JAVA PACKAGE

PACKAGE [FOLDER / CONTAINER / DIRECTORY]

- Set of classes and interfaces are grouped into a single entity is called as packages
- Packages are containers for both classes and interfaces.
- Packages are stored in a hierarchical order (tree order) and explicitly imported into class definition.
- Package statements are usually placed at the top of the program code.

TYPES OF PACKAGES

- In java, the package is classified as two types. They are
    1. System packages / Java API / Java Standard Library (JSL)
    2. User Definied Packages

1. SYSTEM PACKAGES (JSL)

- It is builtin java package
- It is used to import single class or entire package (whole classes) at a time
- It is arranged in hierarchical order (tree order)
- It is placed at the top of the program code
- The main package is "java". It contains several sub packages. These packages contain various classes. These built-in classes contain several built-in methods.
- Examples of java standard packages
  - java.util.*, java.io.*, java.net.*, java.sql.*, java.awt.*, etc, …

Default package

java.lang.*,

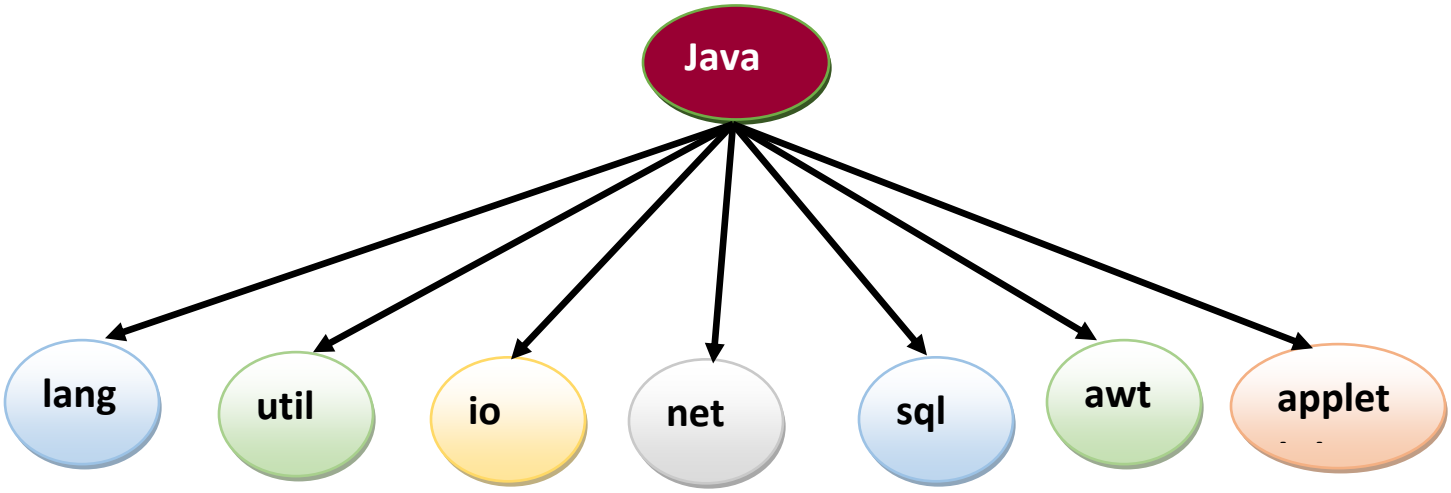
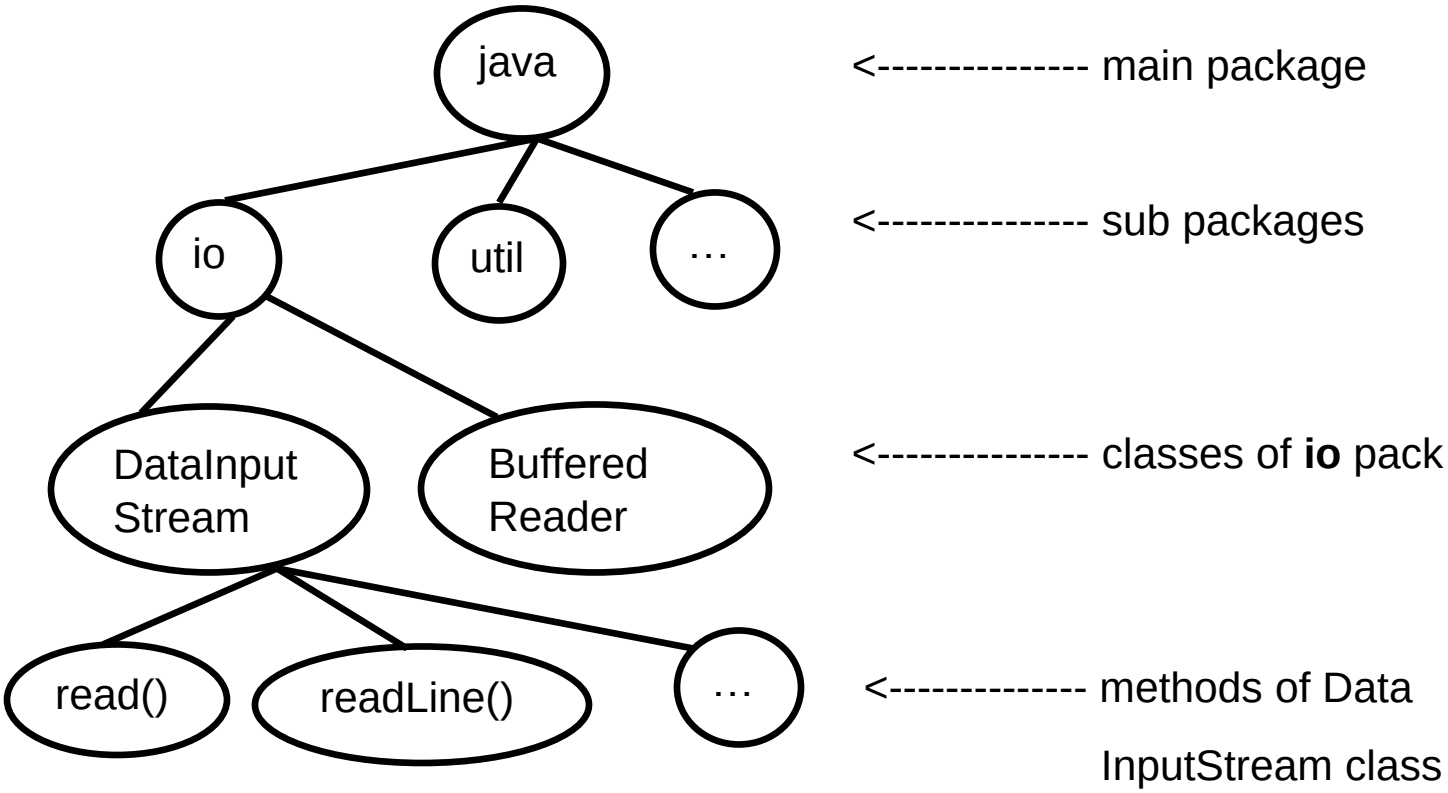


