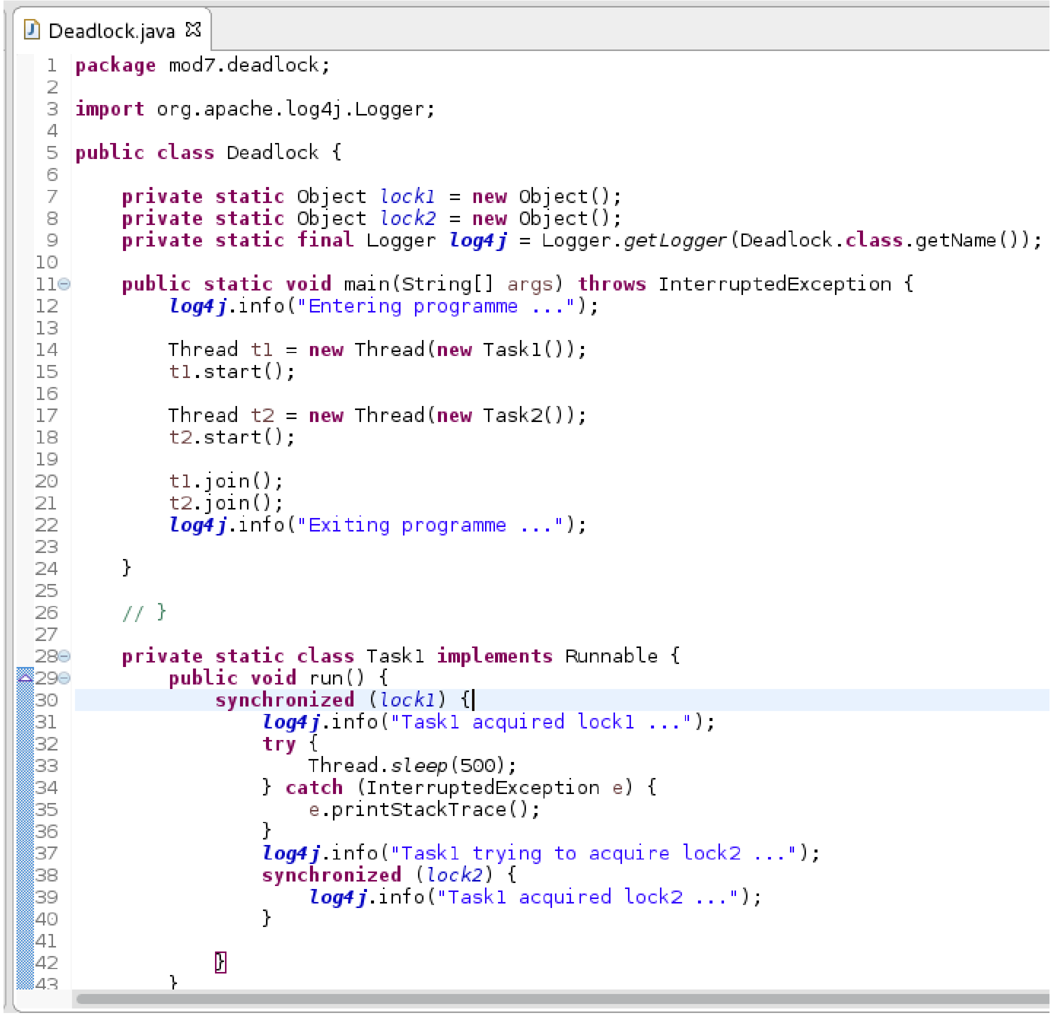
Module 7 - Java Concurrency

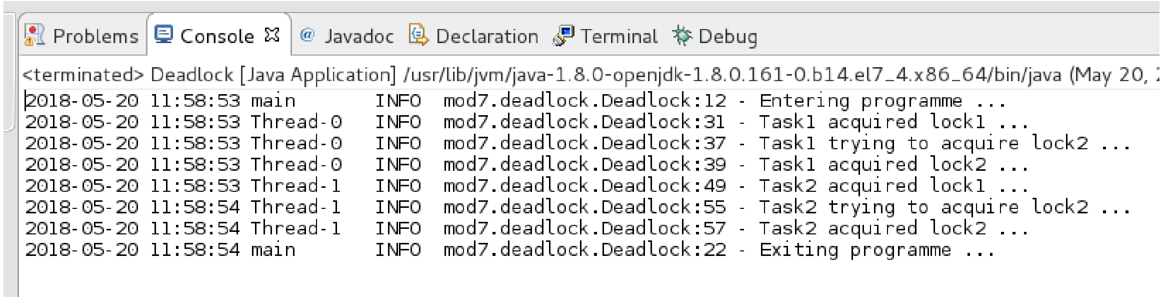
Advanced Java Certification Training

Akram M'Tir

