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

Отчёт

по лабораторной работе №6
по курсу “Программирование на языке Java”
на тему “Сетевое взаимодействие 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 клиент-серверные приложения с использованием стандартных классов Java.

Лабораторное задание

Модифицировать приложение из предыдущей лабораторной работы, реализовав клиент-серверную архитектуру, обеспечивающую распределенное вычисление определенного интеграла на нескольких вычислительных узлах (клиентах) при этом каждый узел использует несколько нитей, как в предыдущей работе. Сервер не занимается вычислениями, а лишь реализует взаимодействие с пользователем и агрегацию результатов вычислений от клиентов. Для передачи данных использовать протокол UDP.

Ход выполнения работы

В лабораторной работе реализован класс UDPClient, отвечающий за получение интервала интегрирования и вычисления части интеграла. Клиент прослушивает порт 9XXX, который задаётся при помощи параметров командной строки, при запуске экземпляра клиента (рис. 1).

```
public static void main(String[] args) {
  final int BASE_PORT = 9000; // Базовый порт, на котором работает сервер

  int portOffset = 0; // Смещение порта по умолчанию
  if (args.length > 0) {
    portOffset = Integer.parseInt(args[0]); // Преобразование первого аргумента в целое число
  }
  int PORT = BASE_PORT + portOffset; // Вычисление итогового порта

    try (DatagramSocket socket = new DatagramSocket(PORT)) {  // Создание соединения
        System.out.println("Port listening...");
```

Рисунок 1 — Реализация распределения портов между клиентами

Сборка jar-архива клиента происходит при помощи команд (рис. 2):

```
javac Runnable/TrapezoidCalculator.java UDPClient.java
jar cfm UDPClient.jar MANIFEST.MF UDPClient.class Runnable/TrapezoidCalculator.class
```

```
javac Runnable/TrapezoidCalculator.java UDPClient.java
jar cfm UDPClient.jar MANIFEST.MF UDPClient.class Runnable/TrapezoidCalculator.class
```

Рисунок 2 — Сборка jar-архива клиента

Содержимое файла MANIFEST.MF: `Main-Class: UDPClient`

В основной программе в классе Main создаются отдельные процессы, которые запускают экземпляры клиентов с нужными параметрами (рис. 3).

```java
for (int i = 0; i < NUM_CLIENTS; i++) {
    try {
        ProcessBuilder processBuilder = new ProcessBuilder("java", "-jar", "/home/sabir/Documents/Java/UDP/UDPClient.jar", String.valueOf(i), "UDPClient");
        processBuilder.inheritIO(); // Для перенаправления ввода/вывода
        Process process = processBuilder.start(); // Запуск клиента
        processes.add(process); // Сохранение процесса в список
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Рисунок 3 — Запуск экземпляров клиентов

После завершения работы программы процессы с экземплярами клиентов завершаются (рис. 4).

Результат работы программы представлен на рисунке 4.

```java
Runtime.getRuntime().addShutdownHook(new Thread(() -> {
    for (Process process : processes) {
        process.destroy(); // Завершение каждого процесса
    }
}));
```

Рисунок 4 — Механизм завершения процессов

Для вычисления интеграла его границы разбиваются на 7 интервалов, которые передаются в запущенные экземпляры клиентов (рис. 5), после чего сервер агрегирует полученые значения и выводит результат (рис. 6).

```java
// Отправка данных на все порты и получение результатов
for (int port : SERVER_PORT) {
    double start = a + i * intervalWidth;
    double end = (i == 7 - 1) ? b : start + intervalWidth;

    double[] data = {start, end, eps};
    StringBuilder sb = new StringBuilder();

    for (double value : data) {
        sb.append(String.format("%f ", value)); // Формирование строки данных
    }

    String dataStr = sb.toString();
    byte[] sendBuffer = dataStr.getBytes();

    // Отправка данных на сервер
    InetAddress serverAddress = InetAddress.getByName(SERVER_ADDRESS);
    DatagramPacket sendPacket = new DatagramPacket(sendBuffer, sendBuffer.length, serverAddress, port);
    socket.send(sendPacket);
    System.out.println("Data sended to port " + port + ": " + dataStr);
    i++;
}
```

Рисунок 5 — Отправка данных в экземпляры программы

```java
for (int port : SERVER_PORT) {
  // Получение ответа от сервера
  byte[] receiveBuffer = new byte[1024];
  DatagramPacket receivePacket = new DatagramPacket(receiveBuffer, receiveBuffer.length);
  socket.receive(receivePacket);

