Package & Java Access specifiers – Public, Private, Protected & Default

Package in Java is a mechanism to encapsulate a group of classes, sub packages and interfaces. Packages are used for:

- Preventing naming conflicts. For example there can be two classes with name Employee in two packages, college.staff.cse.Employee and college.staff.ee.Employee
- Making searching/locating and usage of classes, interfaces, enumerations and annotations easier
- Providing controlled access: protected and default have package level access control. A protected member is accessible by classes in the same package and its subclasses. A default member (without any access specifier) is accessible by classes in the same package only.
- Packages can be considered as data encapsulation (or data-hiding).

All we need to do is put related classes into packages. After that, we can simply write an import class from existing packages and use it in our program. A package is a container of a group of related classes where some of the classes are accessible are exposed and others are kept for internal purpose.
We can reuse existing classes from the packages as many time as we need it in our program.

How packages work?

Package names and directory structure are closely related. For example if a package name is college.staff.cse, then there are three directories, college, staff and cse such that cse is present in staff and staff is present college. Also, the directory college is accessible through CLASSPATH variable, i.e., path of parent directory of college is present in CLASSPATH. The idea is to make sure that classes are easy to locate.

Package naming conventions : Packages are named in reverse order of domain names, i.e., org.kiitsca.practice. For example, in a college, the recommended convention is college.tech.cse, college.tech.ee, college.art.history, etc.

Adding a class to a Package : We can add more classes to a created package by using package name at the top of the program and saving it in the package directory. We need a new java file to define a public class, otherwise we can add the new class to an existing .java file and recompile it.

Subpackages: Packages that are inside another package are the subpackages. These are not imported by default, they have to imported explicitly. Also, members of a subpackage have no access privileges, i.e., they are considered as different package for protected and default access specifiers.
Example : java.lang, java.util, java.applet, swing, sql, io, net, awt, javax.servlet

import java.util.*;

util is a subpackage created inside java package.

import java.util.Scanner;

Accessing classes inside a package

Consider following two statements :

```
// import the Vector class from util package.
import java.util.vector;

// import all the classes from util package
import java.util.*;
```

First Statement is used to import Vector class from util package which is contained inside java.
Second statement imports all the classes from util package.

```
// All the classes and interfaces of this package will be accessible but not subpackages.
import package.*;

// Only mentioned class of this package will be accessible.
import package.classname;

// Class name is generally used when two packages have the same class name. For example in
//below code both packages have date class so using a fully qualified name to avoid conflict
import java.util.Date;
import my.packag.Date;

// Java program to demonstrate accessing of members when corresponding classes are imported
and not imported.
import java.util.Vector;

public class ImportDemo
{
  public ImportDemo()
  {
    // java.util.Vector is imported, hence we are able to access directly in our code.
    Vector newVector = new Vector();

    // java.util.ArrayList is not imported, hence we were referring to it using the complete
    // package.
    java.util.ArrayList newList = new java.util.ArrayList();
  }

  public static void main(String arg[])
  {
    new ImportDemo();
  }
}
```

Types of packages:

Built-in Packages

These packages consist of a large number of classes which are a part of Java **API**.Some of the commonly used built-in packages are:

1) **java.lang:** Contains language support classes(e.g classed which defines primitive data types, math operations). This package is automatically imported.

2) **java.io:** Contains classed for supporting input / output operations.
3) **java.util:** Contains utility classes which implement data structures like Linked List, Dictionary and support ; for Date / Time operations.
4) **java.applet:** Contains classes for creating Applets.
5) **java.awt:** Contain classes for implementing the components for graphical user interfaces (like button , ;menus etc).
6) **java.net:** Contain classes for supporting networking operations.
7) **Java.sql:** Contain classes, and interfaces for supporting database connection.

Note: Many more inbuilt packages are available.

User-defined packages

These are the packages that are defined by the user. First we create a directory **myPackage** (name should be same as the name of the package). Then create the **MyClass** inside the directory with the first statement being the **package names**.

```
// Name of the package must be same as the directory under which this file is saved
package myPackage;

public class MyClass
{
   public void getNames(String s)
   {
      System.out.println(s);
   }
}
```

Now we can use the MyClass class in our program.

```
/* import 'MyClass' class from 'names' myPackage */
import myPackage.MyClass;
//import myPackage.*;

public class PrintName
{
  public static void main(String args[])
  {
    // Initializing the String variable
    // with a value
    String name = "KIITSCA";

    // Creating an instance of class MyClass in
    // the package.
    MyClass obj = new MyClass();
    obj.getNames(name);
  }
}
```

Note : MyClass.java must be saved inside the **myPackage** directory since it is a part of the package.

