Contents

1 Objective

The aim of this assignment is to create a simple multi-threaded server-client application where multiple users can exchange message or text with each other with some additional features.

2 Implementation Summary

The following table shows the implementation summary of the assignment:

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

Table 1: Implementation Summary

3 Implementation Challenges

During the implementation of the assignment i had faced several challenges which are described in the following subsection.

3.1 User Register and Login

To join the system, an existing user have to log in and a new user must have to register. For Login, user have to write (login). Then, the window will prompt the user to input the username and password which will be sent to the server as a string such as (username:password). For Registration, user have to write (sigup). Then the window will prompt the user to input Full name, Username and Password, which are sent to the server as a string such as (username:password:name). Some challenges i had faced in this phase are: checking the validity of the username and password for login, checking whether an user is trying to register with an username that already exists, storing the information in a file.

3.2 Online User Lists

To view the online user lists, an user must have to be logged on to the system. To view the list, user first have to write the command (Online User Lists). This command is sent to the server as a String and the server processes the request to the client. This is quite challenging for me. For this i have maintained an array list, where the logged in users are stored. So when the command executes, the members of the array list are shown to the corresponding client.

3.3 Friend Request

In the system, one can send friend request and accept friend request. For sending friend request, user have to write the command (Send Request), then the window will prompt the user to input the name of the person whom he/she wants to send friend request. The sent request will appear to the other user as a notification for friendship. For accepting a friend request, user have to write the command (Accept Request), and then he/she have to write the name of the person whose friend request he/she wants to accept. This is the most challenging part i had faced in the whole assignment.

3.4 Unicast

This means in the system, an user can send message to only one recipient but the condition is that the recipient must belong to his/her friend list. For this challenging task, i have to check the friendship before sending a unicast message. For sending unicast message, user must write the command (Unicast) which will prompt the user to give input the name of the recipient and the message.

3.5 Multicast

That means in the system, an user can send message to more than one recipient but the condition is that all the recipients must belong to his/her friend list. For this challenging task, i have to check the friendship of all the recipients with the user before sending a multicast message. For sending multicast message, user must write the command (Multicast) which will prompt the user to give input the name of the recipients and the message. The name of multiple recipients must be written as (name1:name2:name3).

3.6 Broadcast

This phase is quite easy. Broadcasting means user sends a message, all his/her friends receive that message. For this task, user must write the command (Broadcast) and enter the message. So all the friends of the user will get the message.

4 Interaction Diagram

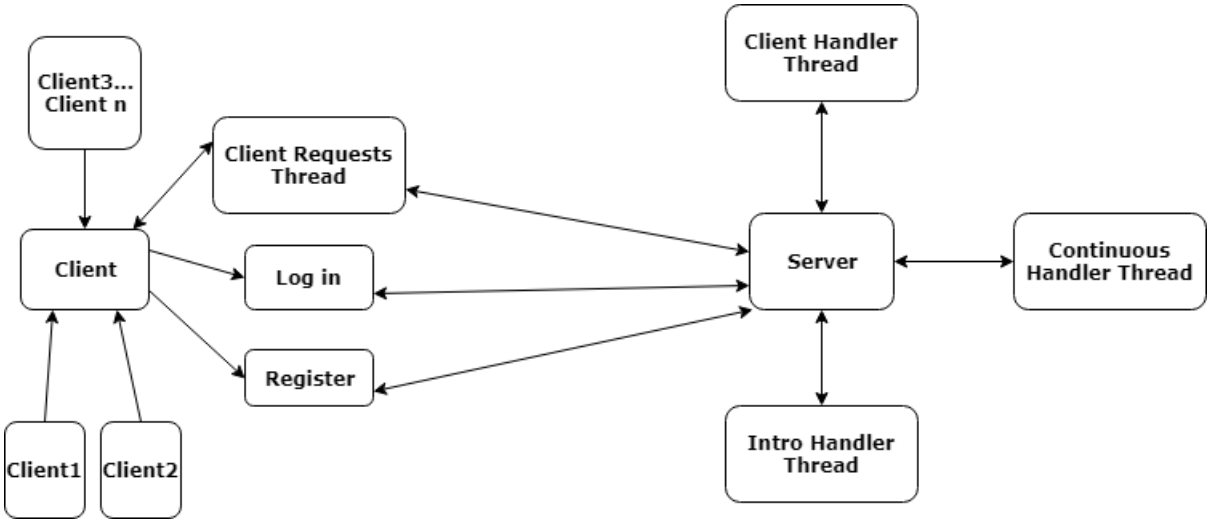

Figure 1: Interaction Diagram of the system.

