| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/ThreadGroup.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/ThreadDeath.html)   [**NEXT CLASS**](http://docs.google.com/java/lang/ThreadLocal.html) | [**FRAMES**](http://docs.google.com/index.html?java/lang/ThreadGroup.html)    [**NO FRAMES**](http://docs.google.com/ThreadGroup.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#3dy6vkm) | [METHOD](#2s8eyo1) |

## **java.lang**

Class ThreadGroup

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

**All Implemented Interfaces:** [Thread.UncaughtExceptionHandler](http://docs.google.com/java/lang/Thread.UncaughtExceptionHandler.html)

public class **ThreadGroup**extends [Object](http://docs.google.com/java/lang/Object.html)implements [Thread.UncaughtExceptionHandler](http://docs.google.com/java/lang/Thread.UncaughtExceptionHandler.html)

A thread group represents a set of threads. In addition, a thread group can also include other thread groups. The thread groups form a tree in which every thread group except the initial thread group has a parent.

A thread is allowed to access information about its own thread group, but not to access information about its thread group's parent thread group or any other thread groups.

**Since:** JDK1.0

| **Constructor Summary** | |
| --- | --- |
| [**ThreadGroup**](http://docs.google.com/java/lang/ThreadGroup.html#ThreadGroup(java.lang.String))([String](http://docs.google.com/java/lang/String.html) name)            Constructs a new thread group. |
| [**ThreadGroup**](http://docs.google.com/java/lang/ThreadGroup.html#ThreadGroup(java.lang.ThreadGroup,%20java.lang.String))([ThreadGroup](http://docs.google.com/java/lang/ThreadGroup.html) parent, [String](http://docs.google.com/java/lang/String.html) name)            Creates a new thread group. |