Figure. Java System Packages (JSL)

SYSTEM PACKAGES LIST

S.N	Name of the Package	Meaning / Purpose
1.	java.lang.* (language package)	- default package in java - supports the basic features of java - supports strings, threads & mathmetical operations - supports exception handling.
2.	java.io.*	- contains the classes for input and output operations - handling runtime input, file and directory related operations.
3.	java.util.* (utility package)	- supports vectors, random numbers, time & calendar. - supports data structures like hashtable, linkedlist, treeset, treemap, stack, queue, etc..
4.	java.awt.* (abstract window toolkit)	-supports for windows GUI applications.
5.	java.sql.*	- supports java database programming operations like CRUD - JDBC API(Application Programming Interface)
6.	java.net.*	- supports networking related operations
7.	java.applet.*	- supports java applet applications

Example



IMPORTING CLASSES / ACCESSING CLASSES

- We can import the class from the particular package using two ways. namely
  - (a) Full path of the class
  - (b) Using import keyword.

(a) Full path of the class

- Here, no need to import system packages (built-in packages) at the beginning of the program
- In this method, the class name and package names are explicitly given.

Syntax

```
java.<package-name>.class-name;
```

Example

```
java.io.DataInputStream ds=new java.io.DataInputStream();
```

(b) using the import keyword

- Here we can import / access a particular class or entire package using import keyword.
- The builtin package must be imported at the beginning of the program using import keyword

Syntax

```
import package-name.class-name; // import only specified class
import package-name.*;          // import all classes
```

Example

```
import java.util.*              // import all classes from util package
import java.io.Scanner;         // import only Scanner class
```

Note

- It is important to note that, the * operator will import all the the builtin classes, interfaces from the mentioned package
- But it will not import sub packages
- If we want to import the sub package, then we should mention the subpackage name in the import statement.

STATIC IMPORT

- In addition to classes, static fields and methods can also be imported from packages with help of static import.
- It eliminates the need of using class name before static members.
- Static import is similar to normal import. But it uses the static keyword after the import statement.

Example

- If we use the static import like "import static java.lang.Math.*;", the following statement is enough. (static import)

```
double rs=pow(5,2);
```

- No need to use the following statement (normal import)

```
double rs=Math.pow(5,2);
```

NOTE

- Unlike normal import, it is important to note that, the meta character "** should be used at the end of the static import statement.

Using Static Import

Calc.java

```
import static java.lang.Math.*;
public class Calc
{
        public static void main(String[] a)
        {
                double st=pow(5,3);                // without using class name
                System.out.println("Power of 5-3 : "+st);
        }
}
```

Output

```
Power of 5-3 : 125.0
```

Using Normal Import

Calc.java

```
import java.lang.Math;          // no need to use * operator here
public class Calc
{
        public static void main(String[] a)
        {
                double st=Math.pow(5,3);      // with using class name
                System.out.println("Power of 5-3 : "+st);
        }
}
```

Output

```
Power of 5-3 : 125.0
```

NAMING CONVENTIONS

- In java, first letter of the class names usually begin with an uppercase letter and the remaining charcters are written in lowercase letters.
- Package names and methods are usually written in lowercase letters to distinguish from package.
- The package, class and method can be named using the standard java naming rules.
- Package name must be unique to avoid ambiguity.

2. USER DEFINED PACKAGE

- It is created by the user using package keyword
- It can contain multiple java source files (.java)
- It is used to organize set of files

STEPS FOR CREATING USER DEFINED PACKAGE

1. Create a directory which has the same name as the package name
2. Open the editor and declare the package at the beginning of the code

Syntax

```
package package-name;
```

Example

```
package info;
```

3. Write class definitions (Creating source code)
4. Save the file (with .java extension) in the directory (current directory) as name of class.java
5. Compile this file using java compiler. After the successful compliation, class file (.byte code) will be created in the same directory

Note

- A java package can have more than one classes.
- But package allows source file with only one one public class (with .java extension) and several non public classes.

PACKAGE HIERARCHY

- It is possible to create the sub packages (package hierarchy)
- This is done by specifiying the multiple names in a main package statement. Each sub packages are separated by dot operator.

Example (System Package)

```
import java.io.*;                    // JSL
```

where,

java	← main package
io	← sub package

Example (User Defined Package (UDP))

