Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №6

По дисциплине: «СПП»

Тема: “Клиент-серверная архитектура приложений Java.”

Выполнил:

Студент 3 курса

Группы ПО-7

Комиссаров А.Е.

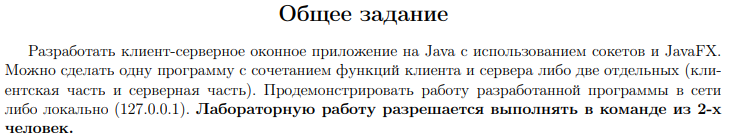
Дмитрук М.А.

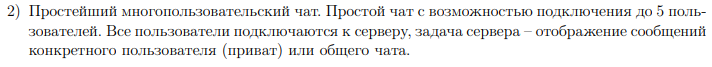
Проверил:

Монтик Н.С.

2023

**Цель работы:** освоить приёмы разработки оконных клиент-серверных приложений на Java с использованием сокетов.





**Код программы:**

**Client.java**

**import javax.swing.\*;**

**import java.net.\*;**

**import java.io.\*;**

**import java.util.\*;**

**/\***

**\* To start the Client-part in console mode use one of the following command**

**\* > java Client**

**\* > java Client username**

**\* > java Client username portNumber**

**\* > java Client username portNumber serverAddress**

**\* at the console prompt**

**\* If the portNumber is not specified 1200 is used**

**\* If the serverAddress is not specified "localHost" is used**

**\* If the username is not specified "Guest" is used**

**\* > java Client**

**\* > java Client Anonymous 1200 localhost**

**\*/**

**public class Client {**

**private ObjectInputStream sInput;**

**private ObjectOutputStream sOutput;**

**private Socket socket;**

**private boolean isGUI;**

**private ClientGUI clientGUI;**

**private JTextArea textGUI;**

**private String server, username;**

**private int port;**

**Client(String server, int port, String username) {**

**this(server, port, username, null, null);**

**this.isGUI = false;**

**}**

**Client(String server, int port, String username, JTextArea textArea, ClientGUI clientGUI) {**

**this.server = server;**

**this.port = port;**

**this.username = username;**

**this.textGUI = textArea;**

**this.isGUI = true;**

**}**

**public boolean start() {**

**try { socket = new Socket(server, port); }**

**catch (Exception exception) {**

**display("Connection error. \n" + exception);**

**return false;**

**}**

**String message = "Connection accepted " + socket.getInetAddress() + ":" + socket.getPort();**

**display(message);**

**try {**

**sInput = new ObjectInputStream(socket.getInputStream());**

**sOutput = new ObjectOutputStream(socket.getOutputStream());**

**}**

**catch (IOException IOException) {**

**display("IOStream error. \n" + IOException);**

**return false;**

**}**

**new ListenFromServer().start();**

**try {**

**sOutput.writeObject(username);**

**}**

**catch (IOException IOException) {**

**display("Login error. \n" + IOException);**

**disconnect();**

**return false;**

**}**

**return true;**

**}**

**private void display(String message) {**

**if (isGUI) {**

**textGUI.append(message + "\n");**

**textGUI.setCaretPosition(textGUI.getText().length() - 1);**

**} else {**

**System.out.println(message);**

**}**

**}**

**void sendMessage(Message message) { // Send to the server**

**try {**

**sOutput.writeObject(message);**

**}**

**catch (IOException exception) {**

**display("Send message error. \n" + exception);**

**}**

**}**

**private void disconnect() {**

**try {**

**if (sInput != null) sInput.close();**

**}**

**catch (Exception ignored) {}**

**try {**

**if (sOutput != null) sOutput.close();**

**}**

**catch (Exception ignored) {}**

**try {**

**if (socket != null) socket.close();**

**}**

**catch (Exception ignored) {}**

**if (clientGUI != null)**

**clientGUI.connectionFailed();**

**}**

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

