**Packages: Putting Classes Together**

**Introduction**

A package can be defined as a group of similar types of classes, interface, enumeration and sub-package. Using package it becomes easier to locate the related classes.

Package in java can be categorized in two form, built-in package and user-defined package. There are many built-in packages such as java, lang, awt, net, io, util etc.,. Here, we will have the detailed learning of creating and using user-defined packages.

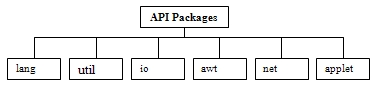
**Advantage of Java Package**

Before discussing how to use them Let see why we should use packages.

* Reusability:  Reusability of code is one of the most important requirements in the software industry. Reusability saves time, effort and also ensures consistency. A class once developed can be reused by any number of programs wishing to incorporate the class in that particular program.
* Easy to locate the files.
* In real life situation there may arise scenarios where we need to define files of the same name. This may lead to “name-space collisions”. Packages are a way of avoiding “name-space collisions”.

**Java API packages**

Java API provides a large number of classes grouped into different packages according to functionality. Most of the time we use the packages available with the Java API.



|  |  |
| --- | --- |
| Package name | Contents |
| java.lang | Language support classes that java compiler itself uses and therefore they are automatically imported. They include classes for primitive types, strings, math functions, threads, exceptions etc., |
| java.util | Language utility classes such as vectors, hash tables, random numbers etc., |
| java.io | Input /output support classes. |
| java.awt | Set of classes for implementing graphical user interface. They include classes for windows, buttons, lists, menus and so on. |
| java.net | Classes for networking. They include classes for communicating with local computers as well as with internet servers |
| java.applet | Classes for creating and implementing applets. |

**Using System packages**

The packages are organized in a hierarchical structure as illustrated in the following figure. This shows that the package named java contains the package **awt**. which in turn contains various classes required for implementing graphical user interface.

**java**

**awt**

**Color**

**Graphics**

**Fonts**

package containing awt package

package containing classes

classes containing methods

There are two ways of accessing the classes stored in a package. The first approach is to use the fully qualified class name of the class that we want to use. This is done by using the package name containing the class and then appending the class name to it using the dot operator. For example. If we want to refer to the class **Color** in the **awt** package, then we may do so as follows:

**java.awt.Color;**

This approach is perhaps the best and easiest one If we need to access the class only once or when we need not have to access any other classes of the package. But, in manysituations, we might want to use a class in a number of places in the program or we may like to use many of the classes contained in a package. We may achievethis easily as follows

|  |
| --- |
| import packagename.classname;  or  import packagename.\*; |

The First statement allows the specified class in the specified package to be imported. The second statement imports every class contained in the specified package.

**Naming Conventions**

Packages can be named using the standard Java naming rules. By convention, however, packages begin with lowercase letters. This makes it easy for users to distinguish package names from class names when looking at an explicit reference to a Class. We know that all class names, again by convention, begin with an uppercase letter For example, look at the following statement

**double y=java . lang . Math . sqrt(x);**

**package class method**

**name name name**

This statement uses it fully qualified class name **Math** to invoke the method **sqrt( )**. Note that methods begin with lowercase letters.

**Creating a package**

Creating a package in java is quite easy. Simply include a package keyword followed by name of the package as the first statement in java source file.

package mypack;

public class employee

{

...statement;

}

The above statement creates a package called **mypack**.

Java uses file system directory to store package. For example the .class for any classes you define to be part of **mypack** package must be stored in a directory called mypack.

**Accessing a Package**

A Java system package can be accessed either using a fully qualified class name or using a shortcut approach through the import statement, We use the import statement when there are many references to a particular package.

The same approaches can be used to access the user-defined packages as well. The import statement can be used to search a list of packages for a particular class. The general form of import statement for searching a class is as follows

|  |
| --- |
| import package1[ .package2 ][ .package3 ] . classname |

Here **package1** in the name of the top level package, **package2** is the name of the package that is inside the package1 and so on, We can have any number of packages in a package hierarchy, Finally, the explicit **classname** is specified.

We can also use another approach as follows:

import package1 . \* ;

The star (\*) indicates that we can access all classes contained in the package directly.

**Using a Package**

Let us consider a simple package with two classes..

package Arith;

public class Addition

{

public int add( int x, int y)

{

return (x+y);

}

}

Packages can be created in two Methods…

**Method 1:**

* Save the file as Addition.java
* Compile the file as javac Addition.java
  + It creates Addition.class file in the same directory.
* Create a sub directory here with the same name as package name( eg: Arith) and copy the .class file into that directory.

**Method 2:**

* Save the file as Addition.java
* Compile the file as javac –d . Addition.java
  + It creates a subfolder named as same as package name(eg: Arith) automatically, and adds Addition.class file in to that directory.

**Adding a class to the Package**

package Arith;

public class Multiplication

{

public int mul( int x, int y)

{

return (x\*y);

}

}

Follow one of the above methods to add a new class to the already existing package ( eg: Arith).

**Example program to access the above package**

import Arith.Addition;

import Arith.Multiplication;

class Test

{

public static void main(String args[ ])

{

Addition ad=new Addition( );

Multiplication ml=new Multiplication( );

int res1=ad.add(10,20);

int res2= ml.mul(10,20);

System.out.println("Addition="+res1);

System.out.println("Multiplication="+res2);

}

}

**Hiding Classes**

When we import a package using asterisk (\*), all public classes are imported. However, we may prefer to “not import" certain classes, That is, we may Like to hide these classes from accessing from outside of the package. Such classes should he declared “not public".

Example:

package p1;

public class A // public class, available outside

{

// body of A

}

class B // not public, hidden

{

// body of B

}