C:\Users\V\AppData\Local\Temp\Rar$DRa0.667\kpi-tits\4.png

**Міністерство освіти та науки України  
Національний технічний університет України  
«Київський політехнічний інститут»  
Факультет прикладної математики  
Кафедра системного програмування і спеціалізованих  
комп’ютерних систем**

**Лабораторна робота №5**з дисципліни

**«Паралельні та розподілені обчислення»**Тема: **««Засоби взаємодії паралельних потоків мови Java»**

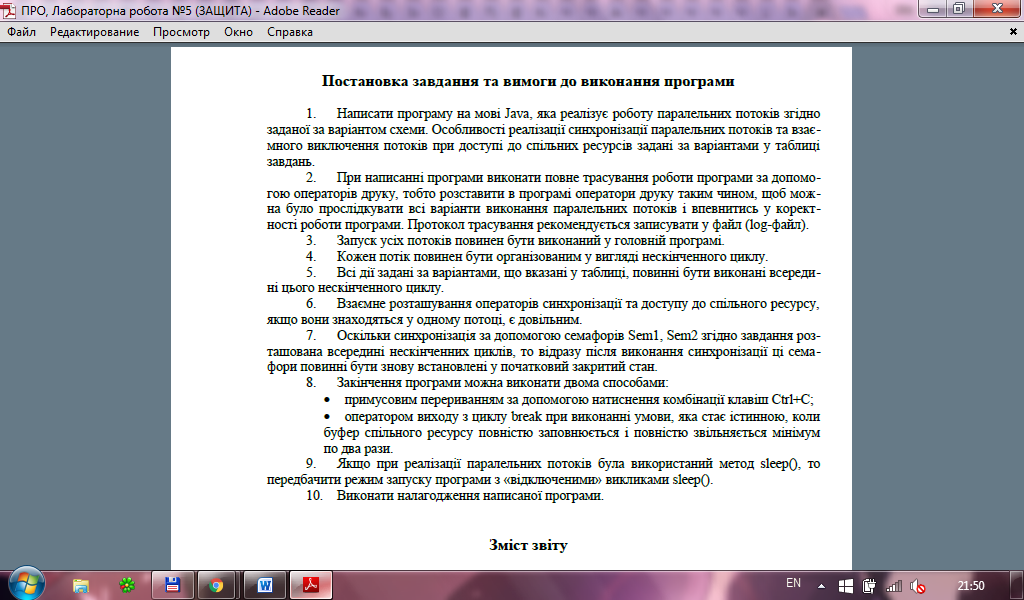
Виконав: Сахнік Іван Сергійович

Студент групи КВ-42

Перевірив\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**м. Київ**

**2016**



**Варіант№18.**

**Код програми:**

Main.java

public class Main{

public static void main(String[] args){

new Producer(1);

new Consumer(2);

new Consumer(3);

new Producer(4);

new Producer(5);

}

}

Global.java

import java.util.concurrent.CyclicBarrier;

import java.util.concurrent.Semaphore;

public class Global{

public static CyclicBarrier CB1 = new CyclicBarrier(3);

public static Semaphore Sem1 = new Semaphore(0, true);

public static Semaphore Sem2 = new Semaphore(0, true);

}

CommonVariables.java

import java.util.Random;

import java.util.concurrent.locks.ReentrantLock;

public class CommonVariables{

private static Random random = new Random();

public static ReentrantLock mutex = new ReentrantLock();

private static byte Byte;

private static short Short;

private static int Int;

private static long Long;

private static char Char;

public CommonVariables(){

CommonVariables.setByte();

CommonVariables.setShort();

CommonVariables.setInt();

CommonVariables.setLong();

CommonVariables.setChar();

}

public static byte getByte(){

return Byte;

}

public static void setByte(){

CommonVariables.Byte = (byte)random.nextInt(127);

}

public static short getShort(){

return Short;

}

public static void setShort(){

CommonVariables.Short = (short)random.nextInt(32767);

