**Packages**

A package is a grouping (or folder structure) of related classes/ interfaces/ enumerations or annotation types providing access protection and name space management.

This helps in:

* Grouping of related type like classes or interfaces.
* Avoiding name conflicts with classes created in same or other projects in the same company.
* You can allow types within the package to have unrestricted access to one another yet still restrict access for types outside the package.

com.qspider.pack1

com.qspider.pack2

B

B

A

|  |  |
| --- | --- |
| package com.qspiders.pack1;  public class A  {  int i;  public void print(){  //System.out.println("i = "+i);  System.out.println("print() method of class A");  }  public static void main(String args[]){  System.out.println("class A package com.qspiders.pack1");  A a1 = new A();  a1.print();  B b1 = new B();  b1.disp();  }  } | package com.qspiders.pack1;  class B{    void disp(){  System.out.println("class B of package com.qspiders.pack1 disp() methods");  }  } |

For compilation:

**\src> javac –d ..\bin com\qspiders\pack1\JavaFileName.java**

For execution

**\bin> java com.qspiders.pack1.(.classFileName)**

**Access specifier:** The access specifier in java indicates the visibility of the members of the class. Java support following.

1. private
2. default (without any keyword)
3. protected
4. public

**Private:**

The private (most restrictive) fields or methods cannot be used for classes and Interfaces. Fields, methods or constructors declared private are strictly controlled, which means they cannot be access by anywhere outside the class.

Example:

**package** com.qsp.app1;

**class** A{

**private** **int** i=10;

**void** print(){

System.out.println("i= " + i);

}

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

A a1=**new** A();

a1.print();

a1.i=20;

a1.print();

}

}

//Separate file Run1.java

package com.qsp.app1;

class Run1{

public static void main(String args[]){

A a1 = new A();

a1.print();

a1.i=100;//CTE

a1.print();

}

}

we cannot access a private variable from a different class of same/another package.

packages com.qsp.pack1;

**class** PrivateConstructor {

**private** PrivateConstructor(){

System.***out***.println("PrivateConstructor() constructor");

}

**void** display(){

System.***out***.println("class Private Constructor(Singletone class)");

}

}

**public** **class** Run{

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

PrivateConstructor pc = new PrivateConstructor(); //CTE

}

}

**IMPORTANT POINTS OF PRIVATE:**

1. Private can be used either for variables/methods/constructor.

2. Private members has visibility within same class, private members cannot be accessed outside the class i.e. from different class of same/other package.

3. If the constructor has given private access, the object is created only within the same class. We cannot create an object using private constructor in different class.

**NOTE:**

A Class can be declared either as public or as default (no keyword), it cannot be declared as private or protected.

While developing a software, the class files related to similar operations are stored in a Package which is called java package. So, this java package consists of

class files which can be used in other packages.

The java package naming is always done by company website's or domain reverse URl. Suppose, If the company website URL is "qspiders.com" then package name will be "com.qspiders.package".

In one package classes of same name is not allowed, whereas same class name can be allowed in multiple packages.

A class in one package (package com.qsp.app1;) can access class from another package (package com.qsp.app2;) by using either fully Qualified class name or import statement.

Fully Qualified class name consists of Package name and class name Ex: com.qsp.pack1.A

Example program to view fully qualified class name

print()

disp()

main()

A

B

A

pack2

pack1

**package** pack1;

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

**int** i = 987;

**public** **void** print(){

System.***out***.println("i = " + i);

}

}

**package** pack2;

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

**int** j=654;

**void** disp(){

System.***out***.println("j = " + j);

}

}

**package** pack2;

//import pack1.A;

**public** **class** B {

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

//A a1 = new A();

//a1.print();

pack1.A a2 = **new** pack1.A();

a2.print();

pack2.A a3 = **new** pack2.A();

a3.disp();

}

}

Note:

We can access a class in pack 1 from another package pack 2 by using fully qualified class name, provided that class which is there in pack 1 should be declared as **public**. And also if we are accessing a method in a class of pack1 from another package pack2 then both class and method should be declared as **public**.

**package** pack1;

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

**public** **int** i = 987;

**public** A(){

System.out.println("No argument constructor");

}

**void** print(){

System.***out***.println("i = " + i);

}

}

**package** pack2;

**import** pack1.A;//1st way

**public** **class** B {

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

A a1 = **new** A();

a1.print();

//pack1.A a2 = new pack1.A();//2nd way

a2.print();

}

}

**DEFAULT:**

* Default can be applied to either class/method/variables/Constructor.
* Default members have visibility within same class and within same package.
* Default members cannot be accessed from different package.