1. For module7.deadlock.Deadlock, modify the code to break the deadlock.

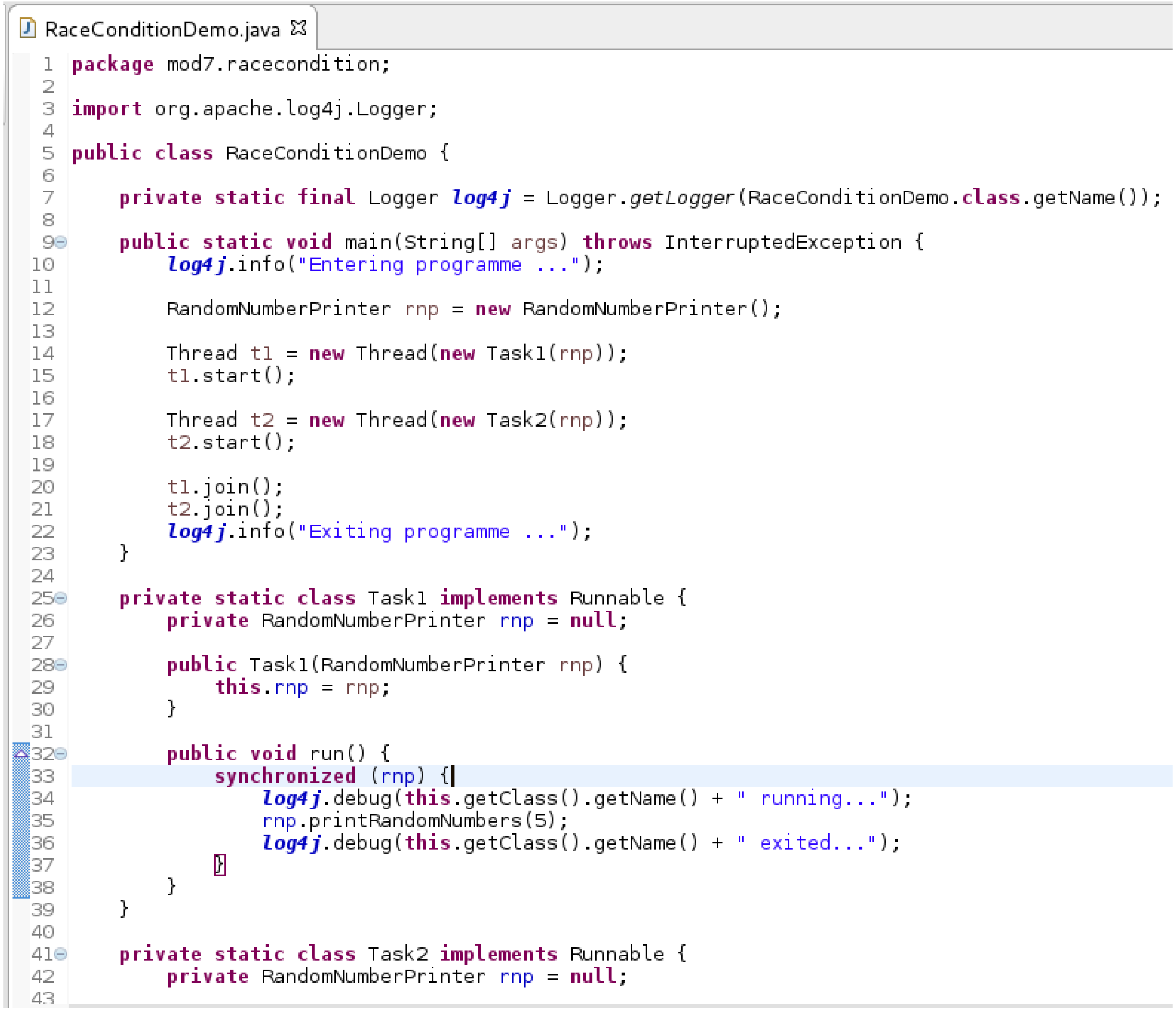


```java
package mod7.deadlock;

import org.apache.log4j.Logger;

public class Deadlock {

    private static Object lock1 = new Object();
    private static Object lock2 = new Object();
    private static final Logger log4j = Logger.getLogger(Deadlock.class.getName());

    public static void main(String[] args) throws InterruptedException {
        log4j.info("Entering programme ...");

        Thread t1 = new Thread(new Task1());
        t1.start();

        Thread t2 = new Thread(new Task2());
        t2.start();

        t1.join();
        t2.join();
        log4j.info("Exiting programme ...");

    }

    // }

    private static class Task1 implements Runnable {
        public void run() {
            synchronized (lock1) {
                log4j.info("Task1 acquired lock1 ...");
