__/__/____

Java Package

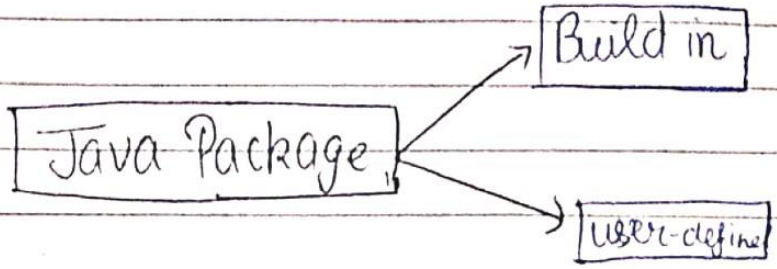
- A Java package is a group of similar types of classes, interface and subpackages.
- It providing access protection and name space management.
- Package in Java can be categorized in two form, built-in Package and User-defined package.
- There are many built-in Packages such as java, lang, awt, swing, net, io, util etc.
- Here, we will have detailed learning of creating and using user-defined packages.

Advantages

→ Java Package is used to categorize the class and interfaces so that they can be easily maintained.

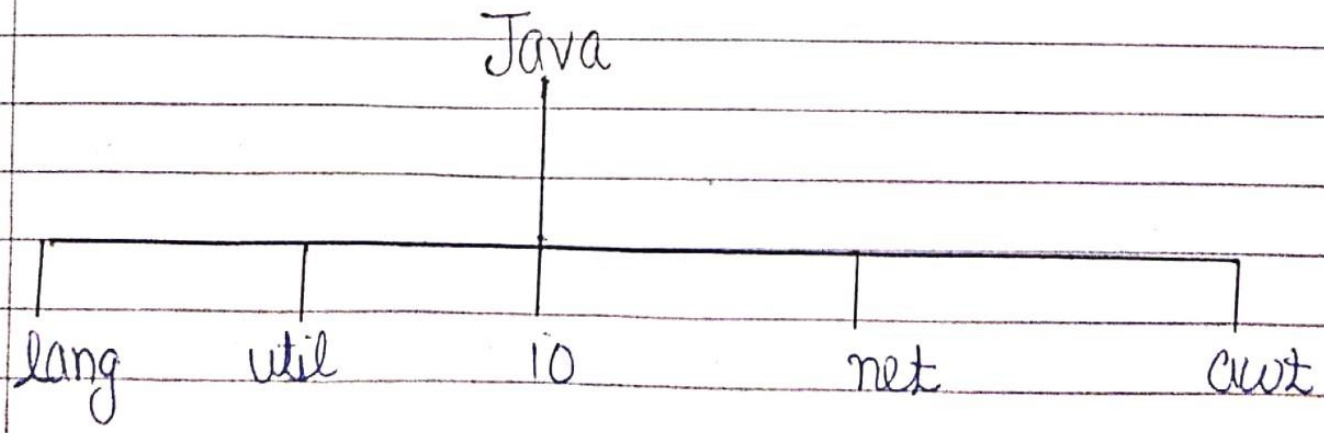
→ Java Package provides access protection.

→ Java Package removes naming collision.



* Java System Packages and their classes :-

Package name	Contents
Java.lang	Language support classes
Java.util	Language utility classes such as vectors, hash numbers, dates etc.
Java.awt	Set of classes for implementing GUT.
Java.io	Input / output support classes
Java.net	classes for networking.
Java.applet	classes for creating and implementing applets.



Types of Packages:-

There are two types of Packages:-

1. Built in packages :- Java API
2. User defined :- Custom packages

1 Built In Packages - The Package which are already created by Java developer people are called pre-defined Packages.

Example:- java.lang, java.applet, Java.awt, java.io, java.net, java.util.

(1) java.lang :- It is the default package also known as heart of Java.

→ It contains classes for primitive types, strings, math functions, threads and exceptions.

→ Without using this package we can't write even a single program and we not need to import this program.

(2) java.util :- It contain utility classes also known as collection framework.
Example:- vectors, stack, Hashset, trees, date etc.

(3) Java.io :- I/O stands for Input/Output.

This Package is very useful to perform input/output operations on file.
Example)- FileWriter, FileReader etc.

(4) Java.applet :- This Package mainly use to develop GUI related application

→ Classes for creating and implementing applets

(5) Java.awt :- awt stands for abstract window tool kit.

→ It is also used to developed GUI application

→ The only difference between applet & awt program is, awt programs are stored alone program and it contain main() unlike applet.

