**TCP Chat Server and Client - Project 1**

**Project Overview**

This project implements a simple TCP chat server that handles multiple clients simultaneously and tracks the total number of messages received since the server started.

**Files Included**

* ChatServer.java - The server application that accepts multiple client connections
* ChatClient.java - The client application that connects to the server
* Documentation.docx - This documentation file

**System Requirements**

* Java Development Kit (JDK) 11 or higher
* Any IDE (IntelliJ IDEA recommended) or command line

**How to Compile**

Open a terminal/command prompt and navigate to the project directory, then compile both files:

A screen shot of a computer

AI-generated content may be incorrect.

**How to Run**

**Step 1: Start the Server**

Open a terminal window and run:



You should see:



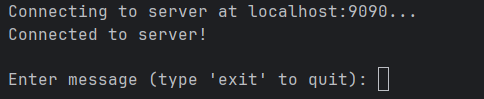
**Keep this terminal window open** - the server needs to stay running.

**Step 2: Start Client #1**

Open a **second** terminal window and run:



You should see:



**Step 3: Start Client #2**

Open a **third** terminal window and run:



Another client instance connects to the same server.

**Testing the Application**

**Send Messages from Both Clients**

**In Client #1 window enter:**



You should see:

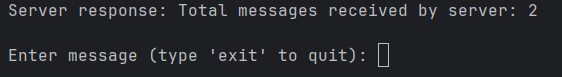
A black screen with white text

AI-generated content may be incorrect.

**In Client #2 window enter:**



You should see:



In Terminal #1 you should see:

A screenshot of a computer

AI-generated content may be incorrect.

**Test Message Counter**

Continue sending messages from both clients and observe:

* Each client receives the incremented total message count
* The counter is shared across all clients
* Messages from different clients increment the same counter

**Test Exit Functionality**

**In any client window, type:**



**Expected behavior:**

* Client displays: Disconnecting from server...
* Client waits 2 seconds
* Client displays: Connection closed.
* Server displays: Client /127.0.0.1:xxxxx is disconnecting...
* Server displays: Client /127.0.0.1:xxxxx disconnected.
* The remaining client continues to work normally

**Client terminal:**

A black background with white text

AI-generated content may be incorrect.

**Server Terminal:**

A black background with white text

AI-generated content may be incorrect.

**Key Features Implemented**

**1. TCP Socket Communication**

* Server uses ServerSocket to listen on port 9090
* Clients use Socket to connect to the server
* Bidirectional communication using input/output streams

**2. Multithreading**

* Server creates a new thread (ClientHandler) for each connected client
* Multiple clients can communicate with the server simultaneously
* Main server thread continues accepting new connections

**3. Thread Synchronization**

* The messageCount variable is shared across all client threads
* The incrementAndGetCount() method uses synchronized keyword
* Prevents race conditions and ensures accurate message counting

**4. Exception Handling**

* Try-catch blocks handle IOException and network errors
* Finally blocks ensure proper resource cleanup
* Graceful handling of unexpected client disconnections

**5. Clean Resource Management**

* Input/output streams are properly closed
* Sockets are closed when clients disconnect
* No resource leaks

**Server output:**

**A screen shot of a message

AI-generated content may be incorrect.**

**Client output:**

**A black background with white text

AI-generated content may be incorrect.**

**Troubleshooting**

**"Port already in use" Error**

* Another application is using port 9090
* Change PORT value in ChatServer.java to a different port (e.g., 8080)
* Update SERVER\_PORT in ChatClient.java to match

**"Connection refused" Error**

* Make sure the server is running before starting clients
* Verify both files use the same port number
* Check that no firewall is blocking the connection

**Client Cannot Connect to Server on Different Machine**

* Change SERVER\_ADDRESS in ChatClient.java from "localhost" to the server's IP address
* Ensure the server's firewall allows incoming connections on port 9090

**Technical Implementation Details**

**Server Architecture:**

* Main Thread: Accepts incoming client connections in an infinite loop
* ClientHandler Threads: Each handles one client's communication independently
* Shared Counter: Protected by synchronization to prevent race conditions

**Client Architecture:**

* Connects to server using TCP socket
* Reads user input from keyboard using Scanner
* Sends messages to server and receives responses
* Implements 2-second delay before closing on "exit" command

**Synchronization Mechanism:**

**A screen shot of a computer

AI-generated content may be incorrect.**

The synchronized keyword ensures only one thread can execute this method at a time, preventing multiple threads from reading/writing the counter simultaneously.