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 an

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()	This method determines if the currently running thr modify the thread group.
2)	Int	activeCount()	This method returns an estimate of the number of a thread group and its subgroups.

3)	Int	activeGroupCount()	This method returns an estimate of the number of a thread group and its subgroups.
4)	Void	destroy()	This method destroys the thread group and all of its
5)	Int	enumerate(Thread[] list)	This method copies into the specified array every ac thread group and its subgroups.
6)	Int	getMaxPriority()	This method returns the maximum priority of the th
7)	String	getName()	This method returns the name of the thread group.
8)	ThreadGroup	getParent()	This method returns the parent of the thread group
9)	Void	interrupt()	This method interrupts all threads in the thread gro
10)	boolean	isDaemon()	This method tests if the thread group is a daemon t
11)	Void	setDaemon(boolean daemon)	This method changes the daemon status of the thre
12)	boolean	isDestroyed()	This method tests if this thread group has been des
13)	Void	list()	This method prints information about the thread gro output.
14)	boolean	parentOf(ThreadGroup g	This method tests if the thread group is either the t or one of its ancestor thread groups.
15)	Void	suspend()	This method is used to suspend all threads in the th
16)	Void	resume()	This method is used to resume all threads in the thr suspended using suspend() method.
17)	Void	setMaxPriority(int pri)	This method sets the maximum priority of the group
18)	Void	stop()	This method is used to stop all threads in the thread
19)	String	toString()	This method returns a string representation of the T

Let's see a code to group multiple
threads.ThreadGroup tg1 = **new** ThreadGroup("Group A");

```
Thread t1 = new Thread(tg1,new MyRunnable(),"one");
Thread t2 = new Thread(tg1,new MyRunnable(),"two");
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.

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

ThreadGroup Example

File: ThreadGroupDemo.java

```
public class ThreadGroupDemo implements Runnable {
    public void run() {
        System.out.println(Thread.currentThread().getName());
    }
   public static void main(String[] args) {
        ThreadGroupDemo runnable = new ThreadGroupDemo();
        ThreadGroup tg1 = new ThreadGroup("Parent ThreadGroup");

        Thread t1 = new Thread(tg1, runnable,"one");
        t1.start();
        Thread t2 = new Thread(tg1, runnable,"two");
        t2.start();
        Thread t3 = new Thread(tg1, runnable,"three");
        t3.start();

        System.out.println("Thread Group Name: "+tg1.getName());
       tg1.list();

    }
}
```

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]
```