**Packages in Java: How to Create and Use Packages in Java?**

1. What is Package in Java?

2. Built-in Packages

3. User Defined Packages

* Creating a Package in Java
* Including a Class in Java Package
* Creating a class inside package while importing another package
* Using fully qualified name while importing a class

4. Static Import in Java

5. Access Protection in Java Packages

6. Points to Remember

# What is Package in Java?

Java package is a mechanism of grouping similar types of classes, interfaces, and sub-classes collectively based on functionality. When software is written in the Java programming language, it can be composed of hundreds or even thousands of individual classes. It makes sense to keep things organized by placing related classes and interfaces into packages.

Using packages while coding offers a lot of advantages like:

* **Re-usability:** The classes contained in the packages of another program can be easily reused
* help us to uniquely identify a class, for example, we can have **Name Conflicts:** Packages company.sales.Employee and company.marketing.Employee classes
* **Controlled Access:** Offers access protection such as protected classes, default classes, and private class
* **Data Encapsulation:** They provide a way to hide classes, preventing other programs from accessing classes that are meant for internal use only
* **Maintainance:** With packages, you can organize your project better and easily locate related classes

It’s a good practice to use packages while coding in Java. As a programmer, you can easily figure out the classes, interfaces, enumerations, and annotations that are related. We have two types of packages in java.

# Types of Packages in Java

Based on whether the package is defined by the user or not, packages are divided into two categories:

1. Built-in Packages
2. User-Defined Packages

# Built-in Packages

Built-in packages or predefined packages are those that come along as a part of JDK (Java Development Kit) to simplify the task of Java programmer. They consist of a huge number of predefined classes and interfaces that are a part of Java API’s. Some of the commonly used built-in packages are java.lang, java.io, java.util, java.applet, etc. Here’s a simple program using a built-in package.

**package** University;

**import** java.util.ArrayList;

**class** BuiltInPackage {  
**public** **static** **void** main(String[] args) {  
ArrayList<Integer> myList = **new** ArrayList<>(3);

myList.add(3);  
myList.add(2);  
myList.add(1);

System.out.println("The elements of list are: " + myList);  
}  
}

**Output:**

The elements of list are: [3, 2, 1]

The ArrayList class belongs to java.util package. To use it, we have to import the package using the import statement. The first line of the code import java.util.ArrayList imports the java.util package and uses ArrayList class which is present in the subpackage util.

# User-Defined Packages

User-defined packages are those which are developed by users in order to group related classes, interfaces, and sub-packages. With the help of an example program, let’s see how to create packages, compile Java programs inside the packages, and execute them.

## Creating a Package in Java

Creating a package in Java is a very easy task. Choose a name for the package and include a package command as the first statement in the Java source file. The java source file can contain the classes, interfaces, enumerations, and annotation types that you want to include in the package. For example, the following statement creates a package named **MyPackage.**

package MyPackage;

The package statement simply specifies to which package the classes defined belongs to.

**Note:** If you omit the package statement, the class names are put into the default package, which has no name. Though the default package is fine for short programs, it is inadequate for real applications.

## Including a Class in Java Package

To create a class inside a package, you should declare the package name as the first statement of your program. Then include the class as part of the package. But, remember that, a class can have only one package declaration. Here’s a simple program to understand the concept.

package MyPackage;

public class Compare {

    int num1, num2;

    Compare(int n, int m) {

        num1 = n;

        num2 = m;

    }

    public void getmax() {

        if (num1 > num2) {

            System.out.println("Maximum value of two numbers is " + num1);

        } else {

            System.out.println("Maximum value of two numbers is " + num2);

        }

    }

    public static void main(String args[]) {

        Compare current[] = new Compare[3];

        current[1] = new Compare(5, 10);

        current[2] = new Compare(123, 120);

        for (int i = 1; i < 3; i++) {

            current[i].getmax();

        }

    }

}

As you can see, I have declared a package named MyPackage and created a class Compare inside that package. Java uses file system directories to store packages. So, this program would be saved in a file as and will be stored in the directory named MyPackage. When the file gets compiled, Java will create a Compare.java **.class**file and store it in the same directory. Remember that name of the package must be same as the directory under which this file is saved.

You might be wondering how to use this Compare class from a class in another package?

