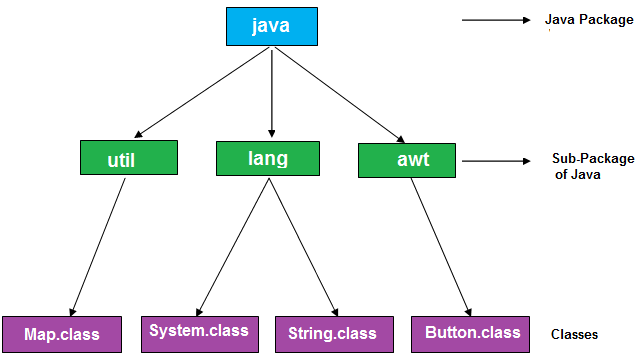
**Package in Java**

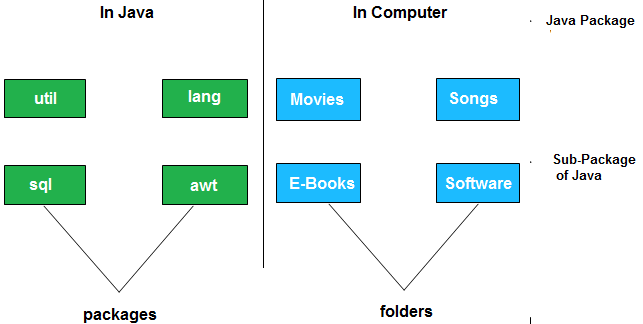
A package is a collection of similar types of classes, interfaces and sub-packages.

**Purpose of package**

The purpose of package concept is to provide common classes and interfaces for any program separately. In other words if we want to develop any class or interface which is common for most of the java programs than such common classes and interfaces must be place in a package.



Packages in Java are the way to organize files when a project has many modules. Same like we organized our files in Computer. For example we store all movies in one folder and songs in other folder, here also we store same type of files in a particular package for example in awt package have all classes and interfaces for design GUI components.



**Advantage of package**

* Package is used to categorize the classes and interfaces so that they can be easily maintained
* Application development time is less, because reuse the code
* Application memory space is less (main memory)
* Application execution time is less
* Application performance is enhance (improve)
* Redundancy (repetition) of code is minimized
* Package provides access protection.
* Package removes naming collision.

**Type of package**

Package are classified into two type which are given below.

1. Predefined or built-in package
2. User defined package

**Predefined or built-in package**

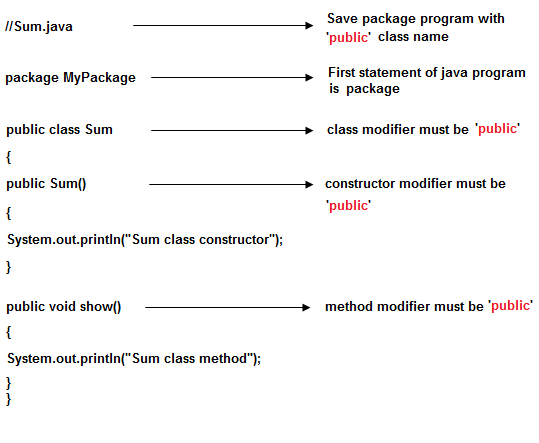
These are the package which are already designed by the Sun Microsystem and supply as a part of java API, every predefined package is collection of predefined classes, interfaces and sub-package.

**User defined package**

If any package is design by the user is known as user defined package. User defined package are those which are developed by java programmer and supply as a part of their project to deal with common requirement.

**Rules to create user defined package**

* package statement should be the first statement of any package program.
* Choose an appropriate class name or interface name and whose modifier must be public.
* Any package program can contain only one public class or only one public interface but it can contain any number of normal classes.
* Package program should not contain any main class (that means it should not contain any main())
* modifier of constructor of the class which is present in the package must be public. (This is not applicable in case of interface because interface have no constructor.)
* The modifier of method of class or interface which is present in the package must be public (This rule is optional in case of interface because interface methods by default public)
* Every package program should be save either with public class name or public Interface name



**Compile package programs**

For compilation of package program first we save program with public className.java and it compile using below syntax:

**Syntax**

javac -d . className.java

**Syntax**

javac -d path className.java

**Explanations:** In above syntax **"-d"** is a specific tool which is tell to java compiler create a separate folder for the given package in given path. When we give specific path then it create a new folder at that location and when we use . (dot) then it crate a folder at current working directory.

**Note:** Any package program can be compile but can not be execute or run. These program can be executed through user defined program which are importing package program.

