Class java.lang.Runtime

```
java.lang.Object
   |
   +----java.lang.Runtime
```

---

public class **Runtime**
extends Object

Method Index

o **exec**(String)
o **exec**(String[])
o **exit**(int)
    Exits the virtual machine with an exit code.
o **freeMemory**()
    Returns the number of free bytes in system memory.
o **gc**()
    Runs the garbage collector.
o **getLocalizedInputStream**(InputStream)
    Localize an input stream.
o **getLocalizedOutputStream**(OutputStream)
    Localize an output stream.
o **getRuntime**()
    Return the runtime.
o **load**(String)
    Loads a dynamic library, given a complete path name.
o **loadLibrary**(String)
    Loads a dynamic library with the specified library name.
o **runFinalization**()
    Runs the finalization methods of any objects pending finalization.
o **totalMemory**()
    Returns the total number of bytes in system memory.
o **traceInstructions**(boolean)
    Enables/Disables tracing of instructions.
o **traceMethodCalls**(boolean)
    Enables/Disables tracing of method calls.

Methods

o **getRuntime**

```
public static Runtime getRuntime()
```

Return the runtime.

o **exit**

```
public void exit(int status)
```

Exits the virtual machine with an exit code. This method does not return, use with caution.
Parameters:
status – exit status, 0 if successful, other values indicate various error types.

o **exec**

```
public Process exec(String command) throws IOException
```

o **exec**

```
public Process exec(String cmdarray[]) throws IOException
```

o **freeMemory**

```
public long freeMemory()
```

Returns the number of free bytes in system memory. This number is not always accurate because it is just an estimation of the available memory. More memory may be freed by calling System.gc() .

o **totalMemory**

```
public long totalMemory()
```

Returns the total number of bytes in system memory.

o **gc**

```
public void gc()
```

Runs the garbage collector.

o **runFinalization**

```
public void runFinalization()
```

Runs the finalization methods of any objects pending finalization. Usually you will not need to

call this method since finalization methods will be called asynchronously by the finalization thread. However, under some circumstances (like running out of a finalized resource) it can be useful to run finalization methods synchronously.

o **traceInstructions**

```
public void traceInstructions(boolean on)
```

Enables/Disables tracing of instructions.
Parameters:
on – start tracing if true

o **traceMethodCalls**

```
public void traceMethodCalls(boolean on)
```

Enables/Disables tracing of method calls.
Parameters:
on – start tracing if true

o **load**

```
public synchronized void load(String filename)
```

Loads a dynamic library, given a complete path name. If you use this from java_g it will automagically insert "_g" before the ".so". Example:
Runtime.getRuntime().load("/home/avh/lib/libX11.so");
Parameters:
filename – the file to load
Throws: UnsatisfiedLinkError
If the file does not exist.
See Also:
getRuntime

o **loadLibrary**

```
public synchronized void loadLibrary(String libname)
```

Loads a dynamic library with the specified library name. The call to LoadLibrary() should be made in the static initializer of the first class that is loaded. Linking in the same library more than once is ignored.
Parameters:
libname – the name of the library
Throws: UnsatisfiedLinkError
If the library does not exist.

o **getLocalizedInputStream**

```
public InputStream getLocalizedInputStream(InputStream in)
```

Localize an input stream. A localized input stream will automatically translate the input from the local format to UNICODE.

o **getLocalizedOutputStream**

```
public OutputStream getLocalizedOutputStream(OutputStream out)
```

Localize an output stream. A localized output stream will automatically translate the output from UNICODE to the local format.