                try {
                    Thread.sleep(500);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
                log4j.info("Task1 trying to acquire lock2 ...");
                synchronized (lock2) {
                    log4j.info("Task1 acquired lock2 ...");
                }

            }
        }
```

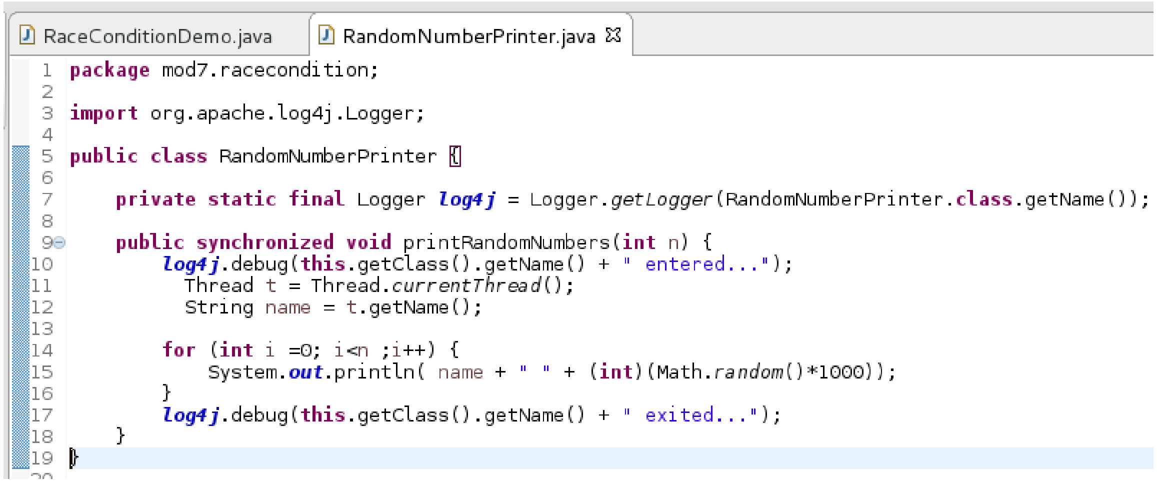


```
<terminated> Deadlock [Java Application] /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.161-0.b14.el7_4.x86_64/bin/java (May 20, 2
2018-05-20 11:58:53 main      INFO  mod7.deadlock.Deadlock:12 - Entering programme ...
2018-05-20 11:58:53 Thread-0  INFO  mod7.deadlock.Deadlock:31 - Task1 acquired lock1 ...
2018-05-20 11:58:53 Thread-0  INFO  mod7.deadlock.Deadlock:37 - Task1 trying to acquire lock2 ...
2018-05-20 11:58:53 Thread-0  INFO  mod7.deadlock.Deadlock:39 - Task1 acquired lock2 ...
