# abstract抽象

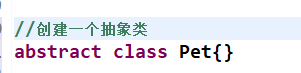
**抽象abstract可以修饰类和方法**

## 1.1 抽象类

### 定义

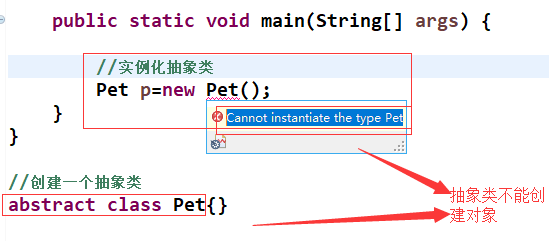
将一些名称一致但是细节不同的行为提取到父类中定义为抽象方法，抽象方法所在的类就是抽象类，用abstract来修饰的类。抽象类中，不一定含有抽象方法，但是抽象方法所在的类一定是抽象类。

### 格式

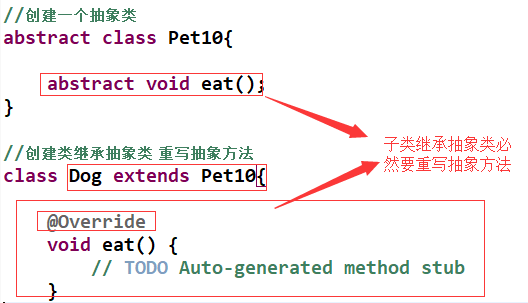


### 特点

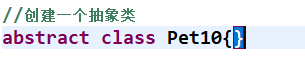
1. 抽象类不可以在Java中创建对象/实例化。即使没有抽象方法也无法创建对象，可以创建匿名内部类。（匿名内部类后续详解）



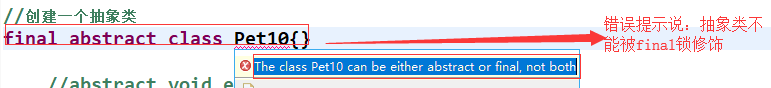
1. 抽象类被子类继承之后，必须重写其中的抽象方法。



1. 抽象类中可以没有抽象方法。抽象类中可以定义一切的属性和方法。



1. 抽象类不能用final修饰。最终类不可以是抽象类。



### 抽象类和普通类的区别

抽象类和普通类的区别就是可以有抽象方法，不可以创建对象。

注意：抽象类之所以不能创建对象是因为在进行编译的时候JVM虚拟机检测到abstract就会将构造方法自动舍弃

## 抽象方法

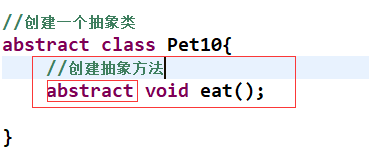
### 定义

如果所有的子类中存在了一些名称一致而细节不同的方法的时候，这个时候可以在父类中声明该行为，此时声明行为的时候不需要添加方法体，所以此时该方法就形成了抽象方法，使用abstract修饰。

简单来讲：就是在父子类进行继承的时候，子类重写父类的方法但是父类的方法在后续并不打算使用，因此就会将父类的方法的方法体删除，子类直接重写即可。

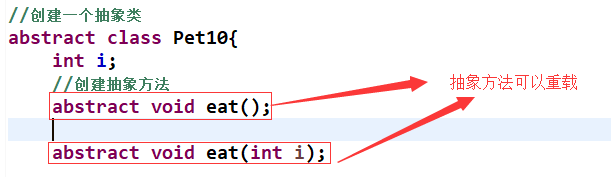
### 格式

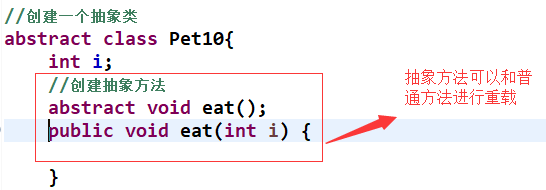
如下图所示：



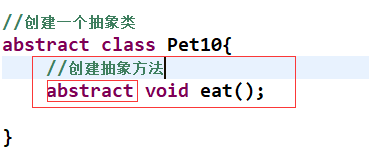
### 特点

1. 抽象方法可以和抽象方法重载，也可以和实体方法重载。





1. 抽象方法没有方法体



### 权限修饰符

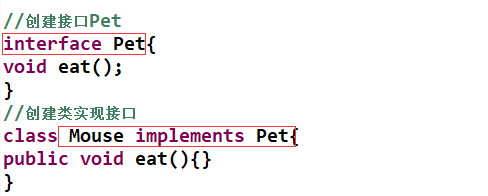
1. 抽象方法不可以被static、final、private修饰，因为final和private修饰符修饰的方法都不可以被重写；static修饰的方法，先于对象存在，没有具体对象没办法加载。
2. 抽象方法可以使用默认权限修饰，要求子类必须和父类同包。
3. 抽象方法可以被protected权限修饰，要求要么同包要么是子类。

# 接口interface

## 定义

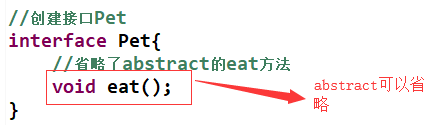
用interface来声明，所有方法都为抽象方法那么我们就称之为接口。通过implements关键字让接口和类产生联系，这个过程叫实现。在接口中使用向上造型来创建对象，就是接口的多态。