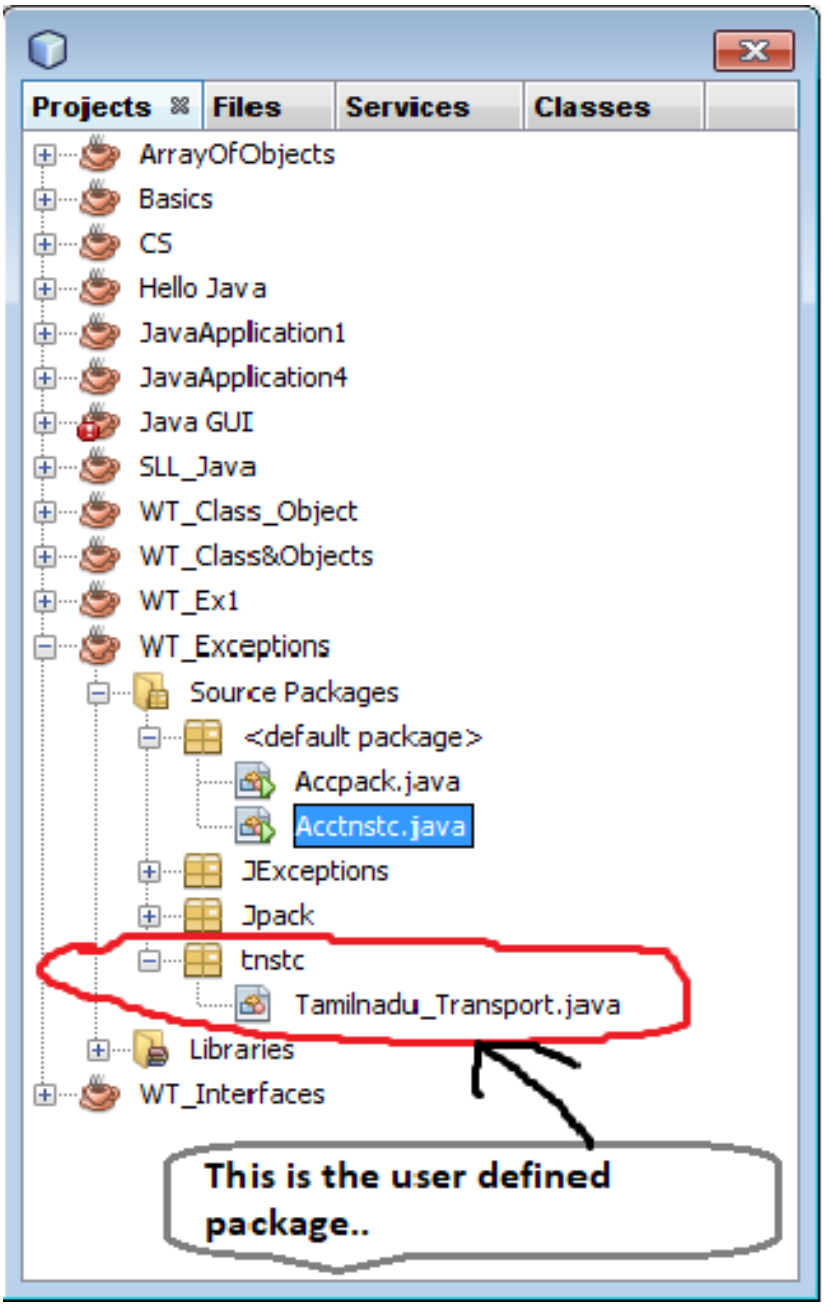
```
package first.second;          // UDP
```

where,

first	← main package
second	← sub package

I. ACCESSING SINGLE CLASS OF SAME PACKAGE

PACKAGE DETAILS



Package name	:	tnstc
No of files	:	1 (Tamilnadu_Transport.java)
No of classes	:	1 (Tamilnadu_Transport)

1. SOURCE CODE

(a) Implementation Class [Compiled Class] / Developer Source Code

(Tamilnadu_Transport.java)

```
package tnstc;
public class Tamilnadu_Transport
{

     public void tptc()      // essential information
     {
          System.out.println("Welcome to Thanthai Periyar Transport
          Corporation-Villupuram");
     }
    public void setc()      // essential information
    {
         System.out.println("Welcome to State Express Transport
         Corporation-Chennai");
     }
    public void atc()       // essential information
    {
        System.out.println("Welcome to Anna Transport Corporation-
        Salem");
    }
    public void mtc()       // essential information
    {
       System.out.println("Welcome to Pallavan Transport Corporation-
       Chennai");
    }
    public void mgr()       // essential information
    {
        System.out.println("Welcome to MGR Transport Corporation-
        Kanchipuram(Villupuram)");
    }
```

```
}
```

(b) Accessing Class [Essential Features] / Client Application

Acctnstc.java

```
// import user defined package as system package
import tnstc.*;
public class Acctnstc
{
// main method
    public static void main(String[] args)
    {
        System.out.println("==================================");
        System.out.println("\t\tAccessing TNSTC Package");
        System.out.println("==================================");

// object creation for tnstc.Tamilnadu_Transport class
        Tamilnadu_Transport obj=new Tamilnadu_Transport();
        obj.tptc();             // essential feature
        System.out.println("-------------------------------------------------------");
        obj.atc();              // essential feature
        System.out.println("-------------------------------------------------------");
        obj.mgr();              // essential feature
        System.out.println("-------------------------------------------------------");
        obj.setc();             // essential feature
        System.out.println("-------------------------------------------------------");
        obj.mtc();              // essential feature
        System.out.println("-------------------------------------------------------");
    }
}
```

