Java内部类

悬赏：20 发布时间：2008-08-04 提问人：wenzhihua1983 (初级程序员)

谁能从概要上和细节上，详细讲述Java的内部类，3Q

采纳的答案

2008-08-04 aidiyuxin (资深程序员)

内部类的一些知识
关键字: java基础
小弟最近学习内部类总结了下面一些内容，参考了一些资料，不足之处请指出，谢谢！
成员内部类：
相当于类的非静态成员，可以用权限修饰符来修饰，包括private、protected、public.
1、定义成员内部类

```
class Outter {
//非静态内部类
class Inner {
//内部类成员
int i = 12;
}
//外部类的普通成员
int j = 0;
}
```

2、外部类之内创建成员内部类对象
语法和普通的创建对象相同，用new操作符调用相应的构造方法即可。注意的是，非静态内部类属于外部类的非静态成员，不能在静态上下文使用。
例子：

```
class Outter {
//非静态内部类
class Inner {
//内部类成员
int i = 12;
```

```
public void innerTest() {
System.out.println(“Inner Class Method”);
}
}
//外部类的普通成员
int j = 0;
public void test() {
Inner inner = new Inner();
inner.innerTest();
}
}
```

3、外部类之外创建成员内部类对象

既然是外部类的非静态成员，就必须在外部类对象存在的情况下使用。

基本语法是

<外部类类名>.<内部类类名> 引用变量名称 = <外部类对象的引用>.new <内部类构造器>;

<外部类类名>.<内部类类名> 引用变量名称 = new <外部类构造器>.new <内部类构造器>;

(1)、创建内部类对象时对象引用必须是<外部类类名>.<内部类类名>

(2)、调用内部类的构造方法不能直接用new 而是要用<外部类对象的引用>.new调用

例子：

```
class Outter {
//非静态内部类
class Inner {
//内部类成员
int i = 12;
public void innerTest() {
System.out.println(“Inner Class Method”);
}
}
//外部类的普通成员
int j = 0;
}
class MainTest {
public static void main(String[] args) {
Outter outter = new Outter();
Outter.Inner inner = outter.new Inner();
inner.innerTest();
}
}
```

在外部类之外访问内部类时需要注意权限修饰符的限制，这点和类成员一样。

3、内部类编译后生成的.class文件名称格式是<外部类类名>$<内部类类名>。

4、内部类与外部类之间的成员互相访问

内部类可以访问外部类的任何成员，包括private成员。

外部类访问内部类的成员需要创建内部类的对象，之后可以访问内部类的任何成员，包括private成员，需要注意的是成员内部类不可以有静态成员。

当外部类的成员和内部类的成员重名时单单用this是区分不了的。在内部类中访问外部类的成员时可以用如下语法区分

<外部类类名>.this.<外部类中需要被访问的成员名>;

局部内部类

内部类定义在方法中成为局部内部类，只在局部有效。该类只为所在的方法块服务。

局部内部类和成员内部类一样可以访问外围类的所有成员，但是不可以访问同在一个局部块的普通局部变量。如果要访问，此局部变量要被声明称final的。
代码块结束后普通的局部变量会消亡，而创建的局部内部类对象并不会随着语句块的结束而消亡。final的局部变量的存储方式和普通的局部变量不同，其不会因为语句块的结束而消失，还会长期存在，因此可以被局部内部类访问。
例子：

```
public class InnerClassTest1 {
public static void main(String[] args) {
Outter outter = new Outter();
ForInner forInner = outter.getInner();
forInner.sayHello();

}
}
class Outter {
ForInner forInner;
public ForInner getInner() {
class Inner implements ForInner {
public void sayHello() {
System.out.println("你好，我是局部内部类对象，我还存在！");
}
}
forInner = new Inner();
return forInner;

}
}
interface ForInner {
void sayHello();
}
```

由于局部内部类只在局部有效，所以不能在外面用局部内部类的引用指向局部内部类的对象，只能用局部内部类实现接口并创建局部内部类对象，在外面用接口引用指向局部内部类对象。
静态方法中的局部内部类只能访问外围类的静态成员，访问不了非静态成员。
局部内部类生成的.class文件名称是<外部类类名>$<n><内部类类名>其中n是该局部的第几个内部类

静态内部类
静态内部类的定义

```
class Outter {
static class Inner {
/****/
}
}
```

由于静态内部类是外部类的静态成员，所以静态内部类只能访问外部类的静态成员。并且创建静态内部类的对象不依赖外部类的对象。在外部类之外创建静态内部类对象的语法如下
<外部类类名>.<内部类类名> 引用变量名 = new <外部类类名>.<内部类构造器>;
例子：

```
class Outter {
static class Inner {
public void sayHello() {
System.out.println(“成功创建静态内部类对象！”);
}
```

```
}
public void getInner() {
//在外部类中创建静态内部类的对象
Inner ii = new Inner();
ii.sayHello();
}
}
public class MainTest {
public static void main(String[] args) {
//在外部类外创建静态内部类的对象
Outter.Inner i = new Outter.Inner();
i.sayHello();
//在外部类中使用静态内部类的对象
new Outter().getInner();
}
}
```