## 格式

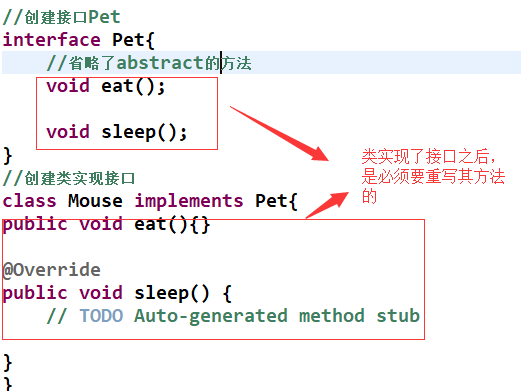


## 特点

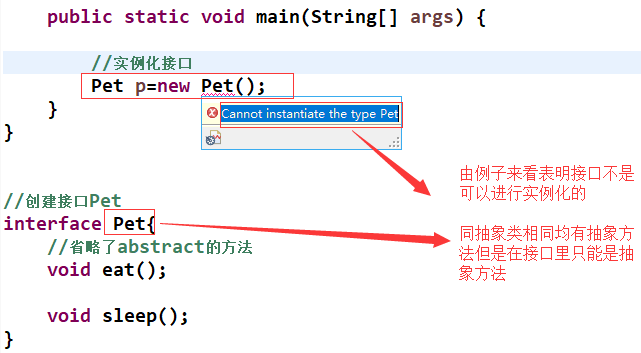
1. 接口中方法的abstract关键字可以省略。



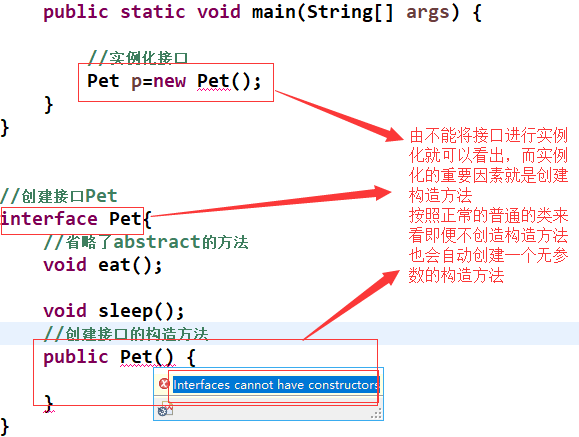
1. 类用implements来实现一个接口。必须实现这个接口中所有的方法。



1. 由于接口中都是抽象方法，接口不能实例化。



1. 接口中没有构造函数。虽然接口在编译完成之后会产生class文件，但是接口不是类。



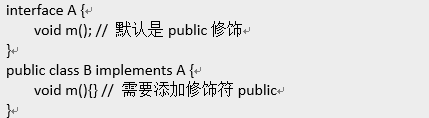
1. 接口中可以定义属性，这个属性默认是一个静态常量。



## 权限修饰符

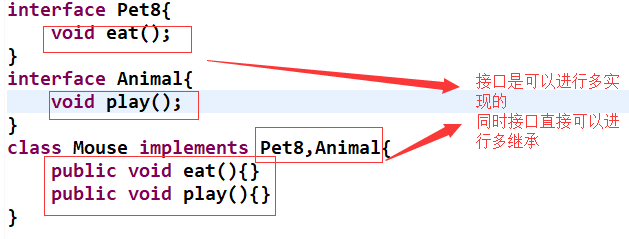
1.接口中的属性默认是用public static final 来修饰，修饰符的位置可以互换。

2.接口中的方法默认用public abstract修饰，而且只能是public修饰的，public可以省略不写。在接口的子类中实现接口的方法记得用public修饰。



## 继承与实现

Java中类支持单继承，多实现。一个类只能继承一个类，但是一个类可以实现多个接口。一旦出现了多实现，那就必不可免的会导致方法调用混乱。



注意：Java中接口之间是多继承，并且接口和类之间是多实现的关系，所以就形成了一张继承关系网，由于在网状结构中寻找一个根节点比较困难，为了提高效率，Java在编译的时候放弃检查接口和类之间是否有实现关系。

当类进行强制转换的时候，JVM在编译的时候会对两个类进行检查，检查这两个类之间是否有继承关系。如果有继承关系，则编译的时候会通过，但是运行的时候不一定正确。如果没有继承关系，则在编译的时候直接报错。

## 作用

统一了结构。接口可以作为模板，配合多态实现解耦。

# 内部类

## 定义

定义在类或者接口中的类就称之为内部类。内部类是封装的第三种形式。

## 分类

内部类根据使用的位置和修饰符不同分为：方法内部类、成员内部类、静态内部类和匿名内部类

## 特点

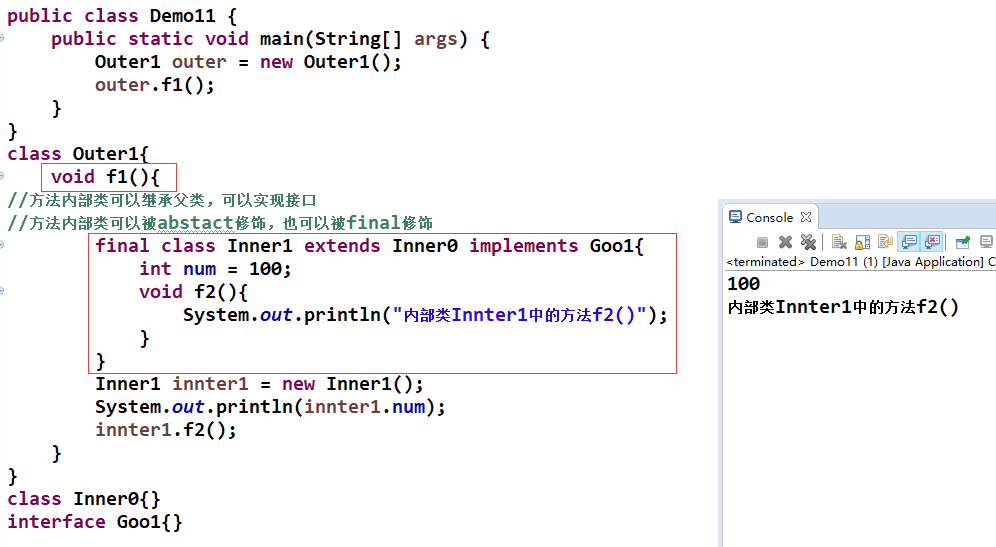
除了静态内部类，其余的内部类中都不允许定义静态属性和静态方法，但是可以定义静态常量。

