МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”
Кафедра систем штучного інтелекту

Лабораторна робота № 13

з дисципліни
«Об’єктно-орієнтоване програмування»

Виконала:
студентка групи КН-109
Пелещак Ю. М.
Викладач:
Гасько Р.Т.

Львів – 2018 р.

Лабораторна робота №13.
Паралельне виконання. Багатопоточність. Ефективність використання.

Мета

● Ознайомлення з моделлю потоків Java.
● Організація паралельного виконання декількох частин програми.
● Вимірювання часу паралельних та послідовних обчислень.
● Демонстрація ефективності паралельної обробки.

Вимоги

1. Використовуючи програми рішень попередніх задач, продемонструвати можливість паралельної обробки елементів контейнера: створити не менше трьох додаткових потоків, на яких викликати відповідні методи обробки контейнера.
2. Забезпечити можливість встановлення користувачем максимального часу виконання (таймаута) при закінченні якого обробка повинна припинятися незалежно від того знайдений кінцевий результат чи ні.
3. Для паралельної обробки використовувати алгоритми, що не змінюють початкову колекцію.
4. Кількість елементів контейнера повинна бути досить велика, складність алгоритмів обробки колекції повинна бути зіставна, а час виконання приблизно однаковий, наприклад:
○ пошук мінімуму або максимуму;
○ обчислення середнього значення або суми;
○ підрахунок елементів, що задовольняють деякій умові;
○ відбір за заданим критерієм;
○ власний варіант, що відповідає обраній прикладної області.
5. Забезпечити вимірювання часу паралельної обробки елементів контейнера за допомогою розроблених раніше методів.
6. Додати до алгоритмів штучну затримку виконання для кожної ітерації циклів поелементної обробки контейнерів, щоб загальний час обробки був декілька секунд.
7. Реалізувати послідовну обробку контейнера за допомогою методів, що використовувались для паралельної обробки та забезпечити вимірювання часу їх роботи.
8. Порівняти час паралельної і послідовної обробки та зробити висновки про ефективність розпаралелювання:
○ результати вимірювання часу звести в таблицю;
○ обчислити та продемонструвати у скільки разів паралельне виконання швидше послідовного.

Код,що відповідає за виконання програми:

```
package ua.lpnuai.peleshchak13;


import java.util.*;
```

```
public class Main {
    public static void main(String[] args) {
        LinkedList<String> ls = new LinkedList<String>();
        ls.push("1");
        ls.push("5");
        ls.push("3");
        ls.push("1");
        ls.push("45");
        ls.push("12");
        ls.push("123");
        ls.push("2");

        First d = new First("First");
        d.start();
        System.out.println("Main Thread:  " + System.currentTimeMillis()*100);

        Add add = new Add(ls,"Second");
        add.start();

        Min min = new Min(ls,"Third");
        min.start();

    }
}


package ua.lpnuai.peleshchak13;

class First extends Thread {
    Thread t;
    String name;
    First (String name) {
        this.name = name;
    }
    public void run () {
        try {
            System.out.println("The FIRST --- Thread began");
            Thread.sleep(5000);
            System.out.println("The FIRST --- Thread ended");
        }
        catch(InterruptedException e) {

        }
    }
    public void start () {
        if (t == null) {
            t = new Thread(this, name);
            t.start();
        }
    }
}
package ua.lpnuai.peleshchak13;

import java.util.Collections;
import java.util.LinkedList;
import java.util.Random;

class Min extends Thread{
    Thread three;
    String name;
    LinkedList<String> list;

    Min(LinkedList<String> ls, String name){
        this.name = name;
        list = ls;
```

```
    }

    public void run() {
        list.push("Q");
        list.push("W");
        list.push("E");
        list.push("R");
        list.push("T");


        LinkedList<Integer> arr = new LinkedList<>();
        Random rand = new Random();
        for(int i = 0; i < list.size(); i++) {

            Integer    k = rand.nextInt(1000);
            arr.add(k);

        }
        list.push("A");
        list.push("B");
        list.push("C");
        list.push("D");


        Integer max = Collections.max(arr);
        Integer min = Collections.min(arr);
        System.out.println("THIRD THRED :  "+arr);

        System.out.println("THIRD THREAD - max element:  "+max);
        System.out.println("THIRD THREAD -  min element:  "+min);
    }
    public void start() {
        if(three == null) {
            three = new Thread(this, name);
            three.start();
        }
    }
}
package ua.lpnuai.peleshchak13;

import java.time.Clock;
import java.time.DateTimeException;
import java.time.Instant;
import java.util.LinkedList;
import java.util.Random;

class Add extends Thread{
    Thread two;
    LinkedList<String> list;
    String name;
    Add(LinkedList<String> ls,String name){
        list =  ls;
        this.name = name;
    }
    public void run() {
        try {
            Instant t = Clock.systemUTC().instant();
            System.out.println("SECOND THREAD - time:  " +t);
        }catch (DateTimeException ex)
        {
        }

        Random rand = new Random();
        for(int i = 0; i < 10; i++) {

            String k = new Integer(rand.nextInt(20) + 1).toString();
```

