# **java**

Launches a Java application.

## Synopsis

**java** [ *options* ] *class* [ *arguments* ]

**java** [ *options* ] **-jar** *file.jar* [ *arguments* ]

**javaw** [ *options* ] *class* [ *arguments* ]

**javaw** [ *options* ] **-jar** *file.jar* [ *arguments* ]

***options***

Command-line options. See [Options](https://docs.oracle.com/javase/7/docs/technotes/tools/windows/java.html" \l "CBBIJCHG).

***class***

The name of the class to be called.

***file.jar***

The name of the JAR file to be called. Used only with the -jar command.

***argument*s**

The arguments passed to the main function.

## Description

The java command starts a Java application. It does this by starting a Java runtime environment, loading a specified class, and calling that class's main method.

The method must be declared public and static, it must not return any value, and it must accept a String array as a parameter. The method declaration has the following form:

public static void main(String[] args)

By default, the first argument without an option is the name of the class to be called. A fully qualified class name should be used. If the -jar option is specified, then the first non-option argument is the name of a JAR file containing class and resource files for the application, with the startup class indicated by the Main-Class manifest header.

The Java runtime searches for the startup class, and other classes used, in three sets of locations: the bootstrap class path, the installed extensions, and the user class path.

Non-option arguments after the class name or JAR file name are passed to the main function.

The javaw command is identical to java, except that with javaw there is no associated console window. Use javaw when you do not want a command prompt window to appear. The javaw launcher will, however, display a dialog box with error information if a launch fails for some reason.

## Options

The launcher has a set of standard options that are supported in the current runtime environment.

In addition, the default Java HotSpot VMs provide a set of non-standard options that are subject to change in future releases. See [Nonstandard Options](https://docs.oracle.com/javase/7/docs/technotes/tools/windows/java.html" \l "BGBJAAEH).

### Standard Options

**-client**

Selects the Java HotSpot Client VM. A 64-bit capable JDK currently ignores this option and instead uses the Java Hotspot Server VM.

For default Java VM selection, see the Server-Class Machine Detection page at  
<http://docs.oracle.com/javase/7/docs/technotes/guides/vm/server-class.html>

**-server**

Selects the Java HotSpot Server VM. On a 64-bit capable JDK, only the Java Hotspot Server VM is supported so the -server option is implicit.

For default a Java VM selection, see the Server-Class Machine Detection page at  
<http://docs.oracle.com/javase/7/docs/technotes/guides/vm/server-class.html>

**-agentlib:*libname*[*=options*]**

Loads native agent library *libname*, for example:

-agentlib:hprof

-agentlib:jdwp=help

-agentlib:hprof=help

See JVMTI Agent Command-Line Options at  
[http://docs.oracle.com/javase/7/docs/platform/jvmti/jvmti.html#starting](http://docs.oracle.com/javase/7/docs/platform/jvmti/jvmti.html" \l "starting)

**-agentpath:*pathname*[*=options*]**

Loads a native agent library by full pathname.

**-classpath *classpath***

**-cp *classpath***

Specifies a list of directories, JAR files, and ZIP archives to search for class files. Separate class path entries with semicolons (;). Specifying -classpath or -cp overrides any setting of the CLASSPATH environment variable.

If -classpath and -cp are not used and *CLASSPATH* is not set, then the user class path consists of the current directory (.).

As a special convenience, a class path element that contains a base name of \* is considered equivalent to specifying a list of all the files in the directory with the extension .jar or .JAR. A Java program cannot tell the difference between the two invocations.

For example, if directory mydir contains a.jar and b.JAR, then the class path element mydir/\* is expanded to a A.jar:b.JAR, except that the order of jar files is unspecified. All jar files in the specified directory, even hidden ones, are included in the list. A class path entry consisting simply of \* expands to a list of all the jar files in the current directory. The *CLASSPATH* environment variable, where defined, will be similarly expanded. Any class path wildcard expansion occurs before the Java VM is started. No Java program will ever see wild cards that are not expanded except by querying the environment. For example, by calling System.getenv("CLASSPATH").

[https://docs.oracle.com/javase/7/docs/technotes/tools/windows/java.html#CBBIJCHG](https://docs.oracle.com/javase/7/docs/technotes/tools/windows/java.html" \l "CBBIJCHG)