## Creating a class inside a package while importing another package

Well, it’s quite simple. You just need to import it. Once it is imported, you can access it by its name. Here’s a sample program demonstrating the concept.

package university;

import MyPackage.Compare;

public class Demo {

    public static void main(String args[]) {

        int n = 10, m = 10;

        Compare current = new Compare(n, m);

        if (n != m) {

            current.getmax();

        } else {

            System.out.println("Both the values are same");

        }

    }

}

**Output:**

Both the values are same

I have first declared the package *university* , then imported the class *Compare* from the package MyPackage. So, the order when we are creating a class inside a package while importing another package is,

* Package Declaration
* Package Import

Well, if you do not want to use the import statement, there is another alternative to access a class file of the package from another package. You can just use fully qualified name while importing a [class](https://www.edureka.co/blog/java-tutorial/#obj).

package university;

public class Demo {

    public static void main(String args[]) {

        int n = 10, m = 11;

        // Using fully qualified name instead of import

        MyPackage.Compare current = new MyPackage.Compare(n, m);

        if (n != m) {

            current.getmax();

        } else {

            System.out.println("Both the values are same");

        }

    }

}

**Output:**

Maximum value of two numbers is 11

In the Demo class, instead of importing the package, I have used a fully qualified name such as MyPackage.Compare to create the object of it. Since we are talking about importing packages, you might as well check out the concept of static import in Java.

# Static Import in Java

Static import feature was introduced in [Java](https://www.edureka.co/blog/what-is-java/) from version 5. It facilitates the Java programmer to access any static member of a class directly without using the fully qualified name.

package MyPackage;

import static java.lang.Math.\*; //static import

import static java.lang.System.\*;// static import

public class StaticImportDemo {

    public static void main(String args[]) {

        double val = 64.0;

        double sqroot = sqrt(val); // Access sqrt() method directly

        out.println("Sq. root of " + val + " is " + sqroot);

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

    }

}

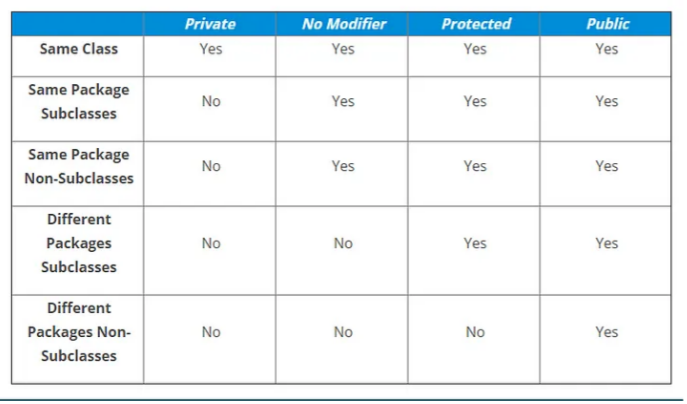
**Output:**Sq. root of 64.0 is 8.0

# Access Protection in Java Packages

You might be aware of various aspects of Java’s access control mechanism and its access specifiers. Packages in Java add another dimension to access control. Both classes and packages are a means of [data encapsulation](https://www.edureka.co/blog/encapsulation-in-java/). While packages act as containers for classes and other subordinate packages, classes act as containers for data and code. Because of this interplay between packages and classes, Java packages addresses four categories of visibility for class members:

* Sub-classes in the same package
* Non-subclasses in the same package
* Sub-classes in different packages
* Classes that are neither in the same package nor sub-classes

The table below gives a real picture of which type access is possible and which is not when using packages in Java:



We can simplify the data in the above table as follows:

1. Anything declared public can be accessed from anywhere
2. Anything declared private can be seen only within that class
3. If access specifier is not mentioned, an element is visible to subclasses as well as to other classes in the same package
4. Lastly, anything declared protected element can be seen outside your current package, but only to classes that subclass your class directly

This way, Java packages provide access control to the classes. Well, this wraps up the concept of packages in Java. Here are some points that you should keep in mind when using packages in [Java](https://www.edureka.co/blog/java-tutorial/).

# Points to Remember

* Every class is part of some package. If you omit the package statement, the class names are put into the default package
* A class can have only one package statement but it can have more than one import package statements
* The name of the package must be the same as the directory under which the file is saved
* When importing another package, package declaration must be the first statement, followed by package import