	Sufficient and high-quality comments, both in-line and method declaration
Completeness of your submission (2 marks)	Something in CULearn	Lots of effort to extract and run	Some effort to extract and run	Adhering to the submission requirements

The due date is based on the time of the ***cuLearn*** server and will be strictly enforced. If you are concerned about missing the deadline, here is a tip: multiple submissions are allowed. So you can always submit a (partial) solution early, and resubmit an improved solution later. This way, you will reduce the risk of running late, for whatever reason (slow computers/networks, unsynchronized clocks, failure of the Internet connection at home, etc.).

In ***cuLearn***, you can manage the submission until the deadline, taking it back, deleting/adding files, etc, and resubmitting it. The system also provides online feedback whether you submitted something for an assignment. It may take a while to learn the submission process, so I would encourage you to experiment with it early and contact the TA(s) in case you have problems, as only assignments properly and timely submitted using ***cuLearn*** will be marked and will earn you assignment credits.