Understanding Java Packages:

1) A package consists of similiar types of Classes, Interfaces & Enums
2) In java there are '2' types of packages are there

Packages
- Predefined Packages
- User-defined Packages

consists of most commonly used Classes & Interfaces by the programmers

3) In Java nearly there are 5000 predefined packages are present.
4) In every Java program one package by default it will be imported & that package is **java.lang**
5) We can access the members of one class from another class of **same package** with out using **import** statement.
6) **'import'** statement is used to connect classes in java application of **different** packages.
7) In every Java program package statement will be the **first** statement.

Q) How to access package from another package?

A) We can import a class from one package into another package in below mentioned **'3'** ways

- By using import packageName.ClassName; [Only the **specified Class** will be imported]
- By using import packageName.*; [**All the classes** which were present in that package will be imported]
- By using Fully Qualified ClassName. [We can **(import)** a class into our program **with out using** import statement]

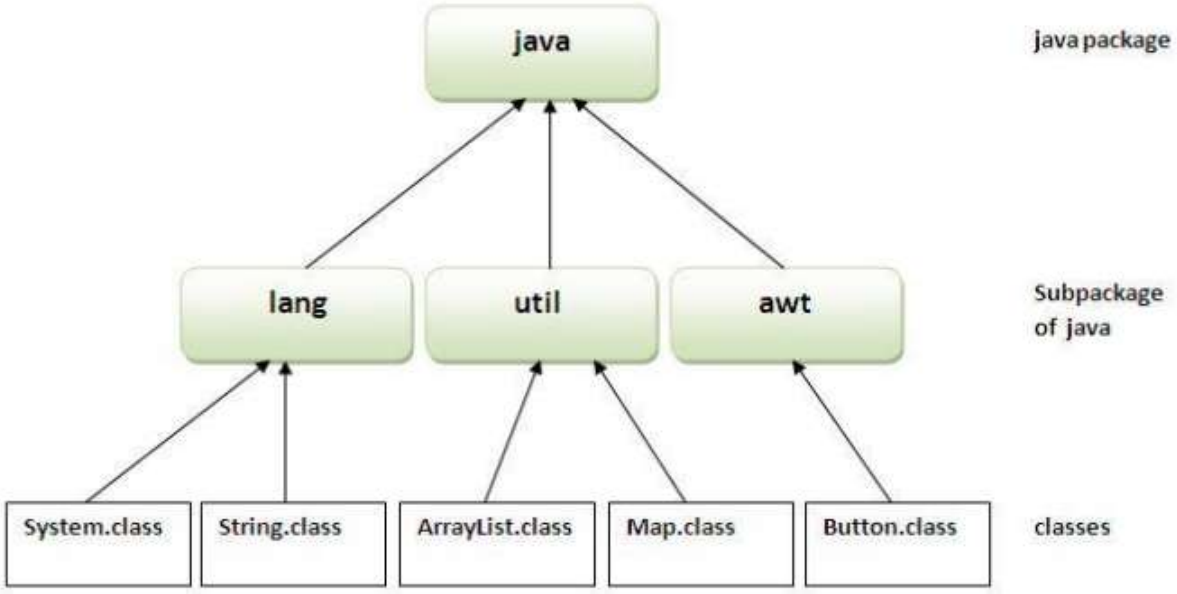
Package in Java

- A package is a group of similar types of classes, interfaces and sub-packages.
- Package 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.
- Java API is having nearly 5000 pre defined packages.
- Package statement will be the first statement in a java program.
- We can access the members of one class from another class of same package.
- 'import' statement is used to connect classes in java application of different packages.

Some Important Packages

Package	Description
java.lang	Lang stands for 'language' ,this got primary classes and interfaces essential for developing a basic java program
java.util	Util stands for 'utility', This package contains useful classes and interfaces like Stack, LinkedList, Hashtable, etc ... These classes are called collections.
java.io	Io stands for 'input and output'. This package contains streams.
java.awt	awt stands for 'abstract window toolkit'. This package helps to develop GUI.
javax.swing	This package helps to develop GUI like java.awt. The 'x' in javax represents that it is an extended package.
java.net	net stands for 'network'. Client-Server programming can be done by using this package.
java.applet	Applets are programs which came from a server into a client and get executed on the client machine on a network.
java.text	This package has two important classes, DateFormat and NumberFormat.
java.sql	Sql stands for 'structured query language'. This package helps to connect to databases.

Structure of a package



Simple example of package

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

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

How to access package from another package?

