Java 接口 Java Enumeration接口

Java 包(package)

为了更好地组织类，Java 提供了包机制，用于区别类名的命名空间。

包的作用

- 1、把功能相似或相关的类或接口组织在同一个包中，方便类的查找和使用。
- 2、如同文件夹一样，包也采用了树形目录的存储方式。同一个包中的类名字是不同的，不同的包中的类的名字是可以相同的，当同时调用两个不同包中相同类名的类时，应该加上包名加以区别。因此，包可以避免名字冲突。
- 3、包也限定了访问权限，拥有包访问权限的类才能访问某个包中的类。

Java 使用包（package）这种机制是为了防止命名冲突，访问控制，提供搜索和定位类（class）、接口、枚举（enumerations）和注释（annotation）等。

包语句的语法格式为：

```
package pkg1[. pkg2[. pkg3…]];
```

例如,一个Something.java 文件它的内容

```
package net.java.util;
public class Something{
...
}
```

那么它的路径应该是 **net/java/util/Something.java** 这样保存的。 package(包) 的作用是把不同的 java 程序分类保存，更方便的被其他 java 程序调用。

一个包（package）可以定义为一组相互联系的类型（类、接口、枚举和注释），为这些类型提供访问保护和命名空间管理的功能。

以下是一些 Java 中的包：

- **java.lang**-打包基础的类
- **java.io**-包含输入输出功能的函数

开发者可以自己把一组类和接口等打包，并定义自己的包。而且在实际开发中这样做是值得提倡的，当你自己完成类的实现之后，将相关的类分组，可以让其他的编程者更容易地确定哪些类、接口、枚举和注释等是相关的。

由于包创建了新的命名空间（namespace），所以不会跟其他包中的任何名字产生命名冲突。使用包这种机制，更容易实现访问控制，并且让定位相关类更加简单。

创建包

创建包的时候，你需要为这个包取一个合适的名字。之后，如果其他的一个源文件包含了这个包提供的类、接口、枚举或者注释类型的时候，都必须将这个包的声明放在这个源文件的开头。

包声明应该在源文件的第一行，每个源文件只能有一个包声明，这个文件中的每个类型都应用于它。

如果一个源文件中没有使用包声明，那么其中的类，函数，枚举，注释等将被放在一个无名的包（unnamed package）中。

例子

让我们来看一个例子，这个例子创建了一个叫做animals的包。通常使用小写的字母来命名避免与类、接口名字的冲突。

在 animals 包中加入一个接口（interface）：

Animal.java 文件代码：

```
/* 文件名: Animal.java */
package animals;
interface Animal {
public void eat();
public void travel();
}
```

接下来，在同一个包中加入该接口的实现：

MammalInt.java 文件代码：

```
package animals;
/* 文件名 : MammalInt.java */
public class MammalInt implements Animal{
public void eat(){
System.out.println("Mammal eats");
}
public void travel(){
System.out.println("Mammal travels");
}
public int noOfLegs(){
return 0;
}
public static void main(String args[]){
MammalInt m = new MammalInt();
m.eat();
m.travel();
}
}
```

然后，编译这两个文件，并把他们放在一个叫做animals的子目录中。 用下面的命令来运行：

```
$ mkdir animals
$ cp Animal.class  MammalInt.class animals
$ java animals/MammalInt
Mammal eats
Mammal travel
```

import 关键字

为了能够使用某一个包的成员，我们需要在 Java 程序中明确导入该包。使用 "import" 语句可完成此功能。

在 java 源文件中 import 语句应位于 package 语句之后，所有类的定义之前，可以没有，也可以有多条，其语法格式为：

```
import package1[.package2…].(classname|*);
```

如果在一个包中，一个类想要使用本包中的另一个类，那么该包名可以省略。

例子

下面的 payroll 包已经包含了 Employee 类，接下来向 payroll 包中添加一个 Boss 类。Boss 类引用 Employee 类的时候可以不用使用 payroll 前缀，Boss类的实例如下。

Boss.java 文件代码：

```
package payroll;
public class Boss
{
public void payEmployee(Employee e)
{
e.mailCheck();
}
}
```

