Homework 6

Deadline: 2015/05/15 09:00

Shin-Jie Lee (李信杰)
Assistant Professor
Computer and Network Center
Department of Computer Science and Information Engineering
National Cheng Kung University

Problem Description

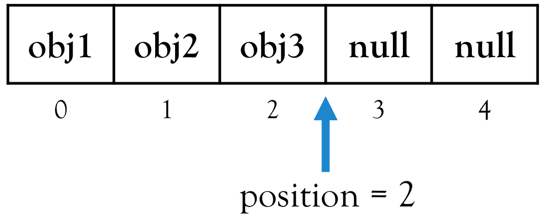
- Infix, postfix notations are two different but equivalent ways of writing expressions.
- For example:
  - Infix notation: X + Y
    - Operators are written in-between their operands.
  - Postfix notation: X Y +
    - Operators are written after their operands.
- Write a program to transform an expression from infix to postfix, and evaluate the postfix expression.
- For example, given an infix expression `2 + 3 * 4`, you should output the expression in the postfix style `2 3 4 * +` and evaluate it to `14`. Each operand and operator is separated by a space.
- Use a stack data structure to solve this problem.

Problem Description

- Your program should be able to detect the following infix expression problems:
  - Divide by zero (ex: 5 / 0 + 3)
  - Overflow (ex: 2147483647 + 1)
  - Invalid expression (ex: 5 + ((2)
- The infix expression only contains the following elements:
  - Integers (May be larger than 2^{31}-1 or less than -2^{31})
  - Operators: +, -, *, /
  - Parentheses: (,)
  - Whitespaces
- The input file is given from the program argument `args[0]`. Each line contains an infix expression to be transformed and evaluated.

Class Design

- Implement a class `Container` with at least the following attributes and operations: (you can add anything you need)
  - `protected Object[] elements`
    - Stores some objects.
  - `protected int position`
    - Holds the index of the last object added to `elements`.
  - `public Container(int)`
    - Initializes the size of `elements`.
  - `public void add(Object)`
    - Adds an object to `elements`.
  - `public Object remove(int)`
    - Removes and returns the object at the specified index in `elements`.
  - `public Object get(int)`
    - Returns the object at the specified index in `elements`.
  - `public boolean isEmpty()`
    - Returns true if `elements` contains no objects.



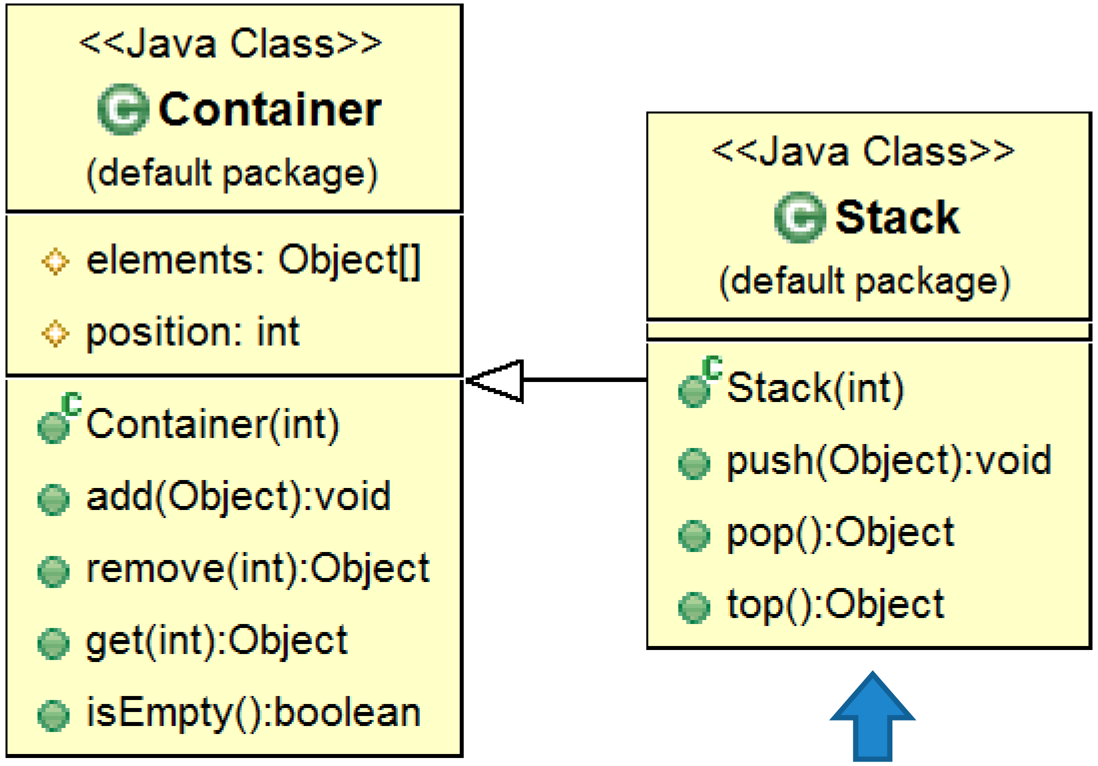
Class Design

- Define a class `Stack` that is derived from `Container` with the following operations:
  - `public Stack(int)`
    - Initializes the size of objects this stack can hold.
  - `public void push(Object)`
    - Pushes an object onto the top of this stack.
  - `public Object pop()`
    - Removes and returns the object at the top of this stack.
  - `public Object top()`
    - Looks at the object at the top of this stack without removing it from the stack.

