[构造方法 1](#_Toc527980525)

[构造方法的作用 2](#_Toc527980526)

[构造方法的重载 3](#_Toc527980527)

[This关键字 3](#_Toc527980528)

[This关键字的另外用法 4](#_Toc527980529)

[Static关键字 5](#_Toc527980530)

[静态和非静态的区别 6](#_Toc527980531)

[构造方法和静态内容的执行顺序 7](#_Toc527980532)

[静态具有唯一性特征 7](#_Toc527980533)

[封装 9](#_Toc527980534)

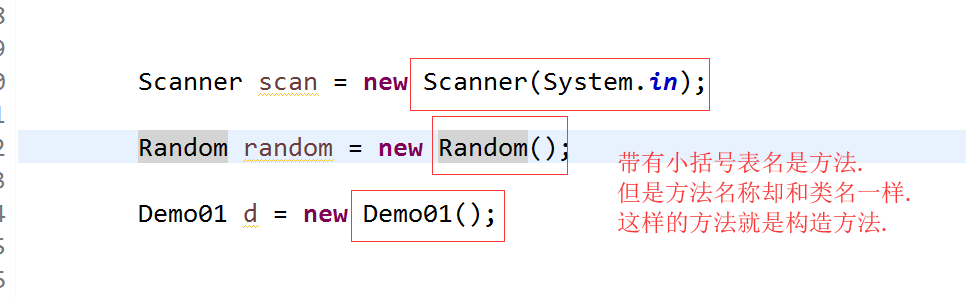
[Get方法 9](#_Toc527980535)

[Set方法 9](#_Toc527980536)

# 构造方法

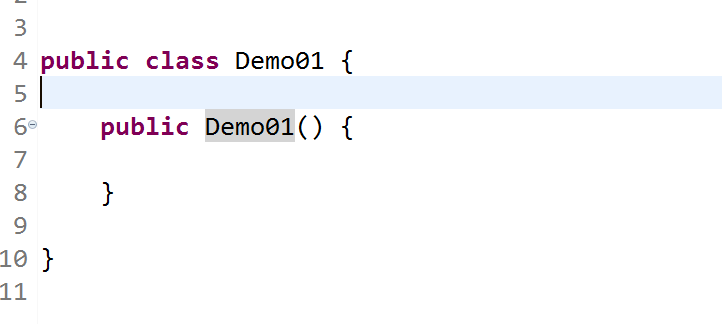
正常情况下每一个类都会有构造方法.构造方法的作用是用于构造对象.

其实我们已经使用过很多次构造方法:



构造方式也是方法.但是和普通方法略微有点不一样

* 方法名称是固定的.只能和类名一样.
* 普通方法可以选择没有返回值使用void来表示.而构造方法连void都没有.



## 构造方法的作用

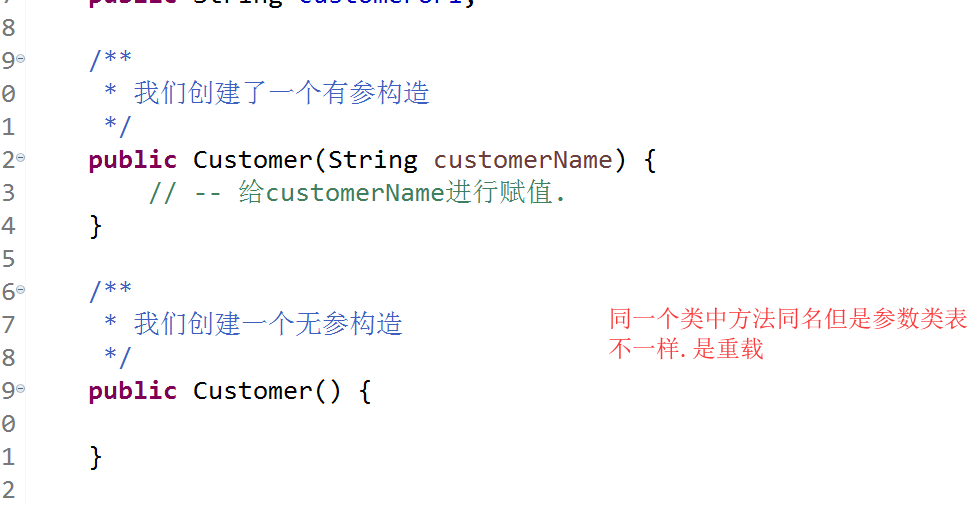
构造方法的作用只有一个构造对象.它只可以被new调用.我们想像调用普通方法那样调用构造方法是不可以的.

