Министерство науки и высшего образования РФ
ФГБОУ ВО «Омский государственный технический университет»
Кафедра «Автоматизированные системы обработки информации и управления»

ОТЧЁТ по лабораторной
работе №5
По дисциплине: Объектно-ориентированное программирование
студента Мархиля Фёдора группы ПИН-212

Пояснительная записка
Шифр работы От-2068998-43-ПИН-212-10 ПЗ
Специальность 09.03.04

Старший преподаватель А.А. Кабанов

Студент Ф. Е. Мархиль

Омск 2022

Задание
Сервер рассылает сообщения только тем клиентам, которые в настоящий момент находятся в on-line.

Цель работы
Реализовать сетевое приложение, использующее протоколы стека TCP/IP.

Ход работы

Server.java

```
import java.io.*;
import java.net.*;
import java.util.LinkedList;

class ServerStaff extends Thread {

    private Socket socket;
    private BufferedReader in; // поток чтения из сокета
    private BufferedWriter out; // поток записи в сокет
    private BufferedReader inputUser;
    private String text;


    public ServerStaff(Socket socket) throws IOException {
        this.socket = socket;
        inputUser = new BufferedReader(new InputStreamReader(System.in));
        in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
        out = new BufferedWriter(new
OutputStreamWriter(socket.getOutputStream()));
        new WriteNotification().start();
        start(); // вызываем run()
    }
    @Override
    public void run() {
        String word;
        try {
            // первое сообщение отправленное сюда - это никнейм
            word = in.readLine();
            try {
                out.write(word + "\n");
                out.flush();
            } catch (IOException ignored) {}

            try {
                while (true) {
                    word = in.readLine();
                    if(word.equals("stop")) {
                        this.downService();
                        break;
                    }
                    System.out.println("Сообщение: " + word);
                    for (ServerStaff vr : Server.serverList) {
                        vr.send(word); // отослать принятое сообщение с
привязанного клиента всем остальным влючая его
                    }
                }
```

```
            } catch (NullPointerException ignored) {}

        } catch (IOException e) {
            this.downService();
        }

    }


     //отсылка одного сообщения клиенту

    private void send(String msg) {
        try {
            out.write(msg + "\n");
            out.flush();
        } catch (IOException ignored) {}

    }
     //отсылка одного уведомления клиенту

    public class WriteNotification extends Thread {

        @Override
        public void run() {
            while (true) {
                String userWord;
                try {
                    userWord = inputUser.readLine(); // сообщения с консоли
                    out.write(userWord + "\n");
                    out.flush(); // чистим
                } catch (IOException e) {}

            }
        }
    }

    private void downService() {
        try {
            if(!socket.isClosed()) {
                socket.close();
                in.close();
                out.close();
                for (ServerStaff vr : Server.serverList) {
                    if(vr.equals(this)) vr.interrupt();
                    Server.serverList.remove(this);
                }
            }
        } catch (IOException ignored) {}
    }
}

public class Server {

    public static final int PORT = 8080;
    public static LinkedList<ServerStaff> serverList = new LinkedList<>();
    private String text;


    public static void main(String[] args) throws IOException {
        ServerSocket server = new ServerSocket(PORT);
        System.out.println("Server Started");
        try {
            while (true) {
```

```
                Socket socket = server.accept();
                try {
                    serverList.add(new ServerStaff(socket));
                } catch (IOException e) {

                    socket.close();
                }
            }

        } finally {
            server.close();
        }
    }
}
```

