**Project Example #1: Building a Socket Server & Client**

Name: Kwon DoHyun

Student ID: 2023065350

Organization: Department of computer science and engineering, Hanyang Univ. (Seoul, Republic of Korea)

**Introduction**

This assignment aims to understand and implement the basic concepts of TCP client-server communication. The client and server use sockets to send and receive data. The client requests a connection to the server, sends a number to the server, the server receives the number, adds its own number, and returns the result to the client. The client returns the result and closes the socket.

**Contents of the attachment**

TCPClient.java (Make Block diagram of TCP Client)

|  |
| --- |
| **package** socket\_programming;  **import** java.io.\*;  **import** java.net.\*;  **class** TCPClient {  **public** **static** **void** main(String argv[]) **throws** Exception {  String host = "localhost"; // Server address  **int** port = 8888; // Server port number  // Mission 1: Open TCP Socket to connect with Server and send a message  //Fill#1, Initializing client-side socket to connect to Server  Socket clientSocket = new Socket(host, post)) // Mission 1  DataOutputStream outToServer = **new DataOutputStream(clientSocket.getOutputStream());**  BufferedReader inFromUser = **new** BufferedReader(**new** InputStreamReader(System.***in***));    BufferedReader inFromServer = **new** BufferedReader(**new** InputStreamReader(clientSocket.getInputStream()));  // Fill#2, Read an integer from the user to make a message to send  System.***out***.print("Enter an integer between 1 and 100: ");  **int** userNumber = Integer.parseInt(inFromUser.readline()); Mission 1  // Fill#3, Send the client name and the entered integer to the server  String clientMessage = "Client of John Q. Smith: " + userNumber;  outToServer.writeBytes(clientMessage + ‘\n’);// Mission 1  // Mission 4: Extract the server's chosen integer from the response and Calculate the sum of the client’s and server’s numbers  // Fill#8, Read and print the response from the server  String serverMessage = inFromServer.readLine(); //Mission 4  System.***out***.println("FROM SERVER: " + serverMessage); //Mission 4  // Extract the server's chosen integer from the response  String[] parts = serverMessage.split(": ");  String serverName = parts[0];  **int** serverNumber = Integer.*parseInt*(parts[1]);  // Fill#9, Calculate and print the sum of the client's and server's numbers  **int** sum = serverNumber + userNumber; //Mission 4  System.***out***.println("Client number: " + userNumber); //Mission 4  System.***out***.println("Server number: " + serverNumber); //Mission 4  System.***out***.println("Sum: " + sum); //Mission 4  // Fill#10, Close socket  clinentSocket.close();  }  } |
|  |

TCPServer.java(Make Block diagram of TCP Server)

|  |
| --- |
| **package** socket\_programming;  **import** java.io.\*;  **import** java.net.\*;  **class** TCPServer {  **public** **static** **void** main(String argv[]) **throws** Exception {  String serverName = "Server of John Q. Smith"; // Server name  **int** port = 8888; // Server port number  // Mission 2: Open TCP Socket to receive message from client  // Fill#4, Initializing the server on a specified port  ServerSocket welcomeSocket new ServerSocket(port); //Mission 2  **while** (**true**) {  // Fill#5, Listen for a TCP connection request.  Socket connectionSocket = welcomeSocket.accept(); //Mission 2  BufferedReader inFromServer = **new** BufferedReader(**new** InputStreamReader(System.***in***));  BufferedReader inFromClient = **new** BufferedReader(**new** InputStreamReader(connectionSocket.getInputStream()));  DataOutputStream outToClient = **new** DataOutputStream(connectionSocket.getOutputStream());  // Fill#6, Read message from client  String clientMessage = inFromClient.readLine(); //Mission 2  System.***out***.println("Received: " + clientMessage);  // Mission 3: Extract client's name and integer from message and send calculated message to client  // Extract client's name and integer  String[] parts = clientMessage.split(": ");  String clientName = parts[0];  **int** clientNumber = Integer.*parseInt*(parts[1]);    System.***out***.print("Enter an integer between 1 and 100: ");  **int** serverNumber = Integer.*parseInt*(inFromServer.readLine());  // Fill#7, Calculate and print the sum of the client's and server's numbers  **int** sum = clientNumber + serverNumber;// Mission 3  System.***out***.println("Client number: " + clientNumber);  System.***out***.println("Server number: " + serverNumber);  System.***out***.println("Sum: " + sum);  // Fill#8, Send server's name and chosen number back to the client  String serverMessage = (serverName + “: “ + serverNumber); // Mission 3  outToClient.writeBytes(serverMessage + ‘\n’); // Mission 3  // Terminate the server if the client sends a number outside the range of 1 to 100  **if** (clientNumber < 1 || clientNumber > 100) {  connectionSocket.close();  welcomeSocket.close();  **break**;  }  }  }  } |

**Instructions: How to run the program**

1. In Eclipse, first run the TCP server.
2. Second, run the TCP Client.
3. Third, send client number.



1. Finally, send server number.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

**Wireshark**

텍스트, 전자제품, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

The connection between the client and server was detected with Wireshark using the tcp.port == 8888 filter.

Client port: 53987

Server port: 8888

3-way handshake

1. SYN packet is sent from client to server to initialize TCP connection.
2. SYN, ACK response is sent from server to client.
3. ACK response is sent from client to server

**How the program works**

텍스트, 도표, 영수증, 평행이(가) 표시된 사진

자동 생성된 설명

1. Server and client both open socket.
2. Client receives input number from user.
3. Client sends data to server.
4. Server receives data from client.
5. Server sends data to client.
6. Client prints result.
7. Close Socket.