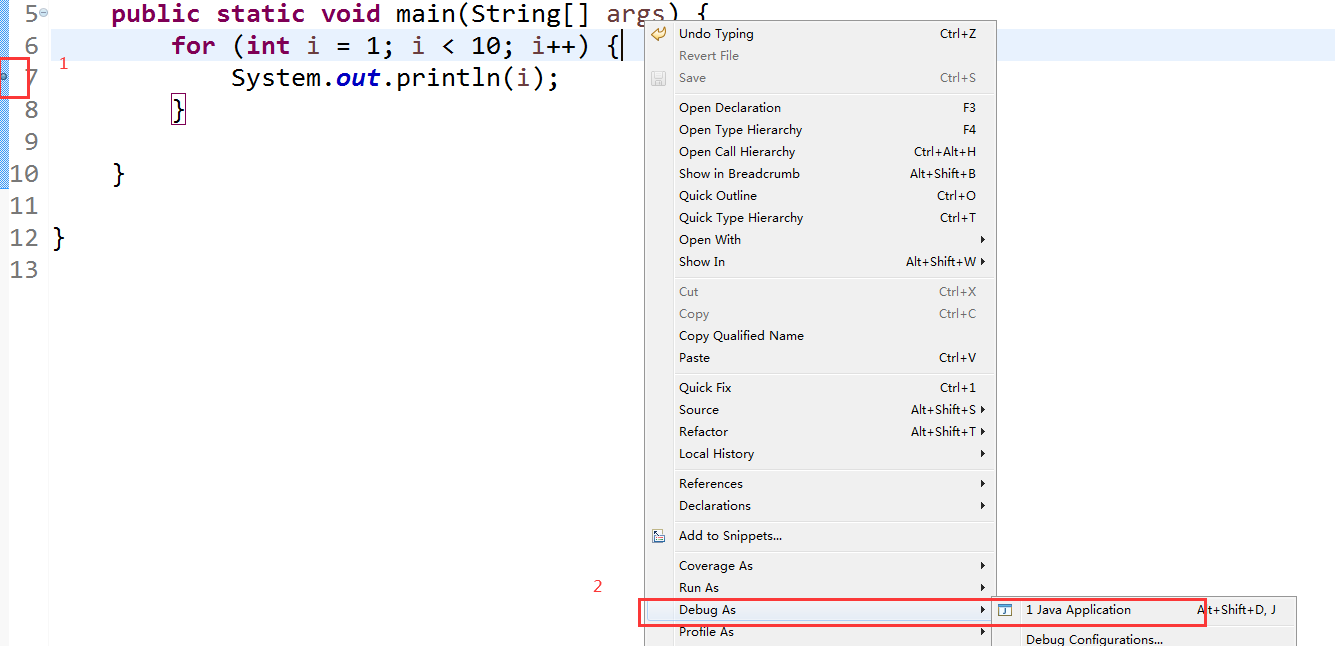
# 回顾:

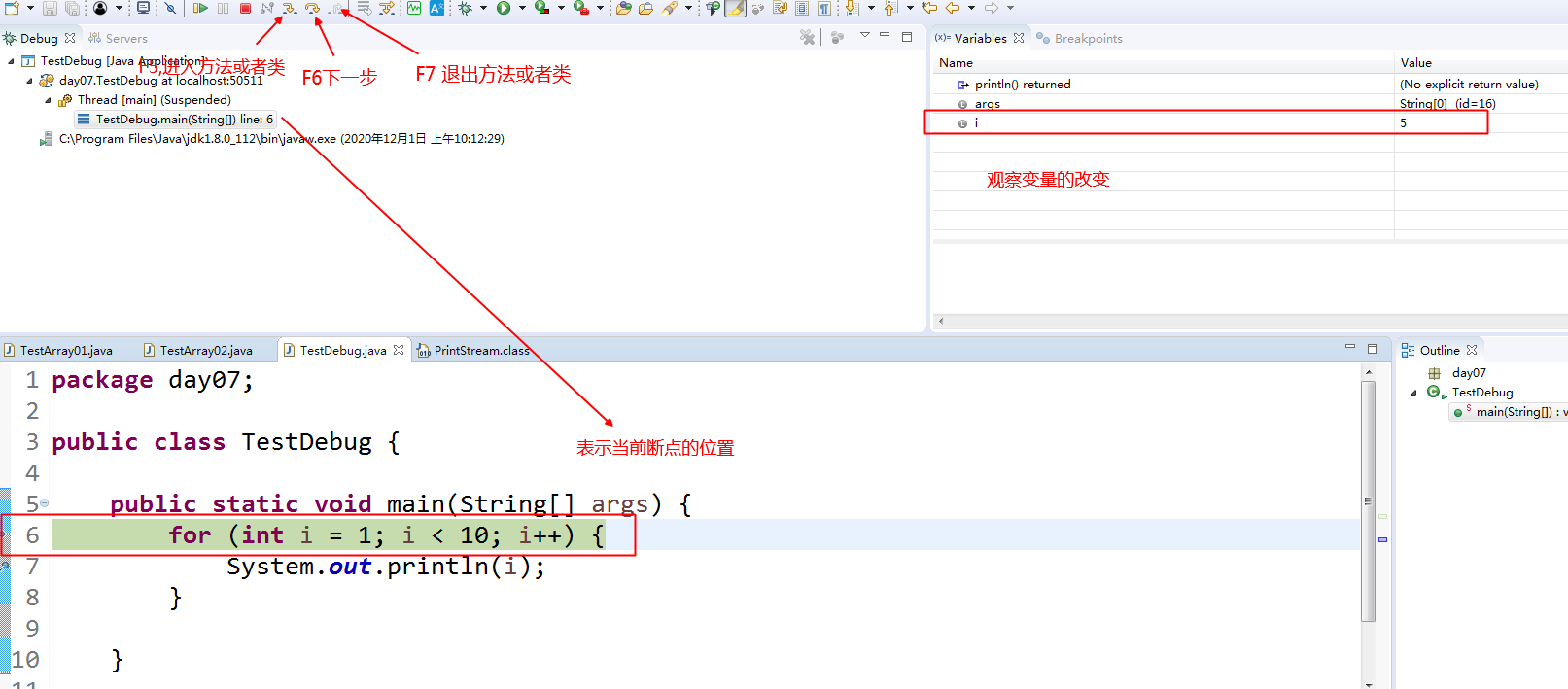
1. 冒泡:概念,实现:2个for循环分别起的作用
2. 二维数组可以理解成表格:有行有列,行当成一维数组,列当成一维数组中的元素,行有下标(几个一维数组),列也有下标(一维数组中的第一个元素)
3. 定义二维数组: 数据类型[][] 数组名=new 数据类型[行数][列数]