5 Limitation and Future Scope of Improvements

Online users are stored in array list, therefore for a significant number of users it is not efficient. I have stored all the information in files. There is high risk for file manipulation, hence the data is not safe. However, in the implementation part, many more conditions can arise and much cautions can be taken to avoid corner cases. I have implemented the first six features. Last two features are not implemented yet. In future, i will try to implement all the features and provide more flexibility both in implementation and user interaction.

6 Discussion

From the implementation of the assignment, i have learned how client and server communicates, connection establishment, multi-threading, client side programming, server side programming, multi-threaded server client application etc. This assignment is helpful for the better understanding of server client application and interaction.

7 Appendix

```
1  package client;
2  import java.io.BufferedReader;
3  import java.io.DataOutputStream;
4  import java.io.IOException;
5  import java.io.InputStreamReader;
6  import java.net.Socket;
7  import java.util.Scanner;
8  import java.util.logging.Level;
9  import java.util.logging.Logger;
10
11 /* @author Shibli */
12
13 public class Client
14 {
15     // Method for user registration
16     public static boolean signUp( String name, String uname, String pass )
17     {
18         try
19         {
20             // Create Client Socket.
21             Socket cSocket = new Socket( "localhost", 4444 );
22
23             //input & output stream with wrapper.
24             BufferedReader inFromServer = new BufferedReader(new InputStreamReader(
    cSocket.getInputStream()));
25             DataOutputStream outToServer = new DataOutputStream(cSocket.getOutputStream
    ());
26
27             //Send to Server the Command for Registration.
28             outToServer.writeBytes("signup" + '\n');
29
30             //Convert the name,username & password into a string & send to server.
31             String full_msg =  uname + ":" + pass + ":" + name;
32             outToServer.writeBytes(full_msg + '\n');
33
34             // Acknowledgement from Server.
35             String ack = inFromServer.readLine();
36
37             // If acknowledgement matched.
38             if( ack.equals("ok") )
39             {
40                 // Print Success message.
41                 System.out.println("Registration Successful.\n");
42
43                 // Two thread operations: Send & Receive for one user.
44                 ClientRequests client1 = new ClientRequests(cSocket, "send");
45                 client1.start();
46                 ClientRequests client2 = new ClientRequests(cSocket, "receive" );
47                 client2.start();
48
49                 return true;
```

```
50              }
51              else
52              {
53                  // Print Unsuccess message.
54                  System.out.println("Unsuccessful. Please Try Again.\n");
55                  return false;
56              }
57          }
58          catch (IOException ex)
59          {
60              Logger.getLogger(Client.class.getName()).log(Level.SEVERE, null, ex);
61          }
62          return false;
63      }
64
65      // Method for user Login.
66      public static boolean signIn( String uname, String pass )
67      {
68          try
69          {
70              // Create Client Socket.
71              Socket clientSocket = new Socket( "localhost", 4444 );
72
73              //input & output stream with wrapper.
74              BufferedReader inFromServer = new BufferedReader(new InputStreamReader(
    clientSocket.getInputStream()));
75              DataOutputStream outToServer = new DataOutputStream(clientSocket.
    getOutputStream());
76
77              //Send to Server the command for Login.
78              outToServer.writeBytes("login" + '\n');
79
80              //Convert user name & password into a String and send to server.
81              String full_msg = uname + ":" + pass;
82              outToServer.writeBytes(full_msg + '\n');
83
84              // Acknowledgement from Server.
85              String ack = inFromServer.readLine();
86
87              // If Acknwledgement Matched.
88              if( ack.equals("ok") )
89              {
90                  // Print Success message.
91                  System.out.println("Login Successful.\n");
92
93                  // Two thread operations: Send & Receive for one user.
94                  ClientRequests client1 = new ClientRequests(clientSocket, "send" );
95                  client1.start();
96                  ClientRequests client2 = new ClientRequests(clientSocket, "receive" );
97                  client2.start();
98
99                  return true;
100             }
101             else
102             {
103                 //Print Unsuccessful Message.
104                 System.out.println("Login Unsuccessful. Try Again\n");
105                 return false;
106             }
107         }
108         catch (IOException ex)
109         {
110             Logger.getLogger(Client.class.getName()).log(Level.SEVERE, null, ex);
111         }
112         return false;
113     }
114
115     public static void main(String[] args)
116     {
117         //User Prompt.
118         System.out.println( "Login? type login, or Register? type signup.\n");
119
120         // Scanner class to string input.
121         Scanner sc = new Scanner( System.in  );
122         String ss  = sc.next();
```

```
123
124         //For login purpose.
125         if( ss.equals("login") )
126         {
127             System.out.print("Your User Name: ");
128             String uname = sc.next();
129             System.out.println();
130             System.out.print("Your Password: ");
131             String pass = sc.next();
132             System.out.println();
133
134             //If fails continue to scan username and password.
135             while( !signIn(uname, pass ) )
136             {
137                 System.out.print("Your User Name: ");
138                 uname = sc.next();
139                 System.out.println();
140                 System.out.print("Your Password: ");
141                 pass = sc.next();
142                 System.out.println();
143             }
144         }
145         else //For registration purpose.
146         {
147
148             System.out.print("Your Full Name: ");
149             sc.nextLine();
150             String name = sc.nextLine();
151             System.out.println();
152             System.out.print("Your User Name: ");
153             String uname = sc.next();
154             System.out.println();
155             System.out.print("Your Password: ");
156             String pass = sc.next();
157             System.out.println();
158
159             //If fails continue to scan username and password.
160             while( !signUp(name, uname, pass ) )
161             {
162                 System.out.print("Enter Your Name: ");
163                 sc.nextLine();
164                 name = sc.nextLine();
165                 System.out.println();
166                 System.out.print("Enter UserName: ");
167                 uname = sc.next();
168                 System.out.println();
169                 System.out.print("Enter Password: ");
170                 pass = sc.next();
171                 System.out.println();
172             }
173         }
174     }
175 }
```