Client.java

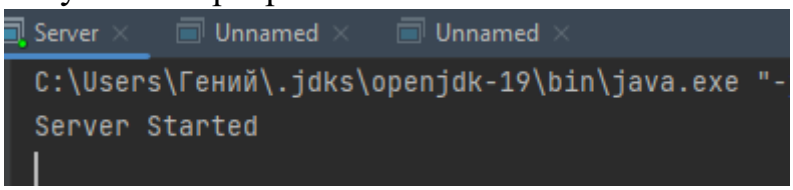
```
import java.net.*;
import java.io.*;
import java.text.SimpleDateFormat;
import java.util.Date;


class ClientStaff {

    private Socket socket; //сокет
    private BufferedReader in; // поток чтения из сокета
    private BufferedWriter out; // поток чтения в сокет
    private BufferedReader inputUser; // поток чтения с консоли
    private String addr; // ip адрес клиента
    private int port; // порт соединения
    private String nickname; // имя клиента
    private Date time;
    private String dtime;
    private SimpleDateFormat dt1;


    public ClientStaff(String addr, int port) {
        this.addr = addr;
        this.port = port;
        try {
            this.socket = new Socket(addr, port);
        } catch (IOException e) {
            System.err.println("Socket failed");
        }
        try {
            // потоки чтения из сокета / записи в сокет, и чтения с консоли
            inputUser = new BufferedReader(new InputStreamReader(System.in));
            in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
            out = new BufferedWriter(new
OutputStreamWriter(socket.getOutputStream()));
            this.pressNickname(); // перед началом необходимо спросит имя
            new ReadMsg().start(); // нить читающая сообщения из сокета в
бесконечном цикле
            new WriteMsg().start(); // нить пишущая сообщения в сокет
приходящие с консоли в бесконечном цикле
        } catch (IOException e) {

            ClientStaff.this.downService();
        }
    }
```

```
    private void pressNickname() { //вход и отсылка уведомления о входе
        System.out.print("Press your nick: ");
        try {
            nickname = inputUser.readLine();
            out.write("Hello " + nickname + "\n");
            out.flush();
        } catch (IOException ignored) {
        }

    }

    private void downService() {
        try {
            if (!socket.isClosed()) {
                socket.close();
                in.close();
                out.close();
            }
        } catch (IOException ignored) {}
    }

    // нить чтения сообщений с сервера
    private class ReadMsg extends Thread {
        @Override
        public void run() {

            String str;
            try {
                while (true) {
                    str = in.readLine(); // ждем сообщения с сервера
                    if (str.equals("stop")) {
                        ClientStaff.this.downService();
                        break; // выходим из цикла если пришло "stop"
                    }
                    System.out.println(str); // пишем сообщение с сервера на
консоль
                }
            } catch (IOException e) {
                ClientStaff.this.downService();
            }
        }
    }

    // нить отправляющая сообщения приходящие с консоли на сервер
    public class WriteMsg extends Thread {

        @Override
        public void run() {
            while (true) {
                String userWord;
                try {
                    userWord = inputUser.readLine(); // сообщения с консоли
                    if (userWord.equals("stop")) {
                        out.write("stop" + "\n");
                        ClientStaff.this.downService(); // харакири
                        break; // выходим из цикла если пришло "stop"
                    } else {
                        out.write(nickname + ": " + userWord + "\n"); //
отправляем на сервер
                    }
                    out.flush(); // чистим
```

