**How to Create PACKAGE in Java: Learn with Example Program**

**What is Package in Java?**

A Package is a collection of related classes. It helps organize your classes into a folder structure and make it easy to locate and use them. More importantly, it helps improve re-usability.

Each package in Java has its unique name and organizes its classes and interfaces into a separate namespace, or name group.

Although interfaces and classes with the same name cannot appear in the same package, they can appear in different packages. This is possible by assigning a separate namespace to each package.

**Syntax:-**

package nameOfPackage;

The following video takes you through the steps of creating a package.

Click [here](https://www.guru99.com/faq.html#1) if the video is not accessible

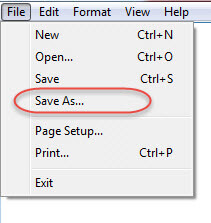
Let's study package with an example. We define a class and object and later compile this it in our package p1. After compilation, we execute the code as a java package.

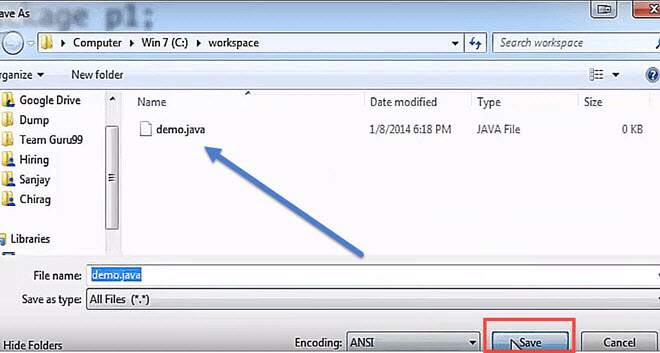
**Step 1)** Consider the following code,

Here,

1. To put a class into a package, at the first line of code define package p1
2. Create a class c1
3. Defining a method m1 which prints a line.
4. Defining the main method
5. Creating an object of class c1
6. Calling method m1

**Step 2)** In next step, save this file as demo.java

[](https://www.guru99.com/images/java/052016_0817_Creatingand2.jpg)

[](https://www.guru99.com/images/java/052016_0817_Creatingand3.jpg)

**Step 3)** In this step, we compile the file.

[](https://www.guru99.com/images/java/052016_0817_Creatingand4.jpg)

The compilation is completed. A class file c1 is created. However, no package is created? Next step has the solution

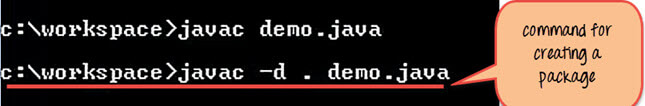
[](https://www.guru99.com/images/java/052016_0817_Creatingand5.jpg)

**Step 4)** Now we have to create a package, use the command

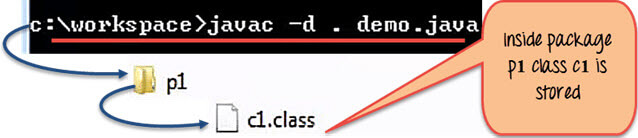
javac –d . demo.java

This command forces the compiler to create a package.

The **"."**operator represents the current working directory.

[](https://www.guru99.com/images/java/052016_0817_Creatingand6.jpg)

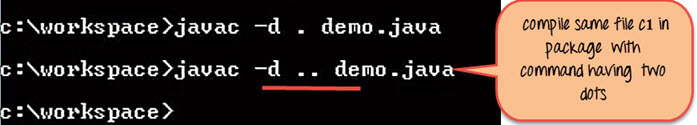
**Step 5)** When you execute the code, it creates a package p1. When you open the java package p1 inside you will see the c1.class file.

[](https://www.guru99.com/images/java/052016_0817_Creatingand7.jpg)

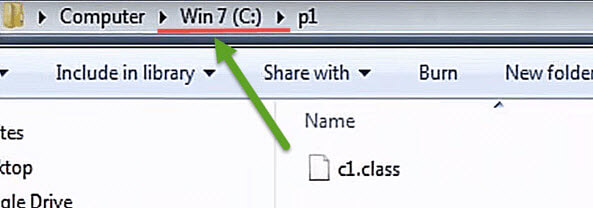
**Step 6)** Compile the same file using the following code

javac –d .. demo.java

Here ".." indicates the parent directory. In our case file will be saved in parent directory which is C Drive