# Debug

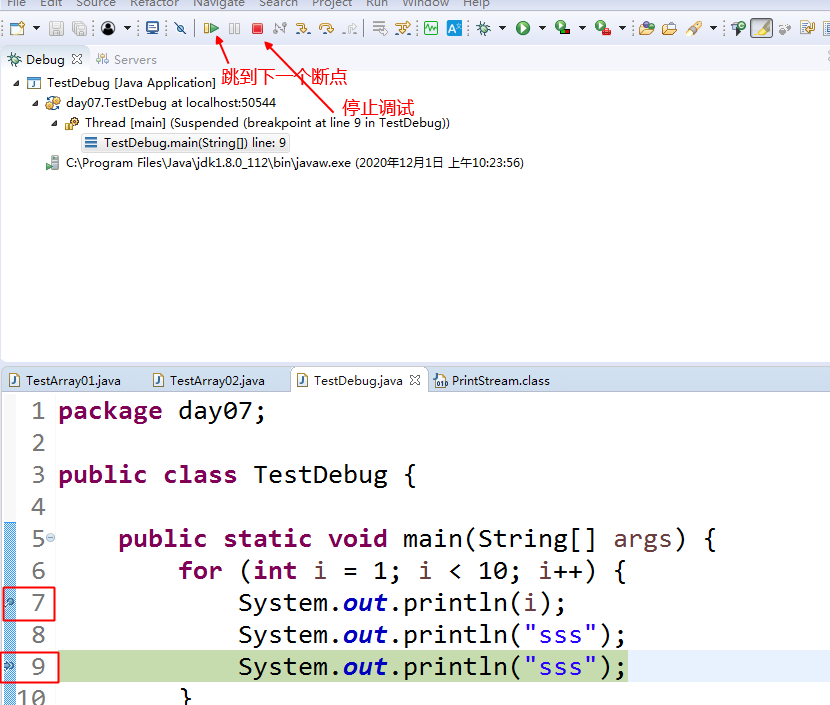
## 1.1设置断点,右键选择 debug as



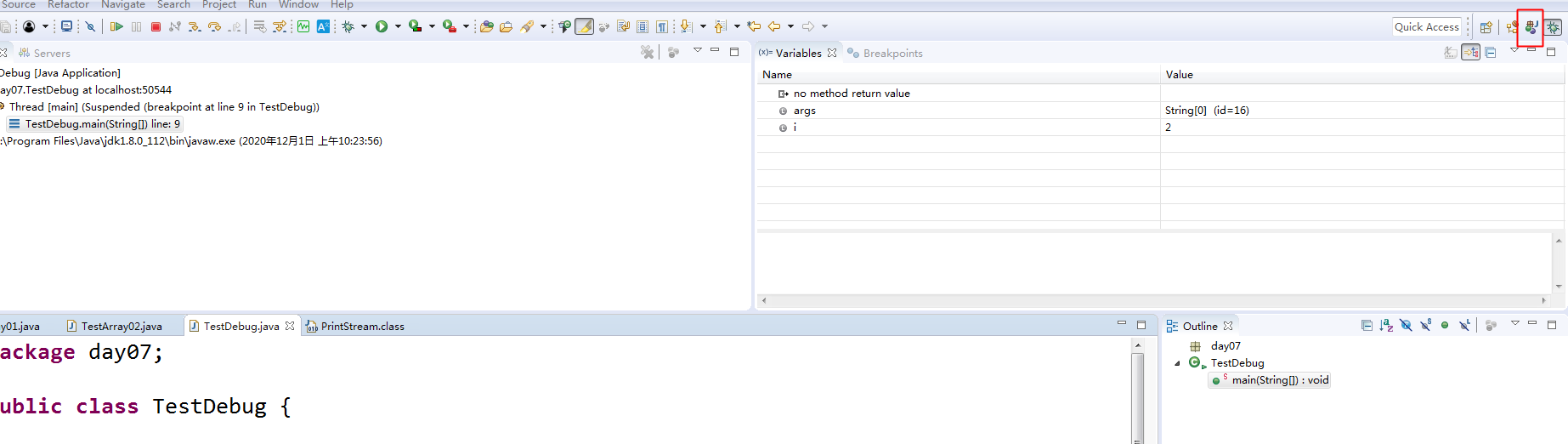
## 1.2调试界面:



## 1.3停止调试

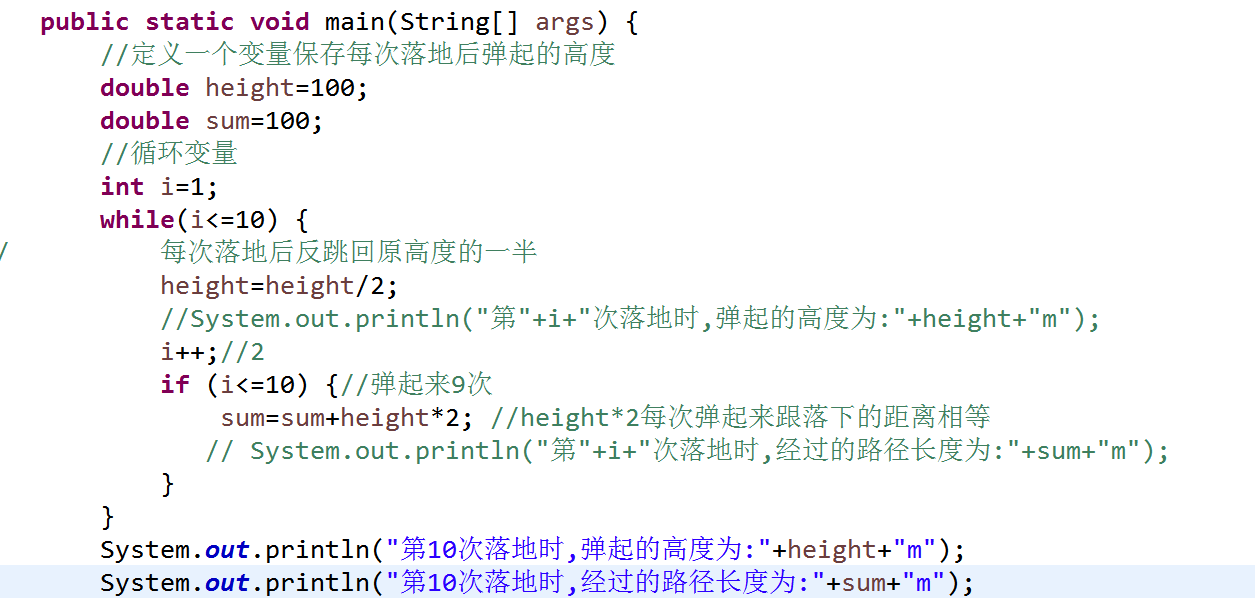


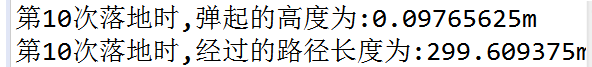
## 1.4回到编码界面



## 1.5小球落地

一球从100米高度自由落下，每次落地后反跳回原高度的一半；再落下，求它在第10次落地时，共经过多少米？第10次落地反弹多高？（DEBUG）





## 1.6题目:猴子吃桃

猴子第一天摘下若干个桃子，当即吃了一半，还不过瘾，又多吃了一个 ;第二天早上又将剩下的桃子吃掉一半，又多吃了一个。以后每天早上都吃了前一天剩下的一半零一个。到第10天早上想再吃时，见只剩下一个桃子了。求第一天共摘了多少。 请使用for和while分别处理;

分析: 一共吃了九天;倒着推理;

第10天早上开始: 没吃, 1

第9天早上: 没吃, (1+1)\*2=>4

第8天早上开始: 没吃, (4+1)\*2 =>10

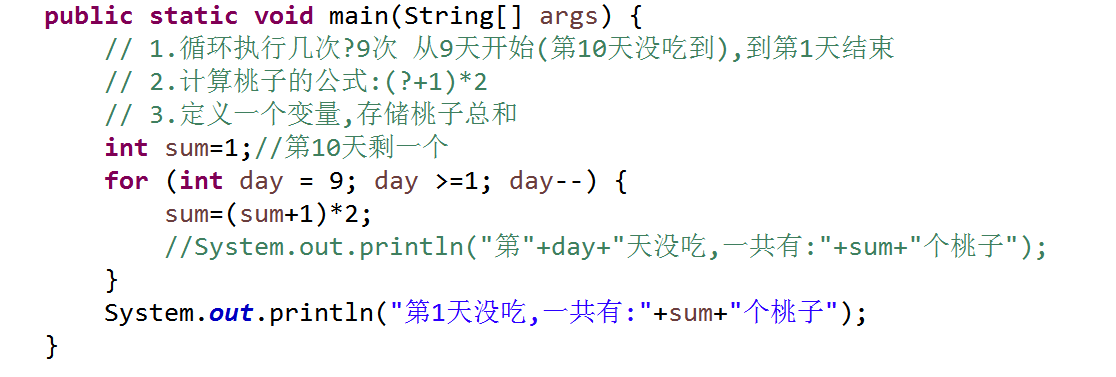
…

第1天早上，摘了，还没吃 (?+1)\*2 =>

以上的分析可以得到：

1. 循环执行几次;

2. 计算桃子个数的表达式 (?+1)\*2



# 类和对象:

## 面向对象:

面向对象的设计可以理解为:

将数据及对数据的操作封装在一起，成为一个不可分割的整体。

同时将具有相同特征的对象抽象成一种新的数据类型---类;

通过对象间的消息传递使整个系统运转，通过类的继承实现代码重用。

# 对象的概念：

概念:具体”明确”的属性值和方法的1个实体,对象是类的”实例”:实际的例子

举例子：

世界杯比赛： 足球 球员 教练 裁判...

球员这个类可以实例化出哪些对象?

C罗: 姓名=C罗 身高 190 体重 150 性别:男 踢球(方法)

梅西: 姓名=梅西身高 190 体重 150 性别:男 踢球(方法)

在Java中**万事万物皆对象(存在的可以被描述的任何事物)**

现实生活中的对象:在座的各位都是对象,电脑,每一台电脑都是一个对象;等等

举例:

裴瑾:个人基本信息(静态的描述) 23岁 性别男 爱好:吃

裴实:个人基本信息(静态的描述) 26岁 性别男 爱好:睡觉

继续细分:

具体说:个人特征称为:属性;;行为称为:方法

对象=属性+方法

对象依赖于类存在（模板-个体实例）

分析过程先有对象后有类，但是开发过程先有类后有对象

类型 引用名 = new 类名();

# 类的概念:

概述:类就是一个模板,它是提供同一类对象的属性和方法.

举例子：

世界杯比赛： 足球 球员 教练 裁判...

C罗: 姓名=C罗 身高 190 体重 150 性别:男 踢球(方法)

梅西: 姓名=梅西 身高 190 体重 150 性别:男 踢球(方法)

以球员为例具有共同的属性:姓名,性别,身高,体重

具有共同方法:踢球,射门,投球……..

