UNIVERSITI KEBANGSAAN MALAYSIA
National University of Malaysia

TTTK1143
Rekabentuk aturcara dan penyelesaian masalah

Tutorial 6
Stack

Name	Mohamad Saiful Nizam Bin Abd Aziz
No. Matric	A179830

TK1143 Program Design

STACK

Section A:

1. Write a java statement to declare and create:
   a) An object variable myStack that represent a data of student's matric number.
   Stack<String> myStack=new Stack<String>();
   b) An object variable yourStack that represent a data of student's weight.
   Stack<Double>yourStack=new Stack<Double>();

2. Complete the following table with the description of each method in Stack.

Method	Description
push( )	Push element into the stack
pop( )	Remove the top element of the stack
peek( )	Retrieve top element without removing it and return the element.
size( )	Get the stack's size
empty( )	This method does not take any parameter. Returns Boolean value.

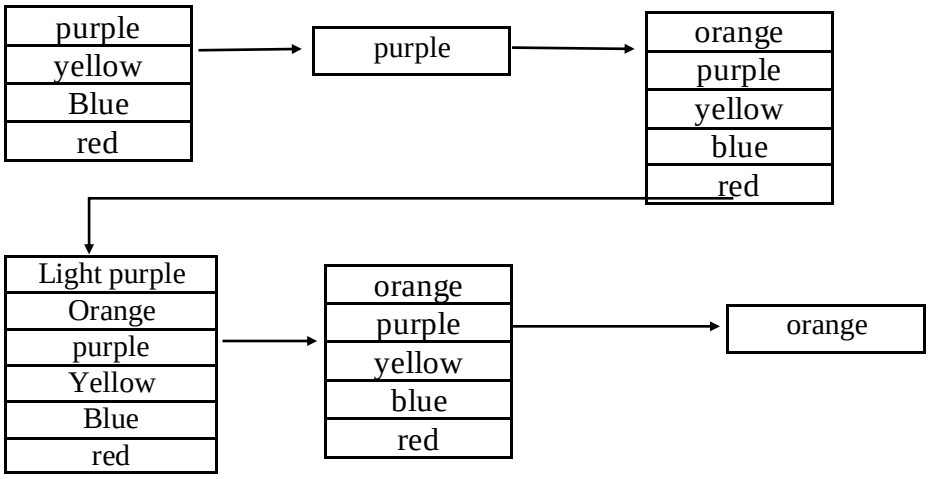
3. Complete the following table that shows a series of stack operations and their effects on an initially empty stack S of integers.

Method	Return Value	Stack Contents
push(9)	-	[9]
push(7)	-	[9, 7]
push(5)	-	[9, 7, 5]
size( )	3	[9, 7, 5]
peek( )	5	[9, 7, 5]
pop( )	5	[9, 7]
isEmpty( )	false	[9, 7]
peek( )	7	[9, 7]
pop( )	7	[9]
pop( )	9	-
isEmpty( )	true	-
push(5)	-	[5]
push(2)	-	[5, 2]
peek( )	2	[5, 2]
push(7)	-	[5, 2, 7]
pop( )	7	[5, 2]
size( )	2	[5, 2]
pop( )	2	[5]
Peek( )	5	[5]

4. Refer to the given program. Draw a diagram to represent the following stack operations, step-by-step as the program executes. Show the output, if any.

```
1  import java.util.Stack;
2
3  public class plateInStack {
4
5      public static void main(String[] args) {
6          Stack<String>plate = new Stack<>();
7          plate.push("Red");
8          plate.push("Blue");
9          plate.push("Yellow");
10         plate.push("Purple");
11         String p1 = plate.peek();
12         plate.push("Orange");
13         plate.push("light " + p1);
14         plate.pop();
15         p1 = plate.peek();
16     }
17 }
```

Answer:



Section B:

