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: Биленко Г.Р.

_____
(оценка)

_____
(подпись, дата)

Симферополь, 2021

Оглавление

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 {
2.
3.      public static final String REQUEST_PREFIX = "Rq";
4.      public static final String RESPONSE_PREFIX = "Rp";
5.
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.         };
12.
13.     }
14.
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.     }
22.
```

```
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.in-
    dexOf(']')).split(",")
28.                 ).map(Integer::valueOf).toList(),
29.                 0
30.         );
31.     }
32.
33.     public static String serialize(ParseInfo info) {
34.         var builder = new StringBuilder();
35.
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.         }
41.
42.         return builder.toString();
43.     }
44.
45.     private static void serializeResponse(ParseInfo info, StringBuilder builder) {
46.         builder.append(info.type());
47.         builder.append(':');
48.         serializeList(info, builder);
49.     }
50.
51.     private static void serializeRequest(ParseInfo info, StringBuilder builder) {
52.         builder.append(info.type());
53.         builder.append(':');
54.         builder.append(info.numberOfValues());
55.     }
56.
57.     private static void serializeList(ParseInfo info, StringBuilder builder) {
58.
59.         builder.append('[');
60.
61.         for (int i = 0; i < info.values().size() - 1; i++) {
62.             builder.append(info.values().get(i));
63.             builder.append(',');
64.         }
65.
66.         builder.append(info.values().get(info.values().size() - 1));
67.
68.         builder.append(']');
69.     }
70.
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, val-
ues, 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

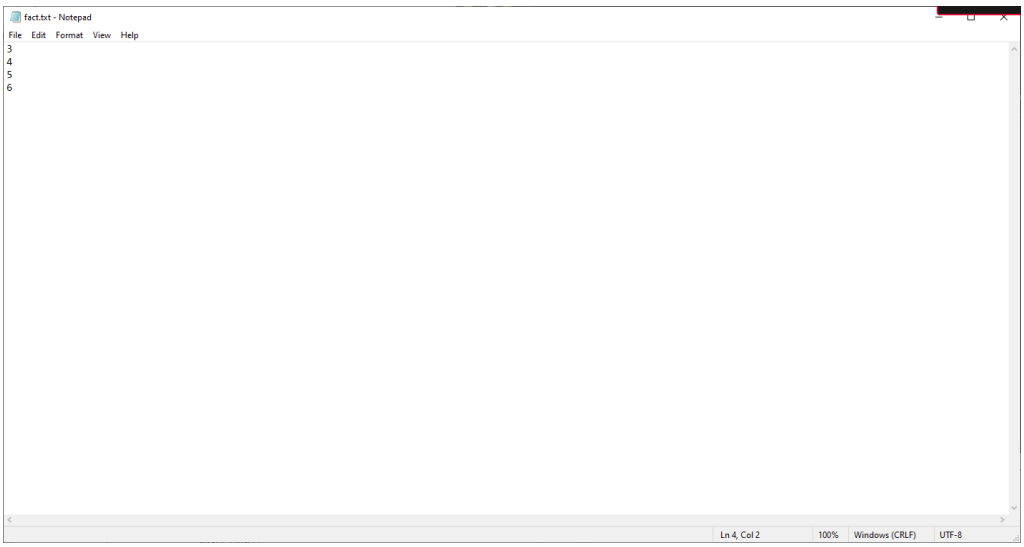

Рис 5 — Исходные данные

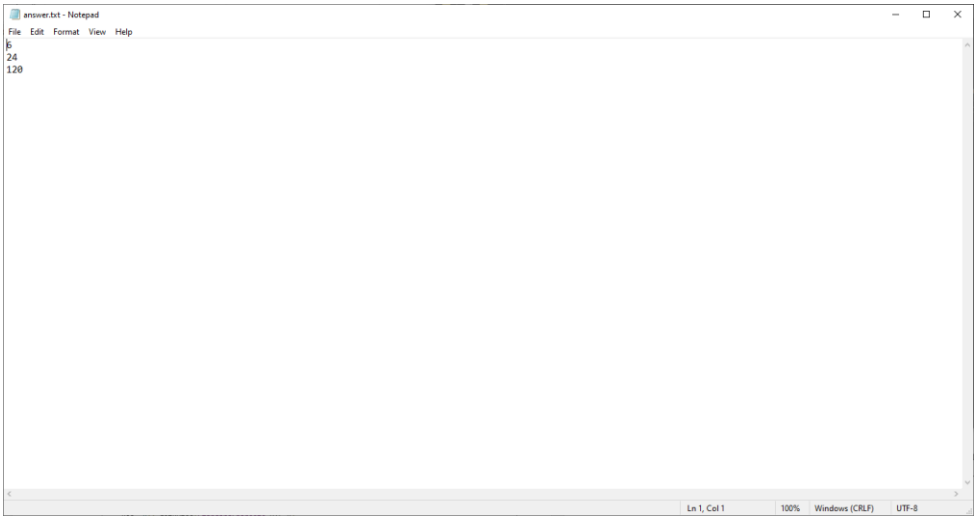

Рис 6 — Результат

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.