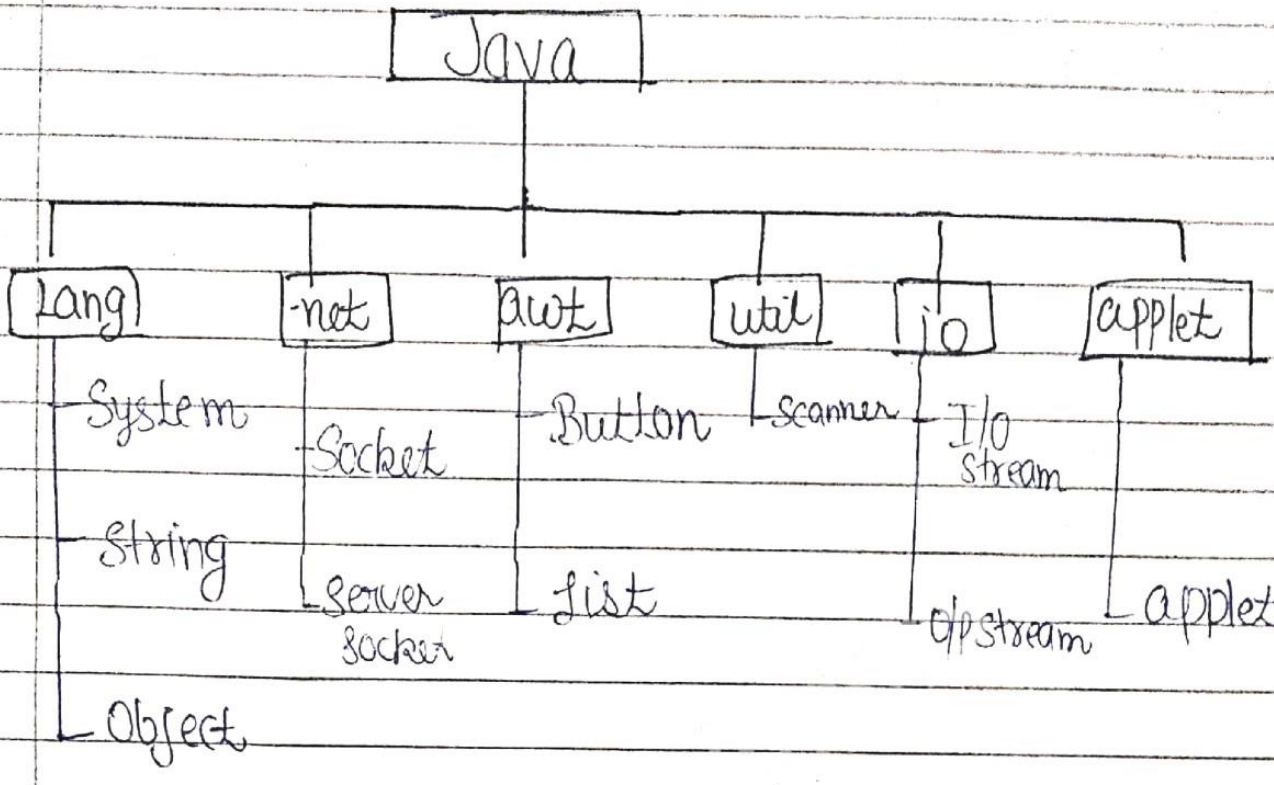
→ Example :- frame, Button, Textfield etc.

(6) Java.net :- They include classes for communicating with local computers as well as with internet servers.

→ URL, URL Connection and so on.

(7) Javax.swing

(8) Java.SQL (JFRAME, JBUTTON).



Example:- (Built in Packages)
Java Program of demostrating use of java.util package.

```
import java.util.Arrays;
class Jutil
{
public static void main (String args[])
{
int[] intArray = { 10, 40, 30, 50, 20 };
Arrays.sort(intArray);
System.out.printf("Sorted Array is : %s", Arrays.
                  toString(intArray));
}
}
```

Output :- [10, 20, 30, 40, 50]

2. User-defined Packages :-

The Package which are created by Java Programmer or user for their own use are called User defined Package.

To create the User defined package we use the package keyword.

Syntax :- package packagename ;

Example :- package pack ;

Note :-

→ package statement should be the first statement in a Java file.

→ Where as to use a package we use import keyword.

Compile and Run

```
package pack;
public class Demo
{
    public static void main (String[] args)
    {
        System.out.println ("jpwebdevelopers");
    }
}
```

Compilation :- package (classes) must be compiled with "-d" option to create package physically

Syntax :- javac -d . Demo.java
↑ filename

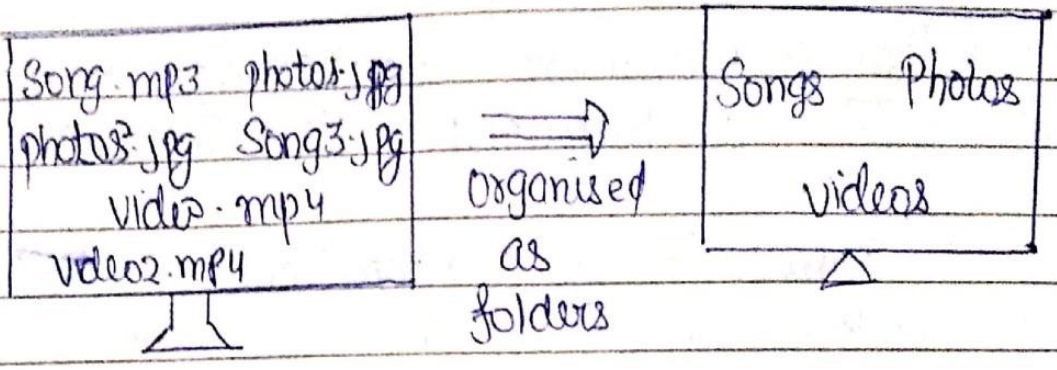
Run :- We must use package name in executing a packaged class.

