Packages: Packages are used in Java in order to prevent naming conflicts, to control access, to make searching/locating and usage of classes, interfaces, enumerations and annotations easier, etc.

Creating a Package: While creating a package, you should choose a name for the package and include a package statement along with that name at the top of every source file that contains the classes, interfaces, enumerations, and annotation types that you want to include in the package.

- The package statement should be the first line in the source file. There can be only one package statement in each source file, and it applies to all types in the file.
- If a package statement is not used then the class, interfaces, enumerations, and annotation types will be placed in the current default package.

Compile a Package: To compile the Java programs with package statements, you have to use -d option .

Syntax:

javac -d . file_name.java

For Example:

```
package house;

public class copy {
    public static void main(String[] args)
    {
        System.out.println("hello this is copy file");
    }
}
```

In package used multiple files: In this method used for create one package after then create files. These files are used same package name. Then calling all function in one file. If a class wants to use another class in the same package, the package name need not be used. Classes in the same package find each other without any special syntax

For Example:

```
package house;
class book
{
    public static void main(String [] args)
    {
```

```
        System.out.println("hello is my first package");
        school s1=new school();
        s1.show();
        hut ht=new hut();
        ht.display();
    }
}
```

The import Keyword: The import keyword is used to import a package, class or interface. A package in Java is used to group related classes. Think of it as a folder in a file directory. We use packages to avoid name conflicts, and to write a better maintainable code.

Syntax:

```
Package package_name;
Import package_name.*;
Class class_name
{
Main_function;
}
```

For Example:

```
package house;
import ship.*;
class book
{
    public static void main(String [] args)
    {
        System.out.println("hello is my first packaGE");
        school s1=new school();
        s1.show();
        hut ht=new hut();
        ht.display();
    }
}
```

Access package from another package: This method use from two ways like that:

- import package. classname;

For Example:

```
package ship;
import house.school;
public class hut
{
    public void display()
    {
        System.out.println("hello this is hut");
    }
   public static void main(String[]  args)
    {
        System.out.println("hello");
        hut ht=new hut();
        ht.display();
        school s1=new school();
        s1.show();
    }
}
```

- import package.*;

For Example:

```
package ship;
import house.*;
public class hut
{
    public void display()
    {
        System.out.println("hello this is hut");
    }
   public static void main(String[] args)
    {
        System.out.println("hello");
        hut ht=new hut();
        ht.display();
        school s1=new school();
        s1.show();
```

```
    }
}
```

Packages are divided into two categories:

- Built in package:

Java.lang: It is the default package, also known as heart of the java because without using this package we can't write a even a single program and we need to import this package.

For Example: System, String, Object, Integer etc.

Program:

```
import static java.lang.Math.*;
import static java.lang.System.*;
class print {
public static void main(String[] args)
    {
        // We are calling static member of System class
        // directly without System class name
        out.println("value of square=4");
out.println(sqrt(4));
out.println("value of power=5*5");
out.println(pow(5, 2));
out.println("value of absulate= -6.3");
out.println(abs(-6.3));
out.println("value of randon 0 between 2= ");
out.println(random()*2);
    }
}
```

java.util: This package is used to implement data structure of java. It contains utility classes also known as collection framework.

For Example: Linkedlist, Stack, Vector, Hashset, Treeset etc.

Program:

```
import java.util.Scanner;
 public class money
{
     void print()
```

```
    {
        Scanner sc=new Scanner(System.in);
System.out.print("Enter the Name:");
        String a=sc.next();
        System.out.println("Hello this is program for money");
        System.out.println(a+"  is a big company");
    }


    public static void main(String[] args)
    {
        money m1=new money();
        m1.print();

    }
}
```

java.io: It stands for input/output this package is useful to person input/output operation on file.

For Example: File, FileWriter, FileReader etc.

java.applet: This package mainly use to develop GUI related application. Applet programmer are web related program created at server but executed at client machine.

For Example: Applet.

java.awt: awt stands for abstract GUI application. The only difference between applet & awt program is, awt program are stand alone program & it contain main() unlike applet.

For Example: Frame, Button, textField etc.

java.net: URL, InetAddress . URL Connection etc.

java.sql: Connection, Statement, Resultset etc.

Java.Swing: JFrame, JButton, JTextField etc.

User defined packages: User-defined packages are those packages that are designed or created by the developer to categorize classes and packages. They are much similar to the built-in that java offers. It can be

imported into other classes and used the same as we use built-in packages. But If we omit the package statement, the class names are put into the default package, which has no name.

To create a package, we're supposed to use the package keyword.

Syntax:

package package-name;

For Example:

```
package example;
public class gfg {
    public void show()
    {
        System.out.println("Hello geeks!! How are you?");
    }
    public static void main(String args[])
    {
        gfg obj = new gfg();
        obj.show();
    }
 }
```

Access Modifier	within class	within package	outside package by subclass only	outside package
Private	Y	N	N	N
Default	Y	Y	N	N
Protected	Y	Y	Y	N
Public	Y	Y	Y	Y

There are four types of Java access modifiers:

Private: The access level of a private modifier is only within the class. It cannot be accessed from outside the class.