You must have seen public, private and protected keywords while practising java programs, these are called access modifiers. An access modifier restricts the access of a class, constructor, data member and method in another class. In java we have four access modifiers:

1. default
2. private
3. protected
4. public

1. Default access modifier

When we do not mention any access modifier, it is called default access modifier. The scope of this modifier is limited to the package only. This means that if we have a class with the default access modifier in a package, only those classes that are in this package can access this class. No other class outside this package can access this class. Similarly, if we have a default method or data member in a class, it would not be visible in the class of another package. Lets see an example to understand this:

Default Access Modifier Example in Java

To understand this example, you must have the knowledge of packages in java.

In this example we have two classes, Test class is trying to access the default method of Addition class, since class Test belongs to a different package, this program would throw compilation error, because the scope of default modifier is limited to the same package in which it is declared.

Addition.java

```
package abcpackage;

public class Addition {
  /* Since we didn't mention any access modifier here, it would be considered as default. */
  int addTwoNumbers(int a, int b){
        return a+b;
  }
}
```

Test.java

```
package xyzpackage;
/* We are importing the abcpackage but still we will get error because the class we are trying to
use has default access modifier. */
import abcpackage.*;
public class Test {
  public static void main(String args[]){
        Addition obj = new Addition();
```

```
/* It will throw error because we are trying to access the default method in another package*/
        obj.addTwoNumbers(10, 21);
  }
}
```

Output:

```
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
The method addTwoNumbers(int, int) from the type Addition is not visible
at xyzpackage.Test.main(Test.java:12)
```

2. Private access modifier

The scope of private modifier is limited to the class only.

- Private Data members and methods are only accessible within the class
- Class and Interface cannot be declared as private
- If a class has private constructor then you cannot create the object of that class from outside of the class.

Let's see an example to understand this:

Private access modifier example in java

This example throws compilation error because we are trying to access the private data member and method of class ABC in the class Example. The private data member and method are only accessible within the class.

```
class ABC{
  private double num = 100;
  private int square(int a){
        return a*a;
  }
}
public class Example{
  public static void main(String args[]){
        ABC obj = new ABC();
        System.out.println(obj.num);
        System.out.println(obj.square(10));
  }
}
```

Output:

```
Compile - time error
```

3. Protected Access Modifier

Protected data member and method are only accessible by the classes of the same package and the subclasses present in any package. You can also say that the protected access modifier is similar to default access modifier with one exception that it has visibility in sub classes. Classes cannot be declared protected. This access modifier is generally used in a parent child relationship.

Protected access modifier example in Java

In this example the class Test which is present in another package is able to call the **addTwoNumbers()** method, which is declared protected. This is because the Test class extends class Addition and the protected modifier allows the access of protected members in subclasses (in any packages).

Addition.java

```
package abcpackage;
public class Addition {

  protected int addTwoNumbers(int a, int b){
        return a+b;
  }
}
```

Test.java

```
package xyzpackage;
import abcpackage.*;
class Test extends Addition{
  public static void main(String args[]){
        Test obj = new Test();
        System.out.println(obj.addTwoNumbers(11, 22));
  }
}
```

Output:

```
33
```

4. Public access modifier

The members, methods and classes that are declared public can be accessed from anywhere. This modifier doesn't put any restriction on the access.

public access modifier example in java

Lets take the same example that we have seen above but this time the method addTwoNumbers() has public modifier and class Test is able to access this method without even extending the Addition class. This is because public modifier has visibility everywhere.

Addition.java

```
package abcpackage;

public class Addition {

  public int addTwoNumbers(int a, int b){
        return a+b;
  }
}
```

Test.java

```
package xyzpackage;
import abcpackage.*;
class Test{
  public static void main(String args[]){
    Addition obj = new Addition();
    System.out.println(obj.addTwoNumbers(100, 1));
  }
}
```

Output:

```
101
```

The scope of access modifiers in tabular form

	Class	Package	Subclass (same package)	Subclass (diff package)	Outside Class
public	Yes	Yes	Yes	Yes	Yes
protected	Yes	Yes	Yes	Yes	No
default	Yes	Yes	Yes	No	No
private	Yes	No	No	No	No