除了静态内部类，其余的内部类都可以使用当前外部类的属性和方法，但是静态内部类只能使用外部类的静态成员。

## 方法内部类

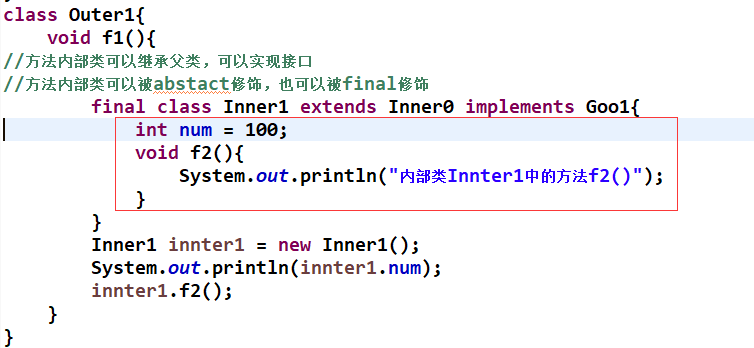
### 3.4.1 定义

定义在方法里的类叫做方法内部类



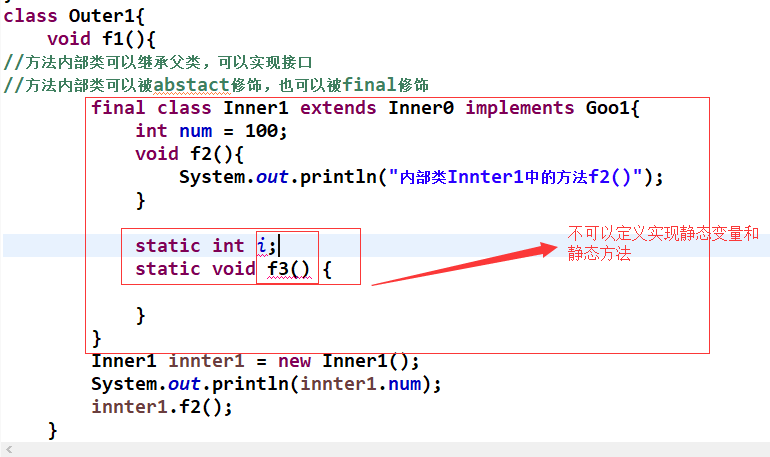
### 3.4.2 特点

1. 可以定义成员属性和成员方法

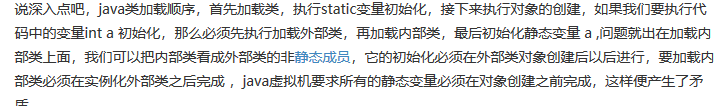


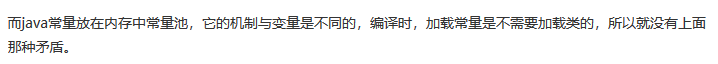
1. 不可以定义静态属性和静态方法

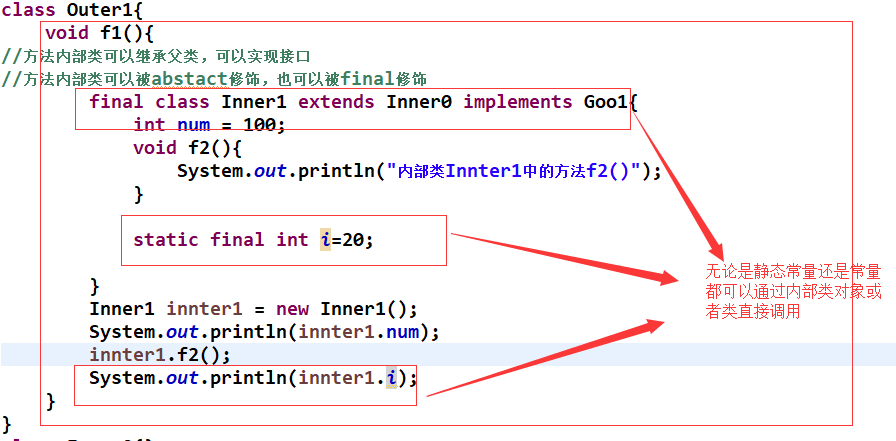
注意：[静态变量](https://www.baidu.com/s?wd=%E9%9D%99%E6%80%81%E5%8F%98%E9%87%8F&from=1012015a&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YdPyRYrjDLPHmYujDzmW640ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHm4nH0srj6dPjDzPW0LnHRYr0)是要占用内存的，在编译时只要是定义为[静态变量](https://www.baidu.com/s?wd=%E9%9D%99%E6%80%81%E5%8F%98%E9%87%8F&from=1012015a&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YdPyRYrjDLPHmYujDzmW640ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHm4nH0srj6dPjDzPW0LnHRYr0)了，系统就会自动[分配内存](https://www.baidu.com/s?wd=%E5%88%86%E9%85%8D%E5%86%85%E5%AD%98&from=1012015a&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YdPyRYrjDLPHmYujDzmW640ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHm4nH0srj6dPjDzPW0LnHRYr0)给他，而[内部类](https://www.baidu.com/s?wd=%E5%86%85%E9%83%A8%E7%B1%BB&from=1012015a&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YdPyRYrjDLPHmYujDzmW640ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHm4nH0srj6dPjDzPW0LnHRYr0)是在宿主类编译完编译的，也就是说，必须有宿主类存在后才能有[内部类](https://www.baidu.com/s?wd=%E5%86%85%E9%83%A8%E7%B1%BB&from=1012015a&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YdPyRYrjDLPHmYujDzmW640ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHm4nH0srj6dPjDzPW0LnHRYr0)，这也就和编译时就为[静态变量](https://www.baidu.com/s?wd=%E9%9D%99%E6%80%81%E5%8F%98%E9%87%8F&from=1012015a&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YdPyRYrjDLPHmYujDzmW640ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHm4nH0srj6dPjDzPW0LnHRYr0)[分配内存](https://www.baidu.com/s?wd=%E5%88%86%E9%85%8D%E5%86%85%E5%AD%98&from=1012015a&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YdPyRYrjDLPHmYujDzmW640ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHm4nH0srj6dPjDzPW0LnHRYr0)产生了冲突，因为系统执行：运行宿主类->静态变量[内存分配](https://www.baidu.com/s?wd=%E5%86%85%E5%AD%98%E5%88%86%E9%85%8D&from=1012015a&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YdPyRYrjDLPHmYujDzmW640ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHm4nH0srj6dPjDzPW0LnHRYr0)->[内部类](https://www.baidu.com/s?wd=%E5%86%85%E9%83%A8%E7%B1%BB&from=1012015a&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YdPyRYrjDLPHmYujDzmW640ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHm4nH0srj6dPjDzPW0LnHRYr0)，而此时内部类的静态变量先于内部类生成，这显然是不可能的，所以不能定义静态变量！