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

✓import packageName.*;

✓import packageName.classname;

✓fully qualified Classname

Using packagename.*

- If you use packagename.* 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.

```
//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 packagename.classname then only declared class of this package will be accessible.

```
//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.

```
//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

Understanding Access Modifiers

- Access modifiers determine whether other classes can use a particular field or invoke a particular method.
- There are '2' levels of access modifiers.

1. *At Class level*: public & default
2. *At member level*: public, private, protected and default

At Class level:

- If a class is declared as public then that is visible to all the classes everywhere.
- If a class is declared as default (package-private)then that is visible to only within its own package.

At Member level:

- At the member level, we can also use the public or default (package-private) just as with class level, and with the same meaning.

same package | different package

Modifier	Class	Package	Sub-Class	World
public	YES	YES	YES	YES
protected	YES	YES	YES	
default	YES	YES		
private	YES			

PUBLIC:

- If a method, variable or constructor is declared as public then we can access them from anywhere.
- When we are accessing the public member its class also should be public otherwise will be getting compile time error.

DEFAULT:

- If a method, variable or constructor is declared as default then we can access them from current package only. So it is also called "PACKAGE -PRIVATE"

PRIVATE:

- If a method, variable or constructor is declared as private then we can access them in the current class only.
- Private is the most restricted access modifier.
- If a constructor is declared as private we can't create a object for that class in other classes.

PROTECTED:

- If a method, variable or constructor is declared as protected then we can access them with in the current package.
- We can use PROTECTED members outside the package only in child class, and we can access them by using child class reference only not from parent class reference.

```java
package com.pack1;

public class ClassA
{

}
```

```java
package com.pack2;

public class ClassX extends ClassA
{
    new ClassA();
}
```

*ClassB.java

Training ▸ src ▸ com.pack1 ▸ ClassB ▸ main(String[]) : void

```java
package com.pack1;

public class ClassB extends ClassA
{
    public static void main(String[] args)
    {
        new ClassA();
    }
}
```

```java
package com.pack1;

public class ClassA
{

}
```

```java
package com.pack2;

import com.pack1.ClassA;
public class ClassX extends ClassA
{
    new ClassA();
}
```

*ClassB.java

Training ▸ src ▸ com.pack1 ▸ ClassB ▸ main(String[]) : void

```java
package com.pack1;

public class ClassB extends ClassA
{
    public static void main(String[] args)
    {
        new ClassA();
    }
}
```

```java
package com.pack1;

public class ClassA
{

}
```

```java
package com.pack2;

import com.pack1.ClassA;
public class ClassX extends ClassA
{
    ClassA aobj=new ClassA();
}
```

ClassB.java

Training ▸ src ▸ com.pack1 ▸ ClassB ▸ main(String[]) : void

```java
package com.pack1;

public class ClassB extends ClassA
{
    public static void main(String[] args)
    {
        new ClassA();
    }
}
```

```java
package com.pack1;

public class ClassA
{

}
```

```java
package com.pack2;

import com.pack1.ClassA;

public class ClassX extends ClassA
{
    ClassA aobj=new ClassA();
}
```

ClassB.java

Training ▸ src ▸ com.pack1 ▸ ClassB ▸ main(String[]) : void

```java
package com.pack1;

public class ClassB extends ClassA
{
    public static void main(String[] args)
    {
        System.out.println("Hi");
    }
}
```

```java
package com.pack1;

public class ClassA
{

}
```

```java
package com.pack2;

import com.pack1.ClassA;

public class ClassX extends ClassA
{
    ClassA aobj=new ClassA();
}
```

ClassB.java System.class

Training ▸ C:\Program Files\Java\jdk-17\lib\jrt-fs.jar ▸ java.lang ▸ System ▸

```java
 *
 * @since   1.0
 */
public final class System {
    /* Register the natives via the static
     *
     * The VM will invoke the initPhase1 m
     * of this class separate from <clinit
     */
    private static native void registerNat
    static {
        registerNatives();
```


```java
package com.pack1;

public class ClassA
{

}
```

```java
package com.pack2;

import com.pack1.ClassA;

public class ClassX extends ClassA
{
    ClassA aobj=new ClassA();
}
```

ClassB.java System.class

Training ▸ C:\Program Files\Java\jdk-17\lib\jrt-fs.jar ▸ java.lang ▸ System ▸


```java
 * Copyright (c) 1994, 2021, Oracle and/or
package java.lang;

import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.Console;
import java.io.FileDescriptor;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.PrintStream;
```

