­­МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

­­­



Лабораторна робота №1

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

Одержати навики побудови багатопотокових алгоритмів обчислювальних процесів, навчитися складати програми з розгалуженими та взаємодіючими процесами на мові Java.

## Теоретична частина

В класі Thread сім конструкторів:

Thread(ThreadGroup group, Runnable target, String name) — створює підпроцес з іменем name, належний групі group і виконуючий метод run() обєкта target. Це основной конструктор, всі інші звертаються до нього з тим чи іншим параметром, рівним null;

Thread() — створюваний підпроцесс буде виконувати свій метод run();

Thread(Runnable target);

Thread(Runnable target, String name);

Thread(String name);

Thread(ThreadGroup group, Runnable target);

Thread(ThreadGroup group, String name).

Імя підпроцесу name не має ніякого значення, воно не використовується віртуальною машиною Java і застосовується тільки для відрізняння підпроцесів в програмі. Після створення підпроцесу його треба запустити методом start(). Віртуальна машина Java почне виконувати метод run() цього обєкта-підпроцеса. Підпроцес завершить роботу після виконання метода run(). Для знищення обєкта-підпроцеса вслід за цим він повинен присвоїти значення null.

## 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 L1ExecutionQueue.java - https://github.com/chaliy/studies-octo-adventure/blob/master/lp/c4\_2/trs/L1ExecutionQueue.java

import java.util.Random;

import java.util.Queue;

import java.util.concurrent.ConcurrentLinkedQueue;

public class L1ExecutionQueue {

private static final Queue<String> queue = new ConcurrentLinkedQueue<String>();

private static final Random rnd = new Random();

private static void work(int ms) {

try{

int wait = (ms + rnd.nextInt(ms)) / 2;

Thread.sleep(wait);

} catch(InterruptedException ex) {

Thread.currentThread().interrupt();

}

}

public static void main(String[] args) {

Thread producer = new Thread(() -> {

for (int x = 0; x < 100; x++) {

String job = "Job" + x;

System.out.println("Додаємо роботу: " + job);

queue.add(job);

work(300);

}

});

Thread consumer = new Thread(() -> {

for (int x = 0; x < 100; x++) {

String job = queue.poll();

if (job == null) {

System.out.println("В черзі немає роботи, чекаємо");

work(500);

} else {

System.out.println("Виконуємо роботу " + job + ", ще залишилось: " + queue.size());

work(1000); // Emulate hard work :)

}

}

});

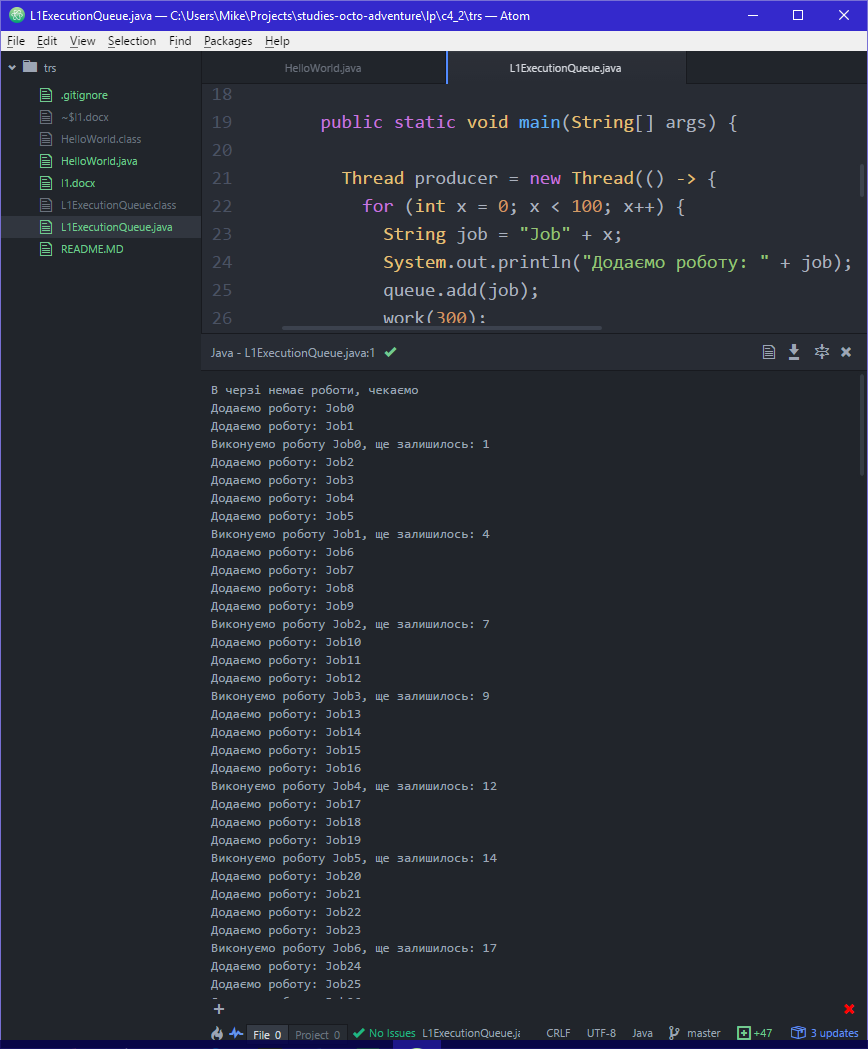
producer.start();

consumer.start();

}

}

## Результат



## Висновки

Одержав навики побудови багатопотокових алгоритмів обчислювальних процесів, навчитився складати програми з розгалуженими та взаємодіючими процесами на мові Java.