If we don’t declare any (access specifier) keyword it becomes default.  
For class name, if we mention simply “class A” in a program compiler will take it as “default class A”

For variable, if we mention simply “int i” in a program compiler will take it as “default int i”.

For method, if we mention simply “ returntype method-name()” then compiler will take it as “default returntype method-name();”

For Constructor, if we mention simply constructor “A(){}” in a program compiler will take it as “default A(){}”.

**Program A:**

package com.qsp.pack1;

class C

{

int i=100;

void print()

{

System.out.println("i= " + i);

}

}

class Run1

{

public static void main(String args[])

{

C c1=new C();

c1.print();

c1.i=200;

c1.print();

}

}

Note: we can also write above program in different java file, when we write like that we need to give package in both java files. Either which ways the O/P will be same.

**Program B:**

package com.qsp.pack1;

class C

{

int i=100;

void print()

{

System.out.println("i= " + i);

}

}

Program B:

package com.qsp.pack1;

class Run1

{

public static void main(String args[])

{

C c1=new C();

c1.print();

c1.i=200;

c1.print();

}

}

Note:

We can access the default class/variable/method/constructor from same class and from the same package(provided that we need to specify the package name in the program).

If we want to access the default class /variable /method /constructor from the another package we cannot do it

**Program C:**

package com.qsp.pack2;

import com.qsp.pack1.C;

class Run2

{

public static void main(String args[])

{

C c1=new C();

c1.print();

c1.i=200;

c1.print();

}

}

NOTE:  
While saving any java file, save the file with the class name containing public class

There cannot be two public class in a single java file.

If the class is defined as default then it will have visibility within the package.

If we want to access the class from another package then we should define the class as public and the members of the class should also defined as public.

Constructor access depends on the class access specifier. If the class is public, constructor that java creates will also be public. If we write our own constructor then we have to give public access, to use the class outside the package.

**PROTECTED:**

Protected members behave like default within the same package, and it will not behave as public in another package. The visibility of protected members is between default and public.

Program A:

package com.qsp.pack1;

public class F{

public int j=100;

protected int i=100;

protected void print(){

System.out.println("i= " + i);

}

}

class Run3{

public static void main(String args[]){

F f1=new F();

f1.print();

f1.i=200;

f1.print();

}

}

Note: In the above program, one public class is there and one default class is there, so you should always save the program as “public classname”.java (Ex: F.java), when you compile it two classes will be created 1.F.class, 2.Run 3.class, you have to execute Run3.class because it contains main method

O/P:

C:\practice\_java\bin>java com.qsp.pack1.Run3

i= 100

i= 200

**Note:**

By observing above program and output we can understand that protected members behave likes default with in same package.

Program B:

package com.qsp.pack2;

import com.qsp.pack1.F;

class Run4{

public static void main(String args[]){

F f1=new F();

f1.print();

f1.i=200;

f1.print();

}

}

Note:

Now in the above class Run4 is stored in pack 2 and we are trying to access protected members of class F which is stored in Pack1 i.e., we are trying to create the object of class F and also we are trying to call the print() method from F.java. which is stored pack1 from pack2.

**NOTE:**

By observing the above program we can understand that we cannot access protected from another package.

In order to solve this problem, we have to use inheritance i.e., to correctly access the protected members of one package from another package we need to use inheritance.

Program A:

package com.qsp.pack2;

import com.qsp.pack1.F;

class Run5 extends F// using inheritance

{

public static void main(String args[])

{

Run5 r1=new Run5();

r1.print();

r1.i=100;

r1.print();

}

}

O/P:

C:\practice\_java\bin>java com.qsp.pack2.Run5

i= 100

i= 100

Here, Run5 is extending class and F is extended class

So we need to create new object of extending class not extended class

print() and i are protected members in class and can only be accessed thorough inheritance.

print() and i members acts as private to this class i.e we cannot access it from another class.

Program:

package com.qsp.pack2;

class Run6 extends Run5// using inheritance

{

public static void main(String args[])

{

Run6 r2=new Run6();

r2.print();

r2.i=100;

r2.print();

}

}

Note: we cannot do like this because print() and i are the private members of Run5

O/P:

Compile time error

**IMPORTANT POINTS ON PROTECTED:**

The protected members have the visibility with the class and within the same package as well, but if we want to access the protected members from another package then it can only be done through inheritance

After inheriting, protected members behave like private members to that class, i.e. we cannot access it from another class.