**Example of package program**

Package program which is save with A.java and compile by javac -d . A.java

**Example**

package mypack;

public class A

{

public void show()

{

System.out.println("Sum method");

}

}

Import above class in below program using import packageName.className

**Example**

import mypack.A;

public class Hello

{

public static void main(String arg[])

{

A a=new A();

a.show();

System.out.println("show() class A");

}

}

**Explanations:** In the above program first we create Package program which is save with A.java and compiled by **"javac -d . A.java"**. Again we import class "A" in class Hello using **"import mypack.A;"** statement.

Example of package by import fully qualified name

//save by A.java

**package** pack;

**public** **class** A{

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

}

//save by B.java

**package** mypack;

**class** B{

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

   pack.A obj = **new** pack.A();//using fully qualified name

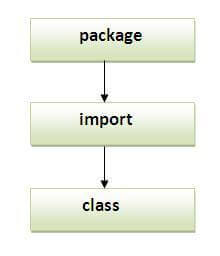
   obj.msg();

  }

}

Output:Hello

#### Note: Sequence of the program must be package then import then class.



## Subpackage in java

Package inside the package is called the **subpackage**. It should be created **to categorize the package further**.

Example of Subpackage

**package** latha.javaprograms;

**class** Simple{

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

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

  }

}

## 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 of package that import the packagename.\*

//save by A.java

**package** pack;

**public** **class** A{

**public** **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();

   obj.msg();

  }

}

Output:Hello

#### 2) Using packagename.classname

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

## Example of package by import package.classname

//save by A.java

**package** pack;

**public** **class** A{

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

}

//save by B.java

**package** mypack;

**import** pack.A;

**class** B{

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

   A obj = **new** A();

   obj.msg();

  }

}

Output:Hello

#### 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.

Example of package by import fully qualified name

//save by A.java

**package** pack;

**public** **class** A{

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

}

//save by B.java

**package** mypack;

**class** B{

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

   pack.A obj = **new** pack.A();//using fully qualified name

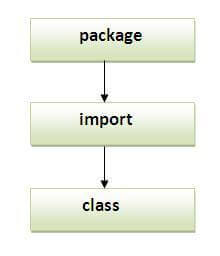
   obj.msg();

  }

}

Output:Hello

#### Note: Sequence of the program must be package then import then class.



## Subpackage in java

Package inside the package is called the **subpackage**. It should be created **to categorize the package further**.

Let's take an example, Sun Microsystem has definded a package named java that contains many classes like System, String, Reader, Writer, Socket etc. These classes represent a particular group e.g. Reader and Writer classes are for Input/Output operation, Socket and ServerSocket classes are for networking etc and so on. So, Sun has subcategorized the java package into subpackages such as lang, net, io etc. and put the Input/Output related classes in io package, Server and ServerSocket classes in net packages and so on.

### Example of Subpackage

**package** latha.programs;

**class** Simple{

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

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

  }

}

|  |
| --- |
| **To Compile:** javac -d . Simple.java |
| **To Run:** java latha.programs.Simple |

Output:Hello subpackage

# Access Modifiers in Java

There are two types of modifiers in Java: **access modifiers** and **non-access modifiers**.

The access modifiers in Java specifies the accessibility or scope of a field, method, constructor, or class. We can change the access level of fields, constructors, methods, and class by applying the access modifier on it.

There are four types of Java access modifiers:

1. **Private**: The access level of a private modifier is only within the class. It cannot be accessed from outside the class.
2. **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.
3. **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.
4. **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.

There are many non-access modifiers, such as static, abstract, synchronized, native, volatile, transient, etc. Here, we are going to learn the access modifiers only.

59.4M

1.1K

C++ vs Java

### Understanding Java Access Modifiers

Let's understand the access modifiers in Java by a simple table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 |

1) Private

The private access modifier is accessible only within the class.

**Simple example of private access modifier**

In this example, we have created two classes A and Simple. 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.

**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

   }

}

2) Default

If you don't use any modifier, it is treated as **default** by default. The default modifier is accessible only within package. It cannot be accessed from outside the package. It provides more accessibility than private. But, it is more restrictive than protected, and public.

**Example of default access modifier**

In this example, we have created two packages pack and mypack. We are accessing the A class from outside its package, since A class is not public, so it cannot be accessed from outside the 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

  }

}

3) Protected

The **protected access modifier** is accessible within package and outside the package but through inheritance only.

The protected access modifier can be applied on the data member, method and constructor. It can't be applied on the class.

It provides more accessibility than the default modifer.

//save by A.java

**package** pack;

**public** **class** A{

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

}

//save by B.java

**package** mypack;

**import** pack.\*;

**class** B **extends** A{

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

   B obj = **new** B();

   obj.msg();

  }

}