Packages

Introduction : A **java package** is a group of similar types of classes, interfaces and sub-packages. A **Package** can be defined as a grouping of related types (classes, interfaces, enumerations and annotations) providing access protection and namespace management.

(or)

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.

Package in java can be categorized in two form, **built-in package** and **user-defined package**.

i. **Built-in packages :** such as java. lang, java.awt, javax.swing, java.util, java.sql etc

ii. **User Defined Packages**: These are created by using **package** statement. To create a package is quite easy: simply include a package command as the first statement in Java source file.

General form for creating a package:

package packagename;
e.g.: package nagapack;

· The first statement in the program must be package statement while creating a package.
· While creating a package except instance variables, declare all the members and the class itself as public then only the public members are available outside the package to other

You can create a hierarchy of packages. To do so, simply separate each package name from the one above it by use of a period. The general form of a multileveled package statement is shown here:

package *pkg1*[.*pkg2*[.*pkg3*]];
e.g package nagendra.mohan.pradeep;

The above statement creates a package hierarchy.

Finding Packages and CLASSPATH :

As just explained, packages are mirrored by directories. This raises an important question: How does the Java run-time system know where to look for packages that you create? The answer has three parts. First, by default, the Java run-time system uses the current working directory as its starting point. Thus, if your package is in a subdirectory of the current directory, it will be found. Second, you can specify a directory path or paths by setting the **CLASSPATH** environmental variable. Third, you can use the **-classpath** option with **java** and **javac** to specify the path to your classes.

Advantages of Java Packages:

1. Java package is used to categorise the classes and interfaces so that they can be easily maintained.
2. Java package provides access protection.
3. Java package removes names collision.

For Creating User-Defined Package we will perform the following steps

1. Include package statement be the first line of java source file
2. Write the java source code
3. Compile : javac -d **destinationfolder** sourcefilename.java

   Here **destination folder** specified where to place generated class files.Then a folder with the given package name is created in the specified destination, and the compiled class files will be placed in that folder.
4. Run : packagename. Sourcefile name.

<u>Sample java Program for packages</u>

```
package pavan;          // package statement   here package name  is pavan
import java.lang.*;
class Balance
{
        String name;
        double bal;
        Balance(String na,double bal)
        {
                name=na;
                this.bal=bal;
        }
        void show()
        {
                System.out.println("A/c Holder name..."+name);
                System.out.println("balance....$"+bal);
        }

}
class AccountBalance
{
        public static void main(String args[])
        {
                Balance b1=new Balance("mahesh",500);
                b1.show();
        }

}
```

Save the above program in **AccountBalance.java**

Compile the program using the following statement

```
D:\> javac –d  .   AccountBalance.java
```

After executing the above statement it creates pavan directory under D drive and also place the generated classes i.e AccountBalance.class and Balance.clas in pavan directory.

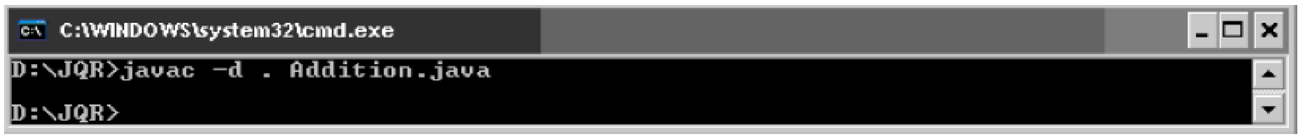
Running the program using the following statement.

```
java pavan.AccountBalance
```

Program 1: Write a program to create a package pack with Addition class.

```
//creating a package
package pack;
public class Addition
{
            private double d1,d2;
           public Addition(double a,double b)
            {
                        d1 = a;     d2 = b;
            }
           public void sum()
          {
             System.out.println ("Sum of two given numbers is : " + (d1+d2) );
          }
}
```

Compiling the above program:



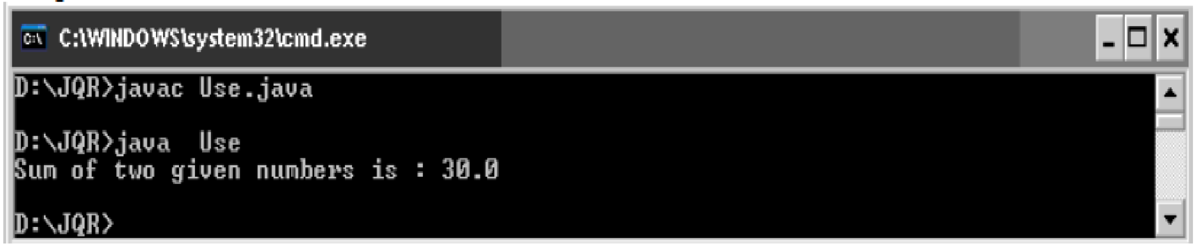
```
D:\JQR>javac -d . Addition.java

D:\JQR>
```

The –d option tells the Java compiler to create a separate directory and place the .class file in that directory (package). The (.) dot after –d indicates that the package should be created in the current directory. So, out package pack with Addition class is ready.

Program 2: Write a program to use the Addition class of package pack.

```
//Using the package pack
import pack.Addition;
class Use
{
    public static void main(String args[])
      {
              Addition ob1 = new Addition(10,20);
              ob1.sum();
       }
}
```

Output:



```
D:\JQR>javac Use.java

D:\JQR>java  Use
Sum of two given numbers is : 30.0

D:\JQR>
```