**Chapter 3 - Unit 2**

**Packages**

* A **package** is a group of similar types of classes, interfaces and sub-packages.
* There are many built-in packages such as java, lang, awt, javax, swing, net, io, util, sql etc.
* **Flow chart :**



* **Advantage of Java Package:**

1) Package is used to categorize the classes and interfaces so that they can be easily maintained.

2) Package provides access protection.

3) Package removes naming collision.

**Defining the Package or Creating a Package**

* **This is the general form of the package statement:**

package pkg;

**Here, pkg is the name of the package.**

* **For example, the following statement creates a package called MyPackage.**

package MyPackage;

* **Example 1 :**

package mypack1;

class Sample

{

public static void main(String args[])

{

System.out.println("Welcome to package concept");

}

}

**Compile :** Z:\>javac -d . sample.java

**Run:** Z:\>java mypack1.sample

**Output :**  Welcome to package concept

* **Note :** 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.
* 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).
* **Example** 2 **:**

package mypack2;

class Sample

{

public static void main(String args[])

{

System.out.println("Welcome to package concept");

}

}

**Compile :**

Z:\>javac -d . sample.java

**Run:**

Z:\>java mypack2.sample

**Output :**

Welcome to package concept

**Importing packages**

**// Import a single class**

import package.name.Class;

**Example :** import java.util.Date;

**// Import the whole package**

import package.name.\*;

**Example :** import java.io.\*;

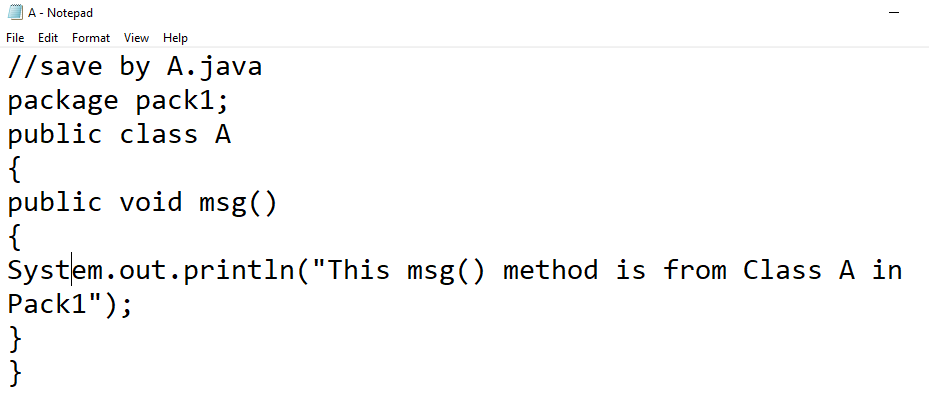
**How to access package from another package?**

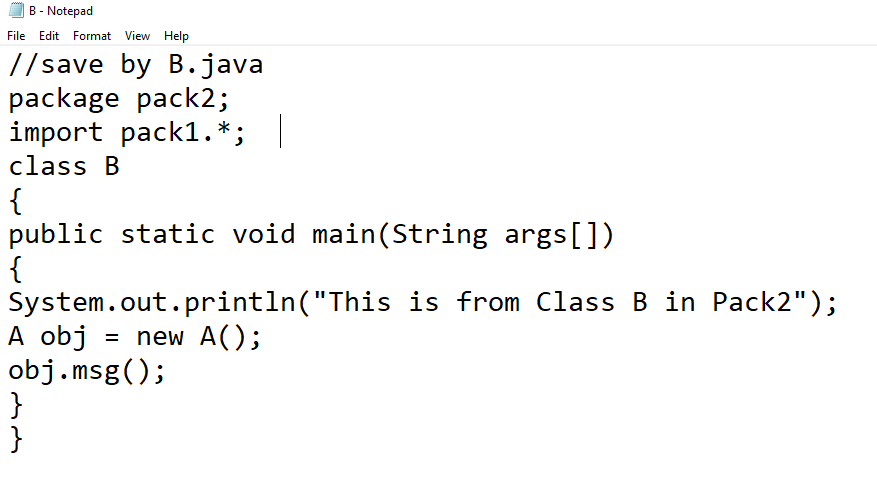
* There are three ways to access the package from outside the package.

