# **Jdk8**

# Lambda表达式

在java中，无法将函数作为参数传递给一个方法，也无法声明返回一个函数的方法

在javascript中，函数参数是一个函数，返回值是另一个函数的情况是非常常见的；javascript是一种非常典型的函数式语言

例子1

|  |
| --- |
| 片段1  package com.gardenia.jdk8;  import javax.swing.\*; import java.awt.event.ActionEvent; import java.awt.event.ActionListener;  public class SwingTest {   public static void main(String[] args) {   JFrame jframe = new JFrame("My JFrame");  JButton jButton = new JButton("My JButton");  // jButton.addActionListener(new ActionListener() { // public void actionPerformed(ActionEvent e) { // System.out.println("Button Pressed!"); // } // });   jButton.addActionListener(event -> {  System.*out*.println("Button Pressed!");  System.*out*.println("Button Pressed!");  System.*out*.println("Button Pressed!");  });   jframe.add(jButton);  jframe.pack();  jframe.setVisible(true);  jframe.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);    }  } |

lambda表达式基本结构：

(param1,param2,param3){

}

函数式接口：

1. 若果一个接口只有一个抽象方法，那么该接口就是一个函数式接口
2. 若果我们在某个接口上声明了FunctionalInterface注解，那么编译器就会按照函数式接口的定义来要求该接口
3. 如果某个接口只有一个抽象方法，的那我们并没有给该接口声明FunctionalInterface注解，那么编译器依旧会将该接口看做是函数式接口。

例子2

|  |
| --- |
| 片段1  package com.gardenia.jdk8;  @FunctionalInterface public interface MyInterface {   void test();   // 加上这个会报错  //String myString();   // 而这个不会报错  // 重写了Object类的方法，不会向抽象接口的方法数量加1  String toString(); }  片段2  package com.gardenia.jdk8;  @FunctionalInterface interface MyInterface2 {   void test();   String toString(); }  public class Test2 {   public void myTest(MyInterface2 myInterface2){  System.*out*.println(1);  myInterface2.test();  System.*out*.println(2);  }    public static void main(String[] args) {  Test2 test2 = new Test2();  // 传统方式  test2.myTest(new MyInterface2() {  @Override  public void test() {  System.*out*.println("myTest");  }  });  // lambda表达式  test2.myTest(()->{  System.*out*.println("mytest");  });   //  MyInterface2 myInterface2 = ()->{  System.*out*.println("hello");  };    } } |

Lambda表达式为Java添加了确实的函数式编程特性，使我们能将函数当做一等公民看待

在将函数作为一等公民的语言中，Lambda表达式的类型是函数。但在Java中，Lambda表达式是对象，他们必须依附于一类特别的对象类型--函数式接口

内部迭代和外部迭代

内部迭代：lambda表达式迭代

# 函数式接口

# 3.方法引用与构造器

# 4.Stream API

# 5.接口中默认方法与静态方法

# 6.新时间日期API

# 7.其他新特性