Санкт-Петербургский Национальный Исследовательский Университет
Информационных Технологий, Механики и Оптики

МФКТиУ, ФПИиКТ, СППО

Лабораторная работа №6

по дисциплине
«Программирование»

Вариант - 31121

Выполнил:
Анищенко Анатолий Алексеевич
Группа: Р3112

Санкт-Петербург
2019 г.

Задание: разделить программу из лабораторной работы №5 на клиентский и серверный модули. Серверный модуль должен осуществлять выполнение команд по управлению коллекцией. Клиентский модуль должен в интерактивном режиме считывать команды, передавать их для выполнения на сервер и выводить результаты выполнения. Команда import должна использовать файл из файловой системы клиента (содержимое файла передается на сервер), load и save - сервера.

Хранящиеся в коллекции объекты должны иметь следующие характеристики:

- имя, название или аналогичный текстовый идентификатор;
- размер или аналогичный числовой параметр;
- характеристику, определяющую местоположение объекта на плоскости/в пространстве;
- время/дату рождения/создания объекта.

Если аналогичные характеристики уже есть, добавлять их не нужно.

Необходимо выполнить следующие требования:

- Коллекцию из ЛР №5 заменить на ее потокобезопасный аналог.
- Операции обработки объектов коллекции должны быть реализованы с помощью Stream API с использованием лямбда-выражений.
- Объекты между клиентом и сервером должны передаваться в сериализованном виде.
- Объекты в коллекции, передаваемой клиенту, должны быть отсортированы по названию.
- Получив запрос, сервер должен создавать отдельный поток, который должен формировать и отправлять ответ клиенту.
- Клиент должен корректно обрабатывать временную недоступность сервера.
- Обмен данными между клиентом и сервером должен осуществляться по протоколу TCP.
- На стороне сервера должен использоваться потоки ввода-вывода а на стороне клиента - сетевой канал.

Отчёт по работе должен содержать:

- Текст задания.
- Диаграмма классов разработанной программы (как клиентского, так и серверного приложения).
- Исходный код программы.
- Выводы по работе.

Код программы:

https://github.com/AnatoliiAnishchenko/ITMO/tree/master/Programming/Lab6

Client.java

```
1 package client;
2
3 import commands.CommandDescriptor;
4 import moominClasses.Moomin;
5 import server.Response;
6 import java.io.*;
7 import java.net.InetSocketAddress;
8 import java.net.Socket;
9 import java.nio.ByteBuffer;
10 import java.nio.channels.SocketChannel;
11 import java.util.Scanner;
```

```java
import java.util.Vector;

public class Client {
    private SocketChannel socket;
    private Scanner scanner;
    private boolean working;
    private Vector<Moomin> moomins;
    private final long WAITING_TIME = 10000;

    public Client(String serverAddress, int port) throws IOException {
        //Set working flag
        working = true;

        //Create socket
        socket = SocketChannel.open(new InetSocketAddress(serverAddress, port));

        scanner = new Scanner(System.in);
        moomins = new Vector<>();
        doCommand("connect");
    }

    private void work() throws IOException {
        if (working) {
            System.out.println("Client is ready to work.");
        } else {
            System.out.println("Client can't work.");
        }

        while (working){
            System.out.println();
            System.out.print("Your command: ");
            String stringIn = scanner.nextLine().trim();
            System.out.println();

            doCommand(stringIn);
        }

        System.out.println("Session end.");
    }

    private byte[] createRequest(String description) throws IOException {
        byte[] sending;
        CommandDescriptor command = new CommandDescriptor(description);

        switch (command.getNAME()) {
            case "exit":
                if (command.getARGS_COUNT() == 0) {
                    working = false;
                }
                break;

            case "import":
                if (command.getARGS_COUNT() == 1) {
                    char[] buf = new char[1024];

                    try (FileReader fr = new FileReader(command.getArguments())) {
                        fr.read(buf);
                        String json = String.valueOf(buf);
                        command.setArguments(json);
                    } catch (FileNotFoundException e){
                        System.err.println("File not found!");
                        command = null;
```

```
                    }
                }
                break;

            case "my":
                moomins.forEach(System.out::println);
                if (moomins.size() == 0) {
                    System.out.println("Collection is empty.");
                }
                command = null;
                break;
        }

        if (command == null) {
            return null;
        }

        try (ByteArrayOutputStream baos = new ByteArrayOutputStream();
            ObjectOutputStream oos = new ObjectOutputStream(baos)){

            oos.writeObject(command);
            oos.flush();
            sending = baos.toByteArray();

            return sending;
        } catch (IOException e) {
            throw e;
        }
    }

    public void doCommand(String command) throws IOException {
        byte[] byteRequest = createRequest(command);

        //If we need to send the command
        if (byteRequest != null) {
            ByteBuffer request = ByteBuffer.wrap(byteRequest);

            while(request.hasRemaining()) {
                socket.write(request);
            }

            ByteBuffer respBuf = ByteBuffer.allocate(4096);

            socket.read(respBuf);

            byte[] byteResponse = respBuf.array();

            try (ByteArrayInputStream bais = new
ByteArrayInputStream(byteResponse);
                ObjectInputStream ois = new ObjectInputStream(bais)) {

                Response response = (Response) ois.readObject();

                System.out.println(response.getDoings());
                moomins = response.getMoomins();
            } catch (IOException | ClassNotFoundException e) {
                e.printStackTrace();
            }
        }
    }

    public static void main(String[] args) {
```

```
        //We need to get server port and address
        String address;
        int port;

        //By arguments
        if (args.length == 2) {
            address = args[0];
            port = Integer.valueOf(args[1]);
        } else {
            //By input stream
            Scanner scanner = new Scanner(System.in);

            System.out.print("Address: ");
            address=scanner.nextLine();

            System.out.print("Port: ");
            port=scanner.nextInt();
        }
        try {
            //Creating the client.Client
            Client client = new Client(address, port);
            System.out.println("Welcome!!!");

            //Start working
            client.work();
        } catch (Exception e) {
            System.err.println("Something gone wrong. Please DEBUG!!!");
            e.printStackTrace();
        }
    }
}
```

