Università degli Studi di Salerno

Dipartimento di Ingegneria dell'Informazione ed Elettrica e Matematica Applicata

Distribuited Programming
Chat Broadcast and OneToOne (GUI)

Anno Accademico 2021/2022

Gruppo 9

Renzulli Giuseppe – 0622701514
g.renzulli4@studenti.unisa.it

Vincenzo Salvati – 0622701550
v.salvati10@studenti.unisa.it

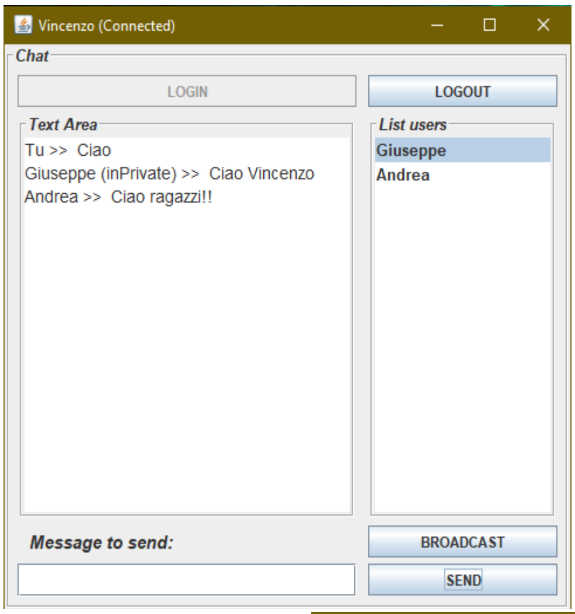
Prof De Maio Carmen

Summary

Chat Broadcast and OneToOne

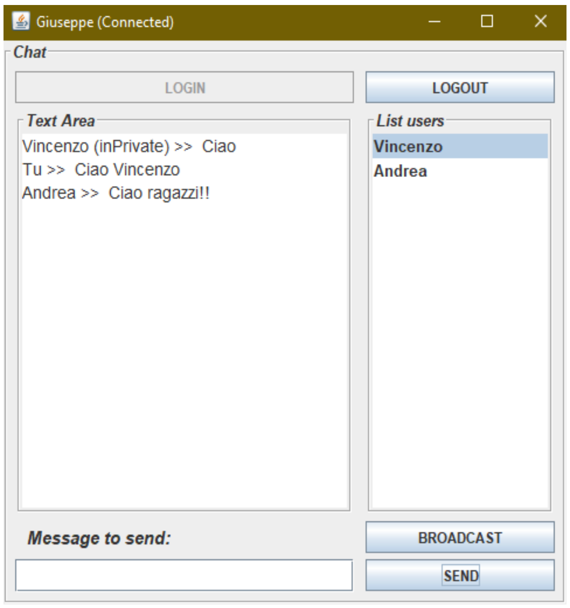
Introduction

This project requests to implement a chat among several clients in an architecture client-server. As far as the chat concerned, it provides two modalities: sending message in broadcast or between two specific clients. In particular, that involves in an indirectly communication among clients because the messages are routed by a single server that, due to the threads, is able to face up with different requests required from several clients. Hence, the next paragraphs describe the packages "client" and "server" that contains the implemented scripts in order to deal with properly these services.



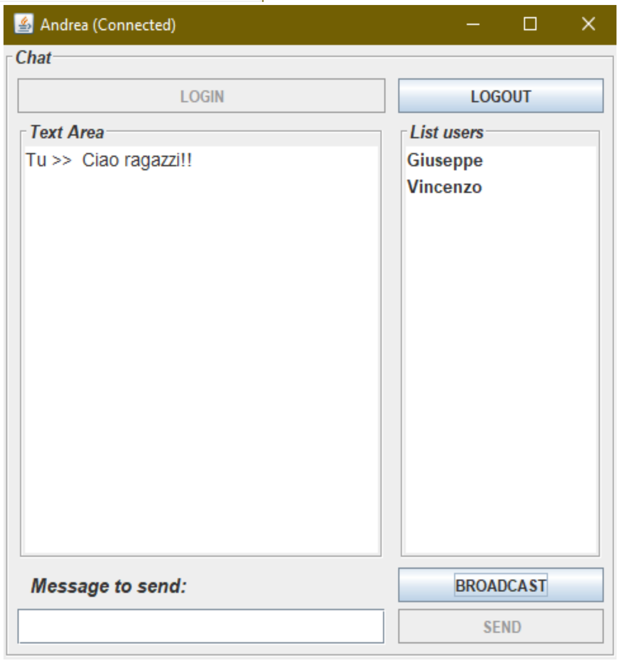
Tu >> Ciao
Giuseppe (inPrivate) >> Ciao Vincenzo
Andrea >> Ciao ragazzi!!