类是描述对象的“基本原型”，它定义一种对象所能拥有的数据和能完成的操作,在面向对象的程序设计中,类是程序的基本单元，最基本的类由一组结构化的数据和在其上的一组操作构成。

将多个对象中公共的部分(只关注其中相同或者相似)提取出来(提取的过程，称为抽象);形成一个集合，这个集合，理解为一些对象的公共特征和行为，称为**类**;

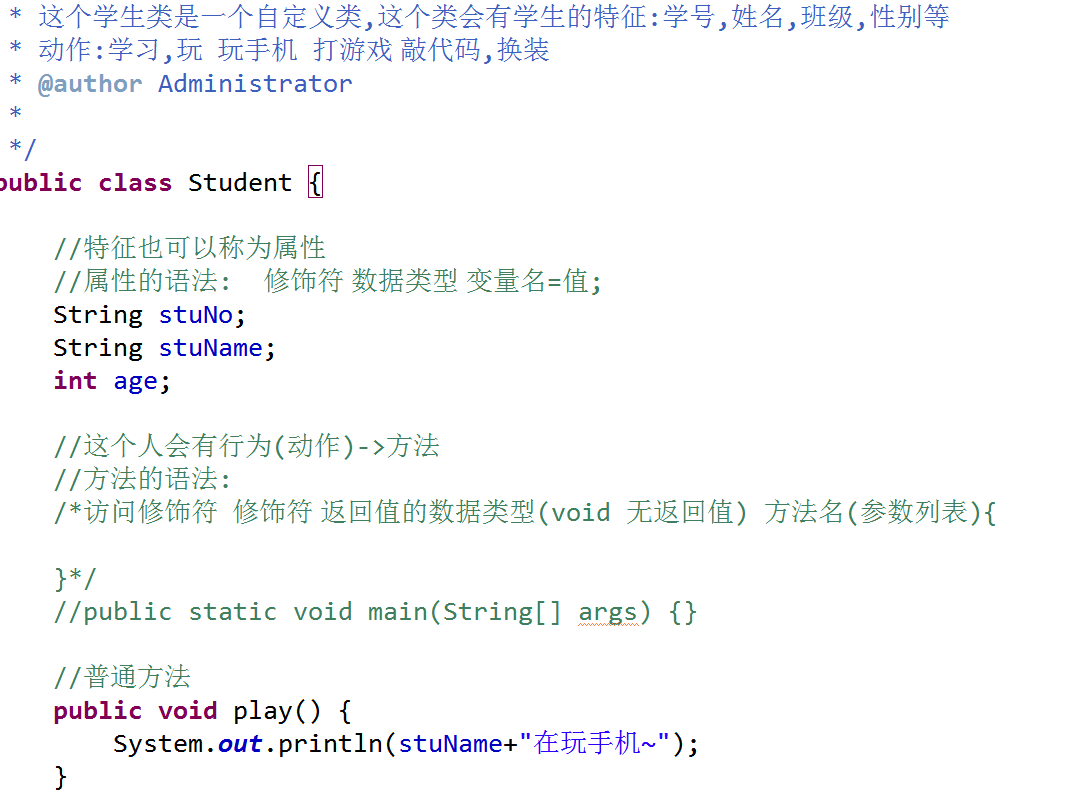
类是一个模板，是一个概念性模型，我们可以通过类，得到具体的对象:

分析的过程先有对象后有类,开发过程先有类后有对象;

创建对象的语法:

类型 引用名 = new 类名();

## 类的基本结构:



## 对象的创建：

类名 引用名 = new 类名();

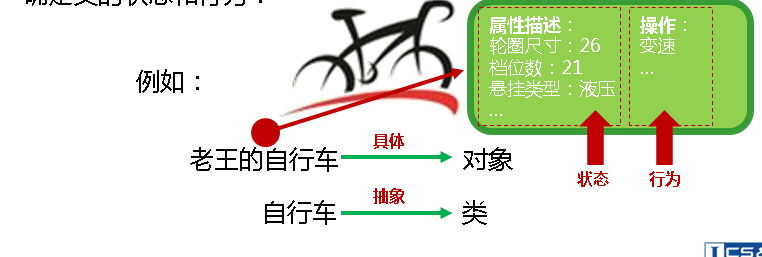
类名 引用名 = new 类名(参数);

测试类:

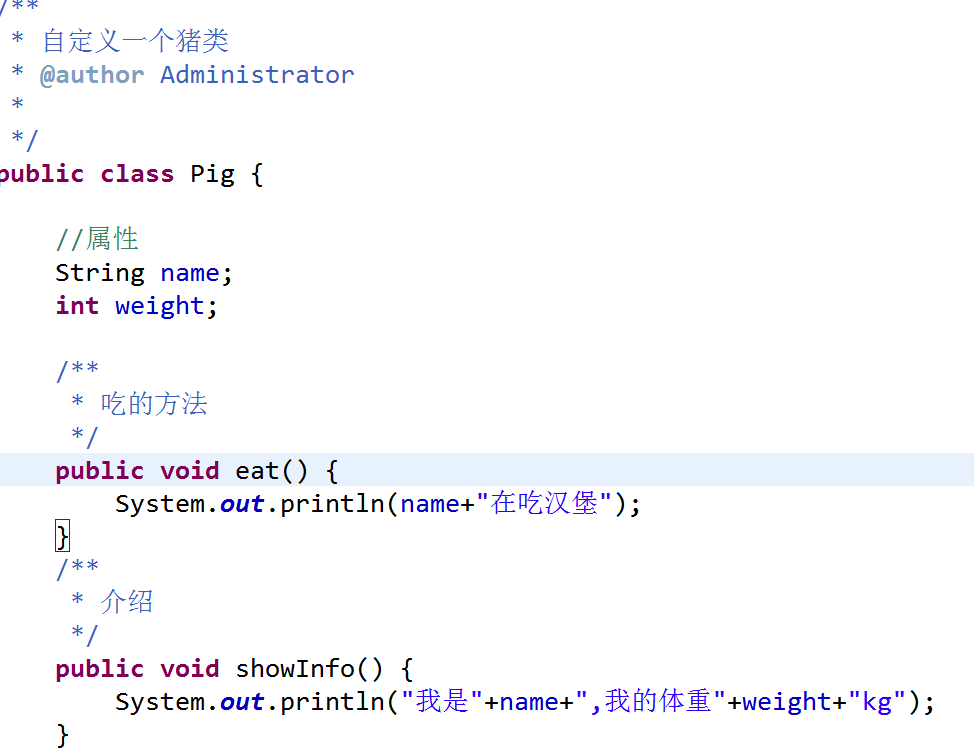


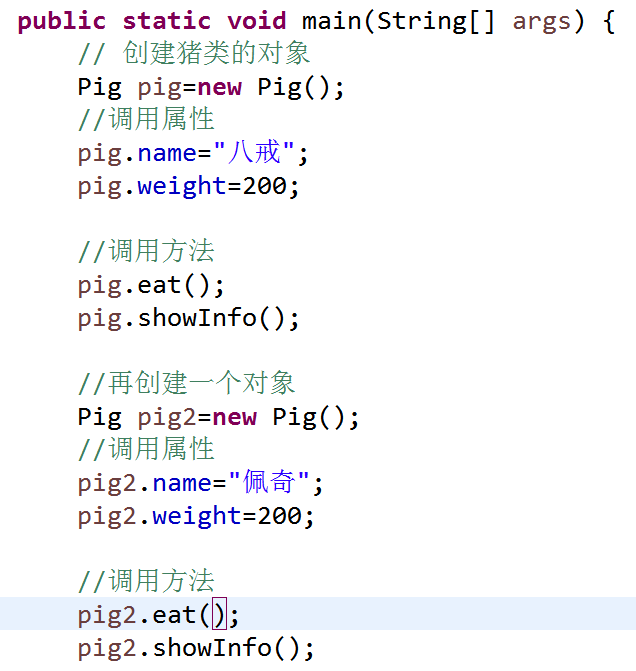
## 类和对象关系与区别：

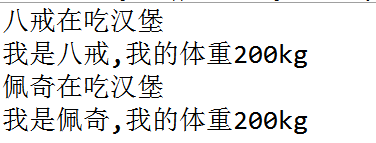
类是同等对象的集合与抽象。它是一块创建现实**对象的模板**。**对象是类的实例**，对象是面向对象编程的核心部分，是实际存在的具体实体，具有明确定义的状态和行为。



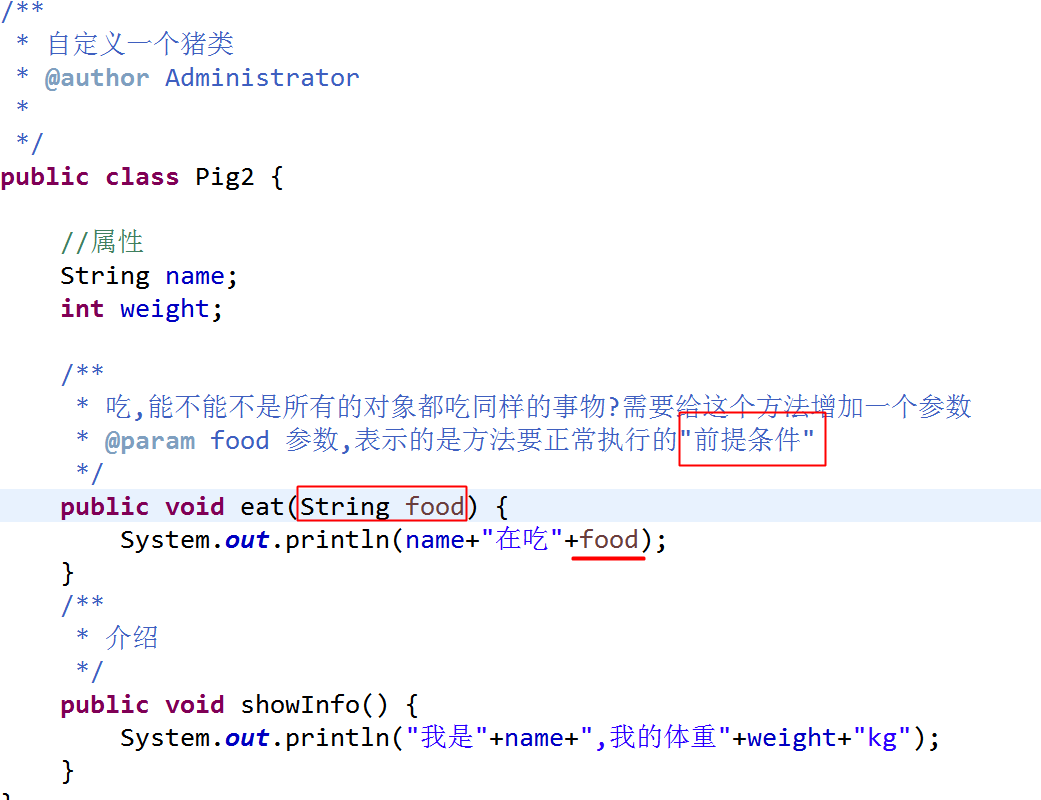
## 课堂练习:自定义一个pig类,这个类中包含了属性和方法

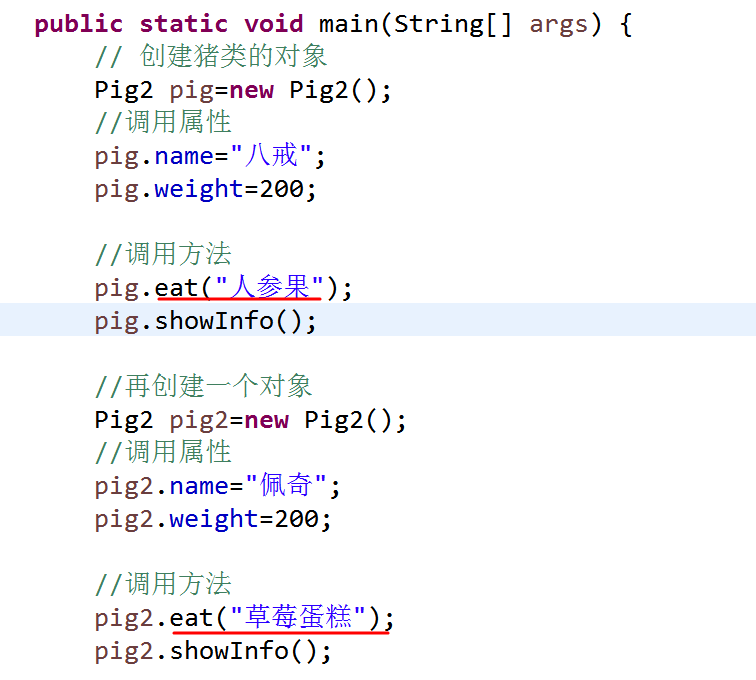


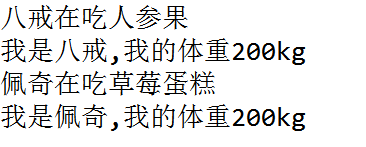




可以这么改:







