| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/Runtime.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/lang/Runnable.html)   [**NEXT CLASS**](http://docs.google.com/java/lang/RuntimeException.html) | [**FRAMES**](http://docs.google.com/index.html?java/lang/Runtime.html)    [**NO FRAMES**](http://docs.google.com/Runtime.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | CONSTR | [METHOD](#3znysh7) | DETAIL: FIELD | CONSTR | [METHOD](#tyjcwt) |

## **java.lang**

Class Runtime

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 **java.lang.Runtime**

public class **Runtime**extends [Object](http://docs.google.com/java/lang/Object.html)

Every Java application has a single instance of class Runtime that allows the application to interface with the environment in which the application is running. The current runtime can be obtained from the getRuntime method.

An application cannot create its own instance of this class.

**Since:** JDK1.0 **See Also:**[getRuntime()](http://docs.google.com/java/lang/Runtime.html#getRuntime())

| **Method Summary** | |
| --- | --- |
| void | [**addShutdownHook**](http://docs.google.com/java/lang/Runtime.html#addShutdownHook(java.lang.Thread))([Thread](http://docs.google.com/java/lang/Thread.html) hook)            Registers a new virtual-machine shutdown hook. |
| int | [**availableProcessors**](http://docs.google.com/java/lang/Runtime.html#availableProcessors())()            Returns the number of processors available to the Java virtual machine. |
| [Process](http://docs.google.com/java/lang/Process.html) | [**exec**](http://docs.google.com/java/lang/Runtime.html#exec(java.lang.String))([String](http://docs.google.com/java/lang/String.html) command)            Executes the specified string command in a separate process. |
| [Process](http://docs.google.com/java/lang/Process.html) | [**exec**](http://docs.google.com/java/lang/Runtime.html#exec(java.lang.String%5B%5D))([String](http://docs.google.com/java/lang/String.html)[] cmdarray)            Executes the specified command and arguments in a separate process. |
| [Process](http://docs.google.com/java/lang/Process.html) | [**exec**](http://docs.google.com/java/lang/Runtime.html#exec(java.lang.String%5B%5D,%20java.lang.String%5B%5D))([String](http://docs.google.com/java/lang/String.html)[] cmdarray, [String](http://docs.google.com/java/lang/String.html)[] envp)            Executes the specified command and arguments in a separate process with the specified environment. |
| [Process](http://docs.google.com/java/lang/Process.html) | [**exec**](http://docs.google.com/java/lang/Runtime.html#exec(java.lang.String%5B%5D,%20java.lang.String%5B%5D,%20java.io.File))([String](http://docs.google.com/java/lang/String.html)[] cmdarray, [String](http://docs.google.com/java/lang/String.html)[] envp, [File](http://docs.google.com/java/io/File.html) dir)            Executes the specified command and arguments in a separate process with the specified environment and working directory. |
| [Process](http://docs.google.com/java/lang/Process.html) | [**exec**](http://docs.google.com/java/lang/Runtime.html#exec(java.lang.String,%20java.lang.String%5B%5D))([String](http://docs.google.com/java/lang/String.html) command, [String](http://docs.google.com/java/lang/String.html)[] envp)            Executes the specified string command in a separate process with the specified environment. |
| [Process](http://docs.google.com/java/lang/Process.html) | [**exec**](http://docs.google.com/java/lang/Runtime.html#exec(java.lang.String,%20java.lang.String%5B%5D,%20java.io.File))([String](http://docs.google.com/java/lang/String.html) command, [String](http://docs.google.com/java/lang/String.html)[] envp, [File](http://docs.google.com/java/io/File.html) dir)            Executes the specified string command in a separate process with the specified environment and working directory. |
| void | [**exit**](http://docs.google.com/java/lang/Runtime.html#exit(int))(int status)            Terminates the currently running Java virtual machine by initiating its shutdown sequence. |
| long | [**freeMemory**](http://docs.google.com/java/lang/Runtime.html#freeMemory())()            Returns the amount of free memory in the Java Virtual Machine. |
| void | [**gc**](http://docs.google.com/java/lang/Runtime.html#gc())()            Runs the garbage collector. |
| [InputStream](http://docs.google.com/java/io/InputStream.html) | [**getLocalizedInputStream**](http://docs.google.com/java/lang/Runtime.html#getLocalizedInputStream(java.io.InputStream))([InputStream](http://docs.google.com/java/io/InputStream.html) in)  **Deprecated.** *As of JDK 1.1, the preferred way to translate a byte stream in the local encoding into a character stream in Unicode is via the InputStreamReader and BufferedReader classes.* |
| [OutputStream](http://docs.google.com/java/io/OutputStream.html) | [**getLocalizedOutputStream**](http://docs.google.com/java/lang/Runtime.html#getLocalizedOutputStream(java.io.OutputStream))([OutputStream](http://docs.google.com/java/io/OutputStream.html) out)  **Deprecated.** *As of JDK 1.1, the preferred way to translate a Unicode character stream into a byte stream in the local encoding is via the OutputStreamWriter, BufferedWriter, and PrintWriter classes.* |
| static [Runtime](http://docs.google.com/java/lang/Runtime.html) | [**getRuntime**](http://docs.google.com/java/lang/Runtime.html#getRuntime())()            Returns the runtime object associated with the current Java application. |
| void | [**halt**](http://docs.google.com/java/lang/Runtime.html#halt(int))(int status)            Forcibly terminates the currently running Java virtual machine. |
| void | [**load**](http://docs.google.com/java/lang/Runtime.html#load(java.lang.String))([String](http://docs.google.com/java/lang/String.html) filename)            Loads the specified filename as a dynamic library. |
| void | [**loadLibrary**](http://docs.google.com/java/lang/Runtime.html#loadLibrary(java.lang.String))([String](http://docs.google.com/java/lang/String.html) libname)            Loads the dynamic library with the specified library name. |
| long | [**maxMemory**](http://docs.google.com/java/lang/Runtime.html#maxMemory())()            Returns the maximum amount of memory that the Java virtual machine will attempt to use. |
| boolean | [**removeShutdownHook**](http://docs.google.com/java/lang/Runtime.html#removeShutdownHook(java.lang.Thread))([Thread](http://docs.google.com/java/lang/Thread.html) hook)            De-registers a previously-registered virtual-machine shutdown hook. |
| void | [**runFinalization**](http://docs.google.com/java/lang/Runtime.html#runFinalization())()            Runs the finalization methods of any objects pending finalization. |
| static void | [**runFinalizersOnExit**](http://docs.google.com/java/lang/Runtime.html#runFinalizersOnExit(boolean))(boolean value)  **Deprecated.** *This method is inherently unsafe. It may result in finalizers being called on live objects while other threads are concurrently manipulating those objects, resulting in erratic behavior or deadlock.* |
| long | [**totalMemory**](http://docs.google.com/java/lang/Runtime.html#totalMemory())()            Returns the total amount of memory in the Java virtual machine. |
| void | [**traceInstructions**](http://docs.google.com/java/lang/Runtime.html#traceInstructions(boolean))(boolean on)            Enables/Disables tracing of instructions. |
| void | [**traceMethodCalls**](http://docs.google.com/java/lang/Runtime.html#traceMethodCalls(boolean))(boolean on)            Enables/Disables tracing of method calls. |

