

java内部类

编辑

收藏31323

java内部类分为：成员内部类、静态嵌套类、方法内部类、匿名内部类。

中文名	java内部类	分 为	成员方法匿名内部类静态嵌套类
外文名	Java inner classes	内部类是	独立的类
		不 能	用普通的方式访问

目录

- 1 内部类的共性

2 成员内部类

3 方法内部类

4 匿名内部类

5 静态嵌套类



java内部类图册

内部类的共性

编辑

- (1)、内部类仍然是一个独立的类，在编译之后内部类会被编译成独立的.class文件，但是前面冠以外部类的类名和\$符号。

(2)、内部类不能用普通的方式访问。内部类是外部类的一个成员，因此内部类可以自由地访问外部类的成员变量，无论是否是private的。

(3)、内部类声明成静态的，就不能随便的访问外部类的成员变量了，此时内部类只能访问外部类的静态成员变量。

成员内部类

编辑

```
class Outer {  
  
    class Inner{  
  
    }  
}
```

编译上述代码会产生两个文件：Outer.class和Outer\$Inner.class。

方法内部类

编辑

把类放在方法内

```
class Outer {  
  
    public void doSomething(){  
  
        class Inner{  
  
            public void seeOuter(){  
  
            }  
  
        }  
  
    }  
}
```

- (1)、方法内部类只能在定义该内部类的方法内实例化，不可以在此方法外对其实例化。

V百科往期

分享

☆

👤

👤

👤

图解百科 | 第617期
当鏢局赶上信息时代，

程序设计类书籍

java完全自... 疯狂java讲义 java经典

java数据结... java经典实例 java编程

java线程 java核心技术... java开发

相关术语

多线程 java虚拟机 中文编

swing 递归算法 maprec

计算机术语

maven 冒泡排序 泛型

(2)、方法内部类对象不能使用该内部类所在方法的非final局部变量。

因为方法的局部变量位于栈上，只存在于该方法的生命期内。当一个方法结束，其栈结构被删除，局部变量成为历史。但是该方法结束之后，在方法内创建的内部类对象可能仍然存在于堆中！例如，如果对它的引用被传递到其他某些代码，并存储在一个成员变量内。正因为不能保证局部变量的存活期和方法内部类对象的一样长，所以内部类对象不能使用它们。

下面是完整的例子：

```
class Outer {

    public void doSomething(){

        final int a =10;

        class Inner{

            public void seeOuter(){

                System.out.println(a);

            }

        }

        Inner in = new Inner();

        in.seeOuter();

    }

    public static void main(String[] args) {

        Outer out = new Outer();

        out.doSomething();

    }

}
```

匿名内部类

编辑

顾名思义，没有名字的内部类。表面上看起来它们似乎有名字，实际那不是它们的名字。

当程序中使用匿名内部类时，在定义匿名内部类的地方往往直接创建该类的一个对象。匿名内部类的声明格式如下：

```
new ParentName ( ) {

    ...// 内部类的定义

} [1]
```

匿名内部类就是没有名字的内部类。什么情况下需要使用匿名内部类？如果满足下面的一些条件，使用匿名内部类是比较合适的：

- 只用到类的一个实例。
- 类在定义后马上用到。
- 类非常小（SUN推荐是在4行代码以下）
- 给类命名并不会导致你的代码更容易被理解。

在使用匿名内部类时，要记住以下几个原则：

- 匿名内部类不能有构造方法。
- 匿名内部类不能定义任何静态成员、静态方法。
- 匿名内部类不能是public,protected,private,static。
- 只能创建匿名内部类的一个实例。

·一个匿名内部类一定是在new的后面，用其隐含实现一个接口或实现一个类。

·因匿名内部类为局部内部类，所以局部内部类的所有限制都对其生效。

A、继承式的匿名内部类



词条统计

浏览次数：126370次
编辑次数：26次历史版本
最近更新：2016-06-19
创建者：[繆媛](#)

搜索推荐

- | | |
|---------|--------|
| 语言培训学校 | 怎么创建网站 |
| 极客学院 | 语言表达能力 |
| 小语言店铺装修 | 尚学堂 |
| 反应釜 | 廖雪峰git |
| 北京语言大学 | 网易公开课 |

