CSE 4226: Network Programming LAB

Assignment 2: Developing A Multi-Threaded Server-Slient Application

Evaluation Phase - I Report

Lab Group: A2 **Name:** Asif Isthiaq **Id:** 14.02.04.034

5 June, 2018

Contents

1 Implementation Summery

Features	Status
(i) User Register and Login	Implemented
(ii) Online User List	Implemented
(iii) Friend Request	Implemented
(iv) Unicast	Implemented
(v) Multicast	Implemented
(vi) Broadcast	Implemented

Table 1: Features

2 Implementation Challenges

2.1 User Register and Login

To join the system, a user requires UserName and Password (UserName:Password).If the user already exists he or she will be logged in . If not new user will be registered.

2.2 Online User Lists

After logging on, a user can see the online user list.(cmd:show$_list$)

2.3 Friend Request

A user can send friend request to other users. The request can be accepted or rejected. A user can have his/her friend list(add$_friend : username, accept : username, deny : username$)

2.4 Unicast

A user can send message to any other online user.(msg:typeMsg:reciver)

2.5 Multicast

A user can send message to a group of online users (for this feature send the message as msg:first-user:second-user:third-user and the message will be delivered to the users separated by ":")(msg:typeMsg:reciver1:reciver2)

2.6 Broadcast

A user can send message to all the online users.(broadcast:typeMsg)

3 Interaction Diagram

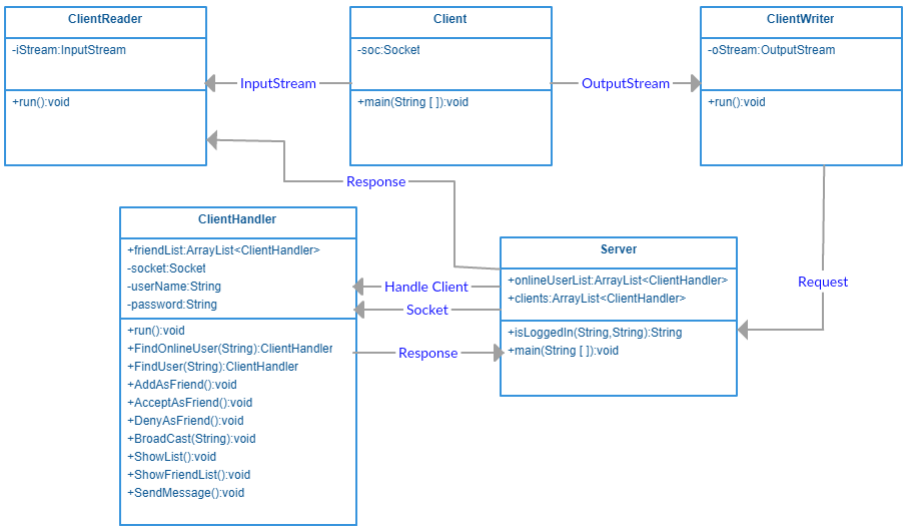

Figure 1: Interaction Diagram

4 Limitation and Future Scope of Improvement

I have implemented all the features.Anyone without knowing the code correctly can't use tha application.If we user GUI it will make the user experience better.

5 Discussion

The aim of this assignment is to create a simple multi-threaded server-client application and that will provide chat list system and add friend list.

A Source Codes

```
1
2 import java.io.*;
```

```
3  import java.net.*;
4  import java.util.*;
5
6  public class Server {
7      public static ArrayList <ClientHandler> clients = new ArrayList<>();
8      public static ArrayList<ClientHandler> onlineUserList = new
       ArrayList<ClientHandler>();
9
10     public static String isLoggedin(String userName, String password){
11         ClientHandler c=null;
12
13         for(int i=0;i<Server.clients.size();i++){
14             if(clients.get(i).getUsername().equals(userName)
15                             && clients.get(i).getPassword().equals(
       password)){
16                 c=clients.get(i);
17                 break;
18             }
19         }
20         if(c==null){
21             return "User doesnt exist";
22         }
23         else{
24             return "Logged in";
25         }
26     }
27
28     public static void main(String args[]){
29         try{
30             ServerSocket sSocket = new ServerSocket(7777);
31             while(true){
32                 Socket cSocket = sSocket.accept();
33                 InputStreamReader inFromClient = new InputStreamReader(
       cSocket.getInputStream());
34                 BufferedReader in = new BufferedReader(inFromClient);
35                 DataOutputStream out = new DataOutputStream(cSocket.
       getOutputStream());
36                 String msg = in.readLine();
37                 System.out.println(msg);
38                 String arr[] = msg.split(":");
39
40                 if(arr[0].equals("sign_in")){
41                     String log = isLoggedin(arr[1], arr[2]);
42                     out.writeBytes(log + '\n');
43                     while(!log.equals("Logged in")){
44                         msg = in.readLine();
45                         System.out.println(msg);
46                         arr = msg.split(":");
```

```
                        if(arr[0].equals("sign_in")){
                            log = isLoggedin(arr[1], arr[2]);
                            out.writeBytes(log + '\n');
                            ClientHandler tempClient=null;
                            for(int i=0;i<Server.clients.size();i++){
                                if(clients.get(i).getUsername().equals(arr[1])
                                        && clients.get(i).getPassword().equals(arr[2])){
                                    tempClient=clients.get(i);
                                    break;
                                }
                            }
                            onlineUserList.add(tempClient);
                            Thread th = new Thread(tempClient);
                            th.start();
                        }
                        else if(arr[0].equals("sign_up")){
                            ClientHandler ch = new ClientHandler(arr[1], arr[2],cSocket);
                            clients.add(ch);
                            System.out.println("Registered " + arr[1]);
                            log = "not logged";
                        }
                    }
                }
            }

        }catch(Exception e){

        }
    }
}
```