**int portNumber = 1200;**

**String serverAddress = "localhost";**

**String userName = "Guest";**

**switch (args.length) {**

**case 3:**

**serverAddress = args[2];**

**case 2:**

**try {**

**portNumber = Integer.parseInt(args[1]);**

**}**

**catch (Exception exception) {**

**System.out.println("Invalid port number.");**

**System.out.println("Usage is: > java Client [username] [portNumber] [serverAddress]");**

**return;**

**}**

**case 1:**

**userName = args[0];**

**case 0:**

**break;**

**default:**

**System.out.println("Usage is: > java Client [username] [portNumber] {serverAddress]");**

**return;**

**}**

**Client client = new Client(serverAddress, portNumber, userName);**

**if (!client.start()) return;**

**Scanner scan = new Scanner(System.in);**

**while (true) {**

**System.out.print("> ");**

**String message = scan.nextLine();**

**if (message.equalsIgnoreCase("LOGOUT")) {**

**client.sendMessage(new Message(Message.LOGOUT, ""));**

**break;**

**}**

**else if (message.equalsIgnoreCase("WHOISIN")) {**

**client.sendMessage(new Message(Message.WHOISIN, ""));**

**}**

**else {**

**System.out.println("New message.");**

**client.sendMessage(new Message(Message.MESSAGE, message));**

**}**

**}**

**client.disconnect();**

**}**

**class ListenFromServer extends Thread {**

**public void run() {**

**while (true) {**

**try {**

**String message = (String) sInput.readObject();**

**if (isGUI) {**

**textGUI.append(message);**

**textGUI.setCaretPosition(textGUI.getText().length() - 1);**

**} else {**

**System.out.println(">" + message);**

**}**

**}**

**catch (IOException exception) {**

**display("Server has close the connection: " + exception);**

**if (clientGUI != null)**

**clientGUI.connectionFailed();**

**break;**

**}**

**catch (ClassNotFoundException ignored) {}**

**}**

**}**

**}**

**}**

**Server.java**

**import java.io.\*;**

**import java.net.\*;**

**import java.text.SimpleDateFormat;**

**import java.util.\*;**

**/\***

**\* This server can be run as a console application or as a GUI**

**\* To run as a console application just:**

**\* > java Server**

**\* > java Server portNumber**

**\* If the port number is not specified 1200-port is used**

**\*/**

**public class Server {**

**private static int uniqueId; // ID counter**

**private ArrayList<ClientThread> clients; // Client threads list**

**private ServerGUI serverGUI;**

**private SimpleDateFormat simpleDateFormat;**

**private int port;**

**private boolean isRunning;**

**public Server(int port) {**

**this(port, null);**

**}**

**public Server(int port, ServerGUI serverGUI) {**

**this.serverGUI = serverGUI;**

**this.port = port;**

**simpleDateFormat = new SimpleDateFormat("HH:mm:ss");**

**clients = new ArrayList<>();**

**}**

**public void start() {**

**isRunning = true;**

**try {**

**ServerSocket serverSocket = new ServerSocket(port);**

**while (isRunning) {**

**display("Waiting for clients on port" + port);**

**Socket socket = serverSocket.accept();**

**if (!isRunning) break;**

**ClientThread thread = new ClientThread(socket);**

**clients.add(thread);**

**thread.start();**

**}**

**try {**

**serverSocket.close();**

**for (ClientThread clientThread : clients) {**

**try {**

**clientThread.sInput.close();**

**clientThread.sOutput.close();**

**clientThread.socket.close();**

**} catch (IOException ignored) {}**

**}**

**}**

**catch (Exception exception) {**

**display("Can't close connections. \n" + exception);**

**}**

**}**

**catch (IOException ioException) {**

**String message = simpleDateFormat.format(new Date()) + " IOException. \n" + ioException;**

**display(message);**

**}**

**}**

**protected void stop() {**

**isRunning = false;**

**try {**

**new Socket("localhost", port);**

**}**

**catch (Exception ignored) {}**

**}**

**private void display(String message) { // Displaying the event**

**String time = simpleDateFormat.format(new Date()) + " " + message;**

**if (serverGUI == null) System.out.println(time);**

**else serverGUI.appendEvent(time + "\n");**

**}**

**private synchronized void broadcast(String message) {**

**String time = simpleDateFormat.format(new Date());**