Server.java

```
package server;

import commands.FileHandler;
import mainClasses.MoominManager;
import moominClasses.Moomin;

import java.io.IOException;
import java.net.InetAddress;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Scanner;
import java.util.Vector;

public class Server {
    private Vector<Moomin> moomins;
    private ServerSocket serverSocket;
    public Server(int port) throws IOException {
        serverSocket = new ServerSocket(port, 0, InetAddress.getLocalHost());

        MoominManager manager = new MoominManager();
        if (manager.getMoomins().addAll(FileHandler.readFile("save.json"))) {
            System.out.println("Collection changed.");
        } else {
            System.out.println("Collection didn't changed.");
        }

        moomins = manager.getMoomins();

        System.out.println("Server started");
```

```
30          System.out.println("IP: " + serverSocket.getLocalSocketAddress());
31      }
32
33      private void listen() throws IOException {
34          while (true) {
35              Socket socket = serverSocket.accept();
36              ClientThread clientThread = new ClientThread(socket, moomins);
37              //clientThread.start();
38              clientThread.run();
39          }
40      }
41
42      public static void main(String[] args) {
43          int port = 1234;
44
45          //By arguments
46          if (args.length > 0) {
47              port = Integer.valueOf(args[0]);
48          } else {
49              //By input stream
50              Scanner scanner = new Scanner(System.in);
51              System.out.print("Port: ");
52              port = scanner.nextInt();
53          }
54
55          try {
56              Server server = new Server(port);
57              server.listen();
58          } catch (IOException e) {
59              e.printStackTrace();
60          }
61      }
62  }
```

