

ใบงานการทดลองที่ 3

เรื่อง อาร์เรย์สตริง และฟังก์ชัน ในภาษาจาวา

1. จุดประสงค์ทั่วไป

- 1.1. รู้และเข้าใจการโปรแกรมเชิงวัตถุร่วมกับอาร์เรย์และสตริง
- 1.2. รู้และเข้าใจการโปรแกรมเชิงวัตถุร่วมกับฟังก์ชัน

2. เครื่องมือและอุปกรณ์

เครื่องคอมพิวเตอร์ 1 เครื่อง ที่ติดตั้งโปรแกรม Eclipse

3. ทฤษฎีการทดลอง

3.1. โครงสร้างข้อมูลแบบ “อาร์เรย์” มีลักษณะเป็นอย่างไร ? มีองค์ประกอบอะไรบ้าง ? อธิบายพร้อมยก ตัวอย่างประกอบ

โดยจะมีโครงสร้าง => ชนิดข้อมูล[] ชื่อตัวแปร={"index0"," index1"," index2"," index3"};

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
```

3.2. การเข้าถึงแต่ละ Element ของอาร์เรย์สามารถทำได้อย่างไร ? อธิบายพร้อมยกตัวอย่างประกอบ

ให้ระบุชื่อตัวแปร[index] = “สิ่งที่เราต้องการแก้ไขข้อมูลใน array”;

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
cars[0] = "Opel";
System.out.println(cars[0]);
// Now outputs Opel instead of Volvo
```

3.3. คำสั่ง length เกี่ยวข้องกับอาร์เรย์อย่างไร ? อธิบายพร้อมยกตัวอย่างประกอบ

เป็นการบอกจำนวนสมาชิกใน array

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
System.out.println(cars.length);
// Outputs 4
```

3.4. จงยกตัวอย่างประกอบในการวนรอบเพื่อแสดงค่าภายในตัวแปรอาร์เรย์ตั้งแต่ค่าแรกจนถึงค่าสุดท้าย

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
for (int i = 0; i < cars.length; i++) {
    System.out.println(cars[i]);
}
```

โค้ดนี้คือการแสดงผลค่า i โดยกำหนดให้แสดงค่าที่ $i < \text{cars.length}$ โดยจะแสดง volvo bmw ford

3.5. จงยกตัวอย่างการใช้งานคำสั่ง for each เพื่อแสดงค่าภายในตัวแปรอาร์เรย์

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
for (String i : cars) {
    System.out.println(i);
}
```

คือการเก็บค่า cars มาไว้ที่ i แสดงข้อมูลที่อยู่ใน array โดยอยู่ในรูปแบบที่สั้นกว่าเดิม

3.6. เหตุใดจึงต้องมีคำสั่ง import java.util.Arrays ; ในส่วนต้นของไฟล์?

ต้องทำการ import ก่อน จึงจะสามารถใช้งานคลาสได้

3.7. คำสั่ง Arrays.copyOf(____ , ____) ; มีหน้าที่ทำอะไร ?

การใช้เมธอดนี้เราสามารถปรับให้ขนาดของอาร์เรย์ใหม่เล็กกว่า หรือ ใหญ่กว่า เดิม ได้โดยข้อมูลที่ถูกลำเลียงมาจะถูกเก็บตามดัชนีเดิม สังเกตว่าเมื่อเราแก้ไขค่าในตัวแปรใหม่ ค่าในตัวแปรเดิมจะไม่เปลี่ยน แสดงว่าอาร์เรย์ที่สำเนามาเป็นคนละอันกับอาร์เรย์เดิม

3.8. จงยกตัวอย่างการประกาศ String และกำหนดค่าคำว่า “Hello World” ในภาษาจาวา

```
package lab3;

public class lab3 {

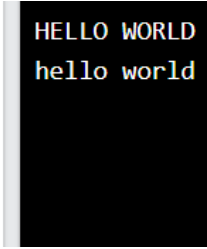
    public static void main(String[] args) {
        System.out.println("Hello World");
    }

}
```

3.9. จงอธิบายและยกตัวอย่างประกอบการใช้งานคำสั่ง toUpperCase() ในภาษาจาวา

คำสั่ง toUpperCase() จะปรับตัวอักษรเป็นตัวพิมพ์ใหญ่ทุกตัว

