МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«КРЫМСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ им. В. И. ВЕРНАДСКОГО»

ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ

Кафедра компьютерной инженерии и моделирования

**«Распределенная считалка»**

Лабораторная работа

по дисциплине «Методы распределенных информационных систем»

студента 4 курса группы ПИ-182(2)

Змитрович Никита Сергеевич

направления подготовки 09.03.04 «ПРОГРАММНАЯ ИНЖЕНЕРИЯ»

|  |  |  |
| --- | --- | --- |
| Проверил: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_  (оценка)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | Биленко Г.Р. |

Симферополь, 2021

Оглавление

[Цели: 3](#_Toc90370369)

[Ход работы 3](#_Toc90370370)

[1. Формат сообщений 3](#_Toc90370371)

[2. Клиент\сервер 4](#_Toc90370372)

[3. Демонстрация работы 6](#_Toc90370373)

[Исключительные ситуации 7](#_Toc90370374)

[Вывод 7](#_Toc90370375)

**Лабораторная работа №1**

# Цели:

* Написание простейшего клиент-серверного приложения на языке программирования Java

# Ход работы

## 1. Формат сообщений

Формат следующий:

Rq\Rp: number\_of\_elements [element, element, element]

Rq/Rp – сокращение от слов Request\Response соответственно

number\_of\_elements – количество элементов, которое хочет запросить клиент, не пустое только в случае, если сообщение начинается с префикса Rq

[element, element, …] – передающиеся данные, не пустое только в случае, если сообщение начинается с префикаса Rp

1. public record ParseInfo(
2. String type,
3. List<Integer> values,
4. Integer numberOfValues
5. ) {}

 Рис. 1 – POJO класс

1. public class Parser {
3. public static final String REQUEST\_PREFIX = "Rq";
4. public static final String RESPONSE\_PREFIX = "Rp";
6. public static ParseInfo deserialize(String message) {
7. return switch (message.substring(0, message.indexOf(':'))) {
8. case REQUEST\_PREFIX -> deserializeRequest(message);
9. case RESPONSE\_PREFIX -> deserializeResponse(message);
10. default -> throw new IllegalArgumentException("Illegal message prefix");
11. };
13. }
15. private static ParseInfo deserializeRequest(String response) {
16. return new ParseInfo(
17. response.substring(0, response.indexOf(':')),
18. Collections.emptyList(),
19. Integer.valueOf(response.substring(response.indexOf(':') + 1))
20. );
21. }
23. private static ParseInfo deserializeResponse(String response) {
24. return new ParseInfo(
25. response.substring(0, response.indexOf(':')),
26. Arrays.stream(
27. response.substring(response.indexOf('[') + 1, response.indexOf(']')).split(",")
28. ).map(Integer::valueOf).toList(),
29. 0
30. );
31. }
33. public static String serialize(ParseInfo info) {
34. var builder = new StringBuilder();
36. switch (info.type()) {
37. case RESPONSE\_PREFIX -> serializeResponse(info, builder);
38. case REQUEST\_PREFIX -> serializeRequest(info, builder);
39. default -> throw new IllegalArgumentException("Illegal message prefix");
40. }
42. return builder.toString();
43. }
45. private static void serializeResponse(ParseInfo info, StringBuilder builder) {
46. builder.append(info.type());
47. builder.append(':');
48. serializeList(info, builder);
49. }
51. private static void serializeRequest(ParseInfo info, StringBuilder builder) {
52. builder.append(info.type());
53. builder.append(':');
54. builder.append(info.numberOfValues());
55. }
57. private static void serializeList(ParseInfo info, StringBuilder builder) {
59. builder.append('[');
61. for (int i = 0; i < info.values().size() - 1; i++) {
62. builder.append(info.values().get(i));
63. builder.append(',');
64. }
66. builder.append(info.values().get(info.values().size() - 1));
68. builder.append(']');
69. }
71. }

Рис. 2 – Сериализатор\десериализатор сообщений собственного формата

## 2. Клиент\сервер

1. public class Client {  
     
    private final InputStream in;  
    private final OutputStream out;  
    private final Socket socket;  
     
    private Client(Socket sock) throws IOException {  
    socket = sock;  
    in = sock.getInputStream();  
    out = sock.getOutputStream();  
    }  
     
    public static Client create(Socket sock) throws IOException {  
    return new Client(sock);  
    }  
     
    public void send(ParseInfo info) throws IOException {  
    out.write(Parser.serialize(info).getBytes(StandardCharsets.UTF\_8));  
    }  
     
    public ParseInfo receive() throws IOException {  
    return Parser.deserialize(readAllBytes(in));  
    }  
     
    public void close() throws IOException {  
    socket.close();  
    }  
     
    private String readAllBytes(InputStream inputStream) throws IOException {  
    var baos = new ByteArrayOutputStream();  
    byte[] buffer = new byte[512];  
     
    var n = inputStream.read(buffer, 0, 512);  
    baos.write(buffer);  
     
    return baos.toString(Charset.defaultCharset());  
    }  
   }

Рис 3 — Клиентская часть

1. public class Server {  
     
    private ServerSocket serverSocket;  
     
    public void start(int port) throws IOException {  
    serverSocket = new ServerSocket(port);  
    while (true)  
    new ClientHandler(serverSocket.accept()).start();  
    }  
     
    private static class ClientHandler extends Thread {  
     
    private final Socket socket;  
    private final InputStream in;  
    private final OutputStream out;  
     
    public ClientHandler(Socket socket) throws IOException {  
    this.socket = socket;  
    this.in = socket.getInputStream();  
    this.out = socket.getOutputStream();  
    }  
     