2018-05-20 11:58:53 Thread-1  INFO  mod7.deadlock.Deadlock:49 - Task2 acquired lock1 ...
2018-05-20 11:58:54 Thread-1  INFO  mod7.deadlock.Deadlock:55 - Task2 trying to acquire lock2 ...
2018-05-20 11:58:54 Thread-1  INFO  mod7.deadlock.Deadlock:57 - Task2 acquired lock2 ...
2018-05-20 11:58:54 main      INFO  mod7.deadlock.Deadlock:22 - Exiting programme ...
```

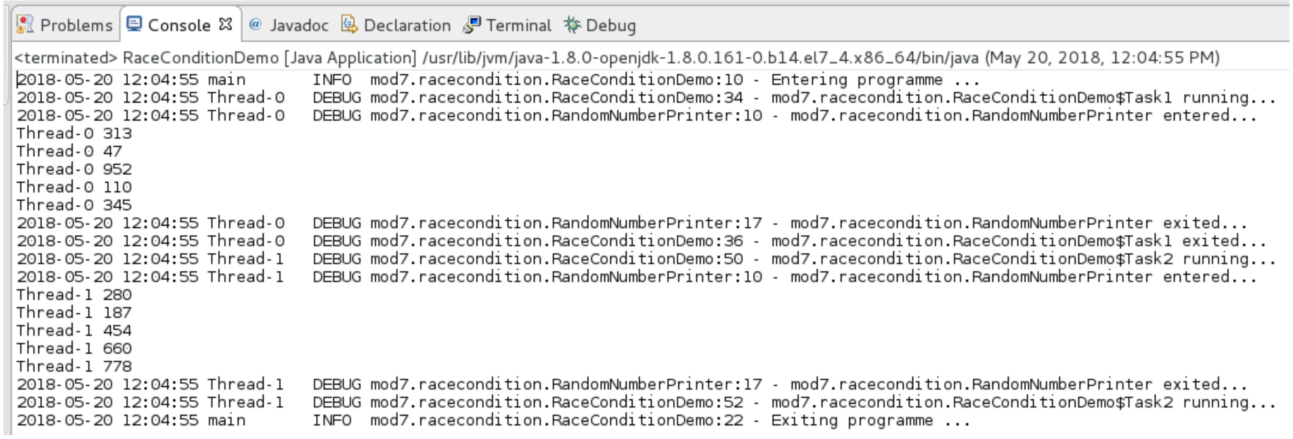
2. For module7.racecondition.RaceConditionDemo - what should I do to print the numbers of the first thread followed by the numbers of the second thread?



```java
package mod7.racecondition;

import org.apache.log4j.Logger;

public class RaceConditionDemo {

    private static final Logger log4j = Logger.getLogger(RaceConditionDemo.class.getName());

    public static void main(String[] args) throws InterruptedException {
        log4j.info("Entering programme ...");

        RandomNumberPrinter rnp = new RandomNumberPrinter();

        Thread t1 = new Thread(new Task1(rnp));
        t1.start();

        Thread t2 = new Thread(new Task2(rnp));
        t2.start();

        t1.join();
        t2.join();
        log4j.info("Exiting programme ...");
    }

    private static class Task1 implements Runnable {
        private RandomNumberPrinter rnp = null;

        public Task1(RandomNumberPrinter rnp) {
            this.rnp = rnp;
        }

        public void run() {
            synchronized (rnp) {
                log4j.debug(this.getClass().getName() + " running...");
                rnp.printRandomNumbers(5);
                log4j.debug(this.getClass().getName() + " exited...");
            }
        }