| **Methods inherited from class java.lang.**[**Object**](http://docs.google.com/java/lang/Object.html) |
| --- |
| [clone](http://docs.google.com/java/lang/Object.html#clone()), [equals](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)), [finalize](http://docs.google.com/java/lang/Object.html#finalize()), [getClass](http://docs.google.com/java/lang/Object.html#getClass()), [hashCode](http://docs.google.com/java/lang/Object.html#hashCode()), [notify](http://docs.google.com/java/lang/Object.html#notify()), [notifyAll](http://docs.google.com/java/lang/Object.html#notifyAll()), [toString](http://docs.google.com/java/lang/Object.html#toString()), [wait](http://docs.google.com/java/lang/Object.html#wait()), [wait](http://docs.google.com/java/lang/Object.html#wait(long)), [wait](http://docs.google.com/java/lang/Object.html#wait(long,%20int)) |

| **Method Detail** |
| --- |

### getRuntime

public static [Runtime](http://docs.google.com/java/lang/Runtime.html) **getRuntime**()

Returns the runtime object associated with the current Java application. Most of the methods of class Runtime are instance methods and must be invoked with respect to the current runtime object.

**Returns:**the Runtime object associated with the current Java application.

### exit

public void **exit**(int status)

Terminates the currently running Java virtual machine by initiating its shutdown sequence. This method never returns normally. The argument serves as a status code; by convention, a nonzero status code indicates abnormal termination.

The virtual machine's shutdown sequence consists of two phases. In the first phase all registered [shutdown hooks](http://docs.google.com/java/lang/Runtime.html#addShutdownHook(java.lang.Thread)), if any, are started in some unspecified order and allowed to run concurrently until they finish. In the second phase all uninvoked finalizers are run if [finalization-on-exit](http://docs.google.com/java/lang/Runtime.html#runFinalizersOnExit(boolean)) has been enabled. Once this is done the virtual machine [halts](http://docs.google.com/java/lang/Runtime.html#halt(int)).

