AHSANULLAH UNIVERSITY OF SCIENCE AND TECHNOLOGY

CSE4226: NETWORK PROGRAMMING LAB
ASSIGNMENT 2

Developing a multi-threaded server-client application using TCP Sockets

Author:
Naimul Haque

Roll:
14.02.04.080 (B1)

Submission date: June 06, 2018

Developing a multi-threaded server-client application

Naimul Haque, *14.02.04.080* ,
Ahsanullah University of Science and Technology

1 Implementation summary:

Following features are asked to be implemented in the assignment of 'Developing a multi-threaded server-client application using TCP Sockets' :

Section	Status
Register and Login	Implemented
Online User Lists	Implemented
Friend Request:	Implemented
Unicast	Implemented
Multicast	Implemented
Broadcast	Implemented
Chat Room	Implemented
History	not Implemented

2 Implementation Challenges

The challenges I found while trying to implement follow features :

1) **Register and Login**
   It was really challenging to implement register because validation of the new user requires checking all client objects' username and password.
   To implement login was a challenging since no database to store the registered users,ArrayList datastructure was used to store the users with validity.

2) **Online User Lists**

   To implement online users I used interface for all clients to react to another client's online status

3) **Friend Request**

   To implement this feature separate functions where implemented to send request from a user and to accept request from another user and update the friendlist accordingly

4) **Unicast**

   This feature was fairly easy to implement using a 'msg user' protocol with server

5) **Multicast**

   Challenge was to implent the protocol is used to multicast 'multicast <n> user1 user2 ... text...' which requires to iterate for n users

6) **Broadcast**

   It was easiest to implement as broadcasting requires send message to all clients with protocol 'broadcast text.....'

7) **Chat Room**

   To implement it required to add users to group which they can join using protocol 'join @group' where sever understand that it's a group by '@' character in front of the name

3 Interaction Diagram

An interaction diagram is show below to give an idea how the chat application works :

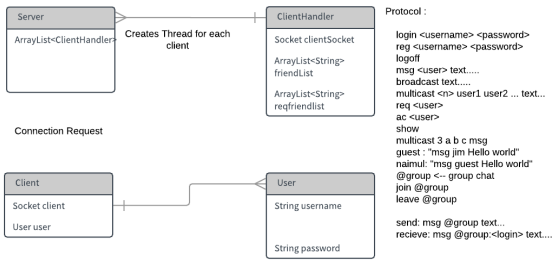

Figure 1. interaction diagram

Appendix A
Code

The Java code for designing a minimum distance to class mean appl :

Server Package:

Server.java

```
/*
 * To change this license header, choose License Headers in Project
     Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package server;

import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;


class Server extends Thread{

    private final int serverPort;
    private ArrayList<ClientHandler> workerList = new ArrayList<>();
    private ArrayList<User> validUsers = new ArrayList<>();

    public Server(int serverport) {
        this.serverPort = serverport;
    }



    @Override
    public void run() {
        ServerSocket serverSocket;
        try {
            serverSocket = new ServerSocket(serverPort);


            while(true){
            System.out.println("The server is waiting for client...");
            Socket clientSocket =  serverSocket.accept();
            System.out.println("The server is conneted to client
                  "+clientSocket);
            ClientHandler clientHandler = new
                  ClientHandler(this,clientSocket);
            workerList.add(clientHandler);
            clientHandler.start();
            }


        } catch (IOException ex) {
            Logger.getLogger(Server.class.getName()).log(Level.SEVERE,
                  null, ex);
        }


    }

    ArrayList<ClientHandler> getWorkerList() {
        return workerList;
    }

    void remove(ClientHandler clientHandler) {
        workerList.remove(clientHandler);
    }

    public ArrayList<User> getValidUsers() {
        return validUsers;
    }

    public void addUser(User user){
        validUsers.add(user);
    }

    public void removeValidUsers(User user) {
        validUsers.remove(user);
    }
}
```

