МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту

Лабораторна робота № 13

з дисципліни
«Об’єктно-орієнтоване програмування»

Варіант - 6

Виконав:
студент групи КН-108
Яковлев В.К.
Викладач:
Гаскьо Р.Т.

Львів – 2018 р.

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 рішення:

```
package com.week;
import java.util.Arrays;
import java.util.Scanner;

public class Lab13 {

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);
        boolean isTrue;
        System.out.print("Enter number of array elements: ");
        int size = input.nextInt();
        int array[] = new int[size], max = array[0], decision,
sum = 0, find, temp=0, z = 0;

        System.out.print("Enter some array elements, please:
");
        for (int i = 0; i < size; i++) {
            array[i] = input.nextInt();
        }

        do {
            System.out.println("Press 1 for parallelised
execution \nPress 2 for typical execution \nPress 3 to compare
times of executions");
            decision = input.nextInt();
            if(decision == 1) {

                System.out.print("Enter element, which you
want to find: ");
                find = input.nextInt();

                for (int i = 0; i < array.length; i++)
                {
                    if (find == array[i]){
                        System.out.println("Element was
found");
                        break;
```

```
                }
            }

            long startTime = System.currentTimeMillis();

            for (int a = 0; a < array.length; a++) {
                if (array[a] > max)
                    max = array[a];
            }

            delayForSecond();

            for( int num : array) {
                sum = sum + num;
            }
            System.out.println("Sum of elements : " + sum);

            System.out.print("Sorted Array: ");
            Arrays.parallelSort(array);
            for (int i : array)
                System.out.print(i + " ");

            System.out.println();

            delayForSecond();

            System.out.print("Prefix: ");
            Arrays.parallelPrefix(array, (x, y) -> x * y);
            for (int a : array)
                System.out.print(a + " ");

            delayForSecond();

            System.out.println();
            System.out.println("Max : " + max);

            delayForSecond();

            long timeSpent = System.currentTimeMillis() -
startTime;
            System.out.println("Time of program executing:
" + timeSpent / 1000 + " seconds");
            isTrue = true;
        }
        else if (decision == 2){
            System.out.print("Enter element which you want
to find: ");
```

```
                find = input.nextInt();

                for (int i = 0; i < array.length; i++)
                {
                    if (find == array[i]){
                        System.out.println("Element was
found");
                        break;
                    }
                }

                long startTime = System.currentTimeMillis();

                for (int a = 0; a < array.length; a++) {
                    if (array[a] > max)
                        max = array[a];
                }

                delayForSecond();

                for( int num : array) {
                    sum = sum + num;
                }
                System.out.println("Sum of elements: " + sum);

                delayForSecond();

                System.out.println("Sorted array: ");
                for (int i = 0; i < array.length; i++) {
                    for (int j = 1; j < (array.length - i); j++)
{

                        if (array[j - 1] > array[j]) {
                            temp = array[j - 1];
                            array[j - 1] = array[j];
                            array[j] = temp;
                        }

                    }
                }

                for(int a = 0; a < array.length; a++)
                    System.out.print(array[a] + " ");
                System.out.println();

                delayForSecond();
```

```
                System.out.print("Prefix: ");
                System.out.print(array[z] + " ");
                do{
                    array[z] = array[z] * array[z + 1];
                    System.out.print(array[z] + " ");
                    z++;
                }while((z+1) != array.length);

                System.out.println();
                delayForSecond();
                long timeSpent = System.currentTimeMillis() -
startTime;
                System.out.println("Time of program executing:
" + timeSpent / 1000 + " seconds");
                isTrue = true;
            }
            else if(decision == 3){
                long startTime1 = System.currentTimeMillis();

                for (int a = 0; a < array.length; a++) {
                    if (array[a] > max)
                        max = array[a];
                }

                for( int num : array) {
                    sum = sum + num;
                }

                Arrays.parallelSort(array);

                Arrays.parallelPrefix(array, (x, y) -> x * y);

                long timeSpent1 = System.currentTimeMillis() -
startTime1;
                long startTime2 = System.currentTimeMillis();

                for (int a = 0; a < array.length; a++) {
                    if (array[a] > max)
                        max = array[a];
                }

                for( int num : array) {
                    sum = sum + num;
                }

                for (int i = 0; i < array.length; i++) {
                    for (int j = 1; j < (array.length - i); j++)
```

```
{
                    if (array[j - 1] > array[j]) {
                        temp = array[j - 1];
                        array[j - 1] = array[j];
                        array[j] = temp;
                    }

                }
            }

            do{
                array[z] = array[z] * array[z + 1];
                z++;
            }while((z+1) != array.length);

            long timeSpent2 = System.currentTimeMillis() -
startTime2;
            System.out.println("Difference in time is " +
(timeSpent1 - timeSpent2) + " milliseconds");
            isTrue = true;
        }
        else{
            System.out.println("Such command is
unsupported!");
            isTrue = false;
        }
    }while (!isTrue);
}

private static void delayForSecond() {
    try {
        Thread.sleep(1000);
    } catch (java.lang.InterruptedException iex) {
        System.out.println(iex);
    }
}
}
```