```
public class Main {
    public static void main(String[] args) {
        String txt = "Hello World";
        System.out.println(txt.toUpperCase());
        System.out.println(txt.toLowerCase());
    }
}
```

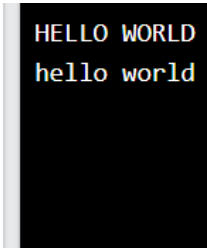


```
HELLO WORLD
hello world
```

3.10. จงอธิบายและยกตัวอย่างประกอบการใช้งานคำสั่ง toLowerCase() ในภาษาจาวา

คำสั่ง toLowerCase() จะปรับตัวอักษรเป็นตัวพิมพ์เล็กทุกตัว

```
public class Main {
    public static void main(String[] args) {
        String txt = "Hello World";
        System.out.println(txt.toUpperCase());
        System.out.println(txt.toLowerCase());
    }
}
```



```
HELLO WORLD
hello world
```

3.11. จงอธิบายและยกตัวอย่างประกอบการใช้งานคำสั่ง indexOf() ในภาษาจาวา

คือ คำสั่ง หรือฟังก์ชันสำหรับค้นหาข้อความ หรือตัวอักษร โดยคำสั่ง indexOf จะคืนค่ากลับมาเป็นลำดับที่ค้นหาข้อความ หรือตัวอักษรดังกล่าวเจอ เป็นชนิดตัวเลข

MyClass.java

```
01. package com.java.myapp;
02.
03. public class MyClass {
04.
05.     public static void main(String[] args) {
06.
07.         String a = new String("Welcome to ThaiCreate.Com Version 2013");
08.         String b = new String("ThaiCreate.Com");
09.
10.         System.out.println(a.indexOf("Version"));
11.         System.out.println(a.indexOf(b));
12.
13.     }
14.
15. }
```

Output



```
26
11
```

3.12. จงอธิบายความแตกต่างระหว่างการเชื่อม String แบบปกติและแบบใช้คำสั่ง concat()

Concat() ใช้ในการเชื่อมต่อข้อความ ใช้งานเหมือนการใช้เครื่องหมาย "+" สะดวกมากกว่าการเพิ่มแบบปกติหลายๆ บรรทัด

3.13. หากต้องการแสดงสัญลักษณ์พิเศษภายในตัวแปร String ควรทำอย่างไร ?

ใช้ **Unicode** ได้ (กรณีที่เป็นพิมพ์ไม่มีปุ่มให้กด)

3.14. จงอธิบายและยกตัวอย่างประกอบการสร้างฟังก์ชันในภาษาจาวา

เมื่อสร้าง **method** ในภาษา **java** สามารถที่จะเรียกใช้งานได้จากส่วนใดๆ ของโปรแกรมก็ได้ขึ้นกับขอบเขตและระดับการเข้าถึงผู้เขียนได้กำหนดขึ้น

```
public class Main {
    static void myMethod() {
        System.out.println("I just got executed!");
    }

