Università degli Studi di Salerno

Dipartimento di Ingegneria dell'Informazione ed Elettrica e Matematica Applicata

Distributed Programming
BattleShip Game (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

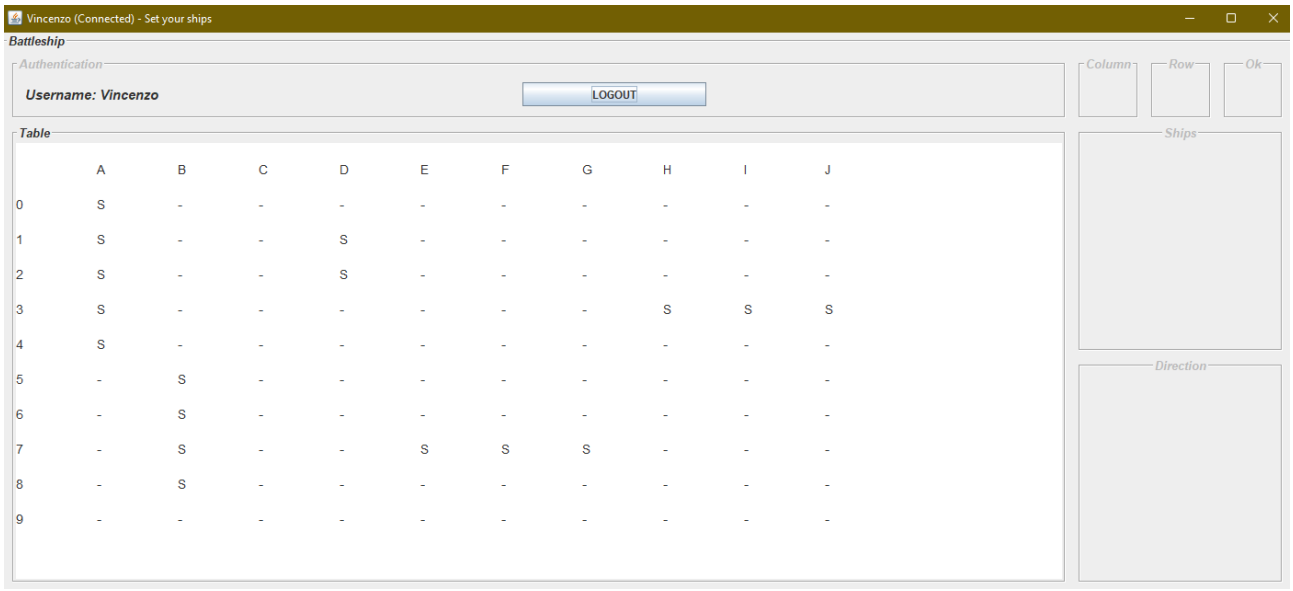
Summary

BattleShip Game

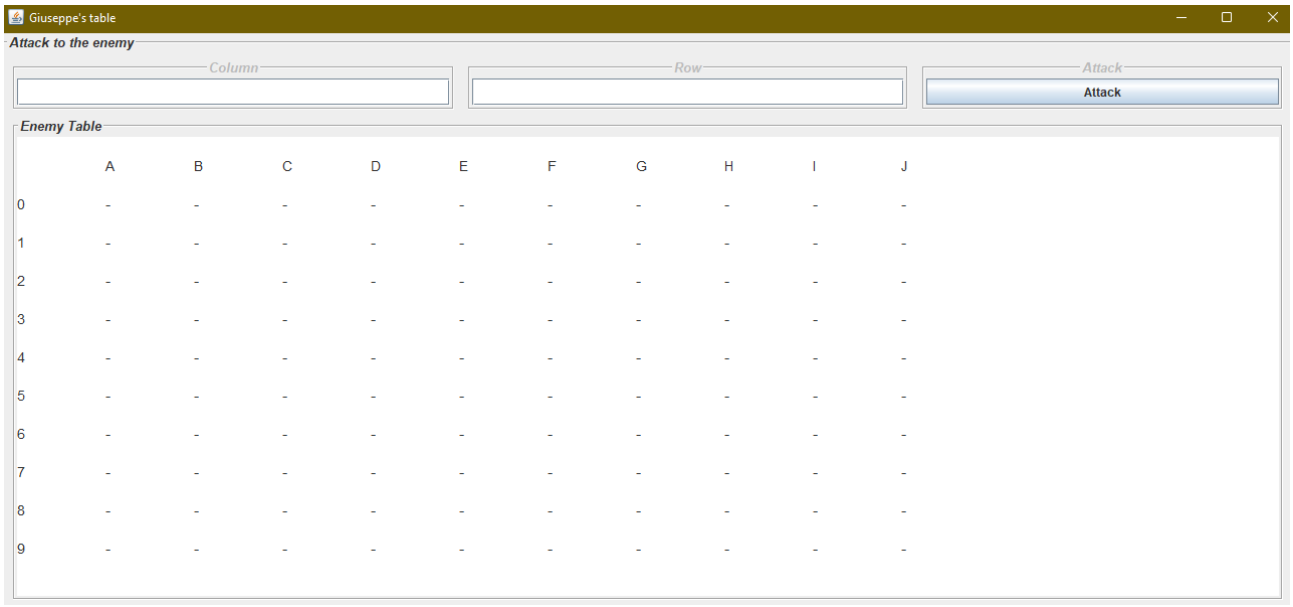
Introduction

This project requests to implement the battleship game between two clients in an architecture client-server. As far as the game concerned, it provides the connection from several clients, which are divided into couples of them. That involves in an indirectly communication among clients because the messages are routed by a single server, that manage their addresses and ports through the UDP protocol, so we do not have a single socket connection between server and client. The server uses threads 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 to deal with properly these services.



Player's GUI



Enemy's GUI

Client

The package client contains different classes:

- **Client.java**: firstly, implements the GUI used for each client to perform the login and to set the ships' positions. So, it allows to create a new datagram socket and a new datagram packet so that it can receive and send messages to the server.
  Then it waits for an "*OK*" from the server to confirm that its username is available, so that it can wait for the challenger. After that the server allows them to position their ships and when they done, the clients notify the server. Only when both have notified the server, the fight begins.
  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.
  There are many kinds of messages for which it waits for:
  - a string which begins with "*LOT*": when it receives this string, disable all buttons, and notify the server with client's logout;
  - a string which begins with "*WAI*": when it receives this string, puts the client in listening for a challenger;
  - string which begins with "*FND*": when it receives this string, it means that a challenger is found and enable the useful button to position the ships;
