Distributed Systems Assignment 2

Distributed Scrabble Game

Group Name: DS_Proj_Two

Group Member:

Xueting Tan 948775

Guannan Li 947196

Ruipeng Wang 904991

Pengfei Xiao 871910

Tutor in charge: Xunyun Liu

Date: 14th October 2018

1 Introduction

The project has implemented a distributed Scrabble Game that allows two or more players to place a character one by one on a grid to generate words in turn. Every time after placing a letter, all players could see changes in their own interfaces at the same time. Then a voting program will be launched to determine if the candidate string is a word. If all the players vote yes, the player who has placed the letter will gain corresponding points. The game will end when all players choose to pass or one user has logged out. Then a result GUI will show up. All the players would go back to the invitation hall and are allowed to start a new game.

The project is implemented by using sockets and threads in Java programming language. All the communications between the clients and the server are based on TCP. String type is used as the basic communication data format. Additionally, a concise and fully functional GUI has been designed to achieve all expected features. Meanwhile, many exceptions are handled to guarantee the concurrency and consistency. In all, this game system allows multiple clients to connect to one multi-threaded server and perform operations concurrently. A thread-per-connection architecture is applied here.

2 Architecture

2.1 UML

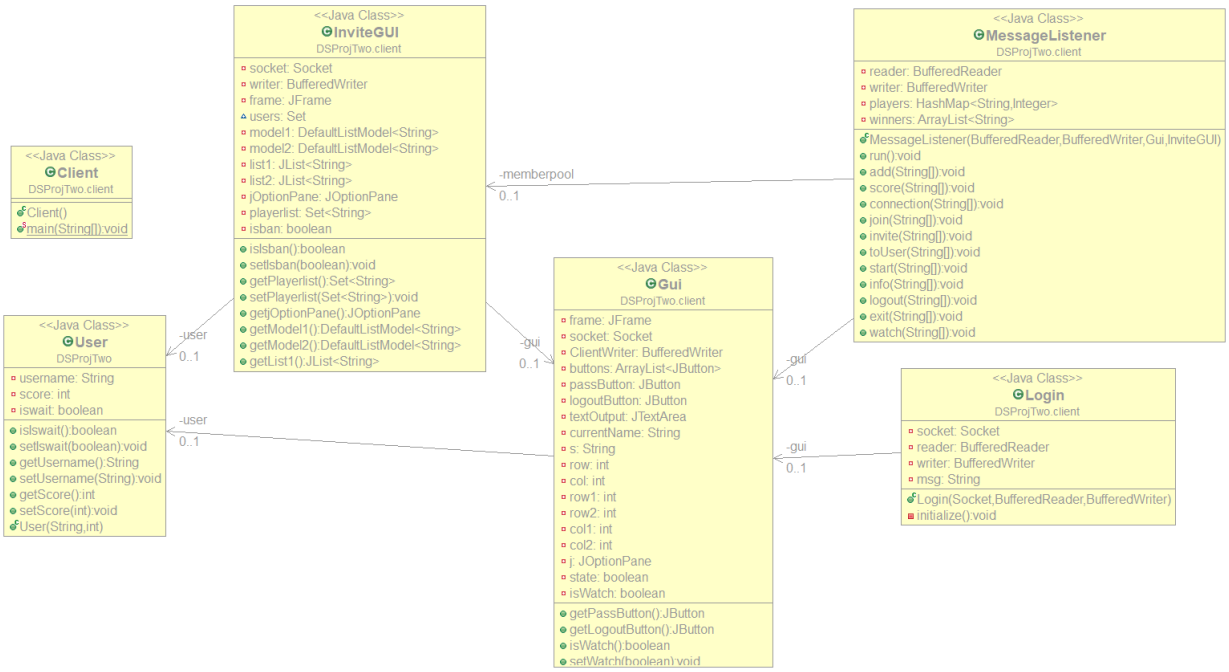

Figure 1a. Client UML diagram

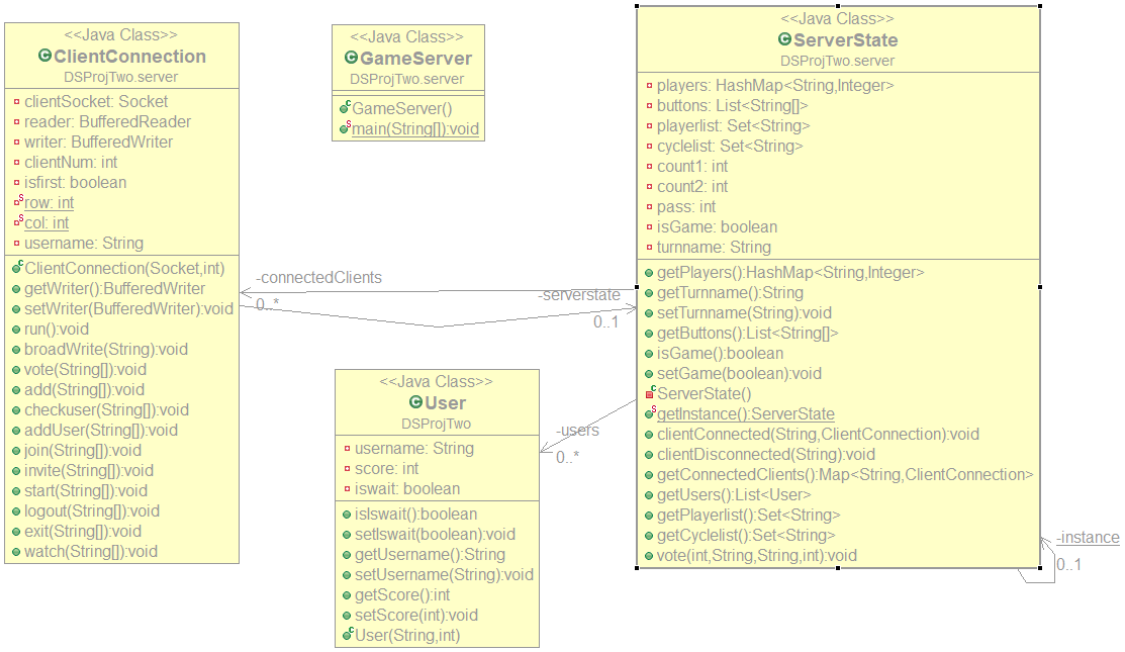