If this method is invoked after the virtual machine has begun its shutdown sequence then if shutdown hooks are being run this method will block indefinitely. If shutdown hooks have already been run and on-exit finalization has been enabled then this method halts the virtual machine with the given status code if the status is nonzero; otherwise, it blocks indefinitely.

The [System.exit](http://docs.google.com/java/lang/System.html#exit(int)) method is the conventional and convenient means of invoking this method.

**Parameters:**status - Termination status. By convention, a nonzero status code indicates abnormal termination. **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - If a security manager is present and its [checkExit](http://docs.google.com/java/lang/SecurityManager.html#checkExit(int)) method does not permit exiting with the specified status**See Also:**[SecurityException](http://docs.google.com/java/lang/SecurityException.html), [SecurityManager.checkExit(int)](http://docs.google.com/java/lang/SecurityManager.html#checkExit(int)), [addShutdownHook(java.lang.Thread)](http://docs.google.com/java/lang/Runtime.html#addShutdownHook(java.lang.Thread)), [removeShutdownHook(java.lang.Thread)](http://docs.google.com/java/lang/Runtime.html#removeShutdownHook(java.lang.Thread)), [runFinalizersOnExit(boolean)](http://docs.google.com/java/lang/Runtime.html#runFinalizersOnExit(boolean)), [halt(int)](http://docs.google.com/java/lang/Runtime.html#halt(int))

### addShutdownHook

public void **addShutdownHook**([Thread](http://docs.google.com/java/lang/Thread.html) hook)

Registers a new virtual-machine shutdown hook.

The Java virtual machine *shuts down* in response to two kinds of events:

* The program *exits* normally, when the last non-daemon thread exits or when the [exit](http://docs.google.com/java/lang/Runtime.html#exit(int)) (equivalently, [System.exit](http://docs.google.com/java/lang/System.html#exit(int))) method is invoked, or
* The virtual machine is *terminated* in response to a user interrupt, such as typing ^C, or a system-wide event, such as user logoff or system shutdown.

A *shutdown hook* is simply an initialized but unstarted thread. When the virtual machine begins its shutdown sequence it will start all registered shutdown hooks in some unspecified order and let them run concurrently. When all the hooks have finished it will then run all uninvoked finalizers if finalization-on-exit has been enabled. Finally, the virtual machine will halt. Note that daemon threads will continue to run during the shutdown sequence, as will non-daemon threads if shutdown was initiated by invoking the [exit](http://docs.google.com/java/lang/Runtime.html#exit(int)) method.

Once the shutdown sequence has begun it can be stopped only by invoking the [halt](http://docs.google.com/java/lang/Runtime.html#halt(int)) method, which forcibly terminates the virtual machine.

Once the shutdown sequence has begun it is impossible to register a new shutdown hook or de-register a previously-registered hook. Attempting either of these operations will cause an [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) to be thrown.

Shutdown hooks run at a delicate time in the life cycle of a virtual machine and should therefore be coded defensively. They should, in particular, be written to be thread-safe and to avoid deadlocks insofar as possible. They should also not rely blindly upon services that may have registered their own shutdown hooks and therefore may themselves in the process of shutting down. Attempts to use other thread-based services such as the AWT event-dispatch thread, for example, may lead to deadlocks.

Shutdown hooks should also finish their work quickly. When a program invokes [exit](http://docs.google.com/java/lang/Runtime.html#exit(int)) the expectation is that the virtual machine will promptly shut down and exit. When the virtual machine is terminated due to user logoff or system shutdown the underlying operating system may only allow a fixed amount of time in which to shut down and exit. It is therefore inadvisable to attempt any user interaction or to perform a long-running computation in a shutdown hook.

Uncaught exceptions are handled in shutdown hooks just as in any other thread, by invoking the [uncaughtException](http://docs.google.com/java/lang/ThreadGroup.html#uncaughtException(java.lang.Thread,%20java.lang.Throwable)) method of the thread's [ThreadGroup](http://docs.google.com/java/lang/ThreadGroup.html) object. The default implementation of this method prints the exception's stack trace to [System.err](http://docs.google.com/java/lang/System.html#err) and terminates the thread; it does not cause the virtual machine to exit or halt.

In rare circumstances the virtual machine may *abort*, that is, stop running without shutting down cleanly. This occurs when the virtual machine is terminated externally, for example with the SIGKILL signal on Unix or the TerminateProcess call on Microsoft Windows. The virtual machine may also abort if a native method goes awry by, for example, corrupting internal data structures or attempting to access nonexistent memory. If the virtual machine aborts then no guarantee can be made about whether or not any shutdown hooks will be run.

