[继承 1](#_Toc528070807)

[继承的好处 2](#_Toc528070808)

[继承的弊端 2](#_Toc528070809)

[Java中继承的使用方式:extends关键字 2](#_Toc528070810)

[Super关键字 5](#_Toc528070811)

[Super和This的用法对比 6](#_Toc528070812)

[Final关键字 6](#_Toc528070813)

[方法的重写 9](#_Toc528070814)

[父类和子类的构造顺序(涉及Static) 10](#_Toc528070815)

[重写和重载的区别 10](#_Toc528070816)

[Java的跟类:Object 10](#_Toc528070817)

[Object中常见的几个重写方法 10](#_Toc528070818)

[单例设计模式 12](#_Toc528070819)

[懒汉式 12](#_Toc528070820)

[饿汉式 12](#_Toc528070821)

[Java的内存管理 12](#_Toc528070822)

# 继承

继承是Java面向对象三大特性之一.Java面向对象有三大特性:

* 封装
* 继承
* 多态

继承的本质是基于已有的扩展新的.Java的设计者使用了一个很准确的英文:extends.而实际上表达我们正常理解的继承应该是:inherit.extends本身所表达的含义就是扩展.

我们把这一套理念应用于class上,就是基于已有的类,扩展新的类.

## 继承的好处

认为是代码的复用也行.因为新的类是基于已有类扩展得到的.那么新的类中就包含了已有类的全部内容.

## 继承的弊端

继承会打破封装性.之所以叫做封装简单说就是不让你看.

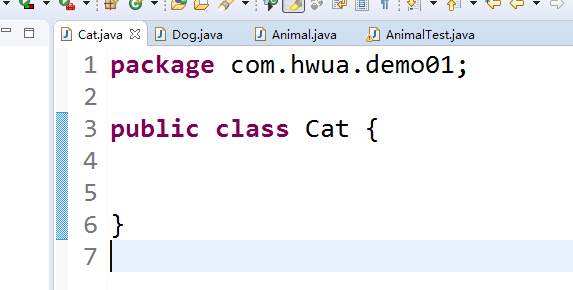
并且如果继承的关系链太长的话.任何一个的改变,都会对那些扩展出来的新类有很大的影响.

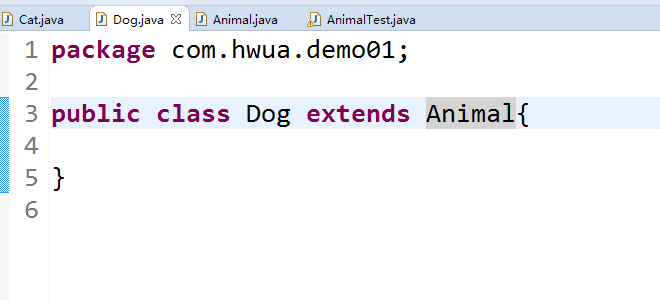
## Java中继承的使用方式:extends关键字

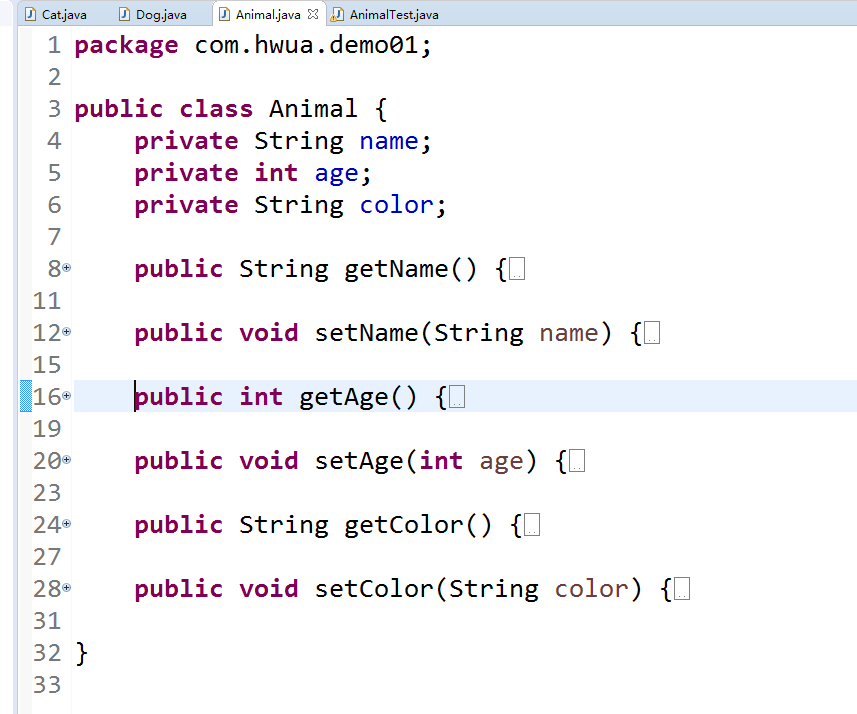
在Java中我们使用extends关键字来代表继承.

