**使用内部类最吸引人的原因是：每个内部类都能独立地继承一个（接口的）实现，所以无论外围类是否已经继承了某个（接口的）实现，对于内部类都没有影响。**

**1、**内部类可以用多个实例，每个实例都有自己的状态信息，并且与其他外围对象的信息相互独立。

**2、**在单个外围类中，可以让多个内部类以不同的方式实现同一个接口，或者继承同一个类。

**3、**创建内部类对象的时刻并不依赖于外围类对象的创建。

**4、**内部类并没有令人迷惑的“is-a”关系，他就是一个独立的实体。

**5、**内部类提供了更好的封装，除了该外围类，其他类都不能访问。



内部了InnerClass可以对外围类OuterClass的属性进行无缝的访问，尽管它是private修饰的。这是因为当我们在创建某个外围类的内部类对象时，此时内部类对象必定会捕获一个指向那个外围类对象的引用，只要我们在访问外围类的成员时，就会用这个引用来选择外围类的成员。

引用内部类我们需要指明这个对象的类型：OuterClasName.InnerClassName。同时如果我们需要创建某个内部类对象，必须要利用外部类的对象通过.new来创建内部类： OuterClass.InnerClass innerClass = outerClass.new InnerClass();。

1. 成员内部类

成员内部类也是最普通的内部类，它是外围类的一个成员，所以他是可以无限制的访问外围类的所有 成员属性和方法，尽管是private的，但是外围类要访问内部类的成员属性和方法则需要通过内部类实例来访问。在成员内部类中要注意两点，**第一：**成员内部类中不能存在任何static的变量和方法；**第二：**成员内部类是依附于外围类的，所以只有先创建了外围类才能够创建内部类。

1. 局部内部类

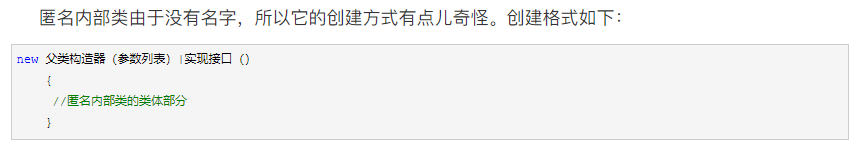
它是嵌套在方法和作用域内的，对于这个类的使用主要是应用与解决比较复杂的问题，想创建一个类来辅助我们的解决方案，到那时又不希望这个类是公共可用的，所以就产生了局部内部类，局部内部类和成员内部类一样被编译，只是它的作用域发生了改变，它只能在该方法和属性中被使用，出了该方法和属性就会失效。





1. 匿名内部类



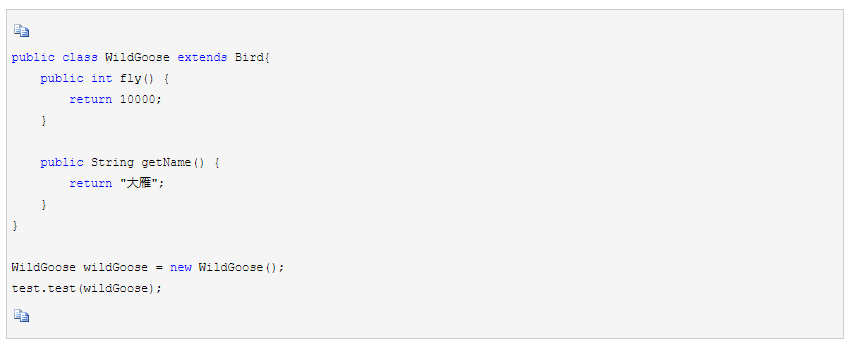


使用匿名内部类我们必须要继承一个父类或者实现一个接口，当然也仅能只继承一个父类或者实现一个接口。同时它也是没有class关键字，这是因为匿名内部类是直接使用new来生成一个对象的引用。当然这个引用是隐式的。



  在Test类中，test()方法接受一个Bird类型的参数，同时我们知道一个抽象类是没有办法直接new的，我们必须要先有实现类才能new出来它的实现类实例。所以在mian方法中直接使用匿名内部类来创建一个Bird实例。

     由于匿名内部类不能是抽象类，所以它必须要实现它的抽象父类或者接口里面所有的抽象方法。



   在这里系统会创建一个继承自Bird类的匿名类的对象，该对象转型为对Bird类型的引用。

      对于匿名内部类的使用它是存在一个缺陷的，就是它仅能被使用一次，创建匿名内部类时它会立即创建一个该类的实例，该类的定义会立即消失，所以匿名内部类是不能够被重复使用。对于上面的实例，如果我们需要对test()方法里面内部类进行多次使用，建议重新定义类，而不是使用匿名内部类。