    }

    private static class Task2 implements Runnable {
        private RandomNumberPrinter rnp = null;
```



```java
package mod7.racecondition;

import org.apache.log4j.Logger;

public class RandomNumberPrinter {

    private static final Logger log4j = Logger.getLogger(RandomNumberPrinter.class.getName());

    public static synchronized void printRandomNumbers(int n) {
        log4j.debug(this.getClass().getName() + " entered...");
          Thread t = Thread.currentThread();
          String name = t.getName();

        for (int i =0; i<n ;i++) {
            System.out.println( name + " " + (int)(Math.random()*1000));
        }
        log4j.debug(this.getClass().getName() + " exited...");
    }
}
```

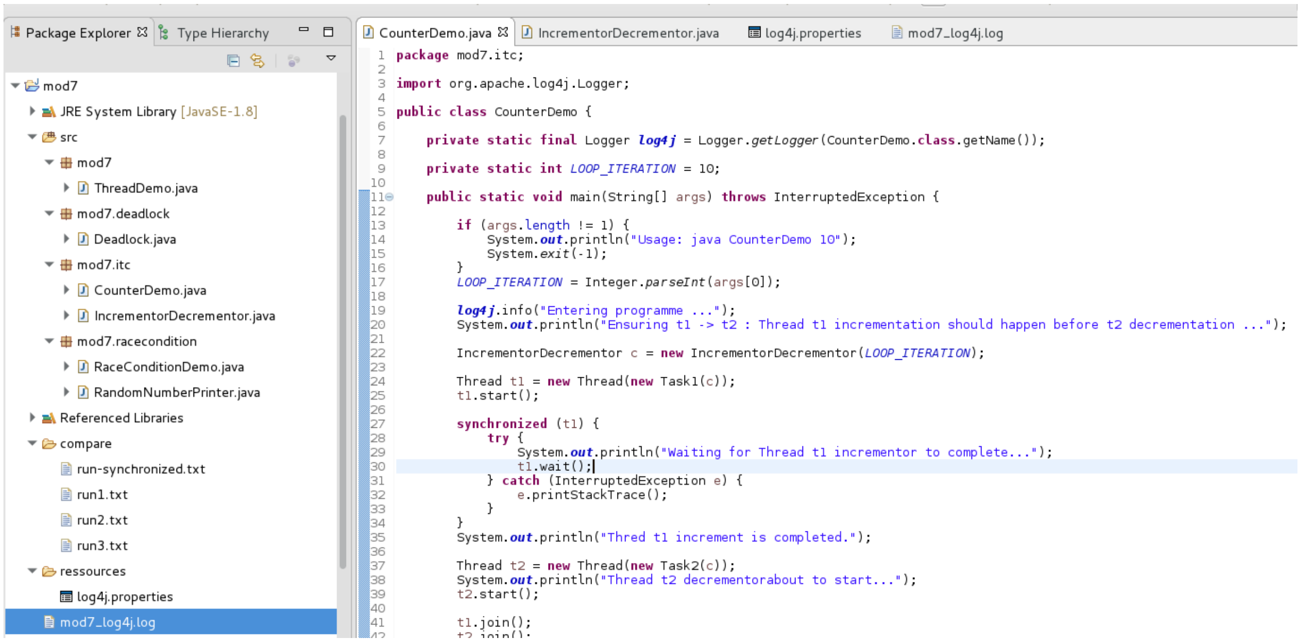


```
<terminated> RaceConditionDemo [Java Application] /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.161-0.b14.el7_4.x86_64/bin/java (May 20, 2018, 12:04:55 PM)
2018-05-20 12:04:55 main       INFO  mod7.racecondition.RaceConditionDemo:10 - Entering programme ...
2018-05-20 12:04:55 Thread-0   DEBUG mod7.racecondition.RaceConditionDemo:34 - mod7.racecondition.RaceConditionDemo$Task1 running...
2018-05-20 12:04:55 Thread-0   DEBUG mod7.racecondition.RandomNumberPrinter:10 - mod7.racecondition.RandomNumberPrinter entered...
Thread-0 313
Thread-0 47
Thread-0 952
Thread-0 110
Thread-0 345
2018-05-20 12:04:55 Thread-0   DEBUG mod7.racecondition.RandomNumberPrinter:17 - mod7.racecondition.RandomNumberPrinter exited...
2018-05-20 12:04:55 Thread-0   DEBUG mod7.racecondition.RaceConditionDemo:36 - mod7.racecondition.RaceConditionDemo$Task1 exited...
