JAVA - THREAD CONTROL

While the suspend( ), resume( ), and stop( ) methods defined by **Thread** class seem to be a perfectly reasonable and convenient approach to managing the execution of threads, they must not be used for new Java programs and obsolete in newer versions of Java.

The following example illustrates how the wait( ) and notify( ) methods that are inherited from Object can be used to control the execution of a thread.

This example is similar to the program in the previous section. However, the deprecated method calls have been removed. Let us consider the operation of this program.

The NewThread class contains a boolean instance variable named suspendFlag, which is used to control the execution of the thread. It is initialized to false by the constructor.

The run( ) method contains a synchronized statement block that checks suspendFlag. If that variable is true, the wait( ) method is invoked to suspend the execution of the thread. The mysuspend( ) method sets suspendFlag to true. The myresume( ) method sets suspendFlag to false and invokes notify( ) to wake up the thread. Finally, the main( ) method has been modified to invoke the mysuspend( ) and myresume( ) methods.

Example:

```
// Suspending and resuming a thread for Java 2
class NewThread implements Runnable {
   String name; // name of thread
   Thread t;
   boolean suspendFlag;
   NewThread(String threadname) {
      name = threadname;
      t = new Thread(this, name);
      System.out.println("New thread: " + t);
      suspendFlag = false;
      t.start(); // Start the thread
   }
   // This is the entry point for thread.
   public void run() {
      try {
      for(int i = 15; i > 0; i--) {
         System.out.println(name + ": " + i);
         Thread.sleep(200);
         synchronized(this) {
            while(suspendFlag) {
               wait();
            }
          }
        }
      } catch (InterruptedException e) {
         System.out.println(name + " interrupted.");
      }
      System.out.println(name + " exiting.");
   }
   void mysuspend() {
      suspendFlag = true;
   }
   synchronized void myresume() {
      suspendFlag = false;
       notify();
   }
}

public class SuspendResume {
   public static void main(String args[]) {
      NewThread ob1 = new NewThread("One");
      NewThread ob2 = new NewThread("Two");
      try {
         Thread.sleep(1000);
         ob1.mysuspend();
         System.out.println("Suspending thread One");
         Thread.sleep(1000);
         ob1.myresume();
         System.out.println("Resuming thread One");
         ob2.mysuspend();
         System.out.println("Suspending thread Two");
         Thread.sleep(1000);
         ob2.myresume();
         System.out.println("Resuming thread Two");
      } catch (InterruptedException e) {
         System.out.println("Main thread Interrupted");
      }
      // wait for threads to finish
      try {
         System.out.println("Waiting for threads to finish.");
         ob1.t.join();
         ob2.t.join();
      } catch (InterruptedException e) {
         System.out.println("Main thread Interrupted");
      }
      System.out.println("Main thread exiting.");
   }
}
```

Here is the output produced by the above program:

```
New thread: Thread[One,5,main]
One: 15
New thread: Thread[Two,5,main]
Two: 15
One: 14
Two: 14
One: 13
Two: 13
One: 12
Two: 12
One: 11
Two: 11
Suspending thread One
Two: 10
Two: 9
Two: 8
Two: 7
Two: 6
Resuming thread One
Suspending thread Two
One: 10
One: 9
One: 8
One: 7
One: 6
Resuming thread Two
Waiting for threads to finish.
Two: 5
One: 5
Two: 4
One: 4
Two: 3
One: 3
Two: 2
One: 2
Two: 1
One: 1
Two exiting.
One exiting.
Main thread exiting.
```