Figure 1b. Server UML diagram

2.2 Sequence diagram

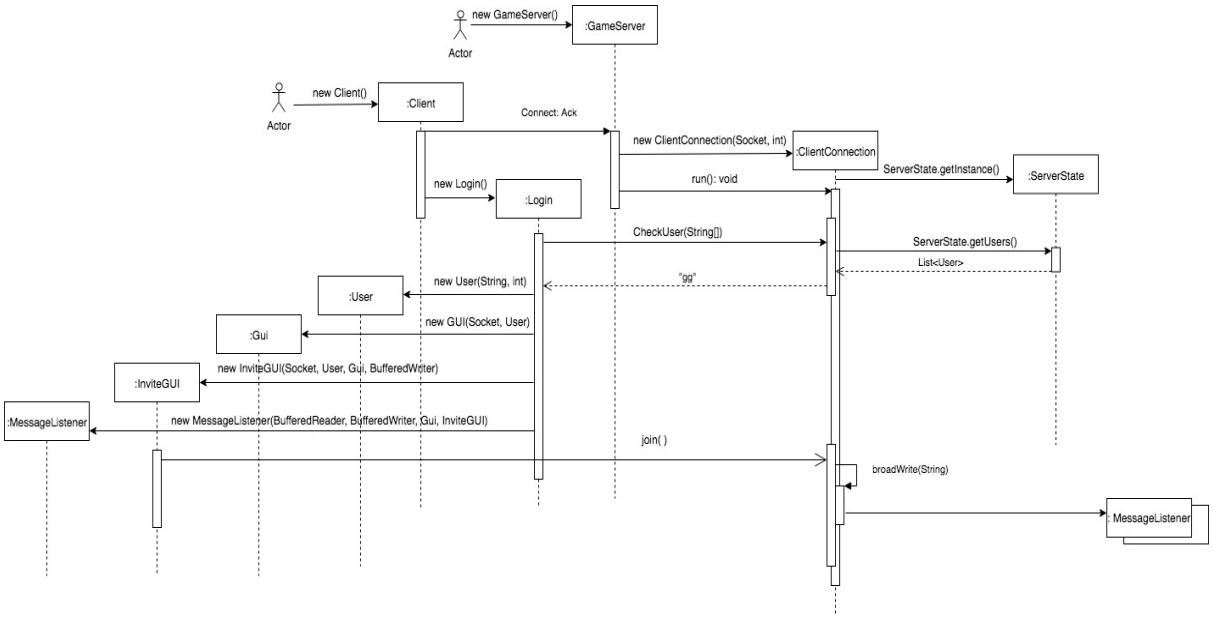

Figure 2. Sequence diagram

3 Description

3.1 protocols and messaging description

In this project, TCP was chosen over UDP since the communication reliability has already been offered by TCP. UDP datagrams can be lost or delivered in a different order to the receiving application, while TCP guarantees the sending order of the message which is quite important in this specific context because the validity of some operations depend on whether or not some other operations are performed before them. For example, if the packet

containing massage “start game” from client to server was delivered before the other packet which carries “join game” from a user, it would lead to a failure of unsuccessful joining for that user, which is an absolute conflict with the real life scenario: the user sends the “join game” request before anyone else starts the game. UDP has some advantages as well. If the quantity of clients gets too high and the massage exchanging gets very frequent, UDP might have a better performance since the packet transmitting speed is relatively higher than TCP. And we could use another way to provide security to UDP protocol. But in this case here, there is no need to do it.

The best data format here is String type which is easy to use. Since all the messages passing between clients and the server are just simple commands like “add” or “invite”, there is no need to use other complicated data formats here. String type is also the best choice to use because all the group members are familiar with this data type.