**String messageAndTime = time + " " + message + "\n";**

**if (serverGUI == null) System.out.print(messageAndTime);**

**else serverGUI.appendRoom(messageAndTime);**

**for (int i = clients.size(); --i >= 0; ) {**

**ClientThread clientThread = clients.get(i);**

**if (!clientThread.writeMessage(messageAndTime)) {**

**clients.remove(i);**

**display("Disconnected client " + clientThread.username);**

**}**

**}**

**}**

**synchronized void remove(int id) {**

**for (int i = 0; i < clients.size(); ++i) {**

**ClientThread clientThread = clients.get(i);**

**if (clientThread.id == id) {**

**clients.remove(i);**

**return;**

**}**

**}**

**}**

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

**int portNumber = 1200;**

**switch (args.length) {**

**case 1:**

**try { portNumber = Integer.parseInt(args[0]); }**

**catch (Exception exception) {**

**System.out.println("Invalid port number.");**

**System.out.println("Usage is: > java Server [portNumber]");**

**return;**

**}**

**case 0:**

**break;**

**default:**

**System.out.println("Usage is: > java Server [portNumber]");**

**return;**

**}**

**Server server = new Server(portNumber);**

**server.start();**

**}**

**class ClientThread extends Thread {**

**Socket socket;**

**ObjectInputStream sInput;**

**ObjectOutputStream sOutput;**

**int id;**

**String username;**

**Message clientMessage;**

**String currentDate;**

**ClientThread(Socket socket) {**

**id = ++uniqueId;**

**this.socket = socket;**

**try {**

**sOutput = new ObjectOutputStream(socket.getOutputStream());**

**sInput = new ObjectInputStream(socket.getInputStream());**

**username = (String) sInput.readObject();**

**display(username + " has connected.");**

**}**

**catch (IOException exception) {**

**display("IOStream create error. \n" + exception);**

**return;**

**}**

**catch (ClassNotFoundException ignored) {}**

**currentDate = new Date().toString();**

**}**

**public void run() {**

**boolean keepGoing = true;**

**while (keepGoing) {**

**try {**

**clientMessage = (Message) sInput.readObject();**

**}**

**catch (IOException exception) {**

**display(username + " stream exception. \n" + exception);**

**break;**

**}**

**catch (ClassNotFoundException exception) {**

**break;**

**}**

**String message = clientMessage.getMessage(); // The message part**

**switch (clientMessage.getType()) { // Switcher of the types**

**case Message.MESSAGE -> broadcast(username + ": " + message);**

**case Message.LOGOUT -> {**

**display(username + " disconnected from room.");**

**keepGoing = false;**

**}**

**case Message.WHOISIN -> {**

**writeMessage("List of the guests connected at " + simpleDateFormat.format(new Date()) + "\n");**

**for (int i = 0; i < clients.size(); ++i) {**

**ClientThread clientThread = clients.get(i);**

**writeMessage((i + 1) + ") " + clientThread.username + " since " + clientThread.currentDate + "\n");**

**}**

**}**

**}**

**}**

**remove(id);**

**close();**

**}**

**private void close() {**

**try {**

**if (sOutput != null) sOutput.close();**

**}**

**catch (Exception ignored) {}**

**try {**

**if (sInput != null) sInput.close();**

**}**

**catch (Exception ignored) {}**

**try {**

**if (socket != null) socket.close();**

**}**

**catch (Exception ignored) {}**

**}**

**private boolean writeMessage(String message) {**

**if (!socket.isConnected()) {**

**close();**

**return false;**

**}**

**try {**

**sOutput.writeObject(message);**

**}**

**catch (IOException exception) {**

**display("Message send error. " + username);**

**display(exception.toString());**

**}**

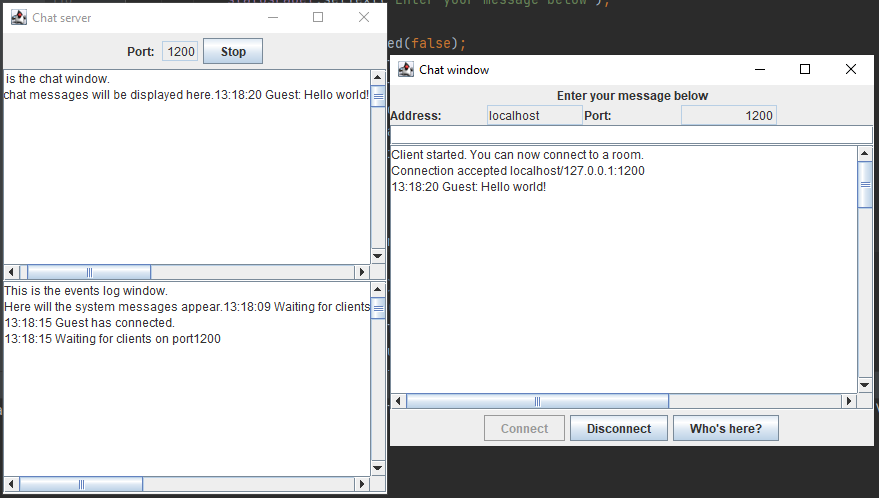
**return true;**

**}**

**}**

**}**

**Результат работы программы:**

****

**Вывод:** освоили приёмы разработки оконных клиент-серверных приложений на Java с использованием сокетов.