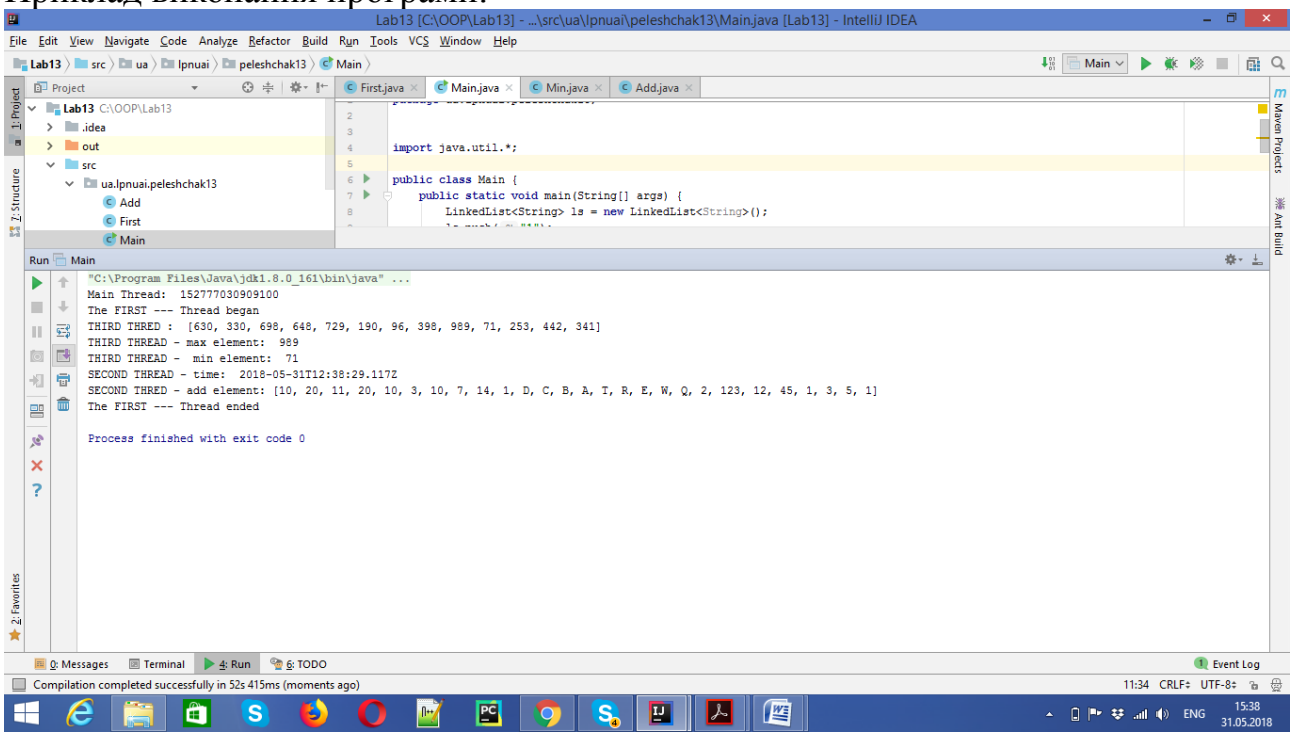
```
                list.push(k);


            }

            System.out.println("SECOND THRED - add element: "+ list);
        }

        public void start() {
            if(two == null) {
                two = new Thread(this,name);
                two.start();
            }
        }
}
```

Приклад виконання програми:



Висновок: під час виконання лабораторної роботи №13 я навчилась працювати з багатопотоковістю.