如果 Boss 类不在 payroll 包中又会怎样？Boss 类必须使用下面几种方法之一来引用其他包中的类。

使用类全名描述，例如：

```
payroll.Employee
```

用 **import** 关键字引入，使用通配符 "*"

```
import payroll.*;
```

使用 **import** 关键字引入 Employee 类:

```
import payroll.Employee;
```

注意：

类文件中可以包含任意数量的 import 声明。import 声明必须在包声明之后，类声明之前。

---

package 的目录结构

类放在包中会有两种主要的结果：

- 包名成为类名的一部分，正如我们前面讨论的一样。
- 包名必须与相应的字节码所在的目录结构相吻合。

下面是管理你自己 java 中文件的一种简单方式：

将类、接口等类型的源码放在一个文本中，这个文件的名字就是这个类型的名字，并以.java作为扩展名。例如：

```
// 文件名 : Car.java
package vehicle;
public class Car {
```

```
// 类实现
}
```

接下来，把源文件放在一个目录中，这个目录要对应类所在包的名字。

```
....\vehicle\Car.java
```

现在，正确的类名和路径将会是如下样子：

- 类名 -> vehicle.Car
- 路径名 -> vehicle\Car.java (在 windows 系统中)

通常，一个公司使用它互联网域名的颠倒形式来作为它的包名.例如：互联网域名是 runoob.com，所有的包名都以 com.runoob 开头。包名中的每一个部分对应一个子目录。

例如：有一个 **com.runoob.test** 的包，这个包包含一个叫做 Runoob.java 的源文件，那么相应的，应该有如下面的一连串子目录：

```
....\com\runoob\test\Runoob.java
```

编译的时候，编译器为包中定义的每个类、接口等类型各创建一个不同的输出文件，输出文件的名字就是这个类型的名字，并加上 .class 作为扩展后缀。 例如：

```
// 文件名: Runoob.java
package com.runoob.test;
public class Runoob {
}
class Google {
}
```

现在，我们用-d选项来编译这个文件，如下：

```
$javac -d . Runoob.java
```

这样会像下面这样放置编译了的文件：

```
.\com\runoob\test\Runoob.class
.\com\runoob\test\Google.class
```

你可以像下面这样来导入所有 **\com\runoob\test** 中定义的类、接口等：

```
import com.runoob.test.*;
```

编译之后的 .class 文件应该和 .java 源文件一样，它们放置的目录应该跟包的名字对应起来。但是，并不要求 .class 文件的路径跟相应的 .java 的路径一样。你可以分开来安排源码和类的目录。

```
<path-one>\sources\com\runoob\test\Runoob.java
<path-two>\classes\com\runoob\test\Google.class
```

这样，你可以将你的类目录分享给其他的编程人员，而不用透露自己的源码。用这种方法管理源码和类文件可以让编译器和java 虚拟机（JVM）可以找到你程序中使用的所有类型。

类目录的绝对路径叫做 **class path**。设置在系统变量 **CLASSPATH** 中。编译器和 java 虚拟机通过将 package 名字加到 class path 后来构造 .class 文件的路径。

<path- two>\classes 是 class path，package 名字是 com.runoob.test,而编译器和 JVM 会在 <path-two>\classes\com\runoob\test 中找 .class 文件。

一个 class path 可能会包含好几个路径，多路径应该用分隔符分开。默认情况下，编译器和 JVM 查找当前目录。JAR 文件按包含 Java 平台相关的类，所以他们的目录默认放在了 class path 中。

设置 CLASSPATH 系统变量

用下面的命令显示当前的CLASSPATH变量：

- Windows 平台（DOS 命令行下）：C:\> set CLASSPATH
- UNIX 平台（Bourne shell 下）：# echo $CLASSPATH

删除当前CLASSPATH变量内容：

- Windows 平台（DOS 命令行下）：C:\> set CLASSPATH=
- UNIX 平台（Bourne shell 下）：# unset CLASSPATH; export CLASSPATH

设置CLASSPATH变量:

- Windows 平台（DOS 命令行下）： C:\> set CLASSPATH=C:\users\jack\java\classes
- UNIX 平台（Bourne shell 下）：# CLASSPATH=/home/jack/java/classes; export CLASSPATH

← Java 接口 Java Enumeration接口 →

运行：编译之后会自己生成文件夹，不要进入这个文件夹，直接运行 `java -cp /home/test test.Run`，其中源文件在 test 文件夹中，包名为 test，启动文件为 **Run.java。**

更多内容参考：Java 中带包（创建及引用）的类的编译与调试

Anne 6个月前 (09-05)