ClientThread.java

```
 1  package server;
 2
 3  import commands.Command;
 4  import commands.CommandDescriptor;
 5  import commands.CommandsManager;
 6  import commands.FileHandler;
 7  import mainClasses.MoominManager;
 8  import moominClasses.Moomin;
 9
10  import java.io.*;
11  import java.net.Socket;
12  import java.util.Vector;
13
14  public class ClientThread extends Thread {
15      private static CommandsManager manager;
16      private final MoominManager moominManager;
17      private Vector<Moomin> moomins;
18      private Socket socket;
19      private boolean isActive;
20
21      public ClientThread(Socket socket, Vector<Moomin> moomins) {
22          moominManager = new MoominManager();
23          moominManager.setMoomins(moomins);
24
25          isActive = true;
26
27          manager = moominManager.getCommandsManager();
```

```
28          manager.addCommand(
29                  new Command("connect", 0) {
30                      @Override
31                      public void execute() {
32                          manager.println("Connected");
33                      }
34                      @Override
35                      public void describe() {
36                          manager.println("Command to check connection.");
37                      }
38                  },
39                  new Command("load", 1) {
40                      @Override
41                      public void execute() {
42                          if
   (moominManager.getMoomins().addAll(FileHandler.readFile(getArguments())))
43                              manager.println("Коллекция изменена");
44                          else manager.println("Коллекция не изменилась");
45                      }
46
47                      @Override
48                      public void describe() {
49                          manager.println("Дополняет коллекцию элементами из файла с
   сервера.");
50                      }
51                  },
52                  new Command("wait", 0) {
53                      @Override
54                      public void execute() {
55                          long start = System.currentTimeMillis();
56
57                          try {
58                              Thread.sleep(100);
59                          } catch (InterruptedException e) {
60                              e.printStackTrace();
61                          }
62
63                          manager.println(Thread.currentThread().getId() + " wait
   for " + (System.currentTimeMillis() - start));
64                      }
65                      @Override
66                      public void describe() {
67                          manager.println("Make a delay.");
68                      }
69                  },
70                  new Command("import", -1) {
71                      @Override
72                      public void execute() {
73                          if
   (moominManager.getMoomins().addAll(moominManager.createMoomin(getArguments())))
74                              manager.println("Коллекция изменена");
75                          else manager.println("Коллекция не изменилась");
76                      }
77
78                      @Override
79                      public void describe() {
80                          manager.println("Дополняет коллекцию элементами из файла
   клиента.");
81                      }
82                  },
83                  new Command("update",0) {
84                      @Override
```

```
                    public void execute() {
                        manager.println("Collection was updated.");
                    }
                    @Override
                    public void describe() {
                        manager.println("Send collection to the client.");
                    }
                },
                new Command("exit",0) {
                    @Override
                    public void execute() {
                        manager.println("Good bye!");

                    }
                    @Override
                    public void describe() {
                        manager.println("Disconnect user from server.");
                    }
                });

        this.socket = socket;
        this.moomins = moomins;
    }

    public void run() {
        while(isActive) {
            byte[] request = new byte[4096];
            try {
                InputStream ins = socket.getInputStream();
                if (ins.read(request) < 0) {
                    continue;
                }
            } catch (IOException e) {
                e.printStackTrace();
            }

            answer(request);
        }
    }

    private void answer(byte[] request) {
        CommandDescriptor command;

        try (ByteArrayInputStream bais = new ByteArrayInputStream(request);
             ObjectInputStream ois = new ObjectInputStream(bais);
             ByteArrayOutputStream baos = new ByteArrayOutputStream();
             ObjectOutputStream oos = new ObjectOutputStream(baos);
             ByteArrayOutputStream bao = new ByteArrayOutputStream();
             PrintStream printStream = new PrintStream(bao)) {

            command = (CommandDescriptor) ois.readObject();

            manager.setPrintStream(printStream);

            if (command.getNAME().equals("exit")) {
                isActive = false;
            }

            manager.doCommand(command);

            printStream.flush();

```

```
147             String doings = new String(bao.toByteArray()).trim();
148 
149             synchronized (System.out) {
150                 System.out.println();
151                 System.out.println("Client's command: " + command.getNAME() + " "
    + command.getArguments());
152                 System.out.println();
153                 System.out.println(doings);
154             }
155 
156             Response response = new Response(doings, moominManager.getMoomins());
157 
158             oos.writeObject(response);
159             oos.flush();
160             try {
161                 OutputStream outs = socket.getOutputStream();
162                 outs.write(baos.toByteArray());
163             } catch (IOException e) {
164                 e.printStackTrace();
165             }
166         } catch (IOException | ClassNotFoundException e) {
167             e.printStackTrace();
168         }
169     }
170 
171     public static CommandsManager getManager() {
172         return manager;
173     }
174 }
```

Выводы: в процессе выполнения лабораторной работы были получены навыки работы с дженериками и потоками ввода/вывода.