| **Method Summary** | |
| --- | --- |
| int | [**activeCount**](http://docs.google.com/java/lang/ThreadGroup.html#activeCount())()            Returns an estimate of the number of active threads in this thread group. |
| int | [**activeGroupCount**](http://docs.google.com/java/lang/ThreadGroup.html#activeGroupCount())()            Returns an estimate of the number of active groups in this thread group. |
| boolean | [**allowThreadSuspension**](http://docs.google.com/java/lang/ThreadGroup.html#allowThreadSuspension(boolean))(boolean b)  **Deprecated.** *The definition of this call depends on* [*suspend()*](http://docs.google.com/java/lang/ThreadGroup.html#suspend())*, which is deprecated. Further, the behavior of this call was never specified.* |
| void | [**checkAccess**](http://docs.google.com/java/lang/ThreadGroup.html#checkAccess())()            Determines if the currently running thread has permission to modify this thread group. |
| void | [**destroy**](http://docs.google.com/java/lang/ThreadGroup.html#destroy())()            Destroys this thread group and all of its subgroups. |
| int | [**enumerate**](http://docs.google.com/java/lang/ThreadGroup.html#enumerate(java.lang.Thread%5B%5D))([Thread](http://docs.google.com/java/lang/Thread.html)[] list)            Copies into the specified array every active thread in this thread group and its subgroups. |
| int | [**enumerate**](http://docs.google.com/java/lang/ThreadGroup.html#enumerate(java.lang.Thread%5B%5D,%20boolean))([Thread](http://docs.google.com/java/lang/Thread.html)[] list, boolean recurse)            Copies into the specified array every active thread in this thread group. |
| int | [**enumerate**](http://docs.google.com/java/lang/ThreadGroup.html#enumerate(java.lang.ThreadGroup%5B%5D))([ThreadGroup](http://docs.google.com/java/lang/ThreadGroup.html)[] list)            Copies into the specified array references to every active subgroup in this thread group. |
| int | [**enumerate**](http://docs.google.com/java/lang/ThreadGroup.html#enumerate(java.lang.ThreadGroup%5B%5D,%20boolean))([ThreadGroup](http://docs.google.com/java/lang/ThreadGroup.html)[] list, boolean recurse)            Copies into the specified array references to every active subgroup in this thread group. |
| int | [**getMaxPriority**](http://docs.google.com/java/lang/ThreadGroup.html#getMaxPriority())()            Returns the maximum priority of this thread group. |
| [String](http://docs.google.com/java/lang/String.html) | [**getName**](http://docs.google.com/java/lang/ThreadGroup.html#getName())()            Returns the name of this thread group. |
| [ThreadGroup](http://docs.google.com/java/lang/ThreadGroup.html) | [**getParent**](http://docs.google.com/java/lang/ThreadGroup.html#getParent())()            Returns the parent of this thread group. |
| void | [**interrupt**](http://docs.google.com/java/lang/ThreadGroup.html#interrupt())()            Interrupts all threads in this thread group. |
| boolean | [**isDaemon**](http://docs.google.com/java/lang/ThreadGroup.html#isDaemon())()            Tests if this thread group is a daemon thread group. |
| boolean | [**isDestroyed**](http://docs.google.com/java/lang/ThreadGroup.html#isDestroyed())()            Tests if this thread group has been destroyed. |
| void | [**list**](http://docs.google.com/java/lang/ThreadGroup.html#list())()            Prints information about this thread group to the standard output. |
| boolean | [**parentOf**](http://docs.google.com/java/lang/ThreadGroup.html#parentOf(java.lang.ThreadGroup))([ThreadGroup](http://docs.google.com/java/lang/ThreadGroup.html) g)            Tests if this thread group is either the thread group argument or one of its ancestor thread groups. |
| void | [**resume**](http://docs.google.com/java/lang/ThreadGroup.html#resume())()  **Deprecated.** *This method is used solely in conjunction with Thread.suspend and ThreadGroup.suspend, both of which have been deprecated, as they are inherently deadlock-prone. See* [*Thread.suspend()*](http://docs.google.com/java/lang/Thread.html#suspend()) *for details.* |
| void | [**setDaemon**](http://docs.google.com/java/lang/ThreadGroup.html#setDaemon(boolean))(boolean daemon)            Changes the daemon status of this thread group. |
| void | [**setMaxPriority**](http://docs.google.com/java/lang/ThreadGroup.html#setMaxPriority(int))(int pri)            Sets the maximum priority of the group. |
| void | [**stop**](http://docs.google.com/java/lang/ThreadGroup.html#stop())()  **Deprecated.** *This method is inherently unsafe. See* [*Thread.stop()*](http://docs.google.com/java/lang/Thread.html#stop()) *for details.* |
| void | [**suspend**](http://docs.google.com/java/lang/ThreadGroup.html#suspend())()  **Deprecated.** *This method is inherently deadlock-prone. See* [*Thread.suspend()*](http://docs.google.com/java/lang/Thread.html#suspend()) *for details.* |
| [String](http://docs.google.com/java/lang/String.html) | [**toString**](http://docs.google.com/java/lang/ThreadGroup.html#toString())()            Returns a string representation of this Thread group. |
| void | [**uncaughtException**](http://docs.google.com/java/lang/ThreadGroup.html#uncaughtException(java.lang.Thread,%20java.lang.Throwable))([Thread](http://docs.google.com/java/lang/Thread.html) t, [Throwable](http://docs.google.com/java/lang/Throwable.html) e)            Called by the Java Virtual Machine when a thread in this thread group stops because of an uncaught exception, and the thread does not have a specific [Thread.UncaughtExceptionHandler](http://docs.google.com/java/lang/Thread.UncaughtExceptionHandler.html) installed. |

| **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()), [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)) |

| **Constructor Detail** |
| --- |

### ThreadGroup

public **ThreadGroup**([String](http://docs.google.com/java/lang/String.html) name)

Constructs a new thread group. The parent of this new group is the thread group of the currently running thread.

The checkAccess method of the parent thread group is called with no arguments; this may result in a security exception.

**Parameters:**name - the name of the new thread group. **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if the current thread cannot create a thread in the specified thread group.**Since:** JDK1.0 **See Also:**[checkAccess()](http://docs.google.com/java/lang/ThreadGroup.html#checkAccess())

### ThreadGroup

public **ThreadGroup**([ThreadGroup](http://docs.google.com/java/lang/ThreadGroup.html) parent,  
 [String](http://docs.google.com/java/lang/String.html) name)

Creates a new thread group. The parent of this new group is the specified thread group.

The checkAccess method of the parent thread group is called with no arguments; this may result in a security exception.

