# 一、this

## 1、this关键字

***·是什么***

它在方法内部使用，即这个方法所属对象的引用；

 它在构造器内部使用，表示该构造器正在初始化的对象。

this 可以理解为当前对象或者当前正在创建的对象

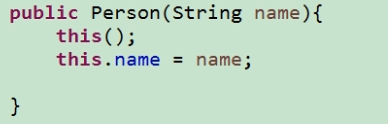
***·原则***

1、在任意方法或构造器内，如果使用当前类的成员变量或成 员方法可以在 其前面添加this， 增强程序的阅读性。不过，**通常我们都习惯省略this**。

2、 当形参与成员变量**同名时**，如果在方法内或构造器内需要 使用成员变 量，**必须添加this来 表明该变量是类的成员变量**

3、使用this访问属性和方法时，如果在本类中未找到，会从**父类**中查找

## 2、this 调用构造器



**this();** 调用空参构造方法

# package

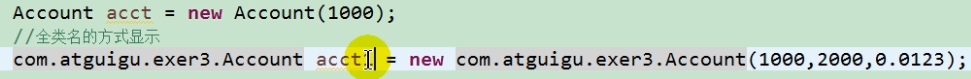
使用**package 声明**类或者接口所在的包，声明在源文件的首行。

# Import

**导入**指定包下的类和接口

***1、*java.lang** 包下的类因为使用频率较高所以不用使用**import** 关键字。

***2、*** 同名的类使用全类名的方式导入



***3***、 import xxx.\* 只能导入 xxx下的所有结构，不能导入 xxx子包下的结构

***4*、** import **static** 导入指定结构下的静态方法或属性，落脚点只能是类中的属性或方法



# super

可以理解为父类的，可以调用属性、方法、构造器，类似this表示当前对象



1、尤其当子父类出现同名成员时，可以用**super**表明调用的是父类中的成员

2、**super**的追溯不仅限于直接父类

3、 **super**和**this**的用法相像，**this**代表本类对象的引用，**super**代表父类的内存 空间的标识

# Instanceof

instanceof关键字的使用  
  
 **a instanceof A:判断对象a是否是类A的实例。如果是，返回true；如果不是，返回false。**  
   
   
 用情境：为了避免在 向下转型时出现ClassCastException的异常，我们在向下转型之前，先  
 进行instanceof的判断，一旦返回true，就进行向下转型。如果返回false，不进行向下转型。  
   
**·如果 a instanceof A返回true,则 a instanceof B也返回true.  
 其中，类B是类A的父类。**

# static

## 1、为什么要有static

我们有时候希望无论是否产生了对象或无论产生了多少对象的情况下，某些特定的数据在**内存空间里只有一份**。

## 2、被static修饰后的成员具备的特点：

1、随着**类的加载**而加载

2、优先于**对象**存在

3、修饰的成员，被所有对象所**共享**

4、访问权限允许时，可不创建对象，直接被类调用

## 3、类属性、类方法的设计思想

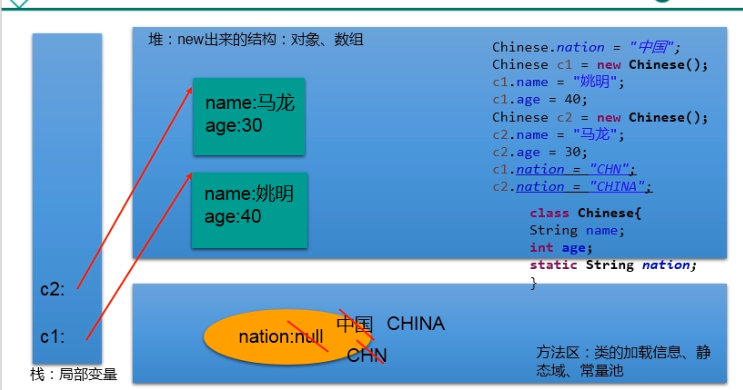
 属性：类属性作为该类各个对象之间共享的变量。在设计类时,分析哪

些属性**不因对象的不同而改变**，将这些属性设置为类属性。相应 的方法设置为类方法。

 方法：如果方法与调用者无关，则这样的方法通常被声明为类方法，由

于不需要创建对象就可以调用类方法，从而**简化了方法的调用**

## 静态变量的内存解析



## 什么时候使用static 关键字

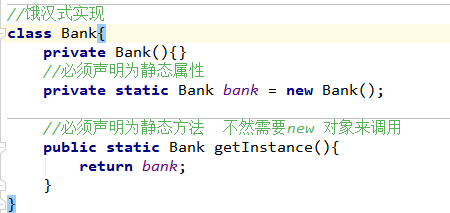
static注意点：  
 5.1 在静态的方法内，**不能使用this**关键字、super关键字  
 5.2 关于静态属性和静态方法的使用，大家都从生命周期的角度去理解。

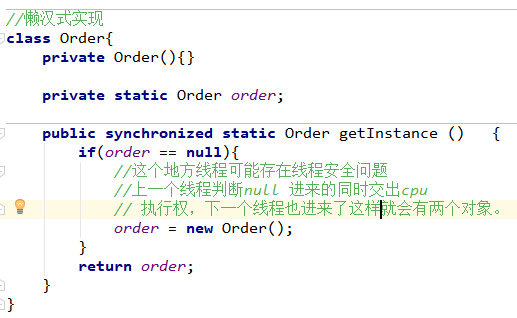
开发中，如何确定一个**属性**是否要声明为static的？  
 > 属性是可以被多个对象所**共享**的，不会随着对象的不同而不同的。  
 > 类中的**常量**也常常声明为static  
   
 开发中，如何确定一个**方法**是否要声明为static的？  
 > 操作静态属性的方法，通常设置为static的  
 > **工具类**中的方法，习惯上声明为static的。 比如：Math、Arrays、Collections

## 单例模式

所谓类的单例设计模式，就是采取一定的方法保证在整个的软件系统中，对 某个类只能存在**一个对象实例**，并且该类只提供一个取得其对象实例的方法。 如果我们要让类在一个虚拟机中只能产生一个对象，我们首先必须将类的构造器的访问权限设置为private，这样，就不能用new操作符在类的外部产生类的对象了，但在类内部仍可以产生该类的对象。**因为在类的外部开始还无法得到类的对象，只能调用该类的某个静态方法以返回类内部创建的对象**，静态方法只能访问类中的静态成员变量，所以，指向类内部产生的该类对象

的变量也必须定义成静态的。





## 单例的使用场景



# 七、final关键字

final:最终的  
   
 1. final可以用来修饰的结构：类、方法、变量  
   
 2. final 用来**修饰一个类**:此类不能被其他类所继承。  
 比如：String类、System类、StringBuffer类  
   
 3. final 用来**修饰方法**：表明此方法不可以被重写  
 比如：Object类中getClass();  
   
 4. final 用来**修饰变量**：此时的"变量"就称为是一个常量  
 4.1 final修饰属性：可以考虑赋值的位置有：显式初始化、代码块中初始化、构造器中初始化  
 4.2 final修饰局部变量：  
 尤其是使用final修饰形参时，表明此形参是一个常量。当我们调用此方法时，给常量形参赋一个实参。一旦赋值以后，就只能在方法体内使用此形参，但不能进行重新赋值。  
**！！注意：不能重新赋值，但是可以改变对象的属性，不能改变变量指向的对象地址。**  
 static final 用来修饰属性：全局常量