```java
package com.pack1;

public class ClassA
{
    public void meth1()
    {
        System.out.println("meth1() called");
    }
}
```

```java
package com.pack2;

public class ClassX
{

}
```

ClassB.java

Training ▸ src ▸ com.pack1 ▸ ClassB ▸ main(String[]) : void

```java
public class ClassB
{
    public void meth2()
    {
        System.out.println("meth2() called");
    }
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();
        aobj.meth1();
    }
}
```

Console

<terminated> ClassB [Java Application] C:\Pr

```
meth1() called
```

```java
package com.pack1;

public class ClassA
{
    public void meth1()
    {
        System.out.println("meth1() called");
    }
}
```

```java
package com.pack2;

import com.pack1.ClassA; // 1st way

public class ClassX
{
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();
        aobj.meth1();
    }
}
```

ClassB.java

Training ▸ src ▸ com.pack1 ▸ ClassB ▸ meth2() : void

```java
public class ClassB
{
    public void meth2()
    {
        System.out.println("meth2() called");
    }
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();
        aobj.meth1();
    }
}
```

Console

<terminated> ClassX [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (21-

```
meth1() called
```

```java
package com.pack1;

public class ClassA
{
    public void meth1()
    {
        System.out.println("meth1() called");
    }
}
```

ClassB.java

Training ▸ src ▸ com.pack1 ▸ ClassB ▸ meth2() : void

```java
public class ClassB
{
    public void meth2()
    {
        System.out.println("meth2() called");
    }
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();
        aobj.meth1();
    }
}
```

```java
package com.pack2;

import com.pack1.ClassA; // 1st way
import com.pack1.ClassB;

public class ClassX
{
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();
        aobj.meth1();

        ClassB bobj=new ClassB();
        bobj.meth2();
    }
}
```

Console

<terminated> ClassX [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (21

```
meth1() called
meth2() called
```

```java
package com.pack1;

public class ClassA
{
    public void meth1()
    {
        System.out.println("meth1() called");
    }
}
```

ClassB.java

Training ▸ src ▸ com.pack1 ▸ ClassB ▸ meth2() : void

```java
public class ClassB
{
    public void meth2()
    {
        System.out.println("meth2() called");
    }
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();
        aobj.meth1();
    }
}
```

```java
package com.pack2;

import com.pack1.ClassA; // 1st way
import com.pack1.ClassB;

//import com.pack1.*; // 2nd way

public class ClassX
{
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();
        aobj.meth1();

        ClassB bobj=new ClassB();
        bobj.meth2();
    }
}
```

Console

<terminated> ClassX [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (21-

```
meth1() called
meth2() called
```

It is highly recommended that we should use 1st way if need to import 90 classes too we should go with the 1st way.

```java
package com.pack1;

public class ClassA
{
    public void meth1()
    {
        System.out.println("meth1() called");
    }
}
```

ClassB.java

Training ▸ src ▸ com.pack1 ▸ ClassB ▸ main(String[]) : void

```java
public class ClassB
{
    public void meth2()
    {
        System.out.println("meth2() called");
    }
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();
        aobj.meth1();
    }
}
```

```java
package com.pack2;

import com.pack1.ClassA; // 1st way *** highly recommended
//import com.pack1.ClassB;

//import com.pack1.*; // 2nd way

public class ClassX
{
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();
        aobj.meth1();

        com.pack1.ClassB bobj=new com.pack1.ClassB();// 3rd way
        bobj.meth2();
    }
}
```

Console

```
<terminated> ClassX [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (21-Mar-2025, 8:30:48 am – 8:30:49 am) [pid:
meth1() called
meth2() called
```

Access Modifiers

```java
package com.pack1;

public class ClassA
{
    public void meth1()
    {
        System.out.println("meth1() called");
    }
    public static void main(String[] args)
    {
        new ClassA().meth1();
    }
}
```

ClassB.java

Training ▸ src ▸ com.pack1 ▸ ClassB ▸ main(String[]) : void

```java
package com.pack1;

public class ClassB
{
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();
        aobj.meth1();
    }
}
```

```java
package com.pack2;

import com.pack1.ClassA;

public class ClassX
{
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();
        aobj.meth1();

    }
}
```

Console

```
<terminated> ClassX [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (21-M
meth1() called
```

If access modifier of class is public

If the Access Modifiers of method is declared as public, we can access throughout the project

It means we can access public method in same class

We can access public methods in different classes of same package

We can access public methods in different classes of different packages.

