Package

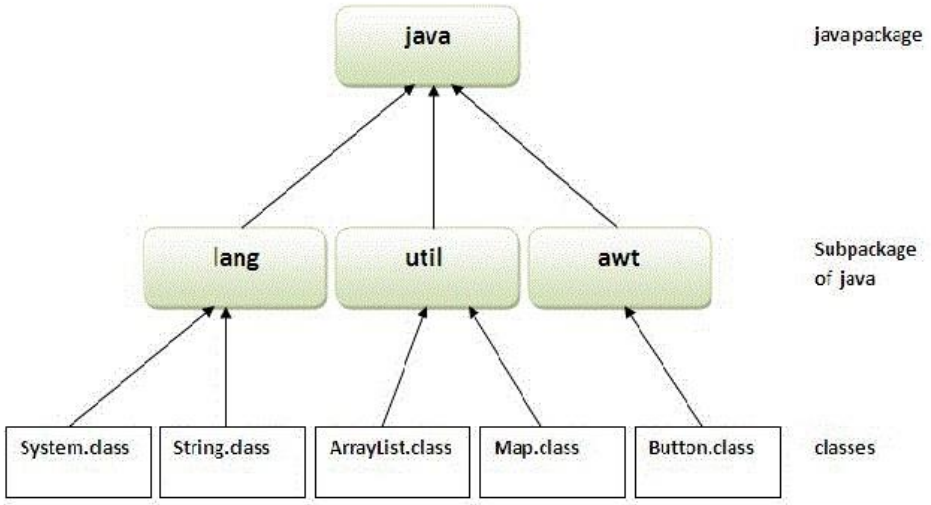
A **package** is a group of similar types of classes, interfaces and sub-packages. Package can be categorized in two form,

1. Built-in package
2. User-defined package.

There are many built-in packages such as java, lang, awt, javax, swing, net, io, util, sql etc.

Advantage of Package

- Package is used to categorize the classes and interfaces so that they can be easily maintained.
- Package provides access protection.
- Package removes naming collision.



The **package keyword** is used to create a package.

```
//save as Hello.java
package mypack;
public class Hello
```

```
{
public static void main(String args[])
{
   System.out.println("Welcome to package");
}
}
```

How to compile the Package (if not using IDE)

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

javac -d directory java_filename

For example

javac -d . Hello.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 the Package (if not using IDE)

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

To Compile: javac -d . Hello.java

To Run: java mypack.Hello

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 dot(.) represents the current folder.

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.classname1;**
3. **Fully qualified name.**

Using packagename.*

If you use package.* then all the classes and interfaces of this package will be accessible but not subpackag.

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

//save by B.java

```
package pkg2;
import pkg.*;
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 pkg;

public class A

{

    public void msg()

        {System.out.println("Hello");    }

}


//save by B.java

package pkg2;
import pkg.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.

Example of package by import fully qualified name

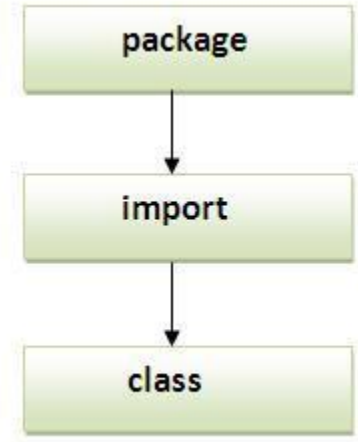
```
//save by A.java

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();
  }
}
```

Output:Hello

Note: 1. If you import a package, subpackages will not be imported. Therefore, If you import a package, all the classes and interface of that package will be imported excluding the classes and interfaces of the subpackages. Hence, you need to import the subpackage as well.

Note: Sequence of the program must be package then import then class.



Access Specifier in Java

- When members of the class are private, they can't be accessed from outside the class body.
- When members of the class are protected, they can be accessed from any class of the same package and child class from the other package.
- When members of the class are public, they are accessible from any class of any package.
- When members of the class are default (not a keyword), they are accessible only from the class of same package.

	Private	Default	Protected	Public
Same class	YES	YES	YES	YES
Same package subclass	NO	YES	YES	YES

Same package **Non-subclass**	NO	YES	YES	YES
Different package subclass	NO	NO	YES	YES
Different package **Non-subclass**	NO	NO	NO	YES