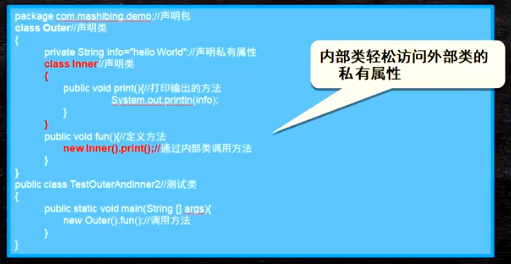
内部类—成员内部类

把一个类定义在另一个类的内部称为内部类

一个java文件中可以包含多个class，但是只能有一个public class



使用内部类：

创建内部类的时候，跟之前的方法不一样，需要在内部类的前面添加外部类来进行修饰

内部类可以方便地访问外部类的私有属性，但是外部类不能直接访问内部类的私有属性，需要创建内部类的对象之后才能访问

内部类中不能定义静态属性

如果外部类和内部类具有相同的成员变量或方法，内部类访问自己的成员变量或方法，如果要访问外部类的成员变量，需使用 **外部类类名.this.属性** 进行访问

**public class** InnerClassDemo {  
 **private int id**;  
 **private** String **name**;

**public void** show(){  
 System.***out***.println(**"show"**);

**new** InnerClass().test(); **//只用 test()是无法调用InnerClass里的**

**test方法的**  
 System.***out***.println(**new** InnerClass().**age**);

**//只用System.out.println(age)是无法调用内部类的私有属性值**

}

**public int** getId() {  
 **return id**;  
 }  
  
 **public void** setId(**int** id) {  
 **this**.**id** = id;  
 }  
  
 **public** String getName() {  
 **return name**;  
 }  
  
 **public void** setName(String name) {  
 **this**.**name** = name;

}

**class** InnerClass{

**private int age**;  
 **public void** test(){  
 System.***out***.println(**"test"**);

System.***out***.println(**id**); **//内部类可以调用外部类的私有属性**  
 System.***out***.println(**name**);   
 }  
 }  
}

**public class** TestInnerClass {  
  
 **public static void** main(String[] args) {  
 InnerClassDemo innerClassDemo = **new** InnerClassDemo();  
 innerClassDemo.show();

System.***out***.println(innerClassDemo.getName());

InnerClassDemo.InnerClass inner = **new** InnerClassDemo().**new**

InnerClass(); **//这行是new内部类的方法**  
 inner.test();  
 }  
}

执行结果：

show

0

null

test

0

null

**public class** Outer {  
 **private** String **name** = **"zhangsan"**;  
 **class** Inner{  
 **private** String **name** = **"lisi"**;  
 **public void** show(){  
 System.***out***.println(**name**);  
 System.***out***.println(**this**.**name**);  
 System.***out***.println(Outer.**this**.**name**);  
 }  
 }  
  
 **public static void** main(String[] args) {  
 Outer.Inner inner = **new** Outer().**new** Inner();  
 inner.show();  
 }  
}

执行结果：

lisi

lisi

zhangsan

如果main方法在外部类内部，则可以省略Outer

Outer.Inner inner = **new** Outer().**new** Inner();

**分类**：

* **匿名内部类**：

当定义了一个类，实现了某个接口的时候，在使用过程中只需要使用一次，没有其他用途。此时考虑到代码编写的简洁，可以考虑不创建具体的类，而采用new interface() (添加未实现的方法)

**public class** AnonymousInnerClassDemo {  
 **public static void** main(String[] args) {  
 System.***out***.println(**"一万行代码"**);  
 **new** Thread(**new** Runner());  
 System.***out***.println(**"一万行代码"**);  
 }  
}  
  
**class** Runner **implements** Runnable{  
 @Override  
 **public void** run() {  
  
 }  
}

这里的Rnner类只用了一次，为这一次新建一个类比较浪费，可以↓这样写

**public class** AnonymousInnerClassDemo {  
 **public static void** main(String[] args) {  
 System.***out***.println(**"一万行代码"**);  
 **new** Thread(**new** Runnable() {  
 @Override  
 **public void** run() {  
   
 }  
 });  
 System.***out***.println(**"一万行代码"**);  
 }  
}

new Runnable()写法相当于new AAA Implements Runnable()，看起来是new了一个接口，其实是new出来一个接口的实现类，而这个接口里的所有的方法都必须要实现

* 静态内部类

在内部类中可以定义静态内部类，使用static关键字进行修饰

语法：

new 外部类类名.内部类().方法名

**外部类类名.内部类 内部类对象名=new 外部类类名.内部类类名();**

使用static声明的内部类不能访问非static的外部属性

**public class** StaticClass {  
  
 **private int id**;  
  
 **public void** test(){  
 System.***out***.println(**"test"**);  
 }  
  
 **static class** InnerClass{  
 **private** String **name**;  
 **public void** show(){  
 System.***out***.println(**"show"**);  
 }  
 }  
  
 **public static void** main(String[] args) {  
 InnerClass innerClass = **new** StaticClass.InnerClass();  
 InnerClass innerClass = **new** StaticClass().**new** InnerClass();**//错误**  
 }  
}

之前new一个内部类是使用**new** StaticClass().**new** InnerClass()这种方法的，但是这里的内部类是static的，不能new，这时把内部类当作一个普通的成员属性，通过 **类名.静态属性** 就可以

* 方法内部类

将内部类定义在外部类的方法中（了解即可）

方法内部类只能在方法中创建对象，因此class的作用域就是当前方法

**public class** MethodInnerClass {  
  
 **public void** show(){  
 System.***out***.println(**"show"**);  
  
 **class** InnerClass{  
 **private** String **name**;  
 **public void** test(){  
 System.***out***.println(**"test "**);  
 }  
 **public void** test(**int** a){  
 System.***out***.println(a);  
 }  
 }  
  
 **new** InnerClass().test(); **//无法在main方法里调用这个new的对象  
 new** InnerClass().test(12);  
 }  
  
 **public static void** main(String[] args) {  
 MethodInnerClass methodInnerClass = **new** MethodInnerClass();  
 methodInnerClass.show();  
 }  
}

执行结果：

show

test

12