JAVA.LANG.THREADGROUP.CHECKACCESS METHOD

Description

The **java.lang.ThreadGroup.checkAccess** method determines if the currently running thread has permission to modify this thread group.

Declaration

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

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

Parameters

- **NA**

Return Value

This method does not return any value.

Exception

- **SecurityException** -- if the current thread is not allowed to access this thread group.

Example

The following example shows the usage of java.lang.ThreadGroup.checkAccess 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");

         // create a child ThreadGroup for parent ThreadGroup
         ThreadGroup cGroup = new ThreadGroup(pGroup, "Child ThreadGroup");

         // create a thread
         Thread t1 = new Thread(pGroup, this);
         System.out.println("Starting " + t1.getName() + "...");
         t1.start();

         // create another thread
         Thread t2 = new Thread(cGroup, this);
         System.out.println("Starting " + t2.getName() + "...");
         t2.start();

         // checking the access for these ThreadGroups.
         try {
            pGroup.checkAccess();
            System.out.println(pGroup.getName() + " has access");
            cGroup.checkAccess();
```

```
            System.out.println(cGroup.getName() + " has access");
         }
            catch (SecurityException ex) {
               System.out.println("No access: " + ex.toString());
            }
         // 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() +
      " finished executing");
   }
}
```

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

```
Starting Thread-0...
Starting Thread-1...
Parent ThreadGroup has access
Child ThreadGroup has access
Thread-0 finished executing
Thread-1 finished executing
```