Name of Student: Pushkar Sane		
Roll Number: 45	**Lab Assignment Number: 1**	
Title of Lab Assignment: Remote Process Communication		
DOP: 05-08-2024	**DOS: 12-08-2024**	
CO Mapped:	**PO Mapped:**	**Signature:**

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.

Description:

A multi-client chat server application where multiple clients chat with each other concurrently. 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.

Code:

Server.java

```
package message;
import java.io.*;
import java.net.*;
import java.util.*;

public class server {
    private static List<ClientHandler> clients = new ArrayList<>();
    private static String privateRecipient = null;

    public static void main(String[] args) {
        try {
            ServerSocket serverSocket = new ServerSocket(9999);
            System.out.println("Server is running and listening on port 9999...");

            while (true) {
                Socket clientSocket = serverSocket.accept();
                System.out.println("New client connected");

                ClientHandler clientHandler = new ClientHandler(clientSocket);
                clients.add(clientHandler);
                clientHandler.start();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    static class ClientHandler extends Thread {
        private Socket clientSocket;
        private PrintWriter writer;
        private String name;

        public ClientHandler(Socket socket) {
            this.clientSocket = socket;
        }

        public void run() {
            try {
                                        BufferedReader reader = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
                writer = new PrintWriter(clientSocket.getOutputStream(), true);

                System.out.print("Enter your name: ");
                name = reader.readLine();
                writer.println("Welcome, " + name + "!");

                String message;
                while ((message = reader.readLine()) != null) {
                    if (message.startsWith("/private")) {
                        String[] parts = message.split(" ", 2);
                        String recipient = parts[1];
                        setPrivateRecipient(recipient);
                        writer.println("Chatting privately with " + recipient);
                    } else if (message.equals("/offprivate")) {
                        setPrivateRecipient(null);
                        writer.println("Chatting with everyone");
                    } else {
```

```
                if (isInPrivateMode()) {
                    sendPrivateMessage(name, privateRecipient, message);
                } else {
                    broadcastMessage(name + ": " + message);
                }
            }
        }
        clients.remove(this);
        writer.close();
        reader.close();
        clientSocket.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

private void broadcastMessage(String message) {
    for (ClientHandler client : clients) {
        client.writer.println(message);
    }
}

private void sendPrivateMessage(String sender, String recipient, String message) {
    for (ClientHandler client : clients) {
        if (client.name.equals(recipient)) {
            client.writer.println("[Private from " + sender + "]: " + message);
        }
    }
}

private synchronized boolean isInPrivateMode() {
    return privateRecipient != null;
}

private synchronized void setPrivateRecipient(String recipient) {
    privateRecipient = recipient;
```

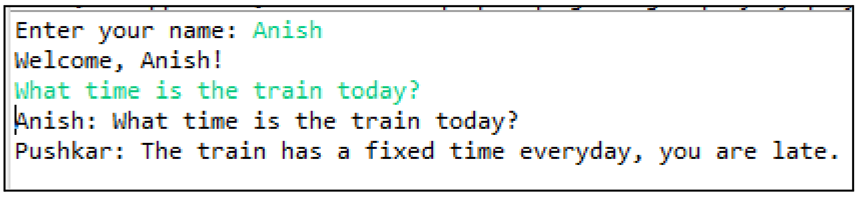
```
        }
    }
}
```

Client.java

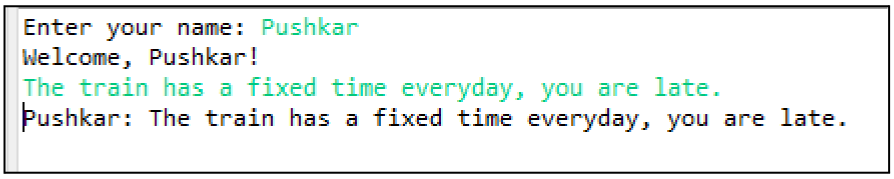
```
package message;
import java.io.*;
import java.net.*;
import java.util.Scanner;

public class client {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter your name: ");
        String name = scanner.nextLine();

        try {
            Socket socket = new Socket("localhost", 9999);
                                        BufferedReader reader = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
            PrintWriter writer = new PrintWriter(socket.getOutputStream(), true);

            Thread readThread = new Thread(() -> {
                try {
                    String message;
                    while ((message = reader.readLine()) != null) {
                        System.out.println(message);
                    }
                } catch (IOException e) {
                    e.printStackTrace();
                }
            });
            readThread.start();

            Thread writeThread = new Thread(() -> {
                try {
```

```
                Scanner inputScanner = new Scanner(System.in);
                String input;
                while (true) {
                    input = inputScanner.nextLine();
                    writer.println(input);
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        });
        writeThread.start();
        writer.println(name);
        readThread.join();
        writeThread.join();

        reader.close();
        writer.close();
        socket.close();
    } catch (IOException | InterruptedException e) {
        e.printStackTrace();
    }
  }
}
```

Output:



```
Enter your name: Anish
Welcome, Anish!
What time is the train today?
Anish: What time is the train today?
Pushkar: The train has a fixed time everyday, you are late.
```



```
Enter your name: Pushkar
Welcome, Pushkar!
The train has a fixed time everyday, you are late.
Pushkar: The train has a fixed time everyday, you are late.
```

Conclusion:

Successfully demonstrated Remote Process Communication.