**Packages in Java with Example Programs**

we are going to discuss **packages in Java** with example programs.

In small projects, all the java files have unique names. So, it is not difficult to put them in a single folder.

But, in the case of huge projects where the number of java files is large, it is very difficult to put files in a single folder because the manner of storing files would be disorganized.

Moreover, if different java files in various modules of the project have the same name, it is not possible to store two java files with the same name in the same folder because it may occur naming conflict.

This problem of naming conflict can be overcome by using the concept of packages.

In Java, APIs consist of one or more packages where packages consist of many classes, classes contain several methods and [fields](https://www.scientecheasy.com/2020/05/variables-in-java.html/).

When you create an application in Java, you should create a  proper folder structure for better reusability, maintenance, and avoiding naming conflict but How?

## Package in Java

A package is nothing but a physical folder structure (directories) that contains a group of related classes, interfaces, and sub-packages according to their functionality.

It provides a convenient way to organize your work. The Java language has various in-built packages.

For example, java.lang, java.util, java.io, and java.net. All these packages are defined as a very clear and systematic packaging mechanism for categorizing and managing.

Let’s understand it with realtime examples.

## Real-time Example of Packages in Java

A real-life example is when you download a movie, song, or game, you make a different folder for each category like movie, song, etc. In the same way, a group of packages in java is just like a library.

The classes and interfaces of a package are like books in the library that can reuse several times when we need them. This reusability nature of packages makes programming easy.

Therefore, when you create any software or application in Java programming language, they contain hundreds or thousands of individual classes and interfaces.

So, they must be organized into a meaningful package name to make proper sense, and reusing these packages in other programs could be easier.

## Advantage of using packages in Java

1. **Maintenance:** Java packages are used for proper maintenance. If any developer newly joined a company, he can easily reach to files needed.

2. **Reusability:** We can place the common code in a common folder so that everybody can check that folder and use it whenever needed.

3. **Name conflict:** Packages help to resolve the naming conflict between the two classes with the same name. Assume that there are two classes with the same name Student.java.

Each class will be stored in its own packages such as stdPack1 and stdPack2 without having any conflict of names.

4. **Organized:** It also helps in organizing the files within our project.

5. **Access Protection:** A package provides access protection. It can be used to provide visibility control. The members of the class can be defined in such a manner that they will be visible only to elements of that package.

## Types of Packages in Java

There are two different types of packages in Java. They are:

## 1. User-defined Package

The package which is defined by the user is called a User-defined package. It contains user-defined classes and interfaces.

## Creating package in Java

Java supports a keyword called “package” which is used to create user-defined packages in java programming. It has the following general form:

package packageName;

Here, packageName is the name of package. The package statement must be the first line in a java source code file followed by one or more classes.

For example:

package myPackage;

public class A {

// class body

}

### Naming Convention to declare User-defined Package in Real-time Project

While developing your project, you must follow some naming conventions regarding packages declaration. Let’s take an example to understand the convention.

See below a complete package structure of the project.

Diagram

Description automatically generated

1. Suppose you are working in IBM and the domain name of IBM is www.ibm.com. You can declare the package by reversing the domain like this:

package com.ibm;

where,

com ➝ It is generally company specification name and the folder starts with com package com.tcs.icici.loan.carloan.penalty;

**Note:** Keep in mind Root folder should be always the same for all the classes.

## How to create Package in Eclipse IDE?

In Eclipse IDE, there are the following steps to create a package in java. They are as follows:

1. Right-click on the src folder as shown in the below screenshot.

* which is called root folder.
* ibm ➝ Company name where the product is developed. It is the subfolder.

2. hdfc ➝ Client name for which we are developing our product or working for the project.

3. loan ➝ Name of the project.

4. homeloan ➝ It is the name of the modules of the loan project. There are a number of modules in the loan project like a Home loan, Car loan, or Personal loan. Suppose you are working for Home loan module.  
  
This is a complete packages structure like a professional which is adopted in the company.  
Another example is:

package com.tcs.icici.loan.carloan.penalty;

**Note:** Keep in mind Root folder should be always the same for all the classes.

## How to create Package in Eclipse IDE?

Graphical user interface, text, application, email

Description automatically generated

Once the package is created, a package folder will be created in your file system where you can create classes and interfaces.

## 2. Predefined Packages in Java (Built-in Packages)

Predefined packages in java are those which are developed by Sun Microsystem. They are also called built-in packages in java.