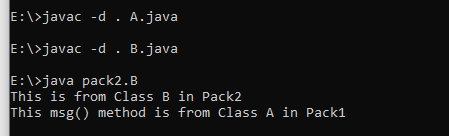
1. import package.\*;
2. import package.classname;
3. fully qualified name.

#### 1) Using packagename.\*

* 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.
* **Example :**



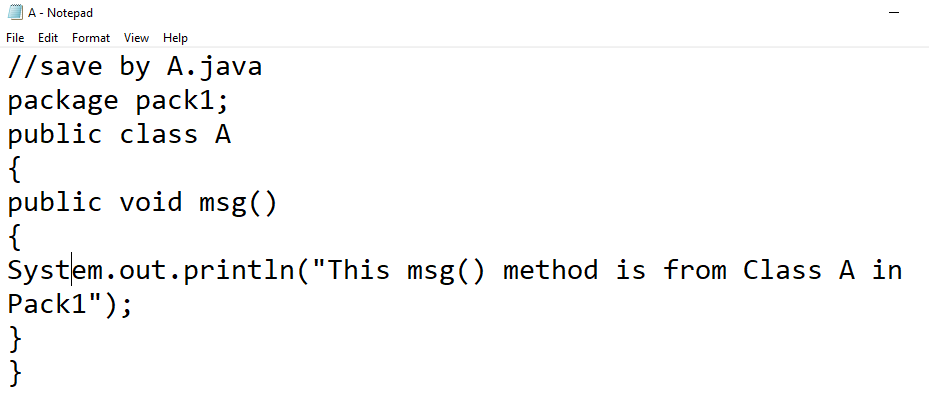


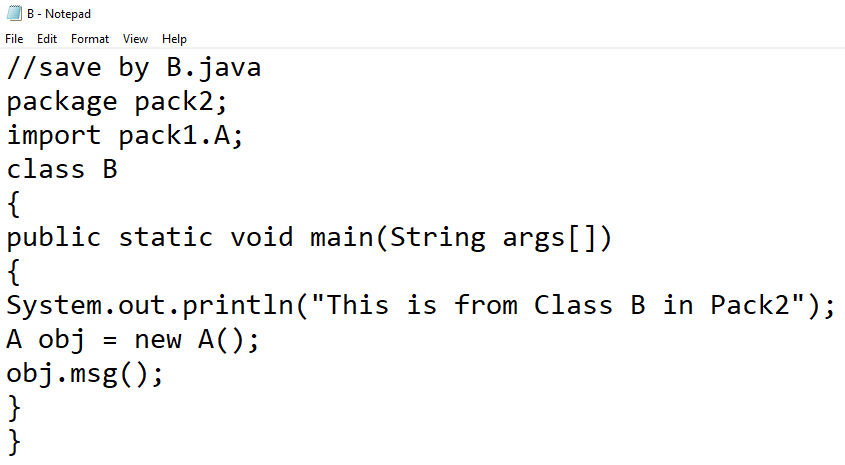


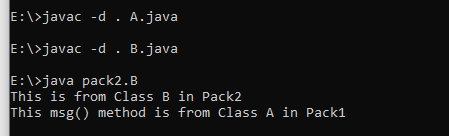
2) Using packagename.classname

If you import package.classname then only declared class of this package will be accessible.

