**//access\_package with\_package\_dot\_asterisk**

// A java package is a group of similar types of

// classes, interfaces and sub-packages.

**package** pack;

**public** **class** A{

**public** **void** msg(){System.out.println("Hello");}

}

// ------------------- FILE 2 --------------------

//save by B.java

**package** mypack;

**import** pack.\*;

**class** B{

**public** **static** **void** main(String args[]){

A obj = **new** A();

obj.msg();

}

}

**//access\_package\_with package\_dot\_classname**

// A java package is a group of similar types of

// classes, interfaces and sub-packages.

**package** pack;

**public** **class** A{

**public** **void** msg(){System.out.println("Hello");}

}

// ------------------- FILE 2 --------------------

//save by B.java

**package** mypack;

**import** pack.A;

**class** B{

**public** **static** **void** main(String args[]){

A obj = **new** A();

obj.msg();

}

}

**//access\_package\_using\_fully\_qualified\_name**

// A java package is a group of similar types of

// classes, interfaces and sub-packages.

**package** pack;

**public** **class** A{

**public** **void** msg(){System.out.println("Hello");}

}

// ------------------- FILE 2 --------------------

//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();

}

}

**// sub\_package\_example**

// A java package is a group of similar types of

// classes, interfaces and sub-packages.

**package** com.practice;

**class** Simple{

**public** **static** **void** main(String args[]){

System.***out***.println("Hello subpackage");

}

}

//there\_can\_be\_only\_one\_public\_class\_with\_same\_name\_as\_file

//save as C.java otherwise Compile Time Error

**class** A{}

**class** B{}

**public** **class** C{}

// putting two public classes in a package

// ------------------- FILE 1 --------------------

//save as A.java

package com.practice;

public class A{}

// ------------------- FILE 2 --------------------

//save as B.java

package com.practice;

public class B{}

// The private access modifier is accessible only within class.

**class** A{

**private** **int** data=40;

**private** **void** msg(){System.***out***.println("Hello java");}

}

**public** **class** Simple{

**public** **static** **void** main(String args[]){

A obj=**new** A();

System.***out***.println(obj.data);//Compile Time Error

obj.msg();//Compile Time Error

}

}

// If you make any class constructor private,

// you cannot create the instance of that class from outside the class.

**class** A{

**private** A(){}//private constructor

**void** msg(){System.***out***.println("Hello java");}

}

**public** **class** Simple{

**public** **static** **void** main(String args[]){

A obj=**new** A();//Compile Time Error

}

}

// A class cannot be private or protected except nested class.

// ------------------- FILE 1 --------------------

//save by A.java

package pack;

class A{

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

}

// ------------------- FILE 2 --------------------

//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

}

}

// A class cannot be private or protected except nested class.

// ------------------- FILE 1 --------------------

//save by A.java

package pack;

public class A{

protected void msg(){System.out.println("Hello");}

}

// ------------------- FILE 2 --------------------

//save by B.java

package mypack;

import pack.\*;

class B extends A{

public static void main(String args[]){

B obj = new B();

obj.msg();

}

}

// A class cannot be private or protected except nested class.

// ------------------- FILE 1 --------------------

//save by A.java

package pack;

public class A{

public void msg(){System.out.println("Hello");}

}

// ------------------- FILE 2 --------------------

//save by B.java

package mypack;

import pack.\*;

class B{

public static void main(String args[]){

A obj = new A();

obj.msg();

}

}

// If you are overriding any method,

// overridden method (i.e. declared in subclass) must not be more restrictive.

**class** A{

**protected** **void** msg(){System.***out***.println("Hello java");}

}

**public** **class** Simple **extends** A{

**void** msg(){System.***out***.println("Hello java");}//C.T.Error

**public** **static** **void** main(String args[]){

Simple obj=**new** Simple();

obj.msg();

}

}

// Encapsulation in java is a process of wrapping code and data together into a single unit

// By providing only setter or getter method, you can make the class read-only or write-only.

// ------------------- FILE 1 --------------------

//save as Student.java

package com.practice;

public class Student{

private String name;

public String getName(){

return name;

}

public void setName(String name){

this.name=name

}

}

// ------------------- FILE 2 --------------------

//save as Test.java

package com.practice;

class Test{

public static void main(String[] args){

Student s=new Student();

s.setName("vijay");

System.out.println(s.getName());

}

}