Министерство науки и высшего образования РФ
Пензенский государственный университет
Кафедра “Вычислительная техника”

Отчёт

по лабораторной работе №5
по курсу “Программирование на языке Java”
на тему “Многопоточность в Java”
Вариант 7

Выполнили студенты гр. 22ВВП2:
Кулахметов С.И.,
Гречихин П.П.,
Андреянов Я.И.

Приняли:
к.т.н., доцент Юрова О.В.
к.т.н., доцент Карамышева Н.С.

Пенза 2025

Цель работы

Научиться создавать многопоточные приложения с использованием стандартных средств языка Java.

Лабораторное задание

Модифицировать приложение из предыдущей лабораторной работы, реализовав вычисление определенного интеграла в семи дополнительных потоках, снимая нагрузку с основного потока и предотвращая "подвисание" графического интерфейса. Для распараллеливания вычислений необходимо реализовать интерфейс Runnable.

Ход выполнения работы

В лабораторной работе реализован класс TrapezoidCalculator, реализующий интерфейс Runnable. В рамках данного класса реализован метод run, выполняющий роль отдельного потока (рис. 1).

```
@Override
public void run() {
    result = trapezoidMethod(a, b, eps);
}

private double trapezoidMethod(double a, double b, double eps) {
    double sum = 0.0;
    double currentX = a;

    while (currentX < b) {
        double nextX = (currentX + eps < b) ? currentX + eps : b;
        sum += (f(currentX) + f(nextX)) * (nextX - currentX) / 2;
        currentX = nextX;
    }

    return sum;
}

private double f(double x) {
    return 1 / Math.log(x);
}
```

Рисунок 1 — Метод run класса TrapezoidCalculator

Для экземпляров класса и потоков созданы специальные массивы. Создание вычислительных потоков организовано в цикле (рис. 2).

```java
double intervalWidth = (b - a) / numThreads;  // Вычисление ширины интервала для каждого потока
List<TrapezoidCalculator> calculators = new ArrayList<>();  // Экземпляры класса TrapezoidCalculator
List<Thread> threads = new ArrayList<>(); // Потоки

for (int i = 0; i < numThreads; i++) {
  // Обработка интервалов
    double start = a + i * intervalWidth;
    double end = (i == numThreads - 1) ? b : start + intervalWidth;
    TrapezoidCalculator calculator = new TrapezoidCalculator(start, end, eps);
    calculators.add(calculator);

    Thread thread = new Thread(calculator);
    threads.add(thread);
    thread.start();
}
```

Рисунок 2 — Создание и организация вычислительных потоков

В качестве инструмента синхронизации использовался метод барьерной синхронизации join (рис. 3).

```java
// Ожидание завершения всех потоков
threads.forEach(thread -> {
  try {
    thread.join();
  } catch (InterruptedException e) {
    e.printStackTrace();
  }
});
```

Рисунок 3 — Синхрониация потоков

Результат работы программы представлен на рисунке 4.

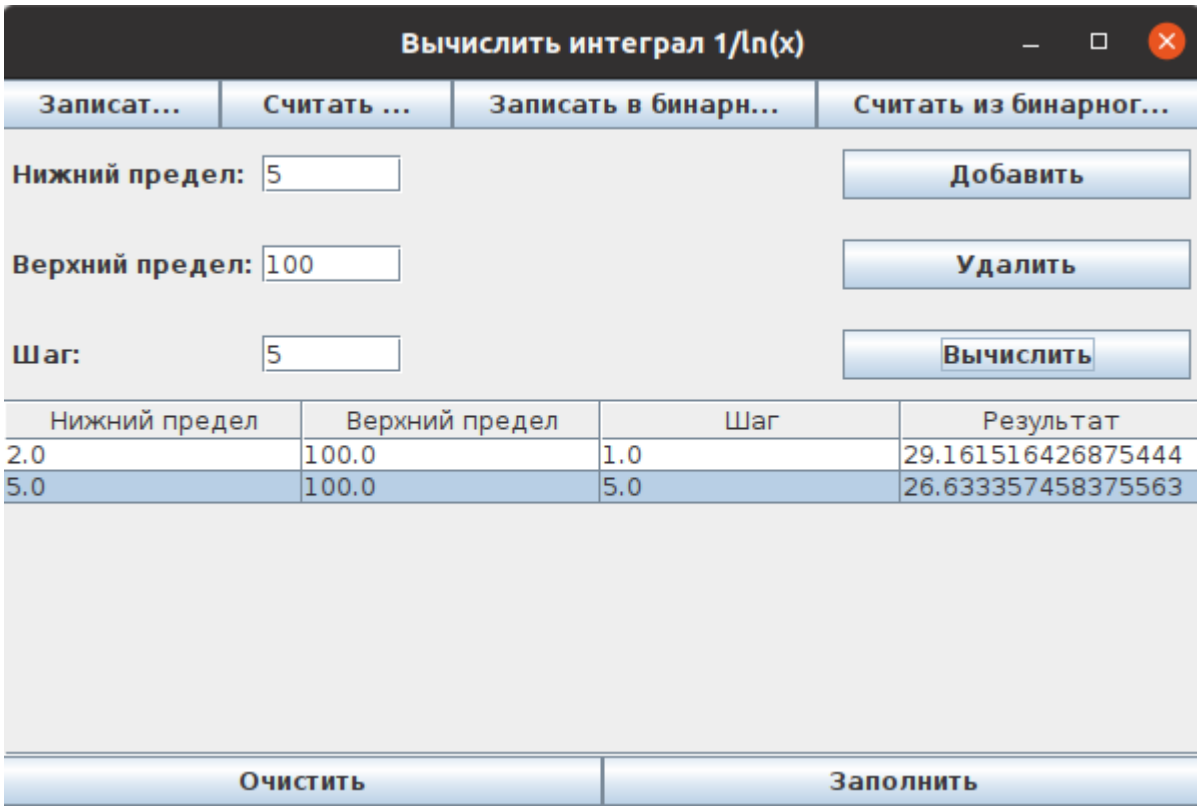

Рисунок 4 — Результат работы программы

Вывод

В ходе выполнения данной лабораторной работы были получены навыки организации многопоточных программ на языке Java. Изучен механизм синхронизации вычислительных потоков join.

Ссылка на репозиторий: https://github.com/KulakhmetovS/Java_Labs