Дисципліна: Технології паралельних обчислень

ПІБ студента: Спаських Микола Дмитрович

Група ІП-12

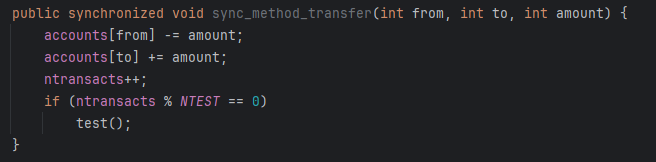
Викладач: Дифучин Антон Юрійович

Лабораторна робота 3

Тема: Завдання до комп’ютерного практикуму 3 «Розробка паралельних програм з використанням механізмів синхронізації: синхронізовані методи, локери, спеціальні типи»

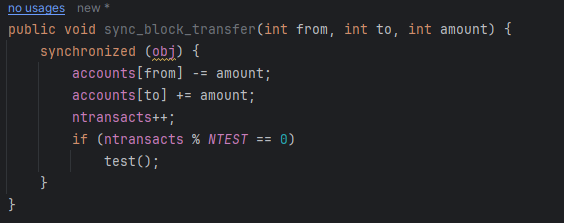
Завдання 1: Реалізуйте програмний код, даний у лістингу, та протестуйте його при різних значеннях параметрів. Модифікуйте програму, використовуючи методи управління потоками, так, щоб її робота була завжди коректною. Запропонуйте три різних варіанти управління.

Синхронізований метод:



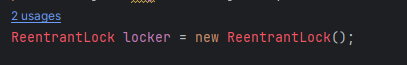
Синхронізований блок:

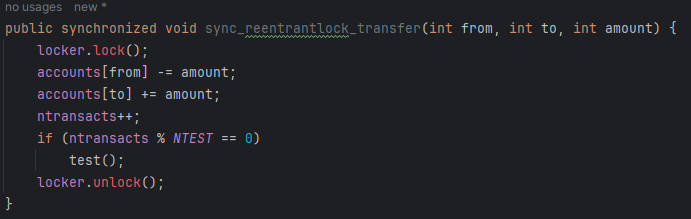




Рентрантлок:







Завдання 2: Реалізуйте приклад Producer-Consumer application

(див. https://docs.oracle.com/javase/tutorial/essential/concurrency/guardmeth.html ).

Модифікуйте масив даних цієї програми, які читаються, у масив чисел

заданого розміру (100, 1000 або 5000) та протестуйте програму. Зробіть

висновок про правильність роботи програми.

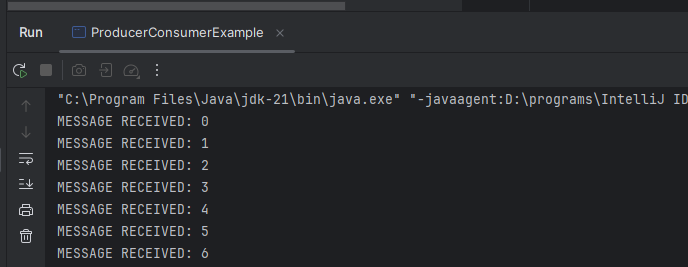
Код виконання:

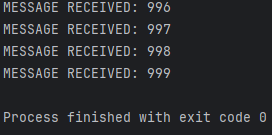
package p2;  
  
import java.util.Random;  
  
public class Consumer implements Runnable {  
 private Drop drop;  
 private int N;  
  
 public Consumer(Drop drop, int n) {  
 this.drop = drop;  
 this.N=n;  
 }  
  
 public void run() {  
 Random random = new Random();  
 for (int message = drop.take(); message!=-1; message = drop.take()) {  
 System.*out*.format("MESSAGE RECEIVED: %d%n", message);  
 try {  
 Thread.*sleep*(random.nextInt(5));  
 } catch (InterruptedException e) {}  
 }  
 }  
}

package p2;  
  
public class Drop {  
 // Повідомлення, відправлене від виробника  
 // до споживача.  
 private int message;  
 // true, якщо споживач повинен чекати,  
 // поки виробник надішле повідомлення,  
 // false, якщо виробник повинен чекати,  
 // поки споживач отримає повідомлення.  
 private boolean empty = true;  
  
 public synchronized int take() {  
 // Чекаємо, поки повідомлення буде доступне.  
 while (empty) {  
 try {  
 wait();  
 } catch (InterruptedException e) {}  
 }  
 // Змінюємо статус.  
 empty = true;  
 // Повідомляємо виробника, що  
 // статус змінився.  
 notifyAll();  
 return message;  
 }  
  
