

## Laborator 7

### Problema 1

```
PCMonitor.java
1 package com.home.lab6.task1;
2
3 public class PCMonitor {
4     final int N = 5;
5     int oldest = 0, newest = 0;
6     volatile int count = 0;
7     int buffer[] = new int[N];
8
9     public synchronized void append(int v) {
10         while (count == N) {
11             try {
12                 wait(); // wait not full
13             } catch
14                 (InterruptedException e) {
15             }
16         }
17         buffer[newest] = v;
18         newest = (newest + 1) % N;
19         count++;
20         notifyAll(); // signal not empty
21     }
22
23     synchronized int take() {
24         int temp;
25         while (count == 0) {
26             try {
27                 wait(); // wait not empty
28             } catch
29                 (InterruptedException e) {
30             }
31         }
32         temp = buffer[oldest];
33         oldest = (oldest + 1) % N;
34         count--;
35         notifyAll(); // signal not full
36         return temp;
37     }
38 }
```

```
Consumer.java
1 package com.home.lab6.task1;
2
3 public class Consumer extends Thread {
4     int q, r, max, min;
5     PCMonitor buffer;
6     int c = 0; // Contor de elemente consumate
7
8     public Consumer(int q, int r, int min, int max, PCMonitor b) {
9         this.q = q;
10        this.r = r;
11        this.min = min;
12        this.max = max;
13        this.buffer = b;
14    }
15
16    int getCounter() {
17        return c;
18    }
19
20    public void run() {
21        int i = min, k;
22        while ((i % q) != r) {
23            i++;
24        }
25        while (i <= max) {
26            k = buffer.take();
27            c++;
28            System.out.println("Consumer " + r + " consumed " + k);
29            i += q;
30        }
31    }
32 }
```

```
1 package com.home.lab6.task1;
2
3
4 public class Producer extends Thread {
5     int q, r, max, min; // min...max, din q in q, incepand cu restul r
6     PCMonitor buffer;
7     int c = 0; // Contor de elemente trimise
8
9     public Producer(int q, int r, int min, int max, PCMonitor b) {
10         this.q = q;
11         this.r = r;
12         this.min = min;
13         this.max = max;
14         this.buffer = b;
15     }
16
17     int getCounter() {
18         return c;
19     }
20
21     public void run() {
22         int i = min;
23         while ((i % q) != r) {
24             i++;
25         } // determina primul numar produs
26         while (i <= max) {
27             buffer.append(i);
28             c++;
29             System.out.println("Producer " + r + " produced " + i);
30             i += q;
31         }
32     }
33 }
34
```

```
1 package com.home.lab6.task1;
2 public class Main {
3     static final int NPTHREADS = 10;
4     static final int NCTHREADS = 5;
5     static final int MINN = 0;
6     static final int MAXN = 76;
7     static Producer pThreads[] = new Producer[NPTHREADS];
8     static Consumer cThreads[] = new Consumer[NCTHREADS];
9     static PCMonitor buffer = new PCMonitor();
10    public static void main(String[] args) {
11        int i;
12        System.out.println("Prodicator - Consumator started");
13        for (i = 0; i < NPTHREADS; i++) {
14            pThreads[i] = new Producer(NPTHREADS, i, MINN, MAXN, buffer);
15            pThreads[i].start();
16        }
17        for (i = 0; i < NCTHREADS; i++) {
18            cThreads[i] = new Consumer(NCTHREADS, i, MINN, MAXN, buffer);
19            cThreads[i].start();
20        }
21        for (i = 0; i < NPTHREADS; i++) {
22            try {pThreads[i].join();}
23            catch (InterruptedException e) {}
24        }
25        for (i = 0; i < NCTHREADS; i++) {
26            try {cThreads[i].join();}
27            catch (InterruptedException e) {}
28        }
29        for (i = 0; i < NPTHREADS; i++) {
30            System.out.println("Prodicator " + i + " produced " + pThreads[i].getCounter() + " elements.");
31        }
32        for (i = 0; i < NCTHREADS; i++) {
33            System.out.println("Consumator " + i + " consumed " + cThreads[i].getCounter() + " elements.");
34        }
35        System.out.println("Prodicator - Consumator finished");
36    }
37 }
38
39
```