ClientHandler.java

```
//Naimul Haque

package server;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.Socket;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.logging.Level;
import java.util.logging.Logger;
import org.apache.commons.lang3.StringUtils;


public class ClientHandler extends Thread{

    private final Socket clientSocket;
    private String login = null;
    private String password = null;
    private boolean isOnline;
    private  Server server;
    private OutputStream outputStream;
    private HashSet<String> groupSet = new HashSet<>();
    private ArrayList<String> friendList = new ArrayList<>();
    private ArrayList<String> reqfriendlist = new ArrayList<>();

    public ClientHandler(Server server,Socket clientSocket) {
        this.server = server;
        this.clientSocket = clientSocket;
        this.isOnline = false;
    }

    @Override
    public void run() {
        try {

            handleClientSocket();

        } catch (IOException ex) {
            Logger.getLogger(ClientHandler.class.getName()).log(Level.SEVERE,
                  null, ex);
        }
    }

    private void handleClientSocket() throws IOException {
        this.outputStream = clientSocket.getOutputStream();

        InputStream inputStream = clientSocket.getInputStream();
        BufferedReader reader = new BufferedReader(new
              InputStreamReader(inputStream));

        String line;

        while((line = reader.readLine())!= null){
            String[] tokens = StringUtils.split(line);

            if(tokens!=null&& tokens.length > 0){
```

```
            String command = tokens[0];

            if (command.equalsIgnoreCase("quit") ||
                  command.equals("logoff")) {
                handleLogoff();
                break;
            }
            else if (command.equalsIgnoreCase("reg")) {

                handleRegister(outputStream,tokens);

            }
            else if(command.equalsIgnoreCase("broadcast")){
                String[] broadcastMsgTokens =
                      StringUtils.split(line,null,2);
                handleBroadcast(broadcastMsgTokens);
            }
            else if(command.equalsIgnoreCase("multicast")){
                handleMulticast(tokens,line);
            }
            else if(command.equalsIgnoreCase("msg"))
            {
                String[] directMsgTokens =
                      StringUtils.split(line,null,3);
                handleMessage(directMsgTokens);
            }
            else if(command.equalsIgnoreCase("join")){
                handleJoin(tokens);
            }
            else if(command.equalsIgnoreCase("show")){
                handleShowList();
            }
            else if(command.equalsIgnoreCase("req")){
                handleRequest(tokens);
            }
            else if(command.equalsIgnoreCase("ac")){
                handleAccept(tokens);
            }
            else if(command.equalsIgnoreCase("leave")){
                handleLeave(tokens);
            }
            else if(command.equalsIgnoreCase("login")){

                handleLogin(outputStream,tokens);
            }
            else{
                String msg = "Unkown Command: " + command + "\n";
                outputStream.write(msg.getBytes());
            }

        }

    }


}

private void handleRegister(OutputStream outputStream, String[]
     tokens) throws IOException {
    if(tokens.length == 3){
        String username = tokens[1];
        String password =  tokens[2];


        User newUser = new User(username, password);

        boolean isValidId = true;

        for(User user: server.getValidUsers()){

            if(user.getUserName().equals(username) ||
                  user.getPassword().equals(password)){
                String error = "Username or password already
                      exit" + "\n";
                outputStream.write(error.getBytes());
                isValidId = false;

                System.err.println("Registration failed");
                break;
            }
        }

        if(isValidId){
            server.addUser(newUser);

            String msg = "Ok login" +"\n";
            outputStream.write(msg.getBytes());
            this.login = newUser.username;
            this.password = newUser.password;

            System.out.println("User logged in successfully :
                  "+login);

            ArrayList<ClientHandler> workerList =
                  server.getWorkerList();

            //send current user all other online users
            for(ClientHandler worker: workerList){

                if (worker.getLogin() != null &&
                      !login.equals(worker.getLogin())) {
                    String msg2 = "online " +
                          worker.getLogin() + "\n";
                    send(msg2);
                }



            }

            //send other online users current user's status
            String onlineMsg = "online " + login + "\n";
            for(ClientHandler worker : workerList){

                if(!login.equals(worker.getLogin())){
                    worker.send(onlineMsg);
                }
            }

        }


    }
}

private void handleLogin(OutputStream outputStream, String[]
     tokens) throws IOException {
    if(tokens.length == 3){
        String login = tokens[1];
        String password =  tokens[2];


        User reqUser = new User(login,password);



        for(User user: server.getValidUsers()){
            if(user.username.equals(reqUser.username) &&
                  user.password.equals(reqUser.password)){
                isOnline = true;
                break;
            }
        }

        if(isOnline){

            String msg = "Ok login" +"\n";
            outputStream.write(msg.getBytes());
            this.login = login;

            System.out.println("User logged in successfully :