**Parameters:**hook - An initialized but unstarted [Thread](http://docs.google.com/java/lang/Thread.html) object **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - If the specified hook has already been registered, or if it can be determined that the hook is already running or has already been run [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - If the virtual machine is already in the process of shutting down [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - If a security manager is present and it denies [RuntimePermission](http://docs.google.com/java/lang/RuntimePermission.html)("shutdownHooks")**Since:** 1.3 **See Also:**[removeShutdownHook(java.lang.Thread)](http://docs.google.com/java/lang/Runtime.html#removeShutdownHook(java.lang.Thread)), [halt(int)](http://docs.google.com/java/lang/Runtime.html#halt(int)), [exit(int)](http://docs.google.com/java/lang/Runtime.html#exit(int))

### removeShutdownHook

public boolean **removeShutdownHook**([Thread](http://docs.google.com/java/lang/Thread.html) hook)

De-registers a previously-registered virtual-machine shutdown hook.

**Parameters:**hook - the hook to remove **Returns:**true if the specified hook had previously been registered and was successfully de-registered, false otherwise. **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - If the virtual machine is already in the process of shutting down [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - If a security manager is present and it denies [RuntimePermission](http://docs.google.com/java/lang/RuntimePermission.html)("shutdownHooks")**Since:** 1.3 **See Also:**[addShutdownHook(java.lang.Thread)](http://docs.google.com/java/lang/Runtime.html#addShutdownHook(java.lang.Thread)), [exit(int)](http://docs.google.com/java/lang/Runtime.html#exit(int))

### halt

public void **halt**(int status)

Forcibly terminates the currently running Java virtual machine. This method never returns normally.

This method should be used with extreme caution. Unlike the [exit](http://docs.google.com/java/lang/Runtime.html#exit(int)) method, this method does not cause shutdown hooks to be started and does not run uninvoked finalizers if finalization-on-exit has been enabled. If the shutdown sequence has already been initiated then this method does not wait for any running shutdown hooks or finalizers to finish their work.

**Parameters:**status - Termination status. By convention, a nonzero status code indicates abnormal termination. If the [exit](http://docs.google.com/java/lang/Runtime.html#exit(int)) (equivalently, [System.exit](http://docs.google.com/java/lang/System.html#exit(int))) method has already been invoked then this status code will override the status code passed to that method. **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - If a security manager is present and its [checkExit](http://docs.google.com/java/lang/SecurityManager.html#checkExit(int)) method does not permit an exit with the specified status**Since:** 1.3 **See Also:**[exit(int)](http://docs.google.com/java/lang/Runtime.html#exit(int)), [addShutdownHook(java.lang.Thread)](http://docs.google.com/java/lang/Runtime.html#addShutdownHook(java.lang.Thread)), [removeShutdownHook(java.lang.Thread)](http://docs.google.com/java/lang/Runtime.html#removeShutdownHook(java.lang.Thread))

### runFinalizersOnExit

[@Deprecated](http://docs.google.com/java/lang/Deprecated.html)  
public static void **runFinalizersOnExit**(boolean value)

**Deprecated.** *This method is inherently unsafe. It may result in finalizers being called on live objects while other threads are concurrently manipulating those objects, resulting in erratic behavior or deadlock.*

Enable or disable finalization on exit; doing so specifies that the finalizers of all objects that have finalizers that have not yet been automatically invoked are to be run before the Java runtime exits. By default, finalization on exit is disabled.

If there is a security manager, its checkExit method is first called with 0 as its argument to ensure the exit is allowed. This could result in a SecurityException.

**Parameters:**value - true to enable finalization on exit, false to disable **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if a security manager exists and its checkExit method doesn't allow the exit.**Since:** JDK1.1 **See Also:**[exit(int)](http://docs.google.com/java/lang/Runtime.html#exit(int)), [gc()](http://docs.google.com/java/lang/Runtime.html#gc()), [SecurityManager.checkExit(int)](http://docs.google.com/java/lang/SecurityManager.html#checkExit(int))

### exec

public [Process](http://docs.google.com/java/lang/Process.html) **exec**([String](http://docs.google.com/java/lang/String.html) command)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Executes the specified string command in a separate process.

This is a convenience method. An invocation of the form exec(command) behaves in exactly the same way as the invocation [exec](http://docs.google.com/java/lang/Runtime.html#exec(java.lang.String,%20java.lang.String%5B%5D,%20java.io.File))(command, null, null).