    @Override public void interrupt() {  
    super.interrupt();  
    try {  
    socket.close();  
    in.close();  
    out.close();  
    } catch (IOException e) {  
    e.printStackTrace();  
    }  
    }  
     
    @Override public void run() {  
    try {  
    var message = Parser.deserialize(readAllBytes(in));  
    switch (message.type()) {  
    case Parser.REQUEST\_PREFIX -> {  
    var file = new BufferedReader(new FileReader("fact.txt"));  
    List<Integer> values = new ArrayList<>();  
    for (int i = 0; i < message.numberOfValues(); i++) {  
    values.add(Integer.valueOf(file.readLine()));  
    }  
     
    out.write(  
    Parser.serialize(new ParseInfo(Parser.RESPONSE\_PREFIX, values, 0)).getBytes(StandardCharsets.UTF\_8)  
    );  
    file.close();  
    }  
     
    case Parser.RESPONSE\_PREFIX -> {  
    var file = new PrintWriter(new FileOutputStream("answer.txt"));  
    message.values().forEach(file::println);  
    file.close();  
    }  
    }  
    } catch (IOException e) {  
    e.printStackTrace();  
    }  
    }  
     
    private static String readAllBytes(InputStream inputStream) throws IOException {  
    var baos = new ByteArrayOutputStream();  
    byte[] buffer = new byte[512];  
     
    var n = inputStream.read(buffer, 0, 512);  
    baos.write(buffer);  
     
