JAVA.LANG.THREADGROUP.SETMAXPRIORITY METHOD

 Copyright © tutorialspoint.com

Description

The **java.lang.ThreadGroup.setMaxPriority** method sets the maximum priority of the group.Threads in the thread group that already have a higher priority are not affected.

Declaration

Following is the declaration for **java.lang.ThreadGroup.setMaxPriority** method

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

Parameters

- **pri** -- This is the new priority of the thread group.

Return Value

This method does not return any value.

Exception

- **SecurityException** -- if the current thread cannot modify this thread group.

Example

The following example shows the usage of java.lang.ThreadGroup.setMaxPriority method.

```
package com.tutorialspoint;

import java.lang.*;

public class ThreadGroupDemo implements Runnable
{
   public static void main(String[] args) {
      ThreadGroupDemo tg = new ThreadGroupDemo();
      tg.func();
   }

   public void func() {
      try {
         // create a parent ThreadGroup
         ThreadGroup pGroup = new ThreadGroup("Parent ThreadGroup");
         // set maximum priority as 8 for Parent
         pGroup.setMaxPriority(Thread.MAX_PRIORITY - 2);

         // create a child ThreadGroup for parent ThreadGroup
         ThreadGroup cGroup = new ThreadGroup(pGroup, "Child ThreadGroup");
         // set maximum priority as 5 for child
         cGroup.setMaxPriority(Thread.NORM_PRIORITY);

         // create a thread with Priority as MAX
         Thread t1 = new Thread(pGroup, this);
         t1.setPriority(Thread.MAX_PRIORITY);
         System.out.println("Starting " + t1.getName() + "...");
         t1.start();

         // create another thread with Priority as MAX
         Thread t2 = new Thread(cGroup, this);
         t1.setPriority(Thread.MAX_PRIORITY);
         System.out.println("Starting " + t2.getName() + "...");
         t2.start();
```

```
         // display the number of active threads with actual priority
         System.out.println("Active threads in \"" + pGroup.getName()
         + "\" = " + pGroup.activeCount());

         // block until the other threads finish
         t1.join();
         t2.join();
      }
      catch (InterruptedException ex) {
         System.out.println(ex.toString());
      }
   }

   // implements run()
   public void run() {

      for(int i = 0;i < 1000;i++) {
         i++;
      }
      System.out.println(Thread.currentThread().getName() +
      " [priority = " + Thread.currentThread().getPriority() +
      "] finished executing.");
   }
}
```

Let us compile and run the above program, this will produce the following result:

```
Starting Thread-0...
Starting Thread-1...
Active threads in group "Parent ThreadGroup" = 2
Thread-0 [priority = 8] finished executing.
Thread-1 [priority = 5] finished executing.
```

Loading [MathJax]/jax/output/HTML-CSS/jax.js