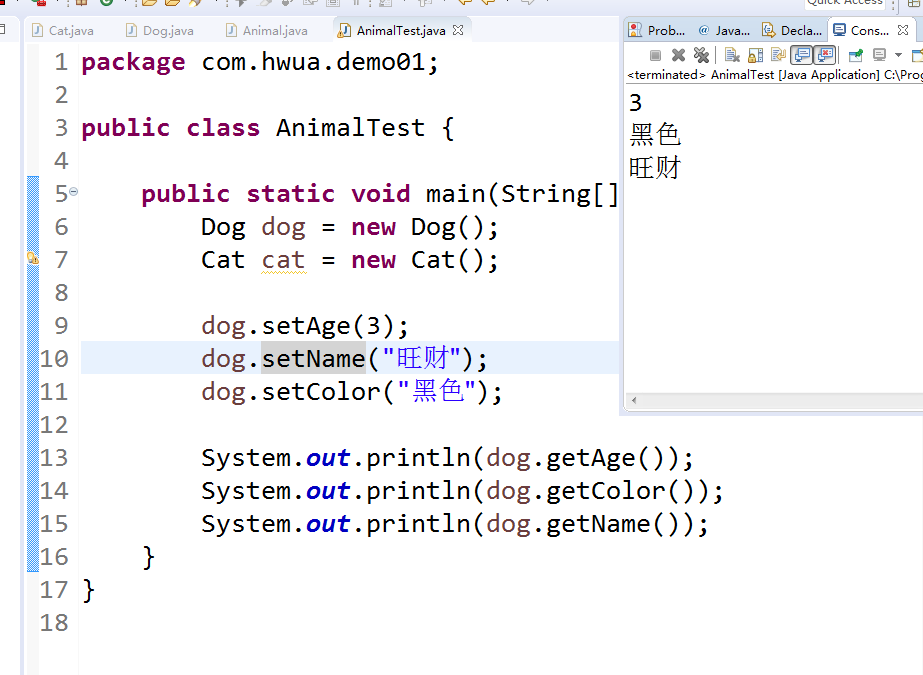


我们通过代码来实现Animal图例的效果









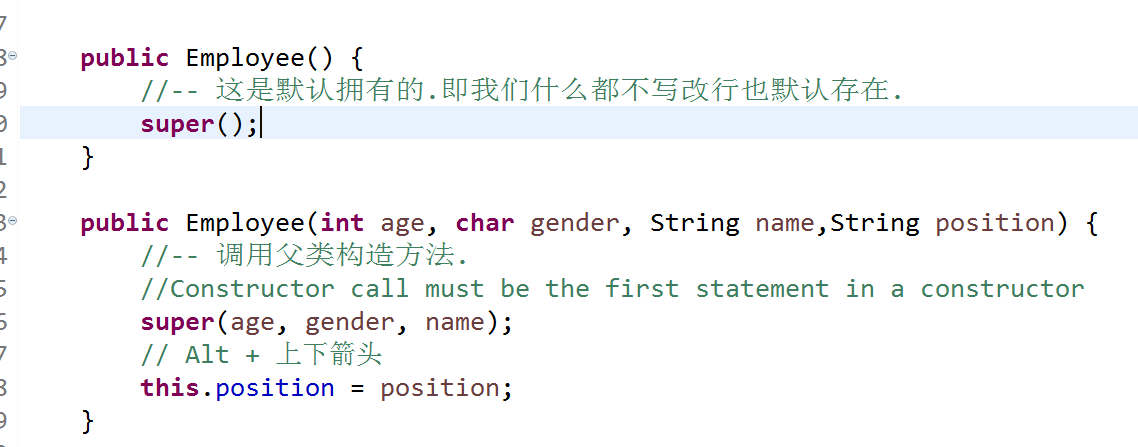
如果我们想要使用继承关系.通过会使用一个表达式来进行判断:X is – a Y.比如猫is -a 动物.即猫是一个动物.如果满足这种关系我们就可以使用继承.简单说就是现实生活中的同类型事物可以用继承.