                  "+login);

            ArrayList<ClientHandler> workerList =
                  server.getWorkerList();

            //send current user all other online users
            for(ClientHandler worker: workerList){

                if (worker.getLogin() != null &&
                      !login.equals(worker.getLogin())) {
                    String msg2 = "online " +
                          worker.getLogin() + "\n";
                    send(msg2);
                }



            }

            //send other online users current user's status
            String onlineMsg = "online " + login + "\n";
            for(ClientHandler worker : workerList){

                if(!login.equals(worker.getLogin())){
                    worker.send(onlineMsg);
                }
            }
        }else {
            String msg = "Error login " + login + "\n";
            outputStream.write(msg.getBytes());

            System.err.println("Login failed for: "+login);
        }
    }
}

String getLogin()
{
    return login;
}

private void send(String msg) throws IOException {
    if(login!=null)
    {
```

```
        outputStream.write(msg.getBytes());
    }

}

private void handleLogoff() throws IOException {
    server.remove(this);
    ArrayList<ClientHandler> workerList = server.getWorkerList();
    //send other online users current user's status
    String onlineMsg = "offline " + login + "\n";
    for (ClientHandler worker : workerList) {

        if (!login.equals(worker.getLogin())) {
            worker.send(onlineMsg);
        }
    }
    clientSocket.close();

}

private void handleMessage(String[] tokens) throws IOException {
    String reciever = tokens[1];
    String msg = tokens[2];


    boolean isGroupMsg = reciever.charAt(0) == '@';

    ArrayList<ClientHandler> workerList = server.getWorkerList();
    for(ClientHandler clientHandler: workerList)
    {
        if(isGroupMsg){

            if(clientHandler.isMemberOfGroupSet(reciever) &&
                  friendList.contains(reciever)){

                String outMessage = "msg " + reciever+":"+login +
                        " " + msg + "\n";
                System.out.println(outMessage);
                clientHandler.send(outMessage);
            }
        }
        else{

            if
                  (clientHandler.getLogin().equalsIgnoreCase(reciever
                  ) && friendList.contains(reciever)) {
                String outMessage = "msg " + login + " " + msg +
                        "\n";
                clientHandler.send(outMessage);
            }
        }
    }
}

private void handleJoin(String[] tokens) {
    if(tokens.length>1){
        String group = tokens[1];
        groupSet.add(group);
        System.out.println("You are added to "+group);
    }
}

private void handleShowList() throws IOException {

        String fred = "fred  ";
        String reqfred = "req ";

        for(String user: friendList){
            fred += user + " ";

        }

        fred += ":";

        for(String user: reqfriendlist){
            reqfred += user + " ";

        }

        reqfred += "\n";

        fred += reqfred;
        outputStream.write(fred.getBytes());


}


private boolean isMemberOfGroupSet(String group){

    return groupSet.contains(group);
}

private void handleLeave(String[] tokens) {
    if(tokens.length>1){
        String group = tokens[1];
        groupSet.remove(group);
    }
}
private void handleRequest(String[] tokens) {
    String user = tokens[1];

    ArrayList<ClientHandler> workers = server.getWorkerList();

    for(ClientHandler worker: workers){
        if (worker.getLogin().equalsIgnoreCase(user)) {
            String outMessage = "Request from " + login + "\n";
            worker.setFriendReq(login);
        }
    }


}
private void handleAccept(String[] tokens) throws IOException {
    String user = tokens[1];



    ArrayList<ClientHandler> workers = server.getWorkerList();

    for(ClientHandler worker: workers){
        if (worker.getLogin().equalsIgnoreCase(user)) {

            worker.setFriend(login);
            setFriend(user);
            removeFriendReq(user);
            break;
        }
    }



}


private void handleBroadcast(String[] tokens) throws IOException {

    String msg = tokens[1];



    ArrayList<ClientHandler> workerList = server.getWorkerList();
    for(ClientHandler clientHandler: workerList)
    {
        if (!clientHandler.getLogin().equalsIgnoreCase(login)) {
            String outMessage = "msg " + login + " " + msg + "\n";
            clientHandler.send(outMessage);
        }

    }
}

private void handleMulticast(String[] tokens,String line) throws
     IOException {
    int n = Integer.parseInt(tokens[1]);

    String[] users = new String[n];
    String[] multicastMsg = StringUtils.split(line, null, n + 3);
    String msg = multicastMsg[n + 2];

    for (int i = 0; i < n; i++) {
        users[i] = tokens[i + 2];

        ArrayList<ClientHandler> workerList =
              server.getWorkerList();
        for (ClientHandler clientHandler : workerList) {

            if
                  (clientHandler.getLogin().equalsIgnoreCase(users[i]))
                  {
                String outMessage = "msg " + login + " " + msg +
                        "\n";
                clientHandler.send(outMessage);
            }

        }
    }



}


public void setFriendReq(String user) {
    reqfriendlist.add(user);
}

public void setFriend(String user) {
    friendList.add(user);
}

public void removeFriendReq(String user) {
    reqfriendlist.remove(user);
```

```
    }

    public void removeFriend(String user) {
        friendList.remove(user);
    }
    public ArrayList<String> getFriendList(){
        return  friendList;
    }

    public ArrayList<String> getReqFriendList(){
        return  reqfriendlist;
    }

}
```