**Example :**





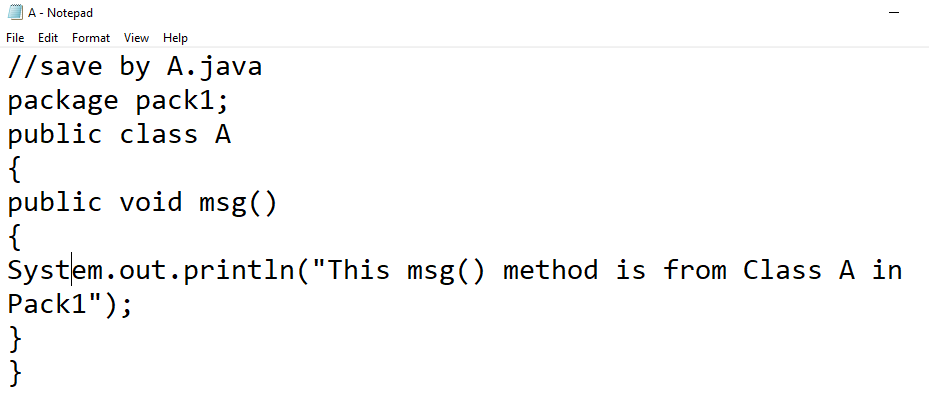


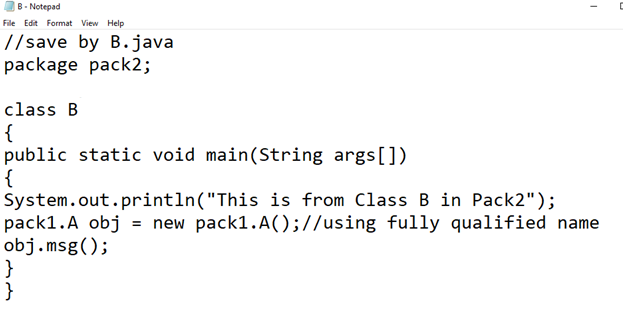
3) Using fully qualified name

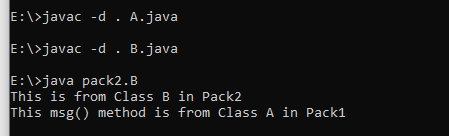
If you use fully qualified name then only declared class of this package will be accessible. Now there is no need to import. But you need to use fully qualified name every time when you are accessing the class or interface.

It is generally used when two packages have same class name e.g. java.util and java.sql packages contain Date class.

**Example :**



****



**Setting CLASSPATH**

* **What is PATH**
* PATH is an environment variable which is used by Operating System to locate the exe files (.exe) or java binaries ( java or javac command).
* **What is CLASSPATH**
* CLASSPATH is also an environment variable which is used by Application ClassLoader to locate and load the .class files. CLASSPATH setting is done to locate the .class files found in another directory.
* Classpath is a parameter in the Java Virtual Machine or the Java compiler that specifies the location of user-defined classes and packages. The parameter may be set either on the command-line, or through an environment variable.
* To check our CLASSPATH on Windows we can open a command prompt and type echo %CLASSPATH%. To check it on a Mac you need to open a terminal and type echo $CLASSPATH.

**Set path**

* Assuming you have installed Java in c:\Program Files\java\jdk directory −
* Right-click on 'My Computer' and select 'Properties'.
* Click the 'Environment variables' button under the 'Advanced' tab.
* Now, alter the 'Path' variable so that it also contains the path to the Java executable. Example, if the path is currently set to 'C:\WINDOWS\SYSTEM32', then change your path to read 'C:\WINDOWS\SYSTEM32;c:\Program Files\java\jdk\bin'.

**Set Classpath**

* Assuming you have stored your Java programs in c:\myprograms\ directory −
* Right-click on 'My Computer' and select 'Properties'.
* Click the 'Environment variables' button under the 'Advanced' tab.
* Now, add the 'CLASSPATH' variable and set the path to the c:\myprograms\'.

**How to Set PATH and CLASSPATH in Windows and Linux/Unix**

**Command to set PATH and CLASSPATH in Windows**

* **PATH:**

set PATH=%PATH%;C:\Program Files\Java\JDK1.8.20\bin

* **CLASSPATH:**

Set CLASSPATH=%CLASSPATH%;C:\Program Files\Java\JDK1.8.20\lib

**Difference between PATH and CLASSPATH**

|  |  |
| --- | --- |
| **PATH** | **CLASSPATH** |
| PATH is an environment variable. | CLASSPATH is also an environment variable. |
| It is used by the operating system to find the executable files (.exe). | It is used by Application ClassLoader to locate the .class file. |
| You are required to include the directory which contains .exe files. | You are required to include all the directories which contain .class and JAR files. |
| PATH environment variable once set, cannot be overridden. | The CLASSPATH environment variable can be overridden by using the command line option -cp or -CLASSPATH to both javac and java command. |