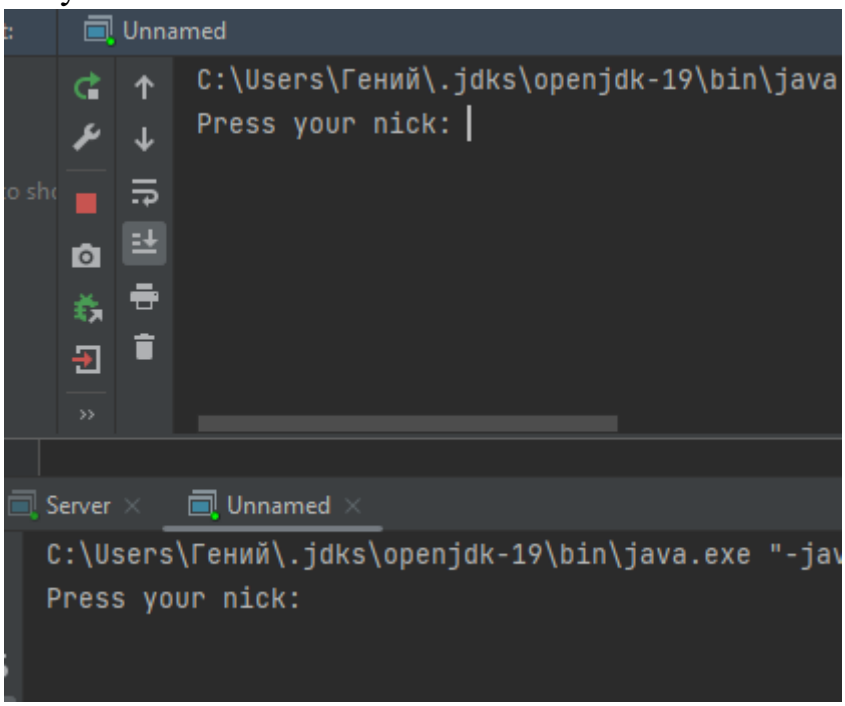
```
                } catch (IOException e) {
                    ClientStaff.this.downService(); // в случае исключения
тоже харакири

                }

            }
        }
    }
}

public class Client {

    public static String ipAddr = "localhost";
    public static int port = 8080;

    public static void main(String[] args) {
        new ClientStaff(ipAddr, port);
    }
}
```