### Output:

```
C:\Users\user\.jdk\corretto-16.
Producer - Consumer started
Producer 5 produced 5
Producer 4 produced 4
Producer 0 produced 0
Producer 8 produced 8
Producer 7 produced 7
Producer 2 produced 2
Producer 6 produced 6
Producer 1 produced 1
Consumer 4 consumed 3
Consumer 4 consumed 5
Consumer 4 consumed 8
Consumer 4 consumed 7
Consumer 2 consumed 4
Producer 9 produced 9
Consumer 3 consumed 2
Producer 0 produced 10
Consumer 2 consumed 9
Producer 8 produced 18
Consumer 4 consumed 6
Producer 1 produced 11
Producer 6 produced 16
Producer 5 produced 15
Producer 3 produced 3
Consumer 1 consumed 1
Producer 0 produced 20
Consumer 4 consumed 16
Producer 4 produced 14
Consumer 2 consumed 11
Producer 2 produced 12
Consumer 3 consumed 15
Producer 9 produced 19
Consumer 2 consumed 12
Producer 3 produced 13
Producer 0 produced 30
Consumer 0 consumed 0
Consumer 4 consumed 10
Producer 8 produced 28
Consumer 1 consumed 18
Producer 0 produced 40
Consumer 4 consumed 13
Producer 2 produced 22
Consumer 0 consumed 28
Consumer 2 consumed 20
Producer 4 produced 24
Producer 8 produced 38
Consumer 3 consumed 14
Consumer 2 consumed 22
Producer 7 produced 17
Consumer 0 consumed 30
Producer 3 produced 23
Producer 0 produced 50
Consumer 4 consumed 24
Producer 5 produced 25
Consumer 1 consumed 19
Producer 8 produced 48
Producer 0 produced 60
Consumer 4 consumed 17
Consumer 0 consumed 23
Consumer 2 consumed 25
Producer 6 produced 26
Producer 1 produced 21
Consumer 3 consumed 40
Producer 3 produced 33
Consumer 2 consumed 50
Producer 5 produced 35
Consumer 0 consumed 26
Producer 7 produced 27
Consumer 4 consumed 21
Consumer 1 consumed 38
Producer 6 produced 36
Consumer 4 consumed 35
```

```
Consumer 2 consumed 12
Producer 3 produced 13
Producer 0 produced 30
Consumer 0 consumed 0
Consumer 4 consumed 10
Producer 8 produced 28
Consumer 1 consumed 18
Producer 0 produced 40
Consumer 4 consumed 13
Producer 2 produced 22
Consumer 0 consumed 28
Consumer 2 consumed 20
Producer 4 produced 24
Producer 8 produced 38
Consumer 3 consumed 14
Consumer 2 consumed 22
Producer 7 produced 17
Consumer 0 consumed 30
Producer 3 produced 23
Producer 0 produced 50
Consumer 4 consumed 24
Producer 5 produced 25
Consumer 1 consumed 19
Producer 8 produced 48
Producer 0 produced 60
Consumer 4 consumed 17
Consumer 0 consumed 23
Consumer 2 consumed 25
Producer 6 produced 26
Producer 1 produced 21
Consumer 3 consumed 40
Producer 3 produced 33
Consumer 2 consumed 50
Producer 5 produced 35
Consumer 0 consumed 26
Producer 7 produced 27
Consumer 4 consumed 21
Consumer 1 consumed 38
Producer 6 produced 36
Consumer 4 consumed 35
```

```
Run: Main (5)
Producer 4 produced 64
Producer 1 produced 71
Consumer 3 consumed 67
Producer 2 produced 52
Consumer 2 consumed 59
Producer 6 produced 76
Consumer 3 consumed 71
Producer 5 produced 75
Producer 4 produced 74
Consumer 0 consumed 52
Consumer 1 consumed 61
Consumer 3 consumed 64
Producer 2 produced 62
Consumer 1 consumed 73
Producer 2 produced 72
Consumer 1 consumed 69
Consumer 1 consumed 75
Consumer 1 consumed 76
Consumer 1 consumed 74
Consumer 1 consumed 62
Consumer 1 consumed 72
Producer 0 produced 8 elements.
Producer 1 produced 8 elements.
Producer 2 produced 8 elements.
Producer 3 produced 8 elements.
Producer 4 produced 8 elements.
Producer 5 produced 8 elements.
Producer 6 produced 8 elements.
Producer 7 produced 7 elements.
Producer 8 produced 7 elements.
Producer 9 produced 7 elements.
Consumer 0 consumed 16 elements.
Consumer 1 consumed 16 elements.
Consumer 2 consumed 15 elements.
Consumer 3 consumed 15 elements.
Consumer 4 consumed 15 elements.
Producer - Consumer finished
Process finished with exit code 0
```

### Observatii:

Monitoarele folosesc variabile de stare si un blocaj pentru sincronizare. Un monitor poate fi definit printr-un tip de date abstracte cu o semnantica de sincronizare specifica. Sincronizarea intre firele de executie prodactor si consumator este gestionata de variabilele de conditie.