Syntax :- java pack.Demo
↑ Package name

Output :- jpwebdevelopers

* Defining a Package :-

A Package can be defined as a grouping of related types (classes, interface) provides access protection.



this.java my.mp3
Song.java jp.java ⇒ Organised as Packages

Creating a package

javac jp.java → Creates jp class.
javac -d . jp.java → Creates a package folder

→ We must first declare the name of the package using the package keyword followed by a package name.

→ The must be the first statement in a java source file.

→ Then we define a class, just as we normally define a class

Example:-

```
                    Package name
                        ↓
package FirstPack;          // Package declarat
public class Myclass // class definition
{
    ----
    ---- (body of class)
}
```

→ file saved as Myclass.java and located in a directory named FirstPack.

→ When the source file is compiled, Java will create a .class file and store in the same directory.

Note:- Declare the Package at the beginning of a file using the form

```
package package_name;
```

→ declare the class that is to be put in the package and declare it public.

```
package pack1;
public class A
{
    public void display() {
        System.out.println("JP");
    }
}
```

* Importing a Package :-

To import Java Package into a class, we need to use Java import keyword which is used to access package and its classes into the Java program.

Ways to access package :-

1. import package.*; (with all classes)
2. import package.classname; (with specified class).

1. Import all classes of package :-

→ If we use packagename.* statement, then all the classes and interfaces of this package will be accessible, but not sub packages.

→ The import keyword is used to make the classes of another package accessible to the current package.

Example :-

```
                    // Save by Jp.java
package pack;
public class Jp
{
```

```
public void msg()
  {
    System.out.println("Hello");
  }
}

// Save by Web.java
package mypack;
import pack.*;

class Web
  {
public static void main(String args[])
  {
    JP obj = new JP();
    obj.msg();
  }
}
```

Output :- Hello

② Using packagename.classname :-

- Java allows us to specify class name along with package name.
- If we use import packagename.classname statement then only the class with name in the package will be available for use.

Example:-

```
package pack;                    //Save by Demo.java
public class Demo
{
 Public void msg()
 {
  System.out.println("Hello");
 }
}
```

```
package mypack;                  //save by Test.java
import pack.Demo;
class Test
{
public static void main (String args[])
 {
 Demo obj = new Demo();
  obj.msg();
 }
}
```

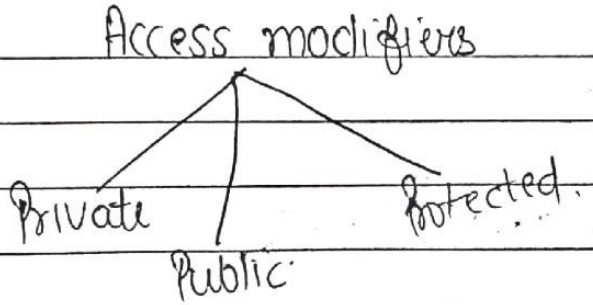
Output :- Hello

Access protection in Java Packages:-

- Access modifiers define the scope of the class and its members (data and methods).
- For example:- private members are accessible within the same class members (methods).

- There are four categories, provided by Java regarding the visibility of the class members between classes and packages.

→ Subclasses in the same package.
→ Non-subclasses in the same package.
→ Subclasses in different packages.
→ Classes that are neither in the same package nor sub classes.



→ Private cannot be seen outside of its class
→ public can be access from anywhere.
→ protected can be accessible in sub class only in the hierarchy.
→ default members are accessible within the same package

	Private	No modifier	Protected	Public
Same class	Yes	Yes	Yes	Yes
Same Package sub class	No	Yes	Yes	Yes
Same Package non-sub class	No	Yes	Yes	Yes
Different " Sub class	No	No	Yes	Yes
Different " Non sub class	No	No	No	Yes

→ A class can have only two access modifier one is default and another is public.

→ If the class has default access then it can only be accessed within the same package by any other code.

→ But if the class has public access that it can be access from anywhere

Example (Using Public, Private, No modifier)

```
package Test;

public class TestClass
{
    public int a;
    int b;
    private int c;

    public void fun1 ()
    {
```

```
void fun2()
{
}

private void fun3()
{
}
}

import Test.TestClass;
public class Main
{
  public static void main(String args[])
  {
    TestClass obj = new TestClass();
    obj.a = 10;          // allowed
    obj.b = 20;          // cannot access
    obj.c = 30;          // cannot access

    obj.fun1();          // allowed
    obj.fun2();          // error cannot access
    obj.fun3();          // error cannot access
  }
}
```

__/__/____

Example (Protected access modifier)

```
package pack;
public class A
{
  protected void msg()
  {
  System.out.println("Hello");
  }
}
```

// Save by A.java

// Save by B.java

```
package mypack;
import pack.*;

class B extends A
{
  public static void main(String args[])
  {
    B obj = new B();
    obj.msg();
  }
}
```

Output :- Hello