Listing 1: Server Class

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

public class ClientHandler implements Runnable{
    private String username;
    private String password;
    private Socket socket;

    public ArrayList <ClientHandler> friendList = new ArrayList<>();

    public ClientHandler(String username, String password, Socket socket
```

```
    ) {
13          this.username = username;
14          this.password = password;
15          this.socket = socket;
16      }
17
18      public void run(){
19          try{
20              System.out.println(username + " logged in");
21              InputStreamReader inFromClient = new InputStreamReader(
    socket.getInputStream());
22              BufferedReader in = new BufferedReader(inFromClient);
23              while(true){
24                  String str = in.readLine();
25                  String arr[] = str.split(":");
26
27                  if(arr[0].equals("cmd") && arr[1].equals("logout")){
28                      break;
29                  }
30                  else if(arr[0].equals("cmd") && arr[1].equals("show_list
    ")){
31                      ShowList();
32                  }
33                  else if(arr[0].equals("cmd") && arr[1].equals("
    show_friend_list")){
34                      ShowFriendList();
35                  }
36                  else if(arr[0].equals("cmd") && arr[1].equals("
    add_to_friend_list")){
37                      ClientHandler ch = FindUser(arr[2]);
38                      if(ch!=null){
39                          friendList.add(ch);
40                      }
41                  }
42                  else if(arr[0].equals("add_friend")){
43                      AddAsFriend(arr[1]);
44                  }
45                  else if(arr[0].equals("accept")){
46                      AcceptAsFriend(arr[1]);
47                  }
48                  else if(arr[0].equals("deny")){
49                      DenyAsFriend(arr[1]);
50                  }
51                  if(arr[0].equals("msg")){
52                      SendMessage(arr);
53                  }
54                  else if(arr[0].equals("broadcast")){
55                      Broadcast(arr[1]);
```

```
56                  }
57              }
58              socket.close();
59          }catch(Exception e){
60
61          }
62      }
63
64      public ClientHandler FindOnlineUser(String str){
65          ClientHandler ch=null;
66          for(int i=0;i<Server.onlineUserList.size();i++){
67              if(Server.onlineUserList.get(i).getUsername().equals(str)){
68                  ch=Server.onlineUserList.get(i);
69                  break;
70              }
71          }
72          if(ch==null){
73              return null;
74          }
75          else{
76              return ch;
77          }
78      }
79
80      public ClientHandler FindUser(String str){
81          ClientHandler ch=null;
82          for(int i=0;i<Server.clients.size();i++){
83              if(Server.clients.get(i).getUsername().equals(str)){
84                  ch=Server.clients.get(i);
85                  break;
86              }
87          }
88          if(ch==null){
89              return null;
90          }
91          else{
92              return ch;
93          }
94      }
95
96      public void AddAsFriend(String str){
97          try{
98              ClientHandler ch = FindUser(str);
99              DataOutputStream out = new DataOutputStream(ch.getSocket().
    getOutputStream());
100             out.writeBytes("Friend Request:" + this.getUsername() + '\n'
    );
101         }catch(Exception e){
```

```
102
103         }
104     }
105
106     public void AcceptAsFriend(String str){
107         try{
108             ClientHandler ch = FindUser(str);
109             DataOutputStream out = new DataOutputStream(ch.getSocket().
    getOutputStream());
110             out.writeBytes("Friend Request Accepted By:" + this.
    getUsername() + '\n');
111             friendList.add(ch);
112             ch.friendList.add(this);
113         }catch(Exception e){
114
115         }
116     }
117     public void DenyAsFriend(String str){
118         try{
119             ClientHandler ch = FindUser(str);
120             DataOutputStream out = new DataOutputStream(ch.getSocket().
    getOutputStream());
121             out.writeBytes("Friend Request Denied By:" + this.
    getUsername() + '\n');
122         }catch(Exception e){
123
124         }
125     }
126
127     public void Broadcast(String str){
128         try{
129             DataOutputStream out;
130             for(int i=0; i<Server.onlineUserList.size(); i++){
131                 if(this!=Server.onlineUserList.get(i)){
132                     out = new DataOutputStream(Server.onlineUserList.get
    (i).getSocket().getOutputStream());
133                     out.writeBytes(this.getUsername()+ ":" + str + '\n')
    ;
134                 }
135             }
136         }catch(Exception e){
137
138         }
139     }
140
141     public void ShowList(){
142         try{
143             DataOutputStream out = new DataOutputStream(this.getSocket()
```