**在使用匿名内部类的过程中，我们需要注意如下几点：**

**1、**使用匿名内部类时，我们必须是继承一个类或者实现一个接口，但是两者不可兼得，同时也只能继承一个类或者实现一个接口。

**2、**匿名内部类中是不能定义构造函数的。

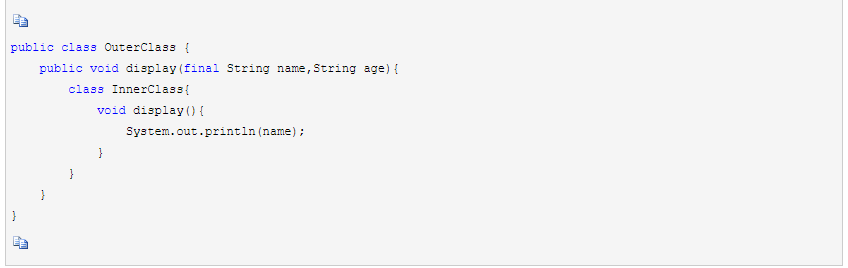
**3、**匿名内部类中不能存在任何的静态成员变量和静态方法。

**4、**匿名内部类为局部内部类，所以局部内部类的所有限制同样对匿名内部类生效。

**5、**匿名内部类不能是抽象的，它必须要实现继承的类或者实现的接口的所有抽象方法。

**注：当所在的方法的形参需要被内部类里面使用时，该形参必须为final。**

当外部类传入的参数需要被内部类调用时，从java程序的角度来看是直接被调用：



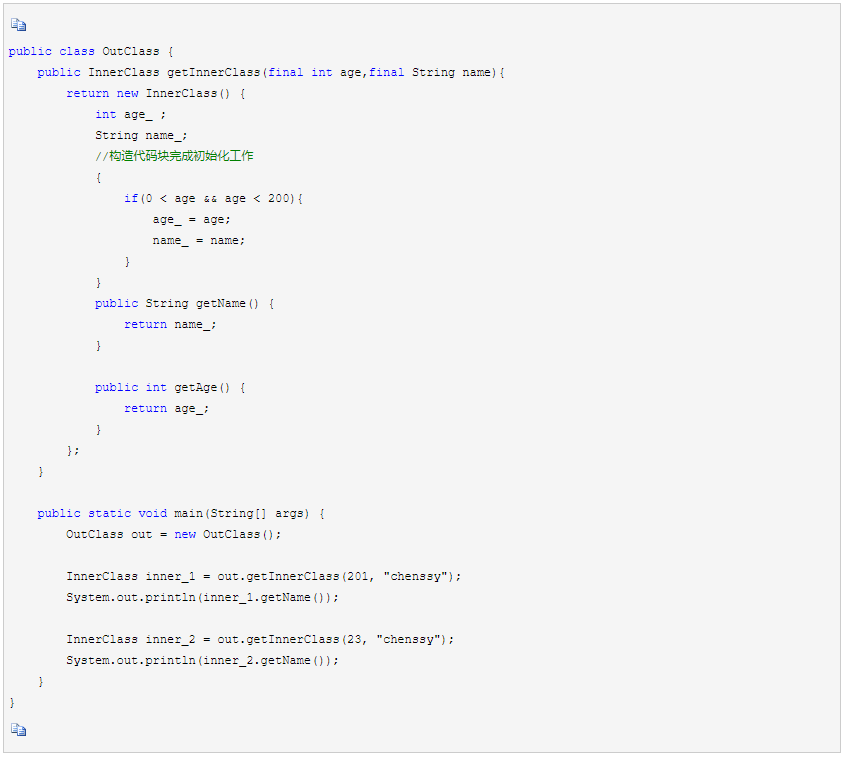
从上面代码中看好像name参数应该是被内部类直接调用？其实不然，在java编译之后实际的操作如下：



所以从上面代码来看，内部类并不是直接调用方法传递的参数，而是利用自身的构造器对传入的参数进行备份，自己内部方法调用的实际上是自己的属性而不是外部方法传递进来的参数。

所以在内部类中的属性和外部方法的参数两者从外表上看是同一个东西，但实际上却不是，所以他们两者是可以任意变化的，也就是说在内部类中我对属性的改变并不会影响到外部的形参，为此使用final限制

**匿名内部类初始化：我们一般都是利用构造器来完成某个实例的初始化工作的，但是匿名内部类是没有构造器的！那怎么来初始化匿名内部类呢？使用构造代码块！利用构造代码块能够达到为匿名内部类创建一个构造器的效果。**



1. 静态内部类

使用static修饰的内部类我们称之为静态内部类，不过我们更喜欢称之为嵌套内部类。静态内部类与非静态内部类之间存在一个最大的区别，我们知道非静态内部类在编译完成之后会隐含地保存着一个引用，该引用是指向创建它的外围内，但是静态内部类却没有。没有这个引用就意味着：

**1、** 它的创建是不需要依赖于外围类的。

**2、** 它不能使用任何外围类的非static成员变量和方法。