我们希望当对象创建时就已经具备的特征值会写在构造方法中.

构造方法对于任何一个类来说都有.且默认情况下JVM会给每个类提供一个无参且隐藏的构造方法.当我们一旦显示的提供任何一个构造方法时,默认的那个就会被覆盖.

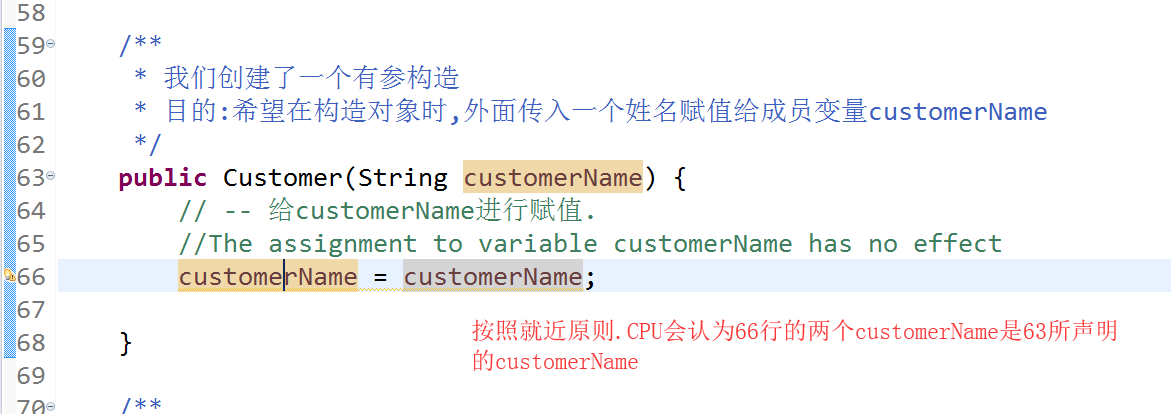
## 构造方法的重载

重载最常见的就是构造方法的重载.



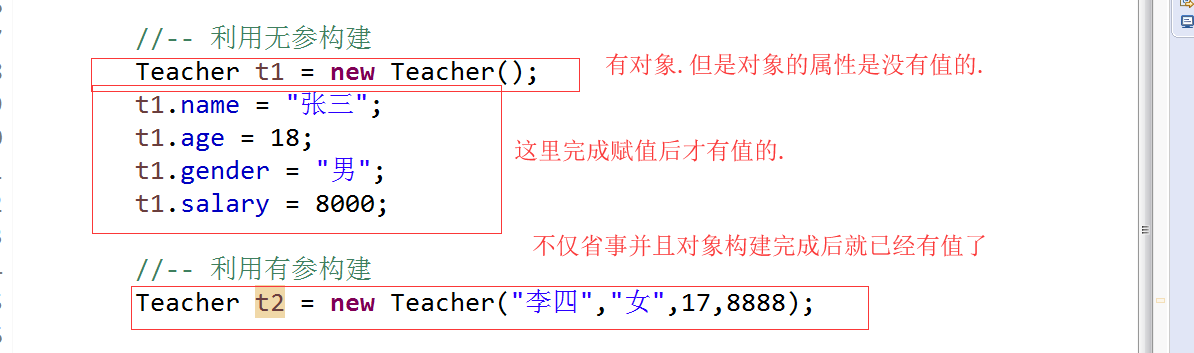
## This关键字

Java中变量赋值时如果遇到同名问题.遵守就近原则.我们可以理解代码中的同名变量是谁跟谁.但是CPU不知道.所以CPU就会查找变量的声明所在.那在查找的过程中.就遵守就近原则



我们通过this来代表当前.通过this.变量代表是成员变量.所以对上面代码进行改写就是如下:





