接口中的内部类

我们在实际开发过程中，如果想要创建某些公共代码，使得它们可以被某个接口的所有不同实现所共用，那么接口内部的嵌套类会显得很方便。也就是说，在接口中可以含有内部类。在这里，向大家展示接口中放置普通成员内部类和抽象成员内部类的情况。

❶ 首先创建接口，接口中定义了普通内部类 InnerClass 和抽象内部类 AbInnerClass

```
package com.imooc.inter;
//接口IOuterInterface
public interface IOuterInterface {
    int TEMP=100;  //常量
    void abMethod();  //抽象方法
    public default void deMethod(){  //jdk1.8后可添加
        System.out.println("接口中默认方法");
    }
    public static void stMethod(){  //jdk1.8后可添加
        System.out.println("接口中静态方法");
    }

    //普通内部类
    public class InnerClass{
        public void show(){
            System.out.println("接口中可定义普通成员内部类");
        }
    }

    //抽象内部类
    public abstract class AbInnerClass{
        public abstract void abInfo();
        public void info(){
            System.out.println("接口中可定义抽象成员内部类");
        }
    }
}
```

❷ 普通成员内部类的实例化

创建接口的实现类 ClassDemo

```java
package com.imooc.inter;
//实现类ClassDemo
public class ClassDemo implements IOuterInterface {

    @Override
    public void abMethod() {
        System.out.println("实现类");
    }
    //获取接口中内部类方法
    public InnerClass getInner(){
        return new InnerClass();
    }
}
```

获取普通内部类对象，调用方法

```java
package com.imooc.test;
import com.imooc.inter.ClassDemo;
import com.imooc.inter.IOuterInterface;
import com.imooc.inter.IOuterInterface.InnerClass;

//测试类：普通成员内部类
public class Test {

    public static void main(String[] args) {
        /*第一种实例化对象方式：
         * 通过 接口名.类名 进行实例化
         */
        IOuterInterface.InnerClass inner=new IOuterInterface.InnerClass();
        inner.show();

        /*第二种实例化对象方式：
         * 通过在实现类中创建接口中内部类获取方法
         * 用实现类对象调用获取方法
         */
        ClassDemo demo=new ClassDemo();
        demo.getInner().show();

        /*第三种实例化对象方式：
         * 将内部类导入后，直接实例化
         */
        InnerClass innerTwo=new InnerClass();
        innerTwo.show();

    }
}
```

运行结果：

```
接口中可定义普通成员内部类
==========================
接口中可定义普通成员内部类
==========================
接口中可定义普通成员内部类
```

❸ 抽象成员内部类的实例化

创建接口的实现类 AbClassDemo

```
package com.imooc.inter;
//实现类AbClassDemo
public class AbClassDemo implements IOuterInterface {

    @Override
    public void abMethod() {

    }
    //继承抽象类AbInnerClass
    public class AbDemo extends AbInnerClass{
        @Override
        public void abInfo() {
            System.out.println("重写接口中抽象类中的抽象方法");
        }
    }
}
```

获取抽象内部类对象，调用方法

```
package com.imooc.test;

import com.imooc.inter.AbClassDemo;
import com.imooc.inter.IOuterInterface;
//测试类：抽象成员内部类
public class TestOne {

    public static void main(String[] args) {
        /*第一种实例化对象方式：
         * 通过 接口名.类名 进行实例化
         * 但是对于抽象类而言，不能直接实例化，所以这里可使用匿名内部类的方式
         */
        IOuterInterface.AbInnerClass abInner=new IOuterInterface.AbInnerClass(){
            public void abInfo(){
                System.out.println("重写抽象类中的抽象方法");
            }
        };
        abInner.abInfo();
        abInner.info();
        System.out.println("======================");
        /*第二种实例化方法：
         * 在实现类中定义内部类继承接口中的抽象内部类
         */
        IOuterInterface.AbInnerClass abInnerOne=new AbClassDemo().new AbDemo();
        abInnerOne.abInfo();
        abInnerOne.info();

    }
}
```

运行结果：

```
重写抽象类中的抽象方法
接口中可定义抽象成员内部类
======================
重写接口中抽象类中的抽象方法
接口中可定义抽象成员内部类
```

在这里只是提供给大家几种实例化接口中内部类的思路和方式，大家也可以用其他方式去进行对象实例化，当然前提是要满足代码规则。