Listing 1: Client.java

```java
package client;
import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.ObjectInputStream;
import java.net.Socket;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;

/* @author Shibli */

public class ClientRequests extends Thread
{
    public Socket cSocket;
    String task;

    //Constructor for the class.
    public ClientRequests(Socket cSocket, String task)
    {
        this.cSocket = cSocket;
        this.task = task;
    }

    //Run method for the thread.
    public void run()
    {
        if( "send".equals(task) )
        {
            while( true )
            {
                try
                {
                    //input & output stream with wrapper.
                    BufferedReader inFromClient = new BufferedReader( new InputStreamReader( System.in ) );
                    DataOutputStream outToServer = new DataOutputStream(cSocket.getOutputStream());

                    //Show Task list.
                    System.out.println("Tasks you can do:");
                     System.out.println("1. Online User Lists");
                      System.out.println("2. Accept Request");
                       System.out.println("3. Send Request");
                        System.out.println("4. Unicast");
                         System.out.println("5. Multicast");
                          System.out.println("6. Broadcast");
                           System.out.println("7. Log out");

                    //Command from Client.
                    String coomand = inFromClient.readLine();
                    System.out.println("");

                    String k = "";
                    if( coomand.equals("Online User Lists")) //online user lists
                    {
                        k = "Online User Lists" + '\n';
                        outToServer.writeBytes(k); // send command to server.
                    }
                    else if( coomand.equals("Accept Request") ) //accept request
                    {
                        k = "Accept Request" + '\n';
                        outToServer.writeBytes(k); // send command to server.

                        System.out.print("Accept Request of the user: ");
                        String name = inFromClient.readLine();
                        System.out.println();
                        outToServer.writeBytes(name + '\n'); //send name to server.

                    }
                    else if( coomand.equals("Send Request") ) //send request
                    {
                        k = "Send Request" + '\n';
                        outToServer.writeBytes(k); //send command to the server.
```

```
74
75                      System.out.print("Send Request to the user: ");
76                      String name = inFromClient.readLine();
77                      System.out.println();
78                      outToServer.writeBytes(name + '\n'); //send name to the server.
79
80                  }
81                  else if( coomand.equals("Unicast") ) //unicast
82                  {
83                      k = "Unicast" + '\n';
84                      outToServer.writeBytes(k);
85
86                      System.out.print("Recipient: ");
87                      String name = inFromClient.readLine();
88                      System.out.println("");
89                      outToServer.writeBytes(name + '\n');
90
91                      System.out.print("Message: ");
92                      String msg = inFromClient.readLine();
93                      System.out.println("");
94                      outToServer.writeBytes(msg + '\n');
95
96                  }
97                  else if(coomand.equals("Multicast") ) //multicast
98                  {
99                      k = "Multicast" + '\n';
100                     outToServer.writeBytes(k);
101
102                     System.out.print("Recipient: ");
103                     String name = inFromClient.readLine();
104                     System.out.println("");
105                     outToServer.writeBytes(name + '\n');
106
107                     System.out.print("Message: ");
108                     String msg = inFromClient.readLine();
109                     System.out.println("");
110                     outToServer.writeBytes(msg + '\n');
111
112                 }
113                 else if(coomand.equals("Broadcast") ) //broadcast
114                 {
115                     k = "Broadcast" + '\n';
116                     outToServer.writeBytes(k);
117
118                     System.out.print("Message: ");
119                     String msg = inFromClient.readLine();
120                     System.out.println("");
121                     outToServer.writeBytes(msg + '\n');
122                 }
123                 else if( "Log out".equals( coomand ) )
124                 {
125                     k = "Log out" + '\n';
126                     outToServer.writeBytes(k);
127                     break;
128                 }
129             }
130             catch (IOException ex)
131             {
132                 Logger.getLogger(ClientRequests.class.getName()).log(Level.SEVERE,
    null, ex);
133             }
134         }
135     }
136     else if( "receive".equals(task) )
137     {
138         try
139         {
140             while( true )
141             {
142                 //input stream & object input stream for arraylist passing.
143                 BufferedReader inFromServer = new BufferedReader(new
    InputStreamReader(cSocket.getInputStream()));
144
145                 // Server sends command.
146                 String sentence = inFromServer.readLine();
```

```
147
148                     // Time to show the online user lists.
149                     if( sentence.equals("Online User Lists") )
150                     {
151                         ObjectInputStream objectInput = new ObjectInputStream(cSocket.
    getInputStream());
152                         try
153                         {
154                             Object object = objectInput.readObject();
155                             ArrayList<String> listusers = new ArrayList<String>();
156                             listusers =  (ArrayList<String>) object;
157                             for( String i: listusers )System.out.println(i);
158                         }
159                         catch (ClassNotFoundException ex)
160                         {
161                             Logger.getLogger(ClientRequests.class.getName()).log(Level.
    SEVERE, null, ex);
162                         }
163                     }
164                     else System.out.println(sentence);
165                 }
166             }
167             catch (IOException ex)
168             {
169                Logger.getLogger(ClientRequests.class.getName()).log(Level.SEVERE, null,
    ex);
170             }
171         }
172     }
173 }
```

