МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

**СЕРГИЕНКО ЛЕВ ЭДУАРДОВИЧ**

# Отчет по
# Лабораторная работа 1
# Разработка многопоточных приложений на языке Java

**Преподаватель**

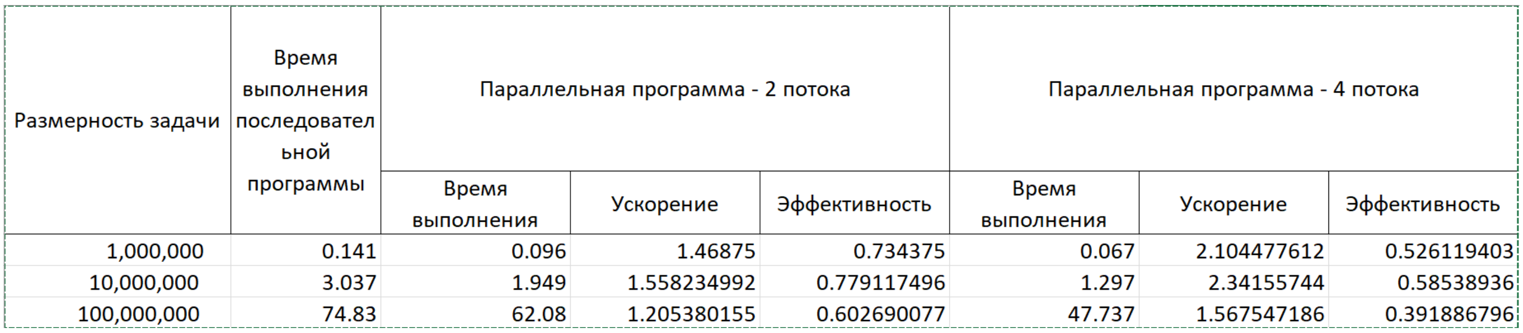
***Кондратьева О.М.***

**2024**

## Код программы

```
package com.example;

import java.util.Scanner;

/*
 * A program that starts several threads,each of which performs the
 * same computation.The user specifies the number of threads.The
 * point is to see that the threads finish in an indeterminate order.
 */

public class Main {

    private final static int MAX = 1_000_000_000;

    /*
     * When a
     * thread belonging to this class is run it will count the* number of
primes
     * between 2 and MAX.
     * It will print the result* to standard output, along with its id
number and
     * the elapsed* time between the start and the end of the computation.
     */

    private static class CountPrimesThread extends Thread {
        int id; // An id number for this thread; specified in the
constructor.
        int left, right;

        public CountPrimesThread(int id, int l, int r) {
            this.id = id;
            this.left = l;
            this.right = r;
        }

        public void run() {
            long startTime = System.currentTimeMillis();
            int count = countPrimes(left, right);
            long elapsedTime = System.currentTimeMillis() - startTime;
            System.out.println("Thread " + id + " counted " +
                    count + " primes in " + (elapsedTime / 1000.0) + "
seconds.");
        }
    }

    /*
     * Start several CountPrimesThreads.The number of threads, between 1
and 30, is
     * specified by the user.
     */

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int numberOfThreads = 0;

        while (numberOfThreads < 1 || numberOfThreads > 30) {
            System.out.print("How many threads do you want to use  (from 1
to 30) ?  ");
            numberOfThreads = scanner.nextInt();
            if (numberOfThreads < 1 || numberOfThreads > 30)
                System.out.println("Please enter a number between 1 and 30
!");
        }

        System.out.println("\nCreating " + numberOfThreads + "
prime-counting threads...");

        CountPrimesThread[] worker = new
CountPrimesThread[numberOfThreads];

        for (int i = 0; i < numberOfThreads; i++) {
            worker[i] = new CountPrimesThread(i, i * MAX / numberOfThreads,
(i + 1) * MAX / numberOfThreads);
        }

        for (int i = 0; i < numberOfThreads; i++) {
            worker[i].start();
        }

        System.out.println("Threads have been created and started.");
    }

    /*
     * Compute and return the number of prime numbers in the range min to
     * max,inclusive.
     */

    private static int countPrimes(int min, int max) {
        int count = 0;
        for (int i = min; i <= max; i++)
            if (isPrime(i))
                count++;
        return count;
    }

    /*
     * Test whether x is a prime number.
     * x is assumed to be greater than 1.
     */
    private static boolean isPrime(int x) {
        assert x > 1;
        int top = (int) Math.sqrt(x);
        for (int i = 2; i <= top; i++)
            if (x % i == 0)
                return false;
        return true;
    }

}
```

## Результат работы



| Размерность задачи | Время выполнения последовательной программы | Параллельная программа - 2 потока | | | Параллельная программа - 4 потока | | |
|---|---|---|---|---|---|---|---|
| | | Время выполнения | Ускорение | Эффективность | Время выполнения | Ускорение | Эффективность |
| 1,000,000 | 0.141 | 0.096 | 1.46875 | 0.734375 | 0.067 | 2.104477612 | 0.526119403 |
| 10,000,000 | 3.037 | 1.949 | 1.558234992 | 0.779117496 | 1.297 | 2.34155744 | 0.58538936 |
| 100,000,000 | 74.83 | 62.08 | 1.205380155 | 0.602690077 | 47.737 | 1.567547186 | 0.391886796 |