| **Name of Student:** Ajay Karthikesan | | | |
| --- | --- | --- | --- |
| **Roll Number:** 57 | | **Practical Number:** 1 | |
| **Aim of Practical:**  To develop a multi-client chat server application where multiple clients chat with each other concurrently, where the messages sent by different clients are first communicated to the server and then the server, on behalf of the source client, communicates the messages to the appropriate destination client. | | | |
| **DOP:** 24.8.23 | | **DOS:** 10.10.23 | |
| **CO Mapped:**  CO1 | **PO Mapped:**  - | **Faculty Signature:** | **Marks:** |

## 

## Practical No. 1

**Aim:**

To develop a multi-client chat server application where multiple clients chat with each other concurrently, where the messages sent by different clients are first communicated to the server and then the server, on behalf of the source client, communicates the messages to the appropriate destination client.

**Theory:**

Remote Procedure Call, Socket class, Server Socket class

**Code:**

File: client/Client.java

package vesitajayk57.chatapp.client;

import java.io.PrintWriter;

import java.net.Socket;

import java.util.Scanner;

public class Client {

public static void main(String[] args) {

String name = "empty";

String reply = "empty";

Scanner sc = new Scanner(System.in);

System.out.println("Enter your name (Please enter your name to join the chat): ");

reply = sc.nextLine();

name = reply;

try (Socket socket = new Socket("localhost", 5000)) {

PrintWriter cout = new PrintWriter(socket.getOutputStream(), true);

ThreadClient threadClient = new ThreadClient(socket);

new Thread(threadClient).start(); // start thread to receive message

cout.println(reply + ": has joined chat-room.");

do {

String message = (name + " : ");

reply = sc.nextLine();

if (reply.equals("logout")) {

cout.println("logout");

break;

}

cout.println(message + reply);

} while (!reply.equals("logout"));

} catch (Exception e) {

System.out.println(e.getStackTrace());

}

}

}

File: client/ThreadClient.java

package vesitajayk57.chatapp.client;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.net.Socket;

import java.net.SocketException;

/\*\*

\* Thread for clients

\*/

public class ThreadClient implements Runnable {

private Socket socket;

private BufferedReader cin;

public ThreadClient(Socket socket) throws IOException {

this.socket = socket;

this.cin = new BufferedReader(new InputStreamReader(socket.getInputStream()));

}

@Override

public void run() {

try {

while (true) {

String message = cin.readLine();

System.out.println(message);

}

} catch (SocketException e) {

System.out.println("You left the chat-room");

} catch (IOException exception) {

System.out.println(exception);

} finally {

try {

cin.close();

} catch (Exception exception) {

System.out.println(exception);

}

}

}

}

File: server/Server.java

package vesitajayk57.chatapp.server;

import java.net.ServerSocket;

import java.net.Socket;

import java.util.ArrayList;

import java.util.HashMap;

public class Server {

public static void main(String[] args) {

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

HashMap<Socket, String> clientNameList = new HashMap<Socket, String>();

try (ServerSocket serversocket = new ServerSocket(5000)) {

System.out.println("Server is started...");

while (true) {

Socket socket = serversocket.accept();

clients.add(socket);

ThreadServer ThreadServer = new ThreadServer(socket, clients, clientNameList);

ThreadServer.start();

}

} catch (Exception e) {

System.out.println(e.getStackTrace());

}

}

}

File: server/ThreadServer.java

package vesitajayk57.chatapp.server;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.PrintWriter;

import java.net.Socket;

import java.net.SocketException;

import java.util.ArrayList;

import java.util.HashMap;

/\*\*

\* Thread for Server

\*/

public class ThreadServer extends Thread {

private Socket socket;

private ArrayList<Socket> clients;

private HashMap<Socket, String> clientNameList;

public ThreadServer(Socket socket, ArrayList<Socket> clients, HashMap<Socket, String> clientNameList) {

this.socket = socket;

this.clients = clients;

this.clientNameList = clientNameList;

}

@Override

public void run() {

try {

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

while (true) {

String outputString = input.readLine();

if (outputString.equals("logout")) {

throw new SocketException();

}

if (!clientNameList.containsKey(socket)) {

String[] messageString = outputString.split(":", 2);

clientNameList.put(socket, messageString[0]);

System.out.println(messageString[0] + messageString[1]);

showMessageToAllClients(socket, messageString[0] + messageString[1]);

} else {

System.out.println(outputString);

showMessageToAllClients(socket, outputString);

}

}

} catch (SocketException e) {

String printMessage = clientNameList.get(socket) + " left the chat room";

System.out.println(printMessage);

showMessageToAllClients(socket, printMessage);

clients.remove(socket);

clientNameList.remove(socket);

} catch (Exception e) {

System.out.println(e.getStackTrace());

}

}

private void showMessageToAllClients(Socket sender, String outputString) {

Socket socket;

PrintWriter printWriter;

int i = 0;

while (i < clients.size()) {

socket = clients.get(i);

i++;

try {

if (socket != sender) {

printWriter = new PrintWriter(socket.getOutputStream(), true);

printWriter.println(outputString);

}

} catch (IOException ex) {

System.out.println(ex);

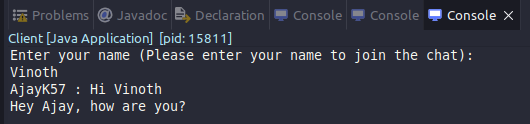
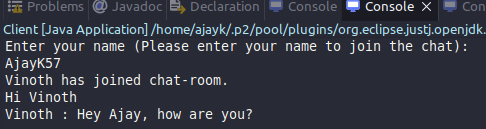
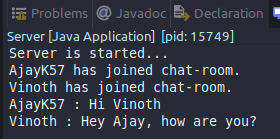
}

}

}

}

**Output:**



**Conclusion:**

I implemented a multi-client chat application using sockets in Java.