Listing 2: ClientRequests.java

```
1  package server;
2  import java.io.IOException;
3  import java.net.ServerSocket;
4  import java.net.Socket;
5  import java.util.ArrayList;
6  import java.util.logging.Level;
7  import java.util.logging.Logger;
8
9  /* @author Shibli */
10
11 public class Server
12 {
13     // Arraylist to store the online users
14     public static ArrayList<ClientHandler> cLists = new ArrayList<ClientHandler>();
15
16     public static void main(String[] args)
17     {
18         try
19         {
20             // Create Server Socket.
21             ServerSocket sSocket = new ServerSocket(4444);
22
23             //This thread will always keep checking
24             ContinuousHandler obj = new ContinuousHandler();
25             obj.start();
26
27             // Continuously accept new client if get one.
28             while( true )
29             {
30                 Socket connectionSocket = sSocket.accept();
31                 IntroHandler newlogIn = new IntroHandler(connectionSocket);
32                 newlogIn.start();
33             }
34         }
35         catch (IOException ex)
36         {
37             Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);
38         }
39     }
40 }
```

Listing 3: Server.java

```java
package server;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.DataOutputStream;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.ObjectOutputStream;
import java.net.Socket;
import java.util.ArrayList;
import java.util.Scanner;
import java.util.StringTokenizer;
import java.util.logging.Level;
import java.util.logging.Logger;
import static server.Server.cLists;

/* @author Shibli */

public class ClientHandler extends Thread
{
    public  Socket cSocket;
    public  String cName;
    public  String uname;

    //Constructor for ClientHandler class.
    public ClientHandler( Socket cSocket, String cName, String uname )
    {
        this.cSocket = cSocket;
        this.cName = cName;
        this.uname = uname;
    }
    // Mehthod to write the accepted friends name in text file
    void addFriends( String frdname )throws IOException
    {
        BufferedWriter writer = new BufferedWriter(new FileWriter("friends.txt", true));
        writer.write(uname + ":" + frdname );
        writer.newLine();
        writer.flush();
        writer.close();
    }

    //Method to store friend requests
    void addFrinedRequest( String frdname )throws IOException
    {
        BufferedWriter writer = new BufferedWriter(new FileWriter("requests.txt", true))
    ;
        writer.write("send:" + uname + ":" + frdname );
        writer.newLine();
        writer.flush();
        writer.close();
    }

    //Method to send Confirmation
    void requestConfirmation( String name ) throws IOException
    {
        BufferedWriter writer = new BufferedWriter(new FileWriter("notifications.txt",
    true));
        writer.write("send:" + name + ":" + uname + " is your friend now.");
        writer.newLine();
        writer.flush();
        writer.close();
    }

    //Method for messaging
     void giveMSG( String name, String msg )throws IOException
     {
        BufferedWriter writer = new BufferedWriter(new FileWriter("message.txt", true));
        StringTokenizer tokens = new StringTokenizer(name,":");
        while (tokens.hasMoreTokens())
        {
            String next = tokens.nextToken();
            if( !checkFriendship(uname, next ) )continue;
            writer.write("send:" + uname + ":" + next + ":" + msg  );
            writer.newLine();
```

```
74          }
75          writer.flush();
76          writer.close();
77      }
78
79       //Method for Checking Friendship
80      boolean checkFriendship( String user1, String user2 )throws IOException
81      {
82          FileReader inputFile = null;
83          inputFile = new FileReader("friends.txt");
84          Scanner parser = new Scanner(inputFile);
85
86          while (parser.hasNextLine())
87          {
88              String line = parser.nextLine();
89              if( "".equals(line) )continue;
90              StringTokenizer tokens = new StringTokenizer( line, ":" );
91              String f = tokens.nextToken();
92              String s = tokens.nextToken();
93              if( f.equals(user1) && s.equals(user2) )return true;
94              else if( s.equals(user1) && f.equals(user2) )return true;