## 类中方法的定义:普通方法

### 目的：

提取共同的代码块放在一个方法内，可以被重复的调用。

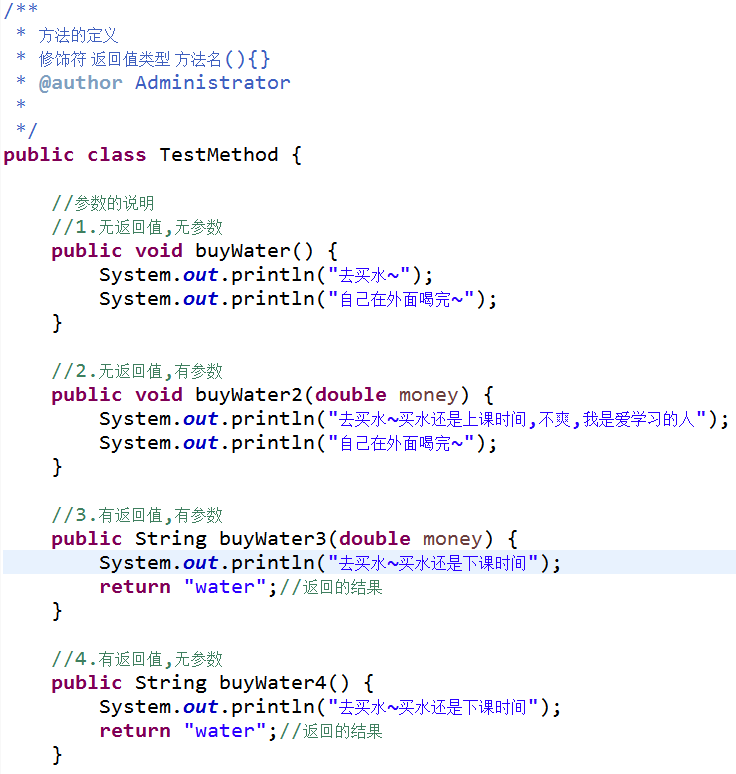
### 方法类型：

无参数无返回值方法

有参数无返回值方法

有参数有返回值方法

无参数有返回值方法



### 自定义方法的语法：

访问修饰符 返回值类型 方法名(参数列表){

// 方法体

// return 值;

}

①访问修饰符：public protected 缺省 private

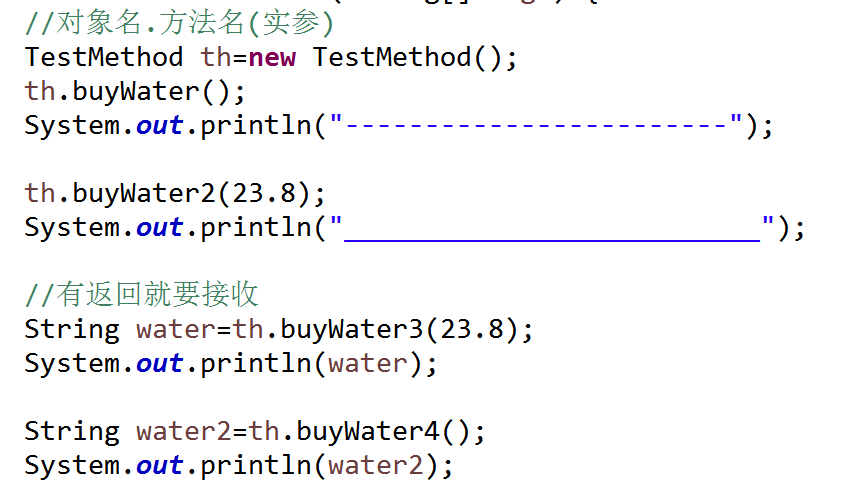
②返回值类型：使用return关键字返回数据，每一个方法只能有一个返回值，返回值类型根据实际情况决定的，取决于返回的值的数据类型。如果一个方法没有返回值，则使用void关键字进行定义

