* Packages in Java

*Packages

* Packages are those class which contains methods without the main method in them.
* Packages are mainly used to reuse the classes which are already created/used in other program.
* We can define many number of classes in the same package.
* Packages are mainly divided into two types. They are

  1. Built in packages

  2. User defined Packages.

*The main use of Packages is:

- *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.

*Built-In Packages

Built Packages are provided for the programme in-order to reduce the burden.Some of the frequently used built in packages are:

*lang

*util

*io

*awt

*net

*applet

* These built in packages are placed in the package “java”.
* In order to use these packages/classes present in the package we need to import them in the current program.
* The syntax for importing packages is: **import java.util.***
* The above statement tells the compiler that you need to import all the classes present in the “util” package,which in turn present in the “java” package.
* The asterisk symbol tells the compiler to import all the classes in present in the util package.If you want to import specific class then we need to mention the name of the class instead of the asterisk.For eg: **import java.util.Scanner**

* **1.java.lang:**language supports classes.These are classes that java compiler itself uses and therefore automatically imported.They include classes for primitive types,stirngs,maths,threads and exceptions.
2.java.util:language utility classes such as vecotrs,hash tables,random numbers,date etc.
3.java.io:input/output support classes.They provide the facility for input/output of data.
4.java.awt:set of classes for implementing graphical user interface.They include classes for windows,buttons,lists
5.java.net: Classes for networking.They include classes for communicating with local computers as well as with internal servers.

*Built in Packages

User- Defined Package

* create a user defined package. For this we need to create a folder with your desired name say *mypackage*.

```
package mypackage;
public class A
 {
   public void display()
    {
      System.out.println("Hello world");
     }
 }
```

* Now we have created a package.We shall now reuse the above defined class in other program.

```
import mypackage.*;
class packagedemo
 {
   public static void main(String arg[])
    {
      A ob = new A();
      ob.display();
    }
 }
```

There fore we have reused the a class defined in other program. Before executing the second program we need to compile the package i.e. javac a.java

*User- Defined Package

*Steps For Creating Package

To create a user defined package the following steps should be involved :-

1: Declare the package at the beginning of a file using the syntax :-

```
package packageName;
```

2: Define the class that is to be put in the package & declare it public.

3: Create a subdirectory under the directory where the main source files are stored.

4: Store the listing as the classname.java file in the subdirectory created.

5: Compile the file. This create .class file in the subdirectory.

```
//save as Simple.java
package mypack;
public class Simple{
 public static void main(String args[]){
    System.out.println("Welcome to package");
   }
}
```

*Simple example of java 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).

* 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

* The -d is a switch that tells the compiler where to put the class file i.e. it represents destination. The . represents the current folder.

How to run java package program

Java also supports the concept of package hierarchy. This is done by specifying multiple names in a package statement, seprated by dots (.).

Ex :- package firstPackage.secondPackage;

This approach allows us to group related classes into a package and their group related package into a larger package. Store this package in a subdirectory named firstpackage/secondPackage.

A java package file can have more than one class definition. In such cases, only one of the classes may be declared public & that class name with .java extension is the source file name. When a source file with more than one class definition is compiled, java creates independent .class files for those classes.

*Accessing a Package

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

i. import package.*;

ii. import package.classname;

iii. fully qualified name.

- 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.

Using packagename.*

Example of package that import the packagename.*

```
//save by A.java
package pack;
public class A{
  public void msg(){System.out.println("Hello");}
}
//save by B.java
package mypack;
import pack.*;

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

Output: Hello

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("Hello");}
}
//save by B.java
package mypack;
import pack.A;

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

Output:Hello

Using fully qualified name

- If you use fully qualified name then only declared class of this package will be accessible. Now there is no need to import. But you need to use fully qualified name every time when you are accessing the class or interface.
- It is generally used when two packages have same class name e.g. java.util and java.sql packages contain Date class.

*Using fully qualified name

```
//save by A.java
package pack;
public class A{
  public void msg(){System.out.println("Hello");}
}
//save by B.java
package mypack;
class B{
  public static void main(String args[]){
   pack.A obj = new pack.A();//using fully qualified name
   obj.msg();
  }
}
```