    return baos.toString(Charset.defaultCharset()).trim();  
    }  
    }  
   }

Рис 4 — Серверная часть

## 3. Демонстрация работы

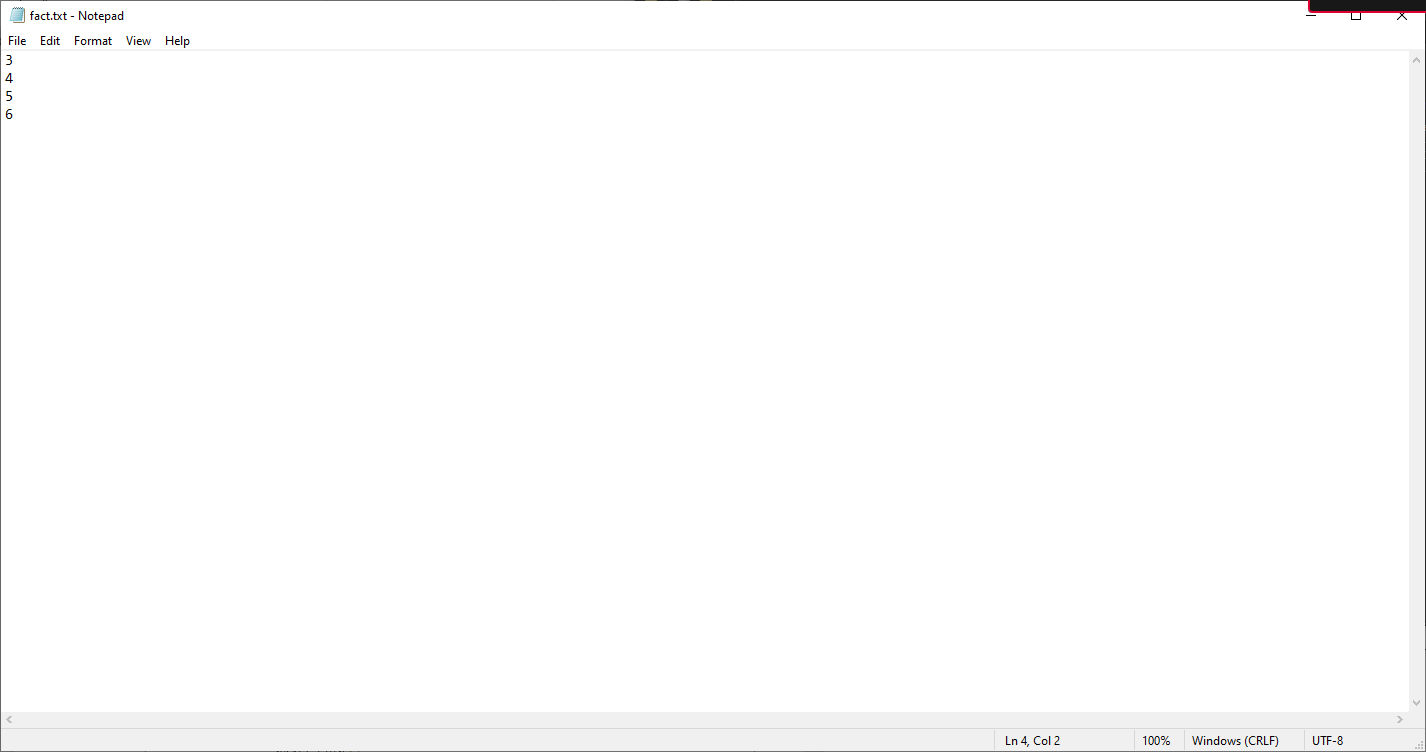


Рис 5 — Исходные данные

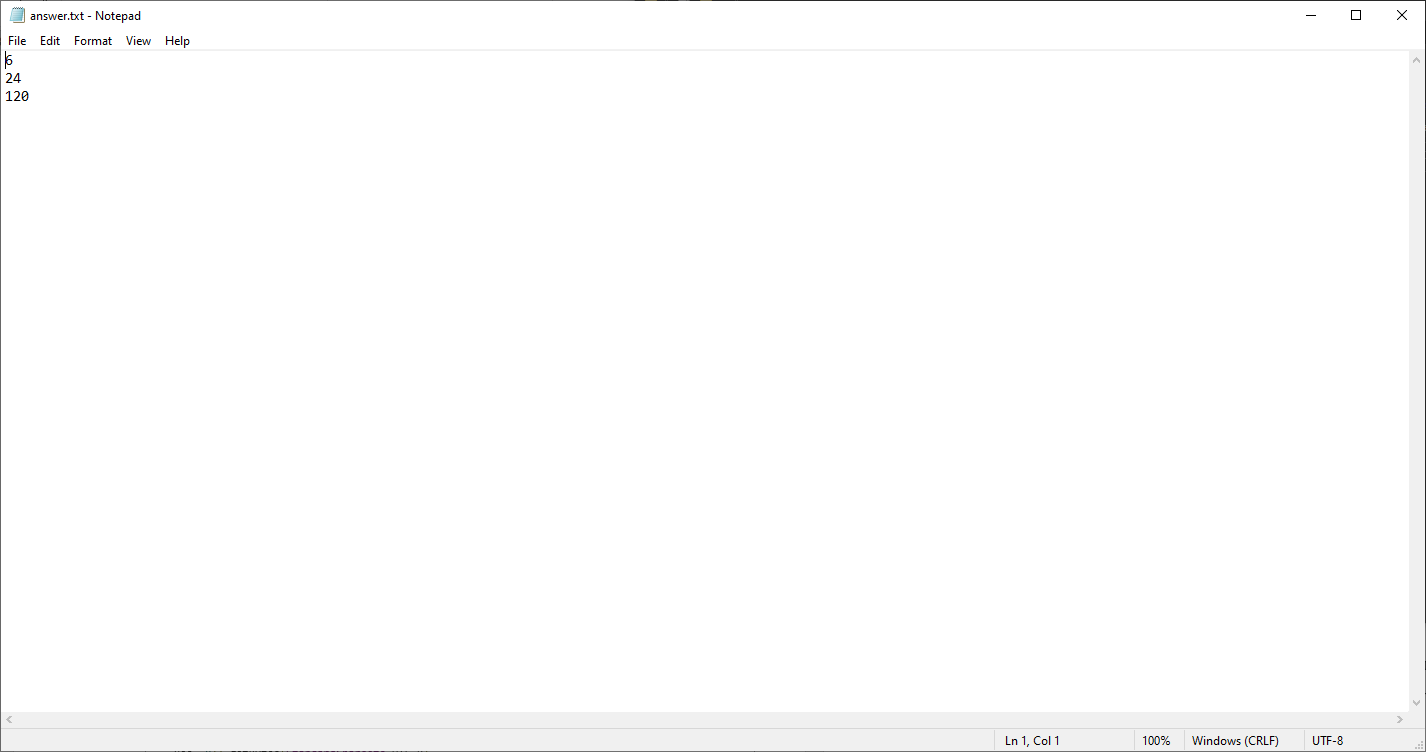


Рис 6 — Результат

## Исключительные ситуации

* Неверный формат сообщений
  + Неверный синтаксис сообщения
  + Длина сообщения превышает 512 байт
  + Не целочисленные данные
  + Слишком длинные числа
* Недостаточно ресурсов сервера для обработки запроса
  + Переполнение очереди запросов
  + Переполнение памяти
  + Отсутствие прав доступа к нужным файлам
* Проблемы со связью
  + Обрыв связи во время передачи сообщения
  + Искажения на линии связи

# Вывод

В ходе выполнения лабораторной работы закрепил знания синтаксиса основных конструкция языка Java, закрепил знания работы с сокетами и потоками данных в языке Java.