Giuseppe
Andrea



Vincenzo (inPrivate) >> Ciao
Tu >> Ciao Vincenzo
Andrea >> Ciao ragazzi!!

Vincenzo
Andrea



Tu >> Ciao ragazzi!!

Giuseppe
Vincenzo

GUI

Client

The package client contains two classes:

- ***Client.java***: firstly, implements the GUI used for each client in order to perform the login and send the messages in the two modalities described before.
  So, it allows to create new socket and a new string which contains the name of the client and these data are send to the server by an OutputStream.
  Then it waits for an "*OK*" from the server to confirm its socket and username so that it can invocate a command "*LST*" that is useful to update the list of clients connected.
  Finally, it runs the thread implemented by an object defined in the *ReceiveMessageFromServer.java* class;
- ***ReceiveMessageFromServer.java***: as it is reported before, this class generate a thread that it continuously listens for the messages received from the server.In particular there are three kinds of messages for which it waits for:
  - a string which begins with "*USR*": when it receives this string, the list of users connected is updated because a client was connected to the server;
  - a string which begins with "*LUS*": when it receives this string, the list of users connected is updated because a client was disconnected to the server;
  - others strings: it prints the message on the message area text.

Server

The package server contains three classes:

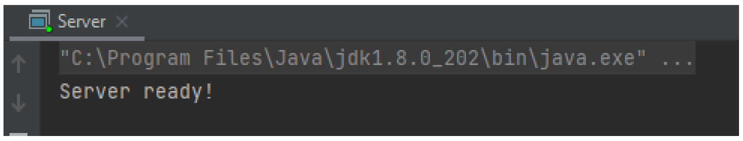
- ***Server.java***: firstly, create several server threads for each client that try to establish a connection with it. Hence, it allows the users login checked its sockets and unrepeated username.
  After notified the clients about the acceptation and if the connection has been successful, this script runs the thread implemented by an object defined in the *SendMessageToClients.java* class;
- ***SendMessageToClients.java***: as it is reported before, this class generate a thread that it continuously listens for the messages received from the clients. In particular there are four kinds of message for which it waits for:
  - a string which begins with "*LOT*": when it receives this string, it notifies the disconnected client to others clients;
  - a string which begins with "*BRD*": when it receives this string, it sends the message received from a client to others clients;
  - a string which begins with "*OTO*": when it receives this string, it sends the message received from a client to a specific client;
  - a string which begins with "*LST*": when it receives this string, it sends the new list of clients in order to allow them to update it;
- ***User.java***: it allows to create an object with specific attributes useful for the server's operations. In particular it contains the attributes:
  - *username*: is the string of the client's name;
  - *outputStream*: is used to communicate with a specific client by its stream;
  - *socket*: is the client's socket.

How to use

To use this application is essential firstly to run the server.java file by an IDE in order to allow the client to establish a connection with it.

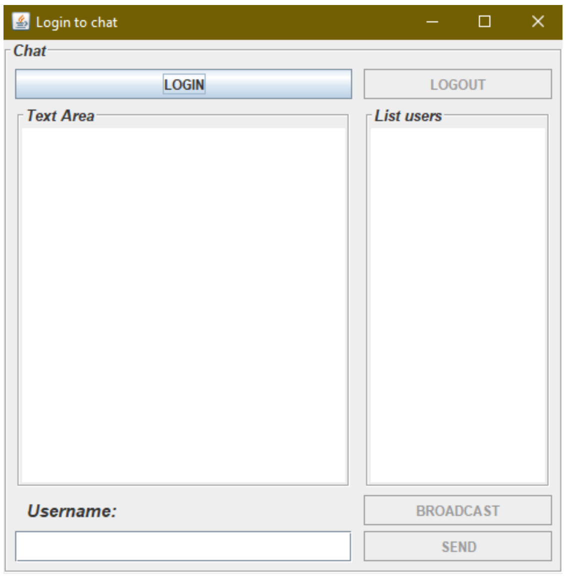
The following steps will help you to use the program:

1. **Start the server:** when the server is ready a message “Server ready!” is printed on the IDE’s console.