2018-05-20 12:04:55 Thread-1   DEBUG mod7.racecondition.RaceConditionDemo:50 - mod7.racecondition.RaceConditionDemo$Task2 running...
2018-05-20 12:04:55 Thread-1   DEBUG mod7.racecondition.RandomNumberPrinter:10 - mod7.racecondition.RandomNumberPrinter entered...
Thread-1 280
Thread-1 187
Thread-1 454
Thread-1 660
Thread-1 778
2018-05-20 12:04:55 Thread-1   DEBUG mod7.racecondition.RandomNumberPrinter:17 - mod7.racecondition.RandomNumberPrinter exited...
2018-05-20 12:04:55 Thread-1   DEBUG mod7.racecondition.RaceConditionDemo:52 - mod7.racecondition.RaceConditionDemo$Task2 running...
2018-05-20 12:04:55 main       INFO  mod7.racecondition.RaceConditionDemo:22 - Exiting programme ...
```

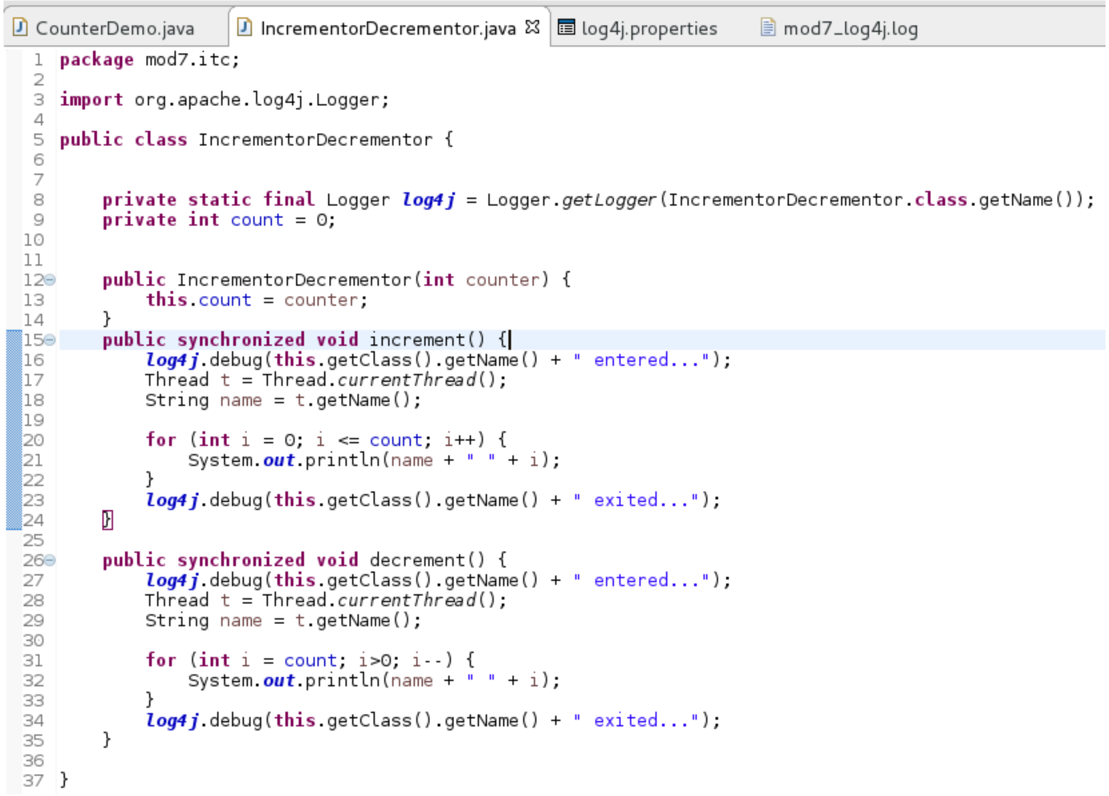
3. Write a programme where one thread writes to a console numbers 1 to N and after it has finished another thread writes the same numbers from N to 1.

- N is passed through the command line.
- The threads can be started in any order.