## Super关键字

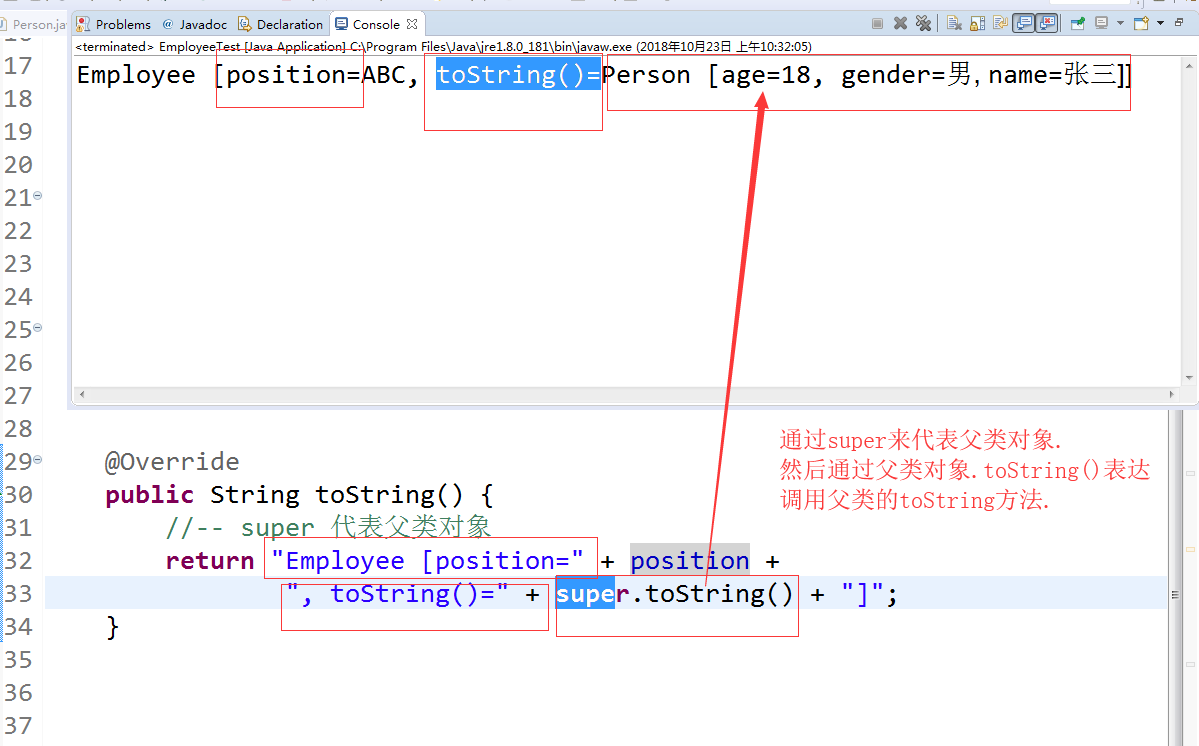
当继承关系成立以后.关键字super就可以被使用了.它具有两重含义.类似于this

* Super()的写法代表父类的构造方法.具体哪一个构造由参数类表决定
* Super. 代表父类对象.