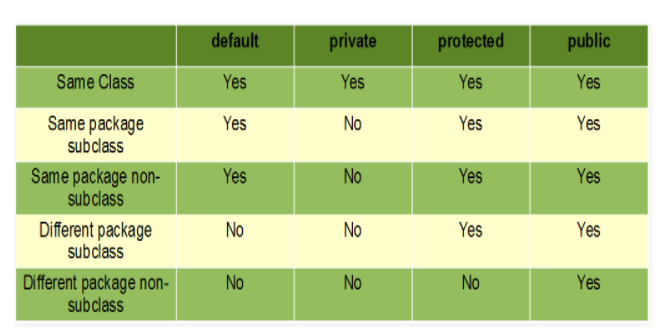
**Modifiers**

* There are two types of modifiers in Java: **access modifiers** and **non-access modifiers**.
* **Non - access modifiers:** we have 7 non-access modifiers. They are used with classes, methods, variables, constructors etc to provide information about their behaviour to JVM.They are :
* [Static](https://www.geeksforgeeks.org/static-keyword-java/)
* [Final](https://www.geeksforgeeks.org/final-keyword-java/)
* Abstract
* [Synchronized](https://www.geeksforgeeks.org/synchronized-in-java/)
* [Transient](https://www.geeksforgeeks.org/transient-keyword-java/)
* [Volatile](https://www.geeksforgeeks.org/volatile-keyword-in-java/)
* [Native](https://www.geeksforgeeks.org/native-keyword-java/)

**Access Modifiers or Access control protection**

* The access modifiers specifies the accessibility or scope of a field, method, constructor, or class.
* Access modifiers in Java helps to restrict the scope of a class, constructor, variable, method, or data member.
* We can change the access level of fields, constructors, methods, and class by applying the access modifier on it.
* The types of access modifiers:

1. Default – No keyword required
2. Private
3. Protected
4. Public



**The description of access modifiers:**

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

**Example 1:**

**A.java**

package p1;

class A

{

privateint data=40;

private void msg()

{

System.out.println("This is private method of same package with same class:msg()");

}

public static void main(String args[])

{

A obj=new A();

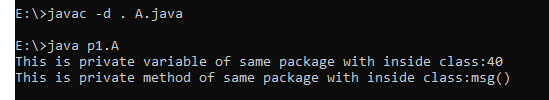
System.out.println("This is private variable of same package with same class:"+obj.data);

obj.msg();

}

}

**Output:**

****

**Example 2:**

**Ademo.java**

package p1;

class A

{

privateint data=40;

private void msg()

{

System.out.println("This is private method of same package with outside class:msg()");

}

}

classAdemo

{

public static void main(String args[])

{

A obj=new A();

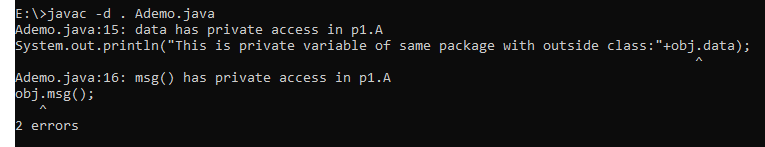
System.out.println("This is private variable of same package with outside class:"+obj.data);

obj.msg();

}

}

**Output: Compile time Error**

****

1. **Default**: The access level of a default modifier is only within the package. It cannot be accessed from outside the package. If you do not specify any access level, it will be the default. (When no access modifier is specified for a particular class, method or a data member, it is said to be having the *default*access modifier).

**Example 1:**

**A.java**

package p1;

class A

{

int data=40;

voidmsg()

{

System.out.println("This is default method of same package with same class:msg()");

}

public static void main(String args[])

{

A obj=new A();

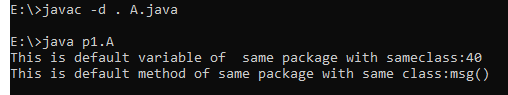
System.out.println("This is default variable of same package with sameclass:"+obj.data);

obj.msg();

}

}

**Output:**

****

**Example 2:**

**demo.java**

package p1;

class A

{

int data=40;

voidmsg()

{

System.out.println("This is default method of same package with sub class:msg()");

}

}

class B extends A

{

}

class demo

{

public static void main(String args[])

{

B obj=new B();

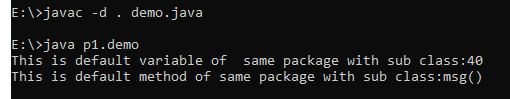
System.out.println("This is default variable of same package with sub class:"+obj.data);

obj.msg();

