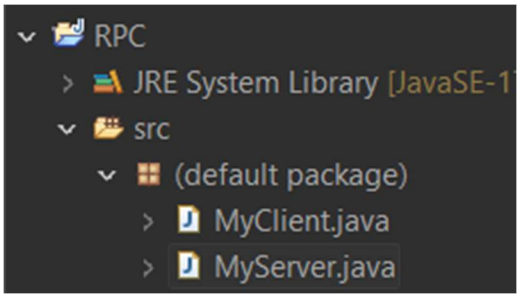
Practical No. 1

Remote Process Communication

Q.1 Write a java program to implement client server application using TCP.

Structure :



Program :

MyServer.java

```
import java.io.*;
import java.net.*;
public class MyServer{
	public static void main(String[] args)
	{
		try{
			//1. Open the Server Socket and Register service on port
			ServerSocket ss =new ServerSocket(2222);
			//2. Wait for the Client Request and accept a
			Socket s=ss.accept();
			DataInputStream in=new DataInputStream(System.in);
			//3. Create I/O streams for communicating to the client
			DataInputStream dis = new DataInputStream(s.getInputStream());
			DataOutputStream dout=new
DataOutputStream(s.getOutputStream());

			//4. Perform communication with
```

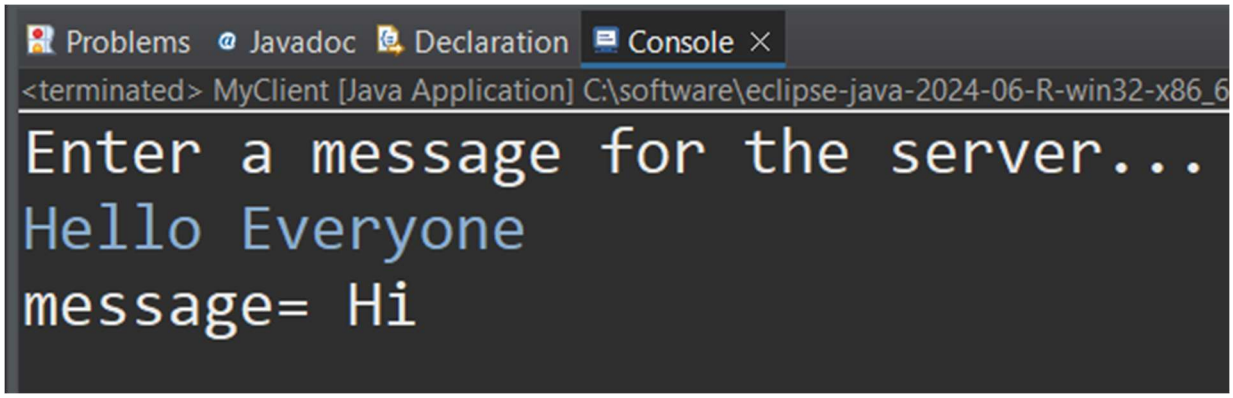
```
                    String str=dis.readUTF()
                    System.out.println("message= "+str);
                    System.out.println("Enter a message for the Client...");
                    @SuppressWarnings("deprecation")
                    String str1 = in.readLine();
                    dout.writeUTF(str1);

                    //5. flush and close
                    dout.flush();
                    dout.close();

                    //6. Close
                    s.close();
                    ss.close();
                }
                catch(Exception e){
                    System.out.println(e);
                }
            }
        }
```

MyClient.java

```
import java.io.*;
import java.net.*;
public class MyClient
{
    public static void main(String[] args){
        try{
//1. Create a Socket Object and Open your connection to a server at port
```

```
                Socket s=new Socket("localhost",2222);
//2. Create I/O streams for communicating with the server.
// Get an output file handle from the socket
                DataOutputStream dout=new DataOutputStream(s.getOutputStream());
// Get an input file handle from the socket and read the input
                DataInputStream dis = new DataInputStream(s.getInputStream());
                DataInputStream in = new DataInputStream(System.in);
                System.out.println("Enter a message for the server...");
                //3.Perform I/O or communication with the
                @SuppressWarnings("deprecation")
                String str = in.readLine();
                dout.writeUTF(str);
                String str1=dis.readUTF();
                System.out.println("message= "+str1);
                //4. flush and close
                dout.flush();
                dout.close();
                dis.close();
                //5. Close
                s.close();
                }
                catch(Exception e){
                        System.out.println(e);
                }
                }
        }
```

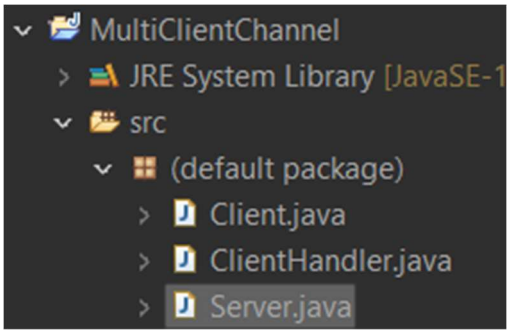
Output :


Problems
Javadoc
Declaration
Console
<terminated> MyClient [Java Application] C:\software\eclipse-java-2024-06-R-win32-x86_6
Enter a message for the server...
Hello Everyone
message= Hi


Problems
Javadoc
Declaration
Console
MyServer [Java Application] C:\software\eclipse-java-2024-06-R-win32-x86_64\eclipse\plugi
message= Hello Everyone
Enter a message for the Client...

Q.2 Develop a JAVA program for multi-client chat server using Socket.

Structure:



Program:

Server.java

```
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;

public class Server {

	private ServerSocket serverSocket;
	public Server(ServerSocket serverSocket) {
		this.serverSocket = serverSocket;
	}

	public void startServer() {
		try {
			while(!serverSocket.isClosed()) {
				Socket socket = serverSocket.accept();
				System.out.println("A new client has connected!");
				ClientHandler clientHandler = new ClientHandler(socket);
```

```
                Thread thread = new Thread((Runnable) clientHandler);
                thread.start();
            }
        }
        catch (IOException e) {
            // TODO: handle exception
            System.out.println("Exception Occurs : " + e.getMessage());
            e.printStackTrace();
        }
    }

    public void closeServerSocket() {
        try {
            if(serverSocket != null) {
                serverSocket.close();
            }
        }
        catch(IOException e) {
            e.printStackTrace();
        }
    }
    public static void main(String[] args) throws IOException{
        // TODO Auto-generated method stub
        ServerSocket serverSocket = new ServerSocket(8867);
        Server server = new Server(serverSocket);
        server.startServer();
    }
}
```

ClientHandler.java

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.net.Socket;
import java.util.ArrayList;

public class ClientHandler implements Runnable {
	public static ArrayList<ClientHandler> clientHandlers = new ArrayList<>();
	private Socket socket;
	private BufferedReader bufferedReader;
	private BufferedWriter bufferedWriter;
	private String clientUserName;
	@SuppressWarnings("unused")
	private String messageToSend;
	public ClientHandler(Socket socket) {
		// TODO Auto-generated constructor stub
		try {
			this.socket = socket;
			//  res allocated
			this.bufferedReader = new BufferedReader(new InputStreamReader(socket.getInputStream()));
			this.bufferedWriter = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));
			this.clientUserName = bufferedReader.readLine();
			clientHandlers.add(this);
			broadcastMessage("Server : " + clientUserName + " has entered the chat guys!");
```

```
        }
        catch(Exception e) {
                closeEverything(socket, bufferedReader, bufferedWriter);
        }
}

@Override
public void run() {
        String messageFromClient;
        while(socket.isConnected()) {
                try {
                        messageFromClient = bufferedReader.readLine();
                        broadcastMessage(messageFromClient);
                }
                catch(Exception e) {
                        closeEverything(socket, bufferedReader, bufferedWriter);
                        break;
                }
        }
}

public void broadcastMessage(String messageToSend) {
        this.messageToSend = messageToSend;
        for(ClientHandler clientHandler : clientHandlers) {
                try {
                        if(!clientHandler.clientUserName.equals(clientUserName)) {
                                clientHandler.bufferedWriter.write(messageToSend);
                                clientHandler.bufferedWriter.newLine();
```

```
                        clientHandler.bufferedWriter.flush();
                    }
                }
                catch(Exception e) {
                    closeEverything(socket, bufferedReader,bufferedWriter);
                }
            }
        }

        public void removeClientHandler() {
            clientHandlers.remove(this);
            broadcastMessage("Server : " + clientUserName + " has left the Chat Guys!");
        }

        public void closeEverything(Socket socket, BufferedReader bufferedReader,
BufferedWriter bufferedWriter) {
            // Lefted 1 member
            removeClientHandler();

            try {
                if(bufferedReader != null) {
                    bufferedReader.close();
                }
                if(bufferedWriter != null) {
                    bufferedWriter.close();
                }
                if(socket != null) {
                    socket.close();
                }
```

```
                }
                catch (Exception e) {
                        // TODO: handle exception
                        e.printStackTrace();
                }
        }
}
```

Client.java

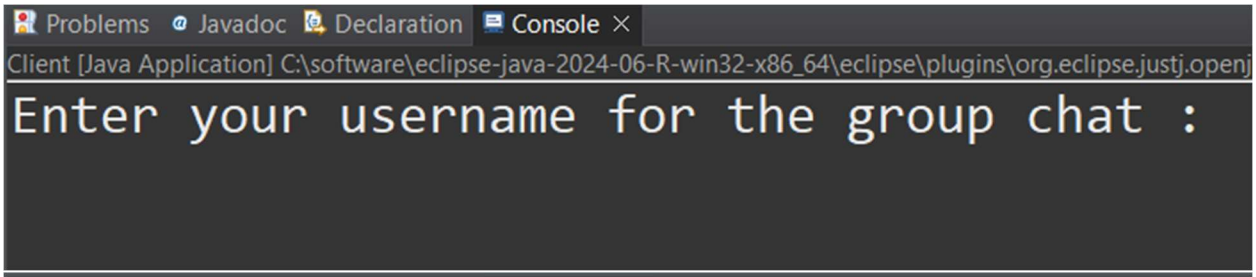
```
// when user start it
import java.io.BufferedWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.io.BufferedReader;
import java.net.Socket;
import java.net.UnknownHostException;
import java.util.Scanner;

public class Client {

        private Socket socket;
        private BufferedReader bufferedReader;
        private BufferedWriter bufferedWriter;
        private String username;

        public Client(Socket socket, String username) {
                try {
                        this.socket = socket;
```

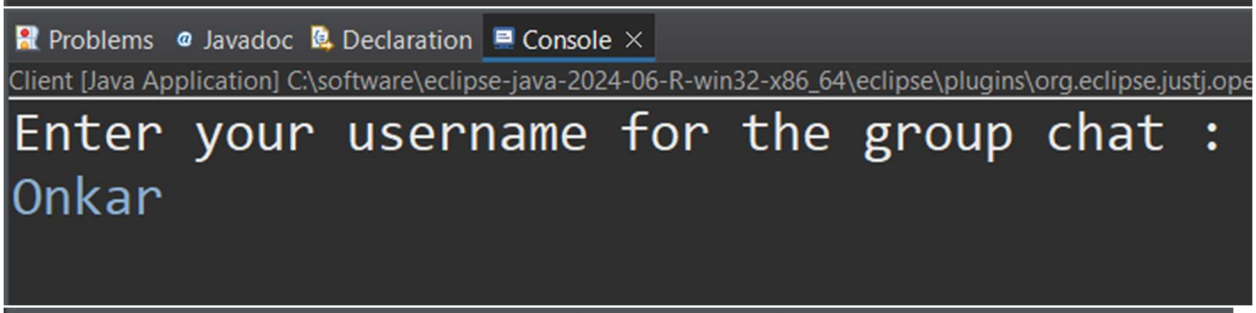
```
                this.bufferedReader = new BufferedReader(new InputStreamReader(socket.getInputStream()));
                this.bufferedWriter = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));
                this.username = username;
            }
            catch (Exception e) {
                // TODO: handle exception
                e.printStackTrace();
            }
        }

        public void sendMessage() {
            try {
                bufferedWriter.write(username);
                bufferedWriter.newLine();
                bufferedWriter.flush();
                Scanner sc = new Scanner(System.in);
                while (socket.isConnected()) {
                    String messageToSend = sc.nextLine();
                    bufferedWriter.write(username + " : "+ messageToSend);
                    bufferedWriter.newLine();
                    bufferedWriter.flush();
                }
                sc.close();
            }
            catch (Exception e) {
                // TODO: handle exception
                e.printStackTrace();
```

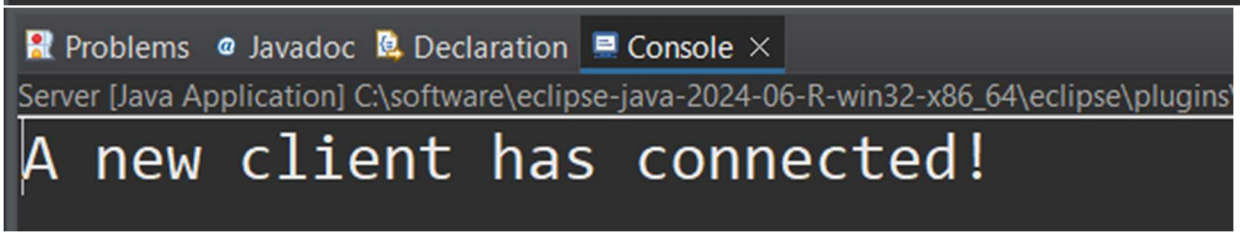
```
        }
    }

    public void listenForMessage() {
        new Thread(new Runnable() {

            @Override
            public void run() {
                // TODO Auto-generated method stub
                String msgFromGroupChat;
                while(socket.isConnected()) {
                    try {
                        msgFromGroupChat = bufferedReader.readLine();
                        System.out.println(msgFromGroupChat);
                    }
                    catch (Exception e) {
                        // TODO: handle exception
                        closeEverything(socket, bufferedReader,
bufferedWriter);
                        e.printStackTrace();
                    }
                }
            }
        }).start();
    }
    public void closeEverything(Socket socket, BufferedReader bufferedReader,
BufferedWriter bufferedWriter) {
        // Lefted 1 member
        try {
```

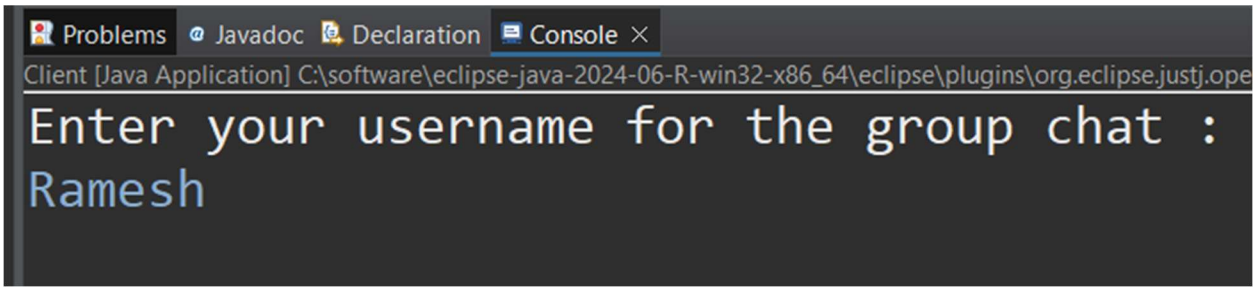
```
                if(bufferedReader != null) {
                        bufferedReader.close();
                }
                if(bufferedWriter != null) {
                        bufferedWriter.close();
                }
                if(socket != null) {
                        socket.close();
                }
        }
        catch (Exception e) {
                // TODO: handle exception
                e.printStackTrace();
        }
    }

    public static void main(String[] args) throws UnknownHostException, IOException {
        // TODO Auto-generated method s
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter your username for the group chat : ");
        String username = sc.nextLine();
        Socket socket = new Socket("localhost",8867);
        Client client = new Client(socket, username);
        client.listenForMessage();
        client.sendMessage();
        sc.close();
    }
}
```

Output:

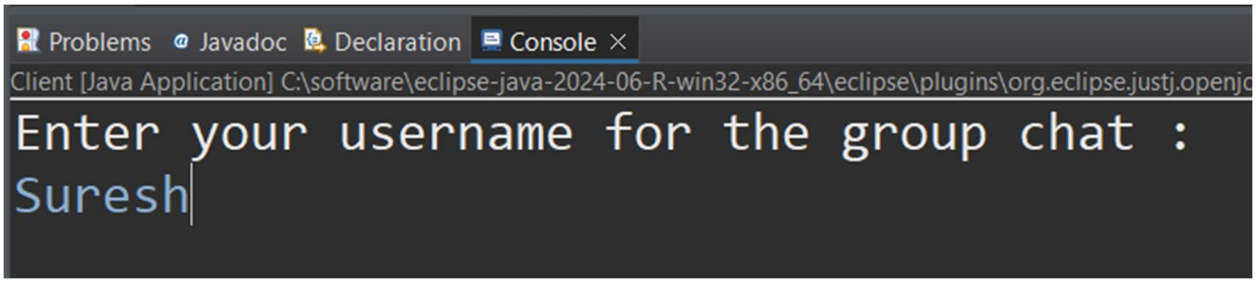

Problems Javadoc Declaration Console
Client [Java Application] C:\software\eclipse-java-2024-06-R-win32-x86_64\eclipse\plugins\org.eclipse.justj.openj
Enter your username for the group chat :

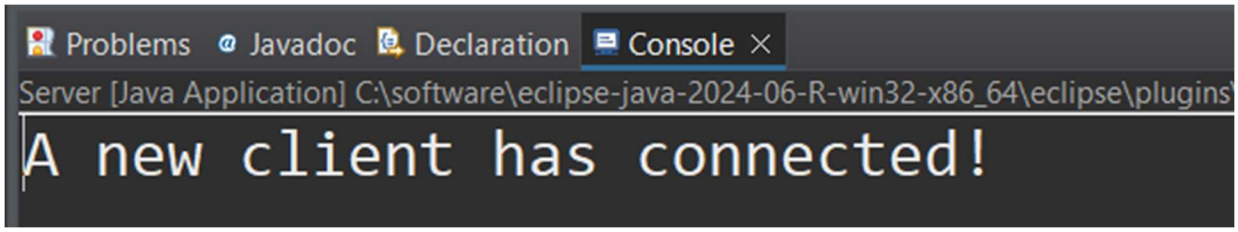

Problems Javadoc Declaration Console
Client [Java Application] C:\software\eclipse-java-2024-06-R-win32-x86_64\eclipse\plugins\org.eclipse.justj.ope
Enter your username for the group chat :
Onkar

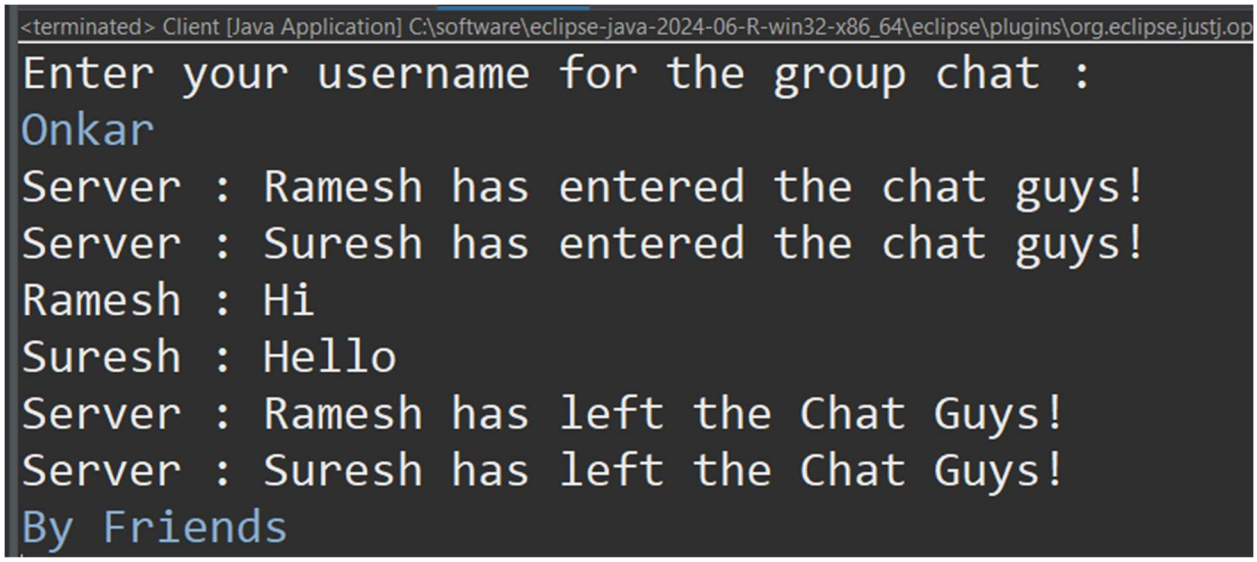

Problems Javadoc Declaration Console
Server [Java Application] C:\software\eclipse-java-2024-06-R-win32-x86_64\eclipse\plugins
A new client has connected!


Problems Javadoc Declaration Console
Client [Java Application] C:\software\eclipse-java-2024-06-R-win32-x86_64\eclipse\plugins\org.eclipse.justj.ope
Enter your username for the group chat :
Ramesh


Problems Javadoc Declaration Console
Server [Java Application] C:\software\eclipse-java-2024-06-R-win32-x86_64\eclipse\plugins
A new client has connected!


Problems Javadoc Declaration Console
Client [Java Application] C:\software\eclipse-java-2024-06-R-win32-x86_64\eclipse\plugins\org.eclipse.justj.openjc
Enter your username for the group chat :
Suresh


Problems
Javadoc
Declaration
Console
Server [Java Application] C:\software\eclipse-java-2024-06-R-win32-x86_64\eclipse\plugins
A new client has connected!


<terminated> Client [Java Application] C:\software\eclipse-java-2024-06-R-win32-x86_64\eclipse\plugins\org.eclipse.justj.op
Enter your username for the group chat :
Onkar
Server : Ramesh has entered the chat guys!
Server : Suresh has entered the chat guys!
Ramesh : Hi
Suresh : Hello
Server : Ramesh has left the Chat Guys!
Server : Suresh has left the Chat Guys!
By Friends