```
    .getOutputStream());
144             String userListStr="";
145             for(int i=0;i<Server.onlineUserList.size();i++){
146                     userListStr= userListStr+Server.onlineUserList.get(i
    ).getUsername()+"  "+(i+2)+".";
147             }
148             out.writeBytes("Online Users:"+"1."+userListStr.substring(0,
    userListStr.length()-2)+'\n');
149
150         }catch(Exception e){
151
152         }
153     }
154
155     public void ShowFriendList(){
156         try{
157             DataOutputStream out = new DataOutputStream(this.getSocket()
    .getOutputStream());
158             String userListStr="";
159             for(int i=0;i<friendList.size();i++){
160                     userListStr= userListStr+friendList.get(i).
    getUsername()+"  "+(i+2)+".";
161             }
162             out.writeBytes("Friend List:"+"1."+userListStr.substring(0,
    userListStr.length()-2)+'\n');
163
164         }catch(Exception e){
165
166         }
167     }
168
169     public void SendMessage(String s[]){
170         int len = s.length;
171         try{
172             DataOutputStream out;
173             for(int i=2; i<len; i++){
174                 ClientHandler ch = FindOnlineUser(s[i]);
175                 if(ch==null){
176                     return;
177                 }
178                 out = new DataOutputStream(ch.getSocket().
    getOutputStream());
179                 out.writeBytes(this.getUsername()+ ":" + s[1] + '\n');
180             }
181         }catch(Exception e){
182
183         }
184     }
```

```
185
186     public String getUsername() {
187         return username;
188     }
189
190     public void setUsername(String username) {
191         this.username = username;
192     }
193
194     public String getPassword() {
195         return password;
196     }
197
198     public void setPassword(String password) {
199         this.password = password;
200     }
201
202     public Socket getSocket() {
203         return socket;
204     }
205
206     public void setSocket(Socket socket) {
207         this.socket = socket;
208     }
209
210 }
```

Listing 2: Client Handler Thread

```
1  import java.io.*;
2  import java.net.*;
3  import java.util.*;
4
5  public class Client {
6      public static void main(String args[]){
7          try{
8              BufferedReader keyRead = new BufferedReader(new
       InputStreamReader(System.in));
9              Socket s = new Socket();
10             Socket soc = new Socket("localhost", 7777);
11             String str;
12             while(true){
13                 System.out.println("UserName:Password->");
14                 str = keyRead.readLine();
15                 DataOutputStream out = new DataOutputStream(soc.
       getOutputStream());
16                 InputStreamReader inStream = new InputStreamReader(soc.
       getInputStream());
```

```
                BufferedReader in = new BufferedReader(inStream);
                out.writeBytes("sign_in:" + str +'\n');
                String response = in.readLine();
                if(response.equals("Logged in")){
                    System.out.println("SuccessFully Logged In.");
                    break;
                }
                else if(response.equals("User doesnt exist")){
                    out.writeBytes("sign_up:" + str + '\n');
                    System.out.println("Registation Procedure Complete."
    );
                }
            }
            ClientReader cr = new ClientReader(soc.getInputStream());
            Thread th = new Thread(cr);
            th.start();
            ClientWriter cw = new ClientWriter(soc.getOutputStream());
            Thread th2 = new Thread(cw);
            th2.start();
        }catch(Exception e){

        }
    }
}
```

Listing 3: Client Class

```
import java.io.*;

public class ClientReader implements Runnable{
    private InputStream stream;

    public ClientReader(InputStream stream) {
        this.stream = stream;
    }
    public void run(){
        InputStreamReader inStream = new InputStreamReader(stream);
        BufferedReader in = new BufferedReader(inStream);
        while(true){
            try{
                String str = in.readLine();
                System.out.println(str);
            }catch(Exception e){

            }
        }
    }

```

```
22      public InputStream getStream() {
23          return stream;
24      }
25
26      public void setStream(InputStream stream) {
27          this.stream = stream;
28      }
29  }
```

Listing 4: Client Reader Thread

```
1   import java.io.*;
2   import java.net.*;
3   import java.util.*;
4
5   public class ClientWriter implements Runnable{
6       private OutputStream stream;
7
8       public ClientWriter(OutputStream stream) {
9           this.stream = stream;
10      }
11
12      public void run(){
13          BufferedReader keyRead = new BufferedReader(new
        InputStreamReader(System.in));
14          DataOutputStream out = new DataOutputStream(stream);
15          try{
16              while(true){
17                  String msg = keyRead.readLine();
18                  out.writeBytes(msg + '\n');
19              }
20          }catch(Exception e){
21
22          }
23      }
24
25      public OutputStream getStream() {
26          return stream;
27      }
28
29      public void setStream(OutputStream stream) {
30          this.stream = stream;
31      }
32  }
```

Listing 5: Client Writer Thread