---

# Class java.lang.ThreadGroup

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

---

public class **ThreadGroup**
extends Object

A group of Threads. A Thread group can contain a set of Threads as well as a set of other Thread groups. A Thread can access its Thread group, but it can't access the parent of its Thread group. This makes it possible to encapsulate a Thread in a Thread group and stop it from manipulating Threads in the parent group.

**Version:**
1.18, 10/21/95
**Author:**
Arthur van Hoff

---

# Constructor Index

o **ThreadGroup**(String)
Creates a new ThreadGroup.
o **ThreadGroup**(ThreadGroup, String)
Creates a new ThreadGroup with a specified name in the specified Thread group.

# Method Index

o **activeCount**()
Returns an estimate of the number of active Threads in the Thread group.
o **activeGroupCount**()
Returns an estimate of the number of active groups in the Thread group.
o **checkAccess**()
Checks to see if the current Thread is allowed to modify this group.
o **destroy**()
Destroys a Thread group.
o **enumerate**(Thread[])

Copies, into the specified array, references to every active Thread in this Thread group.

- **enumerate**(Thread[], boolean)
  Copies, into the specified array, references to every active Thread in this Thread group.
- **enumerate**(ThreadGroup[])
  Copies, into the specified array, references to every active Thread group in this Thread group.
- **enumerate**(ThreadGroup[], boolean)
  Copies, into the specified array, references to every active Thread group in this Thread group.
- **getMaxPriority**()
  Gets the maximum priority of the group.
- **getName**()
  Gets the name of this Thread group.
- **getParent**()
  Gets the parent of this Thread group.
- **isDaemon**()
  Returns the daemon flag of the Thread group.
- **list**()
  Lists this Thread group.
- **parentOf**(ThreadGroup)
  Checks to see if this Thread group is a parent of or is equal to another Thread group.
- **resume**()
  Resumes all the Threads in this Thread group and all of its sub groups.
- **setDaemon**(boolean)
  Changes the daemon status of this group.
- **setMaxPriority**(int)
  Sets the maximum priority of the group.
- **stop**()
  Stops all the Threads in this Thread group and all of its sub groups.
- **suspend**()
  Suspends all the Threads in this Thread group and all of its sub groups.
- **toString**()
  Returns a String representation of the Thread group.

# Constructors

- **ThreadGroup**

```
public ThreadGroup(String name)
```

Creates a new ThreadGroup. Its parent will be the Thread group of the current Thread.

**Parameters:**

name – the name of the new Thread group created

- **ThreadGroup**

```
public ThreadGroup(ThreadGroup parent,
                   String name)
```

Creates a new ThreadGroup with a specified name in the specified Thread group.

**Parameters:**
parent – the specified parent Thread group
name – the name of the new Thread group being created
**Throws:** NullPointerException
If the given thread group is equal to null.

# Methods

o **getName**

```
public final String getName()
```

Gets the name of this Thread group.

o **getParent**

```
public final ThreadGroup getParent()
```

Gets the parent of this Thread group.

o **getMaxPriority**

```
public final int getMaxPriority()
```

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

o **isDaemon**

```
public final boolean isDaemon()
```

Returns the daemon flag of the Thread group. A daemon Thread group is automatically destroyed when it is found empty after a Thread group or Thread is removed from it.

o **setDaemon**

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

Changes the daemon status of this group.
**Parameters:**
daemon – the daemon boolean which is to be set.

o **setMaxPriority**

```
public final synchronized void setMaxPriority(int pri)
```

Sets the maximum priority of the group. Threads that are already in the group **can** have a higher priority than the set maximum.

**Parameters:**
pri – the priority of the Thread group

o **parentOf**

```
public final boolean parentOf(ThreadGroup g)
```

Checks to see if this Thread group is a parent of or is equal to another Thread group.
**Parameters:**
g – the Thread group to be checked
**Returns:**
true if this Thread group is equal to or is the parent of another Thread group; false otherwise.

o **checkAccess**

```
public final void checkAccess()
```

Checks to see if the current Thread is allowed to modify this group.
**Throws:** SecurityException
If the current Thread is not allowed to access this Thread group.

o **activeCount**

```
public synchronized int activeCount()
```

Returns an estimate of the number of active Threads in the Thread group.

o **enumerate**

```
public int enumerate(Thread list[])
```

Copies, into the specified array, references to every active Thread in this Thread group. You can use the activeCount() method to get an estimate of how big the array should be.
**Parameters:**
list – an array of Threads
**Returns:**
the number of Threads put into the array

o **enumerate**

```
public int enumerate(Thread list[],
                     boolean recurse)
```

Copies, into the specified array, references to every active Thread in this Thread group. You can use the activeCount() method to get an estimate of how big the array should be.
**Parameters:**
list – an array list of Threads
recurse – a boolean indicating whether a Thread has reapearred

**Returns:**
the number of Threads placed into the array.

o **activeGroupCount**

```
public synchronized int activeGroupCount()
```

Returns an estimate of the number of active groups in the Thread group.

o **enumerate**

```
public int enumerate(ThreadGroup list[])
```

Copies, into the specified array, references to every active Thread group in this Thread group. You can use the activeGroupCount() method to get an estimate of how big the array should be.
**Parameters:**
list – an array of Thread groups
**Returns:**
the number of Thread groups placed into the array.

o **enumerate**

```
public int enumerate(ThreadGroup list[],
                     boolean recurse)
```

Copies, into the specified array, references to every active Thread group in this Thread group. You can use the activeGroupCount() method to get an estimate of how big the array should be.
**Parameters:**
list – an array list of Thread groups
recurse – a boolean indicating if a Thread group has reappeared
**Returns:**
the number of Thread groups placed into the array.

o **stop**

```
public final synchronized void stop()
```

Stops all the Threads in this Thread group and all of its sub groups.

o **suspend**

```
public final synchronized void suspend()
```

Suspends all the Threads in this Thread group and all of its sub groups.

o **resume**

```
public final synchronized void resume()
```

Resumes all the Threads in this Thread group and all of its sub groups.

o **destroy**

```
public final synchronized void destroy()
```

Destroys a Thread group. This does **NOT** stop the Threads in the Thread group.
**Throws:** IllegalThreadStateException
If the Thread group is not empty or if the Thread group was already destroyed.

o **list**

```
public synchronized void list()
```

Lists this Thread group. Useful for debugging only.

o **toString**

```
public String toString()
```

Returns a String representation of the Thread group.
**Overrides:**
toString in class Object

---