③参数列表：某一个方法执行的时候需要一些前提条件，此时可以将这些条件理解成参数，从外部传递进来。参数列表个数0-N个

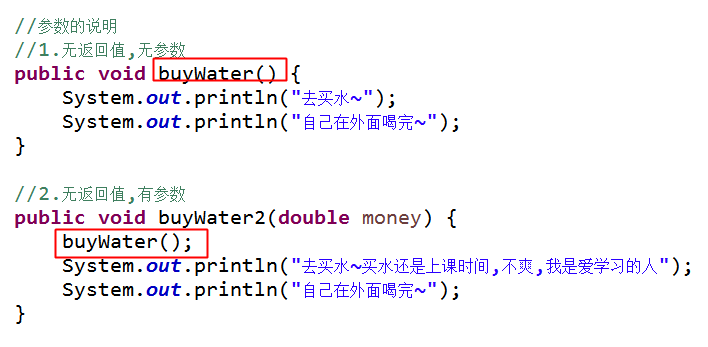
方法名(数据类型 形参名**,** 数据类型 形参名){....}

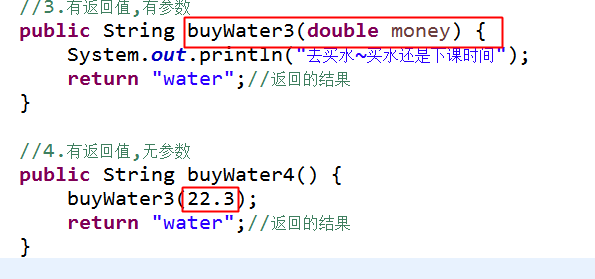
### 方法的调用：

①通过对象名.方法名(实际参数)进行调用

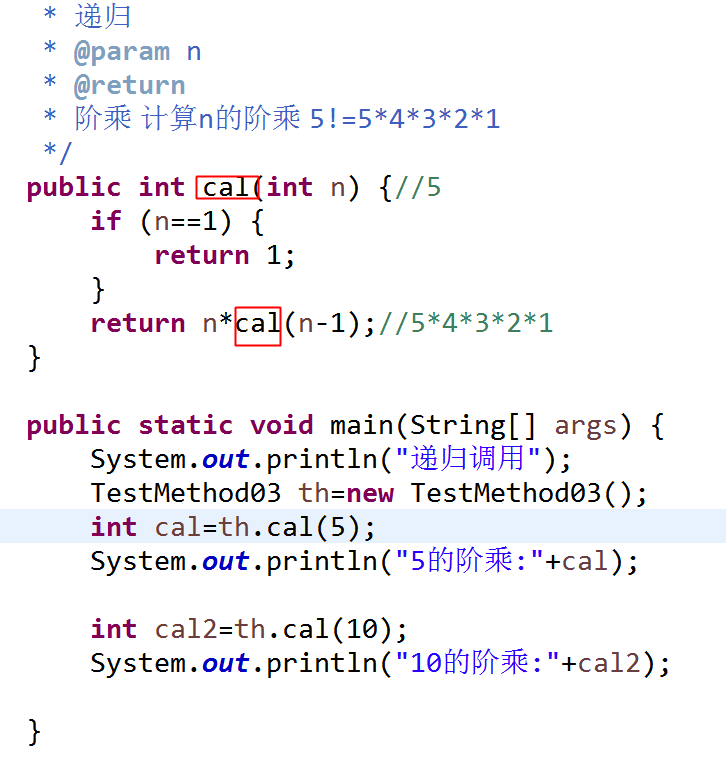


②同一个类中方法之间相互调用 通过方法名(实际参数)调用

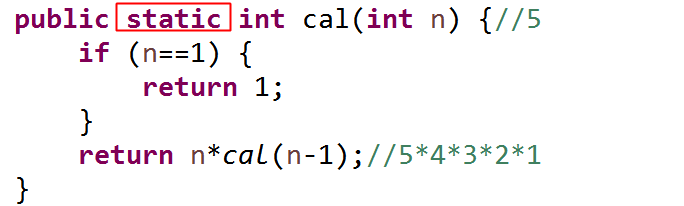


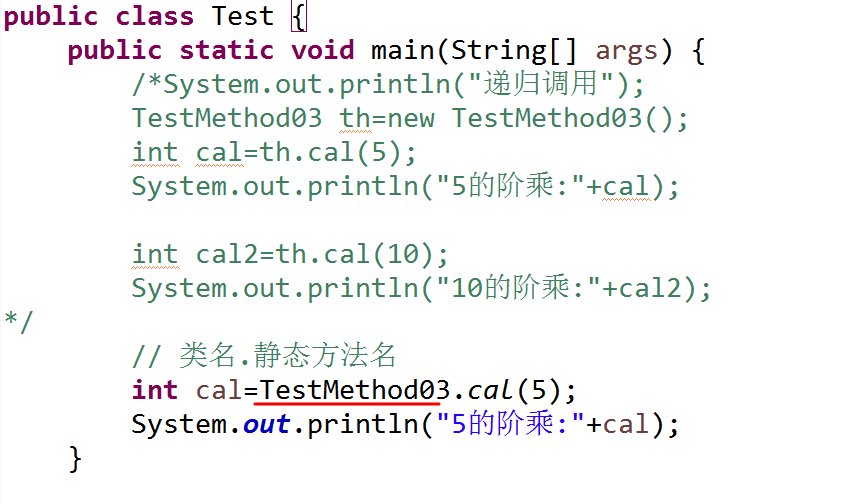


③方法自己调用自己，递归调用，必须杜绝死循环情况。



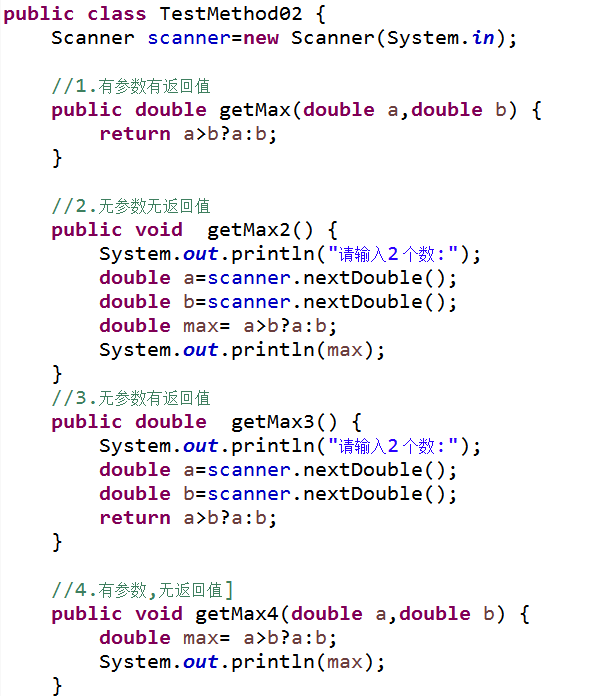
④用static修饰的方法称为静态方法，直接使用 类名.静态方法名(实际参数)调用



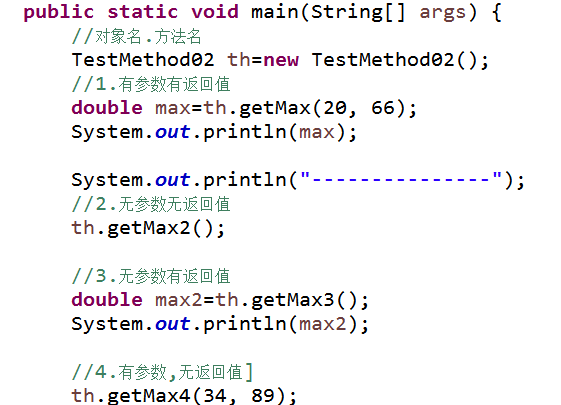


### 课堂练习：

1 使用无参数/无返回值 ，无参数/有返回值，有参数/无返回值/，有参数/有返回值，完成同样的 求两个数字的最大值?



