# Lambda表达式

函数式编程思想

在数学中,函数就是有输入量和输出量的一套设计方案,也就是拿数据做操作.

面向对象强调:必须通过对象的形式来做事情.

函数式思想则尽量忽略面向对象的复杂语法,敲掉做什么,而不是说明形式去做.

Lambda表达式的格式和使用

格式: ()->{System.out.println("夏天到了,我们可以游泳了!");}

(形参) -> {代码块}

():里面没有内容,可以看成方法的形参为空;

->:用箭头指向后面要做的事情;

{}:代码块,就是要做的内容;

示例:

public static void main(String[] args) {

//匿名内部类在括号中创建接口的实现类对象,重写抽象方法

goSwimming(new Swimming() {

@Override

public void swimming() {

System.out.println("夏天到了,我们可以游泳了!");

}

});

//Lambda表达式,函数式编程思想

goSwimming(()->{

System.out.println("夏天到了,我们可以游泳了!");

});

}

public static void goSwimming(Swimming swimming){

swimming.swimming();

}

Lambda表达式的使用前提:

1.有一个接口;

2.接口中有且只有一个抽象方法;

使用Lambda表达式的几种情况:

无参无返回值: () -> {代码块}

有参无返回值: (形参) -> {代码块}

无参有返回值: () -> {代码块 return flag;}

有参有返回值: (形参) -> {代码块 return flag;}

<这里的形参和返回值看的都是接口抽象方法,并不是看测试类中形参为接口的调用方法>

Lambda表达式的省略模式

省略规则:

1.参数类型可以省略,但是有多个参数的情况下,要全部省略,不能只省略一个;

2.如果参数有且只有一个,除了参数类型,还可以省略小括号;

3.如果代码块的语句只有一条,可以省略大括号和分号,甚至return语句.

Lambda表达式和匿名内部类的区别:

所需类型不同

1.匿名内部类:操作对象可以是接口,抽象类,甚至具体类;

2.Lambda表达式:操作对象只能是接口.

使用限制不同

1.如果接口中有且仅有一个抽象方法,可以使用Lambda表达式也可以使用匿名内部类;

2.如果接口中有多个抽象方法,只能使用匿名内部类.

实现原理不同

1.匿名内部类:编译之后,out文件夹内产生一个单独的.class字节码文件;

2.Lambda表达式:编译之后,没有一个单独的字节码文件,对应的字节码会在运行的时候动态生成.