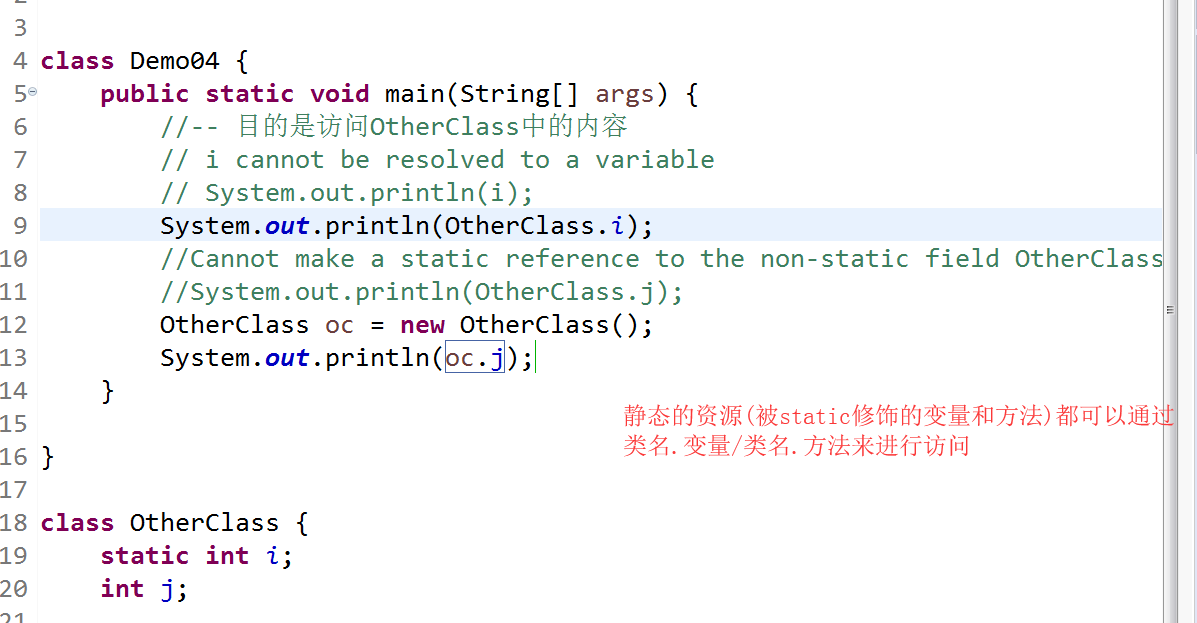
### This关键字的另外用法

This关键字除了用于帮助JVM来识别同名变量以外还可以被用来调用本类的其它构造方法.



