**МІНІСТЕРСТВО ОСВІТИ І НАУКИ**

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАІНИ**

**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»**

**ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ**

**КАФЕДРА СИСТЕМНОГО ПРОГРАМУВАННЯ І СПЕЦІАЛІЗОВАНИХ КОМП`ЮТЕНИХ СИСТЕМ**

**ЛАБОРАТОРНАЯ РОБОТА №5**

з дисципліни «**Паралельні та розподілені обчислення**»

Тема: «**Засоби взаємодії паралельних потоків мови Java**»

|  |  |
| --- | --- |
|  | Виконав  студент 2 курсу  гр. КВ-43  Вітовщик Михайло Васильович  Перевірив(ла):  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Київ – 2016

***Постановка задачі***

1.Написати програму на мові Java, яка реалізує роботу паралельних потоків згідно

заданої за варіантом схеми. Особливості реалізації синхронізації паралельних потоків та взаємного виключення потоків при доступі до спільних ресурсів задані за варіантами у таблиці завдань.

2.При написанні програми виконати повне трасування роботи програми за допомогою операторів друку, тобто розставити в програмі оператори друку таким чином, щоб можна було прослідкувати всі варіанти виконання паралельних потоків і впевнитись у коректності роботи програми. Протокол трасування рекомендується записувати у файл (log-файл).

3.Запуск усіх потоків повинен бути виконаний у головній програмі.

4.Кожен потік повинен бути організованим у вигляді нескінченного циклу.

5.Всі дії задані за варіантами, що вказані у таблиці, повинні бути виконані всередині цього нескінченного циклу.

6.Взаємне розташування операторів синхронізації та доступу до спільного ресурсу,

якщо вони знаходяться у одному потоці, є довільним.

7.Оскільки синхронізація за допомогою семафорів Sem1, Sem2 згідно завдання розташована всередині нескінченних циклів, то відразу після виконання синхронізації ці семафори повинні бути знову встановлені у початковий закритий стан.

8.Закінчення програми можна виконати двома способами:

• примусовим перериванням за допомогою натиснення комбінації клавіш Ctrl+C;

• оператором виходу з циклу break при виконанні умови, яка стає істинною, коли буфер спільного ресурсу повністю заповнюється і повністю звільняється мінімумпо два рази.

9.Якщо при реалізації паралельних потоків була використаний метод sleep(), то передбачити режим запуску програми з «відключеними» викликами sleep().

10. Виконати налагодження написаної програми.

***3 варіант***

***Текст програми***

import java.util.concurrent.BrokenBarrierException;

import java.util.concurrent.CyclicBarrier;

import java.util.concurrent.Semaphore;

import java.util.concurrent.locks.ReentrantLock;

class CR1{

public static final int MaxBufSize = 19;

public static final int MinBufSize = 0;

int buf[] = new int[MaxBufSize+1];

int count = 0;

int ind = 0;

int next\_c = 0, next\_p = 0;

boolean IsEmpty = ind == MinBufSize;

boolean IsFull = ind == MaxBufSize;

synchronized void consumer(String name){

while (IsEmpty)

try

{

wait();

}

catch (InterruptedException e)

{

System.out.println("InterruptedException");

}

System.out.println(name + "--- buf[" + next\_c + "] = " + buf[next\_c] + " \tDELETED");

buf[next\_c] = 0;

ind--;

next\_c = (next\_c+1)%(MaxBufSize+1);

IsEmpty = ind == MinBufSize;

IsFull = false;

notify();

}

synchronized void producer(String name){

while (IsFull)

try

{

wait();

}

catch (InterruptedException e)

{

System.out.println("InterruptedException");

}

System.out.println(name + "+++ ind: " + next\_p +" => " + count + " \tCREATED");

buf[next\_p] = count;

ind++;

count++;

next\_p = (next\_p+1)%(MaxBufSize+1);

IsFull = (ind-1) == MaxBufSize;

IsEmpty = false;

notify();

}

}

class CR2{

int a;

boolean b;

char c;

double d;

CR2(int \_a, boolean \_b, char \_c, double \_d){

\_a = a;