静态内部类实际上已经脱离了外部类的控制，创建对象时也不再需要外部类对象的存在，实质上只是一个放置在别的类中的普通类而已。

匿名内部类

基于继承的匿名内部类，语法如下：

```
new <匿名内部类要继承父类的对应构造器> {
//匿名内部类类体
};
```

基于实现接口的匿名内部类，语法如下：

```
new <接口名> {
//匿名内部类类体，实现接口中的所有方法
}
```

匿名内部类中使用外面的变量要被声明成final的。

匿名内部类对象初始化的代码可以写在其非静态块中

匿名内部类生成的.class文件是<外部类类名>$<n>.class n是该类的第几个匿名内部类。

各种内部类可用的修饰符

成员内部类

final、abstract、public、private、protected、static

静态内部类

final、abstract、public、private、protected

局部内部类

final、abstract

匿名内部类

不能对匿名内部类使用修饰符

内部接口

定义在类中的内部接口无论是否被static修饰都是静态成员。

内部接口声明成private的意味着只能被外部类中的某个内部类来实现。

内部接口不能扮演局部的角色，否则编译报错。因为接口的设计初衷是对外公布，让很多类实现，而局部的角色违背了接口的设计初衷。

接口中的内部接口属于接口的成员，具有接口成员的所有属性，不能用private进行修饰。

外部接口外实现内部接口的语法如下：

```
class <类名> implements <外部接口名>.<内部接口名> {
//类体
```

}

通过这个看也好
http://loving.javaeye.com/blog/190775

提问者对于答案的评价：
good

问题答案可能在这里 → 寻找更多解答

- corejava辅导（7--1）
- 【解惑】领略内部类的“内部”
- java内部类
- java内部类访问权限
- java 项目发布的问题

其他回答

它是一个单独的类，可以随意直接访问外部类的所以变量（包括private），这是通过this的关系形成的，使得内部类的对象可以随意的访问外不类中的所有成员。在内部类中访问外部类的覆盖成员可以用classname.this. 。
内部类对象的产生，
不能直接用new，先要产生一外部类对象后在引用这个对象的成员（内部类）。即在访问时要有一个指向。说明是那个类的内部类对象。所以要先产生一外部类对象 outer.inner =out.new inner();

内存图--JAVA中凡是用new产生的对象都在堆内存中，它的引用保存在栈内存中。

对象的产生--用new ， 用对象引用赋值。
方法内部的内的使用范围只能在方法体内部，
内部类可以放在函数中，条件中，语句块中。不管它嵌套多深，都可以随意访问外部类

内部的访问权限，和方法相同。private public default protected.还可以是final, abstract, static(不能在访问外部类的非静态方法和变量)
非静态的内部类中不能定义静态的变量和方法。

实例化内部类时先要实例化外部类。可以重写构造方法。利用outer.super()调用父类构造方法来建立内外类间的this关系。

example

```
class Car
{
        class wheel
       {
       }

     }
class planeWheel extends Car.wheel
```

```
  {

   planeWheel(Car car)
    {
       Car.super();
    }
  public static void main(String[] args)
  {
      Car car = new Car();
      PlaneWheel pw =new Planewheel();
  }
}
```

接口中的方法和变量都是public abstract的，实现方法时不能低于此访问权限
匿名的内部类，
lenj (初级程序员) 2008-08-04
我觉得这种问题楼主应该主动出击，这是个大范围的东西，光凭一个答案肯定楼主是得不到最满意的结果的，建议楼主看书，百度，google，相信会收到比提这个问题更好的效果！
ag_sherry (中级程序员) 2008-08-04
Google了一下,看到这个文章貌似写得还不错的说.
小小的推荐一下:http://www.blogjava.net/raylong1982/archive/2007/10/24/155439.html
ham (架构师) 2008-08-04
java核心技术(第7版)216页---228页
小虫1313 (初级程序员) 2008-08-04
我感觉一楼二楼说的很对。。
musiclee (初级程序员) 2008-08-05
Java内部类和静态内部类的调用方式
内部类

```
public class Test {

class A{
public void setA(){

}
}

public static void main(String[] args){
Test t=new Test();

}
}
```