1. 可以定义静态常量和或是直接常量

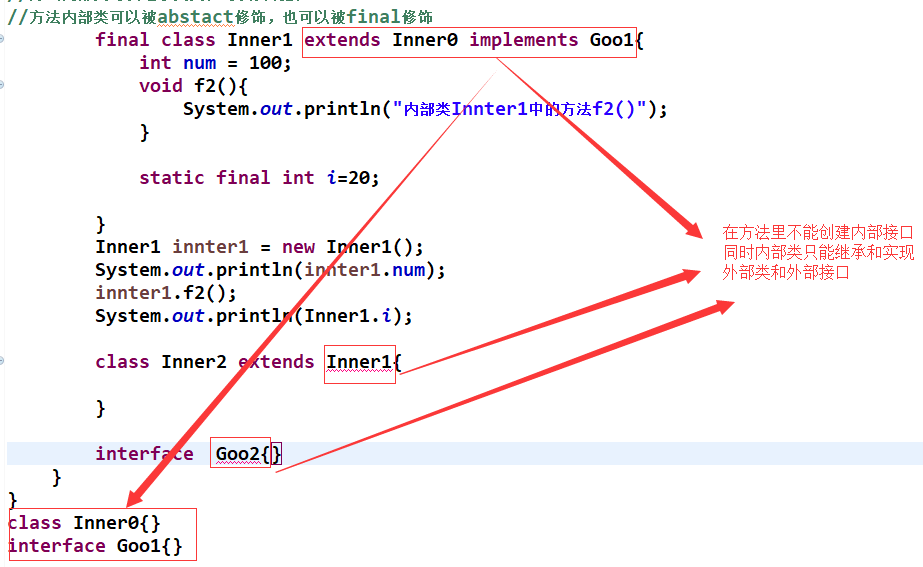






1. 可以继承其他类和实现接口

注意：在这个继承和实现只是只能继承和实现外部类的接口和类，内部类和内部接口是不能被继承和实现的



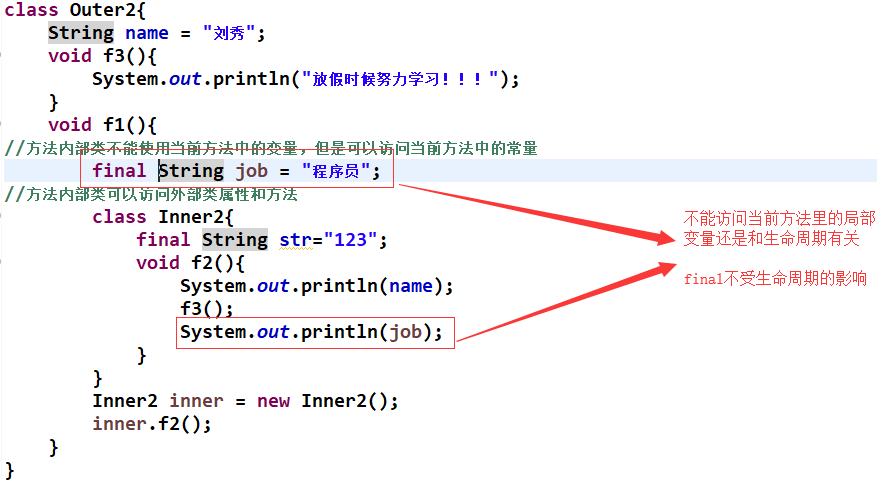
1. 可以使用外部类的属性和方法



1. 不可以使用当前方法里的局部变量

注意：内部类的生命周期和方法中的局部变量是不一样的，内部类是也是一个类，是存储在堆中，也只有当对该类的引用消失时，内部类才会消亡。而方法的局部变量是存储在堆栈中的，当调用结束时就会退栈，即在内存中这个属性就消失了。也就是说，内部类的生命周期超过了方法中局部变量的生命周期，内部类可能会调用到已经消失的属性，因此内部类不能访问方法中的局部变量。  
解决方法就是在局部变量前加修饰符final  
，此时局部变量就会存在常量池中，并在栈内存开辟空间指向方法区，生命周期跟工程的生命周期是一样的，此时内部类就可以访问方法中的局部变量。