**Parameters:**parent - the parent thread group.name - the name of the new thread group. **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the thread group argument is null. [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if the current thread cannot create a thread in the specified thread group.**Since:** JDK1.0 **See Also:**[SecurityException](http://docs.google.com/java/lang/SecurityException.html), [checkAccess()](http://docs.google.com/java/lang/ThreadGroup.html#checkAccess())

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

### getName

public final [String](http://docs.google.com/java/lang/String.html) **getName**()

Returns the name of this thread group.

**Returns:**the name of this thread group.**Since:** JDK1.0

### getParent

public final [ThreadGroup](http://docs.google.com/java/lang/ThreadGroup.html) **getParent**()

Returns the parent of this thread group.

First, if the parent is not null, the checkAccess method of the parent thread group is called with no arguments; this may result in a security exception.

**Returns:**the parent of this thread group. The top-level thread group is the only thread group whose parent is null. **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if the current thread cannot modify this thread group.**Since:** JDK1.0 **See Also:**[checkAccess()](http://docs.google.com/java/lang/ThreadGroup.html#checkAccess()), [SecurityException](http://docs.google.com/java/lang/SecurityException.html), [RuntimePermission](http://docs.google.com/java/lang/RuntimePermission.html)

### getMaxPriority

public final int **getMaxPriority**()

Returns the maximum priority of this thread group. Threads that are part of this group cannot have a higher priority than the maximum priority.

**Returns:**the maximum priority that a thread in this thread group can have.**Since:** JDK1.0 **See Also:**[setMaxPriority(int)](http://docs.google.com/java/lang/ThreadGroup.html#setMaxPriority(int))

### isDaemon

public final boolean **isDaemon**()

Tests if this thread group is a daemon thread group. A daemon thread group is automatically destroyed when its last thread is stopped or its last thread group is destroyed.

**Returns:**true if this thread group is a daemon thread group; false otherwise.**Since:** JDK1.0

### isDestroyed

public boolean **isDestroyed**()

Tests if this thread group has been destroyed.

**Returns:**true if this object is destroyed**Since:** JDK1.1

### setDaemon

public final void **setDaemon**(boolean daemon)

Changes the daemon status of this thread group.

First, the checkAccess method of this thread group is called with no arguments; this may result in a security exception.

A daemon thread group is automatically destroyed when its last thread is stopped or its last thread group is destroyed.

**Parameters:**daemon - if true, marks this thread group as a daemon thread group; otherwise, marks this thread group as normal. **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if the current thread cannot modify this thread group.**Since:** JDK1.0 **See Also:**[SecurityException](http://docs.google.com/java/lang/SecurityException.html), [checkAccess()](http://docs.google.com/java/lang/ThreadGroup.html#checkAccess())

### setMaxPriority

public final void **setMaxPriority**(int pri)

Sets the maximum priority of the group. Threads in the thread group that already have a higher priority are not affected.

First, the checkAccess method of this thread group is called with no arguments; this may result in a security exception.

If the pri argument is less than [Thread.MIN\_PRIORITY](http://docs.google.com/java/lang/Thread.html#MIN_PRIORITY) or greater than [Thread.MAX\_PRIORITY](http://docs.google.com/java/lang/Thread.html#MAX_PRIORITY), the maximum priority of the group remains unchanged.

Otherwise, the priority of this ThreadGroup object is set to the smaller of the specified pri and the maximum permitted priority of the parent of this thread group. (If this thread group is the system thread group, which has no parent, then its maximum priority is simply set to pri.) Then this method is called recursively, with pri as its argument, for every thread group that belongs to this thread group.

**Parameters:**pri - the new priority of the thread group. **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if the current thread cannot modify this thread group.**Since:** JDK1.0 **See Also:**[getMaxPriority()](http://docs.google.com/java/lang/ThreadGroup.html#getMaxPriority()), [SecurityException](http://docs.google.com/java/lang/SecurityException.html), [checkAccess()](http://docs.google.com/java/lang/ThreadGroup.html#checkAccess())

### parentOf

public final boolean **parentOf**([ThreadGroup](http://docs.google.com/java/lang/ThreadGroup.html) g)

Tests if this thread group is either the thread group argument or one of its ancestor thread groups.