1. What is the output of the following program. Draw the content of stack.

```
1  import java.util.Stack;
2
3  public class myStack {
4      public static void main(String[] args) {
5          Stack<Integer>myStack = new Stack<Integer>();
6          int n = 12;
7          while (n > 0) {
8              myStack.push(n%2);
9              n = n/2;
10         }
11         String result = "";
12         while (!myStack.isEmpty())
13             result += myStack.pop();
14         System.out.println(result);
15     }
16 }
```

Answer :
1
11
110
1100

2. Complete the following program, trace and give the output.

```
1  import java.util.*;
2  public class stackProgram{
3      public static void main(String[] args){
4          Stack<String> stack1 = __ new Stack<String>();  __;
5          //insert "A","B","C","D",in the stack
6          - stack1.push("A");
7          – stack1.push("B");
8          – stack1.push("C");
9          – stack1.push("D");
10         // Prints the top of the stack ("D")
11         System.out.println("Top element is: " + _____stack1.peek();_____);
12         // search from the top of the stack
13         System.out.println("C is present at " + stack1.search("C") + " positions from top");
14         stack1.pop( );  // removing the top ("D")
15         _ stack1.pop();       // removing the next top ("C")
16         // Returns the number of elements present in the stack
17         System.out.println("Stack size is " + stack1.size());
18         // check if stack is empty
19         if (__ stack1.empty()  _)
20             System.out.println("Stack is Empty");
21         else
22             System.out.println("Stack is not Empty");
23     }
24 }
```

Answer :

```
Top element is: D
C is present at 2 position from
top
Stack size is 2
Stack is not Empty
```

3. Trace the output of the following program and explain how the program executes.

```
1  import java.util.*;
2  public class intStack{
3      public static void main(String[] args){
4          Stack<Integer>stackOne = new Stack<Integer>( );
5          System.out.println(stackOne.empty( ));
6          for (int i = 0; i < 5; i++)
7              stackOne.push(new Integer(i*10));
8          System.out.println(stackOne.peek());
9
10         Stack <Integer> stackTwo = new Stack<Integer>( );
11         while (!stackOne.empty())
12             stackTwo.push(stackOne.pop());
13         System.out.println(stackTwo.peek());
14     }
15 }
```

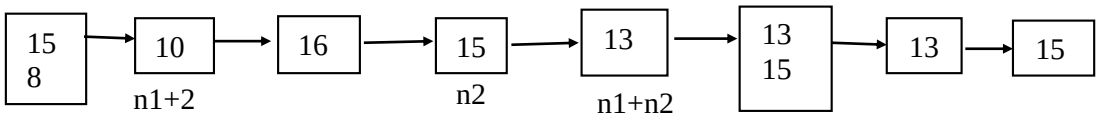
Answer :

```
true
40
0
```

4. Trace the given program and draw the content of stack step-by-step as the program executes.

```
1  import java.util.Stack;
2  public class playStack {
3      public static void main(String[] args) {
4          Stack<Integer>s = new Stack<Integer>();
5          int n1, n2;
6          s.push(15);
7          s.push(8);
8          n1 = s.peek() + 2;
9          s.push(n1+6);
10         n2 = s.peek() ;
11         s.push(n1 + n2);
12         n2 = s.peek() ;
13         s.pop();
14         s.push(13);
15         n1 = s.peek();
16         s.pop();
17         while (!s.empty()) {
18             System.out.println (s.peek());
19             s.pop();
20         }
21     }
22 }
```

Answer :



Output:
16
8
15

5. Trace the output of the following program using the sample input given. Explain the implementation of stack in this program.

Sampel Input
2 i like to eat chocolate # java programming #

```
1  import  java.util.Scanner;
2  import  java.util.Stack;
3
4  public  class  playWithStack  {
5
6      public  static  void  main(String[]  args)  {
7          Scanner  sc  =  new  Scanner(System.in);
8          Stack<String>strStack  =  new  Stack<String>();
9          int  size,x;
10         x=sc.nextInt();
11         for(int  i=0;i<x;i++){
12             size=0;
13             String n = sc.next();
14             while(!n.equals("#")){
15                 strStack.push(n); size++;
16                 n = sc.next();
17             }
18             System.out.print(size  +  ":  ");
19             while(!strStack.empty()){
20                 System.out.print(strStack.pop()  +  "  ");
21             }
22         }
23     }
24 }
```

Answer :
5: chocolate eat to like i
2: programming java

Section C:

There are four (4) questions given in this section. Your task is to write the program to implement data structure Stack for each case.

