

## วัตถุประสงค์

A. เข้าใจการ implement Functional Interface ด้วย lambda expression

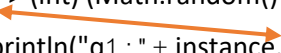
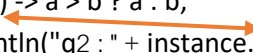
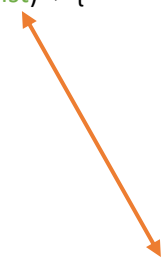
## กิจกรรมที่ 1 ศึกษา lambda expression

เราสามารถใช้ lambda expression ในการสร้าง instance ของ anonymous class ที่ implements functional\_interface ไวยากรณ์ของ lambda expression คือ **(param<sub>1</sub> [, ..., param<sub>n</sub>]) -> { }** โดยมีธรรมเนียมการเขียนดังต่อไปนี้

1. สามารถละ ( ) หากมี param เพียงตัวเดียว (กรณีอื่นต้องมี)
2. สามารถละ { } และ return หากมี expression / statement เดียว (กล่าวได้อีกแบบว่า หากมี block ต้องมี return)

ทั้ง 2 ธรรมเนียมนี้ เพื่อให้ได้ดั่งอ่านได้ง่ายขึ้น

ตัวอย่างการ implement functional interface มีดังนี้

<pre>public interface FuncInterface1 {     public int method1(); // give a number }</pre>	<pre>static void q1() {     FuncInterface1 instance;     instance = () -&gt; (int) (Math.random() * 1000);     System.out.println("q1 : " + instance.method1()); }</pre> 
<pre>public interface FuncInterface2 {     public int method2(int a, int b); // larger }</pre>	<pre>static void q2() {     FuncInterface2 instance;     instance = (a, b) -&gt; a &gt; b ? a : b;     System.out.println("q2 : " + instance.method2(5, 7)); }</pre> 
<pre>public interface FuncInterface3 {     public int method3(List&lt;Integer&gt; input);     // sumEvenElement }</pre>	<pre>static void q3() {     FuncInterface3 instance = (list) -&gt; {         int sum = 0;         for (int i : list)             if (i % 2 == 0)                 sum += i;         return sum;     };     System.out.println("q3 : " +         instance.method3(List.of(2, 5, 7, 4))); }</pre> 

<pre>public interface FuncInterface4&lt;T&gt; {     public T method4(T input); // factorial }</pre> <p>เราสามารถใช้ไวยากรณ์ generic เพื่อให้ T เป็น type ตามการ implement</p>	<pre>static void q4() {     FuncInterface4&lt;String&gt; instance1;     instance1 = (str) -&gt; {         String rev = "";         for (int i = str.length() - 1; i &gt;= 0; i--)             rev += str.charAt(i);         return rev;     };     System.out.println("q4a : " +         instance1.method4("abcd"));      FuncInterface4&lt;Integer&gt; instance2;     instance2 = (factorial) -&gt; {         int result = 1;         for (int i = 2; i &lt;= factorial; i++)             result *= i;         return result;     };     System.out.println("q4b : " + instance2.method4(5)); }</pre>
<pre>public interface FuncInterface5 {     public double method5(double width /*         , double length */); // surfaceArea }</pre> <p>จากที่ lambda expression สามารถสร้าง instance ของ anonymous class implements functional_interface ...เราสามารถส่ง instance นั้นเป็น parameter ได้ (และ สามารถส่งพร้อมกับ param อื่นๆ ได้)</p>	<pre>static void q5() {     double circleArea = callInterface5((radius/* , blank */         -&gt; Math.PI * 2.0 * radius);     DecimalFormat df = new DecimalFormat("#.##");     System.out.println("q5 : " + df.format(circleArea)); }  static private double callInterface5(FuncInterface5 instance) {     return instance.method5(3/* ,0 */); }</pre>

หมายเหตุ เราสามารถเขียนข้างต้นเป็น anonymous class ได้ทุกกรณีข้างต้น เช่น

```
static void q3() {
    FuncInterface3 lambdaWay = (List<Integer> input) -> {
        int sum = 0;
        for (int v : input)
            if (v % 2 == 0)
                sum += v;
        return sum;
    }; //print(lambdaWay.method3(List.of(2,5,7,4)));
    FuncInterface3 instance = new FuncInterface3() {
        public int method3(List<Integer> input) {
            int sum = 0;
            for (int v : input)
                sum = v % 2 == 0 ? sum + v : sum;
            return sum;
        }
    };
    System.out.println(instance.method3(List.of(2,5,7,4)));
}
```

## กิจกรรมที่ 2

กำหนด

```
public interface ArrayProcessor {  
    public int calculate(int[] arr);  
}
```

และ

```
public class Lab5_xxyyyy {  
    static int[] data = { 28, 58, 8, 77, 48, 39 };  
    static ArrayProcessor q1NumberOfEvenElement; // 4  
    static ArrayProcessor q2IndexOfLargestEvenValue; // 1  
    static ArrayProcessor myMedian; // n/2th element of sorted = 48  
  
    public static void main(String[] args) {  
        q1();           // 4  
        q2();           // 1  
        oneline();       // 48  
    }  
    ....  
    static void oneline() {  
        int[] tmp = Arrays.copyOf(data, data.length);  
        Arrays.sort(tmp);  
        /* q3 */ //one statement  
        System.out.println(myMedian.calculate(tmp)); // 48  
    }  
}
```

ส่ง Lab5\_xxyyyy.java ที่ main method เรียก q1() q2() และ q3

หมายเหตุ ในทาง CS เราสามารถกำหนด **median** ให้หมายถึงค่าที่ปรากฏในข้อมูลได้ ...หลายกรณีที่เราต้องการค่าตรงกลาง เพื่อที่มีอยู่จริงไปใช้ประมวลผล

กำหนดส่ง TBA