1. 可以使用当前方法里的常量



### 3.4.3作用

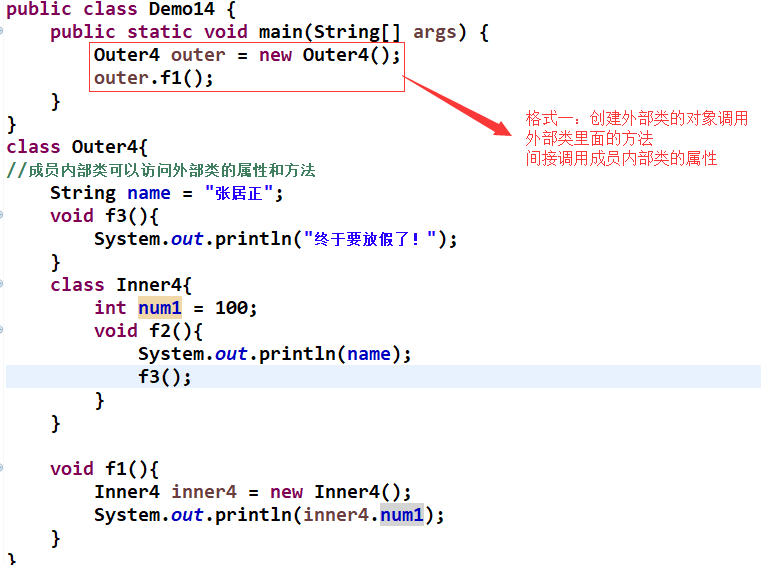
方法内部类是为了私有本类方法中的参数。内部类可以继承可以实现。

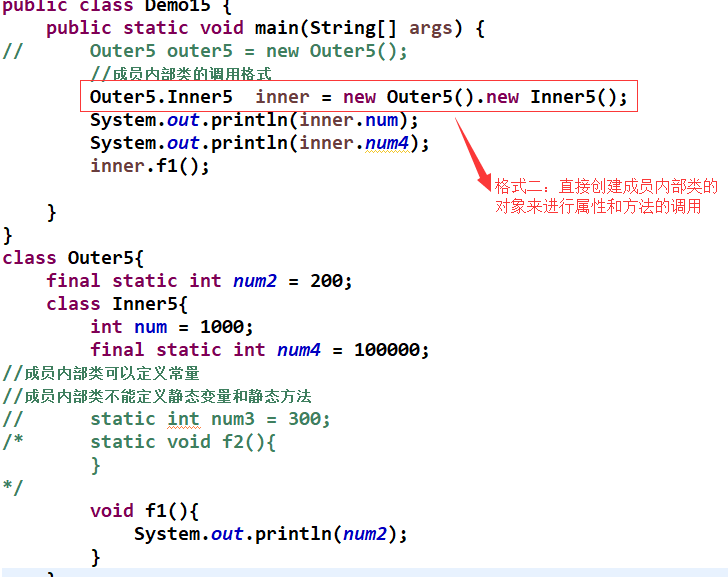
## 成员内部类

### 3.5.1 定义

定义在类内方法外的类叫做成员内部类，也就是成员变量的位置，利用外部类对象来创建成员内部类对象。

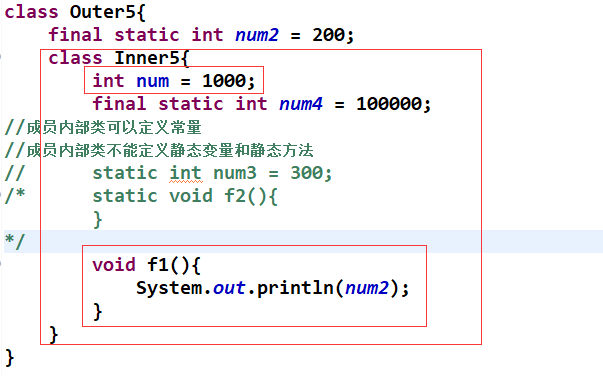
### 3.5.2调用格式



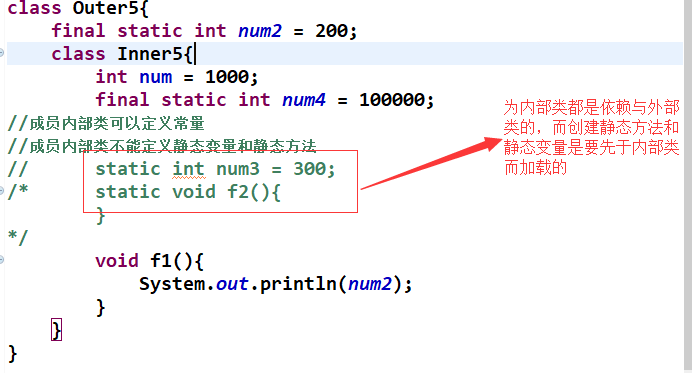


### 3.5.3 特点

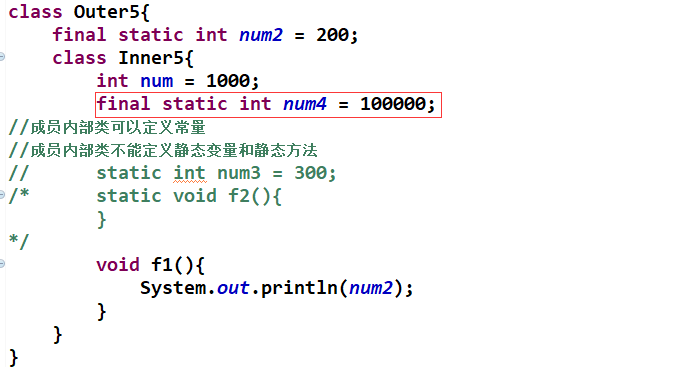
