Java Programming-III(2)

By:- Jagdish Chandra Pandey

H.O.D(I.T.), G.P. Dwarahat/Officiating Principal, G.P.Chaunaliya

Java Packages

- In java a unique name had to be used for each class to avoid name collisions but after a while without some way to manage the name space, we could runout of convenient, descriptive names for individual classes. Java provides a mechanism called Package for partioning the class-name space into more manageable chunks.

- The Package is both a naming and a visibility control mechanism as we can define classes inside a package that are not accessible by code outside that package.

Java Packages

- A package in Java is used to group related classes. Think of it as **a folder in a file directory**. We use packages to avoid name conflicts, and to write a better maintainable code. Packages are divided into two categories:

1. Built-in Packages (packages from the Java API)

2. User-defined Packages (create your own packages)

Built-in Packages(Application Programming Interface)

- The Java API is a library of prewritten classes, that are free to use, included in the Java Development Environment.

- The library contains components for managing input, database programming, and much much more. The complete list can be found at Oracles website: https://docs.oracle.com/javase/8/docs/api/.

- The library is divided into **packages** and **classes**. Meaning you can either import a single class (along with its methods and attributes), or a whole package that contain all the classes that belong to the specified package.

- To use a class or a package from the library, you need to use the import keyword:-

Syntax:-

import *package.name.Class*; // Import a single class import *package.name*.*; // Import the whole package

Import a Class

- If you find a class you want to use, for example, the Scanner class, **which is used to get user input**, write the following code:

Example:-

```
import java.util.Scanner;
```

In the example above, java.util is a package, while Scanner is a class of the java.util package.

- To use the Scanner class, create an object of the class and use any of the available methods found in the Scanner class documentation. In our example, we will use the nextLine() method, which is used to read a complete line:-

```
import java.util.Scanner;
class MyClass
  {
      public static void main(String[] args)
       {
         Scanner myObj = new Scanner(System.in);
          System.out.println("Enter username");
          String userName = myObj.nextLine();
           System.out.println("Username is: " + userName);
           }
       }
```

- To import a whole package, end the sentence with an asterisk sign (*). The following example will import ALL the classes in the java.util package:

```
import java.util.*;
```

User-defined Packages

- To create a package, use the package keyword:-

```
package mypack;
class MyPackageClass
{
 public static void main(String[] args)
 {
  System.out.println("This is my package!");
 }
}
```

Java uses a file system directory to store them. Just like folders on your computer:-

```
└── root
    └── mypack
        └── MyPackageClass.java
```

Save the file as **MyPackageClass.java**, and compile it:

C:\Users*Your Name*>javac MyPackageClass.java

Then compile the package:

C:\Users*Your Name*>javac -d . MyPackageClass.java

This forces the compiler to create the "mypack" package.

The -d keyword specifies the destination for where to save the class file. You can use any directory name, like c:/user (windows), or, if you want to keep the package within the same directory, you can use the dot sign ".", like in the example above. When we compiled the package in the example above, a new folder was created, called "mypack".

To run the **MyPackageClass.java** file, write the following:

C:\Users*Your Name*>java mypack.MyPackageClass

The output will be:

This is my package!

User-defined Packages

- Include a Package command as the first statement in a java source file. Any classes within that file will belong to the specified package.
- The Package statement defines a name space in which classes are stored. If we omit the package statement, the class names are put into the default Package, which has no – name.
- We can create hierarchy of Packages, for doing this simply separate each Package name from one above it by use of a period”.”. General form for this is:-

package pkg1.pkg2.pkg3;

for ex:- a package declared as:-

package java.awt.image

need to be stored in java\awt\image in a windows environment.

Finding Packages and classpath

- For looking for packages that we create java run time by default, uses the current working directory as its starting point.

- We can specify a directory path or paths by setting CLASSPATH environmental variable. or you can use –classpath option with java and javac to specify the path to your classes. For ex:- in order for a program to find package Mypack one of 3 things must be true:-

1. Either the program can be executed from a directory immediately above Mypack.

2. or the CLASSPATH must be set to include the path to Mypack

3. or the –classpath option must specify the path to Mypack when the program is run via java

- When the second two options are used, the class path must not include Mypack, itself. It must specify the path to Mypack. For ex:- in a windows environment if the path to Mypack is:-

  C:\Myprogram\java\Mypack

  Then the classpath to Mypack is:-

  C:\Myprogram\java

- The easiest way is to create the package derectories below our current development directory, put the class files into the appropriate directories, and then execute the programs from the development directory.