Super()目前只能运用于子类(扩展出来的类)的构造方法中.调用父类的构造方法.具体哪一个构造方法由参数列表决定.并且super()在子类的构造方法中必须位于第一行.



当我们在子类中需要使用父类对象.或者使用父类中已经声明过的方式.我们就可以通过super.来实现.



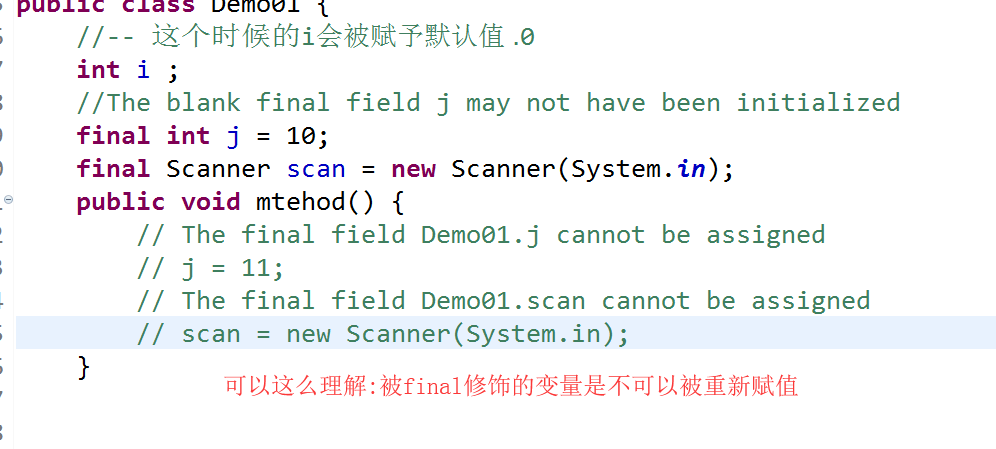
## Super和This的用法对比

Super和This很类似.都有两种写法,对应两种含义.

## Final关键字

Final代表最终的意思,即不可以被修改.final可以用来修饰以下几个内容:

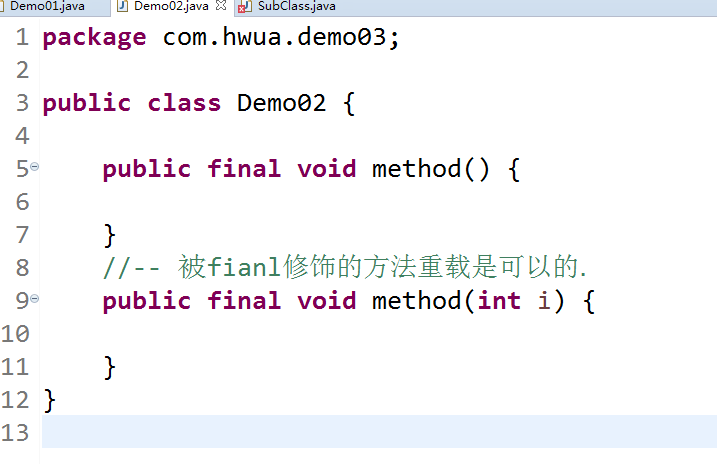
* 变量
  + 原生类型变量
  + 引用类型变量

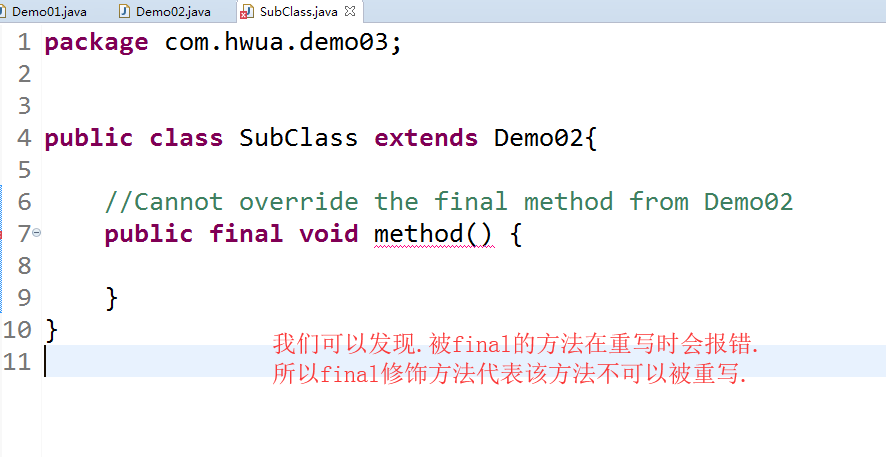


* 方法

在实验之前首先思考,方法有什么是可变的.方法的修改可以说是设计两块

1. 重载
   1. 同类同名不同参.
2. 重写
   1. 继承关系下,子类重写父类方法体.



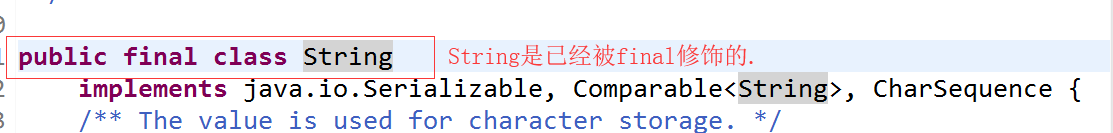


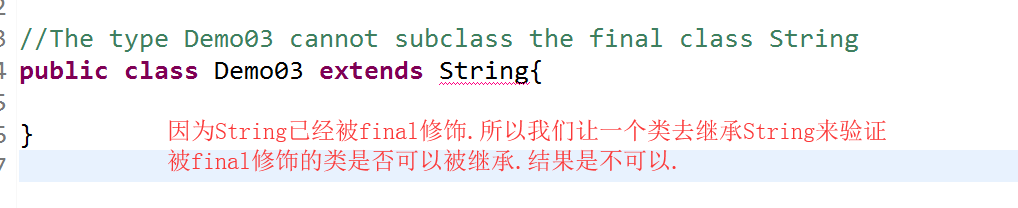
* 类

如上,思考类可以有哪些改变.目前我们学到的对类的操作只有一个就是继承.猜测被final修饰的类不可以被继承.

曾经有一道面试题:String这个类,是否可以被继承.

在Eclipse中使用快接键Ctrl + shift + T 来打开一个类.





关于final还有一道面试题,堪称long long ago.在Java中有三个关键字:final finally(异常哪里讲) 和 finalize

## 方法的重写

在继承关系下.子类基于父类进行功能扩展.比如Animal和Dog. Animal 是父类,Dog是子类.父类中所有非私有的内容子类都直接继承.(关于私有的一部分.继承了但是不可以访问.受访问权限限制).

父类中所有的非私有方法,子类都拥有.我们可以认为子类和父类拥有相同的行为.但是子类可以有自己的实现方式.这种现象我们叫做子类重写父类的方法.保留方法声明不变.更改方法体.