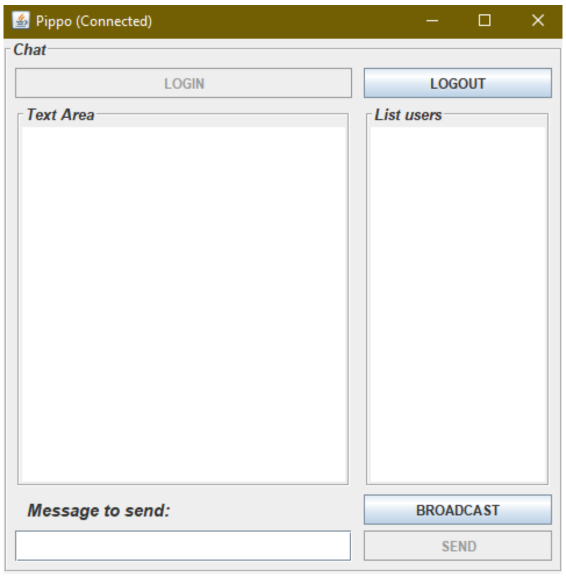


```
"C:\Program Files\Java\jdk1.8.0_202\bin\java.exe" ...
Server ready!
```

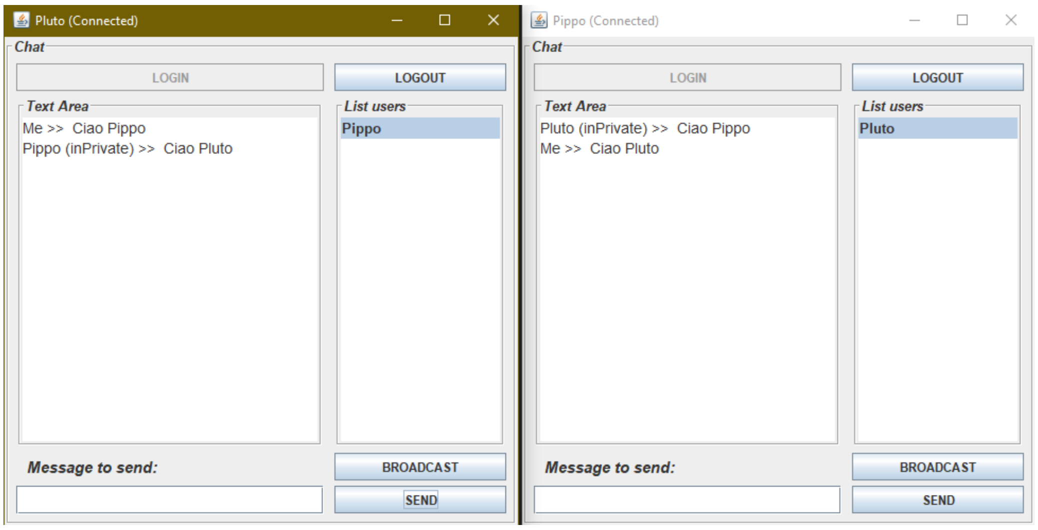
2. **Run the client:** it is possible run the client.java file by an IDE so that it will appear GUI which drives you during the login.



3. **Chose the username:** now it is possible to insert your username in the text field under “Username:” and if the chosen name is not used you will finally connect.



4. **Start the chat:** you can notice that the bottoms Logout and Broadcast are enable and, finally, if another client will be run, on the "List user" will appear its name which is selectable in order to chat in private with it.



Me >> Ciao Pippo
Pippo (inPrivate) >> Ciao Pluto

Pluto (inPrivate) >> Ciao Pippo
Me >> Ciao Pluto

P.S.

- If one client performs the logout, its name disappears from the list of all the user connected;
- It is impossible connect more client with the same name;
- The broadcast messages will be read from all the user connected, instead of "SEND" button which allows to chat in private way with a specific selected client.