内部类

在Java语言中，可以把一个类定义到另外一个类的内部，在类里面的这个类就叫做内部类，外面的类叫做外部类。在这种情况下，这个内部类可以被看成外部类的一个成员（与类的属性和方法类似）。还有一种类被称为顶层（Top-level）类，指的是类定义代码不嵌套在其他类定义中的类。

内部类主要有以下四种：静态内部类，成员内部类，局部内部类，匿名内部类。其定义方法如下：

```
class outerClass{
     static class innerClass{}//静态内部类
}
```

静态内部类是指被声明为static的内部类。它可以不依赖于外部类实例而被实例化，而通常的内部类需要在外部类实例化后才能实例化。静态内部类不能与外部类有相同的名字，不能访问外部类·的普通成员变量，只能访问外部类中的静态成员和静态方法（包括私有类型）

```
class outerClass{
    class innerClass{}//成员内部类（普通内部类）
}
```

一个静态内部类，如果去掉static关键字，就成为成员内部类。成员内部类为非静态内部类，它可自由引用外部类的属性和方法，无论这些属性和方法是静态的和非静态的。但是它与一个实例绑在了一起，不可以定义静态的属性和方法。只有在外部的类被实例化后，这个内部类才能被实例化。注意：非静态内部类中不能有静态成员。

```
class outerClass{
    public void memberFunction(){
       class innerClass{}//局部内部类
    }
}
```

局部内部类指的是定义在一个代码块内的类，其作用范围为其所在的代码块，是内部类中使用较少的一种类型。局部内部类像局部变量一样，不能被public,protected,private,static修饰，只能访问方法中定义为final类型的局部变量。

```
public class MyFrame extends Frame{
     //外部类
     public MyFrame(){
        addWindowListener(new WindowAdapter()
        {//匿名内部类
           public void windowClosing(WindowEvent e){
              dispose();
              System.exit(0);
            }
        });
```

```
    }
}
```

匿名内部类是一种没有类名的内部类，不使用关键字class,extends和implements，没有构造方法，它必须继承其他类或实现其他接口。匿名内部类的一般好处是使代码更加简洁紧凑，但易读性下降。一般应用于GUI（图形用户界面）编程中实现事件处理等。

静态内部类和非静态内部类一样，都是在被调用时才会被加载

静态内部类和静态方法一样，静态方法只有被调用的时候才会执行，而静态内部类是只有在第一次调用的时候才会进行初始化。

静态内部类的加载不需要依附外部类，在使用时才加载。不过在加载静态内部类的过程中也会加载外部类。