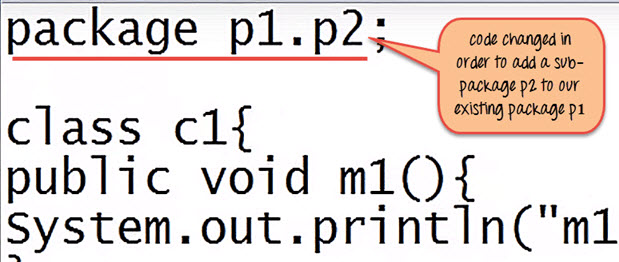
[](https://www.guru99.com/images/java/052016_0817_Creatingand8.jpg)

File saved in parent directory when above code is executed.

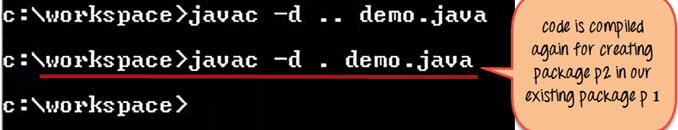
[](https://www.guru99.com/images/java/052016_0817_Creatingand9.jpg)

**Step 7)** Now let's say you want to create a sub package p2 within our existing java package p1. Then we will modify our code as

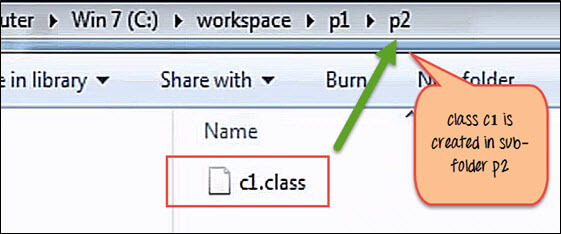
package p1.p2

[](https://www.guru99.com/images/java/052016_0817_Creatingand10.jpg)

**Step 8)** Compile the file

[](https://www.guru99.com/images/java/052016_0817_Creatingand11.jpg)

As seen in below screenshot, it creates a sub-package p2 having class c1 inside the package.

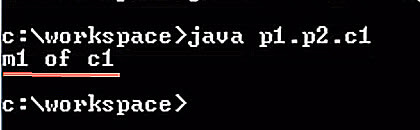
[](https://www.guru99.com/images/java/052016_0817_Creatingand12.jpg)

**Step 9)** To execute the code mention the fully qualified name of the class i.e. the package name followed by the sub-package name followed by the class name -

java p1.p2.c1

[](https://www.guru99.com/images/java/052016_0817_Creatingand13.jpg)

This is how the package is executed and gives the output as "m1 of c1" from the code file.

[](https://www.guru99.com/images/java/052016_0817_Creatingand14.jpg)

**Importing packages**

To create an object of a class (bundled in a package), in your code, you have to use its fully qualified name.

**Example:**

java.awt.event.actionListner object = new java.awt.event.actionListner();

But, it could become tedious to type the long dot-separated package path name for every class you want to use. Instead, it is recommended you use the import statement.

**Syntax**

import packageName;

Once imported, you can use the class without mentioning its fully qualified name.

import java.awt.event.\*; // \* signifies all classes in this package are imported

import javax.swing.JFrame // here only the JFrame class is imported

//Usage

JFrame f = new JFrame; // without fully qualified name.

**Example**: To import package

**Step 1)** Copy the code into an editor.

package p3;

import p1.\*; //imports classes only in package p1 and NOT in the sub-package p2

class c3{

public void m3(){

System.out.println("Method m3 of Class c3");

}

public static void main(String args[]){

c1 obj1 = new c1();

obj1.m1();

}

}

**Step 2)** Save the file as Demo2.java. Compile the file using the command **javac –d . Demo2.java**

**Step 3)**Execute the code using the command **java p3.c3**

**Packages - points to note:**

* To avoid naming conflicts packages are given names of the domain name of the company in reverse Ex: com.guru99. com.microsoft, com.infosys etc.
* When a package name is not specified, a class is in the default package (the current working directory) and the package itself is given no name. Hence you were able to execute assignments earlier.
* While creating a package, care should be taken that the statement for creating package must be written before any other import statements

// not allowed

import package p1.\*;

package p3;

//correct syntax

package p3;

import package p1.\*;

the *java.lang package* is imported by default for any class that you create in Java.