\_b = b;

\_c = c;

\_d = d;

}

}

class SS{

public static Semaphore sem1 = new Semaphore(0, true);

public static Semaphore sem2 = new Semaphore(0, true);

}

class CB{

public static CyclicBarrier cb1 = new CyclicBarrier(2);

public static CyclicBarrier cb2 = new CyclicBarrier(2);

}

class Producer implements Runnable{

Thread t;

private int id;

CR1 cr1;

CR2 cr2;

CB cb;

SS sem;

Producer (CR1 cr1, int id, SS sem, CR2 cr2, CB cb){

this.cr1 = cr1;

this.id = id;

this.sem = sem;

this.cr2 = cr2;

this.cb = cb;

t = new Thread (this, "Producer " + id);

t.start();

}

public void run()

{

while (true)

{

if (this.id == 5){

sem.sem2.release();

try {

sem.sem1.acquire();

}catch (InterruptedException e){

System.out.println( t.getName() + "interrupted");

}

}

if (!cr1.IsFull) {

cr1.producer(t.getName());

}

if (this.id == 1){

try {

cb.cb1.await();

}

catch (InterruptedException e){

System.out.println(e.getMessage());

}

catch (BrokenBarrierException e){

System.out.println(e.getMessage());

}

}

if (this.id == 1){

cr2.a++;

cr2.b = !cr2.b;

System.out.println(t.getName() + " changed " + "a = " + cr2.a + " b = " + cr2.b);

}

}

}

}

class Consumer implements Runnable{

Thread t;

private int id;

CR1 cr1;

CR2 cr2;

CB cb;

SS sem;

Consumer (CR1 cr1, int id, SS sem, CR2 cr2, CB cb){

this.cr1 = cr1;

this.id = id;

this.sem = sem;

this.cr2 = cr2;

this.cb = cb;

t = new Thread (this, "Consumer " + id);

t.start();

}

public void run()

{

while (true)

{

if (this.id == 4){

sem.sem1.release();

try {

sem.sem2.acquire();

}catch (InterruptedException e){

System.out.println( t.getName() + "interrupted");

}

}

if (!cr1.IsEmpty) {

cr1.consumer(t.getName());

}

if (this.id == 2){

try {

cb.cb2.await();

}

catch (InterruptedException e){

System.out.println(e.getMessage());

}

catch (BrokenBarrierException e){

System.out.println(e.getMessage());

}

}

else if (this.id == 3){

try {

cb.cb1.await();

cb.cb2.await();

}

catch (InterruptedException e){

System.out.println(e.getMessage());

}

catch (BrokenBarrierException e){

System.out.println(e.getMessage());

}

}

if (this.id == 2){

if(++cr2.c > 'z') cr2.c = 'a';

cr2.b = !cr2.b;

System.out.println(t.getName() + " changed " + "c = \'" + cr2.c + "\' b = " + cr2.b);

}

else if (this.id == 3){

cr2.d += 0.1;

cr2.b = !cr2.b;

System.out.println(t.getName() + " changed " + "d = " + cr2.d + " b = " + cr2.b);

}

}

}

}

class Main

{

public static void main (String args[])

{

System.out.println ("To stop running press Ctrl+C");

CR1 MyCR1 = new CR1();

CR2 MyCR2 = new CR2(0, true, 'a', 0.1);

SS sem = new SS();

CB cb = new CB();

Producer P1 = new Producer(MyCR1, 1, sem, MyCR2, cb);

Consumer P2 = new Consumer(MyCR1, 2, sem, MyCR2, cb);

Consumer P3 = new Consumer(MyCR1, 3, sem, MyCR2, cb);

Consumer P4 = new Consumer(MyCR1, 4, sem, MyCR2, cb);

Producer P5 = new Producer(MyCR1, 5, sem, MyCR2, cb);

try{

P1.t.join();

P2.t.join();

P3.t.join();

P4.t.join();

P5.t.join();

}catch(InterruptedException e){

System.out.println("Main thread interrupted");

}

System.out.println("\n" + "1 " + "Thread is alive: " + P1.t.isAlive());

