# Static静态

static本身是一个修饰符，可以修饰变量、方法、内部类、代码块。静态是从类的层面来看的，已经超越了对象。

## 1.1静态变量

### 定义

static修饰变量那么这个变量我们就称之为静态变量，又称为成员变量或者类变量。

### 特点

静态变量随着类的加载而加载到了方法区中的静态区，并且在静态区中自动赋予了一个默认值。静态变量先于对象而存在，所以静态变量可以通过类名来调用，也可以通过对象来调用。该类产生的所有的对象实际上存的是该静态变量在静态区中的地址，静态变量是被所有对象所共享的。

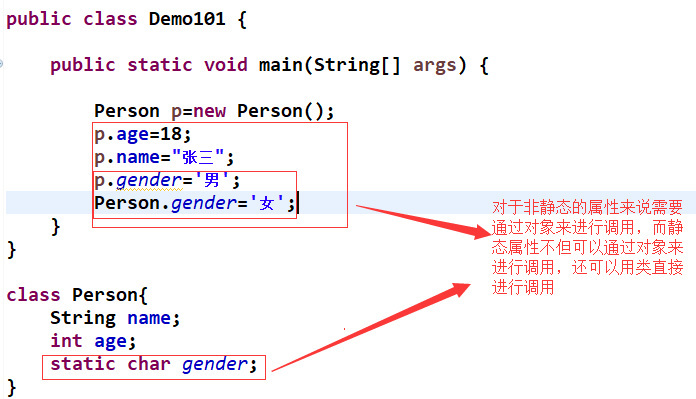
### 优点

提高了复用性、安全性、使代码结构更加紧密。

1. 使属性私有化---隐藏信息，实现细节。
2. 使属性值更符合要求---可以对成员进行更精确的控制。
3. 提高了代码的安全性---类内部的结构可以自由修改
4. 良好的封装能够减少耦合。

