ThreadGroup in Java

Java provides a convenient way to group multiple threads in a single object. In such way, we can suspend, resume or interrupt group of threads by a single method call.

Note: Now suspend(), resume() and stop() methods are deprecated.

Java thread group is implemented by *java.lang.ThreadGroup* class.

A ThreadGroup represents a set of threads. A thread group can also include the other thread group. The thread group creates 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 it cannot access the information about its thread group's parent thread group or any other thread groups.

Constructors of ThreadGroup class

There are only two constructors of ThreadGroup class.

|  |  |  |
| --- | --- | --- |
| **No.** | **Constructor** | **Description** |
| 1) | ThreadGroup(String name) | creates a thread group with given name. |
| 2) | ThreadGroup(ThreadGroup parent, String name) | creates a thread group with given parent group and name. |

Methods of ThreadGroup class

There are many methods in ThreadGroup class. A list of ThreadGroup methods are given below.

|  |  |  |  |
| --- | --- | --- | --- |
| **S.N.** | **Modifier and Type** | **Method** | **Description** |
| 1) | void | [checkAccess()](https://www.javatpoint.com/java-threadgroup-checkaccess-method) | This method determines if the currently running thread has  permission to modify the thread group. |
| 2) | int | [activeCount()](https://www.javatpoint.com/java-threadgroup-activecount-method) | This method returns an estimate of the number of  active threads in the thread group and its subgroups. |
| 3) | int | [activeGroupCount()](https://www.javatpoint.com/java-threadgroup-activegroupcount-method) | This method returns an estimate of the number of active  groups in the thread group and its subgroups. |
| 4) | void | [destroy()](https://www.javatpoint.com/java-threadgroup-destroy-method) | This method destroys the thread group and all of its subgroups. |
| 5) | int | [enumerate(Thread[] list)](https://www.javatpoint.com/java-threadgroup-enumerate-method) | This method copies into the specified array every active  thread in the thread group and its subgroups. |
| 6) | int | [getMaxPriority()](https://www.javatpoint.com/java-threadgroup-getmaxpriority-method) | This method returns the maximum priority of the thread group. |
| 7) | String | [getName()](https://www.javatpoint.com/java-threadgroup-getname-method) | This method returns the name of the thread group. |
| 8) | ThreadGroup | [getParent()](https://www.javatpoint.com/java-threadgroup-getparent-method) | This method returns the parent of the thread group. |
| 9) | void | [interrupt()](https://www.javatpoint.com/java-threadgroup-interrupt-method) | This method interrupts all threads in the thread group. |
| 10) | boolean | [isDaemon()](https://www.javatpoint.com/java-threadgroup-isdaemon-method) | This method tests if the thread group is a daemon thread group. |
| 11) | void | [setDaemon(boolean daemon)](https://www.javatpoint.com/java-threadgroup-setdaemon-method) | This method changes the daemon status of the thread group. |
| 12) | boolean | [isDestroyed()](https://www.javatpoint.com/java-threadgroup-isdestroyed-method) | This method tests if this thread group has been destroyed. |
| 13) | void | [list()](https://www.javatpoint.com/java-threadgroup-list-method) | This method prints information about the thread group to the  standard output. |
| 14) | boolean | [parentOf(ThreadGroup g](https://www.javatpoint.com/java-threadgroup-parentof-method) | This method tests if the thread group is either the thread group  argument or one of its ancestor thread groups. |
| 15) | void | [suspend()](https://www.javatpoint.com/java-threadgroup-suspend-method) | This method is used to suspend all threads in the thread group. |
| 16) | void | [resume()](https://www.javatpoint.com/java-threadgroup-resume-method) | This method is used to resume all threads in the thread group  which was suspended using suspend() method. |
| 17) | void | [setMaxPriority(int pri)](https://www.javatpoint.com/java-threadgroup-setmaxpriority-method) | This method sets the maximum priority of the group. |
| 18) | void | [stop()](https://www.javatpoint.com/java-threadgroup-stop-method) | This method is used to stop all threads in the thread group. |
| 19) | String | [toString()](https://www.javatpoint.com/java-threadgroup-tostring-method) | This method returns a string representation of the Thread group. |

Let's see a code to group multiple threads.

1. ThreadGroup tg1 = **new** ThreadGroup("Group A");
2. Thread t1 = **new** Thread(tg1,**new** MyRunnable(),"one");
3. Thread t2 = **new** Thread(tg1,**new** MyRunnable(),"two");
4. Thread t3 = **new** Thread(tg1,**new** MyRunnable(),"three");

Now all 3 threads belong to one group. Here, tg1 is the thread group name, MyRunnable is the class that implements Runnable interface and "one", "two" and "three" are the thread names.

Now we can interrupt all threads by a single line of code only.

1. Thread.currentThread().getThreadGroup().interrupt();

ThreadGroup Example

*File: ThreadGroupDemo.java*

1. **public** **class** ThreadGroupDemo **implements** Runnable{
2. **public** **void** run() {
3. System.out.println(Thread.currentThread().getName());
4. }
5. **public** **static** **void** main(String[] args) {
6. ThreadGroupDemo runnable = **new** ThreadGroupDemo();
7. ThreadGroup tg1 = **new** ThreadGroup("Parent ThreadGroup");
9. Thread t1 = **new** Thread(tg1, runnable,"one");
10. t1.start();
11. Thread t2 = **new** Thread(tg1, runnable,"two");
12. t2.start();
13. Thread t3 = **new** Thread(tg1, runnable,"three");
14. t3.start();
16. System.out.println("Thread Group Name: "+tg1.getName());
17. tg1.list();
19. }
20. }

Output:

one

two

three

Thread Group Name: Parent ThreadGroup

java.lang.ThreadGroup[name=Parent ThreadGroup,maxpri=10]

Thread[one,5,Parent ThreadGroup]

Thread[two,5,Parent ThreadGroup]

Thread[three,5,Parent ThreadGroup]