­­МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

­­­



Лабораторна робота №2

з курсу «**Технології розподілених систем та паралельних обчислень**»

для студентів базового напрямку 6.08.04 "Комп’ютерні науки"

(заочна форма навчання)

Варіант 11

Виконав студент гр. КНз-41

Чалий Михaйло

Перевірив

Фармага І.В.

­­

Львів 2016

## Мета роботи

Отримати навики побудови застосування методів синхронізації обчислювальних процесів в розгалужених та взаємодіючих процесах на мові Java.

## Теоретична частина

Ключем до синхронізації є концепція монітору (також називають *семафором*). Монітор – це об‘єкт, який використовується як взаємне виключення (*mutually exclusive lock – mutex*) або м‘ютекс. Тільки один потік одночасно може володіти монітором. Коли потік запитує блокування, кажуть що він входить в монітор. Всі інші потоки, які пробують зайти у заблокований монітор, будуть призупинені до тої пори, поки перший потік на вийде із монітору. Про всі інші потоки кажуть, що вони очікують монітор. Потік, котрий володіє монітором, може, при бажанні, повторно зайти в нього.

Синхронізацію в інших мовах, таких як С чи С++ використовувати непросто. Це тому, що ці мови самі по собі не підтримують синхронізацію. Замість того, щоб синхронізувати потоки, ваша програма повинна використовувати примітиви операційної системи. На щастя, оскільки Java peaлізує синхронізацію через мовні елементи, велика частина складності, асоційована з синхронізацією, зникає.

## Завдання

Варіант 11. Програма моделює обслуговування одного потоку процесів одним центральним процесором комп'ютера з однією чергою. Якщо черговий процес генерується в мить, коли процесор вільний, процес поступає на обробку в процесор, інакше процес поступає в чергу. Якщо процесор вільний і в черзі є процеси, процес віддаляється з черги. Визначте максимальну довжину черги.

## Реалізація

Лістінг L2PrintSynchronized.java - https://github.com/chaliy/studies-octo-adventure/blob/master/lp/c4\_2/trs/ L2PrintSynchronized.java

import java.util.Random;

import java.util.Queue;

public class L2PrintSynchronized {

private static final Random rnd = new Random();

private static final Object sync = new Object();

private static Thread createSyncCaller(String msg){

return new Thread(() -> {

try{

synchronized(sync) {

System.out.print("[" + msg);

Thread.sleep(500);

System.out.println("]");

}

} catch(InterruptedException ex) {

Thread.currentThread().interrupt();

}

});

}

private static Thread createCaller(String msg){

return new Thread(() -> {

try{

System.out.print("[" + msg);

Thread.sleep(500);

System.out.println("]");

} catch(InterruptedException ex) {

Thread.currentThread().interrupt();

}

});

}

public static void main(String[] args) {

// Не синхронізована версія

Thread[] callers = {

createCaller("Прізвище Ім'я Побатькові"),

createCaller("Чалий Михайло Васильович"),

createCaller("КНз-41"),

createCaller("mike@chaliy.name")

};

for(Thread caller : callers) { caller.start(); }

// Cинхронізована версія

Thread[] syncCallers = {

createSyncCaller("Прізвище Ім'я Побатькові"),

createSyncCaller("Чалий Михайло Васильович"),

createSyncCaller("КНз-41"),

createSyncCaller("mike@chaliy.name")

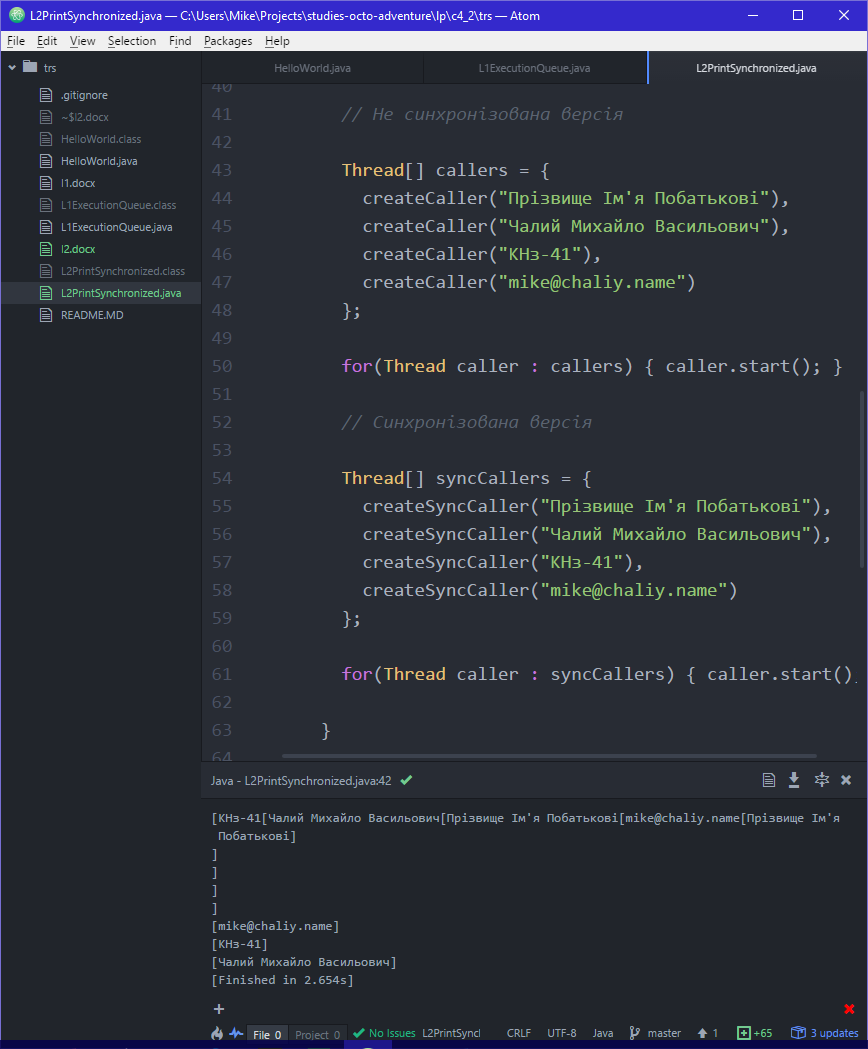
};

for(Thread caller : syncCallers) { caller.start(); }

}

}

## Результат



## Висновки

Отримав навики побудови застосування методів синхронізації обчислювальних процесів в розгалужених та взаємодіючих процесах на мові Java.