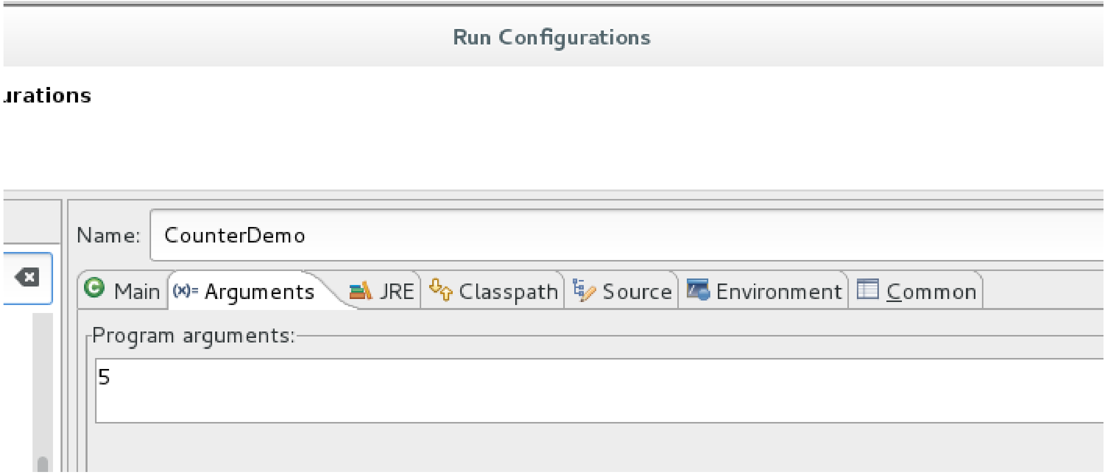
[Hint: Use wait()/notify() methods, inter-thread communication]



```java
package mod7.itc;

import org.apache.log4j.Logger;

public class CounterDemo {

    private static final Logger log4j = Logger.getLogger(CounterDemo.class.getName());

    private static int LOOP_ITERATION = 10;

    public static void main(String[] args) throws InterruptedException {

        if (args.length != 1) {
            System.out.println("Usage: java CounterDemo 10");
            System.exit(-1);
        }
        LOOP_ITERATION = Integer.parseInt(args[0]);

        log4j.info("Entering programme ...");
        System.out.println("Ensuring t1 -> t2 : Thread t1 incrementation should happen before t2 decrementation ...");

        IncrementorDecrementor c = new IncrementorDecrementor(LOOP_ITERATION);

        Thread t1 = new Thread(new Task1(c));
        t1.start();

        synchronized (t1) {
            try {
                System.out.println("Waiting for Thread t1 incrementor to complete...");
                t1.wait();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        System.out.println("Thred t1 increment is completed.");

        Thread t2 = new Thread(new Task2(c));
        System.out.println("Thread t2 decrementorabout to start...");
        t2.start();

        t1.join();
        t2.join();
```



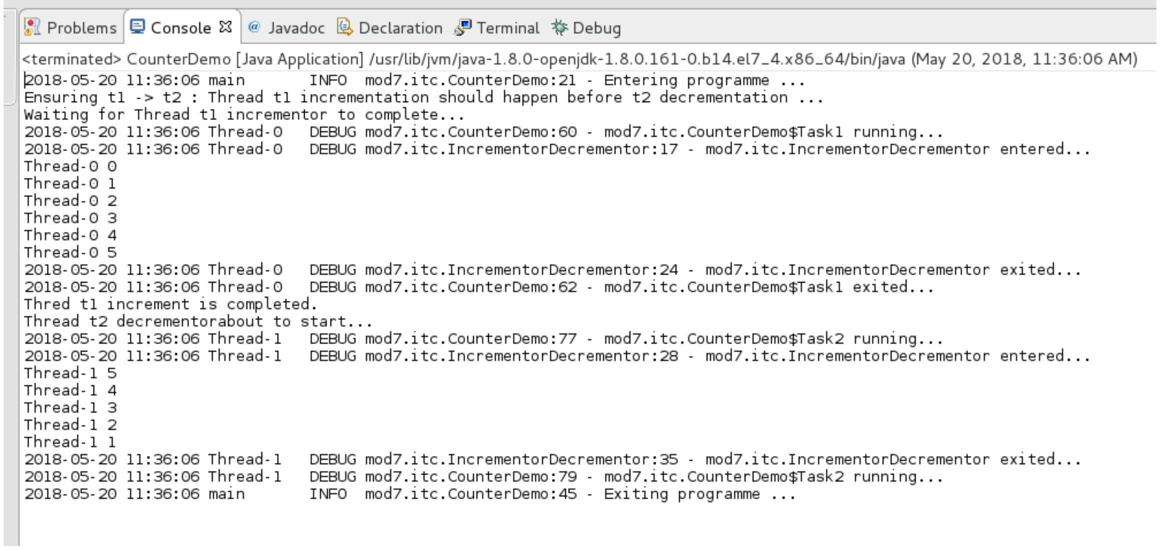
```java
package mod7.itc;

import org.apache.log4j.Logger;

public class IncrementorDecrementor {


    private static final Logger log4j = Logger.getLogger(IncrementorDecrementor.class.getName());
    private int count = 0;


    public IncrementorDecrementor(int counter) {
        this.count = counter;
    }
    public synchronized void increment() {
        log4j.debug(this.getClass().getName() + " entered...");
        Thread t = Thread.currentThread();
        String name = t.getName();

        for (int i = 0; i <= count; i++) {
            System.out.println(name + "  " + i);
        }