95          }
96          return false;
97      }
98      // Method for broadcast messaging.
99      void broadcasting(String msg )throws IOException
100     {
101         BufferedWriter writer = new BufferedWriter(new FileWriter("message.txt", true));
102         FileReader inputFile = null;
103         inputFile = new FileReader("friends.txt");
104         Scanner parser = new Scanner(inputFile);
105
106         while (parser.hasNextLine())
107         {
108             String line = parser.nextLine();
109             if( "".equals(line) )continue;
110             StringTokenizer tokens = new StringTokenizer( line, ":" );
111             String f = tokens.nextToken();
112             String s = tokens.nextToken();
113             if( f.equals(uname) )
114             {
115                 writer.write("send:" + f + ":" + s + ":" + msg  );
116                 writer.newLine();
117             }
118             else if( s.equals(uname) )
119             {
120                 writer.write("send:" + s + ":" + f + ":" + msg  );
121                 writer.newLine();
122             }
123         }
124         writer.flush();
125         writer.close();
126     }
127
128     public void delUser()
129     {
130         for( ClientHandler i: cLists )
131         {
132             if( i.uname.equals(uname) )
133             {
134                 cLists.remove(i);
135                 break;
136             }
137         }
138     }
139
140     //Run Method for the thread.
141     public void run()
142     {
143         try
144         {
145            while( true )
146            {
147                DataOutputStream outToClient = new DataOutputStream(cSocket.
    getOutputStream() );
```

```
148                 BufferedReader inFromClient = new BufferedReader( new InputStreamReader(
    cSocket.getInputStream()));
149                 String cmd = inFromClient.readLine();
150 
151                 if( "Online User Lists".equals(cmd))
152                 {
153                     outToClient.writeBytes("Online User Lists" + '\n' );
154                     ArrayList<String> allUsers = new ArrayList<String>();
155                     for( ClientHandler i: cLists )
156                     {
157                         String names =  "Name : " + i.cName + ", UserName : " + i.uname ;
158                         allUsers.add( names );
159                     }
160                     ObjectOutputStream objectOutput = new ObjectOutputStream(cSocket.
    getOutputStream());
161                     objectOutput.writeObject(allUsers);
162                 }
163                 else if("Accept Request".equals(cmd) )
164                 {
165                     String name = inFromClient.readLine();
166                     addFriends( name );
167                     outToClient.writeBytes("New Friendship with " + name + '\n' );
168                     requestConfirmation( name );
169                 }
170                 else if( "Send Request".equals(cmd) )
171                 {
172                     String name = inFromClient.readLine();
173                     addFrinedRequest( name );
174                 }
175                 else if( "Unicast".equals(cmd) )
176                 {
177                     String user = inFromClient.readLine();
178                     String msg = inFromClient.readLine();
179                     giveMSG( user, msg );
180                 }
181                 else if( "Multicast".equals(cmd) )
182                 {
183                     String user = inFromClient.readLine();
184                     String msg = inFromClient.readLine();
185                     giveMSG( user, msg );
186                 }
187                 else if( "Broadcast".equals(cmd) )
188                 {
189                     String msg = inFromClient.readLine();
190                     broadcasting( msg );
191                 }
192                 else if( "Log out".equals(cmd) )
193                  {
194                      delUser();
195                      outToClient.writeBytes("Logged out of the system."  + '\n' );
196                      break;
197                  }
198             }
199         }
200         catch (IOException ex)
201         {
202                 Logger.getLogger(ClientHandler.class.getName()).log(Level.SEVERE, null,
    ex);
203         }
204     }
205 }
```