    public static void main(String[] args) {
        myMethod();
    }
}

// Outputs "I just got executed!"
```

3.15. อธิบายข้อแตกต่างระหว่าง Pass by value และ Pass by reference

Pass by Value คือ การส่งค่า (value) เป็น **argument** ของฟังก์ชัน ดังนั้นค่าที่ทำในฟังก์ชันจึงไม่ส่งผลต่อตัวแปรนอกฟังก์ชัน **Pass by Reference** คือ การส่งตัวแปร (variable) เป็น **argument** ของฟังก์ชัน ดังนั้นตัวแปรที่มีการดำเนินการใด ๆ ในฟังก์ชันจะส่งผลให้ตัวแปรนอกฟังก์ชันมีการเปลี่ยนแปลงด้วย

3.16. ความแตกต่างระหว่างการประกาศฟังก์ชันแบบ void กับแบบ int, double, float, string คืออะไร ?

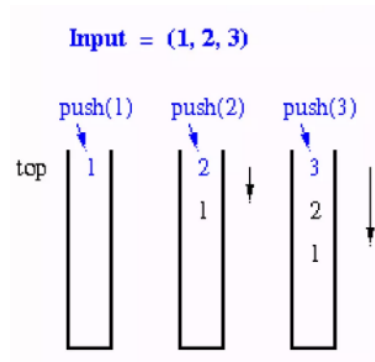
type เป็นประเภทของฟังก์ชันสำหรับการส่งค่ากลับ ประเภทของฟังก์ชันนั้นจะเป็นเหมือนประเภทของตัวแปร เช่น **Integer, Floating, Double** หรือแบบอ็อบเจ็ค ประเภทแบบ **void** หมายความว่าฟังก์ชันไม่มีค่าที่ต้องส่งกลับ

3.17. โครงสร้างข้อมูลแบบ Stack แตกต่างกับ Array อย่างไร ?

Stack จะเป็นเหมือนกล่องใส่ข้อความเมื่อ **push** ค่าเข้าไปก็ **push** เข้าไปที่ละค่าเมื่อ **pop** ออกก็จะเอาค่าที่ **push** เข้าไปล่าสุดออกมา ส่วน **array** จะเป็นการสร้าง **element** ขึ้นมาเก็บหลายๆค่า อ้างอิงตำแหน่งต่างๆเรียกว่า **index**

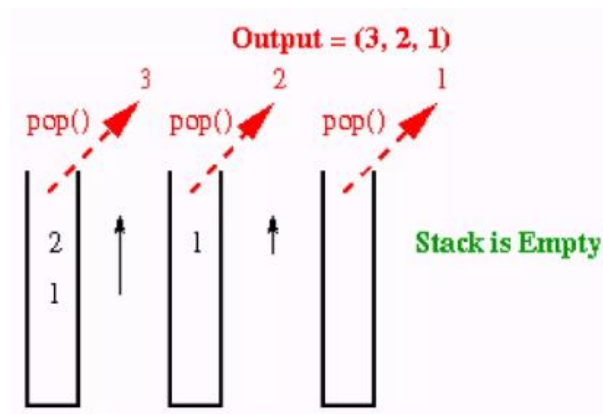
3.18. อธิบายพร้อมยกตัวอย่างประกอบกระบวนการทำงานของคำสั่ง Push ในโครงสร้างข้อมูลแบบ Stack

push() – เป็นการเพิ่ม **element** ลงไปในส่วนบนสุดของ **Stack** หรือ **Pushing (storing)**



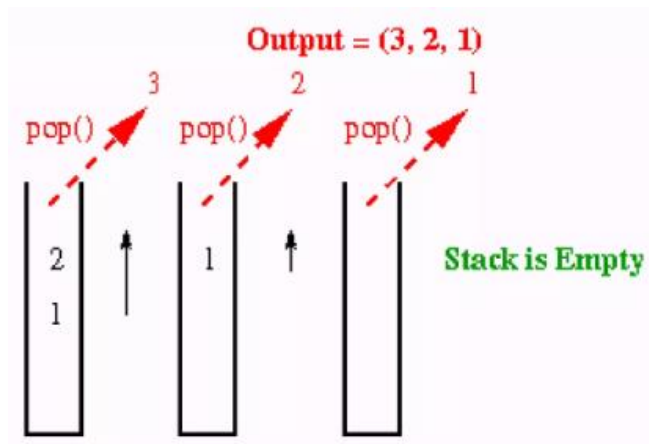
3.19. อธิบายพร้อมยกตัวอย่างประกอบกระบวนการทำงานของคำสั่ง Pop ในโครงสร้างข้อมูลแบบ Stack

pop() – เป็นการลบ **element** ออกจากส่วนบนสุดของ **Stack** หรือ **Removing (accessing)**



3.20. อธิบายพร้อมยกตัวอย่างประกอบกระบวนการทำงานของคำสั่ง isEmpty ในโครงสร้างข้อมูลแบบ Stack

isEmpty() – เป็นการตรวจสอบว่า **Stack** ว่างหรือไม่



3.21. อธิบายพร้อมยกตัวอย่างประกอบความหมายของคำว่า Stack overflow

ปัญหาที่เกิดจากหน่วยความจำระหว่างเรียกเมทอดไม่พอ จนเกิดปัญหา **Stack overflow** หรือปัญหาสแตกล้น (อธิบายสั้น ๆ คือ การจองใช้หน่วยความจำจนเต็ม)ปัญหานี้มาจากความผิดพลาดของผู้เขียนโปรแกรมเอง ที่ไม่มีการออกแบบโปรแกรมและตรวจสอบให้ดี

4. ลำดับขั้นการปฏิบัติการ

4.1. จงแก้โจทย์ปัญหาดังต่อไปนี้

4.1.1. จงเขียนโปรแกรมเพื่อสุ่มค่าเข้าไปในอาร์เรย์ 1 มิติตามจำนวนค่าที่ รับจากผู้ใ้ โดยค่าที่ ถูกสุ่มจะ ต้องเป็นตัวเลขจำนวนเต็มที่อยู่ระหว่าง 0 ถึง 99 เท่านั้น