  // Преобразование полученных данных в строку
  String response = new String(receivePacket.getData(), 0, receivePacket.getLength());
  double result = Double.parseDouble(response);
  totalSum += result; // Суммирование результата
}
```

Рисунок 6 — Получение обработанных данных с клиентов

Результат работы программы представлен на иллюстрациях 7 и 8.

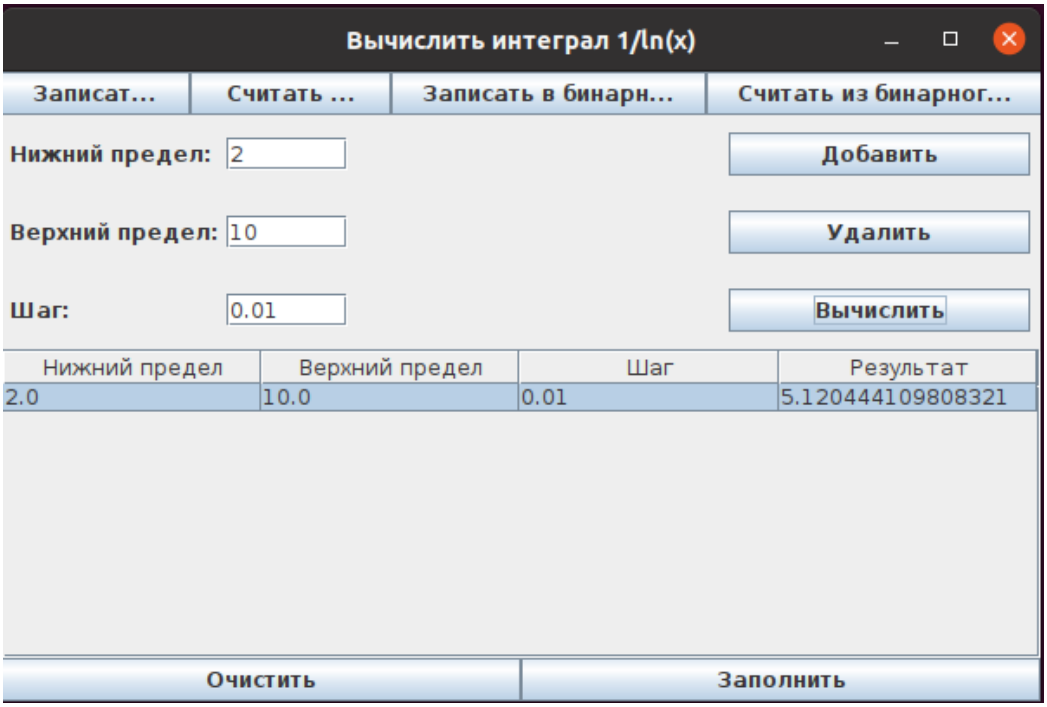

Рисунок 7 — Рассчитанный интеграл

```
sabir@sabir-Modern-15-B11M:~/Documents/Java/Lab6$ java Main
Port listening...
Port listening...
Port listening...
Port listening...
Port listening...
Port listening...
Port listening...
Data sended to port 9000: 2.000000 3.142857 0.010000
Data sended to port 9001: 3.142857 4.285714 0.010000
Data sended to port 9002: 4.285714 5.428571 0.010000
Data sended to port 9003: 5.428571 6.571429 0.010000
Data sended to port 9004: 6.571429 7.714286 0.010000
Data sended to port 9005: 7.714286 8.857143 0.010000
Data sended to port 9006: 8.857143 10.000000 0.010000
Sended: 1.2457671688990575
Sended: 0.5408729131146865
Sended: 0.6389885067406841
Sended: 0.7255425802702631
Sended: 0.8777402175309463
Sended: 0.5819166601823242
Sended: 0.5096160630703581
```

Рисунок 8 — Логи сервера

При запуске основной програмы запускаются экземпляры клиентов, осуществляющие вычисления (рис. 9). Как только пользователь завершает работу с основной программой, она завершает работу процессов, в которых запущены экземпляры клиентов (рис. 10).

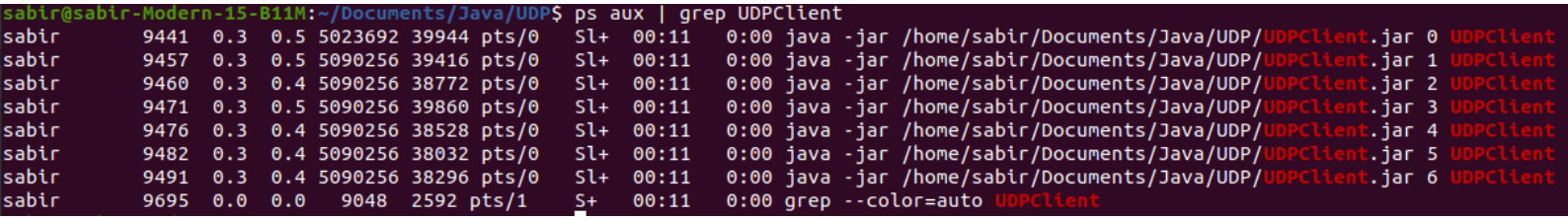

Рисунок 9 — Запущенные экземпляры клиентов

Рисунок 10 — Завершена работа клиентов

Вывод

В ходе выполнения данной лабораторной работы были получены навыки организации сетевого взаимодействия программ на языке Java посредствам протокола передачи данных UDP. Изучен инструмент сборки jar-архивов программ и их выполение при помощи процесса, созданного в другой java-программе.

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