**Parameters:**g - a thread group. **Returns:**true if this thread group is the thread group argument or one of its ancestor thread groups; false otherwise.**Since:** JDK1.0

### checkAccess

public final void **checkAccess**()

Determines if the currently running thread has permission to modify this thread group.

If there is a security manager, its checkAccess method is called with this thread group as its argument. This may result in throwing a SecurityException.

**Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if the current thread is not allowed to access this thread group.**Since:** JDK1.0 **See Also:**[SecurityManager.checkAccess(java.lang.ThreadGroup)](http://docs.google.com/java/lang/SecurityManager.html#checkAccess(java.lang.ThreadGroup))

### activeCount

public int **activeCount**()

Returns an estimate of the number of active threads in this thread group. The result might not reflect concurrent activity, and might be affected by the presence of certain system threads.

Due to the inherently imprecise nature of the result, it is recommended that this method only be used for informational purposes.

**Returns:**an estimate of the number of active threads in this thread group and in any other thread group that has this thread group as an ancestor.**Since:** JDK1.0

### enumerate

public int **enumerate**([Thread](http://docs.google.com/java/lang/Thread.html)[] list)

Copies into the specified array every active thread in this thread group and its subgroups.

First, the checkAccess method of this thread group is called with no arguments; this may result in a security exception.

An application might use the activeCount method to get an estimate of how big the array should be, however *if the array is too short to hold all the threads, the extra threads are silently ignored.* If it is critical to obtain every active thread in this thread group and its subgroups, the caller should verify that the returned int value is strictly less than the length of list.

Due to the inherent race condition in this method, it is recommended that the method only be used for informational purposes.

**Parameters:**list - an array into which to place the list of threads. **Returns:**the number of threads put into the array. **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if the current thread does not have permission to enumerate this thread group.**Since:** JDK1.0 **See Also:**[activeCount()](http://docs.google.com/java/lang/ThreadGroup.html#activeCount()), [checkAccess()](http://docs.google.com/java/lang/ThreadGroup.html#checkAccess())

### enumerate

public int **enumerate**([Thread](http://docs.google.com/java/lang/Thread.html)[] list,  
 boolean recurse)

Copies into the specified array every active thread in this thread group. If the recurse flag is true, references to every active thread in this thread's subgroups are also included. If the array is too short to hold all the threads, the extra threads are silently ignored.

First, the checkAccess method of this thread group is called with no arguments; this may result in a security exception.

An application might use the activeCount method to get an estimate of how big the array should be, however *if the array is too short to hold all the threads, the extra threads are silently ignored.* If it is critical to obtain every active thread in this thread group, the caller should verify that the returned int value is strictly less than the length of list.

Due to the inherent race condition in this method, it is recommended that the method only be used for informational purposes.

**Parameters:**list - an array into which to place the list of threads.recurse - a flag indicating whether also to include threads in thread groups that are subgroups of this thread group. **Returns:**the number of threads placed into the array. **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if the current thread does not have permission to enumerate this thread group.**Since:** JDK1.0 **See Also:**[activeCount()](http://docs.google.com/java/lang/ThreadGroup.html#activeCount()), [checkAccess()](http://docs.google.com/java/lang/ThreadGroup.html#checkAccess())

### activeGroupCount

public int **activeGroupCount**()

Returns an estimate of the number of active groups in this thread group. The result might not reflect concurrent activity.

Due to the inherently imprecise nature of the result, it is recommended that this method only be used for informational purposes.

**Returns:**the number of active thread groups with this thread group as an ancestor.**Since:** JDK1.0

### enumerate

public int **enumerate**([ThreadGroup](http://docs.google.com/java/lang/ThreadGroup.html)[] list)

Copies into the specified array references to every active subgroup in this thread group.

First, the checkAccess method of this thread group is called with no arguments; this may result in a security exception.

An application might use the activeGroupCount method to get an estimate of how big the array should be, however *if the array is too short to hold all the thread groups, the extra thread groups are silently ignored.* If it is critical to obtain every active subgroup in this thread group, the caller should verify that the returned int value is strictly less than the length of list.

Due to the inherent race condition in this method, it is recommended that the method only be used for informational purposes.

