22.1、闭包

# 1、理解闭包的概念

闭包：

闭包能够将一个方法作为一个变量去存储，这个方法有能力去访问所在类的自由变量。

如何用变量去存储方法？：

纯天然的解决办法是：内部类。内部类能够访问外部类的所有属性及方法。

隐藏具体实现是内部类的作用之一，

如何保证隐藏具体实现的同时还能将闭包传递到外部使用？

让内部类实现通用接口，然后将内部类对象向上转型为接口类型。

上述解决办法就是Java最常用的闭包实现办法（内部类+接口）

闭包价值

闭包的价值在于可以作为函数对象或者匿名函数，持有上下文数据，

作为第一级对象进行传递和保存。闭包广泛用于回调函数、函数式编程中。

## 1、建立内部类实现对变量属性的控制

### 1、下面对于内部类的实现用了两种方式，

#### 1、一种类似于方法的形式，需要返回接口对象Active

#### 2、另一种则是比较传统的内部类，实现方式

|  |
| --- |
| **public** **class** Milk {    **public** **final** **static** String name = "纯牛奶";//名称  **private** **static** **int** *num* = 16;//数量  **public** **Milk**(){  System.out.println(name+"：16/每箱");  }    /\*\*  \* 闭包，通过调用Active的方法实现对Milk私有变量num进行修改。  \* **@return** 返回一个喝牛奶的动作 ，和下面的了内部类操作是一样的  \*/  **public** Active **HaveMeals**(){  **return** **new** Active(){  **public** **void** **drink**(){  **if**(*num* == 0){  System.out.println("木有了，都被你丫喝完了.");  **return**;  }  *num*--;  System.out.println("喝掉一瓶牛奶");  }  };  }    /\* public class HaveMeals implements Active{    @Override  public void drink(){  if(num == 0){  System.out.println("木有了，都被你丫喝完了.");  return;  }  num--;  System.out.println("喝掉一瓶牛奶");  }  }\*/    /\*\*  \* 获取剩余数量  \*/  **public** **void** **currentNum**()  {  System.out.println(name+"剩余："+*num*);  }  }  /\*\*  \* 通用接口  \*/  **interface** Active  {  **void** **drink**();  } |

## 2、开始测试闭包的使用

### 1、注意调用方式

#### 1、Active haveMeals = m.HaveMeals();

#### 2、Active haveMeals = new Milk().new HaveMeals();

|  |
| --- |
| **public** **class** PersonMain {  /\*\*  \*  \* **@author** :HealerJean:  \* **@date** :2017年11月16日 下午3:30:32  \* **@Description**:特别注意：闭包会导致资源不被回收，  \* 在main方法中将m设为null，使用haveMeals继续调用drink方法仍然会喝掉一瓶牛奶，  \* 说明Milk对象并没有被释放掉。  \*/    **public** **static** **void** **main**(String[] args) {  //买一箱牛奶  Milk m = **new** Milk();  //内部类，同下  Active haveMeals = m.HaveMeals();    // 调用静态内部类  // Active haveMeals = new Milk().new HaveMeals();  //没事喝一瓶  haveMeals.drink();  //有事喝一瓶  haveMeals.drink();    //看看还剩多少？  m.currentNum();  }  } |

## 3、测试结果，