调用方式：

```
public class Test2 {

public static void main(String[] args){
Test test=new Test();
Test.A t=test.new A();
t.setA();
}
```

```
}

静态内部类
调用静态内部类的非静态方法：
public class Test {

static class A{
public void setA(){

}
}
}

public class Test2 {

public static void main(String[] args){
Test.A a=new Test.A();
a.setA();
}
}

调用静态内部类的静态方法：
public class Test {

static class A{
static public void setA(){

}
}
}

public class Test2 {

public static void main(String[] args){
Test.A.setA();
}
}

new Outer.Inner(); // 可以
new Inner(); // 在Outer类内部可以
new foo.Outer.Inner(); // 在包外做内部类实例化, 或者先导包再像第一个那样写.
```
miky (初级程序员) 2008-08-05
引用

我觉得这种问题楼主应该主动出击，这是个大范围的东西，光凭一个答案肯定楼主是得不到最满意的结果的，建议楼主看书，百度，google，相信会收到比提这个问题更好的效果！
ag_sherry (中级程序员) 2008-08-04

同意这个观点，目的精确的时候收获会更大，所以老师一直都要求要预习（扯远了 ）
astroxl (初级程序员) 2008-08-05
```
```

引用
我觉得这种问题楼主应该主动出击，这是个大范围的东西，光凭一个答案肯定楼主是得不到最满意的结果的，建议楼主看书，百度，google，相信会收到比提这个问题更好的效果！

我比较赞同，楼上其他朋友的答案说不定也是Google搜索的结果，建议自己先翻翻《Thinking in Java》，然后再把自己关于内部类的想法或是疑问发到坛子里讨论。
xml (初级程序员) 2008-08-06
觉得内部类主要是为了那种一次性问题来的，当然不排除为了解决多继承而使用内部类
laorer (中级程序员) 2008-08-06
我支持上面的miky
Java2008gjy (初级程序员) 2008-08-06
主要是限制外部类对这些内部功能的访问权限而编写内部类的。
mmikey (初级程序员) 2008-08-07
内部类就是定义在其他类中的类。比如：

```
public class outside
{
int d;
.....
public void f()
{
.....
}
class in
{
public void fIn()
{
data++;
f();
.....
}
.....
}
}
```

内部类可以直接用其外部类的属性和方法，所以不用把data和f（ ）的引用传给内部类。内部类主要是支持其外部类的工作的。

我觉得可以先看一些浅显的书，我看的是java语言程序设计 机械工业 Y.Daniel Liang 讲得蛮清楚的
zqjshiyingxiong (初级程序员) 2008-08-07
待解决问题数: 380
已解决问题数: 15964
已关闭问题数: 38486

问答分类

Java编程和Java企业应用

企业应用 Java综合 Struts Hibernate Spring Tomcat OO Swing 设计模式 JBoss SOA DAO iBATIS 领域模型

Web前端技术

EXT JavaScript Web综合 JQuery AJAX DWR CSS GWT UI dojo prototype JavaFX YUI

移动编程和手机应用开发

Android J2ME iPhone WAP 移动综合 Symbian BlackBerry Maemo

C/C++编程

C++ C D语言

Ruby编程

rails ruby

Python编程

python django GAE

PHP编程

PHP

Flash编程和RIA

Flex Flash ActionScript AIR

Microsoft .Net

C# .net ASP.net Windows Windows Mobile WPF SilverLight

综合技术

Database Linux oracle 编程综合 数据结构和算法 mysql SQLServer Erlang DB2 OS Unix FP PostgreSQL MacOSX Haskell

入门讨论

入门技术 IT厂商

软件开发和项目管理

项目管理 配置管理 软件测试 敏捷开发 UML 单元测试 UseCase TDD CMM XP UP

行业应用

互联网 电信 咨询 网络应用 金融 电子政务 教育 物流 嵌入式 搜索引擎 制造 SAAS 医疗 网游 浏览器 交通

招聘求职

职场话题 求职经验 企业点评 招聘职位 面试秘籍

海阔天空

工作 生活 JavaEye IT八卦 公告 读书 IT资讯 情感 游戏 大众软件 申诉 大众硬件 笑话 理财 活动 体育 影视 旅游

答题高手

- lovewhzlq CTO (1252 - 13930)
- yourgame CTO (570 - 5183)
- 蔡华江 资深架构师 (317 - 3287)
- xiaolongfeixiang 资深架构师 (301 - 2994)
- atian25 架构师 (252 - 2304)
- Anddy 架构师 (193 - 2155)
- lizhi92574 架构师 (187 - 2015)
- RednaxelaFX 架构师 (181 - 1844)
- 7454103 架构师 (179 - 2329)
- playfish 架构师 (176 - 1846)

查看全部排名>>

问题频道帮助

- 提问的智慧
- 如何获得积分及等级说明
- 如何关闭问题
- RSS

- 首页
- 资讯
- 论坛
- 问答
- 专栏
- 博客
- 圈子
- 招聘
- 服务
- 搜索

- Java
- Web
- Ruby
- Python