}

public static int getInt(){

return Int;

}

public static void setInt(){

CommonVariables.Int = random.nextInt();

}

public static long getLong(){

return Long;

}

public static void setLong(){

CommonVariables.Long = random.nextLong();

}

public static char getChar(){

return Char;

}

public static void setChar(){

CommonVariables.Char = (char)random.nextInt(255);

}

public static void updateVariables(int i){

System.out.println("Thread" + i + " changes: ");

switch (i){

case 2:

setByte();

setChar();

setInt();

setLong();

break;

case 3:

setShort();

setLong();

setInt();

setChar();

break;

case 5:

setByte();

setShort();

setLong();

setChar();

break;

}

synchronized (System.out){

showValue(Byte);

showValue(Short);

showValue(Int);

showValue(Long);

showValue(Char);

}

}

private static void showValue(byte Type){

System.out.println(" Byte value: " + getByte());

}

private static void showValue(short Type){

System.out.println(" Short value: " + getShort());

}

private static void showValue(int Type){

System.out.println(" Int value: " + getInt());

}

private static void showValue(long Type){

System.out.println(" Long value: " + getLong());

}

private static void showValue(char Type){

System.out.println(" Char value: " + getChar());

}

}

CommonStack.java

import java.util.Stack;

import java.util.concurrent.locks.Condition;

import java.util.concurrent.locks.ReentrantLock;

public class CommonStack{

private static final int MaxSize = 99;

private static Stack<Integer> stack = new Stack<Integer>();

private static final ReentrantLock mutex = new ReentrantLock();

private static final Condition not\_empty = mutex.newCondition();

private static final Condition not\_full = mutex.newCondition();

public static void pop(int threadNumber){

mutex.lock();

while (stack.size() == 0)

try{

not\_full.await();

}

catch (InterruptedException e){

e.printStackTrace();

}

System.out.println("Thread" + threadNumber + " gets: " + stack.pop());

not\_empty.signal();

mutex.unlock();

}

public static void push(int threadNumber){

mutex.lock();

while (stack.size() == MaxSize)

try{

not\_empty.await();

}

catch (InterruptedException e){

e.printStackTrace();

}

stack.push(threadNumber);

System.out.println("Thread" + threadNumber + " puts: " + threadNumber);

not\_full.signal();

mutex.unlock();

}

}

Producer.java

import java.util.concurrent.BrokenBarrierException;

public class Producer implements Runnable{

private int threadNumber;

public Producer(int num){

threadNumber = num;

System.out.println("Thread" + threadNumber + " started");

new Thread(this, "Thread" + threadNumber + "\_Producer").start();

}

public void run(){

while (true){

CommonStack.push(threadNumber);

if(threadNumber == 1){

try{

//Global.Sem1.acquire();

Global.Sem2.release();

System.out.println("P1 acquire Sem2");

System.out.println("P1 start waiting for Sem1");

//Global.Sem2.release();

Global.Sem1.acquire();

System.out.println("P1 finish waiting for Sem1");

} catch (InterruptedException e){

e.printStackTrace();

}

}

if(threadNumber == 4){

try{

//Global.Sem2.acquire();

Global.Sem1.release();

System.out.println("P4 release Sem1");

System.out.println("P4 start waiting for Sem2");

//Global.Sem1.release();

Global.Sem2.acquire();

System.out.println("P4 finish waiting for Sem2");

} catch (InterruptedException e){

e.printStackTrace();

}

}

if(threadNumber == 5){

try{

CommonVariables.mutex.lock();

CommonVariables.updateVariables(threadNumber);

CommonVariables.mutex.unlock();

Global.CB1.await();

} catch (InterruptedException e){

e.printStackTrace();

} catch (BrokenBarrierException e){

e.printStackTrace();

}

}

}

}

}

Consumer.java

import java.util.concurrent.BrokenBarrierException;

public class Consumer implements Runnable{

private int threadNumber;

public Consumer(int num){

threadNumber = num;