注意：实际开发过程里，static能少用就少用

### 静态内存流程

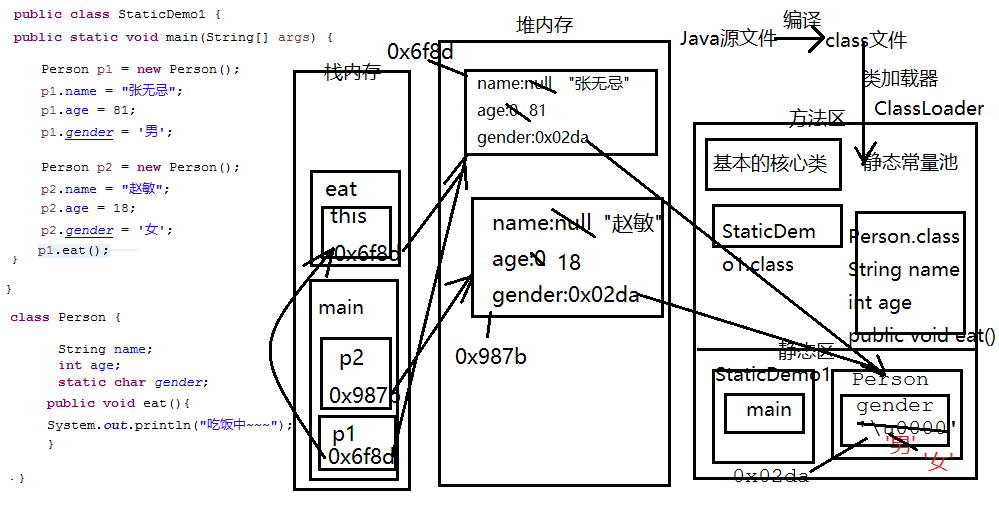


详解：如下图所示：

在这里会因出方法区的概念：

存储类信息。存储类的基本信息的区域称之为静态常量池。存储静态属性和静态方法的地方称之为静态区。

一个类在第一次真正使用的时候才会在方法区中加载——所以类只加载一次，加载完成之后就不再从方法区中移除。（搭配下图使用）



注意：在方法区java源文件—编译--.class文件通过类加载器加载到方法区---会先把基本的核心类加载到方法区

详解：

是对.class文件进行操作，首先根据.class文件利用类加载器在方法区中加载核心类库，然后在寻找哪个类中拥有main函数，发现StaticDemo1中拥有main函数，将该类加载到方法区中，并将main函数加载到方法区中的静态区，然后在执行函数。执行时发现创建了类Person的对象p1，并进行初始化，此时，先将该类在方法区中开辟空间，并将除静态属性外的所有属性放到方法去中，然后再去栈中为p1开辟空间，在堆中为该对象开辟空间，并在空间中为所有的属性（包括静态属性）赋予初值，再次注意，静态方法的初值为在静态区中的空间的地址。在后面为该对象的各属性赋值覆盖原来的值。

## 静态方法

### 定义

static修饰方法就叫静态方法，也叫类方法。

例如：

Arrays.sort()

Arrays.toString()

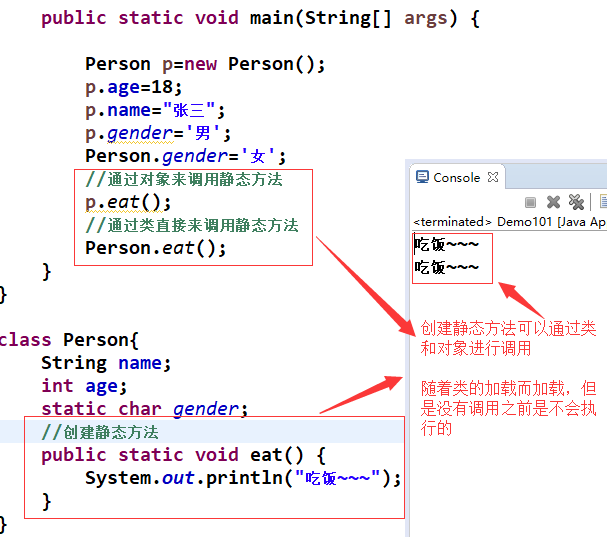
Arrays.copyOf()

System.arraycopy()都是静态方法。

### 特点

在类加载的时候加载到了方法区中的静态区，只是存储在静态区，在方法被调用的时候到栈内存中执行。静态区中的元素不归属于某一个对象而是归属于类。静态方法先于对象而存在的，所以静态方法可以通过类名来调用，也可以通过对象来调用。

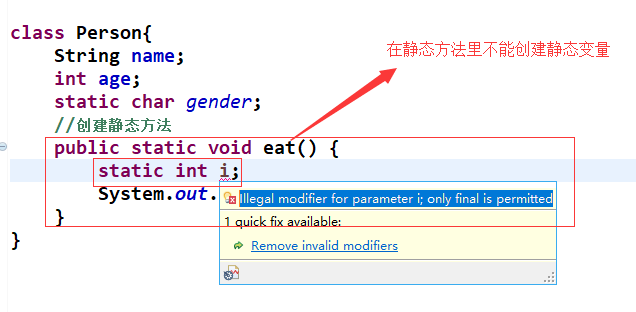
注意：类只加载一次，是在第一次使用的时候才加载到方法区，而且加载到方法区中之后就不再移除了