```
package lab3;
import java.util.Random;
import java.util.Scanner;

public class lab3 {

    public static void main(String[] args) {
        System.out.print("Enter your random Number : ");
        int n;
        Scanner num = new Scanner(System.in);
        n = num.nextInt();
        num.close();

        int[] value = new int[n];
        Random rand = new Random();

        for(int i = 0; i <= n; i++){
            value[i] = rand.nextInt(100);
            System.out.println("Array [" + i + "] =" + value[i] );
        }
    }
}
```

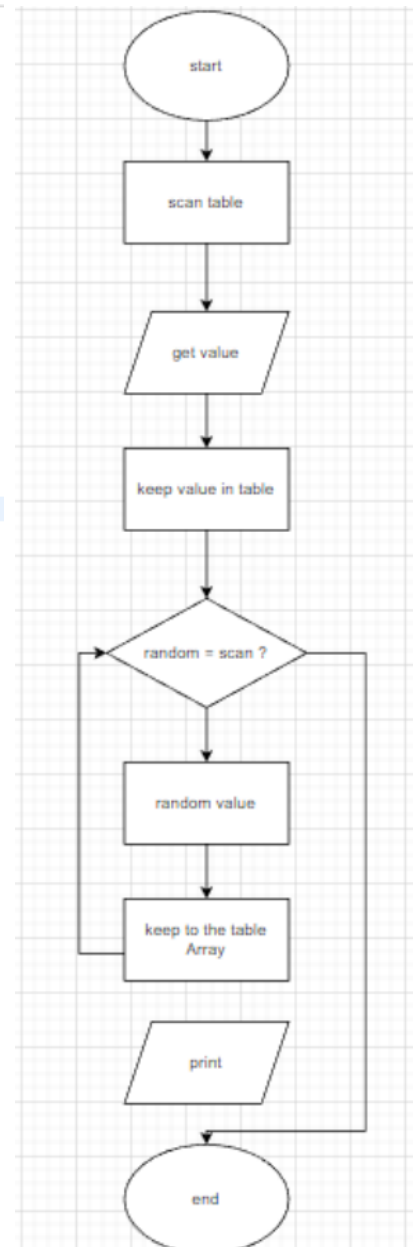
Enter your random Number : 4

Array [0] =27

Array [1] =64

Array [2] =64

Array [3] =71



4.2. จงแก้โจทย์ปัญหาดังต่อไปนี้

4.2.1. จงเขียนฟังก์ชันการจัดการ โครงสร้างข้อมูลแบบ Stack พร้อมจำลองการทำงานโดยการเรียกใช้

คำสั่งพื้นฐานดังต่อไปนี้

คำสั่ง Push(String Value) ; เพื่อนำข้อมูลเข้าไปเก็บไว้ใน Stack

คำสั่ง Pop() ; เพื่อนำข้อมูลบนสุดออกจาก Stack

คำสั่ง isEmpty() ; เพื่อตรวจสอบข้อมูลใน Stack ว่ามีอยู่หรือไม่

คำสั่ง Top() ; เพื่อตรวจสอบข้อมูลที่อยู่ชั้นบนสุด

คำสั่ง CheckStack() ; เพื่อตรวจสอบค่าภายใน Stack ทั้งหมด

คำสั่ง SetStackSize(int size) ; เพื่อกำหนดขนาดเริ่มต้นของ Stack

```
package LAB00P;

import java.util.*;
import java.util.Scanner;

public class LAB3_2 {

    public static void main(String[] args) {

        int n;
        int i = 0;
        String text;

        System.out.print("Input Stack Size : ");
        Scanner sc = new Scanner(System.in);
        n = sc.nextInt();

        Stack<String> stack = new Stack<String>();
        stack.setSize(n);
        stack.clear();

        do{
            System.out.println("----- ");
            System.out.println("Stack Fn ");
            System.out.println("----- ");
            System.out.println("1 : Push ");
            System.out.println("2 : Pop ");
            System.out.println("3 : isEmpty ");
            System.out.println("4 : Top ");
            System.out.println("5 : CheckStack ");
            System.out.println("10 : END");
            System.out.println("----- ");