1.成员内部类可以定义成员变量和成员方法



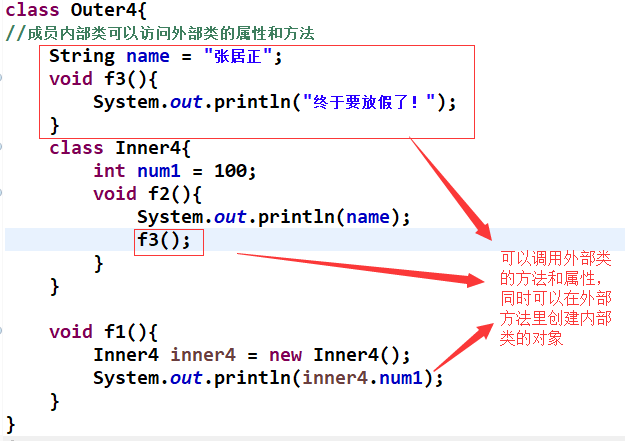
2.不可以定义静态变量和静态方法



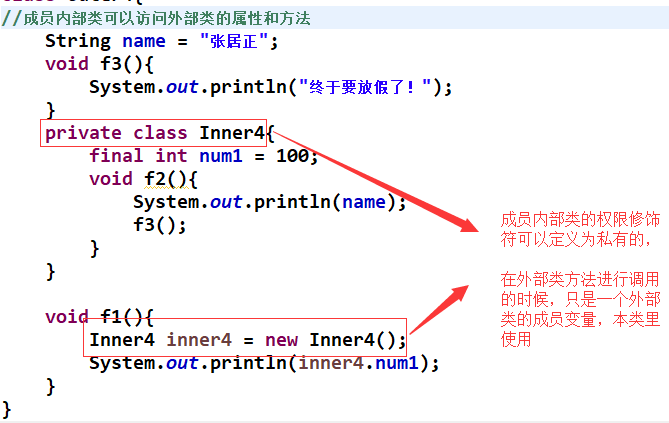
3.可以定义静态常量



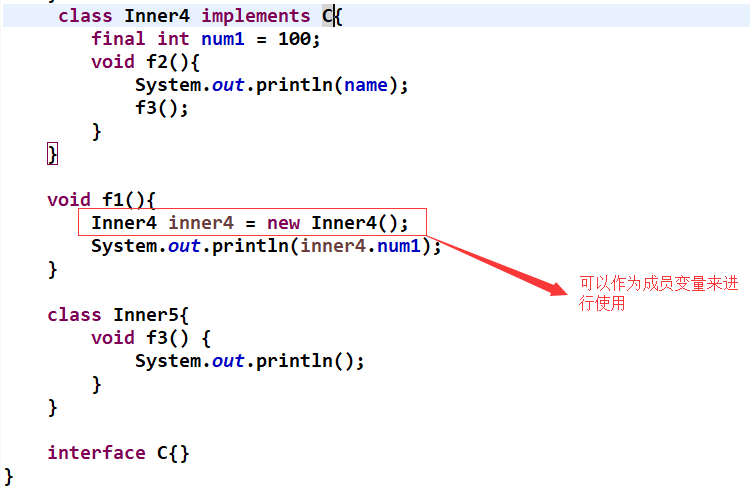
4.可以使用外部类中的成员属性和静态属性，以及方法



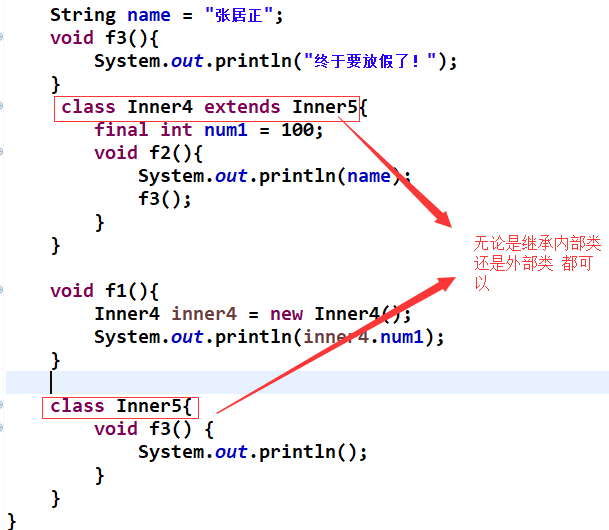
5.内部类的权限可以定义为私有的



6.作为外部类中的成员变量来使用



7.可以继承或者实现

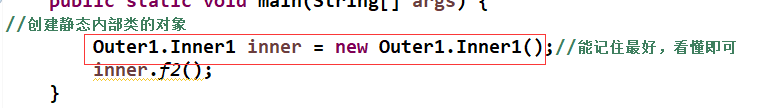


## 静态内部类

### 3.6.1 定义

用static修饰的成员内部类叫做静态内部类。可以直接利用外部类来创建静态内部类的对象。

