**Лабораторная работа**

**Тема**: Работа с потоками в Java

**Задание**

* 1. Для задачи согласно индивидуальному варианту распараллелить на java двумя способами, как в примерах, добавив генерацию массива большей размерности случайными числами
  2. Засечь время работы. Сравнить работу программ по времени.
  3. Оформить отчет и защитить.

Индивидуальный вариант выбирается циклически, 18-ый выполняет первый и т.д.

**Примеры решения**

### ****Способ 1. Использование потоков с**** ExecutorService

ExecutorService позволяет управлять пулом потоков, давая вам более гибкий контроль над многозадачностью. Мы можем создать несколько потоков, каждый из которых будет искать максимум в части массива, а затем объединить результаты.

#### Пример с ExecutorService:

import java.util.concurrent.\*;

public class ParallelMaxFinderWithExecutor {

public static void main(String[] args) throws InterruptedException, ExecutionException {

// Исходный массив

int[] arr = {2, 5, 8, 1, 3, 9, 4, 7, 6, 11, 13, 19, 17, 12};

// Используем ExecutorService для параллельной обработки

ExecutorService executor = Executors.newFixedThreadPool(4); // Пул из 4 потоков

int numThreads = 4;

int length = arr.length;

int chunkSize = (int) Math.ceil((double) length / numThreads);

// Список для хранения результатов

List<Future<Integer>> futures = new ArrayList<>();

// Разбиваем массив на части и назначаем задачи потокам

for (int i = 0; i < numThreads; i++) {

final int start = i \* chunkSize;

final int end = Math.min((i + 1) \* chunkSize, length);

futures.add(executor.submit(() -> findMax(arr, start, end)));

}

// Ожидаем завершения всех задач и находим максимальное значение

int max = Integer.MIN\_VALUE;

for (Future<Integer> future : futures) {

max = Math.max(max, future.get());

}

executor.shutdown();

System.out.println("Максимальное число в массиве: " + max);

}

// Метод для нахождения максимума в подмассиве

private static int findMax(int[] arr, int start, int end) {

int max = arr[start];

for (int i = start + 1; i < end; i++) {

max = Math.max(max, arr[i]);

}

return max;

}

}

**Как работает:**

1. **ExecutorService:** создаем пул потоков с использованием Executors.newFixedThreadPool(), в котором можно задать количество потоков. Это позволяет распределить работу между несколькими потоками.
2. **Разбиение массива:** делим массив на несколько частей. Каждая часть передается в отдельный поток для поиска максимального элемента.
3. **Результат:** Потоки выполняются параллельно, каждый из них находит максимум в своем подмассиве. После этого объединяем результаты, используя Future.get(), и находим максимальное значение среди всех частей массива.

**Преимущества:**

* **Гибкость:** ExecutorService дает больше контроля над количеством потоков и позволяет управлять ими вручную.
* **Управление ресурсами:** ExecutorService эффективно управляет ресурсами, таким как пул потоков.

### ****Способ**** 2. ****Использование**** Streams ****(Параллельные потоки)****

Java Streams API поддерживает параллельное выполнение задач с использованием метода .parallel(). Это позволяет легко распараллелить задачу поиска максимума без явного создания потоков.

#### Пример с использованием параллельных потоков:

import java.util.Arrays;

public class ParallelMaxWithStreams {

public static void main(String[] args) {

// Исходный массив

int[] arr = {2, 5, 8, 1, 3, 9, 4, 7, 6, 11, 13, 19, 17, 12};

// Параллельный поток для нахождения максимума

int max = Arrays.stream(arr)

.parallel() // Использование параллельного потока

.max()

.orElseThrow(() -> new IllegalArgumentException("Массив не может быть пустым"));

System.out.println("Максимальное число в массиве: " + max);

}

}

### Как работает:

1. **Arrays.stream(arr)** создает поток данных из массива.
2. **parallel()** преобразует поток в параллельный, который будет автоматически использовать несколько потоков для обработки данных.
3. **max()** находит максимальное значение в потоке.
4. **orElseThrow()** гарантирует, что в случае пустого массива выбрасывается исключение.

### Преимущества:

* **Простота:** Код становится намного более простым и читаемым.
* **Автоматическое управление потоками:** Java сама управляет количеством потоков, которые будут использоваться.
* **Легкость интеграции:** Параллельные потоки в Java легко интегрируются в существующие решения.

**Варианты заданий**

1. Реализовать решение задачи в соответствии с вариантом.

**Варианты задач**

1. Найти произведения элементов каждой строки двумерной матрицы
2. Нахождение max и min в матрице.
3. Дана целочисленная прямоугольная матрица. Определить: максимум в каждой строке матрицы.
4. Дана целочисленная прямоугольная матрица. Определить: минимум в каждой строке матрицы.
5. Дана целочисленная прямоугольная матрица. Определить: максимум в каждом столбце матрицы.
6. Дана целочисленная прямоугольная матрица. Определить: минимум в каждом столбце матрицы.
7. Дана целочисленная прямоугольная матрица. Определить: номер столбца, в которой находится самая длинная серия одинаковых элементов.
8. Дана целочисленная прямоугольная матрица. Определить: количество строк, содержащих хотя бы один нулевой элемент.
9. Сформировав равновеликие прямоугольные матрицы А и В, построить результирующую матрицу С по правилу: если аij<bij, то сij=aij + bij ; если аij>bij, то сij= аij -bij .
10. Найти номер столбца двухмерного массива целых чисел, для которого среднеарифметическое значение его эле­ментов максимально.
11. Найти минимальный элемент среди максимальных элементов строк двухмерного массива целых чисел. Опре­делить номер строки и столбца для такого элемента.
12. Найти максимальный среди минимальных элементов столбцов двухмерного массива целых чисел. Определить номер строки и столбца для такого элемента.
13. Найти все неповторяющиеся элементы двухмерно­го массива целых чисел.
14. Дан двухмерный массив. Изменить его эле­менты следующим образом:

элементы нечетных строк переписать справа налево, элементы четных строк удвоить;

1. Рассчитать определенный интеграл по квадратурной формуле Симпсона



1. Рассчитать определенный интеграл по квадратурной формуле прямоугольников



1. Рассчитать определенный интеграл по квадратурной формуле трапеций