System.out.println("\n" + "2 " + "Thread is alive: " + P2.t.isAlive());

System.out.println("\n" + "3 " + "Thread is alive: " + P3.t.isAlive());

System.out.println("\n" + "4 " + "Thread is alive: " + P4.t.isAlive());

System.out.println("\n" + "5 " + "Thread is alive: " + P5.t.isAlive());

}

}

***Тест роботи програми***

To stop running press Ctrl+C

Producer 1+++ ind: 0 => 0 CREATED

Producer 1 changed a = 1 b = true

Producer 1+++ ind: 1 => 1 CREATED

Producer 1 changed a = 2 b = false

Producer 1+++ ind: 2 => 2 CREATED

Producer 1 changed a = 3 b = true

Producer 1+++ ind: 3 => 3 CREATED

Producer 1 changed a = 4 b = false

Producer 1+++ ind: 4 => 4 CREATED

Producer 1 changed a = 5 b = true

Consumer 2--- buf[0] = 0 DELETED

Consumer 2 changed c = '#' b = false

Producer 1+++ ind: 5 => 5 CREATED

Producer 1 changed a = 6 b = true

Producer 1+++ ind: 6 => 6 CREATED

Producer 1 changed a = 7 b = false

Consumer 2--- buf[1] = 1 DELETED

Consumer 2 changed c = '#' b = true

Producer 1+++ ind: 7 => 7 CREATED

Producer 1 changed a = 8 b = false

Consumer 2--- buf[2] = 2 DELETED

Consumer 2 changed c = '#' b = true

Producer 1+++ ind: 8 => 8 CREATED

Producer 1 changed a = 9 b = false

Consumer 2--- buf[3] = 3 DELETED

Consumer 2 changed c = '#' b = true

Producer 1+++ ind: 9 => 9 CREATED

Producer 1 changed a = 10 b = false

Consumer 2--- buf[4] = 4 DELETED

Consumer 2 changed c = '' b = true

Producer 1+++ ind: 10 => 10 CREATED

Producer 1 changed a = 11 b = false

Consumer 2--- buf[5] = 5 DELETED

Consumer 2 changed c = '#' b = true

Producer 1+++ ind: 11 => 11 CREATED

Producer 1 changed a = 12 b = false

Consumer 2--- buf[6] = 6 DELETED

Consumer 2 changed c = '' b = true

Producer 1+++ ind: 12 => 12 CREATED

Producer 1 changed a = 13 b = false

Consumer 2--- buf[7] = 7 DELETED

Consumer 2 changed c = ' b = true

Producer 1+++ ind: 13 => 13 CREATED

Producer 1 changed a = 14 b = false

Consumer 2--- buf[8] = 8 DELETED

Consumer 2 changed c = ' ' b = true

Producer 1+++ ind: 14 => 14 CREATED

Producer 1 changed a = 15 b = false

Consumer 2--- buf[9] = 9 DELETED

Consumer 2 changed c = '

' b = true

Producer 1+++ ind: 15 => 15 CREATED

Producer 1 changed a = 16 b = false

Consumer 2--- buf[10] = 10 DELETED

Consumer 2 changed c = '

' b = true

Producer 1+++ ind: 16 => 16 CREATED

Producer 1 changed a = 17 b = false

Consumer 2--- buf[11] = 11 DELETED

Consumer 2 changed c = '

' b = true

Producer 1+++ ind: 17 => 17 CREATED

Producer 1 changed a = 18 b = false

Consumer 2--- buf[12] = 12 DELETED

' b = true changed c = '

