```
/* Copyright (C) 2006 M. Ben-Ari. See copyright.txt */
```

Semaphore S ← (1,ϕ) **int N ←0**	
P	**q**
int temp **P1: do 10 times** **P2: wait(S)** **P3: temp ←n** **P4: n ← temp +1** **P5: signal(S)**	**int temp** **q1: do 10 times** **q2: wait(S)** **q3: temp ←n** **q4: n ← temp +1** **q5: signal(S)**

```
import java.util.concurrent.Semaphore;

class CountSem extends Thread {

    static volatile int n = 0;

    static Semaphore        s = new Semaphore(1);

    public void run() {

      int temp;

      for (int i = 0; i < 10; i++) {

            try { s.acquire(); } catch (InterruptedException e) {}

        temp = n;

            if (Math.random() < 0.2) Thread.yield();

        n = temp + 1;

            s.release();

      }

    }

    public static void main(String[] args) {

      CountSem p = new CountSem();
      CountSem q = new CountSem();

      p.start();

      q.start();

      try { p.join(); q.join(); }

      catch (InterruptedException e) { }

      System.out.println("The value of n is " + n);

    }

}
```