A short package example

```
Package Mypack;
class Balance
{
string name;
double bal;
Balance(string n, double b)
{
name = n;
bal = b;
}
void show()
{
If(bal<0)
    System.out.print("→");
System.out.println(name + ":$"+ bal);
}
}
class AccountBalance
{
public static void main(String args[])
   Balance current[] = new Balance[3];
   current[0] = new Balance("K.J.Fielding",123.23);
   current[1] = new Balance("will tell", 157.02);
    current[2] = new Balance("Tom Jackson", -12.33);
  for(int i = 0; i<3; i++)
     current[i].show();
}
}
```

Call this file AccountBalance.java and put it in a directory called Mypack. Next compile the file. Make sure that resulting .class file is also in the Mypack directory. Then try executing the AccountBalance class, using following command line:-

java Mypack.AccountBalance

You will need to be in the directory above Mypack when you execute this command.

Access protection

- Anything declared public can be accessed from anywhere. Anything declared private cannot be seen outside of its class.

- When a member does not have an explicit access specification, it is visible to subclasses as well as to other classes in the same package. This is default access.

- If you want to allow an element to be seen outside your current package, but only to classes that subclass your class directly, the declare that element protected.

- A class has only two possible access levels:- Default and Public. When a class is declared as public it is accessible by any other code. If a class has default access, then it can only be accessed by other code within same package.

An Access Example

This is file protection.java

```
package p1;
public class protection
{
int n =1;
private int n_pri = 2;
protected int n_pro = 3;
public int n_pub = 4;
public protection()
{
System.out.println("base constructor");
System.out.println("n = "+n);
System.out.println("n_pri = " + n_pri);
System.out.println("n_pro = " + n_pro);
System.out.println("n_pub = " + n_pub);
}
}
```

This is file derived.java

```
package p1;
class derived extends protection
{
derived()
{
System.out.println("derived constructor");
System.out.println("n = " +n);
//class only
// System.out.println("n_pri = " +n_pri);
System.out.println("n_pro = " +n_pro);
System.out.println("n_pub = " +n_pub);
}
}
```

This is file samepackage.java

```
package p1;
class samepackage
{
samepackage()
{
protection p = new protection();
System.out.println("Same package constructor");
System.out.println("n = " + p.n);
//class only
//System.out.println("n_pri = +p.n_pri);
System.out.println("n_pro = " + p.n_pro);
System.out.println("n_pub = " +p.n_pub);
}
}
```

This is file protection2.java

```
package p2;
class protection2 extends p1.protection
{
protection2
{
System.out.println("derived other package constructor");
//class or package only
//System.out.println("n = "+n);
//class only
//system.out.println("n_pri = " +n_pri);
System.out.println("n_pro = "+n_pro);
System.out.println("n_pub = " +n_pub);
}
}
```

Continued on next page.....

An Access Example

This is file otherpackage.java

```
package p2;
class otherpackage
{
otherpackage()
{
p1.protection p = new p1.protection();
System.out.println("other package constructor");
//class or package only
//System.out.println("n = "+p.n);
//class only
//System.out.println("n_pri = " +p.n_pri);
//class,subclass or package only
//System.out.println("n_pro = "  +p.n_pro);
System.out.println("n_pub = " +p.n_pub);
}
}
```

Test file for package p1

```
//Demo package p1
package p1;
public class demo
{
public static void main(string args[])
{
protection ob1 = new protection();
derived ob2 = new derived();
samepackage ob3 = new samepackage();
}
}
```

Test file for package p2

```
package p2;
public class demo
{
public class demo
{
public static void main(String args[])
{
protection2 ob1 = new protection2();
otherpackage ob2 = new otherpackage();
}
}
```

Continued on next page….

An Access Example

- This example has two packages. The classes for the two different packages need to be stored in directories named after their respective packages- in this case p1 and p2.

- The source for the first package defines 3 classes:- protection, derived and same package.

- The second class derived is a subclass of protection in the same package p1, this grants derived access to every variable in protection except for n_pri, the private one. The third class same package is not a subclass of protection, but is in the same package, also has access to all but n_pri.

- In other package p2 two classes are defined. The first class protection2 is a subclass of p1.protection. This grants access to all of p1.protection's variables except for n_pri and n, the variable declared with the default protection. The default only allows access from within class or the package, not extra-package subclasses. Finally the class otherpackage has access to only one variable, n_pub which was declared public.

Access protection

Class type	Private	No Modifier (Default access)	Protected	Public
Same class	Yes	Yes	Yes	Yes
Same package class	No	Yes	Yes	Yes
Same package non-subclass	No	Yes	Yes	Yes
Different package subclass	No	No	Yes	Yes
Different package non-subclass	No	No	No	Yes


"THANK
YOU"