**Parameters:**command - a specified system command. **Returns:**A new [Process](http://docs.google.com/java/lang/Process.html) object for managing the subprocess **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - If a security manager exists and its [checkExec](http://docs.google.com/java/lang/SecurityManager.html#checkExec(java.lang.String)) method doesn't allow creation of the subprocess [IOException](http://docs.google.com/java/io/IOException.html) - If an I/O error occurs [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - If command is null [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - If command is empty**See Also:**[exec(String[], String[], File)](http://docs.google.com/java/lang/Runtime.html#exec(java.lang.String%5B%5D,%20java.lang.String%5B%5D,%20java.io.File)), [ProcessBuilder](http://docs.google.com/java/lang/ProcessBuilder.html)

### exec

public [Process](http://docs.google.com/java/lang/Process.html) **exec**([String](http://docs.google.com/java/lang/String.html) command,  
 [String](http://docs.google.com/java/lang/String.html)[] envp)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Executes the specified string command in a separate process with the specified environment.

This is a convenience method. An invocation of the form exec(command, envp) behaves in exactly the same way as the invocation [exec](http://docs.google.com/java/lang/Runtime.html#exec(java.lang.String,%20java.lang.String%5B%5D,%20java.io.File))(command, envp, null).

**Parameters:**command - a specified system command.envp - array of strings, each element of which has environment variable settings in the format *name*=*value*, or null if the subprocess should inherit the environment of the current process. **Returns:**A new [Process](http://docs.google.com/java/lang/Process.html) object for managing the subprocess **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - If a security manager exists and its [checkExec](http://docs.google.com/java/lang/SecurityManager.html#checkExec(java.lang.String)) method doesn't allow creation of the subprocess [IOException](http://docs.google.com/java/io/IOException.html) - If an I/O error occurs [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - If command is null, or one of the elements of envp is null [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - If command is empty**See Also:**[exec(String[], String[], File)](http://docs.google.com/java/lang/Runtime.html#exec(java.lang.String%5B%5D,%20java.lang.String%5B%5D,%20java.io.File)), [ProcessBuilder](http://docs.google.com/java/lang/ProcessBuilder.html)

### exec

public [Process](http://docs.google.com/java/lang/Process.html) **exec**([String](http://docs.google.com/java/lang/String.html) command,  
 [String](http://docs.google.com/java/lang/String.html)[] envp,  
 [File](http://docs.google.com/java/io/File.html) dir)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Executes the specified string command in a separate process with the specified environment and working directory.

This is a convenience method. An invocation of the form exec(command, envp, dir) behaves in exactly the same way as the invocation [exec](http://docs.google.com/java/lang/Runtime.html#exec(java.lang.String%5B%5D,%20java.lang.String%5B%5D,%20java.io.File))(cmdarray, envp, dir), where cmdarray is an array of all the tokens in command.

More precisely, the command string is broken into tokens using a [StringTokenizer](http://docs.google.com/java/util/StringTokenizer.html) created by the call new [StringTokenizer](http://docs.google.com/java/util/StringTokenizer.html)(command) with no further modification of the character categories. The tokens produced by the tokenizer are then placed in the new string array cmdarray, in the same order.

**Parameters:**command - a specified system command.envp - array of strings, each element of which has environment variable settings in the format *name*=*value*, or null if the subprocess should inherit the environment of the current process.dir - the working directory of the subprocess, or null if the subprocess should inherit the working directory of the current process. **Returns:**A new [Process](http://docs.google.com/java/lang/Process.html) object for managing the subprocess **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - If a security manager exists and its [checkExec](http://docs.google.com/java/lang/SecurityManager.html#checkExec(java.lang.String)) method doesn't allow creation of the subprocess [IOException](http://docs.google.com/java/io/IOException.html) - If an I/O error occurs [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - If command is null, or one of the elements of envp is null [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - If command is empty**Since:** 1.3 **See Also:**[ProcessBuilder](http://docs.google.com/java/lang/ProcessBuilder.html)

### exec

public [Process](http://docs.google.com/java/lang/Process.html) **exec**([String](http://docs.google.com/java/lang/String.html)[] cmdarray)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Executes the specified command and arguments in a separate process.

This is a convenience method. An invocation of the form exec(cmdarray) behaves in exactly the same way as the invocation [exec](http://docs.google.com/java/lang/Runtime.html#exec(java.lang.String%5B%5D,%20java.lang.String%5B%5D,%20java.io.File))(cmdarray, null, null).

