一. Lambda与函数式接口

1. Lambda 允许把函数作为一个方法的参数（函数作为参数传递进方法中）。

适用于->函数式接口：只有一个方法的接口。

2. 优点：

使用 Lambda 表达式可以使代码变的更加简洁紧凑，并且可以方便更快编程。

3. 语法

(parameters) -> expression

或

(parameters) ->{ statements; }

4. 特征

* **可选类型声明：**不需要声明参数类型，编译器可以统一识别参数值。
* **可选的参数圆括号：**一个参数无需定义圆括号，但多个参数需要定义圆括号。
* **可选的大括号：**如果主体包含了一个语句，就不需要使用大括号。
* **可选的返回关键字：**如果主体只有一个表达式返回值则编译器会自动返回值，大括号需要指定明表达式返回了一个数值。

5. demo

**public class** LambdaDemo {  
 **public static void** main(String[] args){  
 *//不声明类型，不加大括号* MathDemo math1 = (a,b) -> a-b;  
 System.***out***.println(math1.operation(3,2));  
 *//声明类型，不加大括号* MathDemo math2 = (**int** a,**int** b) -> a+b;  
 System.***out***.println(math2.operation(3,2));  
 *//声明类型，加大括号* MathDemo math3 = (**int** a,**int** b) -> { **return** a\*b; };  
 System.***out***.println(math3.operation(3,2));  
 *//单个参数可以不加括号* SayMessage say1 = message -> System.***out***.println(message);  
 say1.say(**"哈哈"**);  
 }  
 **interface** MathDemo{  
 **int** operation(**int** a,**int** b);  
 }  
 **interface** SayMessage{  
 **void** say(String message);  
 }  
}

二. Lambda与foreach

*//使用lambda实现foreach*Map items = **new** HashMap<>();  
items.put(**"A"**, 10);  
items.put(**"B"**, 20);  
items.put(**"C"**, 30);  
items.put(**"D"**, 40);  
items.put(**"E"**, 50);  
items.put(**"F"**, 60);  
items.forEach((k,v)->{System.***out***.println(k+**":"**+v);});  
  
List list = **new** ArrayList<>();  
list.add(**"A"**);  
list.add(**"B"**);  
list.add(**"C"**);  
list.add(**"D"**);  
list.add(**"E"**);  
list.forEach( v -> System.***out***.println(v));