## Problema 2

```
home > lab6 > task2 > Philosopher
Monitor.java x Global.java x STATE.java x Philosopher.java x Main.java x
1 package com.home.lab6.task2;
2
3 public class Philosopher extends Thread {
4     public Monitor monitor;
5     public int ID;
6
7     public Philosopher(Monitor monitor, int ID) {
8         this.monitor = monitor;
9         this.ID = ID;
10    }
11
12    public void run() {
13        do {
14            monitor.go(ID);
15        } while (true);
16    }
17
18 }
```

```
home > lab6 > task2 > STATE
Monitor.java x Global.java x STATE.java x Philosopher.java x Main.java x
1 package com.home.lab6.task2;
2
3 public interface STATE {
4     int HUNGRY=0, THINKING=1, EATING=2;
5 }
```

```
home > lab6 > task2 > Global
Monitor.java x Global.java x STATE.java x Philosopher.java x Main.java x
1 package com.home.lab6.task2;
2
3 public class Global {
4     private static int num;
5     static{
6         setNum(5);
7     }
8
9     public static int Left(int i){
10         final int idx = (i - 1 + getNum()) % getNum();
11         return idx;
12     }
13     public static int Right(int i){
14         final int idx = (i + 1) % getNum();
15         return idx;
16     }
17
18     public static int getNum() {
19         return num;
20     }
21
22     public static void setNum(int num) {
23         Global.num = num;
24     }
25
26 }
```

```
home > lab6 > task2 > Monitor
Monitor.java Global.java STATE.java Philosopher.java Main.java
1 package com.home.lab6.task2;
2
3 import java.util.concurrent.locks.Condition;
4 import java.util.concurrent.locks.ReentrantLock;
5 import java.util.stream.IntStream;
6
7 public class Monitor {
8
9     private final ReentrantLock entLock;
10    private final Condition self[];
11    private int state[];
12
13    public Monitor(int num) {
14        entLock = new ReentrantLock();
15        self = new Condition[num];
16        state = new int[num];
17
18        int i = 0;
19        if (i < num) {
20            do {
21                self[i] = entLock.newCondition();
22                state[i] = STATE.THINKING;
23                i++;
24            } while (i < num);
25        }
26    }
27
28    void go(int who) {
29        try {
30            pickup(who);
31            putdown(who);
32        } catch (InterruptedException e) {
33            e.printStackTrace();
34        }
35    }
36
37    void pickup(int who) throws InterruptedException {
38        entLock.lock();
39        state[who] = STATE.HUNGRY;
40        System.out.println("Philosopher " + who + " is hungry.\n");
```

```
home > lab6 > task2 > Monitor
Monitor.java Global.java STATE.java Philosopher.java Main.java
34    }
35    }
36
37    void pickup(int who) throws InterruptedException {
38        entLock.lock();
39        state[who] = STATE.HUNGRY;
40        System.out.println("Philosopher " + who + " is hungry.\n");
41        Test(who);
42
43        switch (state[who]) {
44            case STATE.EATING:
45                break;
46            default:
47                self[who].await();
48                break;
49        }
50        entLock.unlock();
51    }
52
53    void Test(int who) {
54        if (state[Global.Left(who)] == STATE.EATING || state[Global.Right(who)] == STATE.EATING || state[who] != STATE.HUNGRY) {
55            return;
56        } else {
57            state[who] = STATE.EATING;
58            System.out.println("Philosopher " + who + " is eating.\n");
59            self[who].signal();
60        }
61    }
62
63    void putdown(int who) {
64        entLock.lock();
65        state[who] = STATE.THINKING;
66        System.out.println("Philosopher " + who + " is thinking.\n");
67        IntStream.of(Global.Left(who), Global.Right(who)).forEach(this::Test);
68        entLock.unlock();
69    }
70    }
71}
```

```
home > lab6 > task2 > Main > main
Monitor.java x Global.java x STATE.java x Philosopher.java x Main.java x
1 package com.home.lab6.task2;
2
3 import java.util.stream.IntStream;
4
5 public class Main {
6     public static void main(String args[]) throws InterruptedException {
7         Monitor monitor = new Monitor(Global.num);
8         Philosopher P[] = IntStream.range(0, Global.num).mapToObj(i -> new Philosopher(monitor, i)).toArray(Philosopher[]::new);
9         IntStream.range(0, Global.num).forEachOrdered(i -> {
10             P[i].start();
11         });
12         int i = Global.num - 1;
13         if (i >= 0) {
14             do {
15                 P[i].join();
16                 i--;
17             } while (i >= 0);
18         }
19     }
20 }
```

### Output:

```
Concurrent-and-Distributed-Systems > Laboratoare > src > com > home > lab6 > task2 >
Project
Run: Main (6) x
Philosopher 2 is thinking.
Philosopher 1 is eating.
Philosopher 2 is hungry.
Philosopher 4 is thinking.
Philosopher 3 is eating.
Philosopher 4 is hungry.
Philosopher 1 is thinking.
Philosopher 0 is eating.
Philosopher 1 is hungry.
Philosopher 3 is thinking.
Philosopher 2 is eating.
Philosopher 3 is hungry.
Philosopher 0 is thinking.
Philosopher 4 is eating.
Philosopher 0 is hungry.
Philosopher 2 is thinking.
Philosopher 1 is eating.
Philosopher 2 is hungry.

Process finished with exit code 130
```