Listing 4: ClientHandler.java

```
1  package server;
2  import java.io.BufferedReader;
3  import java.io.BufferedWriter;
4  import java.io.DataOutputStream;
5  import java.io.FileNotFoundException;
6  import java.io.FileReader;
7  import java.io.FileWriter;
8  import java.io.IOException;
9  import java.io.InputStreamReader;
10 import java.net.Socket;
11 import java.util.Scanner;
12 import java.util.StringTokenizer;
```

```
13 import java.util.logging.Level;
14 import java.util.logging.Logger;
15 import static server.Server.cLists;
16
17 /* @author Shibli */
18 public class IntroHandler extends Thread
19 {
20     public Socket cSocket;
21     //Constructor of the class
22     public IntroHandler( Socket cSocket )
23     {
24         this.cSocket = cSocket;
25     }
26     //Method to save all the details
27     void saveDetails( String name, String uname, String pass ) throws IOException
28     {
29         BufferedWriter writer = new BufferedWriter(new FileWriter("detail.txt", true));
30         writer.newLine();
31         writer.append(uname + ":" + pass + ":" + name );
32         writer.close();
33     }
34     //Method to verify an user with username & password.
35     public boolean validity1( String uname, String pass )
36     {
37         FileReader inputFile = null;
38         try
39         {
40             inputFile = new FileReader("detail.txt");
41             Scanner parser = new Scanner(inputFile);
42             while (parser.hasNextLine())
43             {
44                 String line = parser.nextLine();
45                 if( "".equals(line) )continue;
46                 StringTokenizer tokens = new StringTokenizer( line, ":" );
47                 String u = tokens.nextToken();
48                 String p = tokens.nextToken();
49                 if( uname.equals(u) && pass.equals(p) )return true;
50             }
51             return false;
52         }
53         catch (FileNotFoundException ex)
54         {
55             Logger.getLogger(IntroHandler.class.getName()).log(Level.SEVERE, null, ex);
56         }
57         finally
58         {
59             try
60             {
61                 inputFile.close();
62             }
63             catch (IOException ex)
64             {
65                 Logger.getLogger(IntroHandler.class.getName()).log(Level.SEVERE, null,
    ex);
66             }
67         }
68         return false;
69     }
70     //Method to check if the registered username already exists
71     public boolean validity2( String uname )
72     {
73         FileReader inputFile = null;
74         try
75         {
76             inputFile = new FileReader("detail.txt");
77             Scanner parser = new Scanner(inputFile);
78             while (parser.hasNextLine())
79             {
80                 String line = parser.nextLine();
81                 if( "".equals(line) ) continue;
82                 StringTokenizer tokens = new StringTokenizer( line, ":" );
83                 String u = tokens.nextToken();
84                 if( (u.equals(uname)) ) return true;
85             }
86             return false;
```

```
87          }
88          catch (FileNotFoundException ex)
89          {
90              Logger.getLogger(IntroHandler.class.getName()).log(Level.SEVERE, null, ex);
91          }
92          finally
93          {
94              try {
95                  inputFile.close();
96              } catch (IOException ex) {
97                  Logger.getLogger(IntroHandler.class.getName()).log(Level.SEVERE, null,
    ex);
98              }
99          }
100         return false;
101     }
102     // Method to get the Full name of corresponding username.
103     String getName(String uname )
104     {
105         String name = "";
106         FileReader inputFile = null;
107         try
108         {
109             inputFile = new FileReader("detail.txt");
110             Scanner parser = new Scanner(inputFile);
111             while (parser.hasNextLine())
112             {
113                 String line = parser.nextLine();
114                 if( "".equals(line) )continue;
115                 StringTokenizer tokens = new StringTokenizer( line, ":" );
116                 String u = tokens.nextToken();
117                 String p = tokens.nextToken();
118                 name = tokens.nextToken();
119                 if( u.equals(uname) )return name;
120             }
121         }
122         catch (FileNotFoundException ex)
123         {
124             Logger.getLogger(IntroHandler.class.getName()).log(Level.SEVERE, null, ex);
125         }
126         finally
127         {
128             try {
129                 inputFile.close();
130             } catch (IOException ex) {
131                 Logger.getLogger(IntroHandler.class.getName()).log(Level.SEVERE, null,
    ex);
132             }
133         }
134         return name;
135     }
136     //Run Method for this thread.
137     public void run()
138     {
139         try {
140
141             BufferedReader inFromClient = new BufferedReader( new InputStreamReader(
    cSocket.getInputStream()));
142             DataOutputStream outToClient = new DataOutputStream(cSocket.getOutputStream
    () );
143             String cmd = inFromClient.readLine();
144
145             if( "login".equals(cmd) )
146             {
147                 String data = inFromClient.readLine();
148                 StringTokenizer tokens = new StringTokenizer( data, ":" );
149                 String uname = tokens.nextToken();
150                 String pass = tokens.nextToken();
151
152                 if( validity1(uname, pass ) )
153                 {
154                     outToClient.writeBytes("ok" +  '\n' );
155                     String name = getName(uname );
156                     ClientHandler CH = new ClientHandler(cSocket, name, uname);
157                     CH.start();
```

```
158                     cLists.add(CH );
159                 }
160                 else outToClient.writeBytes("not ok" +  '\n' );
161             }
162             else if( "signup".equals(cmd) )
163             {
164 
165                 String data = inFromClient.readLine();
166                 StringTokenizer tokens = new StringTokenizer( data, ":" );
167                 String uname = tokens.nextToken();
168                 String pass = tokens.nextToken();
169                 String name = tokens.nextToken();
170                 if( validity2( uname ) )outToClient.writeBytes("not ok" +  '\n' );
171                 else
172                 {
173                     saveDetails(name, uname, pass );
174                     outToClient.writeBytes("ok" +  '\n' );
175                     ClientHandler CH = new ClientHandler(cSocket, name, uname);
176                     cLists.add(CH );
177                     CH.start();
178                 }
179             }
180         }
181         catch (IOException ex) {
182             Logger.getLogger(IntroHandler.class.getName()).log(Level.SEVERE, null, ex);
183         }
184     }
185 }
```