## 父类和子类的构造顺序(涉及Static)

### 重写和重载的区别

## Java的跟类:Object



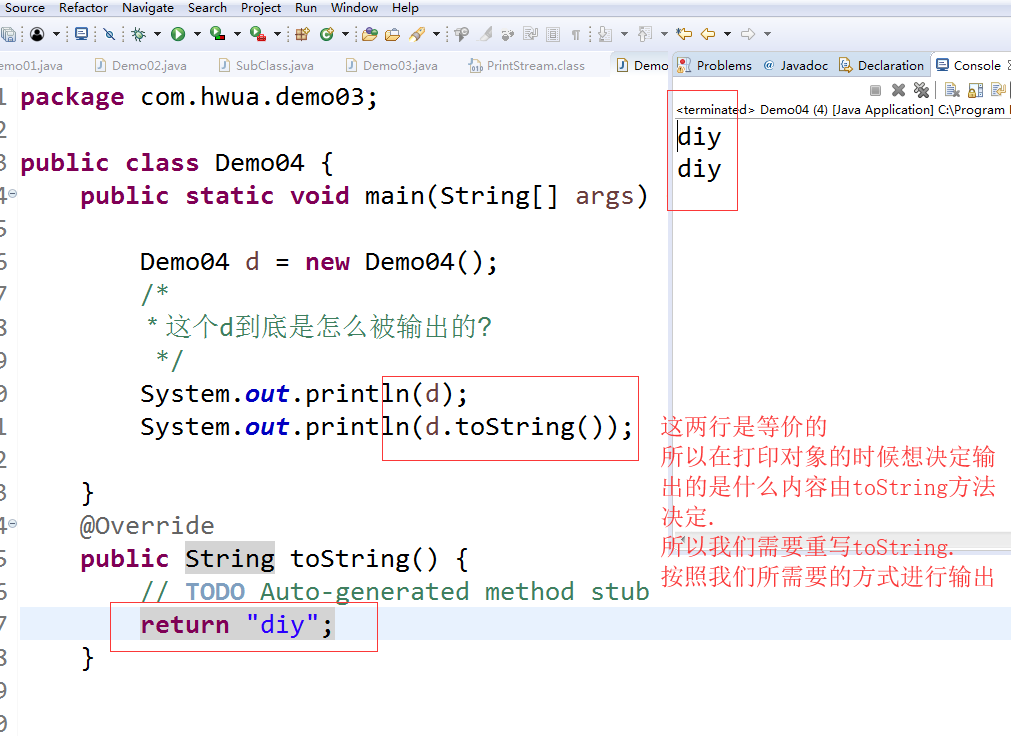
在Java中设计了一个跟类:Object.默认所有类都继承Object.哪怕在类上没有任何显式的extends.所以任何一个类的对象都可以使用Object中的方法.

### Object中常见的几个重写方法

#### toString()方法

用于打印对象的字段和字段值.

为什么需要重写toString方法呢.这个和输出的方式有关.



#### equals()方法

用于引用数据类型间是否相等的比较.本质是利用equals来判断对象的特征是否一样.

一般写equals要遵守以下流程

* 判断对象是否存在
* 判断类型是否匹配
* 转换参数的数据类型
* 判断特征值是否相等.

