**Packages in Java**

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.

**Package naming conventions :** Packages are named in reverse order of domain names, i.e., org.geeksforgeeks.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 :

import java.util.\*;

util is a subpackage created inside java package.

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 only 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;

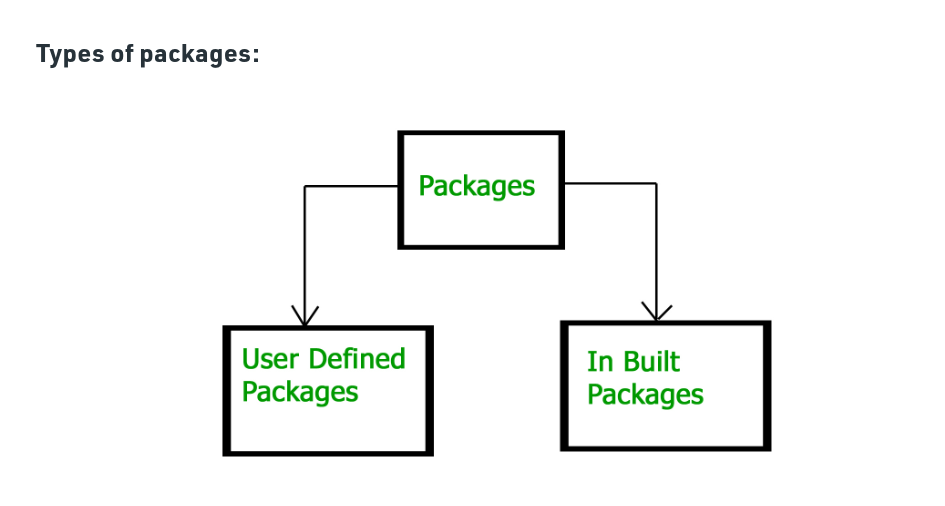
// Class name is generally used when two packages have the same

// 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;// in-built

import my.package.Date; // user defined

**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.

**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.

**Using Static Import**

Static import is a feature introduced in Java programming language ( versions 5 and above ) that allows members ( fields and methods ) defined in a class as public static to be used in Java code without specifying the class in which the field is defined.

Following program demonstrates static import :

// Note static keyword after import.

import static java.lang.System.\*;

class StaticImportDemo

{

public static void main(String args[])

{

// We don't need to use 'System.out'

// as imported using static.

out.println("AIET");

}

}

Output:

AIET

**Handling name conflicts**

The only time we need to pay attention to packages is when we have a name conflict . For example both, java.util and java.sql packages have a class named Date. So if we import both packages in program as follows:

import java.util.\*;

import java.sql.\*;

//And then use Date class, then we will get a compile-time error :

Date today ; //ERROR-- java.util.Date or java.sql.Date?

The compiler will not be able to figure out which Date class do we want. This problem can be solved by using a specific import statement:

import java.util.Date;

import java.sql.\*;

If we need both Date classes then, we need to use a full package name every time we declare a new object of that class.

For Example:

java.util.Date deadLine = new java.util.Date();

java.sql.Date today = new java.sql.Date();

Illustration of user-defined packages:

Creating our first package:

File name – ClassOne.java

package package\_one;

public class ClassOne {

public void methodClassOne() {

System.out.println("Hello there I am ClassOne");

}

}

Creating our second package:

File name – ClassTwo.java

package package\_two;

public class ClassTwo {

public void methodClassTwo(){

System.out.println("Hello there I am ClassTwo");

}

}

Making use of both the created packages:

File name – Testing.java

import package\_one.ClassOne;

import package\_two.ClassTwo;

public class Testing {

public static void main(String[] args){

ClassOne a = new ClassOne();

ClassTwo b = new ClassTwo();

a.methodClassOne();

b.methodClassTwo();

}

}

Output:

Hello there I am ClassOne

Hello there I am ClassTwo

**Important points:**

* Every class is part of some package.
* If no package is specified, the classes in the file goes into a special unnamed package (the same unnamed package for all files).
* All classes/interfaces in a file are part of the same package. Multiple files can specify the same package name.
* If package name is specified, the file must be in a subdirectory called name (i.e., the directory name must match the package name).
* We can access public classes in another (named) package using: package-name.class-name

**Importing Packages**

Given that packages exist and are a good mechanism for compartmentalizing diverse classes from each other, it is easy to see why all of the built-in Java classes are stored in packages. There are no core Java classes in the unnamed default package; all of the standard classes are stored in some named package. Since classes within packages must be fully qualified with their package name or names, it could become tedious to type in the long dot-separated package path name for every class you want to use. For this reason, Java includes the import statement to bring certain classes, or entire packages, into visibility. Once imported, a class can be referred to directly, using only its name. The import statement is a convenience to the programmer and is not technically needed to write a complete Java program. If you are going to refer to a few dozen classes in your application, however, the import statement will save a lot of typing.

In a Java source file, import statements occur immediately following the package statement (if it exists) and before any class definitions. This is the general form of the import statement:

import pkg1 [.pkg2].(classname | \*);

Here, pkg1 is the name of a top-level package, and pkg2 is the name of a subordinate package inside the outer package separated by a dot (.). There is no practical limit on the depth of a package hierarchy, except that imposed by the file system. Finally, you specify either an explicit classname or a star (\*), which indicates that the Java compiler should import the entire package. This code fragment shows both forms in use:



In a Java source file, import statements occur immediately following the package statement (if it exists) and before any class definitions. This is the general form of the import statement:

import pkg1 [.pkg2].(classname | \*);

Here, pkg1 is the name of a top-level package, and pkg2 is the name of a subordinate package inside the outer package separated by a dot (.). There is no practical limit on the depth of a package hierarchy, except that imposed by the file system. Finally, you specify either an explicit classname or a star (\*), which indicates that the Java compiler should import the entire package. This code

fragment shows both forms in use:All of the standard Java SE classes included with Java begin with the name java.

The basic language functions are stored in a package called java.lang. Normally, you have to import every package or class that you want to use, but since Java is useless without much of the functionality in java.lang, it is implicitly imported by the compiler for all programs. This is equivalent to the following line being at the top of all of your programs:

If a class with the same name exists in two different packages that you import using the star form, the compiler will remain silent, unless you try to use one of the classes. In that case, you will get a compile-time error and have to explicitly name the class specifying its package.

It must be emphasized that the import statement is optional. Any place you use a class name, you can use its fully qualified name, which includes its full package hierarchy. For example, this fragment uses an import statement: