Experiment – 04

Advance Java Lab (5CS4-24)

Class – B.Tech III Year, V Sem.

Objective:

Write a java program for two way TCP communication for server and client. It should look like a simple chat application.

Code:

GossipClient.java

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

public class GossipClient {
    private static final String SERVER_ADDRESS = "localhost";
    private static final int SERVER_PORT = 12345;

    public static void main(String[] args) {
        try {
            Socket socket = new Socket(SERVER_ADDRESS, SERVER_PORT);
            System.out.println("Connected to the chat server!");

            // Setting up input and output streams
            PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
            BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

            // Start a thread to handle incoming messages
            new Thread(() -> {
                try {
                    String serverResponse;
                    while ((serverResponse = in.readLine()) != null) {
                        System.out.println(serverResponse);
```

```
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
        }).start();

        // Read messages from the console and send to the server
        Scanner scanner = new Scanner(System.in);
        String userInput;
        while (true) {
            userInput = scanner.nextLine();
            out.println(userInput);
        }

    } catch (IOException e) {
        e.printStackTrace();
    }
  }
}
```

<u>GossipServer.java</u>

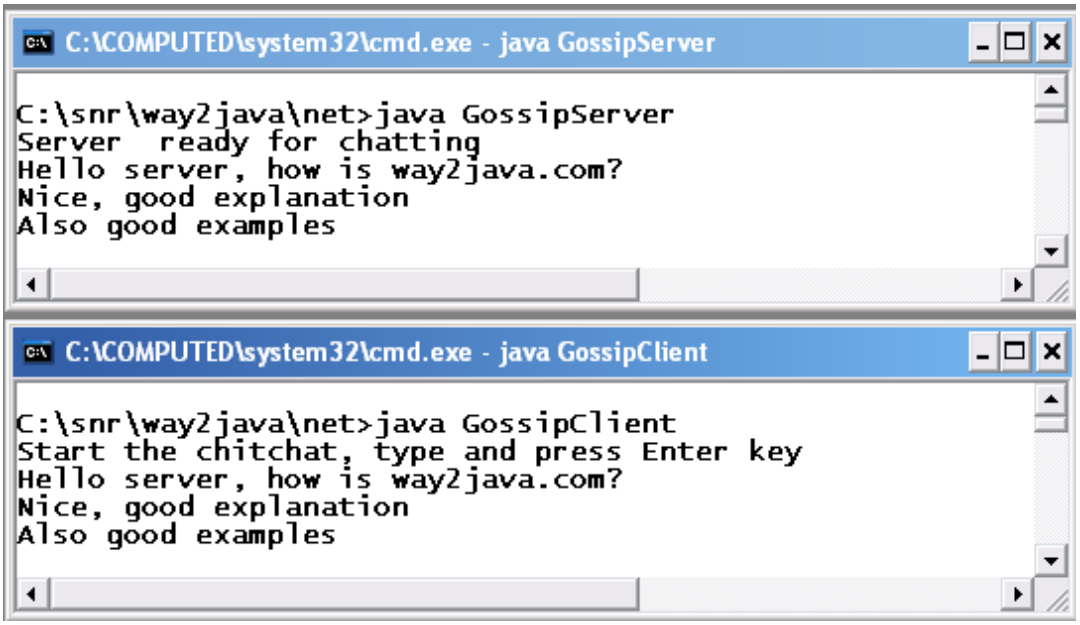
```
//Server program to handle multiple
// Clients with socket connections
import java.io.*;
import java.net.*;
import java.util.concurrent.CopyOnWriteArrayList;

public class GossipServer {
  private static final int PORT = 1234;
  private static CopyOnWriteArrayList<ClientHandler> clients = new CopyOnWriteArrayList<>();

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

```
        try {
            ServerSocket serverSocket = new ServerSocket(PORT);
            System.out.println("Server is running and waiting for connections..");

            // Accept incoming connections
            while (true) {
                Socket clientSocket = serverSocket.accept();
                System.out.println("New client connected: " + clientSocket);

                // Create a new client handler for the connected client
                ClientHandler clientHandler = new ClientHandler(clientSocket);
                clients.add(clientHandler);
                new Thread(clientHandler).start();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    // Broadcast a message to all clients except the sender
    public static void broadcast(String message, ClientHandler sender) {
        for (ClientHandler client : clients) {
            if (client != sender) {
                client.sendMessage(message);
            }
        }
    }

    // Internal class to handle client connections
    private static class ClientHandler implements Runnable {
        private Socket clientSocket;
        private PrintWriter out;
```

```
    private BufferedReader in;
    private String Username; // Use Username consistently

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

        try {
            // Create input and output streams for communication
            out = new PrintWriter(clientSocket.getOutputStream(), true);
            in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    // Run method to handle client communication
    @Override
    public void run() {
        try {
            // Get the username from the client
            Username = getUsername(); // Use Username consistently
            System.out.println("User " + Username + " connected."); // Use Username consistently

            out.println("Welcome to the chat, " + Username + "!"); // Use Username consistently
            out.println("Type Your Message");
            String inputLine;

            // Continue receiving messages from the client
            while ((inputLine = in.readLine()) != null) {
                System.out.println("[" + Username + "]: " + inputLine); // Use Username consistently
```

```
                // Broadcast the message to all clients
                broadcast("[" + Username + "]: " + inputLine, this); // Use Username consistently
            }

            // Remove the client handler from the list
            clients.remove(this);

            // Close the input and output streams and the client socket
            in.close();
            out.close();
            clientSocket.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    // Get the username from the client
    private String getUsername() throws IOException {
        out.println("Enter your username:");
        return in.readLine();
    }

    public void sendMessage(String message) {
        out.println(message);
        out.println("Type Your Message");
    }
  }
}
```

Output:


C:\COMPUTED\system32\cmd.exe - java GossipServer
C:\snr\way2java\net>java GossipServer
Server ready for chatting
Hello server, how is way2java.com?
Nice, good explanation
Also good examples
C:\COMPUTED\system32\cmd.exe - java GossipClient
C:\snr\way2java\net>java GossipClient
Start the chitchat, type and press Enter key
Hello server, how is way2java.com?
Nice, good explanation
Also good examples