What is a Package?

A package is a namespace that organizes a set of related classes and interfaces. Conceptually you can think of packages as being similar to different folders on your computer. You might keep HTML pages in one folder, images in another, and scripts or applications in yet another. Because software written in the Java programming language can be composed of hundreds or *thousands* of individual classes, it makes sense to keep things organized by placing related classes and interfaces into packages.

The Java platform provides an enormous class library (a set of packages) suitable for use in your own applications. This library is known as the "Application Programming Interface", or "API" for short. Its packages represent the tasks most commonly associated with general-purpose programming. For example, a String object contains state and behavior for character strings; a File object allows a programmer to easily create, delete, inspect, compare, or modify a file on the filesystem; a Socket object allows for the creation and use of network sockets; various GUI objects control buttons and checkboxes and anything else related to graphical user interfaces. There are literally thousands of classes to choose from. This allows you, the programmer, to focus on the design of your particular application, rather than the infrastructure required to make it work.

The Java Platform API Specification contains the complete listing for all packages, interfaces, classes, fields, and methods supplied by the Java SE platform. Load the page in your browser and bookmark it. As a programmer, it will become your single most important piece of reference documentation.

Creating and Using Packages

To make types easier to find and use, to avoid naming conflicts, and to control access, programmers bundle groups of related types into packages.

Definition: A *package* is a grouping of related types providing access protection and name space management. Note that *types* refers to classes, interfaces, enumerations, and annotation types. Enumerations and annotation types are special kinds of classes and interfaces, respectively, so *types* are often referred to in this lesson simply as *classes and interfaces*.

The types that are part of the Java platform are members of various packages that bundle classes by function: fundamental classes are in java.lang, classes for reading and writing (input and output) are in java.io, and so on. You can put your types in packages too.

Suppose you write a group of classes that represent graphic objects, such as circles, rectangles, lines, and points. You also write an interface, Draggable, that classes implement if they can be dragged with the mouse.

```
//in the Draggable.java file
public interface Draggable {
    ...
}

//in the Graphic.java file
public abstract class Graphic {
    ...
}

//in the Circle.java file
public class Circle extends Graphic
    implements Draggable {
    . . .
}

//in the Rectangle.java file
public class Rectangle extends Graphic
    implements Draggable {
    . . .
}

//in the Point.java file
public class Point extends Graphic
    implements Draggable {
    . . .
}
```

```
//in the Line.java file
public class Line extends Graphic
    implements Draggable {
    . . .
}
```

You should bundle these classes and the interface in a package for several reasons, including the following:

- You and other programmers can easily determine that these types are related.
- You and other programmers know where to find types that can provide graphics-related functions.
- The names of your types won't conflict with the type names in other packages because the package creates a new namespace.
- You can allow types within the package to have unrestricted access to one another yet still restrict access for types outside the package.

Summary of Creating and Using Packages

To create a package for a type, put a package statement as the first statement in the source file that contains the type (class, interface, enumeration, or annotation type).

To use a public type that's in a different package, you have three choices:

(1) Use the fully qualified name of the type,

(2) Import the type, or

(3) Import the entire package of which the type is a member.

The path names for a package's source and class files mirror the name of the package.

You might have to set your CLASSPATH so that the compiler and the JVM can find the .class files for your types.

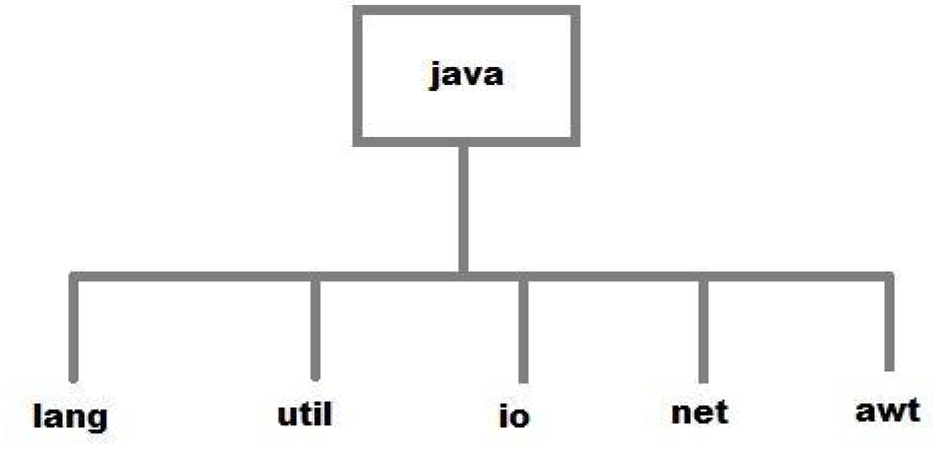
Java Package

Packages are used in Java, in-order to avoid name conflicts and to control access of class, interface and enumeration etc.

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 are categorized into two forms

- Built-in Package:-Existing Java package for example java.lang, java.util etc.
- User-defined-package:- Java package created by user to categorized classes and interface



Creating a package

Creating a package in java is quite easy. Simply include a package command 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 to define to be part of **mypack** package must be stored in a directory called mypack

Example of package creation

```
package mypack;
class Book{
 String bookname;
 String author;
 Book(String b, String c){
  this.bookname = b;
  this.author = c;
 }
 public void show() {
  System.out.println(bookname+" "+ author);
 }
}

class test{
 public static void main(String[] args) {
  Book bk = new Book("java","Herbert");
  bk.show();
 }
}
```

To run this program:

- Create a directory under your current working development directory(i.e. JDK directory), name it as **mypack**.
- compile the source file
- Put the class file into the directory you have created.
- Execute the program from development directory.

NOTE: Development directory is the directory where your JDK is install.

Uses of java package

Package is a way to organize files in java, it is used when a project consists of multiple modules. It also helps resolve naming conflicts. Package's access level also allows you to protect data from being used by the non-authorized classes.

import keyword

import keyword is used to import built-in and user-defined packages into your java source file. So that your class can refer to a class that is in another package by directly using its name.

There are 3 different ways to refer to class that is present in different package

1. **Using fully qualified name** (But this is not a good practice.)

   Example :

   ```
   class MyDate extends java.util.Date {
    //statement;
   }
   ```

2. **import the only class you want to use.**

   Example :

   ```
   import java.util.Date;
   class MyDate extends Date {
    //statement.
   }
   ```

3. **import all the classes from the particular package**

Example :

```
import java.util.*;
class MyDate extends Date {
//statement;
}
```

import statement is kept after the package statement.

Example:

```
package mypack;
import java.util.*;
```

But if you are not creating any package then import statement will be the first statement of your java source file.

Static import

static import is a feature that expands the capabilities of **import** keyword. It is used to import **static** member of a class. We all know that static member are referred in association with its class name outside the class. Using **static import**, it is possible to refer to the static member directly without its class name. There are two general form of static import statement.

- The first form of **static import** statement, import only a single static member of a class

  Syntax: import static *package.class-name.static-member-name;*

  Example

  import static java.lang.Math.sqrt;   //importing static method **sqrt** of **Math** class

- The second form of **static import** statement,imports all the static member of a class

  Syntax: import static *package.class-type-name.*;*

  Example

```
import static java.lang.Math.*;  //importing all static member of Math class
```

Example without using static import

```
public class Test{
   public static void main(String[] args)    {
      System.out.println(Math.sqrt(144));
   }
}
```

Output:

```
12
```

Example using static import

```
import static java.lang.Math.*;
public class Test {
   public static void main(String[] args)    {
      System.out.println(sqrt(144));
   }
}
```

Output:

```
12
```