**Parameters:**list - an array into which to place the list of thread groups. **Returns:**the number of thread groups put into the array. **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if the current thread does not have permission to enumerate this thread group.**Since:** JDK1.0 **See Also:**[activeGroupCount()](http://docs.google.com/java/lang/ThreadGroup.html#activeGroupCount()), [checkAccess()](http://docs.google.com/java/lang/ThreadGroup.html#checkAccess())

### enumerate

public int **enumerate**([ThreadGroup](http://docs.google.com/java/lang/ThreadGroup.html)[] list,  
 boolean recurse)

Copies into the specified array references to every active subgroup in this thread group. If the recurse flag is true, references to all active subgroups of the subgroups and so forth are also included.

First, the checkAccess method of this thread group is called with no arguments; this may result in a security exception.

An application might use the activeGroupCount method to get an estimate of how big the array should be, however *if the array is too short to hold all the thread groups, the extra thread groups are silently ignored.* If it is critical to obtain every active subgroup in this thread group, the caller should verify that the returned int value is strictly less than the length of list.

Due to the inherent race condition in this method, it is recommended that the method only be used for informational purposes.

**Parameters:**list - an array into which to place the list of threads.recurse - a flag indicating whether to recursively enumerate all included thread groups. **Returns:**the number of thread groups put into the array. **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if the current thread does not have permission to enumerate this thread group.**Since:** JDK1.0 **See Also:**[activeGroupCount()](http://docs.google.com/java/lang/ThreadGroup.html#activeGroupCount()), [checkAccess()](http://docs.google.com/java/lang/ThreadGroup.html#checkAccess())

### stop

[@Deprecated](http://docs.google.com/java/lang/Deprecated.html)  
public final void **stop**()

**Deprecated.** *This method is inherently unsafe. See* [*Thread.stop()*](http://docs.google.com/java/lang/Thread.html#stop()) *for details.*

Stops all threads in this thread group.

First, the checkAccess method of this thread group is called with no arguments; this may result in a security exception.

This method then calls the stop method on all the threads in this thread group and in all of its subgroups.

**Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if the current thread is not allowed to access this thread group or any of the threads in the thread group.**Since:** JDK1.0 **See Also:**[SecurityException](http://docs.google.com/java/lang/SecurityException.html), [Thread.stop()](http://docs.google.com/java/lang/Thread.html#stop()), [checkAccess()](http://docs.google.com/java/lang/ThreadGroup.html#checkAccess())

### interrupt

public final void **interrupt**()

Interrupts all threads in this thread group.

First, the checkAccess method of this thread group is called with no arguments; this may result in a security exception.

This method then calls the interrupt method on all the threads in this thread group and in all of its subgroups.

**Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if the current thread is not allowed to access this thread group or any of the threads in the thread group.**Since:** 1.2 **See Also:**[Thread.interrupt()](http://docs.google.com/java/lang/Thread.html#interrupt()), [SecurityException](http://docs.google.com/java/lang/SecurityException.html), [checkAccess()](http://docs.google.com/java/lang/ThreadGroup.html#checkAccess())

### suspend

[@Deprecated](http://docs.google.com/java/lang/Deprecated.html)  
public final void **suspend**()

**Deprecated.** *This method is inherently deadlock-prone. See* [*Thread.suspend()*](http://docs.google.com/java/lang/Thread.html#suspend()) *for details.*

Suspends all threads in this thread group.

First, the checkAccess method of this thread group is called with no arguments; this may result in a security exception.

This method then calls the suspend method on all the threads in this thread group and in all of its subgroups.

**Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if the current thread is not allowed to access this thread group or any of the threads in the thread group.**Since:** JDK1.0 **See Also:**[Thread.suspend()](http://docs.google.com/java/lang/Thread.html#suspend()), [SecurityException](http://docs.google.com/java/lang/SecurityException.html), [checkAccess()](http://docs.google.com/java/lang/ThreadGroup.html#checkAccess())

### resume

[@Deprecated](http://docs.google.com/java/lang/Deprecated.html)  
public final void **resume**()

**Deprecated.** *This method is used solely in conjunction with Thread.suspend and ThreadGroup.suspend, both of which have been deprecated, as they are inherently deadlock-prone. See* [*Thread.suspend()*](http://docs.google.com/java/lang/Thread.html#suspend()) *for details.*

Resumes all threads in this thread group.

First, the checkAccess method of this thread group is called with no arguments; this may result in a security exception.