  - a string which begins with "*FGT*": when it receives this string, the clients have finish to position the ships and the fight begin. The clients also create an object *Enemy*, so that another frame will be create that is useful to attack the corresponding challenger;
  - a string which begins with "*ATK*": when it receives this string, the client gets the two coordinates indirectly from the enemy, updates its table by them to mark the cells with a *MISS* or a *HIT*, prints the new table on the own *Table_AreaText* of the GUI and finally send it to the enemy (in this case the ships are hidden);
  - a string which begins with "*UPD*": when it receives this string, the client updates the enemy's table;
  - a string which begins with "*WIN*": when it receives this string, the clients get the winner's username and the battle ends;
- **Enemy.java**: this class generate another frame for each client to allow the view and the attack of the enemy table;
- **Grid.java**: this class generate an object *Grid*, that represent a cell of the table. Each cell could be marked as *VOID*, *HIT* or *MISS*, to represent the status of the specific cell. It also provides information about if a ship is positioned or not and about ship's attributes, in particular length and direction;
- **Table.java**: due to the previous class's description, the class Table can create Grid matrix 10*10 and finally it takes in account the points of a player. In this class there are two important methods:
  - addShip (Ship ship): due to the coordinates and the direction chosen by the player, sets the respectively ship on the table;
  - printTableToString (int type):
    - *type = 0*, prints the enemy's table hiding the ships' positions;
    - *type = 1*, prints the positioned ships;
    - *type = 2*, prints the user's table with both ships and hit ships;
- **Ships.java**: create an object Ships with four possible directions (*UP*, *RIGHT*, *DOWN*, *LEFT*), the ships' length and its row and col coordinates;
- **Player.java**: defines two objects: Table and Ships, for a specific client.