**Parameters:**cmdarray - array containing the command to call and its arguments. **Returns:**A new [Process](http://docs.google.com/java/lang/Process.html) object for managing the subprocess **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - If a security manager exists and its [checkExec](http://docs.google.com/java/lang/SecurityManager.html#checkExec(java.lang.String)) method doesn't allow creation of the subprocess [IOException](http://docs.google.com/java/io/IOException.html) - If an I/O error occurs [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - If cmdarray is null, or one of the elements of cmdarray is null [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - If cmdarray is an empty array (has length 0)**See Also:**[ProcessBuilder](http://docs.google.com/java/lang/ProcessBuilder.html)

### exec

public [Process](http://docs.google.com/java/lang/Process.html) **exec**([String](http://docs.google.com/java/lang/String.html)[] cmdarray,  
 [String](http://docs.google.com/java/lang/String.html)[] envp)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Executes the specified command and arguments in a separate process with the specified environment.

This is a convenience method. An invocation of the form exec(cmdarray, envp) behaves in exactly the same way as the invocation [exec](http://docs.google.com/java/lang/Runtime.html#exec(java.lang.String%5B%5D,%20java.lang.String%5B%5D,%20java.io.File))(cmdarray, envp, null).

**Parameters:**cmdarray - array containing the command to call and its arguments.envp - array of strings, each element of which has environment variable settings in the format *name*=*value*, or null if the subprocess should inherit the environment of the current process. **Returns:**A new [Process](http://docs.google.com/java/lang/Process.html) object for managing the subprocess **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - If a security manager exists and its [checkExec](http://docs.google.com/java/lang/SecurityManager.html#checkExec(java.lang.String)) method doesn't allow creation of the subprocess [IOException](http://docs.google.com/java/io/IOException.html) - If an I/O error occurs [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - If cmdarray is null, or one of the elements of cmdarray is null, or one of the elements of envp is null [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - If cmdarray is an empty array (has length 0)**See Also:**[ProcessBuilder](http://docs.google.com/java/lang/ProcessBuilder.html)

### exec

public [Process](http://docs.google.com/java/lang/Process.html) **exec**([String](http://docs.google.com/java/lang/String.html)[] cmdarray,  
 [String](http://docs.google.com/java/lang/String.html)[] envp,  
 [File](http://docs.google.com/java/io/File.html) dir)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Executes the specified command and arguments in a separate process with the specified environment and working directory.

Given an array of strings cmdarray, representing the tokens of a command line, and an array of strings envp, representing "environment" variable settings, this method creates a new process in which to execute the specified command.

This method checks that cmdarray is a valid operating system command. Which commands are valid is system-dependent, but at the very least the command must be a non-empty list of non-null strings.

If envp is null, the subprocess inherits the environment settings of the current process.

[ProcessBuilder.start()](http://docs.google.com/java/lang/ProcessBuilder.html#start()) is now the preferred way to start a process with a modified environment.

The working directory of the new subprocess is specified by dir. If dir is null, the subprocess inherits the current working directory of the current process.

If a security manager exists, its [checkExec](http://docs.google.com/java/lang/SecurityManager.html#checkExec(java.lang.String)) method is invoked with the first component of the array cmdarray as its argument. This may result in a [SecurityException](http://docs.google.com/java/lang/SecurityException.html) being thrown.

Starting an operating system process is highly system-dependent. Among the many things that can go wrong are:

* The operating system program file was not found.
* Access to the program file was denied.
* The working directory does not exist.

In such cases an exception will be thrown. The exact nature of the exception is system-dependent, but it will always be a subclass of [IOException](http://docs.google.com/java/io/IOException.html).

**Parameters:**cmdarray - array containing the command to call and its arguments.envp - array of strings, each element of which has environment variable settings in the format *name*=*value*, or null if the subprocess should inherit the environment of the current process.dir - the working directory of the subprocess, or null if the subprocess should inherit the working directory of the current process. **Returns:**A new [Process](http://docs.google.com/java/lang/Process.html) object for managing the subprocess **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - If a security manager exists and its [checkExec](http://docs.google.com/java/lang/SecurityManager.html#checkExec(java.lang.String)) method doesn't allow creation of the subprocess [IOException](http://docs.google.com/java/io/IOException.html) - If an I/O error occurs [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - If cmdarray is null, or one of the elements of cmdarray is null, or one of the elements of envp is null [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - If cmdarray is an empty array (has length 0)**Since:** 1.3 **See Also:**[ProcessBuilder](http://docs.google.com/java/lang/ProcessBuilder.html)

### availableProcessors

public int **availableProcessors**()

Returns the number of processors available to the Java virtual machine.

This value may change during a particular invocation of the virtual machine. Applications that are sensitive to the number of available processors should therefore occasionally poll this property and adjust their resource usage appropriately.

**Returns:**the maximum number of processors available to the virtual machine; never smaller than one**Since:** 1.4

### freeMemory

public long **freeMemory**()

Returns the amount of free memory in the Java Virtual Machine. Calling the gc method may result in increasing the value returned by freeMemory.