 public synchronized void put(int message) {  
 // Чекаємо, поки повідомлення буде отримане.  
 while (!empty) {  
 try {  
 wait();  
 } catch (InterruptedException e) {}  
 }  
 // Змінюємо статус.  
 empty = false;  
 // Зберігаємо повідомлення.  
 this.message = message;  
 // Повідомляємо споживача, що статус  
 // змінився.  
 notifyAll();  
 }  
}

package p2;  
  
import java.util.Random;  
  
public class Producer implements Runnable {  
 private Drop drop;  
 private int[] mas;  
 private int N;  
  
 public Producer(Drop drop, int n) {  
 this.drop = drop;  
 this.N=n;  
 mas = new int[N];  
 for (int i = 0; i<N; i++){  
 mas[i] = i;  
 }  
 }  
  
 public void run() {  
 Random random = new Random();  
  
 for (int i = 0; i < mas.length; i++) {  
 drop.put(mas[i]);  
 try {  
 Thread.*sleep*(random.nextInt(5));  
 } catch (InterruptedException e) {}  
 }  
 drop.put(-1);  
 }  
}

package p2;  
  
public class ProducerConsumerExample {  
 public static void main(String[] args) {  
 int N=100; //1000, 10000  
 Drop drop = new Drop();  
 (new Thread(new Producer(drop, N))).start();  
 (new Thread(new Consumer(drop, N))).start();  
 }  
}

Результат виконання (для 1000):  




Програма спрацювала правильно.

Завдання 3: Реалізуйте роботу електронного журналу групи, в якому зберігаються оцінки з однієї дисципліни трьох груп студентів. Кожного тижня лектор і його 3 асистенти виставляють оцінки з дисципліни за 100-бальною шкалою.

Код:

package p3;  
  
public class jornal {  
 public String [][][][] mas = new String [3][20][12][2]; // група, студент, тиждень, лектор чи асистент  
  
 public synchronized void put (int group, int student, int week, int lector, String grade){  
 mas[group][student][week][lector]=grade;  
 }  
 public void print() {  
 for(int gro = 0; gro < 3; gro ++) {  
 System.*out*.print("\n Group "+gro+":\n");  
 for (int st = 0; st < 20; st++) {  
 System.*out*.print("\n Student "+st+":\n");  
 for (int l=0; l<2; l++){  
 if (l==0) {  
 System.*out*.print("As: ");  
 }  
 else System.*out*.print("\nLe: ");  
 for (int w = 0; w < 12; w++) {  
 System.*out*.print(mas[gro][st][w][l]+" ");  
 }  
 }  
 }  
 }  
 }  
}

package p3;  
public class main {  
 public static void main(String[] args) throws InterruptedException {  
 jornal j = new jornal();  
 Thread[] threads = new Thread[4]; // Масив для зберігання посилань на створені потоки  
 Runnable r = () -> {  
 threads[0] = new Thread(new producer(j, "Lecturer 1", 1, new int[]{0,1,2}));  
 threads[1] = new Thread(new producer(j, "Assistant 1", 0, new int[]{0}));  
 threads[2] = new Thread(new producer(j, "Assistant 2", 0, new int[]{1}));  
 threads[3] = new Thread(new producer(j, "Assistant 3", 0, new int[]{2}));  
  
 for (Thread t : threads) {  
 t.start(); // Запускаємо кожен потік  
 }  
 };  
  
 Thread t = new Thread(r);  
 t.start();  
 t.join(); // Очікуємо завершення основного потоку  
  
 try {  
 for (Thread thread : threads) {  
 thread.join(); // Очікуємо завершення кожного потоку  
 }  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
  
 j.print();  
 }  
}

package p3;  
import java.util.Random;  
public class producer implements Runnable{  
 private jornal j;  
 private String name;  
 private int[] groups;  
 private int lector;  
 public producer(jornal j, String who, int lector, int[] g){  
 this.j=j;  
 name = who;  
 groups = g;  
 this.lector=lector;  
 }  
 public void run () {  
 Random random = new Random();  
 for (int g : groups){  
 for(int stud=0; stud<20; stud ++){  
 for(int w=0; w<12; w++) {  
 j.put(g, stud, w, lector, random.nextInt(100) + " (" + name+ ");");  
 }  
 }  
 }  
 }  
}

Результат виконання:  