Listing 5: IntroHandler.java

```
1  package server;
2  import java.io.BufferedWriter;
3  import java.io.DataOutputStream;
4  import java.io.FileNotFoundException;
5  import java.io.FileReader;
6  import java.io.FileWriter;
7  import java.io.IOException;
8  import java.util.ArrayList;
9  import java.util.Scanner;
10 import java.util.StringTokenizer;
11 import java.util.logging.Level;
12 import java.util.logging.Logger;
13 import static server.Server.cLists;
14 
15 /* @author Shibli */
16 
17 public class ContinuousHandler extends Thread
18 {
19     public void resetFR( ArrayList<String> resetFR_LIST )throws IOException
20     {
21         BufferedWriter writer = new BufferedWriter(new FileWriter("requests.txt", false)
    );
22         for( String i: resetFR_LIST ){
23             System.out.println(i);
24             writer.write( i );
25             writer.newLine();
26         }
27         writer.flush();
28         writer.close();
29     }
30 
31     public void FR() throws IOException
32     {
33         ArrayList<String> FR_LIST = new ArrayList<String>();
34         boolean check = false;
35         try
36         {
37             FileReader inputFile = null;
38             inputFile = new FileReader("requests.txt");
39             Scanner parser = new Scanner(inputFile);
40             while (parser.hasNextLine())
41             {
42                 String line = parser.nextLine();
43                 if( "".equals(line) )continue;
44                 StringTokenizer tokens = new StringTokenizer( line, ":" );
```

```
45                  String type = tokens.nextToken();
46                  String from = tokens.nextToken();
47                  String to = tokens.nextToken();
48                  if( "send".equals(type) )
49                  {
50                      for( ClientHandler i: cLists )
51                      {
52                          if( i.uname.equals(to) )
53                          {
54                              type = "receive";
55                              try
56                              {
57                                  DataOutputStream outToClient = new DataOutputStream(i.
    cSocket.getOutputStream() );
58                                  outToClient.writeBytes("One friend request from : " +
    from + '\n' );
59                              }
60                              catch (IOException ex)
61                              {
62                                  Logger.getLogger(ContinuousHandler.class.getName()).log(
    Level.SEVERE, null, ex);
63                              }
64                              check = true;
65                          }
66                      }
67                  }
68                  FR_LIST.add(type + ":" + from + ":" + to );
69              }
70          }
71          catch (FileNotFoundException ex)
72          {
73              Logger.getLogger(ContinuousHandler.class.getName()).log(Level.SEVERE, null,
    ex);
74          }
75          if( check == true )resetFR(FR_LIST );
76      }
77
78      public void resetN( ArrayList<String> resetN_LIST )throws IOException
79      {
80          BufferedWriter writer = new BufferedWriter(new FileWriter("notifications.txt",
    false));
81          for( String i: resetN_LIST ){
82              writer.write( i );
83              writer.newLine();
84          }
85          writer.flush();
86          writer.close();
87      }
88
89      public void NOT() throws IOException
90      {
91          ArrayList<String> NOT_LIST = new ArrayList<String>();
92          boolean check = false;
93          try
94          {
95              FileReader inputFile = null;
96              inputFile = new FileReader("notifications.txt");
97              Scanner parser = new Scanner(inputFile);
98              while (parser.hasNextLine())
99              {
100                 String line = parser.nextLine();
101                 if( "".equals(line) )continue;
102                 StringTokenizer tokens = new StringTokenizer( line, ":" );
103                 String type = tokens.nextToken();
104                 String to = tokens.nextToken();
105                 String notification = tokens.nextToken();
106                 if( "send".equals(type) )
107                 {
108                     for( ClientHandler i: cLists )
109                     {