### 3.6.2 调用格式

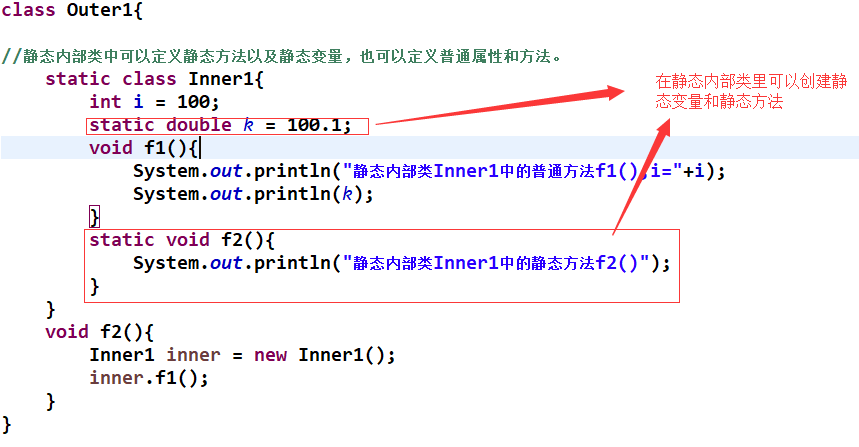


### 3.6.3 特点

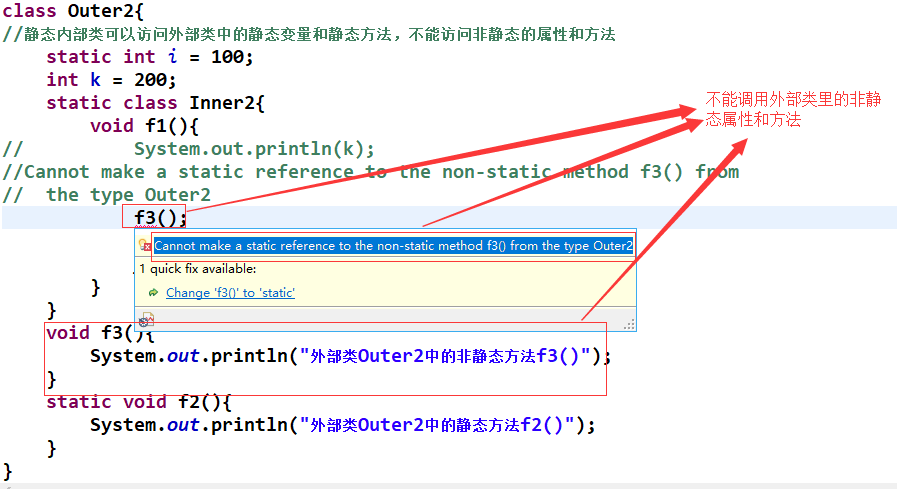
1.可以定义成员属性和成员方法



2.可以定义静态属性和静态方法



3.不允许使用外部类里面的非静态属性和静态方法



## 匿名内部类

### 3.7.1 定义

没有名字的内部类叫做匿名内部类。包含成员匿名内部类，方法匿名内部类。

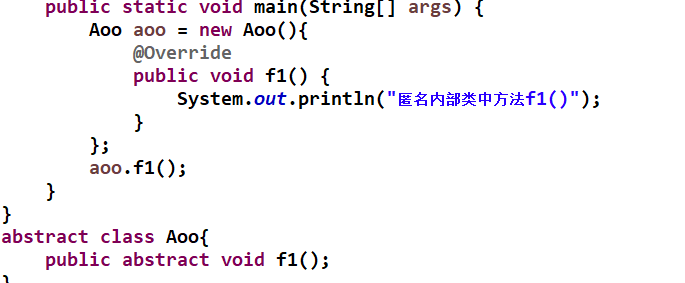
注意：抽象类可以创建匿名内部类，实体类只要不是最终类就可以创建匿名内部类。当利用接口创建匿名内部类的时候，实际上是实现了对应的接口。