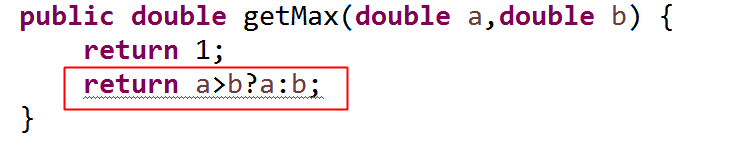
方法的调用:



### 关于return:

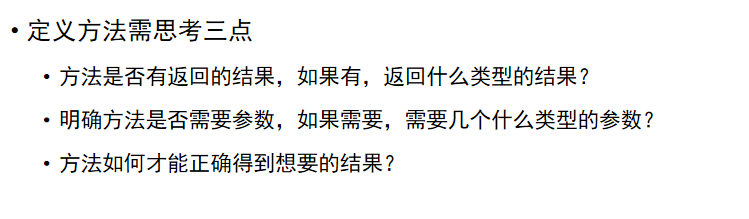
一个方法中的return最多只能执行一次，return意味这个方法的执行已经终止;

这个方法的后续代码将不会执行



## 继方法相关

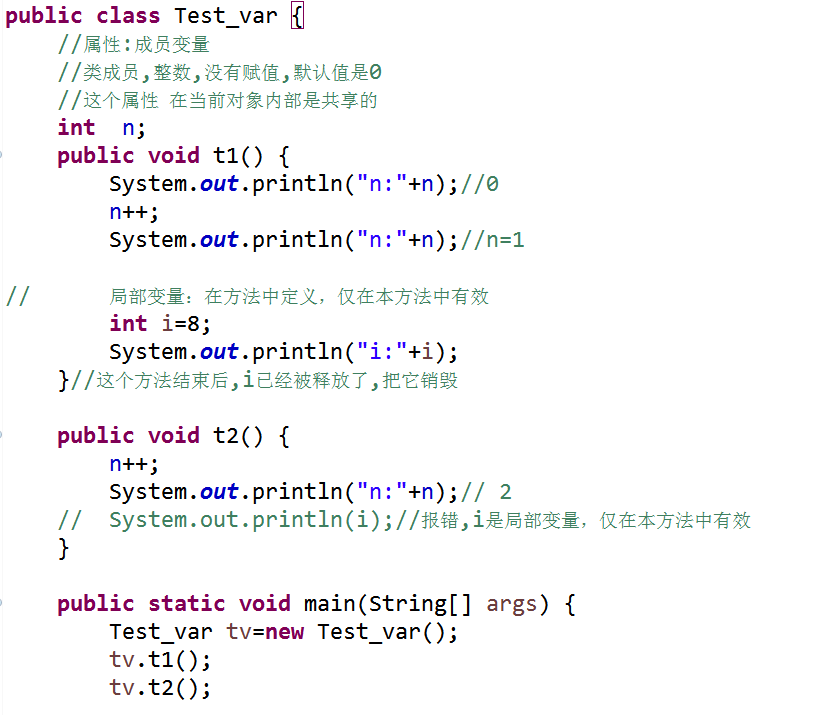
### 定义方法需思考的三点



### 方法体中的局部变量-->变量的作用域

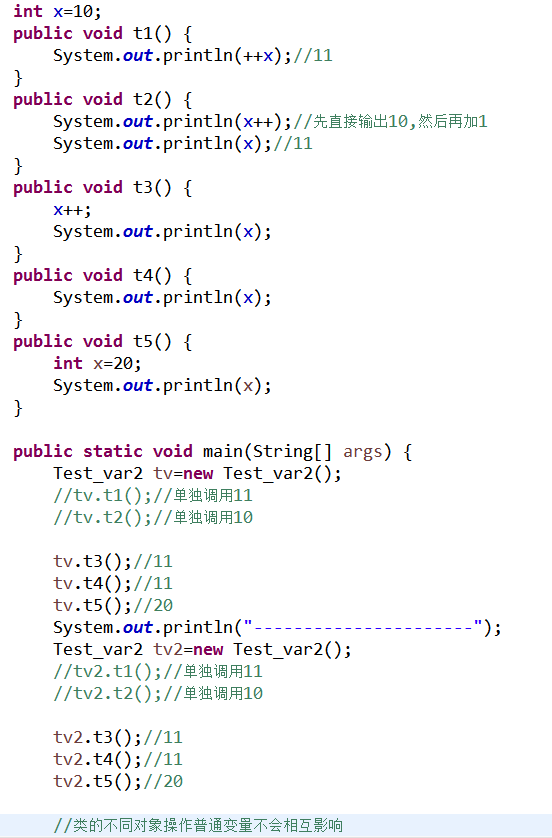
成员变量：在类中定义，在当前对象的内部是共享的

局部变量：在方法中定义，仅在本方法中有效



### 小题目：

类的不同对象操作普通变量不会相互影响。每一个对象都有一份普通成员变量的拷贝。



# 总结

1. debug:①打断点 ②右键debug as ③F8:跳到下一个断点 F6下一步 F7退出方法/类 F5进去方法 ctrl+F2:停止调试
2. 类:类是对象模板
3. 对象:对象是类的实例
4. 类:属性 方法
5. 创建对象: 类名 对象名= new 类名();
6. 方法:4种

7.变量的作用域:成员变量 局部变量