110                         if( i.uname.equals(to) )
111                         {
112                             type = "receive";
113                             try
114                             {
```

```
115                                 DataOutputStream outToClient = new DataOutputStream(i.
    cSocket.getOutputStream() );
116                                 outToClient.writeBytes("Notifications: " + notification
    + '\n' );
117                             }
118                             catch (IOException ex)
119                             {
120                                 Logger.getLogger(ContinuousHandler.class.getName()).log(
    Level.SEVERE, null, ex);
121                             }
122                             check = true;
123                         }
124                     }
125                 }
126             NOT_LIST.add(type + ":" + to + ":" + notification );
127         }
128         }
129         catch (FileNotFoundException ex)
130         {
131             Logger.getLogger(ContinuousHandler.class.getName()).log(Level.SEVERE, null,
    ex);
132         }
133         if( check == true )resetN(NOT_LIST );
134     }
135
136     public void resetMSG(ArrayList<String> resetMSG_LIST)throws IOException
137     {
138         BufferedWriter writer = new BufferedWriter(new FileWriter("message.txt", false))
    ;
139         for( String i: resetMSG_LIST ){
140             writer.write( i );
141             writer.newLine();
142         }
143         writer.flush();
144         writer.close();
145     }
146
147     public void MESEG() throws IOException
148     {
149         ArrayList<String> MESEG_LIST = new ArrayList<String>();
150         boolean check = false;
151         try
152         {
153             FileReader inputFile = null;
154             inputFile = new FileReader("message.txt");
155             Scanner parser = new Scanner(inputFile);
156             while (parser.hasNextLine())
157             {
158                 String line = parser.nextLine();
159                 if( "".equals(line) )continue;
160                 StringTokenizer tokens = new StringTokenizer( line, ":" );
161                 String type = tokens.nextToken();
162                 String from = tokens.nextToken();
163                 String to = tokens.nextToken();
164                 String msg = tokens.nextToken();
165                 if( "send".equals(type) )
166                 {
167                     for( ClientHandler i: cLists )
168                     {
169                         if( i.uname.equals(to) )
170                         {
171                             type = "receive";
172                             try
173                             {
174                                 DataOutputStream outToClient = new DataOutputStream(i.
    cSocket.getOutputStream() );
175                                 outToClient.writeBytes("Message from " + from + " : " +
    msg +  '\n' );
176                             }
177                             catch (IOException ex)
178                             {
179                                 Logger.getLogger(ContinuousHandler.class.getName()).log(
    Level.SEVERE, null, ex);
180                             }
181                             check = true;
```

```
182                             }
183                         }
184                     }
185                     MESEG_LIST.add(type + ":" + from + ":" + to + ":" + msg );
186                 }
187             }
188             catch (FileNotFoundException ex)
189             {
190                 Logger.getLogger(ContinuousHandler.class.getName()).log(Level.SEVERE, null,
        ex);
191             }
192             if( check == true ){
193                 resetMSG(MESEG_LIST );
194             }
195         }
196 
197         public void run()
198         {
199             while( true )
200             {
201                 try
202                 {
203                     FR();
204                     MESEG();
205                     NOT();
206                 }
207                 catch (IOException ex)
208                 {
209                     Logger.getLogger(ContinuousHandler.class.getName()).log(Level.SEVERE,
        null, ex);
210                 }
211             }
212         }
213 }
```

Listing 6: ContinuousHandler.java