

Java Packages

[< Previous](#)[Next >](#)

Java Packages & API

A package in Java is used to group related classes. Think of it as **a folder in a file directory**. We use packages to avoid name conflicts, and to write a better maintainable code. Packages are divided into two categories:

- Built-in Packages (packages from the Java API)
- User-defined Packages (create your own packages)

Built-in Packages

The Java API is a library of prewritten classes, that are free to use, included in the Java Development Environment.

The library contains components for managing input, database programming, and much much more. The complete list can be found at Oracles website:

<https://docs.oracle.com/javase/8/docs/api/>.

The library is divided into **packages** and **classes**. Meaning you can either import a single class (along with its methods and attributes), or a whole package that contain all the classes that belong to the specified package.

To use a class or a package from the library, you need to use the **import** keyword:

Syntax

```
import package.name.Class; // Import a single class
import package.name.*; // Import the whole package
```

Import a Class

If you find a class you want to use, for example, the `Scanner` class, **which is used to get user input**, write the following code:

Example

```
import java.util.Scanner;
```

In the example above, `java.util` is a package, while `Scanner` is a class of the `java.util` package.

To use the `Scanner` class, create an object of the class and use any of the available methods found in the `Scanner` class documentation. In our example, we will use the `nextLine()` method, which is used to read a complete line:

Example

Using the `Scanner` class to get user input:

```
import java.util.Scanner;

class MyClass {
    public static void main(String[] args) {
        Scanner myObj = new Scanner(System.in);
        String userName;

        System.out.println("Enter username");
        userName = myObj.nextLine();

        System.out.println("Username is: " + userName);
    }
}
```

[Run example »](#)

Import a Package

There are many packages to choose from. In the previous example, we used the `Scanner` class from the `java.util` package. This package also contains date and time facilities, random-number generator and other utility classes.

To import a whole package, end the sentence with an asterisk sign (`*`). The following example will import ALL the classes in the `java.util` package:

Example

```
import java.util.*;
```

User-defined Packages

To create your own package, you need to understand that Java use a file system directory to store them. Just like folders on your computer:

Example

```
└─ root
  └─ mypack
    └─ MyPackageClass.java
```

To create a package, use the `package` keyword:

MyPackageClass.java

```
package mypack;

class MyPackageClass {
    public static void main(String[] args) {
        System.out.println("This is my package!");
    }
}
```

[Run example »](#)

Save the file as **MyPackageClass.java**, and compile it:

```
C:\Users\Your Name>javac MyPackageClass.java
```

Then compile the package:

```
C:\Users\Your Name>javac -d . MyPackageClass.java
```

This forces the compiler to create the "mypack" package.

The **-d** keyword specifies the destination for where to save the class file. You can use any directory name, like c:/user (windows), or, if you want to keep the package within the same directory, you can use the dot sign " . ", like in the example above.

Note: The package name should be written in lower case to avoid conflict with class names.

When we compiled the package in the example above, a new folder was created, called "mypack".

To run the **MyPackageClass.java** file, write the following:

```
C:\Users\Your Name>java mypack.MyPackageClass
```

The output will be:

```
This is my package!
```

[< Previous](#)[Next >](#)

Copyright 1999-2018 by Refsnes Data. All Rights Reserved.