Catedra Calculatoare

Raport

PCD

Lucrarea de laborator nr.4

**Tema: Sincronizarea thread-urilor în Java.**

Varianta 7

A îndeplinit:st.gr. C-161 Marjina Alexandru

A controlat: lector superior Rotaru Lilia

**Chișinău-2018**

Probleme propuse:

X producători generează aleatoriu F obiecte care sunt consumate de Y consumatori. De afişat informaţia despre producerea şi consumarea obiectelor, mesajele despre cazurile când “depozitul e gol sau plin”. Toate operaţiile se efectuează până când fiecare consumator este îndestulat cu Z obiecte.

Dimensiunea depozitului este D. Valorile se iau din tabelul 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nr. | X | Y | Z | D | Tip Obiecte |
| 7 | 3 | 3 | 5 | 6 | Vocale |
| \* fiecare producător poate produce câte 2 obiecte de fiecare dată | | | | | |

**Listingul Programului:**

public class Main {  
  
 public static void main(String[] args) throws InterruptedException {  
 Depozit d=new Depozit(6);  
 Producator p1=new Producator("P1",d);  
 Consumator c1=new Consumator("C1",d,5);  
 Producator p2=new Producator("P2",d);  
 Consumator c2=new Consumator("C2",d,5);  
 Producator p3=new Producator("P3",d);  
 Consumator c3=new Consumator("C3",d,5);  
 p1.start();  
 p2.start();  
 p3.start();  
 c1.start();  
 c2.start();  
 c3.start();  
 try {  
 Thread.*sleep*(2000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 if(!c1.isAlive()&&!c2.isAlive()&&!c3.isAlive()){  
// p1.valabil=true;  
// p2.valabil=true;  
// p3.valabil=true;  
 p1.stop();  
 p2.stop();  
 p3.stop();  
 }  
 System.*out*.println(c1.isAlive());  
 System.*out*.println(c2.isAlive());  
 System.*out*.println(c3.isAlive());  
 System.*out*.println(p1.isAlive());  
 System.*out*.println(p2.isAlive());  
 System.*out*.println(p3.isAlive());  
 c1.ConsumAfisat();  
 c2.ConsumAfisat();  
 c3.ConsumAfisat();  
 }  
}

import java.util.\*;  
public class Depozit {  
 public int size;  
 public Queue<Character> depozit = new LinkedList<>();  
  
 public Depozit(int size){  
 this.size=size;  
 }  
 public synchronized void Stocheaza(char p1,char p2) throws InterruptedException {  
 if(Liber()){  
 System.*out*.println("In depozit adaugat- "+p1+ " spatiu liber ramas- "+(this.size-depozit.size()));  
 depozit.add(p1);  
 if(Liber()){  
 System.*out*.println("In depozit adaugat- "+p2+ " spatiu liber ramas- "+(this.size-depozit.size()));  
 depozit.add(p2);  
 }  
 else{  
 System.*out*.println("Depozitul nu are spatiu "+p2+" nu se aduaga in depozit");  
 }  
 notify();  
 }  
 else{  
 while (!Liber()){  
 System.*out*.println("Depozitul este plin producatorii astepata");  
 wait();  
 }  
  
 }  
  
 }  
 public synchronized Character Consuma() throws InterruptedException {  
 if(!depozit.isEmpty()){  
 Character temp=depozit.remove();  
 System.*out*.println("A fost scos din depozit- "+temp);  
 notify();  
 return temp;  
 }  
 else{  
 while (depozit.isEmpty())  
 {  
 System.*out*.println("Depozitul e gol Consumatorii astepata");  
 wait();  
 }  
 return 0;  
 }  
 }  
  
 public boolean Liber(){  
 if(depozit.size()<=this.size)  
 return true;  
 else  
 return false;  
 }  
  
  
}

public class Producator extends Thread {  
 private char[] vowel = { 'a', 'e', 'i', 'o', 'u' };  
 private Random random = new Random();  
 private String Nume="";  
 private Depozit d;  
 public boolean valabil=false;  
 private int count=0;  
 private int capacity=4;  
  
 public Producator(String Nume,Depozit d){  
 this.Nume=Nume;  
 this.d=d;  
 }  
  
 public void run(){  
 int rN1,rN2;  
 while (count<=capacity) {  
 rN1=random.nextInt(vowel.length)+0;  
 rN2=random.nextInt(vowel.length)+0;  
 System.*out*.println(this.Nume+" a produs="+vowel[rN1]+" si "+vowel[rN2]);  
// System.out.println(this.Nume+" a produs="+vowel[rN2]);  
 try {  
 d.Stocheaza(vowel[rN1],vowel[rN2]);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 //if(valabil) this.stop();  
 count++;  
 }  
  
 }  
  
  
}

import java.util.LinkedList;  
import java.util.Queue;  
  
public class Consumator extends Thread {  
 private String Nume="";  
 public Queue<Character> consumari = new LinkedList<>();  
 private int size;  
 private Depozit d;  
  
  
 public Consumator(String Nume,Depozit d,int consum){  
 this.Nume=Nume;  
 this.d=d;  
 this.size=consum;  
 }  
 public void run(){  
 while (!Indestulat()) {  
 try {  
 Character c = d.Consuma();  
 consumari.add(c);  
 System.*out*.println(this.Nume + " a consumat " + c);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 } if(Indestulat()){  
 System.*out*.println(this.Nume+" Este indestulat");  
 // this.stop();  
 }  
  
 }  
 public Boolean Indestulat(){  
 if(consumari.size()==this.size){  
 return true;  
 }  
 else  
 return false;  
 }  
 public void ConsumAfisat(){  
 System.*out*.print(this.Nume+" a consumat in final= ");  
 while (!consumari.isEmpty()){  
 System.*out*.print(consumari.remove());  
 }  
 System.*out*.println();  
 }  
}