- | | |
|-------------|--------------|
| 1 mvc教程 | 12 spring 教程 |
| 2 django 教程 | 13 flask |
| 3 单片机培训机构 | 14 净水器厂家 |
| 4 otherwise | 15 web前端培训 |
| 5 linux培训 | 16 外贸网站建 |
| 6 uft | 17 ps培训 |
| 7 编程学习入门 | 18 日语基础学 |
| 8 网站制作公司 | 19 网页制作 |
| 9 榻榻米床价格 | 20 国外网站设 |
| 10 spring教程 | 21 高速摄像机 |
| 11 文献库 | 22 公司 |

分享



百科美图编辑赛
等你参加



```
public class Car {  
  
    public void drive(){  
  
        System.out.println("Driving a car!");  
  
    }  
  
    public static void main(String[] args) {  
  
        Car car = new Car(){  
  
            public void drive() {  
  
                System.out.println("Driving another car!");  
  
            }  
  
        };  
  
        car.drive();  
  
    }  
}
```

结果输出了：Driving another car! Car引用变量不是引用Car对象，而是Car匿名子类的对象。

B、接口式的匿名内部类。

```
interface Vehicle {  
  
    public void drive();  
  
}  
  
class Test{  
  
    public static void main(String[] args) {  
  
        Vehicle v = new Vehicle(){  
  
            public void drive(){  
  
                System.out.println("Driving a car!");  
  
            }  
  
        };  
  
        v.drive();  
  
    }  
}
```

上面的代码很怪，好像是在实例化一个接口。事实并非如此，接口式的匿名内部类是实现了一个接口的匿名类。而且只能实现一个接口。

C、参数式的匿名内部类。

```
class Bar{  
  
    void doStuff(Foo f){  
  
        f.foo();  
  
    }  
  
}  
  
interface Foo{  
  
    void foo();  
  
}  
  
class Test{
```

分享



百科美图编辑赛
等你参加



```
static void go(){

    Bar b = new Bar();

    b.doStuff(new Foo(){

        public void foo(){

            System.out.println("foofy");

        }

    });

}
```

静态嵌套类

 编辑

静态内部类中可以定义静态或者非静态的成员。

从技术上讲，静态嵌套类不属于内部类。因为内部类与外部类共享一种特殊关系，更确切地说是对实例的共享关系。而静态嵌套类则没有上述关系。它只是位置在另一个类的内部，因此也被称为顶级嵌套类。

静态的含义是该内部类可以像其他**静态成员**一样，没有外部类对象时，也能够访问它。静态嵌套类仅能访问外部类的静态成员和方法。

```
class Outer{

    static class Inner{}

}

class Test {

    public static void main(String[] args){

        Outer.Inner n = new Outer.Inner();

    }

}
```

在静态方法中定义的内部类也是**StaticNested Class**，这时候不能在类前面加**static**关键字，静态方法中的**StaticNested Class**与普通方法中的内部类的应用方式很相似，它除了可以直接访问外部类中的**static**的成员变量，还可以访问静态方法中的局部变量，但是，该局部变量前必须加**final**修饰符。

为什么需要内部类

典型的情况是，内部类继承自某个类或实现某个接口，内部类的代码操作创建其他外围类的对象。所以你可以认为内部类提供了某种进入其外围类的窗口。使用内部类最吸引人的原因是：

每个内部类都能独立地继承自一个（接口的）实现，所以无论外围类是否已经继承了某个（接口的）实现，对于内部类都没有影响。如果没有内部类提供的可以继承多个具体的或抽象的类的能力，一些设计与编程问题就很难解决。从这个角度看，内部类使得**多重继承**的解决方案变得完整。接口解决了部分问题，而内部类有效地实现了“多重继承”。

参考资料

- 桂珠、张平、陈爱国. **Java**面向对象程序设计（jdk1.6）第三版：北京邮电大学出版社，2005

分享











猜你喜欢

什么是java

java能做什么

java有什么用

什么是java编程

学java哪个学校好

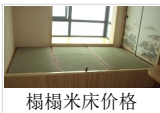
学java怎么样

java学校

学java

小吊车

蚂蚁微贷



百科美图编辑赛
等你参加

② 新手上路

- 成长任务
- 编辑入门
- 编辑规则
- 百科术语

📖 我有疑问

- 我要质疑
- 我要提问
- 参加讨论
- 意见反馈

🗉 投诉建议

- 举报不良信息
- 未通过词条申诉
- 投诉侵权信息
- 封禁查询与解封

©2017 Baidu 使用百度前必读 | 百科协议 | 百度百科合作平台 | 京ICP证030173号 

京公网安备11000002000001号

分享

▼









百科美图编辑赛
等你参加

