---

# Class java.lang.System

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

---

public final class **System**
extends Object

This Class provides a system–independent interface to system functionality. One of the more useful things provided by this Class are the standard input and output streams. The standard input streams are used for reading character data. The standard output streams are used for printing. For example:

```
System.out.println("Hello World!");
```

This Class cannot be instantiated or subclassed because all of the methods and variables are static.

**Version:**
1.53, 09/19/95
**Author:**
Arthur van Hoff

---

# Variable Index

o **err**
Standard error stream.
o **in**
Standard input stream.
o **out**
Standard output stream.

# Method Index

o **arraycopy**(Object, int, Object, int, int)
Copies an array from the source array, beginning at the specified position, to the specified position of the destination array.
o **currentTimeMillis**()
Returns the current time in milliseconds GMT since the epoch (00:00:00 UTC, January 1,

1970).

- **exit**(int)
  Exits the virtual machine with an exit code.
- **gc**()
  Runs the garbage collector.
- **getProperties**()
  Gets the System properties.
- **getProperty**(String)
  Gets the System property indicated by the specified key.
- **getProperty**(String, String)
  Gets the System property indicated by the specified key and def.
- **getSecurityManager**()
  Gets the system security interface.
- **getenv**(String)
  Obsolete.
- **load**(String)
  Loads a dynamic library, given a complete path name.
- **loadLibrary**(String)
  Loads a dynamic library with the specified library name.
- **runFinalization**()
  Runs the finalization methods of any objects pending finalization.
- **setProperties**(Properties)
  Sets the System properties to the specified properties.
- **setSecurityManager**(SecurityManager)
  Sets the System security.

# Variables

- **in**

```
public static InputStream in
```

  Standard input stream. This stream is used for reading in character data.

- **out**

```
public static PrintStream out
```

  Standard output stream. This stream is used for printing messages.

- **err**

```
public static PrintStream err
```

  Standard error stream. This stream can be used to print error messages. Many applications read in data from an InputStream and output messages via the PrintStream out statement. Often applications rely on command line redirection to specify source and destination files. A problem with redirecting standard output is the incapability of writing messages to the screen if

the output has been redirected to a file. This problem can be overcome by sending some output to PrintStream out and other output to PrintStream err. The difference between PrintStream err and PrintStream out is that PrintStream err is often used for displaying error messages but may be used for any purpose.

# Methods

o **setSecurityManager**

```
public static void setSecurityManager(SecurityManager s)
```

Sets the System security. This value can only be set once.
**Parameters:**
s – the security manager
**Throws:** SecurityException
If the SecurityManager has already been set.

o **getSecurityManager**

```
public static SecurityManager getSecurityManager()
```

Gets the system security interface.

o **currentTimeMillis**

```
public static long currentTimeMillis()
```

Returns the current time in milliseconds GMT since the epoch (00:00:00 UTC, January 1, 1970). It is a signed 64 bit integer, and so it will not overflow until the year 292280995.
**See Also:**
Date

o **arraycopy**

```
public static void arraycopy(Object src,
                             int src_position,
                             Object dst,
                             int dst_position,
                             int length)
```

Copies an array from the source array, beginning at the specified position, to the specified position of the destination array. This method does not allocate memory for the destination array. The memory must already be allocated.
**Parameters:**
src – the source data
srcpos – start position in the source data
dest – the destination
destpos – start position in the destination data
length – the number of array elements to be copied

**Throws:** ArrayIndexOutOfBoundsException
If copy would cause access of data outside array bounds.
**Throws:** ArrayStoreException
If an element in the src array could could not be stored into the destination array due to a type mismatch

o **getProperties**

```
public static Properties getProperties()
```

Gets the System properties.

o **setProperties**

```
public static void setProperties(Properties props)
```

Sets the System properties to the specified properties.
**Parameters:**
props – the properties to be set

o **getProperty**

```
public static String getProperty(String key)
```

Gets the System property indicated by the specified key.
**Parameters:**
key – the location of the system property

o **getProperty**

```
public static String getProperty(String key,
                                 String def)
```

Gets the System property indicated by the specified key and def.

o **getenv**

```
public static String getenv(String name)
```

Obsolete. Gets an environment variable. An environment variable is a system dependent external variable that has a string value.
**Parameters:**
name – the name of the environment variable
**Returns:**
the value of the variable, or null if the variable is not defined.

o **exit**

```
public static 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.

**See Also:**

exit

o **gc**

```
public static void gc()
```

Runs the garbage collector.

**See Also:**

gc

o **runFinalization**

```
public static void runFinalization()
```

Runs the finalization methods of any objects pending finalization.

**See Also:**

gc

o **load**

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

Loads a dynamic library, given a complete path name.

**Parameters:**

filename – the file to load

**Throws:** UnsatisfiedLinkError

If the file does not exist.

**See Also:**

load

o **loadLibrary**

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

Loads a dynamic library with the specified library name.

**Parameters:**

libname – the name of the library

**Throws:** UnsatisfiedLinkError

If the library does not exist.

**See Also:**

loadLibrary

---