### 注意事项

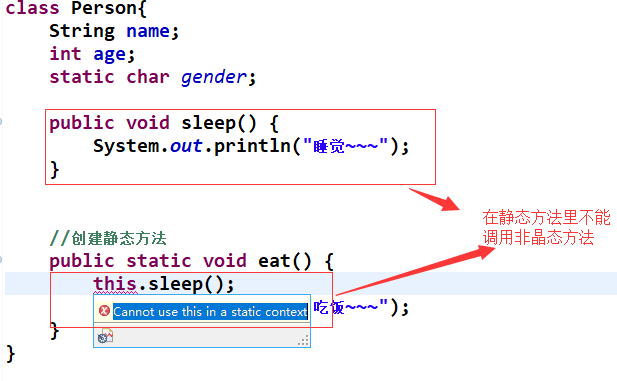
1. 静态变量可以定义到静态方法中吗？——不可以

静态方法在静态区中只存储不执行，当被调用的时候才在栈中执行。而静态变量是根据类的加载而加载，上来就要进行初始化



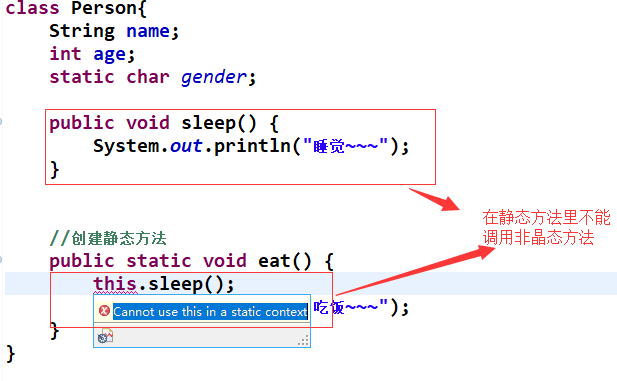
1. 静态方法中可以直接调用本类中的非静态方法吗?——不可以。

在Java中所有非静态方法和非静态属性都是通过对象来调用的。



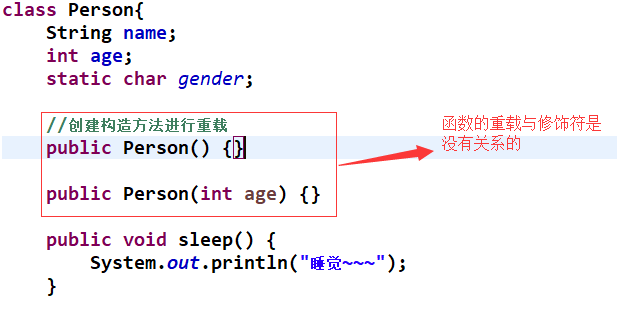
1. 在主函数中可以使用this和super吗？——不可以

主函数就是静态的，使用上图道理相同，super与this都是和对象是相关联的而静态先于对象而存在



1. 静态方法可以重载吗？——可以的。

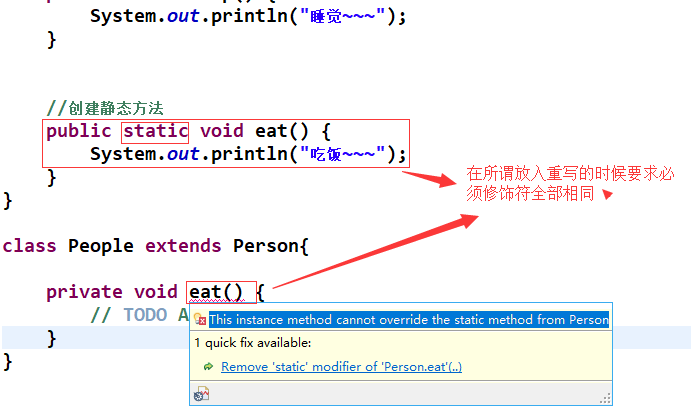
重载对修饰符没有要求的，对返回值也没有要求。



1. 静态方法可以重写（方法的覆盖）吗？——不可以。
2. 静态方法可以存在方法签名完全一致的静态方法吗？——可以

但这不是重写。但是也适用于重写的那套规则。