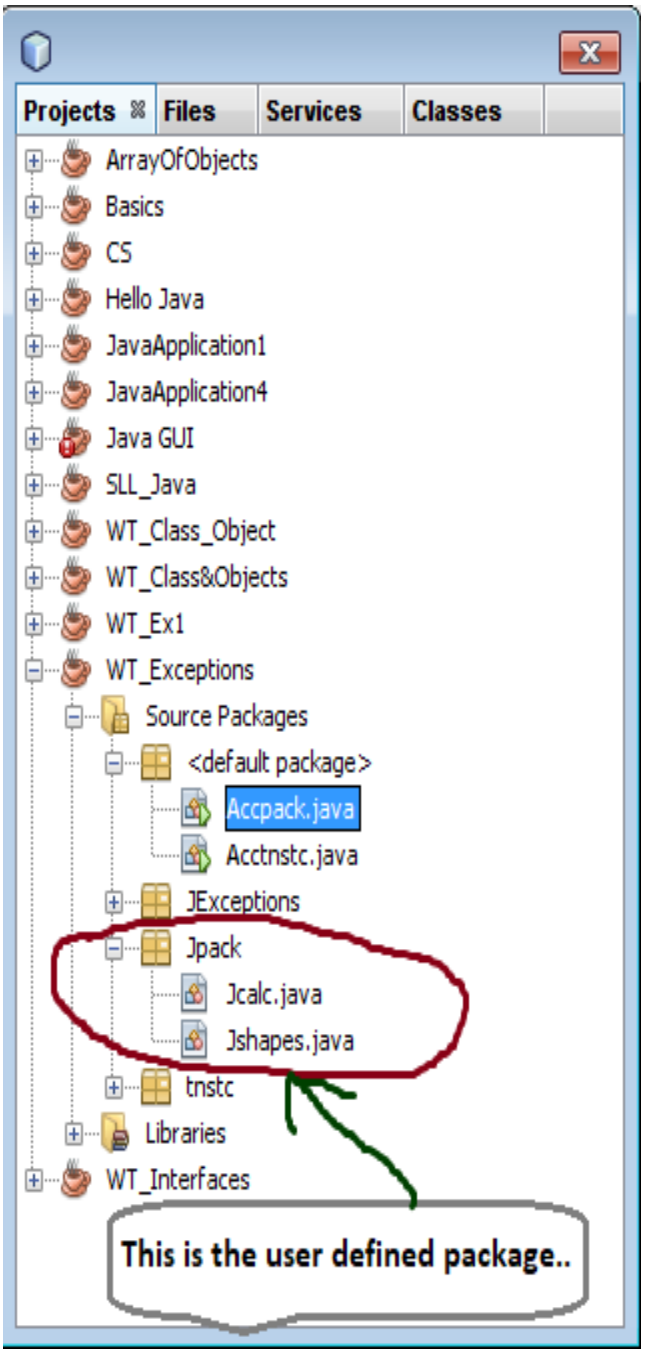
2. OUTPUT

Output - WT_Exceptions (run)

```
run:
==================================================================
                 Accessing TNSTC Package
==================================================================
Welcome to Thanthai Periyar Transport Corporation-Villupuram
------------------------------------------------------------------
Welcome to Anna Transport Corporation-Salem
------------------------------------------------------------------
Welcome to MGR Transport Corporation-Kanchipuram(Villupuram)
------------------------------------------------------------------
Welcome to State Express Transport Corporation-Chennai
------------------------------------------------------------------
Welcome to Pallavan Transport Corporation-Chennai
------------------------------------------------------------------
BUILD SUCCESSFUL (total time: 0 seconds)
```

II. ACCESSING MULTIPLE CLASSES OF SAME PACKAGE

PACKAGE DETAILS



Package name	:	Jpack
No of files	:	2 (Tamilnadu_Transport.java)
No of classes	:	2 {Jcalc.java, Jshapes.java}