**Returns:**an approximation to the total amount of memory currently available for future allocated objects, measured in bytes.

### totalMemory

public long **totalMemory**()

Returns the total amount of memory in the Java virtual machine. The value returned by this method may vary over time, depending on the host environment.

Note that the amount of memory required to hold an object of any given type may be implementation-dependent.

**Returns:**the total amount of memory currently available for current and future objects, measured in bytes.

### maxMemory

public long **maxMemory**()

Returns the maximum amount of memory that the Java virtual machine will attempt to use. If there is no inherent limit then the value [Long.MAX\_VALUE](http://docs.google.com/java/lang/Long.html#MAX_VALUE) will be returned.

**Returns:**the maximum amount of memory that the virtual machine will attempt to use, measured in bytes**Since:** 1.4

### gc

public void **gc**()

Runs the garbage collector. Calling this method suggests that the Java virtual machine expend effort toward recycling unused objects in order to make the memory they currently occupy available for quick reuse. When control returns from the method call, the virtual machine has made its best effort to recycle all discarded objects.

The name gc stands for "garbage collector". The virtual machine performs this recycling process automatically as needed, in a separate thread, even if the gc method is not invoked explicitly.

The method [System.gc()](http://docs.google.com/java/lang/System.html#gc()) is the conventional and convenient means of invoking this method.

### runFinalization

public void **runFinalization**()

Runs the finalization methods of any objects pending finalization. Calling this method suggests that the Java virtual machine expend effort toward running the finalize methods of objects that have been found to be discarded but whose finalize methods have not yet been run. When control returns from the method call, the virtual machine has made a best effort to complete all outstanding finalizations.

The virtual machine performs the finalization process automatically as needed, in a separate thread, if the runFinalization method is not invoked explicitly.

The method [System.runFinalization()](http://docs.google.com/java/lang/System.html#runFinalization()) is the conventional and convenient means of invoking this method.

**See Also:**[Object.finalize()](http://docs.google.com/java/lang/Object.html#finalize())

### traceInstructions

public void **traceInstructions**(boolean on)

Enables/Disables tracing of instructions. If the boolean argument is true, this method suggests that the Java virtual machine emit debugging information for each instruction in the virtual machine as it is executed. The format of this information, and the file or other output stream to which it is emitted, depends on the host environment. The virtual machine may ignore this request if it does not support this feature. The destination of the trace output is system dependent.

If the boolean argument is false, this method causes the virtual machine to stop performing the detailed instruction trace it is performing.

**Parameters:**on - true to enable instruction tracing; false to disable this feature.

### traceMethodCalls

public void **traceMethodCalls**(boolean on)

Enables/Disables tracing of method calls. If the boolean argument is true, this method suggests that the Java virtual machine emit debugging information for each method in the virtual machine as it is called. The format of this information, and the file or other output stream to which it is emitted, depends on the host environment. The virtual machine may ignore this request if it does not support this feature.

Calling this method with argument false suggests that the virtual machine cease emitting per-call debugging information.

**Parameters:**on - true to enable instruction tracing; false to disable this feature.

### load

public void **load**([String](http://docs.google.com/java/lang/String.html) filename)

Loads the specified filename as a dynamic library. The filename argument must be a complete path name, (for example Runtime.getRuntime().load("/home/avh/lib/libX11.so");).

First, if there is a security manager, its checkLink method is called with the filename as its argument. This may result in a security exception.

This is similar to the method [loadLibrary(String)](http://docs.google.com/java/lang/Runtime.html#loadLibrary(java.lang.String)), but it accepts a general file name as an argument rather than just a library name, allowing any file of native code to be loaded.

The method [System.load(String)](http://docs.google.com/java/lang/System.html#load(java.lang.String)) is the conventional and convenient means of invoking this method.

**Parameters:**filename - the file to load. **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if a security manager exists and its checkLink method doesn't allow loading of the specified dynamic library [UnsatisfiedLinkError](http://docs.google.com/java/lang/UnsatisfiedLinkError.html) - if the file does not exist. [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if filename is null**See Also:**[getRuntime()](http://docs.google.com/java/lang/Runtime.html#getRuntime()), [SecurityException](http://docs.google.com/java/lang/SecurityException.html), [SecurityManager.checkLink(java.lang.String)](http://docs.google.com/java/lang/SecurityManager.html#checkLink(java.lang.String))

### loadLibrary

public void **loadLibrary**([String](http://docs.google.com/java/lang/String.html) libname)

Loads the dynamic library with the specified library name. A file containing native code is loaded from the local file system from a place where library files are conventionally obtained. The details of this process are implementation-dependent. The mapping from a library name to a specific filename is done in a system-specific manner.