        log4j.debug(this.getClass().getName() + " exited...");
    }

    public synchronized void decrement() {
        log4j.debug(this.getClass().getName() + " entered...");
        Thread t = Thread.currentThread();
        String name = t.getName();

        for (int i = count; i>0; i--) {
            System.out.println(name + "  " + i);
        }
        log4j.debug(this.getClass().getName() + " exited...");
    }

}
```



Name: CounterDemo

Program arguments:

```
5
```



<terminated> CounterDemo [Java Application] /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.161-0.b14.el7_4.x86_64/bin/java (May 20, 2018, 11:36:06 AM)

```
2018-05-20 11:36:06 main       INFO  mod7.itc.CounterDemo:21 - Entering programme ...
Ensuring t1 -> t2 : Thread t1 incrementation should happen before t2 decrementation ...
Waiting for Thread t1 incrementor to complete...
2018-05-20 11:36:06 Thread-0   DEBUG mod7.itc.CounterDemo:60 - mod7.itc.CounterDemo$Task1 running...
2018-05-20 11:36:06 Thread-0   DEBUG mod7.itc.IncrementorDecrementor:17 - mod7.itc.IncrementorDecrementor entered...
Thread-0 0
Thread-0 1
Thread-0 2
Thread-0 3
Thread-0 4
Thread-0 5
2018-05-20 11:36:06 Thread-0   DEBUG mod7.itc.IncrementorDecrementor:24 - mod7.itc.IncrementorDecrementor exited...
2018-05-20 11:36:06 Thread-0   DEBUG mod7.itc.CounterDemo:62 - mod7.itc.CounterDemo$Task1 exited...
Thred t1 increment is completed.
Thread t2 decrementorabout to start...
2018-05-20 11:36:06 Thread-1   DEBUG mod7.itc.CounterDemo:77 - mod7.itc.CounterDemo$Task2 running...
2018-05-20 11:36:06 Thread-1   DEBUG mod7.itc.IncrementorDecrementor:28 - mod7.itc.IncrementorDecrementor entered...
Thread-1 5
Thread-1 4
Thread-1 3
Thread-1 2
Thread-1 1
2018-05-20 11:36:06 Thread-1   DEBUG mod7.itc.IncrementorDecrementor:35 - mod7.itc.IncrementorDecrementor exited...
2018-05-20 11:36:06 Thread-1   DEBUG mod7.itc.CounterDemo:79 - mod7.itc.CounterDemo$Task2 running...
2018-05-20 11:36:06 main       INFO  mod7.itc.CounterDemo:45 - Exiting programme ...
```