

Міністерство освіти і науки України
Національний технічний університет України «Київський
політехнічний інститут імені Ігоря Сікорського»

Звіт

з лабораторної роботи № 8

з дисципліни

«Мова програмування Java»

Київ 2025

Я обрала Варіант 2, який полягає у розробці консольного додатка для паралельного обчислення суми елементів масиву за допомогою фреймворку ForkJoin.

Реалізація представлена у файлі ArraySumForkJoin.java та базується на такій логіці:

- Створення даних: Я реалізувала створення та ініціалізацію масиву розмірністю 1,000,000 елементів випадковими числами в діапазоні від 0 до 100.

- Рекурсивна логіка: Для обчислення суми я описала клас SumTask, який наслідує RecursiveTask. Програма рекурсивно ділить масив на дві частини, створюючи окремі завдання для кожної.

- Для оптимізації я встановила поріг (Threshold) у 20 елементів. Це означає, що розподіл масиву триває доти, доки підзадача не стане достатньо малою для ітеративного обчислення. Роботу цих задач організовано за допомогою ForkJoinPool, що дозволило ефективно задіяти ресурси процесора.

Нижче представлений код

```
task_8_2 >  ArraySumForkJoin.java
1  import java.util.concurrent.ForkJoinPool;
2  import java.util.concurrent.RecursiveTask;
3  import java.util.Random;
4
5  public class ArraySumForkJoin {
6
7      // sum calculating
8      static class SumTask extends RecursiveTask<Long> {
9          private static final int THRESHOLD = 20; // treshold
10         private final int[] array;
11         private final int start;
12         private final int end;
13
14         public SumTask(int[] array, int start, int end) {
15             this.array = array;
16             this.start = start;
17             this.end = end;
18         }
19     }
```

```

19
20     @Override
21     protected Long compute() {
22         // if elements amount < 20, calculate sum
23         if (end - start <= THRESHOLD) {
24             long sum = 0;
25             for (int i = start; i < end; i++) {
26                 sum += array[i];
27             }
28             return sum;
29         } else {
30             // devide array into 2 parts
31             int mid = start + (end - start) / 2;
32             SumTask leftTask = new SumTask(array, start, mid);
33             SumTask rightTask = new SumTask(array, mid, end);

34             // start tasks in paralell
35             leftTask.fork();
36             long rightResult = rightTask.compute();
37             long leftResult = leftTask.join();

38             return leftResult + rightResult;
39         }
40     }
41 }
42 }
43 }
44 }
```

```

45     public static void main(String[] args) {
46         int size = 1_000_000; //
47         int[] array = new int[size];
48         Random random = new Random();

49         // initiate array with random numbers
50         for (int i = 0; i < size; i++) {
51             array[i] = random.nextInt(101);
52         }

53         ForkJoinPool pool = new ForkJoinPool();
54         long startTime = System.currentTimeMillis();

55         long totalSum = pool.invoke(new SumTask(array, 0, array.length));

56         long endTime = System.currentTimeMillis();

57         // results output
58         System.out.println("Сума елементів масиву: " + totalSum);
59         System.out.println("Час виконання (ForkJoin): " + (endTime - startTime) + " мс");

60         long checkSum = 0;
61         for (int x : array) checkSum += x;
62         System.out.println("Перевірочна сума (Single-thread): " + checkSum);
63     }
64 }
```

Результати компіляції та демонстрація роботи програми наведені нижче

```
● ruszlanapavliuk@POL02-0349 task_8_2 % java ArraySumForkJoin.java
Сума елементів масиву: 49956499
Час виконання (ForkJoin): 41 мс
Перевірочна сума (Single-thread): 49956499
```