JAVA - THREAD CONTROL

Core Java provides a complete control over multithreaded program. You can develop a multithreaded program which can be suspended, resumed or stopped completely based on your requirements. There are various static methods which you can use on thread objects to control their behavior. Following table lists down those methods:

SN	Methods with Description
1	**public void suspend()** This method puts a thread in suspended state and can be resumed using resume() method.
2	**public void stop()** This method stops a thread completely.
3	**public void resume()** This method resumes a thread which was suspended using suspend() method.
4	**public void wait()** Causes the current thread to wait until another thread invokes the notify().
5	**public void notify()** Wakes up a single thread that is waiting on this object's monitor.

> *Be aware that latest versions of Java has deprecated the usage of suspend( ), resume( ), and stop( ) methods and so you need to use available alternatives.*

Example:

```
class RunnableDemo implements Runnable {
   public Thread t;
   private String threadName;
   boolean suspended = false;

   RunnableDemo( String name){
       threadName = name;
       System.out.println("Creating " +  threadName );
   }
   public void run() {
      System.out.println("Running " +  threadName );
      try {
         for(int i = 10; i > 0; i--) {
            System.out.println("Thread: " + threadName + ", " + i);
            // Let the thread sleep for a while.
            Thread.sleep(300);
            synchronized(this) {
            while(suspended) {
               wait();
            }
           }
          }
     } catch (InterruptedException e) {
         System.out.println("Thread " +  threadName + " interrupted.");
     }
     System.out.println("Thread " +  threadName + " exiting.");
   }

   public void start ()
   {
      System.out.println("Starting " +  threadName );
      if (t == null)
      {
         t = new Thread (this, threadName);
         t.start ();
      }
   }
   void suspend() {
      suspended = true;
   }
   synchronized void resume() {
      suspended = false;
       notify();
   }
}

public class TestThread {
   public static void main(String args[]) {

      RunnableDemo R1 = new RunnableDemo( "Thread-1");
      R1.start();

      RunnableDemo R2 = new RunnableDemo( "Thread-2");
      R2.start();

      try {
         Thread.sleep(1000);
         R1.suspend();
         System.out.println("Suspending First Thread");
         Thread.sleep(1000);
         R1.resume();
         System.out.println("Resuming First Thread");
         R2.suspend();
         System.out.println("Suspending thread Two");
         Thread.sleep(1000);
         R2.resume();
         System.out.println("Resuming thread Two");
      } catch (InterruptedException e) {
         System.out.println("Main thread Interrupted");
      }
      try {
         System.out.println("Waiting for threads to finish.");
         R1.t.join();
         R2.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:

```
Creating Thread-1
Starting Thread-1
Creating Thread-2
Starting Thread-2
Running Thread-1
Thread: Thread-1, 10
Running Thread-2
Thread: Thread-2, 10
Thread: Thread-1, 9
Thread: Thread-2, 9
Thread: Thread-1, 8
Thread: Thread-2, 8
Thread: Thread-1, 7
```

```
Thread: Thread-2, 7
Suspending First Thread
Thread: Thread-2, 6
Thread: Thread-2, 5
Thread: Thread-2, 4
Resuming First Thread
Suspending thread Two
Thread: Thread-1, 6
Thread: Thread-1, 5
Thread: Thread-1, 4
Thread: Thread-1, 3
Resuming thread Two
Thread: Thread-2, 3
Waiting for threads to finish.
Thread: Thread-1, 2
Thread: Thread-2, 2
Thread: Thread-1, 1
Thread: Thread-2, 1
Thread Thread-1 exiting.
Thread Thread-2 exiting.
Main thread exiting.
```