User.java

```
package server;


public class User {
    String username;
    String password;

    public User(String username,String password) {
        this.username = username;
        this.password = password;
    }

    public String getUserName(){
        return username;
    }

    public String getPassword(){
        return password;
    }
}
```

ServerMain.java

```
//Naimul Haque

package server;


public class ServerMain {
    public static void main(String[] args)  {

        int port = 9876;
        Server server = new Server(port);
        server.start();


    }
}
```

Client Package:

Client.java

```
package client;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.Socket;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;
import org.apache.commons.lang3.StringUtils;

public class Client {

    private String serverName;
    private int serverPort;
    private Socket socket;
    private OutputStream serverOut;
    private InputStream serverIn;
    private BufferedReader bufferedIn;

    private ArrayList<UserStatusListener> userStatusListeners = new
          ArrayList<>();
    private ArrayList<MessageListener> messageListeners = new
          ArrayList<>();
    private ArrayList<FriendListListener> friendListListeners = new
          ArrayList<>();

    public Client(String serverName,int serverPort)
{
    this.serverName =serverName;
    this.serverPort=serverPort;
}



boolean connect() {

    try {
        this.socket = new Socket(serverName,serverPort);
        this.serverOut = socket.getOutputStream();
        this.serverIn = socket.getInputStream();
        this.bufferedIn = new BufferedReader(new
              InputStreamReader(serverIn));
        return true;

    } catch (IOException ex) {
        Logger.getLogger(Client.class.getName()).log(Level.SEVERE,
              null, ex);
    }
    return false;
}

boolean login(String login, String password) throws IOException {
    String cmd = "login " + login + " " +password+"\n";

    serverOut.write(cmd.getBytes());

    String response = bufferedIn.readLine();
    System.out.println("Server response: "+response);

    if(response.equalsIgnoreCase("ok login")){
        startMessageReader();
        return true;
    }
    else{
        return false;
    }
}

void joinGroup(String groupName) throws IOException{
    String cmd = "join @" + groupName + "\n";


    serverOut.write(cmd.getBytes());

}

void leaveGroup(String groupName) throws IOException{
    String cmd = "leave @" + groupName + "\n";

    serverOut.write(cmd.getBytes());

}

boolean register(String login, String password) throws
      IOException {
    String cmd = "reg " + login + " " +password+"\n";

    serverOut.write(cmd.getBytes());

    String response = bufferedIn.readLine();
    System.out.println("Server response: "+response);

    if(response.equalsIgnoreCase("ok login")){
        startMessageReader();
        return true;
    }
    else{
        return false;
    }
}

public void logoff() throws IOException {
    String cmd = "logoff" +"\n";

    serverOut.write(cmd.getBytes());

}


private void startMessageReader() {

    Thread t = new Thread(){
        @Override
        public void run() {
            readMessageLoop();
        }



    };

    t.start();
}
```

```
private void readMessageLoop() {
    String line;

    try {
        while((line = bufferedIn.readLine())!=null){
            String[] tokens = StringUtils.split(line);