1. SOURCE CODE

(a.1) Implementation Class [Compiled Class] / Developer Code

1. Jcalc.java

```
package Jpack;
public class Jcalc
{
// add the two numbers
   public int add(int a, int b)          // essential information
   {
      return (a+b);
   }
// subtract the two numbers
   public int sub(int a, int b)          // essential information
   {
      return (a-b);
   }
// multiply the two numbers
   public int mul(int a, int b)          // essential information
   {
      return (a*b);
   }
// divide the two numbers
   public int div(int a, int b)          // essential information
   {
      return (a/b);
   }

// power of two numbers
   public int power(int a, int b)        // essential information
   {
      int rs=1;
      if(a!=0)
```

```
            for(int i=0;i<b;i++)
            {
                rs*=a;
            }
        else
            rs=0;
        return rs;
    }
}
```

(a.2) Implementation Class [Compiled Class] / Client Application

2. Jshapes.java

```
package Jpack;
public class Jshapes
{
// area of the Square
    public int square(int x)                    // essential information
    {
        return (x*x);
    }
// area of the Circle
    public double circle(int r)                 // essential information
    {
        return(3.14*r*r);
    }
// area of the Rectangle
    public long rectangle(int l,int w)          // essential information
    {
        return(l*w);
    }
}
```

(b) Accessing Class [Essential Features]

Accpack.java

```
// import user defined package as system package
import Jpack.*;
public class Accpack
{
// main method
   static public void main(String[] args)
   {
      System.out.println("=====================================");
      System.out.println("\tAccessing Multiple classess of Jpack Package");
      System.out.println("=====================================");
// create objects for Jcalc & Jshapes classes
      Jcalc cc=new Jcalc();
      Jshapes obj=new Jshapes();
      System.out.println("\tAccessing Jcalc Class");
// call all the methods using calc object
      int r1=cc.add(50, 50);
      int r2=cc.mul(10, 10);
      int r3=cc.div(200, 25);
      int r4=cc.sub(500, 250);
      int r5=cc.power(5, 3);
// print the results
      System.out.println("Addition\t: "+r1);
      System.out.println("Product\t\t: "+r2);
      System.out.println("Divison\t\t: "+r3);
      System.out.println("Subtraction\t: "+r4);
      System.out.println("Power\t\t: "+r5);
      System.out.println("-------------------------------------------------------");
      System.out.println("\tAccessing Jshapes Class");
// call all the methods using shape object
      int r6=obj.square(5);
```

```
        double r7=obj.circle(10);
        long r8=obj.rectangle(20, 5);
// print the results
        System.out.println("Area of the Square\t: "+r6);
        System.out.println("Area of the Circle\t: "+r7);
        System.out.println("Area of the Rectangle\t: "+r8);
        System.out.println("-------------------------------------------------------");
    }
}
```

2. OUTPUT

```
Output - WT_Exceptions (run)
run:
=========================================================
        Accessing Multiple classess of Jpack Package
=========================================================
        Accessing Jcalc Class
Addition        : 100
Product         : 100
Divison         : 8
Subtraction     : 250
Power           : 125
---------------------------------------------------------
        Accessing Jshapes Class
Area of the Square      : 25
Area of the Circle      : 314.0
Area of the Rectangle   : 100
---------------------------------------------------------
BUILD SUCCESSFUL (total time: 0 seconds)
```

HIDING CLASSES FROM PACKAGES

- It is possible to hide various classes from mentioned package.
- When we import a particular package using * symbol, only all public classes are imported. But all non public classes are omitted. In this way, we can hide any classes based on the demands.

Example

```
package itmit;

class firstyear                    // hidden
{
    // user code
}
public class secondyear            // visible any where.
{
    // user code
}
class thirdyear                    // hidden
{
    // user code
}

...

import itmit.*;
firstyear fy=new firstyear();              // error: firstyear is not available
secondyear sy=new secondyear();            // OK: secondyear is available
thirdyear ty=new thirdyear();              // error: thirdyear is not available
```

Note

- Here classes firstyear and thirdyear are not accessible. Since they are not public.
- These classes are used internally in their own package. This is called as hiding classes.