注意：相同的方法要么都是普通的方法，要么都是静态方法



## 静态代码块

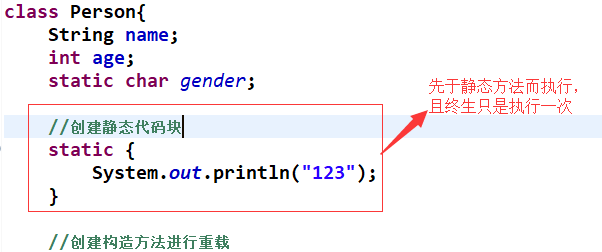
### 定义

在类中成员的位置用static修饰用{}括起来的代码块。

静态代码块针对的是类，所以也可以叫做类代码块。

实际上静态代码块是随着类的加载而加载到静态区，在类创建对象或者执行方法之前执行一次，终其一生只执行一次。

#### 格式



### 执行顺序

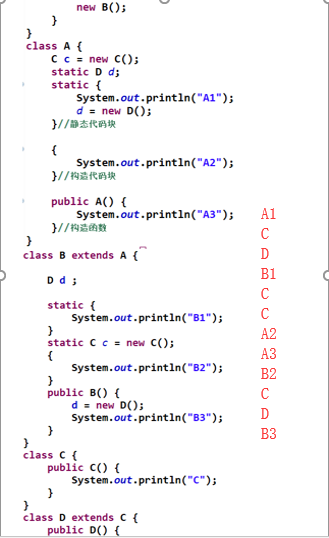
在这个类第一次被真正使用（第一次创建对象/调用方法）的时候执行一次。如果一个类包含多个静态代码块，则按照书写顺序执行。由于类只在第一次使用的时候加载，所以静态代码块也只执行一次。

代码执行顺序：先父类后子类，先静态后动态。（先父子类的静态，后父子类的动态）静态优先，父类优先。

代码是从上到下，从左到右依次编译执行。

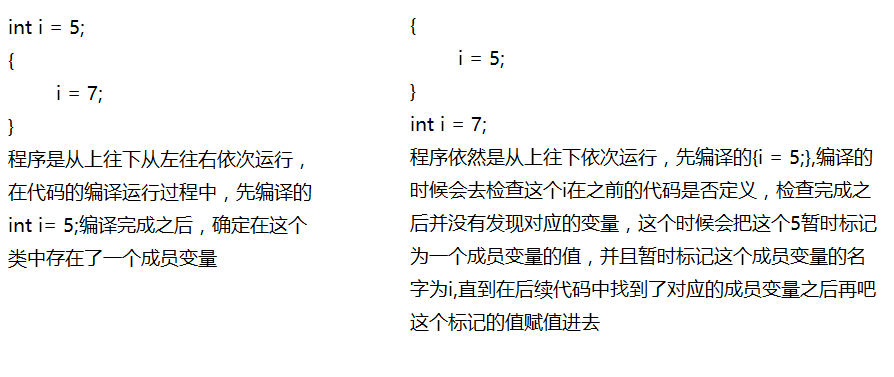
创建子类对象的时候需要先创建父类对象→加载父类→执行父类静态代码块→执行子类代码块→父类构造代码块→父类构造函数→子类构造代码块→子类构造函数