            if(tokens != null && tokens.length>0){
                String cmd = tokens[0];
                if(cmd.equalsIgnoreCase("online")){
                    handleOnline(tokens);
                }
                else if(cmd.equalsIgnoreCase("offline")){
                    handleOffline(tokens);
                }
                else if(cmd.equalsIgnoreCase("msg")){
                    String[] directMsgTokens =
                          StringUtils.split(line,null,3);
                    handleMessage(directMsgTokens);
                }
                else if(cmd.equalsIgnoreCase("fred")){
                    String[] list = line.split(":");
                    handleFriendList(list[0], list[1]);
                }
            }
        }
    } catch (IOException ex) {
        Logger.getLogger(Client.class.getName()).log(Level.SEVERE,
              null, ex);
        try {
            socket.close();
        } catch (IOException ex1) {
            Logger.getLogger(Client.class.getName()).log(Level.SEVERE,
                  null, ex1);
        }
    }
}

private void handleOnline(String[] tokens) {
    String login =  tokens[1];
    for(UserStatusListener listener: userStatusListeners){
        listener.online(login);
    }

}

private void handleOffline(String[] tokens) {

    String login =  tokens[1];
    for(UserStatusListener listener: userStatusListeners){
        listener.offline(login);
    }
}


private void handleMessage(String[] tokensMsg) {

    String login = tokensMsg[1];
    String msg = tokensMsg[2];

    for(MessageListener listener: messageListeners){
        listener.onMessage(login, msg);
    }

}

private void handleFriendList(String fred, String reqFred){

    for(FriendListListener listener: friendListListeners){
        listener.onFriendListShow(fred, reqFred);
    }
}

void sendBroadcast(String msg) throws IOException {
    String cmd = "broadcast "+msg+"\n";
    serverOut.write(cmd.getBytes());
}

void acceptReq(String user) throws IOException {
    String cmd = "ac "+user+"\n";
    serverOut.write(cmd.getBytes());
}

void sendMulticast(String[] users,int n,String msg) throws
     IOException {
    String multiUsers = "";

    for(String user : users){
        multiUsers += user +" ";
    }

    String cmd = "multicast "+n+" "+multiUsers+" "+msg+"\n";
    serverOut.write(cmd.getBytes());
}

void sendRequest(String reciever) throws IOException {
    String cmd = "req "+reciever+"\n";
    serverOut.write(cmd.getBytes());
}

void sendMessage(String reciever, String msg) throws IOException {
    String cmd = "msg "+reciever+" "+msg+"\n";
    serverOut.write(cmd.getBytes());
}

void showFriendList() throws IOException {
    String cmd = "show "+"\n";
    serverOut.write(cmd.getBytes());
}

public void addUserStatusListener(UserStatusListener listener){
    userStatusListeners.add(listener);

}



public void ramoveUserStatusListener(UserStatusListener listener){
    userStatusListeners.remove(listener);
}

public void addMessageListener(MessageListener listener){
    messageListeners.add(listener);
}


public void ramoveMessageListener(MessageListener listener){
    messageListeners.remove(listener);
}


public void addFriendListListener(FriendListListener listener){
    friendListListeners.add(listener);
}


public void ramoveFriendListListener(FriendListListener listener){
    friendListListeners.remove(listener);
}


}
```

