Ministerul Educaţiei, al Culturii și Cercetării al Republicii Moldova

Universitatea Tehnică a Moldovei

Departamentul Informatică și Ingineria Sistemelor

**RAPORT**

Lucrarea de laborator nr.5

ASO

A efectuat:

st. gr. C-171 D. Melniciuc

A verificat:

dr., conf.univ. L.Rotaru

Chişinău 2020

Varianta:4

Scrieţi un program pentru X (unde X este egal cu numărul variantei plus numărul subgrupei) scriitori

şi Y (unde Y este egal cu numărul variantei înmulţit cu 2 plus numărul subgrupei) cititori,

fiecare scriitor scrie cîte Z (unde Z este egal cu numărul variantei plus 3) obiecte.

X (unde X este egal cu numărul variantei plus numărul subgrupei)

Y (unde Y este egal cu numărul variantei înmulţit cu 2 plus numărul subgrupei)

Z (unde Z este egal cu numărul variantei plus 3) obiecte.

X scriitori 5

Y cititori 9

Z carti 7

Listingul programului:

import java.util.concurrent.Semaphore;

import java.util.concurrent.ThreadLocalRandom;

public class DiningPhilosophers {

static int philosophersNumber = 8;

static Philosopher philosophers[] = new Philosopher[philosophersNumber];

static Fork forks[] = new Fork[philosophersNumber];

static class Fork {

public Semaphore mutex = new Semaphore(1);

void grab() {

try {

mutex.acquire();

}

catch (Exception e) {

e.printStackTrace(System.out);

}

}

void release() {

mutex.release();

}

boolean isFree() {

return mutex.availablePermits() > 0;

}

}

static class Philosopher extends Thread {

public int number;

public Fork leftFork;

public Fork rightFork;

Philosopher(int num, Fork left, Fork right) {

number = num;

leftFork = left;

rightFork = right;

}

public void run(){

System.out.println("philosopher #" + number + " is meditating");

while (true) {

leftFork.grab();

System.out.println("philosopher #" + number + " grabs left chopstick.");

rightFork.grab();

System.out.println("philosopher #" + number + " grabs right chopstick.");

eat();

leftFork.release();

System.out.println("philosopher #" + number + " releases left chopstick.");

rightFork.release();

System.out.println("philosopher #" + number + " releases right chopstick.");

}

}

void eat() {

try {

int sleepTime = ThreadLocalRandom.current().nextInt(0, 1000);

System.out.println("\tphilosopher #" + number + " eats for " + sleepTime);

Thread.sleep(sleepTime);

}

catch (Exception e) {

e.printStackTrace(System.out);

}

}

}

public static void main(String argv[]) {

try { new ProcessBuilder("cmd", "/c", "cls").inheritIO().start().waitFor(); }

catch (Exception e1) { System.out.println(e1); }

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

forks[i] = new Fork();

}

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

philosophers[i] = new Philosopher(i, forks[i], forks[(i + 1) % philosophersNumber]);

philosophers[i].start();

}

while (true) {

try {

// sleep 1 sec

Thread.sleep(1000);

// check for deadlock

boolean deadlock = true;

for (Fork f : forks) {

if (f.isFree()) {

deadlock = false;

break;

}

}

if (deadlock) {

Thread.sleep(1000);

System.out.println("There is a deadlock!");

break;

}

}

catch (Exception e) {

e.printStackTrace(System.out);

}

}

System.exit(0);

}

}

Output:

philosopher #1 is meditating

philosopher #7 is meditating

philosopher #5 is meditating

philosopher #6 is meditating

philosopher #1 grabs left chopstick.

philosopher #7 grabs left chopstick.

philosopher #0 is meditating

philosopher #1 grabs right chopstick.

philosopher #3 is meditating

philosopher #3 grabs left chopstick.

philosopher #3 grabs right chopstick.

philosopher #2 is meditating

philosopher #4 is meditating

philosopher #7 grabs right chopstick.

philosopher #6 grabs left chopstick.

philosopher #5 grabs left chopstick.

philosopher #3 eats for 880

philosopher #7 eats for 85

philosopher #1 eats for 56

philosopher #1 releases left chopstick.

philosopher #2 grabs left chopstick.

philosopher #1 releases right chopstick.

philosopher #1 grabs left chopstick.

philosopher #7 releases left chopstick.

philosopher #6 grabs right chopstick.

philosopher #0 grabs left chopstick.

philosopher #7 releases right chopstick.

philosopher #6 eats for 648

philosopher #6 releases left chopstick.

philosopher #5 grabs right chopstick.

philosopher #7 grabs left chopstick.

philosopher #6 releases right chopstick.

philosopher #5 eats for 963

philosopher #3 releases left chopstick.

philosopher #2 grabs right chopstick.

philosopher #3 releases right chopstick.

philosopher #4 grabs left chopstick.

philosopher #2 eats for 793

philosopher #2 releases left chopstick.

philosopher #2 releases right chopstick.

philosopher #1 grabs right chopstick.

philosopher #1 eats for 567

philosopher #3 grabs left chopstick.

philosopher #5 releases left chopstick.

philosopher #5 releases right chopstick.

philosopher #4 grabs right chopstick.

philosopher #4 eats for 663

philosopher #6 grabs left chopstick.

Hurray! There is a deadlock!

C:\code\universitate\aso>

C:\code\universitate\aso>