Министерство образования Республики Беларусь

Министерство науки и высшего образования Российской Федерации

МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Автоматизированные системы управления»

**Лабораторная работа № 4**

**Сетевое программирование с сокетами и каналами**

Выполнил: ст. гр. ИСиТ-191

Дювбанов В. В.

Проверил: ст. преподаватель

Денисевич Д. А.

Могилев, 2021

**Цель работы:** приобретение навыков проектирования и разработки приложений в архитектуре клиент-сервер.

**Задание для самостоятельной работы.**

Чат. Клиент посылает через сервер сообщение, которое получают все клиенты. Список клиентов хранится на сервере в файле.

**Java код:**

package javaapplication3;

import javaapplication3.Status;

import java.io.\*;

import java.net.Socket;

import java.util.ArrayList;

public class EchoThread extends Thread {

private Socket socket;

private BufferedReader in;

private PrintWriter out;

private String username;

private Status status;

private ArrayList<Socket> sockets;

public EchoThread(Socket clientSocket, ArrayList<Socket> sockets) {

this.socket = clientSocket;

try {

in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

out = new PrintWriter(new OutputStreamWriter(socket.getOutputStream()), true);

} catch (IOException e) {

return;

}

this.sockets = sockets;

status = new Status(clientSocket);

}

public void run() {

try {

username = in.readLine();

out.println(username + " в чате");

} catch (IOException e) {

e.printStackTrace();

}

new Thread(new Status(socket)).start();

while (true) {

try {

String text = in.readLine();

if (text.equals("quit")) {

socket.close();

return;

}

System.out.println("Echoing: " + username + ": " + text);

for (Socket socket:sockets) {

PrintWriter bc = new PrintWriter(new OutputStreamWriter(socket.getOutputStream()));

bc.println(username + ": " + text);

bc.flush();

}

} catch (IOException e) {

out.println(username + " вышел");

System.out.println("disconnected");

return;

}

}

}

}

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit this template

\*/

package javaapplication4;

import java.io.\*;

import java.net.Socket;

import java.nio.file.Files;

import java.nio.file.Path;

public class Main {

private static Socket socket;

private static BufferedReader reader;

private static BufferedReader in;

private static PrintWriter out;

public static void main(String[] args) throws IOException {

socket = new Socket("localhost", 8088);

reader = new BufferedReader(new InputStreamReader(System.in));

in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

out = new PrintWriter(new OutputStreamWriter(socket.getOutputStream()), true);

System.out.println("Соединение установлено:");

System.out.println("Введите имя: ");

String username = reader.readLine();

File file = new File("C:/Users/TheWalkSide/Documents/NetBeansProjects/JavaApplication3/documents/listOfClients.txt");

PrintWriter pw = new PrintWriter(file);

pw.write(username);

pw.close();

out.write(username + "\n");

out.flush();

String welcomeUser = in.readLine();

System.out.println(welcomeUser);

new Thread(new javaapplication4.Reader(socket)).start();

while(true){

System.out.println("->");

String word = reader.readLine();

out.write(word + "\n");

out.flush();

}

}

}

package javaapplication3;

import java.io.IOException;

import java.io.OutputStreamWriter;

import java.io.PrintWriter;

import java.net.Socket;

public class Status extends Thread{

private Socket socket;

private PrintWriter out;

public Status(Socket clientSocket) {

this.socket = clientSocket;

try {

out = new PrintWriter(new OutputStreamWriter(socket.getOutputStream()), true);

} catch (IOException e) {

e.printStackTrace();

}

}

public void run(){

while(true){

try {

Thread.sleep(30000);

} catch (InterruptedException e) {

e.printStackTrace();

}

out.println("Your status is online");

}

}

}

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit this template

\*/

package javaapplication4;

import java.io.\*;

import java.net.Socket;

import java.nio.file.Files;

import java.nio.file.Path;

public class Main {

private static Socket socket;

private static BufferedReader reader;

private static BufferedReader in;

private static PrintWriter out;

public static void main(String[] args) throws IOException {

socket = new Socket("localhost", 8088);

reader = new BufferedReader(new InputStreamReader(System.in));

in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

out = new PrintWriter(new OutputStreamWriter(socket.getOutputStream()), true);

System.out.println("Соединение установлено:");

System.out.println("Введите имя: ");

String username = reader.readLine();

File file = new File("C:/Users/TheWalkSide/Documents/NetBeansProjects/JavaApplication3/documents/listOfClients.txt");

PrintWriter pw = new PrintWriter(file);

pw.write(username);

pw.close();

out.write(username + "\n");

out.flush();

String welcomeUser = in.readLine();

System.out.println(welcomeUser);

new Thread(new javaapplication4.Reader(socket)).start();

while(true){

System.out.println("->");

String word = reader.readLine();

out.write(word + "\n");

out.flush();

}

}

}

package javaapplication4;

import java.io.\*;

import java.net.Socket;

public class Reader extends Thread{

private Socket socket;

private BufferedReader in;

public Reader(Socket clientSocket) {

this.socket = clientSocket;

try {

in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

} catch (IOException e) {

e.printStackTrace();

}

}

public void run(){

while(true){

try {

String serverWord = in.readLine();

System.out.println(serverWord);

} catch (IOException e) {

e.printStackTrace();

}

}

}

}

**Вывод:** в ходе работы были приобритены навыки проектирования и разработки приложений в архитектуре клиент-сервер.