UserMain.java

```
package client;

import com.sun.org.apache.xerces.internal.xs.PSVIProvider;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;


public class UserMain {


    public static void main(String[] args) throws IOException {
        Client client = new Client("localhost",9876);

        client.addUserStatusListener(new UserStatusListener() {
            @Override
            public void online(String login) {
                System.out.println("Online: "+login);
            }

            @Override
            public void offline(String login) {
                System.out.println("Offline: "+login);
            }
        });

        client.addMessageListener(new MessageListener() {
            @Override
            public void onMessage(String source, String msg) {
                System.out.println("Message from : "+source+" :
                      "+msg);


            }


        });

        client.addFriendListListener(new FriendListListener() {
            @Override
            public void onFriendListShow(String fred, String reqFred)
                  {
                System.out.println(fred);
                System.out.println(reqFred);
            }
        });
            if (!client.connect()) {
            System.err.println("Connection Failed.");
        } else {
            System.out.println("Connection Successful!");

            System.out.println("1. Login");
```

```
        System.out.println("2. Sign Up");

        BufferedReader reader = new BufferedReader(new
              InputStreamReader(System.in));

        String input1 = reader.readLine();

        if (Integer.parseInt(input1) == 1) {
            System.out.println("Enter you user name ");

            String userName = reader.readLine();

            System.out.println("Enter you password");

            String password = reader.readLine();

            if (client.login(userName, password)) {
                System.out.println("Login Successful");


                hangleLogin(client,reader);
                //client.sendMessage("naimul","hello world");
            } else {
                System.err.println("Login failed");
            }
        } else if (Integer.parseInt(input1) == 2) {

            System.out.println("Enter you user name ");

            String userName = reader.readLine();

            System.out.println("Enter you password");

            String password = reader.readLine();

            if (client.register(userName, password)) {
                System.out.println("Login Successful");

                hangleLogin(client, reader);

                //client.sendMessage("naimul","hello world");
            } else {
                System.err.println("Login failed");
            }
        }



       // client.logoff();
    }
}

static void hangleLogin(Client client,BufferedReader reader)
      throws IOException {

    boolean isOnline = true;

    while (isOnline) {
        System.out.println("1. Send direct message");
        System.out.println("2. Send broadcast message");
        System.out.println("3. Join Group");
        System.out.println("4. Leave Group");
        System.out.println("5. Group Chat");
        System.out.println("6. Multicast ");
        System.out.println("7. Quit or logoff");
        System.out.println("8. Send Request: ");
        System.out.println("9. Show Friendlist: ");
        System.out.println("10. Accept Friend req: ");

        String input2 = reader.readLine();
        switch (Integer.parseInt(input2)) {
            case 1:
                System.out.print("To: ");
                String user = reader.readLine();

                System.out.print("Type msg: ");
                String msg = reader.readLine();

                client.sendMessage(user, msg);
                break;
            case 2:
                System.out.print("Type msg: ");
                client.sendBroadcast(reader.readLine());

                break;
            case 3:
                System.out.print("Join to: ");
                String group = reader.readLine();
                client.joinGroup(group);

                System.out.println("You are Joined to @"+group);
                break;
            case 4:
                System.out.print("Group to leave : ");
                String leavegroup = reader.readLine();
                client.leaveGroup(leavegroup);

                System.out.println("You are removed from
                      @"+leavegroup);
                break;
            case 5:
                System.out.print("Type Group Name: ");
                String groupName = reader.readLine();

                groupName = "@"+groupName;

                System.out.print("Type msg: ("+groupName+"): ");
                String msg3 = reader.readLine();
                client.sendMessage(groupName, msg3);

                break;
            case 6:
                System.out.print("Type Number of users: ");

                int n = Integer.parseInt(reader.readLine());

                String[] users = new String[n];

                System.out.println("Enter the users: ");

                for(int i = 0; i < n ; i++){
                    String username = reader.readLine();

                    users[i] = username;
                }

                System.out.println("Type message: ");

                String msg2 = reader.readLine();

                client.sendMulticast(users, n, msg2);
                break;
            case 7:
                client.logoff();
                isOnline = false;
                break;
            case 8:
                System.out.print("Request user: ");
                String reqUser = reader.readLine();
                client.sendRequest(reqUser);

                System.out.println("A request sent to "+reqUser);

                client.showFriendList();
                break;

            case 9:

                client.showFriendList();
                break;
            case 10:
                client.showFriendList();

                System.out.print("Type username: ");

                String acFrd = reader.readLine();
                client.acceptReq(acFrd);
                break;
        }

    }
}
```

MessageListener.java

```
package client;

public interface MessageListener {
    public void onMessage(String source,String msg);

}
```

FriendListListener.java

```
/*
 * To change this license header, choose License Headers in Project
      Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package client;

/**
 *
 * @author User
 */
public interface FriendListListener {
    public void onFriendListShow(String fred,String reqFred);
}
```

UserStatusListener.java

```
package client;

public interface UserStatusListener {
    public void online(String login);
    public void offline(String login);

}
```