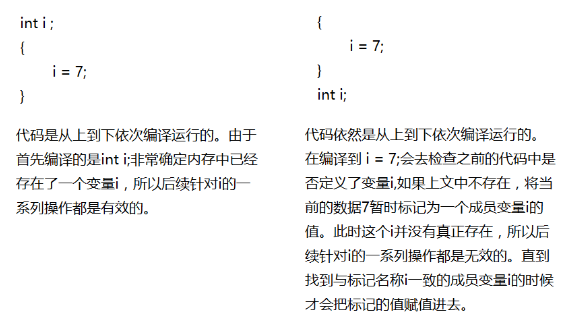
举例说明：



### 扩展

对于成员变量和构造代码块之间的执行顺序，如下图所示：





# 最终final

Final可以修饰数据、方法和类。

## 常量

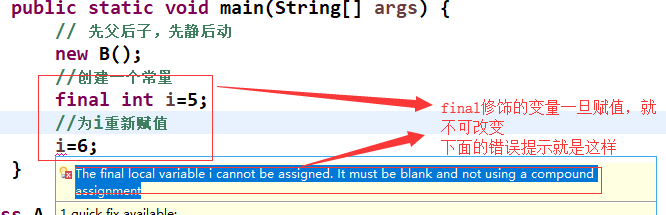
### 定义

当final修饰数据（基本类型和引用类型）的时候，表示这个变量的值不可变，称之为常量。终其一生只能赋值一次。

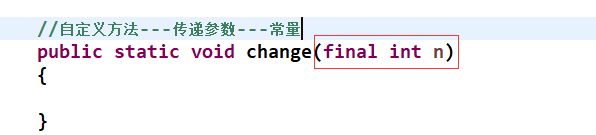
在Java中所说的常量往往是指静态常量。因为实质上只有静态常量才是独有的一个。

### 特点

1.常量在定义好之后不可改变，final固定的是栈内存中的数值。

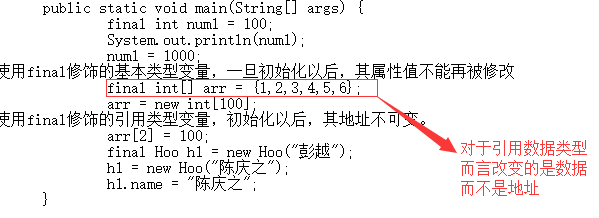


2.常量可以作为参数传递。

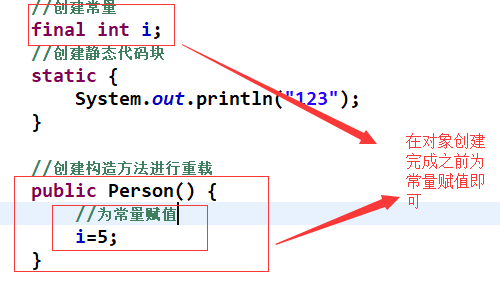


3.对引用类型而言，final固定的是其在栈中的地址不可变。

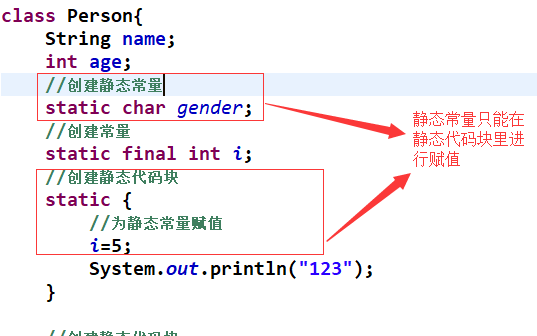
数组在栈内存中存储的是地址，用final修饰，是不能改变数组的地址，但数组的值可以改变。对于对象而言，对象的引用不能改变，但是引用的属性值是可以进行改变的。



4.成员常量只要是在对象创建完成之前（构造方法/函数执行结束之前）赋初始值即可。



5.静态成员常量（static final）只要在类加载完成之前给值即可,而且只能在静态代码块中赋值。



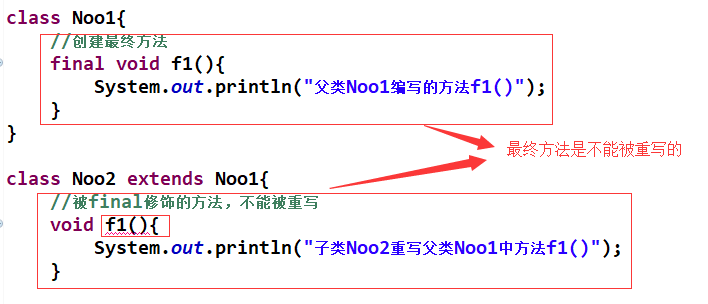
## 最终方法

### 定义

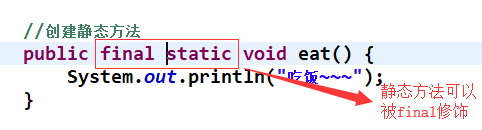
final修饰方法的时候，这个方法就是最终方法。

### 特点

1. 最终方法不可以被重写，可以重载，可以被继承。



1. 静态方法可以被final修饰。



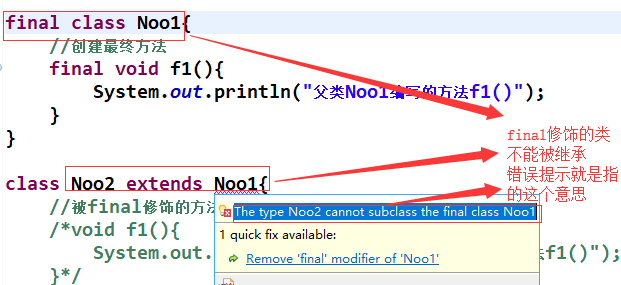
## 最终类

### 定义

final修饰类那么这个类就是最终类。

### 特点

1. 最终类不可以被继承，也不能有匿名内部类形式。（匿名内部类后续详细说）



1. 现阶段最终类的方法不能被重写。

由于最终类不能被继承，因此重写也是不可以的，同过上图就可看出