            System.out.print("Input ");
            i = sc.nextInt();
            System.out.println("");
            System.out.println("----- ");

            switch(i) {

                case 1:
                    System.out.print("Push : ");
                    Scanner sct = new Scanner(System.in);
                    text = sct.nextLine();
                    if(stack.size() == n) {
```

```
if(stack.size() == n) {
                        System.out.println("----| STACK OVERFLOW!!!!!!");
                    }else {
                        stack.push(text);
                    }
                    break;

                case 2:
                    if(stack.size() == 0) {
                        System.out.println("----| STACK IS EMPTY");
                    }else {
                        System.out.println("Pop");
                        stack.pop();
                    }
                    //end if
                    break;

                case 3:
                    if(stack.isEmpty() == true) {
                        System.out.println("----| Yes");
                    }else {
                        System.out.println("----| No");
                    }
                    //end if
                    break;

                case 4:
                    if(stack.size() == 0) {
                        System.out.println("----| NULL");
                    }else {
                        System.out.println("----| Top : "+stack.peek());
                    }
                    //end if
                    break;

                case 5:
                    System.out.println("----| STACK : "+stack);
                    break;

                case 10:
                    i = 10;
                    break;
            }
        }
    }
}
```

5. สรุปผลการปฏิบัติการ

ได้เรียนรู้เกี่ยวกับพวกการใช้ method , array , stack ต่างๆ เพื่อนำไปใช้ในการฝึกทำโจทย์

6. คำถามท้ายการทดลอง

6.1. ข้อควรระวังในการใช้งาน Array ในภาษาจาวาคืออะไร ?

ข้อมูลที่จะเก็บในกล่องนี้ได้ ต้องเป็นข้อมูลชนิดเดียวกัน และที่สำคัญก็ต้องแจ้งไว้ล่วงหน้าด้วยว่าเราจะใช้ที่กล่อง

6.2. ข้อควรระวังในการใช้งาน String ในภาษาจาวาคืออะไร ?

1. ข้อความจะต้องอยู่ภายใน Double Quote ("")

2. หลีกเลี่ยงการต่อ String โดยใช้การ + โดยเฉพาะเมื่อต่อ String จำนวนมาก เพราะ Performance ไม่ดี

3. หากเขียนโปรแกรมที่เป็น Thread safe ให้ใช้ StringBuilder เนื่องจากมี Performance ที่ดีกว่า (การต่อ String โดยทั่วไปจะเข้าข่ายข้อนี้มากที่สุด)

4. หากต้องการต่อ String แบบ Thread safe เนื่องจากโปรแกรมที่เราเขียนไม่มีการป้องกัน Thread safe ก็สามารถใช้ StringBuffer (การต่อ String ทั่วไปไม่ควรดักข้อนี้)

6.3. ฟังก์ชันในภาษาจาวาไม่สามารถใช้งานแบบ Pass by reference ในภาษาซีได้ คุณมีแนวทางการแก้ไขปัญหานี้ได้อย่างไร ?

การเปรียบเทียบค่าระหว่าง String ซึ่งเป็น Object หนึ่งของ Java สามารถทำได้โดยใช้ (==) ถึงแม้ว่า ถ้าใช้กับ Object จะเป็นการเทียบ reference ก็ตาม หรือใช้ .equals()

6.4. โครงสร้างข้อมูลแบบ Stack แตกต่างกับโครงสร้างข้อมูลแบบ Array อย่างไร ?

เป็นโครงสร้างข้อมูลแบบเชิงเส้น ที่มีการใส่ข้อมูลเข้า และนำข้อมูลออกเพียงด้านเดียว ดังนั้น ข้อมูลที่เข้าไปอยู่ใน stack ก่อนจะออกจาก stack หลังข้อมูลที่เข้าไปใน stack ที่หลัง นั่นคือ การ "เข้าทีหลังแต่ออกก่อน" (Last In First Out : LIFO) ต่างจาก array โดยที่ array จะทำหน้าที่เก็บค่าเลขๆ โดยจะเรียงข้อมูลเอาไว้เป็นช่องๆ สามารถดึงข้อมูลออกใช้ได้อย่างอิสระ