Министерство науки и высшего образования РФ

ФГАОУ ВО «Омский государственный технический университет»

Кафедра «Автоматизированные системы обработки информации и управления»

# ОТЧЁТ

По лабораторной работе №5

По дисциплине: Объектно-ориентированное программирование студента Яценко Артёма Александровича группы ПИН-212

# Пояснительная записка

Шифр работы От-2068998-51-ПИН-212-1 ЛР

Специальность 09.03.04

Старший преподаватель А.А. Кабанов

Студент А.А. Яценко

Омск 2024

**Цель работы:** в лабораторной работе №5 необходимо реализовать сетевое приложение, использующее протоколы стека TCP/IP.

**Задание:** Клиент посылает через сервер сообщение другому клиенту.

**Ход работы**

1. Листинг кода программы

**Server:**

import java.io.\*;  
import java.net.ServerSocket;  
import java.net.Socket;  
import java.util.ArrayList;  
import java.util.List;  
  
public class server {  
 private List<ClientHandler> clients;  
  
 public static void main(String[] args) {  
 server server = new server();  
 server.start(5050);  
 }  
  
 public void start(int port) {  
 clients = new ArrayList<>();  
  
 try {  
 // Создание серверного сокета  
 ServerSocket serverSocket = new ServerSocket(port);  
 System.*out*.println("Сервер запущен на порту " + port);  
  
 while (true) {  
 // Ожидание подключения клиента  
 Socket clientSocket = serverSocket.accept();  
 System.*out*.println("Новый клиент подключился");  
  
 // Создание и запуск обработчика клиента  
 ClientHandler clientHandler = new ClientHandler(clientSocket);  
 clients.add(clientHandler);  
 new Thread(clientHandler).start();  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 // Отправка сообщения всем клиентам, кроме отправителя  
 public synchronized void broadcastMessage(ClientHandler sender, String message) {  
 for (ClientHandler client : clients) {  
 if (client != sender) {  
 client.sendMessage(message);  
 }  
 }  
 }  
  
 // Удаление клиента из списка  
 public synchronized void removeClient(ClientHandler client) {  
 clients.remove(client);  
 System.*out*.println("Клиент отключился");  
 }  
  
 // Обработчик клиента  
 class ClientHandler implements Runnable {  
 private Socket clientSocket;  
 private BufferedReader reader;  
 private PrintWriter writer;  
  
 public ClientHandler(Socket clientSocket) {  
 this.clientSocket = clientSocket;  
 }  
  
 @Override  
 public void run() {  
 try {  
 // Получение потоков ввода/вывода для обмена данными с клиентом  
 InputStream inputStream = clientSocket.getInputStream();  
 reader = new BufferedReader(new InputStreamReader(inputStream));  
 OutputStream outputStream = clientSocket.getOutputStream();  
 writer = new PrintWriter(outputStream, true);  
  
 String message;  
 while ((message = reader.readLine()) != null) {  
 // Отправка сообщения всем клиентам  
 broadcastMessage(this, message);  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 } finally {  
 // Закрытие соединения и удалениеклиента из списка при отключении  
 try {  
 reader.close();  
 writer.close();  
 clientSocket.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 removeClient(this);  
 }  
 }  
  
 // Отправка сообщения клиенту  
 public void sendMessage(String message) {  
 writer.println(message);  
 }  
 }  
}

**Client:**

import java.io.\*;  
import java.net.Socket;  
  
public class client {  
 public static void main(String[] args) {  
 try {  
 // Установка соединения с сервером  
 Socket socket = new Socket("localhost", 5050);  
 // Получение потоков ввода/вывода для обмена данными с сервером  
 OutputStream outputStream = socket.getOutputStream();  
 PrintWriter writer = new PrintWriter(outputStream, true);  
  
 // Получение потока ввода для чтения ответа от сервера  
 InputStream inputStream = socket.getInputStream();  
 BufferedReader reader = new BufferedReader(new InputStreamReader(inputStream));  
  
 // Чтение ответов от сервера в отдельном потоке  
 new Thread(new ServerResponseReader(reader)).start();  
  
 // Чтение пользовательского ввода и отправка сообщений на сервер  
 BufferedReader userInputReader = new BufferedReader(new InputStreamReader(System.*in*));  
 String userInput;  
 while ((userInput = userInputReader.readLine()) != null) {  
 writer.println(userInput);  
 }  
  
 // Закрытие соединения  
 writer.close();  
 outputStream.close();  
 reader.close();  
 inputStream.close();  
 socket.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
}  
  
// Класс для чтения ответов от сервера в отдельном потоке  
class ServerResponseReader implements Runnable {  
 private BufferedReader reader;  
  
 public ServerResponseReader(BufferedReader reader) {  
 this.reader = reader;  
 }  
  
 @Override  
 public void run() {  
 try {  
 String response;  
 while ((response = reader.readLine()) != null) {  
 System.*out*.println("Получено сообщение от сервера: " + response);  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
}

1. Тестирование кода

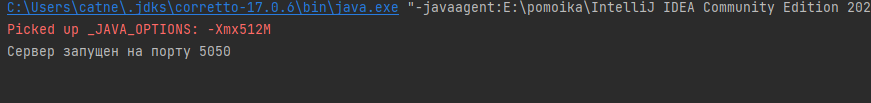


Рис. 1. – Сервер запущен.

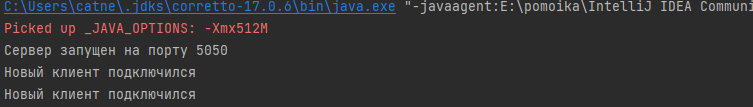
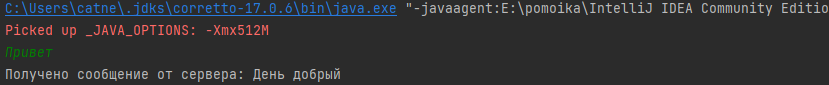


Рис. 2. – Подключение пользователей.



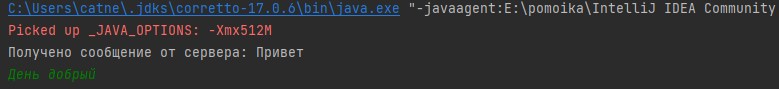


Рис. 3. – Общение пользователей (вид от лица каждого из них).

**Вывод:** во время выполнения лабораторное работы было реализовано сетевое приложение, использующее протоколы стека TCP/IP, и выполнено задание, в результате которого клиент может посылать через сервер сообщение другому клиенту.