Работа программы

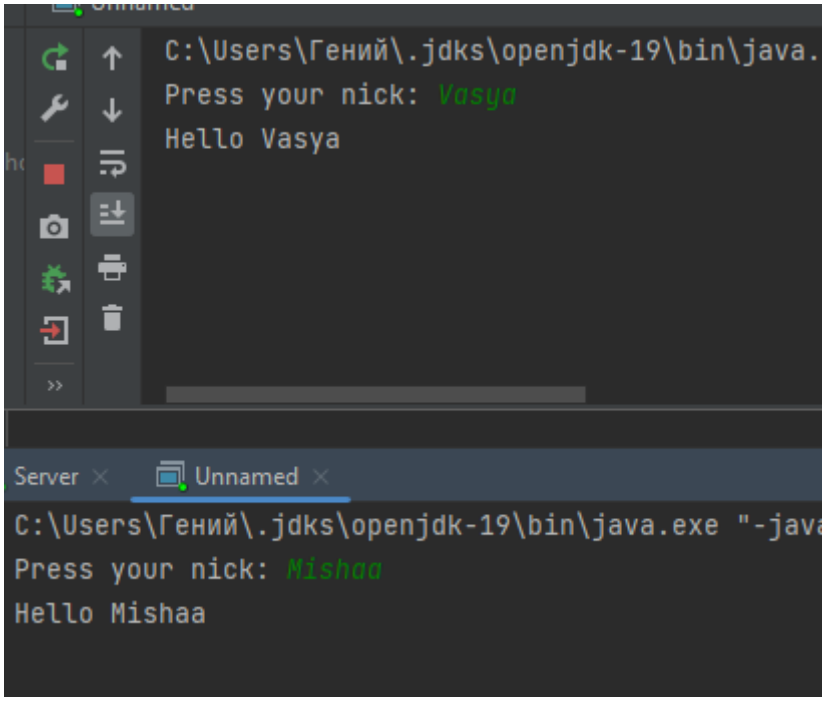
Запускаем сервер



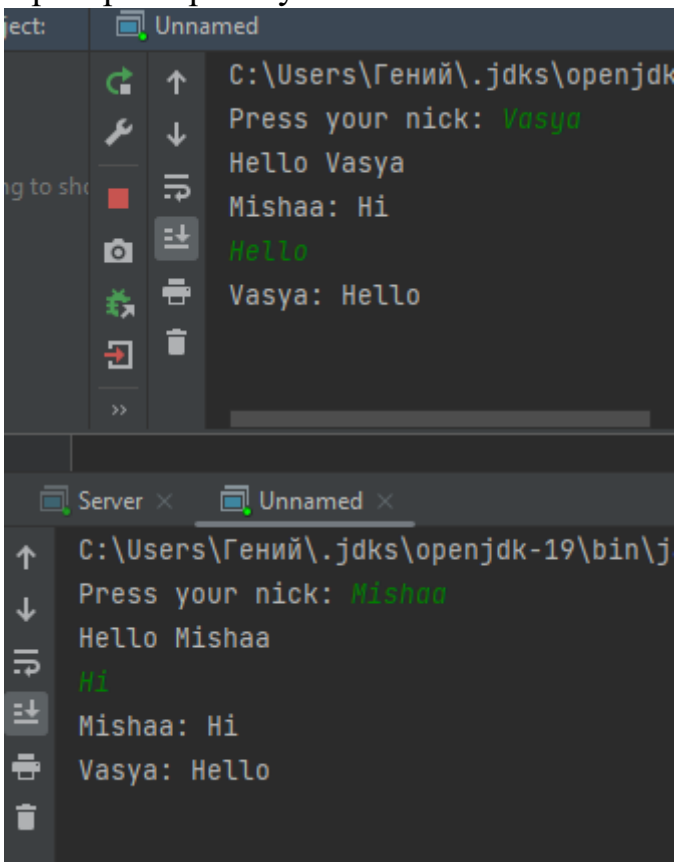
Запускаем клиенты



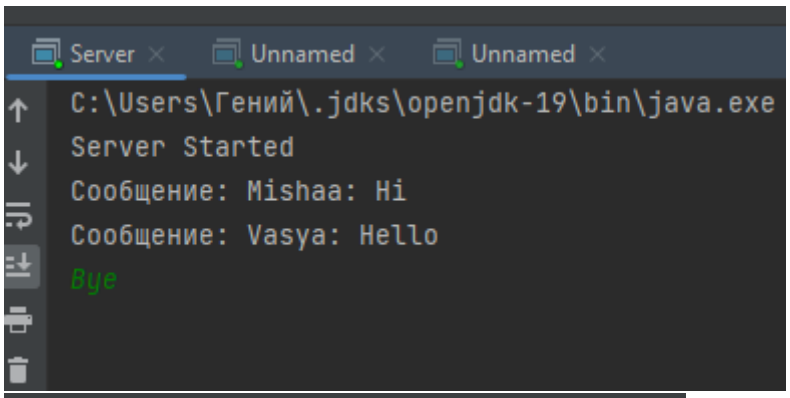
Вводим никнеймы

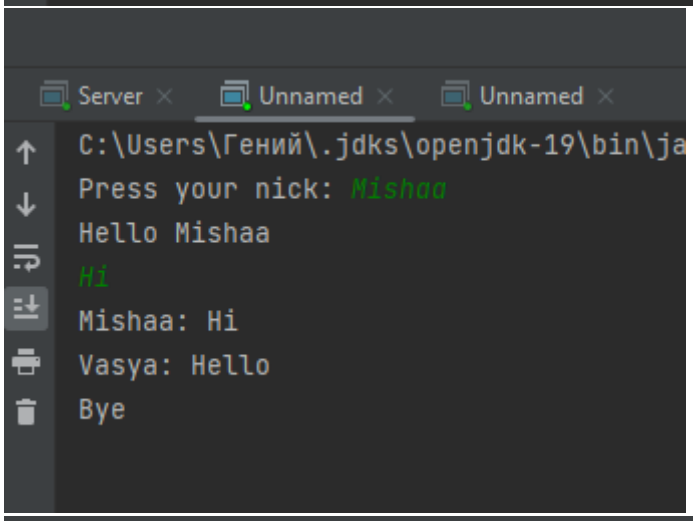


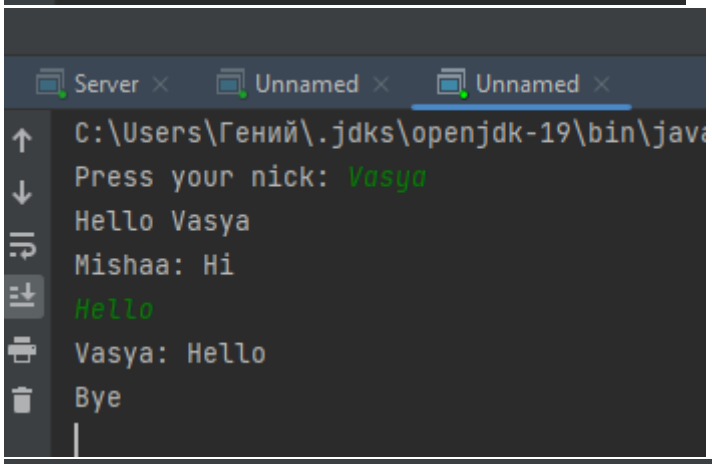
Проверяем работу чата

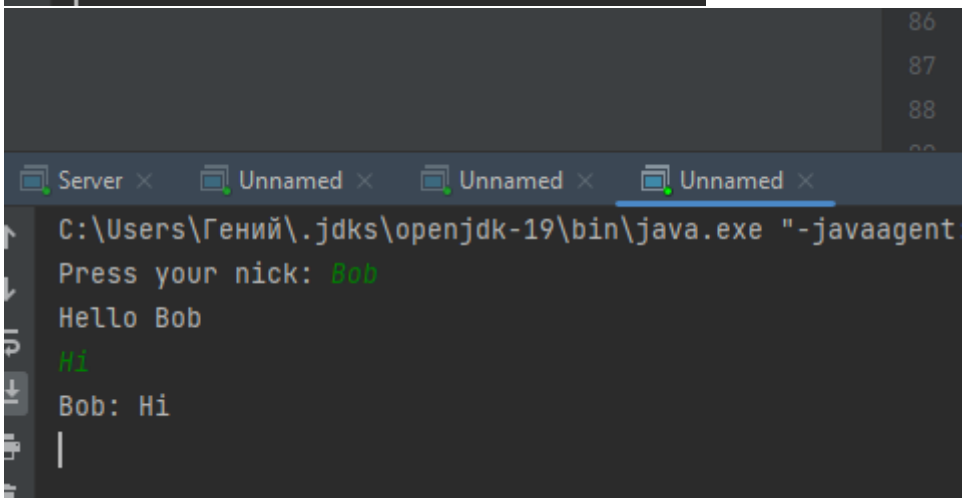


Мы ввели текст на сервере, и он был отправлен всем пользователям онлайн.


Server
Unnamed
Unnamed
C:\Users\Гений\.jdks\openjdk-19\bin\java.exe
Server Started
Сообщение: Mishaa: Hi
Сообщение: Vasya: Hello
Bye


Server
Unnamed
Unnamed
C:\Users\Гений\.jdks\openjdk-19\bin\ja
Press your nick: Mishaa
Hello Mishaa
Hi
Mishaa: Hi
Vasya: Hello
Bye


Server
Unnamed
Unnamed
C:\Users\Гений\.jdks\openjdk-19\bin\java
Press your nick: Vasya
Hello Vasya
Mishaa: Hi
Hello
Vasya: Hello
Bye


Server
Unnamed
Unnamed
Unnamed
C:\Users\Гений\.jdks\openjdk-19\bin\java.exe "-javaagent
Press your nick: Bob
Hello Bob
Hi
Bob: Hi

Вывод

Во время выполнения работы мною было реализовано сетевое приложение, использующее протоколы стека TCP/IP, и выполнено задание. Суть задания заключалась в создании чата, с рассылкой сообщений с сервера только онлайн пользователям.