
Class java.lang.SecurityManager

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

---

public class **SecurityManager**
extends Object

Security Manager. An abstract class that can be subclassed to implement a security policy. It allows the inspection of the classloaders on the execution stack.

Author:
Arthur van Hoff
Version:
1.19, 10/24/95

Constructor Index

Method Index

Checks to see if the system command is executed by trusted code.

- **checkExit**(int)
  Checks to see if the system has exited the virtual machine with an exit code.
- **checkLink**(String)
  Checks to see if the specified linked library exists.
- **checkListen**(int)
  Checks to see if a server socket is listening to the specified local port that it is bounded to.
- **checkPackageAccess**(String)
  Check if an applet can access a package.
- **checkPackageDefinition**(String)
  Check if an applet can define classes in a package.
- **checkPropertiesAccess**()
  Checks to see who has access to the System properties.
- **checkRead**(int)
  Checks to see if an input file with the specified system dependent file descriptor gets created.
- **checkRead**(String)
  Checks to see if an input file with the specified system dependent file name gets created.
- **checkSetFactory**()
  Check if an applet can set a networking–related object factory.
- **checkTopLevelWindow**()
  Checks to see if top–level windows can be created by the caller.
- **checkWrite**(int)
  Checks to see if an output file with the specified system dependent file descriptor gets created.
- **checkWrite**(String)
  Checks to see if an output file with the specified system dependent file name gets created.
- **classDepth**(String)
  Return the position of the stack frame containing the first occurrence of the named class.
- **classLoaderDepth**()
- **currentClassLoader**()
  The current ClassLoader on the execution stack.
- **getClassContext**()
  Gets the context of this Class.
- **inClass**(String)
  Returns true if the specified String is in this Class.
- **inClassLoader**()
  Returns a boolean indicating whether or not the current ClassLoader is equal to null.

Constructors

- **SecurityManager**

```
protected SecurityManager()
```

Constructs a new SecurityManager.
Throws: SecurityException
If the security manager cannot be created.

Methods

o getClassContext

```
protected Class[] getClassContext()
```

Gets the context of this Class.

o currentClassLoader

```
protected ClassLoader currentClassLoader()
```

The current ClassLoader on the execution stack.

o classDepth

```
protected int classDepth(String name)
```

Return the position of the stack frame containing the first occurrence of the named class.
Parameters:
name – classname of the class to search for

o classLoaderDepth

```
protected int classLoaderDepth()
```

o inClass

```
protected boolean inClass(String name)
```

Returns true if the specified String is in this Class.
Parameters:
name – the name of the class

o inClassLoader

```
protected boolean inClassLoader()
```

Returns a boolean indicating whether or not the current ClassLoader is equal to null.

- **checkCreateClassLoader**

```
public void checkCreateClassLoader()
```

Checks to see if the ClassLoader has been created.
Throws: SecurityException
If a security error has occurred.

- **checkAccess**

```
public void checkAccess(Thread g)
```

Checks to see if the specified Thread is allowed to modify the Thread group.
Parameters:
g – the Thread to be checked
Throws: SecurityException
If the current Thread is not allowed to access this Thread group.

- **checkAccess**

```
public void checkAccess(ThreadGroup g)
```

Checks to see if the specified Thread group is allowed to modify this group.
Parameters:
g – the Thread group to be checked
Throws: SecurityException
If the current Thread group is not allowed to access this Thread group.

- **checkExit**

```
public void checkExit(int status)
```

Checks to see if the system has exited the virtual machine with an exit code.
Parameters:
status – exit status, 0 if successful, other values indicate various error types.
Throws: SecurityException
If a security error has occurred.

- **checkExec**

```
public void checkExec(String cmd)
```

Checks to see if the system command is executed by trusted code.
Parameters:
cmd – the specified system command
Throws: SecurityException
If a security error has occurred.

- **checkLink**

```
public void checkLink(String lib)
```

Checks to see if the specified linked library exists.
Parameters:
lib – the name of the library
Throws: SecurityException
If the library does not exist.

- **checkRead**

```
public void checkRead(int fd)
```

Checks to see if an input file with the specified system dependent file descriptor gets created.
Parameters:
fd – the system dependent file descriptor
Throws: SecurityException
If a security error has occurred.

- **checkRead**

```
public void checkRead(String file)
```

Checks to see if an input file with the specified system dependent file name gets created.
Parameters:
file – the system dependent file name
Throws: SecurityException
If the file is not found.

- **checkWrite**

```
public void checkWrite(int fd)
```

Checks to see if an output file with the specified system dependent file descriptor gets created.
Parameters:
fd – the system dependent file descriptor
Throws: SecurityException
If a security error has occurred.

- **checkWrite**

```
public void checkWrite(String file)
```

Checks to see if an output file with the specified system dependent file name gets created.
Parameters:

file – the system depepndent file name

Throws: SecurityException

If the file is not found.

- **checkConnect**

```
public void checkConnect(String host,
                         int port)
```

Checks to see if a socket has connected to the specified port on the the specified host.

Parameters:

host – the host name port to connect to

port – the protocol port to connect to

Throws: SecurityException

If a security error has occurred.

- **checkListen**

```
public void checkListen(int port)
```

Checks to see if a server socket is listening to the specified local port that it is bounded to.

Parameters:

port – the protocol port to connect to

Throws: SecurityException

If a security error has occurred.

- **checkAccept**

```
public void checkAccept(String host,
                        int port)
```

Checks to see if a socket connection to the specified port on the specified host has been accepted.

Parameters:

host – the host name to connect to

port – the protocol port to connect to

Throws: SecurityException

If a security error has occurred.

- **checkPropertiesAccess**

```
public void checkPropertiesAccess()
```

Checks to see who has access to the System properties.

Throws: SecurityException

If a security error has occurred.

- **checkTopLevelWindow**

```
public boolean checkTopLevelWindow()
```

  Checks to see if top–level windows can be created by the caller. A return of false means that the window creation is allowed but the window should indicate some sort of visual warning. Returning true means the creation is allowed with no special restrictions. To disallow the creation entirely, this method should throw a SecurityException.

- **checkPackageAccess**

```
public void checkPackageAccess(String pkg)
```

  Check if an applet can access a package.

- **checkPackageDefinition**

```
public void checkPackageDefinition(String pkg)
```

  Check if an applet can define classes in a package.

- **checkSetFactory**

```
public void checkSetFactory()
```

  Check if an applet can set a networking–related object factory.