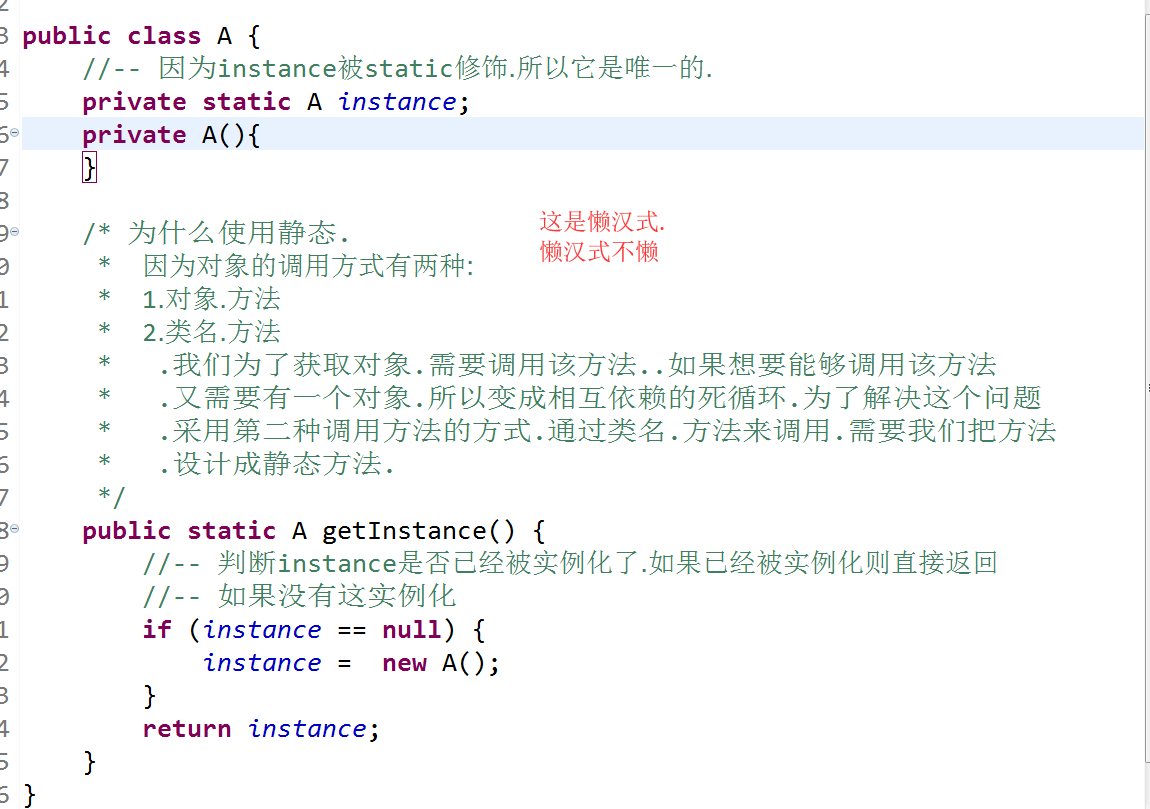
#### 其它方法

* Clone 克隆
* Finalize 通过GC可以回收对象和垃圾回收机制有关.
* hashCode 获取哈希码:31

# 单例设计模式

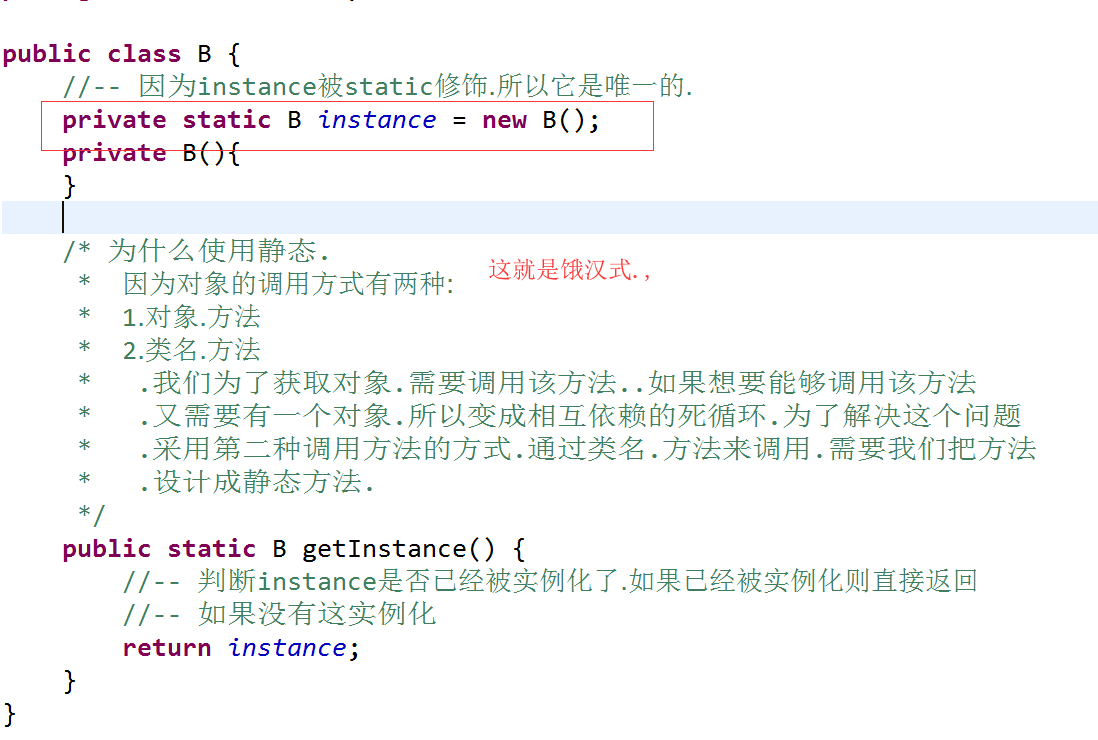
单例就是字面意思:单个实例.只有一个对象.

## 懒汉式



## 饿汉式

饿汉式比懒汉式还要懒,因为它饿了.写法和懒汉式基本一样.