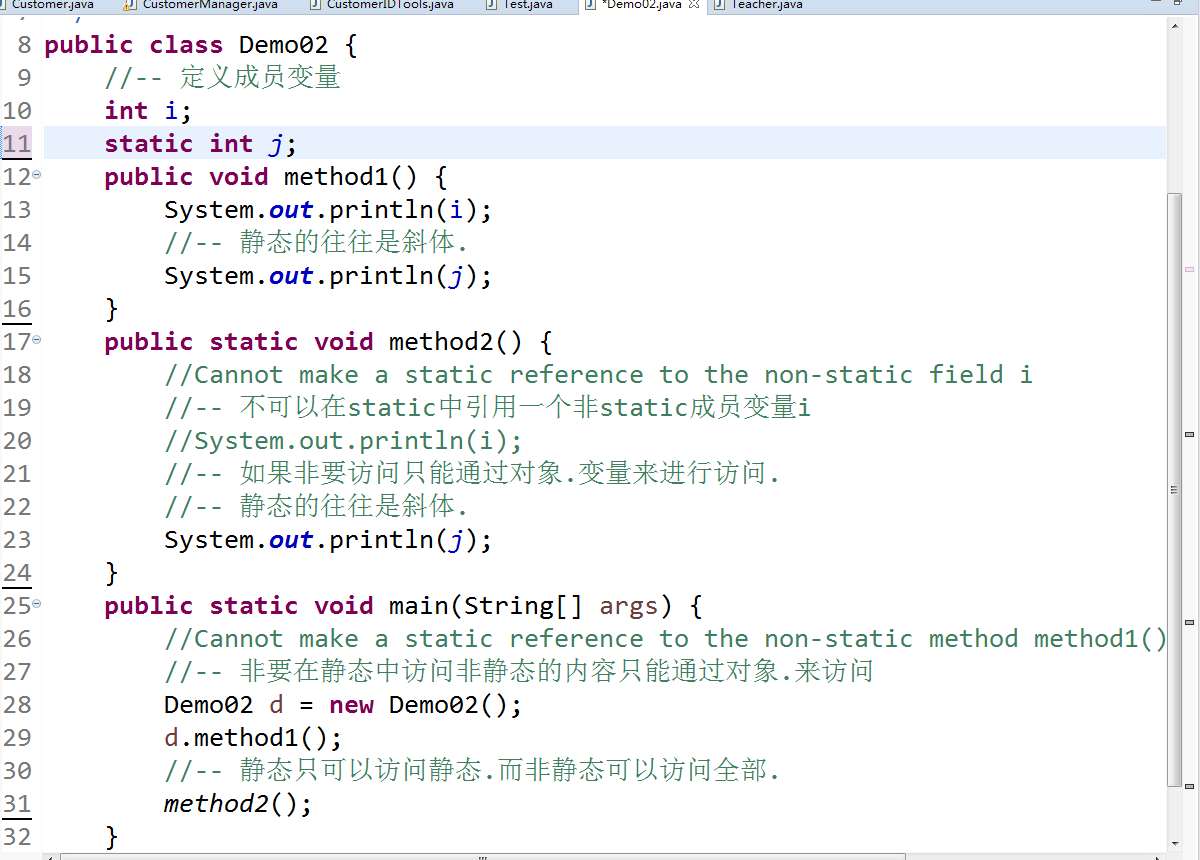
# Static关键字

Static我们把它叫做静态.因为访问方式是通过类名.进行调用所以凡是静态的内容都是属于类的.这个说法是为了区别于非静态内容需要对象.来进行访问.所以非静态内容也被称之为是属于对象的.



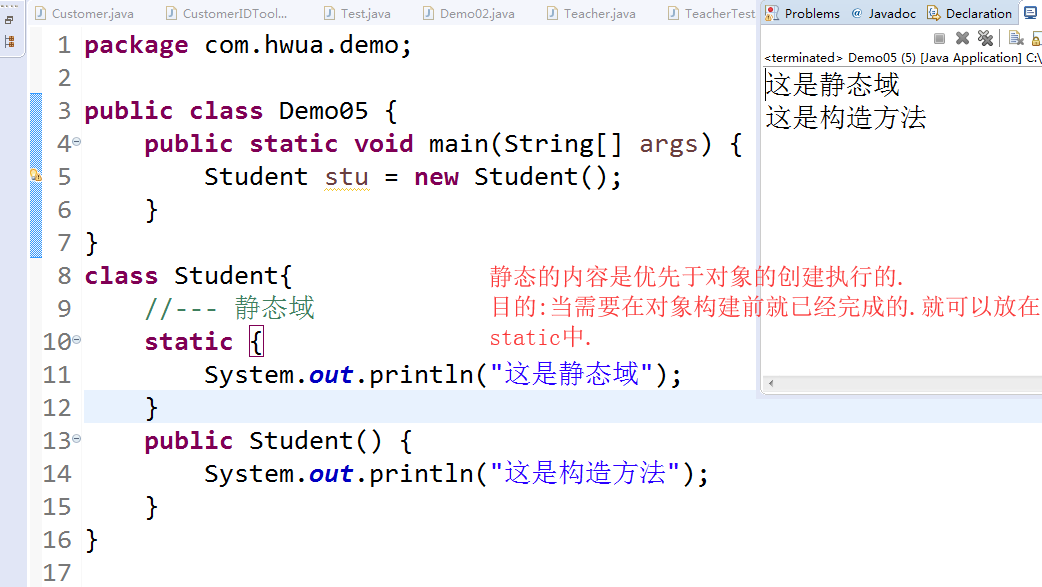
## 静态和非静态的区别

静态只可以访问静态,而非静态可以访问静态和非静态.



## 构造方法和静态内容的执行顺序

这曾经是面试过程中的常考点.它考察的是当代码被执行时,代码中内容的执行顺序.

