Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Лабораторна робота №13

з курсу:

“ООП”

Виконав:
ст. гр. КН-110
Шаварський
Максим
Прийняв:
Гасько Р.Т.

Львів – 2018 р.

Лабораторна робота № 13

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 програми

```
import java.util.ArrayList;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

class Ticker implements Runnable
{
    private ArrayList arr;
    private String function;
    public Ticker(ArrayList arr,String funtion)
    {
        this.arr = arr;
        this.function = funtion;
    }
    public static Integer Min(ArrayList arr)
    {
        int min = arr.indexOf(0);

        for(int i = 0; i < arr.size();i++)
        {
            if(min > arr.indexOf(i))
            {
                min = arr.indexOf(i);
            }
        }

        return min;
    }
    public static Integer Max(ArrayList arr)
    {
        int max = arr.indexOf(0);

            for (int i = 0; i < arr.size(); i++) {
                if (max < arr.indexOf(i)) {
                    max = arr.indexOf(i);
                }
            }

        return max;
    }
    public static Integer Sum(ArrayList arr)
    {
        int sum=0;

        for(int i = 0; i < arr.size();i++)
        {
            sum+= arr.indexOf(i);
        }
        return sum;
    }
    @Override
    public void run() {
        switch (function){
            case "Min":
                System.out.println(function +": " + Min(this.arr));
```

```
                break;
            case "Max":System.out.println(function +": " + Max(this.arr));
                break;
            case "Sum":System.out.println(function +": " + Sum(this.arr));
                break;
            default:
                System.out.println("Incorrect input.");
                break;
        }
    }
}
public class lab
{
    public static void main(String[] args)
    {
        long ls1,en1,ls2,en2;
         ArrayList<Integer> al = new ArrayList();
         for (int i =0; i < 1000;i++)
         {
             al.add(1 +(int)( Math.random() * 1000));
         }
         Ticker first = new Ticker(al,"Min");
        Ticker second = new Ticker(al,"Max");
        Ticker thierd = new Ticker(al,"Sum");
        ls1 = System.nanoTime();
        ExecutorService ES = Executors.newFixedThreadPool(3);
        ES.submit(first);
        ES.submit(second);
        ES.submit(thierd);
        en1 = System.nanoTime();
        System.out.println("\nParallel executing time: "+(en1-ls1));
        ls2= System.nanoTime();
        first.Min(al);
        second.Max(al);
        thierd.Sum(al);
        en2 = System.nanoTime();
        System.out.println("Successively executing time: "+(en2-ls2));
    }
}
```