[Package（包） 2](#_Toc71224333)

[1. 包之间的访问 2](#_Toc71224334)

[2. 包之间的继承&protected 4](#_Toc71224335)

[3. import关键字 4](#_Toc71224336)

[4. jar包——java压缩包 5](#_Toc71224337)

# Package（包）

为什么要用包？

我们刚开始学习java时，编写的.java源程序编译后会生成.class文件，这些.java和.class文件默认会在同一个文件夹下，久而久之，这个文件夹下就会有很多的.java和.class文件，为了方便进行管理，我们会将它们分开放在不同的文件夹下，这就是包的由来。其实，包就是我们创建的文件夹。

包的作用是：

1. 对类文件进行分类管理
2. 给类提供多层命名空间：这种用法是比如我有两个Test.java文件，它们时不能放在同一个文件夹下的，但是通过创建多层文件夹，我们可以把它们分开存放。

包的定义：

1. 包必须定义在代码的第一行：package mypack;
2. 当你定义了报名后，.class文件会被放在指定的包下，所以你执行时的类名就是 包名.类名。

## 1. 包之间的访问

不同的包之间可以进行访问，但是可能会遇到一些问题，那么我们来用一个例子试一下，注意，这个例子不借助任何开发软件如Eclipse或者Intelij IDEA，而是用命令行来测试。

1. 打开命令行，进入任何一个你想用的文件夹，如d: 进入d 盘，

* 你可以选择命令行创建文件夹和文件：

mkdir testjava创建文件夹，然后cd testjava进入该文件夹。

* 也可以选择直接打开D盘然后新建文件夹和文件。

1. 我们在testjava文件夹下创建一个java文件：DemoA.java:

package packa;

class DemoA{

public void show(){

System.out.println(“hello package!”);

}

}

我们用下面的命令它编译到指定的路径下，注意，这个路径必须已经存在，如：

javac -d C:\javaapp DemoA.java

发现报错：javac: directory not found: C:\javaapp

这是因为C:\javaapp这个目录不是系统认定的包目录，所以我们可以临时设置classpath

路径：

set classpath= C:\javaapp

然后重复刚刚的编译命令就成功了：javac -d C:\javaapp DemoA.java

这时，在C盘的javaapp文件夹下会有一个packa文件夹，里面有一个DemoA.class文件

1. 然后我们在testjava文件夹下创建另一个类：PackageDemo.java ，让它处于另一个包里面，并在这个类里访问DemoA，要怎么做？

我们知道，有了包以后，类名全称是包名.类名。所以PackageDemo.java应该是：

package mypack;

class PackageDemo{

public static void main(String[] args){

packa.DemoA a = new packa.DemoA();

a.show();

}

}

然后我们对这个文件进行编译：

javac -d C:\javaapp PackageDemo.java

发现报错：

PackageDemo.java:5: error: package packa does not exist

packa.DemoA a = new packa.DemoA();

错误信息是packa包不存在，编译是在当前目录下编译的，但是packa包在c:\javaapp下

，所以设置临时classpath： set classpath=c:\javaapp，然后重复编译命令。

1. 编译后发现又报错：

PackageDemo.java:5: error: DemoA is not public in packa; cannot be accessed from outside package

packa.DemoA a = new packa.DemoA();

这个错误信息是DemoA不是公共的类，所以不能从外部访问到。这里的意思是一个包中的类如果没有用public修饰，那么它就只能在这个包内被访问。

那么我们就要修改DemoA类的修饰符为public：

package packa;

public class DemoA{

public void show(){

System.out.println(“hello package!”);

}

}

然后重新编译DemoA：javac -d c:\javaapp DemoA.java，成功。

再编译PackageDemo: javac -d c:\javaapp PackageDemo.java， 成功，此时会在c盘的javaapp文件夹下有一个新的文件夹mypack，里面有一个PackageDemo.class文件。

1. 然后运行PackageDemo，注意有了包以后，运行时类名全称是包名.类名：

java mypack.PackageDemo

这时就可以在命令行看到打印的信息了。

## 2. 包之间的继承&protected

不同包之间的类，如果有继承关系，只需要记得extends 时父类名全称必须带包名即可。

这里，我们要引入一个新的权限修饰符：protected

它是父类提供给子类的一种保护权限，意思是用protected修饰的成员，包括变量和方法，只有它的子类才可以访问。

权限修饰符总结：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 权限修饰符 | 同一类中访问 | 同一包中非子类访问 | 子类中访问 | 不同包中非子类访问 |
| public | yes | yes | Yes | yes |
| protected | yes | yes | yes | no |
| default | Yes | yes | no | no |
| private | yes | no | no | no |

## 3. import关键字

我们之前的实例中，当你需要访问别的包中的类时，必须使用类的全名称，也就是包名.类名，但是这样其实很麻烦，如果你的命名层次很多的话，代码看起来会很繁琐。所以java提供了一个关键字import来导入别的包中的类：

如：import packa.DemoA;

那么在PackageDemo类中，你就可以直接引用DemoA了，而不需要再写packa.DemoA.

如果packa这个包中所有的类你都要引用，那么你可以把每一个类都import，或者可以简化成：import packa.\*；

这里的\*是通配符，意思是所有东西。

但是在实际开发当中，最好是用单个导入，不要用通配符，因为单个导入直接就知道在哪个包里，如果使用通配符，导入多个包的内容时，效率比较慢。而且如果两个包中有同名类的话，会导致问题。

Java 本身也提供了很多软件包，当导入这些包时就可以使用里面的内容。

## 4. jar包——java压缩包

我们知道windows系统压缩文件夹或者文件时常用的是WinRAR或者是winzip。压缩后就变成了一个比较小的压缩文件。

那么java中也有类似的压缩工具jar.exe（它在jdk文件夹下的bin文件夹内），它可以把一个存储了.class文件的文件夹压缩成.jar文件。详细命令可以查看文档。

Jar文件最强大的地方在于，可以直接运行它下面的类，前提是你需要把jar包所在的目录配置到classpath下，然后就可以import 其中的类，或者java 类名来运行其中的类。

除了java自己已经有了软件包以外，我们还经常需要用到一些互联网上别人写好的jar包，它们都有对应的文档，以后真正做项目时会对这点有深刻的了解。

jar 命令格式：

jar {ctxui} [vfm0Me] [jar-file] [manifest-file] [entry-point] [-C dir] files…

选项包括：

-c 创建新的jar包（jar包指一个jar文件，如：main.jar）

-t 列出jar包中的目录

-x 解压缩jar包

-u 更新现有的jar包

-v 在标准输出中生成详细输出

-f 指定jar包的名称

-m 指定清单文件

-e 为可执行的jar包指定应用程序入口

-0 仅存储，不使用任何zip压缩

-M 不创建条目的清单文件

-i 为指定的jar文件生成索引信息

-C 更改为指定的目录并包含其中的文件

示例一：将两个字节码文件归档到一个名为main.jar的归档文件中：

jar cvf main.jar Foo.class Bar.class

示例二：使用现有的清单文件 "mymanifest.mf" 并将 foo/ 目录中的所有文件归档到 "main.jar" 中：

jar cvfm main.jar mymainifest.mf -C foo/