Package

Introduction

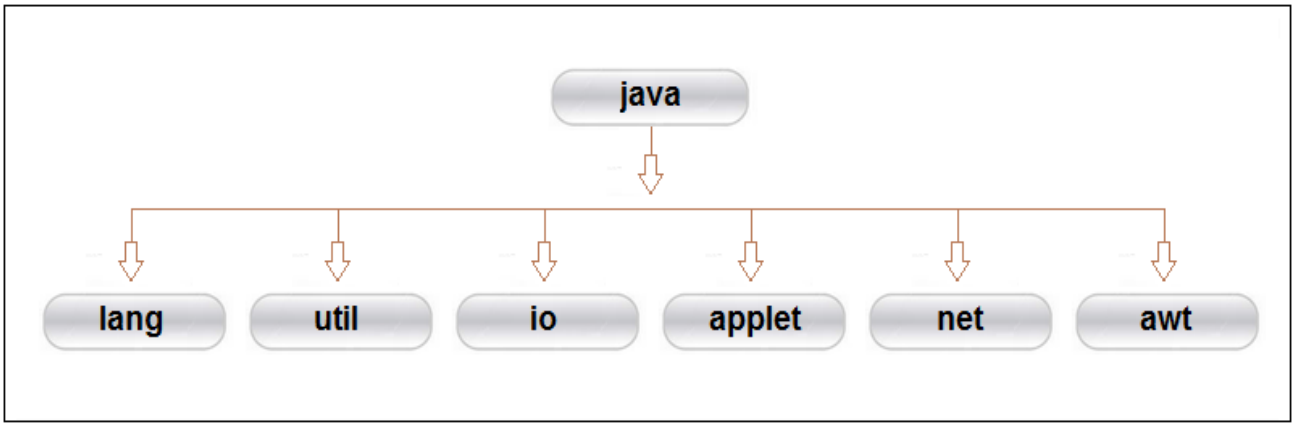
Package in Java is a mechanism to encapsulate a group of classes, sub packages and interfaces. Packages are used for:

- Preventing naming conflicts. For example there can be two classes with name Employee in two packages, college.staff.cse.Employee and college.staff.ee.Employee
- Making searching/locating and usage of classes, interfaces, enumerations and annotations easier
- Providing controlled access: protected and default have package level access control. A protected member is accessible by classes in the same package and its subclasses. A default member (without any access specifier) is accessible by classes in the same package only.
- Packages can be considered as data encapsulation (or data-hiding).

All we need to do is put related classes into packages. After that, we can simply write an import class from existing packages and use it in our program. A package is a container of a group of related classes where some of the classes are accessible are exposed and others are kept for internal purpose.
We can reuse existing classes from the packages as many time as we need it in our program.
Java Packages are classified into two types. First is known as Java API packages and second is known as user defined packsges.

JAVA API Packages

Java **APl(Application Program Interface)** provides a large numbers of classes grouped into different packages according to functionality. Most of the time we use the packages available with the the Java API. Following figure shows the system packages that are frequently used in the programs.



Java System Packages and Their Classes

java.lang	Language support classes. They include classes for primitive types, string, math functions, thread and exceptions.
java.util	Language utility classes such as vectors, hash tables, random numbers, data, etc.
java.io	Input/output support classes. They provide facilities for the input and output of data.
java.applet	Classes for creating and implementing applets.
java.net	Classes for networking. They include classes for communicating with local computers as well as with internet servers.
java.awt	Set of classes for implementing graphical user interface. They include classes for windows, buttons, lists, menus and so on.

Using System Packages

The packsges are organizes in a hierarchical structure as shown in figure below. This shows that the package named **java** contains the package **awt** , which in turn contains various classes required for implementing graphical user interface.

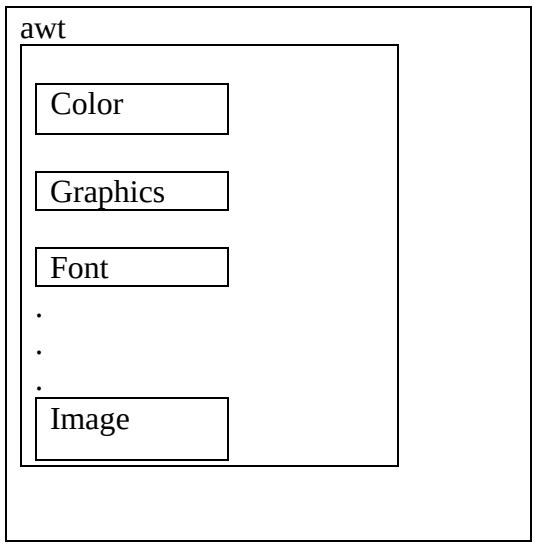


Fig. Hierarchical representation of java.awt package

There are two ways of accessing the classes stored in a package.

First approach is done by using the package name that containing class and then appending the class name to it using the dot operator.

Java.awt.Color

Here awt is Package within the package java and Color is class name.
We use the class in number of programs or use many classes contained in the package.
Syntax.

```
import packagename.classname;
        Or
import packagename.*;
```

These are known as import statements and used at the top of the program file.

```
import java.awt.Color;
```

it import the class Color and the classname can be directly used in the program.

```
import java.awt.*;
```

It will bring all the classes of java.awt package.

Naming Conventions

Packages can be named using the standard java naming rules.
By convention , packages begin with lowercase letters.This makes it easy for users to distinguish packsge names from class names.
By convention , all class names begin with an uppercase letters.
Ex.

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

lang – package name
Math – class name
Sqrt() – method name
This statement uses a only class name Math to invoke the method sqrt().

Creating Packages

To create our own package , we must first declare the name of package using the package keyword followed by a package name. This must be first statement in a java source file.
Then we define a class , just as we normally defined a class.
Ex.

```
Package firstpackage;
Public class FirstClass
{
        …….
        ……. ( Body of class )
}
```

Here package name is firstPackage.
The class FirstClass is now considered a part of this package.

This file is saved as FirstClass.java and located in a directory named firstPackage. When the source file is compiled , java will create a .class file and store it in the same directory.

To create our own package following steps are there.

1. Declare the package at the beginning of a file using the form
   package packagename;
2. Define the class that is to be put in the package and declare it public.
3. Create a subdirectory under the directory where the main source files are stored.
4. Store the programs as classname.java file in the subdirectory created.
5. Compile the file . This creates .class file in subdirectory.

Remember that name of the package must be same as the directory under which this file is saved.

Accessing a Package

Java system packages can be accessed by using import statement.
Same approach is used to access the user defined packages as well.
Ex.

```
import packagename.*;
```

here the package name may denote a single package or a hierarchy of packages. The star (*) indicates the entire package hierarchy.

```
import firstPackage.secondPackage.MyClass ;
```

by the use of this statement ,all the members of the class MyClass can be directly accessed using the class name .

Using a Package

We consider a simple program for use classes from other packages.
The listing below shows a package named package1 containing a single ClassA.

```
package  package1 ;
public class ClassA
{
public void displayA( )
{
System.out.println(" Class A" );
}
}
```

This file ClassA.java and stored in the subdirectory package1.Now compile this java file.The resultant ClassA.class will be store in the same subdirectory.
Now consider the program

```
import package.ClassA ;
class PackageTest
```

```
{
Public static void main(String args[ ])
{
ClassA obj = new ClassA( ) ;
Obj.displayA( ) ;
}
}
```

This program that imports the class ClassA from the package package1. The file is saved as PackageTest.java and then compiled. The source file and the compiled file would be saved in the directory of which package1 was a subdirectory. Now we can run the program and obtain the results.

Adding a Class to a Package

It is simple to add a class to an existing package. Consider the following package :

```
package  p1
public Class A
{
        // body of A
}
```

The package p1 contains one public class by name A. Suppose we want to add another class B to this package. This can be done as follows :

1. Define the class and make it public.
2. Place the package statement
   ```
   Package  p1 ;
   ```
   Before the class definition as follows :
   ```
   Package p1
   Public Class B
   {
           // body of B
   }
   ```
3. Store this as B.java file under the directory p1.
4. Compile B.java file. This will create a B.class file and place it in the directory p1.
   Now , the package p1 will contain both the classes A and B . The statement like
   ```
   import p1. * ;
   ```
   will import both of them.

****