**МОЛДАВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет Математики и Информатики**

**Департамент Информатики**

Лабораторная работа № 5

Программирование на JAVA

Создание нитей процессов (Threads)

Проверил: профессор, др. Gheorghe Latu

Выполнил: Mamaliga Artur grupa I2302-1

Кишинев, 2024

# 1. Введение

Цель данной лабораторной работы – изучить работу с многопоточностью на языке программирования Java, а также освоить основные механизмы синхронизации нитей (потоков) для совместной работы над задачей.

# 2. Задание

Разработать программу, в которой два потока (нити) будут складывать элементы двух половин матрицы размером 4х4. Первый поток будет складывать первую половину (первые две строки), а второй поток будет складывать вторую половину (две последние строки).

# 3. Описание программы

Программа состоит из основного класса Program и класса MatrixSumThread. Класс MatrixSumThread наследует класс Thread и выполняет подсчёт суммы элементов определённого диапазона строк матрицы. Каждый поток обрабатывает свою часть матрицы, и по завершении работы потоков основной поток собирает результаты и выводит общую сумму элементов матрицы.

# 4. Код программы

public class Program {  
 public static void main(String[] args) {  
  
 var matrix = new int[][]{  
 {5, 5, 5, 5},  
 {5, 5, 5, 5},  
 {9, 10, 11, 12},  
 {13, 14, 15, 16}  
 };  
  
 MatrixSumThread thread1 = new MatrixSumThread(matrix, 0, 1); // Rows 0 and 1  
 MatrixSumThread thread2 = new MatrixSumThread(matrix, 2, 3); // Rows 2 and 3  
  
 thread1.start();  
 thread2.start();  
  
 try {  
 thread1.join();  
 thread2.join();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
  
 int totalSum = thread1.getSum() + thread2.getSum();  
 System.*out*.println("Total sum of the matrix: " + totalSum);  
 }  
}

MatrixSumThread.java  
public class MatrixSumThread extends Thread  
{  
 private int[][] matrix;  
 private int startRow;  
 private int endRow;  
 private int sum;  
  
 public MatrixSumThread(int[][] matrix, int startRow, int endRow) {  
 this.matrix = matrix;  
 this.startRow = startRow;  
 this.endRow = endRow;  
 }  
  
 @Override  
 public void run() {  
 sum = 0;  
 for (int i = startRow; i <= endRow; i++) {  
 for (int num : matrix[i]) {  
 sum += num;  
 }  
 }  
 System.*out*.println(Thread.*currentThread*().getName() + " sum: " + sum);  
 }  
 public int getSum() {  
 return sum;  
 }  
}

5. Результат работы программы

В результате выполнения программы на консоль выводится сумма элементов, посчитанная каждым потоком, а также общая сумма всех элементов матрицы.

## Пример вывода программы:

Thread-0 sum: 40  
Thread-1 sum: 100  
Total sum of the matrix: 140

# 6. Заключение

В ходе выполнения лабораторной работы были изучены основы многопоточности в языке Java, а также методы управления потоками с помощью классов Thread и методов join(). Программа успешно выполняет параллельный подсчет сумм элементов матрицы, что демонстрирует возможность разделения задач между потоками для повышения эффективности выполнения программы.