For Example:

```
class A
{
    private static  int d=22;
    private void  show(int s)
```

```
    {
        s=d;

        System.out.println("hello this is private message");
    }
    public static void main (String [] args)
    {
        A ab=new A();

        System.out.println("value of d="+d);
    }
}
```

For Example: we have created two classes A and B. A class contains private data member and private method. We are accessing these private members from outside the class, so there is a compile-time error.
Note: (In this program create error because private method not run in another class.)

```
class A
{
    private static  int d;
    private void  show(int s)
    {       d=s;
        System.out.println("hello this is private message");
        System.out.println("value of d="+d);
    }
}
class B
{
    public static void main (String [] args)
    {
        A ab=new A();
        ab.show(12);
    }
}
```

Note: If you make any class constructor private, you cannot create the instance of that class from outside the class.

For example:

```
class A{
    private A()
    {
System.out.println("hello this is constructor");
    }        //private constructor
    void msg()
    {
        System.out.println("Hello java");
    }
    }
    public class Simple
    {
     public static void main(String args[])
     {
       A obj=new A();
       obj.show();//Compile Time Error
     }
    }
```

Default: The access level of a default modifier is only within the package and class. It cannot be accessed from outside the package. If you do not specify any access level, it will be the default.

For Example:

```
 class A
{
    int d;
   void  show(int s)
    {       d=s;
        System.out.println("hello this is private message");
        System.out.println("value of d="+d);
    }

    public static void main (String [] args)
    {
        A ab=new A();
        ab.show(12);
```

```
    }
}
```

Example: part:2: In this Example show as the default modifier is run in the within package.

```
package goat;

 class sheep {
    void show()
    {
        System.out.println("This is last package");
    }
    public static void main(String[] args)
    {
        System.out.println("hello this is third package");
    sheep sp=new sheep();
    sp.show();
    }
}
```

Example: part:3: In this Example show as the default modifier is not run in the another package.

```
//save by A.java
package pack;
class A{
  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();            //Compile Time Error
   obj.msg();                 //Compile Time Error
  }
```

```
}
```

Protected: The access level of a protected modifier is within the package and outside the package through child class. If you do not make the child class, it cannot be accessed from outside the package.

For Example: Note(it is show as the protected modifier is run within class).

```
class A
{
   int d;
  protected void  show(int s)
   {     d=s;
      System.out.println("hello this is protected message");
      System.out.println("value of d="+d);
   }

   public static void main (String [] args)
   {
      A ab=new A();
      ab.show(12);
   }
}
```

Example part:2 : Note(it is show as the protected modifier is run another class).

```
class A
{
    int d;
   protected void  show(int s)
    {       d=s;
        System.out.println("hello this is protected message");
        System.out.println("value of d="+d);
    }
}
class B{
    public static void main (String [] args)
```

```
    {
        A ab=new A();
        ab.show(12);
    }
  }
```

Example part:3 : Note(it is show as the protected modifier is run package).

```
package goat;

 class sheep {
   protected void show()
    {
        System.out.println("This is last package");
    }
    public static void main(String[] args)
    {
        System.out.println("hello this is third package");
    sheep sp=new sheep();
    sp.show();
    }
}
```

Example part:3 : Note(it is show as the protected modifier is run outside package).

Note: First Package:

```
package goat;
public class hit
{
    protected void display()
    {
        System.out.println("hello this is fourth package");
    }
}
```

Note: second package:

```
package goat;

 class sheep {
   protected void show()
```

```
    {
        System.out.println("This is last package");
    }
    public static void main(String[] args)
    {
        System.out.println("hello this is third package");
    sheep sp=new sheep();
    sp.show();
    hit ht=new hit();
    ht.display();
    }
}
```

Example: Note: In this example, we have created the two packages bank and goat. The A class of bank package is public, so can be accessed from outside the package. But print() method of this package is declared as protected, so it can be accessed from outside the class only through inheritance. (In this program protected modifier run in outside the package in sub class , with use inheritance.)

Part one:

```
package bank;
public class money
{
    protected void print()
    {
        System.out.println("Hello this is program for money");
    }
}
```

Part two:

```
package goat;
import bank.*;
 class sheep extends money
  {
   protected void show()
    {
        System.out.println("This is last package");
    }
```

```
    public static void main(String[] args)
    {
        System.out.println("hello this is third package");
    sheep sp=new sheep();
    sp.show();
    hit ht=new hit();
    ht.display();
    sheep mn=new sheep();
   mn.print();
    }
}
```

Part three: Note:(in this program show as the protected modifier is not run the outside the package without using inheritance).

```
package goat;
import bank.*;
 class sheep
  {
   protected void show()
    {
        System.out.println("This is last package");
    }
    public static void main(String[] args)
    {
        System.out.println("hello this is third package");
    sheep sp=new sheep();
    sp.show();
    hit ht=new hit();
    ht.display();
    money mn=new money();
   mn.print();
    }
}
```

Public: The access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package.

For Example:

```
public class A
{
    int d;
  public void  show(int s)
    {       d=s;
        System.out.println("hello this is public message");
        System.out.println("value of d="+d);
    }
}
class B{
    public static void main (String [] args)
    {
        A ab=new A();
        ab.show(12);
    }
  }
```

Example: Note(In this program run in another package).

Part: one

```
package bank;
public class money
{
    public void print()
    {
        System.out.println("Hello this is program for money");
    }
}
```

Part: two

```
package goat;
import bank.*;
 class sheep
  {
   protected void show()
    {
        System.out.println("This is last package");
```

```
    }
    public static void main(String[] args)
    {
        System.out.println("hello this is third package");
    sheep sp=new sheep();
    sp.show();
    hit ht=new hit();
    ht.display();
    money mm=new money();
    mm.print();
    }
}
```