Server

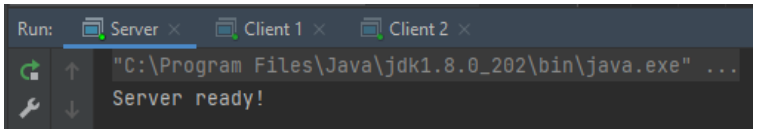
The package server contains three classes:

- **Server.java**: firstly, create several server threads to allow client's requests. There are many kinds of messages for which it waits for:
  - a string which begins with "*LOT*": when it receives this string, disconnects both clients when one of them left the game;
  - a string which begins with "*LIN*": when it receives this string, it allows the clients login checked its unrepeated username, and if it is successful, all kinds of requests can be served. Furthermore, synchronizes a player with the respective enemy;
  - a string which begins with "*FNS*": when it receives this string, communicates to the clients that the match has started because both have positioned the ships;
  - a string which begins with "*CRD*": when it receives this string, sends the attack's coordinates to the right challenger;
  - a string which begins with "*TAB*": when it receives this string, sends the table (*type = 0* that is the table with hidden ships) to the attacker;
  - a string which begins with "*LOS*": when it receives this string, communicates the username's winner to both of clients;
- **Players.java**: class containing the synchronized clients for the battle, memorizing their username, address, port and a boolean to indicate who has positioned its ships;
- **User.java**: class providing attributes for a single client as username, address and port.

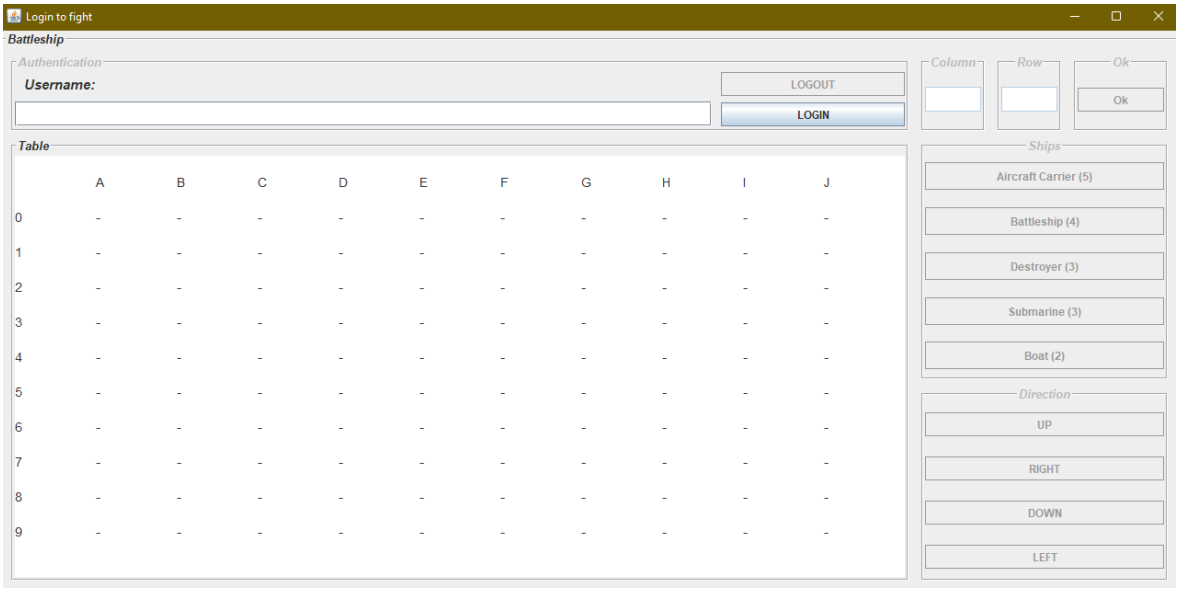
How to use

To use this application is essential firstly to run the server.java file by an IDE. The following steps will help you to use the program:

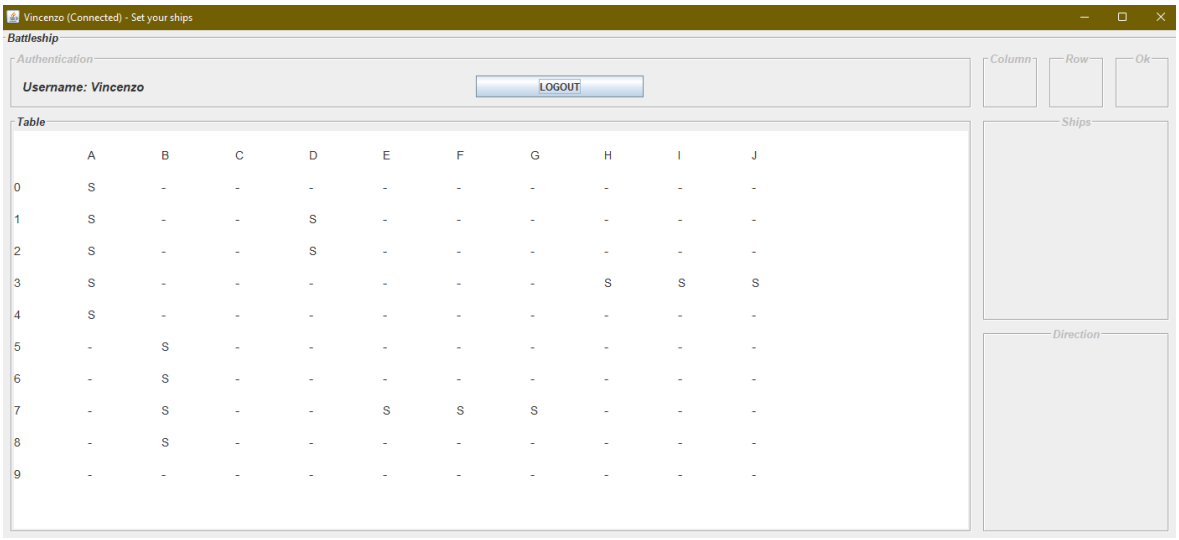
1. **Start the server**: when the server is ready a message “Server ready!” is printed on the IDE’s console.



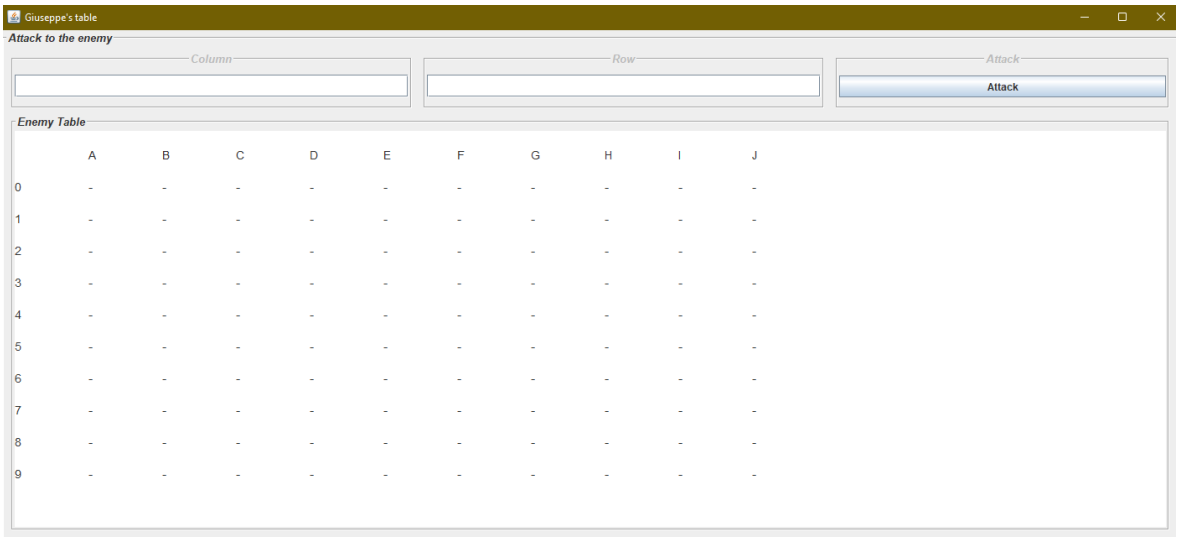
2. **Run the client**: it is possible to run the client.java file by an IDE so that it will appear GUI which drives you during the login.
   In this frame it is possible to insert the username in the text field under the label “Username:” and if the chosen name is not used, the client will finally connect.
   It has to be repeated for almost 2 clients.



3. **Position the ships**: select the ship, then set the coordinates and finally set the direction.



4. **Start the battle**: when both clients have finished to position their ships, the enemy frame will appear and it is possible insert the coordinates to attack.



P.S.

- To win it is necessary destroy all the enemy's ships (the looser count 17 points for the enemy);
- It is possible to attack once each for each pair of clients;
- When one client performs the logout, if it is synchronized with a challenger, the challenger is automatically disconnected.