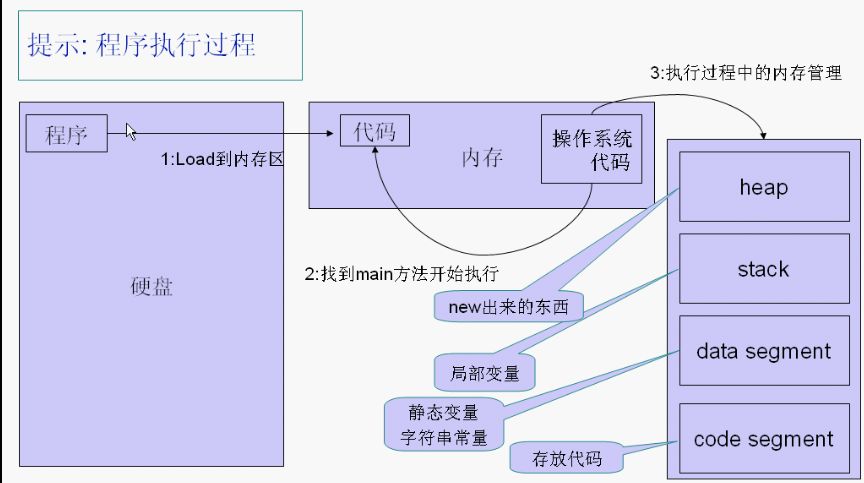


# Java的内存管理

在Java中我们一般认为内存分为4块结构

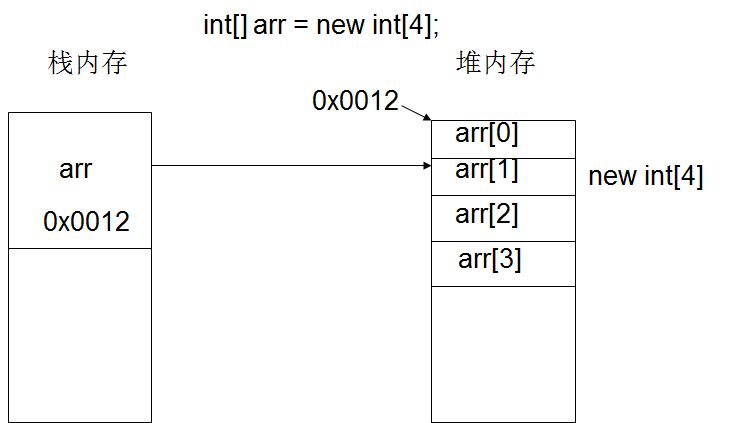
* 堆
* 栈
* 代码块
* 数据块

重点要理解的是堆和栈



我们要了解计算机中可以存储数据的地方有以下几个

* 寄存器
  + CPU中的寄存器.空间最小.速度最快.CPU不受控制.
* 栈
  + 在Java中,所有的原生数据类型和引用变量都位于栈中.栈中的引用变量保存堆中对象的首地址.
  + 栈的空间比寄存器大(大的也有限),但是速度比寄存器慢(仅次于寄存器)
* 堆
  + 空间相比较栈和寄存器是最大的.但是速度是最慢的.
  + 在Java中有new出来的内容都在堆中



# 预习

1. 多态
   1. 向上转型和向下转型
   2. 静态绑定和动态绑定
   3. 关键字:instance of(算是其它类型的运算符.)
2. 设计模式
   1. 简单工厂设计模式
   2. 单例 + 工厂的组合写法.
   3. 工厂来验证多态.
3. 基本的配置文件
   1. Properties配置文件
4. 抽象(abstract)

# 家庭作业

1. 设计动物类,宠物类,猫类,狗类,兔子,利用继承来实现功能的扩展（10.23，18:51）
2. 在动物中设计叫的行为.子类重写该行为提供自己的实现方式. （10.23，18:51）
3. 重写equals和toString.熟练使用两个方法的从重写方式.（10.23，18:51）

# 测试(未讲)

1.在有继承关系,测试子类和父类中构造方法和静态域的执行顺序.