内部类

武永亮

讲授思路

- 概述
- 成员内部类
- 局部内部类
- 匿名内部类
- 静态内部类

概述

- 内部类（inner class）是定义在另一个类中的类。
- 内部类的特点：
  - 内部类是一种编译时的语法，编译后生成的两个类是独立的两个类。（对于一个名为outer的外部类和其内部定义的名为inner的内部类。编译完成后出现outer.class和outer$inner.class两类）。
  - 可以自由访问外部类的任何成员（包括私有成员），但外部类不能直接访问内部类的成员。
  - 内部类可为静态，可以用protected、private修饰，而外部类只能使用public和缺省的包访问权限。
  - 用内部类定义在外部类中不可访问的属性。这样就在外部类中实现了比外部类的private还要小的访问权限。

概述

- 内部类的形式：
  - 成员内部类
  - 局部内部类
  - 匿名内部类
  - 静态内部类

讲授思路

- 概述
- 成员内部类
- 局部内部类
- 匿名内部类
- 静态内部类

成员内部类-概念

- 成员内部类：作为外部类的一个成员存在，与外部类的属性、方法并列。内部类和外部类的实例变量可以共存。
- 举例：

```
class OuterClass{
        OuterClass() {
                System.out.println("Outer class");
        }
        class InnerClass {  //定义内部类
                InnerClass(){
                        System.out.println("Inner class");
                }
        }
}
```

成员内部类-创建对象

- 建立内部类对象时应注意：
  - 在外部类的内部可以直接使用inner s=new inner();（因为外部类知道inner是哪个类，所以可以生成对象）。
  - 在外部类的外部，要生成（new）一个内部类对象，需要首先建立一个外部类对象（外部类可用），然后在生成一个内部类对象。

成员内部类-创建对象

- 举例：

```
public class Test{
    public static void main(string[] args){
        OuterClass outer = new OuterClass();

        //创建内部类对象
        OuterClass.InnerClass  inner = outer.new  InnerClass ();
        //或
        // OuterClass.InnerClass  inner  = new OuterClass().new  InnerClass ();
    }
}
```

成员内部类-访问

- 在内部类中访问内部类变量：this.属性。
- 内部类作为外部类的成员，可以访问外部类的私有成员或属性。访问方式：外部类名.this.属性。
- 在外部类的外部访问内部类，内部类的对象.对象名。

讲授思路

- 概述
- 成员内部类
- 局部内部类
- 匿名内部类
- 静态内部类

局部内部类

- 在方法中定义的内部类称为局部内部类，类似局部变量。
- 局部内部类不能加修饰符public、protected和private，其范围为定义它的代码块。

局部内部类

```
class OuterClass{
        //称为外包方法
        public void display() {
            //定义局部内部类
            class InnerClass {
                    public void displayInner() {
                            System.out.println("display inner class");
                    }
            }
            InnerClass in = new InnerClass();
            in.displayInner();
        }
}
```

局部内部类-特点

- 可以访问外包方法之外的外部类之内的所有成员。还可以访问所在外包方法中的final类型的参数。

局部内部类-特点

- 举例：

```
class OuterClass {
        //称为外包方法
        public void display (final  boolean  isPrint) {
            class InnerClass {
                    public void displayInner() {
                        if(isPrint) {
                            System.out.println("display inner class");
                        }
                    }
            }
            InnerClass in = new InnerClass();
            in.displayInner();
        }
}
```

讲授思路

- 概述
- 成员内部类
- 局部内部类
- 匿名内部类
- 静态内部类

局部内部类-特点

- 局部内部类不能声明为接口。
- 要想使用局部内部类时，需要生成外部类对象，通过外部类对象调用外包方法，在方法中才能调用局部内部类。

匿名内部类

- 将局部内部类的使用再深入一步。假如只创建这个类的一个对象，就不必对内部类命名了。这种类被称为匿名内部类。

```
interface  anonymous{     //定义一个接口
        void  display();
}
class NoClassName{
        public void print(){
                anonymous anon = new anonymous (){  //定义匿名类
                    public  void  display()
                    {
                        System.out.println(“implement anonymous “);
                    }
                };
                anon.display();
        }
}
```

匿名内部类-特点

- 匿名类没有类名，它必须继承一个类或是实现一个接口。不能有显示的extends和implement子句。
- 匿名类不能有构造函数，因为它没有类名。可以通过new
- <父类名>的方法创建对象，匿名类的创建和定义同时进行。
- 匿名类只能一次性的创建其对象。
- 匿名类可以在方法体中，也可以在参数列表中。

匿名内部类-注意事项

- 匿名内部类一定是在new的后面其隐含实现一个接口或一个类，没有类名。
- 匿名内部类不能是抽象类。
- 匿名内部类必须实现它的抽象父类或者接口里的所有抽象方法。

讲授思路

- 概述
- 成员内部类
- 局部内部类
- 匿名内部类
- 静态内部类

静态内部类

- 静态内部类定义在类中，在任何方法之外，用static定义。
- 举例：

```
class Outer {
        public void display(){}
        protected  static  class Inner{ }//静态成员类
}
```

- 静态内部类的对象可以直接生成。

```
class Test {
        public static void mian(string[] args) {
                Outer.Inner in = new  Outer.Inner ();
        }
}
```

静态内部类-特点

- 静态内部类能直接访问外部类的静态成员，不能直接访问外部类的实例成员。
- 静态内部类里面可以定义静态成员，其他内部类不可以。

总结

- 概述
- 成员内部类
- 局部内部类
- 匿名内部类
- 静态内部类

课后阅读

- 总结使用内部类的原因。
- 总结内部类的语法、特点。

Thank You