These packages consist of a large number of predefined classes, interfaces, and methods that are used by the programmer to perform any task in his programs.

Java APIs contains the following predefined packages, as shown in the below figure:

Diagram

Description automatically generated

### Core packages:

1. **Java.lang:** lang stands for language. The Java language package consists of java classes and interfaces that form the core of the Java language and the JVM. It is a fundamental package that is useful for writing and executing all Java programs.

Examples are classes, objects, String, Thread, predefined data types, etc. It is imported automatically into the Java programs.

2. **Java.io:** io stands for input and output. It provides a set of I/O streams that are used to read and write data to files. A stream represents a flow of data from one place to another place.

3. **Java util:** util stands for utility. It contains a collection of useful utility classes and related interfaces that implement data structures like LinkedList, Dictionary, HashTable, stack, vector, Calender, data utility, etc.

4. **Java.net:** net stands for network. It contains networking classes and interfaces for networking operations. The programming related to client-server can be done by using this package.

### Window Toolkit and Applet:

1. **Java.awt:** awt stands for abstract window toolkit. The Abstract window toolkit packages contain the GUI(Graphical User Interface) elements such as buttons, lists, menus, and text areas. Programmers can develop programs with colorful screens,  paintings, and images, etc using this package.

2. **Java.awt.image:** It contains classes and interfaces for creating images and colors.

3. **Java.applet:** It is used for creating applets. Applets are programs that are executed from the server into the client machine on a network.

4. **Java.text:** This package contains two important classes such as DateFormat and NumberFormat. The class DateFormat is used to format dates and times. The NumberFormat is used to format numeric values.

5. **Java.sql:** SQL stands for the structured query language. This package is used in a Java program to connect databases like Oracle or Sybase and retrieve the data from them.

**Key points:**

1. Java predefined supports a group of packages that contains a group of classes and interfaces. These classes and interfaces consist of a group of methods.

For example, Java language contains a package called java.lang which contains string class, StringBuffer class, StringBuilder class, all wrapper classes,  runnable interface, etc. String class contains a number of methods such as length(), toUpperCase(), toLowerCase() etc.

2. Java contains 14 predefined packages which are main packages. These 14 predefined packages contain nearly 150 sub-packages that consist of a minimum of 7 thousand classes. These 7 thousand classes contain approx 7 lakhs methods.

3. Up to Java 1.7 version contains 13 predefined packages. From Java 1.8 version onwards, one new package is introduced called java.time.

### How to see list of Predefined packages in Java?

Follow the following steps to see the list of predefined packages in Java.

1. Go to programs files and open them.

2. Now go to Java folder and open it. You will see two folders such as JDK and JRE.

3. Go to JDK folder, extract the src folder. After extracting it, go to Java folder. Here, you will see 14 predefined packages folders such as applet, awt, beans, io, lang, math, net, nio, rmi, security, sql, text, time, and util.

4. Now you open lang package and scroll down. You can see classes like String, StringBuffer, StringBuilder, Thread, etc.

## Java Packages Example Program

Let us make an example program where we will create a user-defined package in a systematic manner.

**Program source code 1:**

// Save as Example.java

// Step 1: Declare package name by reversing domain name, project name 'java', and module name is core java.

package com.scientecheasy.java.corejava;

// Step 2: Declare class name.

public class Example

{

public static void main(String[] args)

{

System.out.println(" My first basic example");

}

}

## How to compile package in Java?

If you are not using any Eclipse IDE, you follow the syntax given below:  
**Syntax:**  
To Compile the application:  javac -d directory javafilename

1. Here, javac means java compiler.

2. -d means directory. It creates the folder structure.

3. .(dot) means the current directory. It places the folder structure in the current working directory.

For example: javac -d.Example.java // Here, Example.java is the file name.

So in this way, you must compile application if the application contains a package statement.

After the compilation, you can see the folder structure in your system like this:

com  
  |—> scientecheasy  
      |——> java  
         |——> corejava  
             |——> Example.class

## How to run Java package program?

You have to use the fully qualified name to execute java code. The fully qualified name means class name with a complete package structure. Use the below syntax to run java code.  
**Syntax:**

java completePackageName.className

Now run the above Java code.  
To Run:   java com.scientecheasy.java.corejava.Example

## Importing packages in Java

There are three approaches to import one package into another package in Java.

1. import package.\*;
2. import package.classname;
3. Using fully qualified name.