PALINDROME	
Input	Standard Input
Output	Standard Output
Programming Elements	loop
Data Structure	stack

Problem Description

Palindrome is a sequence of characters, a word, phrase, number or sequence of words which reads the same backward as forward. Example of palindrome words are katak, civic and anna

Input

The first line input contains an integer which determines the number of test cases. Each of following lines represent words.

3
racecar
java
madam

Output

For each test case, the output contains respond either 'It is a Palindrome' or 'Not a palindrome'.

It is a Palindrome
Not a palindrome
It is a Palindrome

Your task is to write a program that will read the word and identify whether it is a palindrome or not.

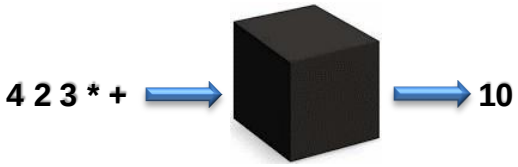
Self Study

1. Explain the implementation of Stack in this case study.

POSTFIX MACHINE	
Input	Standard Input
Output	Standard Output
Java Elements	Loop, Selection
Data Structure	stack

Problem Description

The Postfix Machine is a program that receives a postfix expression and calculate the value of the expression. The postfix form represents a natural way to evaluate expressions because precedence rules are not required. The black box shown below represents the Postfix Machine. If we give the input 4 2 3 * +, the program will give the output 10. How the evaluation is done?



The evaluation is done as follows:

```
4 2 3 * +       [ Compute the last two operands with the first operator that is 2 * 3]
4 6 +           [ Repeat the computation]
10              [ This is the result]
```

Scope of the program:

- The operator is a binary operator: addition, multiplication, division and subtraction.
- The operand is not more than 3 digit integer.
- The expression ends with symbol semicolon.
- The expression only contains operand, operator and semicolon that are separated by spaces.

Your task is to write a program for postfix machine.

Input

The input contains only symbol +, -, *, / and positive number with not more than 3 digits. Each of the symbol and numbers are separated with a single space. The input ends with symbol semicolon. Input example:

```
4 2 3 * + ;
120 3 4 * - 5 + ;
4 2 3 * +
```

Output

The output will be an integer. Based on the above example, the output will be:

```
10
113
ERROR: no END OF STRING in the expression
```

Solution

Postfix Machine Algorithm:

Read the string
Split the string into tokens
Repeat until token is semicolon

When a binary operator is seen:
the two operands are popped from the stack,
the operator is evaluated, and the result is pushed back onto the stack.
When an operand is seen:
it is pushed onto a stack.

Read next token

When the complete postfix expression is evaluated:
the result should be a single item on the stack

Basic Structure of the program:

```
public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        Stack myStack = new Stack();
        String InStr;

        InStr = in.nextLine(); // input string
        StringTokenizer st = new StringTokenizer(InStr);
        while (st.hasMoreTokens()) {
              String nextT = st.nextToken();
              if(!(nextT.equals(";"))) {
                    PART A1
              }
              else
                 break;
        }
        PART B1
}
```

Complete structure of the program/solution: PostfixMachine.

```
1   import java.util.*;
2
3   public class PostfixMachine {
4
5     public static void main(String[] args) {
6         Scanner in = new Scanner(System.in);
7         Stack myStack = new Stack();
8         String lnStr;
9         Integer f1, f2;
10        int result;
11
12        lnStr = in.nextLine(); // input string
13        StringTokenizer st = new StringTokenizer(lnStr);
14        while (st.hasMoreTokens()) {
15            String nextT = st.nextToken();
16            if(!(nextT.equals(";"))) {
17                if (isOperator(nextT)){ //
18                    // >>> ADD YOUR CODE I HERE <<<
19
20                }
21                else {
22                    // >>> ADD YOUR CODE II HERE <<<
23                }
24             }
25             else
26                break;
27         }
28       // >>> ADD YOUR CODE III HERE <<<
29
30    } // end main()
31
32   static boolean isOperator(String tmp) {
33       if ((tmp.equals("+"))||(tmp.equals("-"))||(tmp.equals("*"))||
34           (tmp.equals("/")))
35               return true;
36        return false;
37    }
38
39    static int evaluate(Integer op1, Integer op2, String s1) {
40        int data1 = op2.intValue();
41        int data2 = op1.intValue();
42
43        if (s1.equals("+")) return (data1 + data2);
44               else if (s1.equals("-")) return (data1 - data2);
45               else if (s1.equals("*")) return (data1 * data2);
46               else if (s1.equals("/")) return (data1 / data2);
47               else return -1;
48    }
49
50  } // end class
```

Self Study

1. Can you guess what is the value for the following postfix expression:

Postfix expression	Value
2 3 4 5 - + *	?
4 2 * 3 +	?

2. In your own words, explain Part A1 and Part B1 from the basic structure solution.
3. Among code I, II and III (from the Worked-Example Solution) which code will be used to implement Part A1 and Part B1 (in Basic Structure-Solution)?
4. Try to run the PostfixMachine program and study the output.
5. Compare your answer from question 1 with the program output from question 4.
   If you get similar answer with the program, then:
   a. Congratulations. This mean you understand the problem.
   If the program give different answer from yours, then:
   b. Discuss with your friends and ask your instructor during tutorial session.
6. Study and try run the Postfix Evaluator program from link Stack | Set 4 (Evaluation of Postfix Expression) - GeeksforGeeks.
7. Compare the differences between Postfix Machine program from GeeksforGeeks (Question 6) and your previous program. Which one is more accurate to perform the postfix machine?

INFIX TO POSTFIX CONVERTER

Input	Standard Input
Output	Standard Output
Java Elements	Loop, Selection
Data Structure	stack

Problem Description

Human can understand better when they read mathematical expression in infix form. In this discussion, infix form means the operator is in the middle of two operands. For example, 4 + 2, I am sure all of you understand this expression. However, for a machine, infix form of an expression is hard to process, it prefers postfix expression. For example, 4 2 + which the results is similar to 4 + 2 that is 6.

The issue is human will provide the input, and machine will process it. Therefore, we need a mechanism to convert the input which is in infix expression (given by the human) to postfix expression. In the real calculator, this process is hidden. However, in this topic, we will display the result of the conversion.

The black box shown below represents the Infix to Postfix Converter. If we give the input 4 + 2 * 3, the converter will give the output 4 2 3 * +. How the conversion is done?

Your task is to write a program to perform infix to postfix converter.

Input

The input contains only symbol +, -, *, / and positive number with not more than 3 digits. Each of the symbol and numbers are separated with a single space. The input ends with symbol semicolon. Input example:

4 + 2 * 3 ;
4 + 2 * 3

Output

The output will be a postfix expression as follows:

4 2 3 * +
ERROR: no END OF STRING in the expression

Solution

Infix to Postfix Converter Algorithm:

Read the first string
Repeat until meet semicolon

When meet Operator:
Pop all stack symbols until a symbol of lower precedence or a right-associative symbol of equal precedence appears.
Then push the operator.

When meet Operand:
Immediately output.
Read next string

When meet End of input:
Pop all remaining stack symbols.

