Java内部类

（1）成员内部类

class A{

class B{

}

}

成员内部类可以无条件访问外部类所有成员属性和成员方法（包括private成员和静态成员）

不过，当成员内部类拥有和外部类同名的成员变量或方法时，会发生隐藏现象，即默认访问成员内部类的成员。如果要访问外部类的同名成员，需要以下面的形式访问：

外部类.this.成员变量

外部类.this.成员方法

外部类如果要访问成员内部类的成员，必须先创建一个成员内部类的对象，再通过指向这个对象的引用来访问

成员内部类是依附于外部类而存在的，也就是说，如果要创建成员内部类的对象，前提是必须存在一个外部类的对象。

Outter outer = new Outter();

Outter.Inner inner = outer.new Inner();

成员内部类可以用private修饰：

public修饰：

protected修饰：

默认访问权限：

外部类只能被public和包访问两种权限修饰

（2）局部内部类

class A{

function(){

class B{

}

}

}

局部内部类的访问权限仅限于方法内或该作用域内。局部内部类就像是方法里面的一个局部变量一样，是不能有public，protected，private以及static这些成员修饰符修饰的。

（3）匿名内部类

常常用于写android代码的事件监听，如下：

btn.setOnClickListener(new OnClickListener(){

pubic void onClick(View v){

}

});

**注意：**

**局部内部类**和**匿名内部类**都**只能访问到final的局部（即该类/对象所在的方法中）变量，所以要引用外面的局部变量是需要final的**，这样在回调时才能找到那个变量。

而如果是**引用外部类的成员变量，就不需要final，因为内部类本身都会含有一个外部类的引用（外部类.this）**

由于成员内部类本身就不在局部方法中，是直接在外部类中，因此并不存在访问权限的限制。

（4）静态内部类

……

另外，一个对象被声明为final，不能更改的是这个对象的引用，而并非这个对象的各个属性值。