First, if there is a security manager, its checkLink method is called with the libname as its argument. This may result in a security exception.

The method [System.loadLibrary(String)](http://docs.google.com/java/lang/System.html#loadLibrary(java.lang.String)) is the conventional and convenient means of invoking this method. If native methods are to be used in the implementation of a class, a standard strategy is to put the native code in a library file (call it LibFile) and then to put a static initializer:

static { System.loadLibrary("LibFile"); }

within the class declaration. When the class is loaded and initialized, the necessary native code implementation for the native methods will then be loaded as well.

If this method is called more than once with the same library name, the second and subsequent calls are ignored.

**Parameters:**libname - the name of the library. **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if a security manager exists and its checkLink method doesn't allow loading of the specified dynamic library [UnsatisfiedLinkError](http://docs.google.com/java/lang/UnsatisfiedLinkError.html) - if the library does not exist. [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if libname is null**See Also:**[SecurityException](http://docs.google.com/java/lang/SecurityException.html), [SecurityManager.checkLink(java.lang.String)](http://docs.google.com/java/lang/SecurityManager.html#checkLink(java.lang.String))

### getLocalizedInputStream

[@Deprecated](http://docs.google.com/java/lang/Deprecated.html)  
public [InputStream](http://docs.google.com/java/io/InputStream.html) **getLocalizedInputStream**([InputStream](http://docs.google.com/java/io/InputStream.html) in)

**Deprecated.** *As of JDK 1.1, the preferred way to translate a byte stream in the local encoding into a character stream in Unicode is via the InputStreamReader and BufferedReader classes.*

Creates a localized version of an input stream. This method takes an InputStream and returns an InputStream equivalent to the argument in all respects except that it is localized: as characters in the local character set are read from the stream, they are automatically converted from the local character set to Unicode.

If the argument is already a localized stream, it may be returned as the result.

**Parameters:**in - InputStream to localize **Returns:**a localized input stream**See Also:**[InputStream](http://docs.google.com/java/io/InputStream.html), [BufferedReader.BufferedReader(java.io.Reader)](http://docs.google.com/java/io/BufferedReader.html#BufferedReader(java.io.Reader)), [InputStreamReader.InputStreamReader(java.io.InputStream)](http://docs.google.com/java/io/InputStreamReader.html#InputStreamReader(java.io.InputStream))

### getLocalizedOutputStream

[@Deprecated](http://docs.google.com/java/lang/Deprecated.html)  
public [OutputStream](http://docs.google.com/java/io/OutputStream.html) **getLocalizedOutputStream**([OutputStream](http://docs.google.com/java/io/OutputStream.html) out)

**Deprecated.** *As of JDK 1.1, the preferred way to translate a Unicode character stream into a byte stream in the local encoding is via the OutputStreamWriter, BufferedWriter, and PrintWriter classes.*

Creates a localized version of an output stream. This method takes an OutputStream and returns an OutputStream equivalent to the argument in all respects except that it is localized: as Unicode characters are written to the stream, they are automatically converted to the local character set.

If the argument is already a localized stream, it may be returned as the result.

**Parameters:**out - OutputStream to localize **Returns:**a localized output stream**See Also:**[OutputStream](http://docs.google.com/java/io/OutputStream.html), [BufferedWriter.BufferedWriter(java.io.Writer)](http://docs.google.com/java/io/BufferedWriter.html#BufferedWriter(java.io.Writer)), [OutputStreamWriter.OutputStreamWriter(java.io.OutputStream)](http://docs.google.com/java/io/OutputStreamWriter.html#OutputStreamWriter(java.io.OutputStream)), [PrintWriter.PrintWriter(java.io.OutputStream)](http://docs.google.com/java/io/PrintWriter.html#PrintWriter(java.io.OutputStream))

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/Runtime.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/lang/Runnable.html)   [**NEXT CLASS**](http://docs.google.com/java/lang/RuntimeException.html) | [**FRAMES**](http://docs.google.com/index.html?java/lang/Runtime.html)    [**NO FRAMES**](http://docs.google.com/Runtime.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | CONSTR | [METHOD](#3znysh7) | DETAIL: FIELD | CONSTR | [METHOD](#tyjcwt) |

[Submit a bug or feature](http://bugs.sun.com/services/bugreport/index.jsp)

For further API reference and developer documentation, see [Java SE Developer Documentation](http://docs.google.com/webnotes/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Copyright 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to [license terms](http://docs.google.com/legal/license.html). Also see the [documentation redistribution policy](http://java.sun.com/docs/redist.html).