如果匿名内部类定义在了方法或者语句内的时候，使用规则和方法是一致的。当定义在类内时，使用方法和成员内部类相同。本质上是继承了对应的类或者实现了对应的接口。

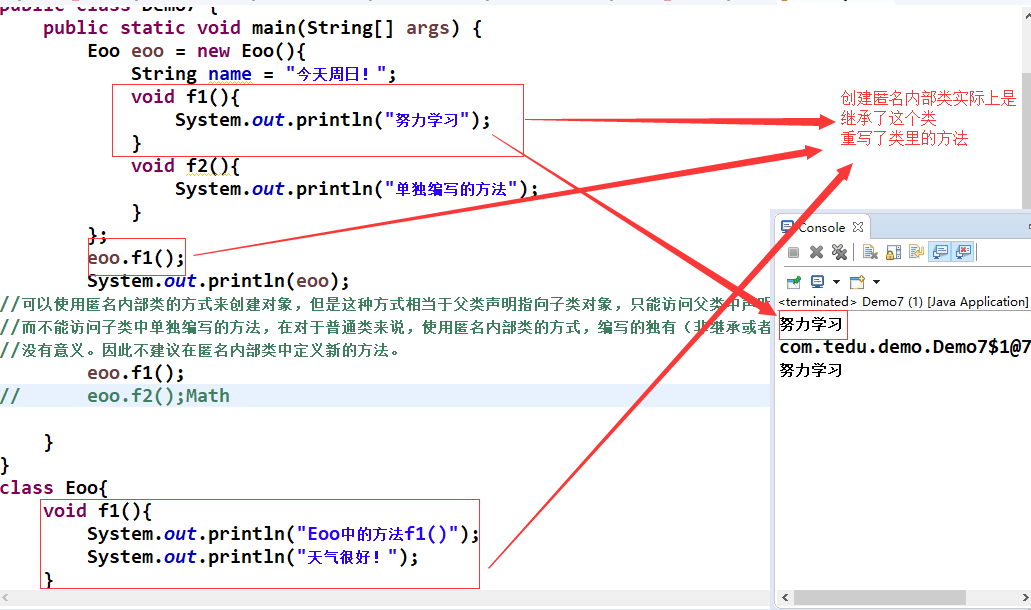
可以使用匿名内部累的方式创建对象

### 3.7.2 特点

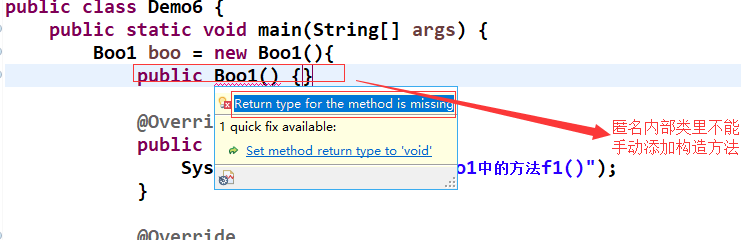
1.匿名内部类本质上是继承了对应的类或者实现对应的接口



2.只要一个类可以被继承，那么这个类就可以出现匿名内部类的形式，当利用一个类来创建一个匿名内部类的时候，实际上这个匿名内部类是继承了这个类



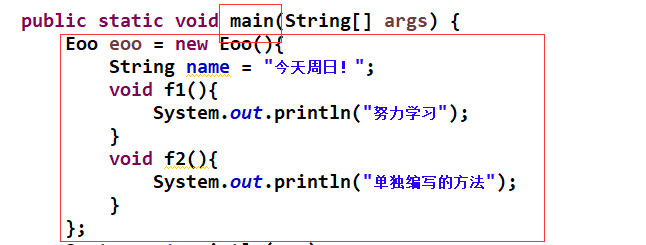
3.匿名内部类有构造函数。但是不能进行手动的添加



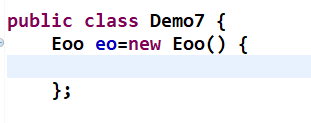
4. 当利用匿名内部类来创建对象的时候，要求这个匿名内部类必须实现父类中的所有抽象方法。



1. 如果匿名内部类定义到了方法中，此时匿名内部类的使用规则和方法内部类一致。



1. 如果匿名内部类定义到了类中，此时匿名内部类的使用规则和成员内部类一致。



## 扩展：内部接口

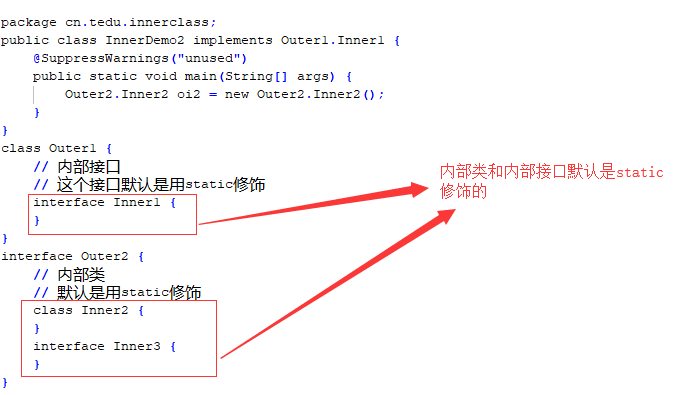
### 3.8.1 定义

定义在类或者接口中的接口叫做内部接口

### 3.8.2 特点

1. 类中定义的接口、接口中定义的类或者是接口默认都是用static修饰。

2. 接口必然要被实现，如果内部接口不是静态的，这个时候就需要一个外部类对象来调用接口，而在声明类实现这个接口的时候，无法创建对象，无法创建对象就无法实现了。



# 包

## 声明语句

声明包用的package，包的产生是为了解决同名文件的问题。

声明格式如下：



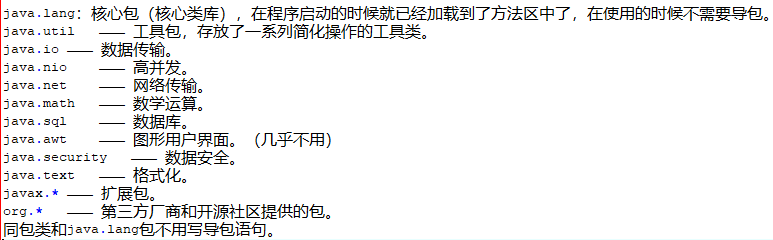
注意：一个Java文件中只允许存在一个package语句，而且这个package语句必须放在整个Java文件的首行。

## 导包语句

导入包用的是import



## 分类



注意：包名不能以java,javax,org开头

# 垃圾分代回收机制

垃圾回收针对的是堆内存。

对象在堆内存中存储，对象在使用完成之后会在不定的某个时刻被垃圾回收器（GC）解析掉。现阶段回收过程无法手动控制。

当调用构造方法的时候，创建好一个对象，因为Java中对每种数据类型都明确给定了大小，在创建对象的时候，会自动计算大小分配内存，所以在内存的回收和 释放的时候也是由Java自己管理。

堆内存分为了新生代/年轻代和老生代。新生代划分为伊甸园区和幸存区。一个对象新创建是放到了伊甸园区，如果这个对象在栈内存中没有引用，那么会在扫描 的时候被解析，释放内存；在伊甸园区经过了一次回收如果依然存活则标记到幸存区。幸存区的扫描频率要略低于伊甸园区。如果在幸存区中经过了多次扫描这个 对象依然没有被解析，则标记到老生代。老生代的扫描频率要远远低于新生代。如果老生代的对象发生了回收，导致程序的卡顿甚至崩溃。

发生在新生代的回收称之为minor gc，即初代回收。

发生在老生代的回收称之为full gc，即完全回收。