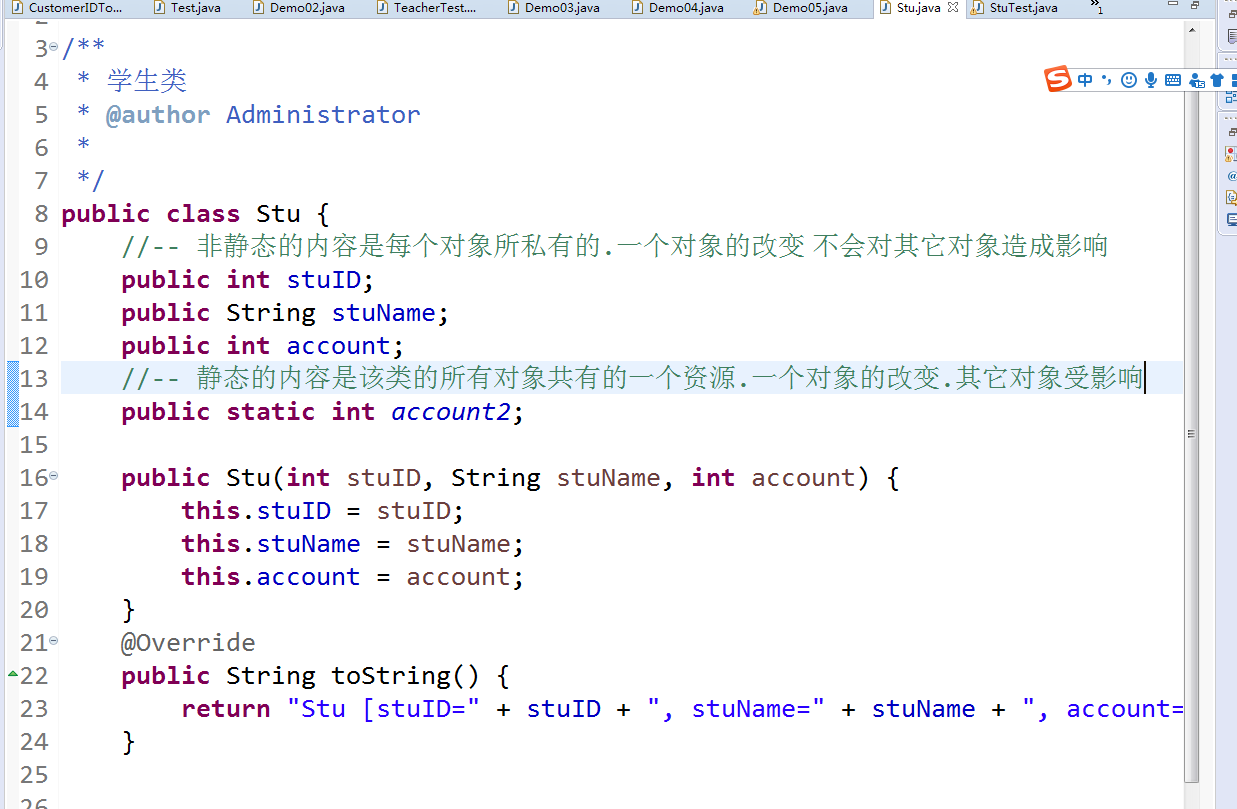


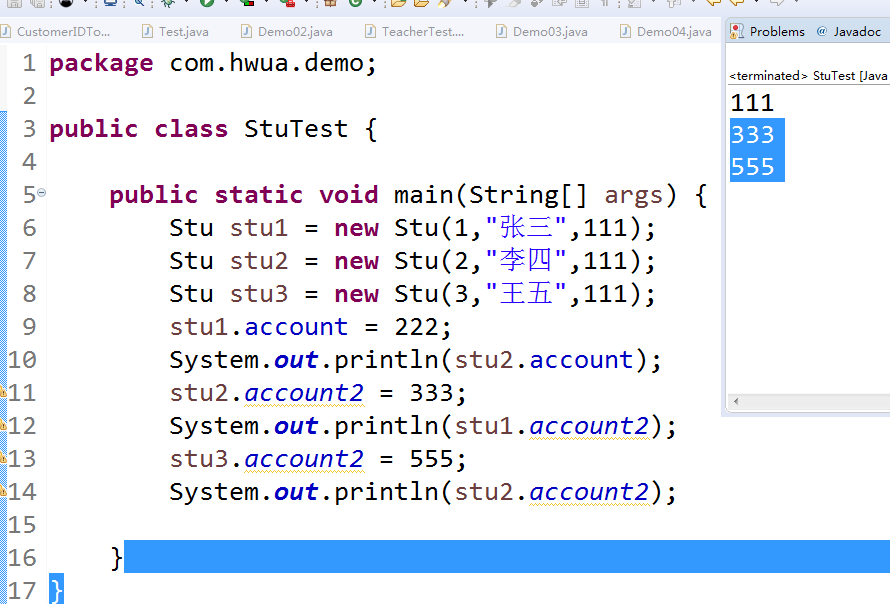
静态是优先于对象构造执行的.至于具体为什么要参考类加载器和类加载过程.这是最后一阶段学习的.

## 静态具有唯一性特征

这是和非静态变量进行对比得到的特征.每一个对象都是一个实例.比如班级中的27名学员.每个人都是一个对象,即都是一个实例.假设这个实例中有字段:银行账号.问这个账号是每个人特有的还是所有人共有的.