- You CANNOT add any other attributes or operations.
- You MUST use `super` in each operation of `Stack`.

Class Design

- You CAN add other attributes or operations you need to `Container`.
- You CANNOT add any other attributes or operations to `Stack`.



Use this class as data structure to solve the problem.

Transform from Infix Expr to Postfix Expr

1

a*b+c ➔ ab*c+

Token	Stack			Position	Output
	[0]	[1]	[2]		
a				-1	a
*	*			0	a
b	*			0	ab
+	+			0	ab*
c	+			0	ab*c
				-1	ab*c+

if precedence(instack) $\leq$ precedence(incoming)
pop the stack

operator	**precedence**
+ -	2
* /	1

Transform from Infix Expr to Postfix Expr

a*(b+c)*d ➔ abc+*d*

Token	Stack [0]	Stack [1]	Stack [2]	Position	Output
a				-1	a
*	*			0	a
(	*	(		1	a
b	*	(		1	ab
+	*	(	+	2	ab
c	*	(	+	2	abc
)	*			0	abc+
*	*			0	abc+*
d	*			0	abc+*d
				-1	abc+*d*

Evaluate a Postfix Expression

abc+*d*

Token	Stack			Position
	[0]	[1]	[2]	
a	a			0
b	a	b		1
c	a	b	c	2
+	a	(b+c)		1
*	a*(b+c)			0
d	a*(b+c)	d		1
*	a*(b+c)*d			0

answer = stack.top

Algorithm

```
transform(infix_expr) {
    for each token in infix_expr do
        switch (token)
            case operand:
                print(token)
            case operator:
                while (precedence(stack.top()) <= precedence(token))
                    print(stack.pop())
                stack.push(token)
            case left-parentheses:
                stack.push(token)
            case right-parentheses:
                while (stack.top() != left-parentheses)
                    print(stack.pop())
                stack.pop()
        end switch
    end for
    while (!stack.isEmpty())
        print(stack.pop())
}
```

Algorithm

```
evaluate(postfix_expr) {
    for each token in postfix_expr do
        switch (token)
            case operand:
                stack.push(token)
            case operator:
                pop two elements from stack and do the operation on them
                stack.push(result)
        end switch
    end for
    return stack.top()
}
```

Input and Output

- For each infix expression, you should output two lines:
  - 1st line: the transformed postfix expression
  - 2nd line: the evaluation result of the 1st line
- If the infix expression is invalid:
  - 1st line: "Invalid expression"
  - 2nd line: "Invalid expression"
- If the evaluation process encountered divide-by-zero problem:
  - 1st line: the transformed postfix expression
  - 2nd line: "Divide by zero"
- If the evaluation process encountered overflow problem:
  - 1st line: the transformed postfix expression
  - 2nd line: "Overflow"

Sample Input and Output

Input	1 + 2 6 + 5 * 4 1 - 2 + 3 - 4 + 5 - 6 + 7 - 8 + 9 - 10 999 - 888 * (777 / 666) - 555 + 444 -6 + 0 - (-6) ((1 + 2) + 3) + 4 + 5) 5 / 0 + 1 2147483647 + 1 1234567890 9876543210
Output	1 2 + 3 6 5 4 * + 26 1 2 - 3 + 4 - 5 + 6 - 7 + 8 - 9 + 10 - -5 999 888 777 666 / * - 555 - 444 + 0 -6 0 + -6 - 0 Invalid expression Invalid expression 5 0 / 1 + Divide by zero 2147483647 1 + Overflow 1234567890 1234567890 9876543210 Overflow

Scoring Criteria and Rules

- Correctness: 80%
  - There will be 16 test cases. (Each for 5%)
- Coding standards: 20%
- Plagiarism is strictly forbidden

- You MUST follow the class design depicted in P3~P5.
- You CANNOT use Java collections framework.
  - java.util.ArrayList
  - java.util.Stack
  - java.util.HashMap
  - … etc

Submission

- Please upload your source code to Moodle
  - Put all classes in one java file
- The file name should be `{STUDENT_ID}_hw6.java`
- Deadline: 2015/05/15 09:00
- No late submission is accepted

If you have any problem about this homework,
please email to: p96024029@mail.ncku.edu.tw（林昆輝）