**Practical No.7**

**Name**: Anuj Rajendra Mane

**Div**: SY A

**Roll No**: 65 **Batch**: A3

**PRN** **No**: 2162701242054

**Title: Program on Package**

**Theory Of Package:**

Java provides a mechanism for partitioning the class name space into more manageable chunks. This mechanism is the package. The package is both a naming and a visibility control mechanism. You can define classes inside a package that are not accessible by code outside that package. You can also define class members that are only exposed to other members of the same package.

A java package is a group of similar types of classes, interfaces and sub-packages.

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, javax, swing, net, io, util, sql etc.

**Advantage of Java Package**

1) Java package is used to categorize the classes and interfaces so that they can be easily maintained.

2) Java package provides access protection.

3) Java package removes naming collision.

4) A group of package called a library. The classes and interfaces of a package are likes books in a library and can be reused several times.

**Defining a Package:**

• Creating a package is quite easy: simply include a package command as the first statement in a Java source file. Any classes declared within that file will belong to the specified package. The package statement defines a name space in which classes are stored.

**Creating a Package:**

•The general form of the package statement:

package pkg; //Here, pkg is the name of the package.

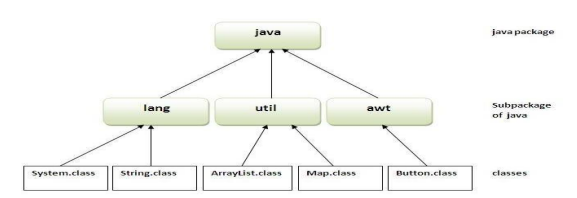
• For example, the following statement creates a package called MyPackage.

package MyPackage;

The package keyword is used to create a package in java. Java uses file system directories to store packages. For example, the .class files for any classes you declare to be part of MyPackage must be stored in a directory called MyPackage.

**For example**,

a package declared as package java.awt.image; needs to be stored in java:awt:image on your UNIX, Windows, or Macintosh file system, respectively.

****•You cannot rename a package without renaming the directory in which the classes are stored.

//save as Simple.java

package mypack;

public class Simple {

public static void main(String args[]) {

System.out.println("Welcome to package");

}

}

**How to compile java package**

If you are not using any IDE, you need to follow the syntax given below:

javac -d directory javafilename

**For example**: javac -d . Simple.java

The -d switch specifies the destination where to put the generated class file. You can use any directory name like /home (in case of Linux), d:/abc (in case of windows) etc. If you want to keep the package within the same directory, you can use . (dot).

**How to run java package program**

You need to use fully qualified name e.g. mypack.Simple etc to run the class.

**To Compile:** javac -d . Simple.java

**To Run**: java mypack.Simple

**Output**: Welcome to package

**How to access package from another package?**

There are three ways to access the package from outside the package.

1. import package.\*;

2. import package.classname;

**1) Using packagename.\***

If you use package.\* then all the classes and interfaces of this package will be accessible but not subpackages. The import keyword is used to make the classes and interface of another package accessible to the current package.

**Example of package that import the packagename.\***

//save by A.java

package pack;

public class A {

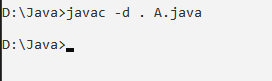
public void msg() {

System.out.println("Hello");

}

}

Compiling and creating package



//save by B.java

package mypack;

import pack.\*;

class B{

public static void main(String args[]){ A obj = new A();

obj.msg();

}

}

**Command**:

javac -d . A.java

javac -d . B.java

java mypack.B

**Output**:



**2) Using packagename.classname:**

If you import package.classname then only declared class of this package will be accessible. Example of package by import package.classname

//save by A.java

package pack;

public class A {

public void msg() {

System.out.println("Hello1");

}

}

//save by B.java

package mypack;

import pack.A;

class B1 {

public static void main(String args[]) {

A obj = new A();

obj.msg();

}

}

**Command:**

javac -d . A.java

javac -d . B1.java

java mypack.B1

**Output**:



**Conclusion**:

I can learn the concepts of package as well as running the programs in Ide’s and Notepad with the help of CMD.

**Completion Date: Co-Ordinator Sign:**