Consumer 2--- buf[13] = 13 DELETED

Consumer 2 changed c = '' b = false

Consumer 2--- buf[14] = 14 DELETED

Consumer 2 changed c = '' b = true

Consumer 3--- buf[15] = 15 DELETED

Consumer 3 changed d = 0.1 b = false

Consumer 2--- buf[16] = 16 DELETED

Consumer 2 changed c = '#' b = true

Producer 1+++ ind: 18 => 18 CREATED

Producer 1 changed a = 19 b = false

Producer 1+++ ind: 19 => 19 CREATED

Producer 1 changed a = 20 b = true

Producer 1+++ ind: 0 => 20 CREATED

Producer 1 changed a = 21 b = false

Producer 1+++ ind: 1 => 21 CREATED

Producer 1 changed a = 22 b = true

Consumer 2--- buf[17] = 17 DELETED

Consumer 2 changed c = '#' b = false

Consumer 2--- buf[18] = 18 DELETED

Consumer 2 changed c = '#' b = true

Consumer 2--- buf[19] = 19 DELETED

Consumer 2 changed c = '#' b = false

Consumer 2--- buf[0] = 20 DELETED

Consumer 2 changed c = '#' b = true

Consumer 2--- buf[1] = 21 DELETED

Consumer 2 changed c = '#' b = false

Consumer 2 changed c = '#' b = true

Consumer 2 changed c = '#' b = false

Consumer 2 changed c = '#' b = true

Consumer 2 changed c = '#' b = false

Consumer 2 changed c = '#' b = true

Consumer 2 changed c = '#' b = false

Consumer 2 changed c = '#' b = true

Consumer 2 changed c = '#' b = false

Consumer 2 changed c = '#' b = true

Consumer 2 changed c = '#' b = false

Consumer 2 changed c = ' ' b = true

Consumer 2 changed c = '!' b = false

Consumer 2 changed c = '"' b = true

Consumer 2 changed c = '#' b = false

Consumer 2 changed c = '$' b = true

Consumer 2 changed c = '%' b = false

Consumer 2 changed c = '&' b = true

Consumer 2 changed c = ''' b = false

Consumer 2 changed c = '(' b = true

Consumer 2 changed c = ')' b = false

Consumer 2 changed c = '\*' b = true

Consumer 2 changed c = '+' b = false

Consumer 2 changed c = ',' b = true

Consumer 2 changed c = '-' b = false

Consumer 2 changed c = '.' b = true

Consumer 2 changed c = '/' b = false

Consumer 2 changed c = '0' b = true

Consumer 2 changed c = '1' b = false

Consumer 2 changed c = '2' b = true

Consumer 2 changed c = '3' b = false

Consumer 2 changed c = '4' b = true

Consumer 2 changed c = '5' b = false

Consumer 2 changed c = '6' b = true

Consumer 2 changed c = '7' b = false

Consumer 2 changed c = '8' b = true

Producer 1+++ ind: 2 => 22 CREATED

Producer 1 changed a = 23 b = false

Producer 1+++ ind: 3 => 23 CREATED

Producer 1 changed a = 24 b = true

Producer 1+++ ind: 4 => 24 CREATED

Producer 1 changed a = 25 b = false

Producer 1+++ ind: 5 => 25 CREATED

Producer 1 changed a = 26 b = true

Producer 1+++ ind: 6 => 26 CREATED

Producer 1 changed a = 27 b = false

Producer 1+++ ind: 7 => 27 CREATED

Producer 1 changed a = 28 b = true

Producer 1+++ ind: 8 => 28 CREATED

Producer 1 changed a = 29 b = false

Producer 1+++ ind: 9 => 29 CREATED

Producer 1 changed a = 30 b = true

Producer 1+++ ind: 10 => 30 CREATED

Producer 1 changed a = 31 b = false

Producer 1+++ ind: 11 => 31 CREATED

Producer 1 changed a = 32 b = true

Producer 1+++ ind: 12 => 32 CREATED

Producer 1 changed a = 33 b = false

Producer 1+++ ind: 13 => 33 CREATED

Producer 1 changed a = 34 b = true

Producer 1+++ ind: 14 => 34 CREATED

Producer 1 changed a = 35 b = false

Producer 1+++ ind: 15 => 35 CREATED

Producer 1 changed a = 36 b = true

Producer 1+++ ind: 16 => 36 CREATED

Producer 1 changed a = 37 b = false

Consumer 3--- buf[2] = 22 DELETED

Consumer 3 changed d = 0.2 b = true

Producer 1+++ ind: 17 => 37 CREATED