Atomic Statements:

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

/\* Programmed by Panu Pitkämäki \*/

/\* Critical section problem with test-and-set \*/

class TestSet extends Thread {

 /\* Number of processes currently in critical section \*/

 static volatile int inCS = 0;

 /\* Common value \*/

 static volatile int common = 0;

 /\* Local value \*/

 int local;

 // char pro;

 synchronized void testAndSet() {

 local = common;

 common = 1;

 }

 public void run() {

 int x=0;

 while (x<4) {

 x++;

 /\* Non-critical section \*/

 do

 testAndSet();

 while (local == 1);

 inCS++;

 Thread.yield();

 /\* Critical section \*/

 System.out.println("Number of processes in critical section: "

 + inCS);

 inCS--;

 common = 0;

 }

 }

 public static void main(String[] args) {

 TestSet p = new TestSet();

 TestSet q = new TestSet();

 p.start();

 q.start();

 }

}

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

/\* Programmed by Panu Pitkämäki \*/

/\* Critical section problem with exchange \*/

class Exchange extends Thread {

 /\* Number of processes currently in critical section \*/

 static volatile int inCS = 0;

 /\* Common value \*/

 static volatile int common = 1;

 /\* Local value \*/

 int local = 0;

 synchronized void exchange() {

 int temp;

 temp = common;

 common = local;

 local = temp;

 }

 public void run() {

 while (true) {

 /\* Non-critical section \*/

 do

 exchange();

 while (local == 0);

 inCS++;

 Thread.yield();

 /\* Critical section \*/

 System.out.println("Number of processes in critical section: "

 + inCS);

 inCS--;

 exchange();

 }

 }

 public static void main(String[] args) {

 Exchange p = new Exchange();

 Exchange q = new Exchange();

 p.start();

 q.start();

 }

}