Java Package

Package 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 create 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
```