System.out.println("Thread" + threadNumber + " started");

new Thread(this, "Thread" + threadNumber + "\_Consumer").start();

}

public void run(){

while (true){

CommonStack.pop(threadNumber);

if(threadNumber == 2){

try{

CommonVariables.mutex.lock();

CommonVariables.updateVariables(threadNumber);

CommonVariables.mutex.unlock();

Global.CB1.await();

} catch (InterruptedException e){

e.printStackTrace();

} catch (BrokenBarrierException e){

e.printStackTrace();

}

}

else if(threadNumber == 3){

try{

CommonVariables.mutex.lock();

CommonVariables.updateVariables(threadNumber);

CommonVariables.mutex.unlock();

Global.CB1.await();

} catch (InterruptedException e){

e.printStackTrace();

} catch (BrokenBarrierException e){

e.printStackTrace();

}

}

}

}

}

**Log file:**

Thread1 started

Thread2 started

Thread3 started

Thread4 started

Thread1 puts: 1

Thread2 gets: 1

Thread5 started

Thread4 puts: 4

Thread5 puts: 5

Thread5 changes:

Byte value: 29

Short value: 10304

Int value: 0

Long value: 3745954445836905285

Char value: ]

Thread2 changes:

Byte value: 114

Short value: 10304

Int value: 364973314

Long value: 3035875524365010787

Char value: T

Thread3 gets: 5

Thread3 changes:

Byte value: 114

Short value: 3370

Int value: -2085770026

Long value: 7497572704224156338

Char value: #

P1 acquire Sem2

P1 start waiting for Sem1

P4 release Sem1

P4 start waiting for Sem2

P4 finish waiting for Sem2

P1 finish waiting for Sem1

Thread5 puts: 5

Thread5 changes:

Thread2 gets: 5

Thread3 gets: 4

Byte value: 99

Short value: 12733

Int value: -2085770026

Long value: 6466030390551300536

Char value: ¯

Thread4 puts: 4

P4 release Sem1

P4 start waiting for Sem2

Thread1 puts: 1

P1 acquire Sem2

P1 start waiting for Sem1

P1 finish waiting for Sem1

Thread1 puts: 1

P1 acquire Sem2

P1 start waiting for Sem1

Thread2 changes:

P4 finish waiting for Sem2

Byte value: 30

Short value: 12733

Int value: -203638227

Long value: -2416140293518763740

Char value: û

Thread4 puts: 4

P4 release Sem1

P4 start waiting for Sem2

P1 finish waiting for Sem1

Thread3 changes:

P4 finish waiting for Sem2

Byte value: 30

Short value: 19254

Int value: -332245954

Long value: -5464317301912434572

Char value: Þ

Thread1 puts: 1

P1 acquire Sem2

P1 start waiting for Sem1

Thread4 puts: 4

P4 release Sem1

P4 start waiting for Sem2

Thread3 gets: 4

P1 finish waiting for Sem1

Thread3 changes:

P4 finish waiting for Sem2

Byte value: 30

Short value: 15185

Int value: -1791903596

Long value: 4896494378606254214

Char value: í

Thread1 puts: 1

P1 acquire Sem2

P1 start waiting for Sem1

Thread5 puts: 5

Thread5 changes:

Thread2 gets: 5

Byte value: 97

Short value: 14486

Int value: -1791903596

Long value: 1605503980702319529

Char value: à

Thread4 puts: 4

P4 release Sem1

P4 start waiting for Sem2

P1 finish waiting for Sem1

Thread2 changes:

P4 finish waiting for Sem2

Byte value: 40

Short value: 14486

Int value: -1745640621

Long value: -5732566342874906265

Char value: Ë

Thread1 puts: 1

P1 acquire Sem2

P1 start waiting for Sem1

Thread4 puts: 4

P4 release Sem1

P4 start waiting for Sem2

P1 finish waiting for Sem1

Thread2 gets: 4

P4 finish waiting for Sem2

Thread2 changes:

Thread4 puts: 4