如果这个账号是非静态的.那则就是每个人私有的.如果这个账号是静态的.则是共有的.





# 封装

封装可以简单理解为打包的意思.因为Java中有访问修饰符.

|  |  |
| --- | --- |
| 访问修饰符 | 含义 |
| Public | 公共的 |
| Private | 私有的 |
| Protected | 受保护的.等同于包.特殊在于继承. |
|  | 默认,等同于包权限(同包可以访问) |

即使我们什么都不写.那类中的变量在同一个包中.也是随处可以访问的.这样说带来的后果是.数据非常的不安全

如同我们办理存款.不是自己把钱放入金库的.我们都交给柜台人员进行存款和取款办理.

银行中的那个玻璃窗隔开了我们和金库.目的是保护金库.效果可以有一下几点

* 看不见
  + 把访问修饰权限降低到私有.只有自己可以看别人都不可以.
* 摸不着
  + 对于数据不再是直接访问而是间接访问.
  + 对于数据的操作可以归纳为2点
    - 读操作
    - 写操作

封装的基本思路可以分为两个步骤

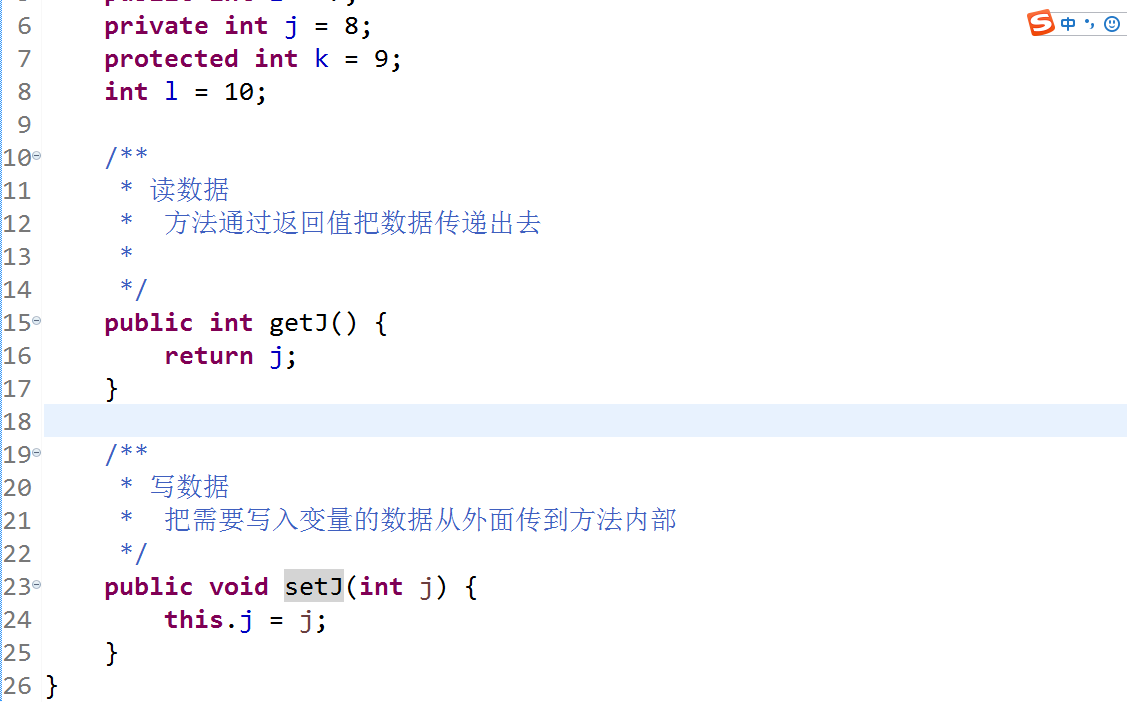
1. 将字段私有化
2. 为字段提供相应的get和set

## Get方法

用于取值.我们通过方法的返回值把数据从方法内部传递给调用方法的地方

## Set方法

用于赋值.把外界需要写入到内部的数据.通过方法的参数传递到方法内部.



## Get和Set方法的取舍

Get和Set方法都必须要提供吗?并不是都需要提供.可以根据实际的需求来提供get或者set或者全部提供.如果是只读的只给get就可以了如果是只写的只给set就行了.如果全部都需要就全部提供.并且我们还可以在set中对传入的数据进行验证.