}

}

**Output:**

****

**Example 3:**

**demo.java**

package p1;

class A

{

int data=40;

voidmsg()

{

System.out.println("This is default method of same package Non sub class:msg()");

}

}

class demo

{

public static void main(String args[])

{

A obj=new A();

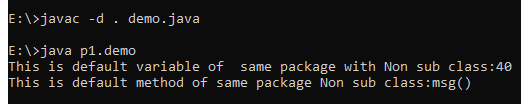
System.out.println("This is default variable of same package with Non sub class:"+obj.data);

obj.msg();

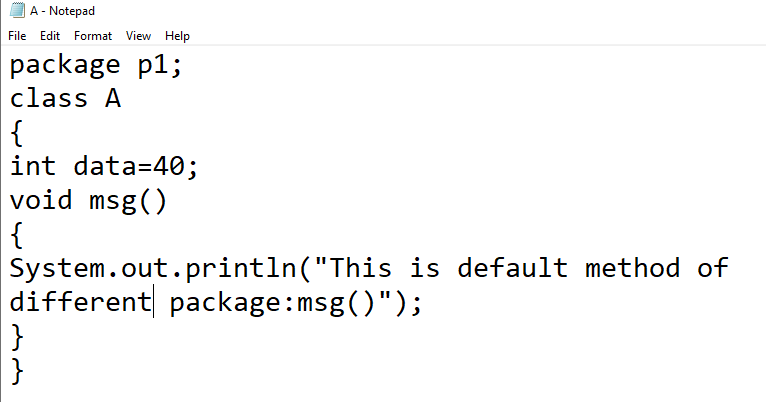
}

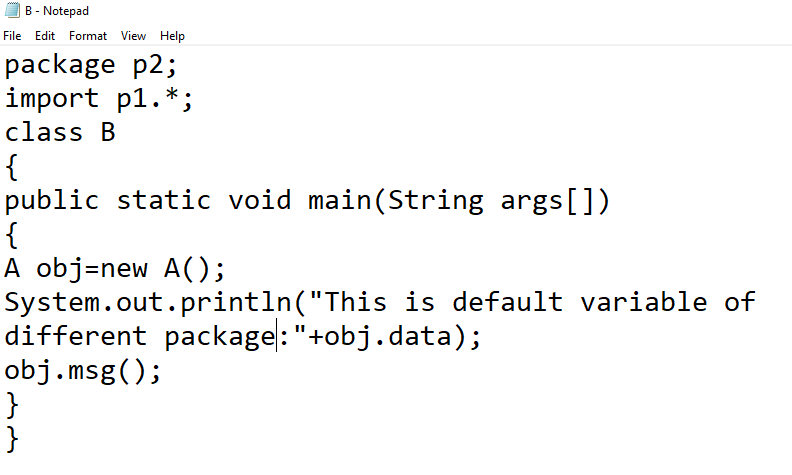
}

**Output:**

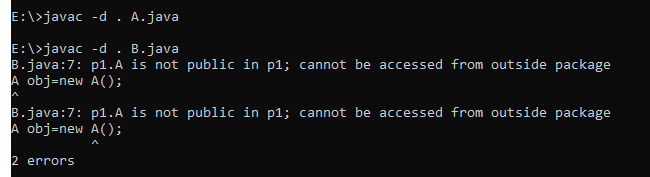
****

**Example 4:**





**Output: Compile time Error**



1. **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.

**Example 1:**

**A.java**

package p1;

class A

{

protectedint data=40;

protected void msg()

{

System.out.println("This is protected method of same package with in same class:msg()");

}

public static void main(String args[])

{

A obj=new A();

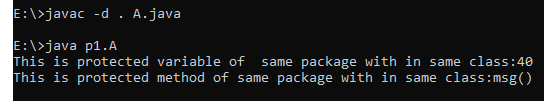
System.out.println("This is protected variable of same package with in same class:"+obj.data);

obj.msg();

}

}

**Output:**



**Example 2:**

**demo.java**

package p1;

class A

{

protectedint data=40;

protected void msg()

{

System.out.println("This is protected method of same package with sub class:msg()");

}

}

class B extends A

{

}

class demo

{

public static void main(String args[])

