目录

1. 什么是内部类？

内部类是指在一个外部类的内部再定义一个类。内部类作为外部类的一个成员，并且依附于外部类而存在的。

内部类可为静态，可用 protected 和 private 修饰（而外部类只能使用 public 和缺省的包访问权限）。

内部类主要有以下几类：成员内部类、局部内部类、静态内部类、匿名内部类

2. 内部类的作用

1) **内部类可以很好的实现隐藏：**一般的非内部类，是不允许有 private 与 protected 权限的，但内部类可以

2) **内部类拥有外围类的所有元素的访问权限**

3) **可是实现多重继承**

4) **可以避免修改接口而实现同一个类中两种同名方法的调用。**

3. 静态内部类的设计意图

在成员内部类中要注意两点：

(1)成员内部类中不能存在任何 static 的变量和方法；

(2)成员内部类是依附于外围类的，所以只有先创建了外围类才能够

创建内部类。

静态内部类与非静态内部类之间存在一个最大的区别：

非静态内部类在编译完成之后会隐含地保存着一个引用，该引用是指向创建它的外围内，但是静态内部类却没有。

没有这个引用就意味着：

1) 它的创建是不需要依赖于外围类的。
2) 它不能使用任何外围类的非 static 成员变量和方法。

4. 成员内部类、静态内部类、局部内部类和匿名内部类的理解，以及项目中的应用

放在一个类的内部的类我们就叫内部类。

一、成员内部类

定义在类内部的非静态类，就是成员内部类。

```
public class Out {
```

```
    private static int a;

    private int b;



    public class Inner {

        public void print() {

            System.out.println(a);

            System.out.println(b);

        }

    }

}
```

二、静态内部类：

定义在类内部的静态类，就是静态内部类。

```
public class Out {

    private static int a;

    private int b;



    public static class Inner {

        public void print() {

            System.out.println(a);

        }
```

```
    }
}
```

应用场景：

Java 集合类 HashMap 内部就有一个静态内部类 Entry。Entry 是 HashMap 存放元素的抽象，HashMap 内部维护 Entry 数组用了存放元素，但是 Entry 对使用者是透明的。像这种和外部类关系密切的，且不依赖外部类实例的，都可以使用静态内部类。

三、局部内部类

定义在方法中的类，就是局部类。（局部内部类是嵌套在方法和作用于内的，对于这个类的使用主要是应用与解决比较复杂的问题，想创建一个类来辅助我们的解决方案，到那时又不希望这个类是公共可用的，所以就产生了局部内部类，局部内部类和成员内部类一样被编译，只是它的作用域发生了改变，它只能在该方法和属性中被使用，出了该方法和属性就会失效。）

```
public class Out {

    private static int a;

    private int b;



    public void test(final int c) {

        final int d = 1;
```

```
        class Inner {

            public void print() {

                System.out.println(a);

                System.out.println(b);

                System.out.println(c);

                System.out.println(d);

            }

        }

    }



    public static void testStatic(final int c) {

        final int d = 1;

        class Inner {

            public void print() {

                System.out.println(a);

                //定义在静态方法中的局部类不可以访问外部类的实例变量

                //System.out.println(b);

                System.out.println(c);

                System.out.println(d);

            }

        }

    }
```

```
}
```

应用场景：

如果一个类只在某个方法中使用，则可以考虑使用局部类。

四、匿名内部类

匿名内部类是没有访问修饰符的。

1） new 匿名内部类，这个类首先是要存在的。

2） 当所在方法的形参需要被匿名内部类使用，那么这个形参就必须为 final。

3） 匿名内部类没有明面上的构造方法，编译器会自动生成一个引用外部类的构造方法。

```
public class Out {

    private static int a;

    private int b;



    private Object obj = new Object() {

        private String name = "匿名内部类";

        @Override

        public String toString() {

            return name;
```

```
        }
    };


    public void test() {
        Object obj = new Object() {
            @Override
            public String toString() {
                System.out.println(b);
                return String.valueOf(a);
            }
        };
        System.out.println(obj.toString());
    }
}
```

应用场景：

匿名内部类使用广泛，比如我们常用的绑定监听的时候

```
view.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
    Toast.makeText(v.getContext(),"click",Toast.LENGTH_SHORT).show();    }
});
```

5. 闭包和局部内部类的区别

1） 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 修饰符的。

2） 闭包（Closure）是一种能被调用的对象，它保存了创建它的作用域的信息。JAVA 并不能显式地支持闭包，但是在 JAVA 中，闭包可以通过“接口+内部类”来实现。

例如：一个接口程序员和一个基类作家都有一个相同的方法 work，相同的方法名，但是其含义完全不同，这时候就需要闭包。

```
class Food{

    public static final String name = "Food";

    private static int num = 20;

    public Food() {

        System.out.println("Delicious Food");

    }



    public Active getEat() {

        return new EatActive();

    }

    private class EatActive implements Active {
```

```
        @Override
        public void eat() {
            if (num == 0) {
                System.out.println("吃货，已经吃没了");
            }
            num --;
            System.out.println("吃货，你吃了一份了");
        }
    }

    public void currentNum() {
        System.out.println("还剩:"+num+"份");
    }
}

interface Active{
    void eat();
}
```