```java
package com.pack1;

public class ClassA
{
    private void meth1()
    {
        System.out.println("meth1() called");
    }
    public static void main(String[] args)
    {
        new ClassA().meth1();
    }
}
```

ClassB.java

Training ▸ src ▸ com.pack1 ▸ ClassB ▸ main(String[]) : void

```java
package com.pack1;

public class ClassB
{
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();
        aobj.meth1();
    }
```

```java
package com.pack2;

import com.pack1.ClassA;

public class ClassX
{
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();
        aobj.meth1();

    }
}
```

Console

<terminated> ClassA [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (21-

~~meth1() called~~

If access modifier of class is public

If the Access Modifiers of method is declared as private, we can access only in the same class.

We cannot access private methods in different classes of same package

We cannot access private methods in different classes of different packages.

```java
package com.pack1;

public class ClassA
{
    void meth1()
    {
        System.out.println("meth1() called");
    }
    public static void main(String[] args)
    {
        new ClassA().meth1();
    }
}
```

ClassB.java

Training ▸ src ▸ com.pack1 ▸ ClassB ▸

```java
package com.pack1;

public class ClassB
{
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();
        aobj.meth1();
    }
```

```java
package com.pack2;

import com.pack1.ClassA;

public class ClassX
{
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();
        aobj.meth1();

    }
}
```

Console

<terminated> ClassB [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (21-

```
meth1() called
```

If access modifier of class is public.

If the Access Modifiers of method is declared as default(package-private), we can access in the same class.

We can access default methods in different classes of the same package

We cannot access default methods in different classes of different packages.

```java
package com.pack1;

public class ClassA
{
    protected void meth1()
    {
        System.out.println("meth1() called");
    }
    public static void main(String[] args)
    {
        new ClassA().meth1();
    }
}
```

ClassB.java

Training ▸ src ▸ com.pack1 ▸ ClassB ▸

```java
package com.pack1;

public class ClassB
{
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();
        aobj.meth1();
    }
```

```java
package com.pack2;

import com.pack1.ClassA;

public class ClassX extends ClassA
{
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();
        //aobj.meth1();

        new ClassX().meth1();
    }
}
```

Console

```
<terminated> ClassX [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (21-
meth1() called
```

If access modifier of class is public.

If the Access Modifier of a method is declared as protected, we can access in the same class.

We can access protected methods in different classes of the same package

We can access protected methods in different classes of different packages, by using inheritance we should use extends concept.

```java
package com.pack1;

class ClassA
{
    public void meth1()
    {
        System.out.println("meth1() called");
    }
    public static void main(String[] args)
    {
        new ClassA().meth1();
    }
}
```

```java
package com.pack1;

public class ClassB
{
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();
        aobj.meth1();
    }
```

```java
package com.pack2;

import com.pack1.ClassA;

public class ClassX extends ClassA
{
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();
        //aobj.meth1();

        //new ClassX().meth1();
    }
}
```

```
<terminated> ClassX [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (21-
meth1() called
```

If access modifier of a class is default.

Access modifier of a method is public.

We cannot import the class into different packages.

```java
package com.pack1;

public class ClassA
{
    public void meth1()
    {
        System.out.println("meth1() called");
    }
    public static void main(String[] args)
    {
        new ClassA().meth1();
    }
}
```

```java
package com.pack1;

public class ClassB
{
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();
        aobj.meth1();
    }
```

```java
package com.pack2;

import com.pack1.ClassA;

public class ClassX extends ClassA
{
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();
        //aobj.meth1();

        new ClassX().meth1();// Accessing Protected method from a
    }
}

//private < default  <  protected <   public
```

```
<terminated> ClassX [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (21-Mar-2025, 8:55:21 am – 8:55:22 am) [pid: 9
meth1() called
```

```java
package com.pack1;

public class ClassA
{
    protected void meth1()
    {
        System.out.println("meth1() called");
    }
    public static void main(String[] args)
    {
        new ClassA().meth1();
    }
}
```

```java
package com.pack2;

import com.pack1.ClassA;

public class ClassX extends ClassA
{
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();
        //aobj.meth1();

        new ClassX().meth1();
        // Accessing Protected method from another package
    }
}

//private < default  <  protected <   public
```

Console

```
<terminated> ClassX [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (21-Mar-2025, 8:55:21 am – 8:55:22
meth1() called
```

Private < default < protected < public