This method then calls the resume method on all the threads in this thread group and in all of its sub groups.

**Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if the current thread is not allowed to access this thread group or any of the threads in the thread group.**Since:** JDK1.0 **See Also:**[SecurityException](http://docs.google.com/java/lang/SecurityException.html), [Thread.resume()](http://docs.google.com/java/lang/Thread.html#resume()), [checkAccess()](http://docs.google.com/java/lang/ThreadGroup.html#checkAccess())

### destroy

public final void **destroy**()

Destroys this thread group and all of its subgroups. This thread group must be empty, indicating that all threads that had been in this thread group have since stopped.

First, the checkAccess method of this thread group is called with no arguments; this may result in a security exception.

**Throws:** [IllegalThreadStateException](http://docs.google.com/java/lang/IllegalThreadStateException.html) - if the thread group is not empty or if the thread group has already been destroyed. [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if the current thread cannot modify this thread group.**Since:** JDK1.0 **See Also:**[checkAccess()](http://docs.google.com/java/lang/ThreadGroup.html#checkAccess())

### list

public void **list**()

Prints information about this thread group to the standard output. This method is useful only for debugging.

**Since:** JDK1.0

### uncaughtException

public void **uncaughtException**([Thread](http://docs.google.com/java/lang/Thread.html) t,  
 [Throwable](http://docs.google.com/java/lang/Throwable.html) e)

Called by the Java Virtual Machine when a thread in this thread group stops because of an uncaught exception, and the thread does not have a specific [Thread.UncaughtExceptionHandler](http://docs.google.com/java/lang/Thread.UncaughtExceptionHandler.html) installed.

The uncaughtException method of ThreadGroup does the following:

* If this thread group has a parent thread group, the uncaughtException method of that parent is called with the same two arguments.
* Otherwise, this method checks to see if there is a [default uncaught exception handler](http://docs.google.com/java/lang/Thread.html#getDefaultUncaughtExceptionHandler()) installed, and if so, its uncaughtException method is called with the same two arguments.
* Otherwise, this method determines if the Throwable argument is an instance of [ThreadDeath](http://docs.google.com/java/lang/ThreadDeath.html). If so, nothing special is done. Otherwise, a message containing the thread's name, as returned from the thread's [getName](http://docs.google.com/java/lang/Thread.html#getName()) method, and a stack backtrace, using the Throwable's [printStackTrace](http://docs.google.com/java/lang/Throwable.html#printStackTrace()) method, is printed to the [standard error stream](http://docs.google.com/java/lang/System.html#err).

Applications can override this method in subclasses of ThreadGroup to provide alternative handling of uncaught exceptions.

**Specified by:**[uncaughtException](http://docs.google.com/java/lang/Thread.UncaughtExceptionHandler.html#uncaughtException(java.lang.Thread,%20java.lang.Throwable)) in interface [Thread.UncaughtExceptionHandler](http://docs.google.com/java/lang/Thread.UncaughtExceptionHandler.html) **Parameters:**t - the thread that is about to exit.e - the uncaught exception.**Since:** JDK1.0

### allowThreadSuspension

[@Deprecated](http://docs.google.com/java/lang/Deprecated.html)  
public boolean **allowThreadSuspension**(boolean b)

**Deprecated.** *The definition of this call depends on* [*suspend()*](http://docs.google.com/java/lang/ThreadGroup.html#suspend())*, which is deprecated. Further, the behavior of this call was never specified.*

Used by VM to control lowmem implicit suspension.

**Parameters:**b - boolean to allow or disallow suspension **Returns:**true on success**Since:** JDK1.1

### toString

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

Returns a string representation of this Thread group.

**Overrides:**[toString](http://docs.google.com/java/lang/Object.html#toString()) in class [Object](http://docs.google.com/java/lang/Object.html) **Returns:**a string representation of this thread group.**Since:** JDK1.0

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/ThreadGroup.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/ThreadDeath.html)   [**NEXT CLASS**](http://docs.google.com/java/lang/ThreadLocal.html) | [**FRAMES**](http://docs.google.com/index.html?java/lang/ThreadGroup.html)    [**NO FRAMES**](http://docs.google.com/ThreadGroup.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#3dy6vkm) | [METHOD](#2s8eyo1) |

[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).