{

B obj=new B();

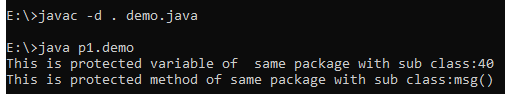
System.out.println("This is protected variable of same package with sub class:"+obj.data);

obj.msg();

}

}

**Output:**

****

**Example 3:**

**B.java**

package p1;

class A

{

protectedint data=40;

protected void msg()

{

System.out.println("This is protected method of same package with Non sub class:msg()");

}

}

class B

{

public static void main(String args[])

{

A obj=new A();

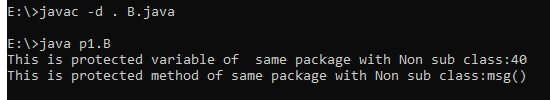
System.out.println("This is protected variable of same package with Non sub class:"+obj.data);

obj.msg();

}

}

**Output:**

****

**Example 4:**

**A.java**

package p1;

public class A

{

protected int data=40;

protected void msg()

{

System.out.println("This is protected method of different package with sub class:msg()");

}

}

**B.java**

package p2;

import p1.\*;

class B extends A

{

public static void main(String args[])

{

B obj=new B();

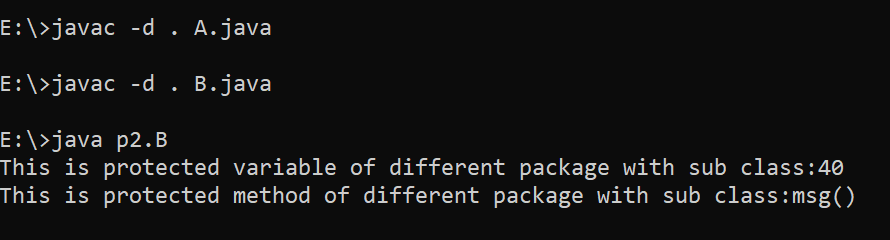
System.out.println("This is protected variable of different package with sub class:"+obj.data);

obj.msg();

}

}

**Output:**

****

**Example 5:**

**A.java**

package p1;

public class A

{

protected int data=40;

protected void msg()

{

System.out.println("This is protected method of different package with Non sub class:msg()");

}

}

**C.java**

package p2;

import p1.\*;

class B extends A

{

}

class C

{

public static void main(String args[])

{

B obj=new B();

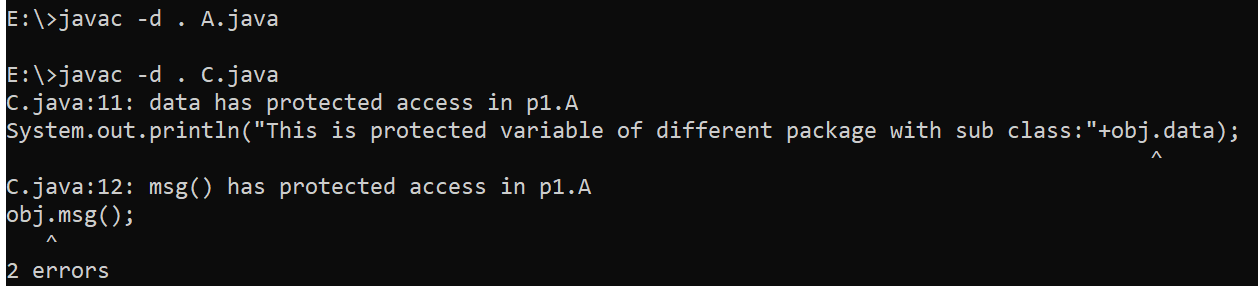
System.out.println("This is protected variable of different package with Non sub class:"+obj.data);

obj.msg();

}

}

**Output: Compile Time Error**

****

1. **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.

**Example 1:**

**A.java**

package p1;

public class A

{

public int data=40;

public void msg()

{

System.out.println("This is public method of different package with Non sub class:msg()");

}

}

**C.java**

package p2;

import p1.\*;

class B extends A

{

}

class C

{

public static void main(String args[])

{

B obj=new B();

System.out.println("This is public variable of different package with Non sub class:"+obj.data);

obj.msg();

}

}

**Output:**