Basic Structure of the program:

```
public static void main(String[] args)  {

    Scanner in = new Scanner(System.in);
    Stack Stack1 = new Stack();
    String temp = " ";
    char symT = ' ';

    String infix = in.nextLine();
    StringTokenizer st = new StringTokenizer(infix);
    while (st.hasMoreTokens()) {
        String nextT = st.nextToken();
        if(!(nextT.equals(";"))) {
            PARTA2
        }
    }
        PARTB1
}
```

Complete structure of the program: Infix2Postfix Converter

```
1  import java.util.*;
2
3  public class Infix2Postfix {
4
5    public static void main(String[] args)  {
6       Scanner in = new Scanner(System.in);
7       Stack Stack1 = new Stack();
8       String temp = " ";
9       char symT = ' ';
10
11      String infix = in.nextLine(); // input string
12      StringTokenizer st = new StringTokenizer(infix);
13      while (st.hasMoreTokens()) {
14         String nextT = st.nextToken();
15         if(!(nextT.equals(";"))) {
16             if (isOperator(nextT)){
17                symT = atoc(nextT);
18                // >>>>> add your code here <<<<<
19             }
20             else
21               // >>>>> add your code here if meet operand <<<<<
22          }
23      }
24        // >>>>> add your code here  <<<<<
25      }
26
27     static char atoc(String str) {
28         char data1 = str.charAt(0);
29            return data1;
30     }
31
32
```

```
33
34      static boolean isHigherThan(char op1, char op2){
35
36              // >>>>> add your code here  <<<<<
37       }
38
39      static boolean isOperator(String tmp) {
40
41              // >>>>> add your code here  <<<<<
42      }
43
44  }
```

Self Study

1. Read https://codeburst.io/conversion-of-infix-expression-to-postfix-expression-using-stack-data-structure-3faf9c212ab8.
2. Can you guess what is the postfix expression for the following infix expression:

Infix expression	Postfix expression
4 * 2 + 3	?
2 + 3 – 5 * 1 / 3	?

3. Run the given *Infix2Postfix* file.
4. Now, compare your answer from question 2 with the output from *Infix2Postfix* file . If you get the similar answer with the program, then:
   a. Congratulations. This mean you understand the problem.
   b. If the program file give different answer from yours, then Ask your friends. Then, ask your tutor.

BALANCE SYMBOL CHECKER

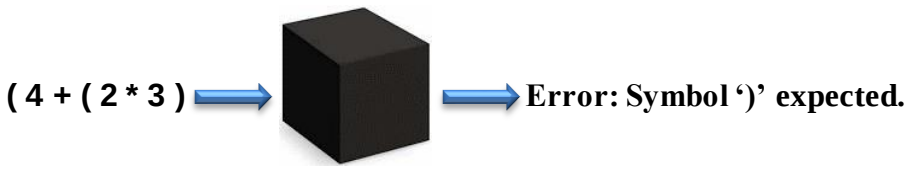
Input	Standard Input
Output	Standard Output
Programming Elements	loop
Data Structure	stack

Problem Description

Compilers check your programs for syntax errors. Frequently, however, a lack of one symbol, such as a missing * / comment-ender, causes the compiler to produce numerous lines of diagnostics without identifying the real error. A useful tool to help debug compiler error messages is a program that checks whether symbols are balanced. In other words, every { must correspond to a }, every [to a], every (to a) and so on.

However, simply counting the numbers of each symbol is insufficient. For example, the sequence [( )] is legal, but the sequence [( ]) is wrong. A stack is useful here because we know that when a closing symbol such as) is seen, it matches the most recently seen unclosed (. Therefore, by placing an opening symbol on a stack, we can easily determine whether a closing symbol makes sense.

The black box shown below represents the Balance Symbol Checker. If we give the input (4 + (2 * 3), the program will display the error message. How this is done?



Challenge yourself to write a program to perform Balance Symbol Checker.

Input

The input contains any symbols including alphabets or numbers. Each of them are separated with a single space. The input ends with symbol semicolon. Input example:

(4 + (2 * 3) ;
(4 + (2 * 3)

Output

The output will be a word either 'Balanced' or 'Unbalanced'. The above input example will give the following output:

Unbalanced
ERROR: no END OF STRING in the expression

Self Study

1. Can you guess what is the output for the following input?

Input	Output
[(5 – 2) * 3]	?
((A * (B + C) / D)	?

2. Explain the implementation of Stack in this case study.