3.2 Implementation details and contributions

3.2.1 Login

After the client is launched, the login interface will show up. A player can put in any username they like. If the username already exists, a warning window will show up to remind the user to change to another name, or they can click the “Random Name” button to randomly generate a username.

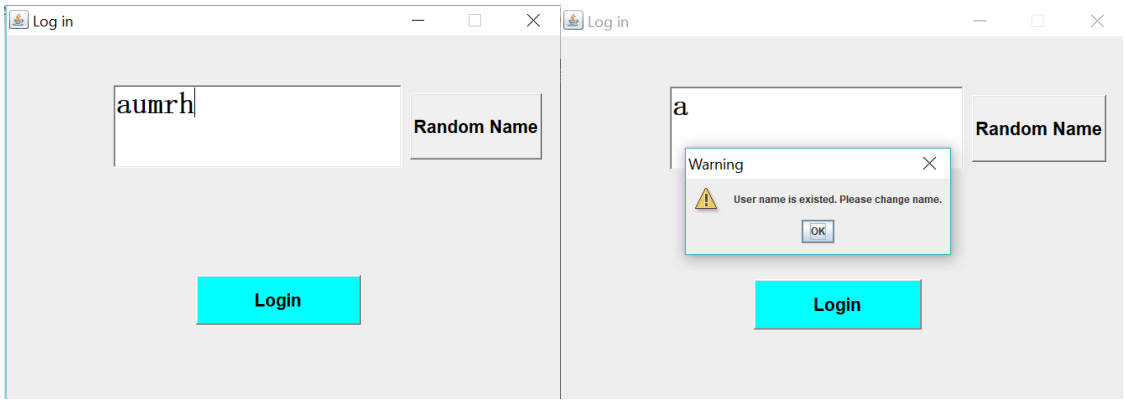

Figure 3. Login interface and duplicate username check

3.2.2 Invitation Hall

When the player has logged in, they will see the invitation hall. If they want to play the game, they can click the “JoinList” button to enter the player list. Once a player is in the player list, they can invite other users by clicking the button “Invite”. Other players will be added in the player list once they accept the invitation. Meanwhile, other players can also enter the player list by clicking the “JoinList” button. When more than two players are in the player list, the game could be started by pressing the “StartGame” button.

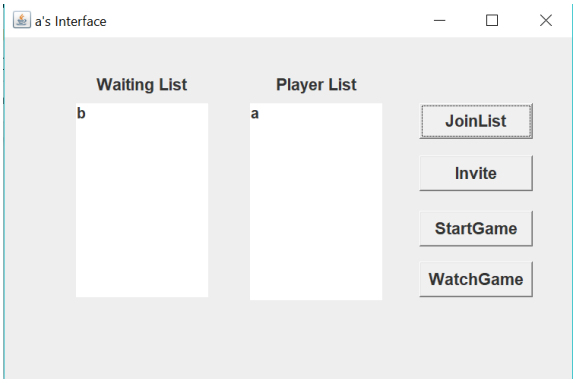

Figure 4. Invitation hall interface

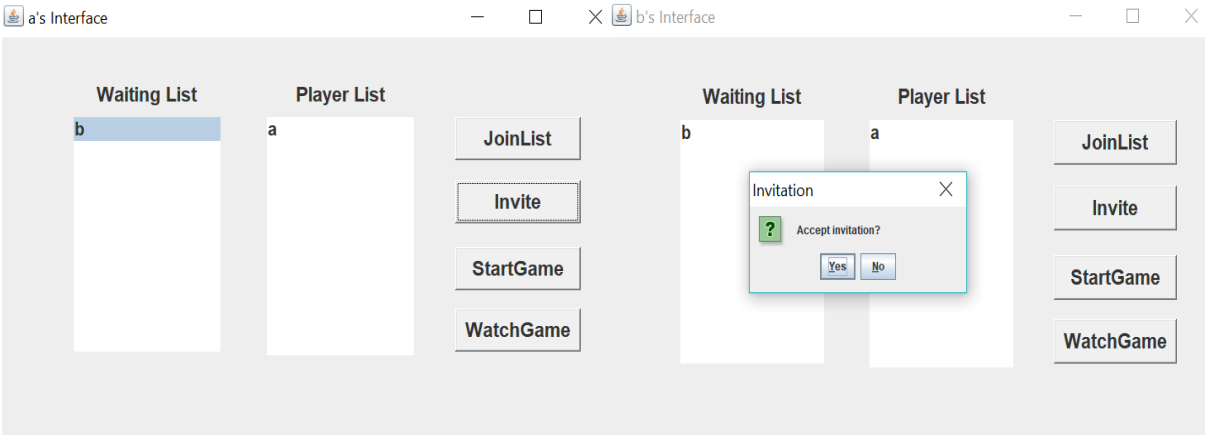

Figure 5. Invitation function

3.2.3 Scrabble Game Process

When the game starts, a 20*20 grid and a score table will be displayed. Every user can place a letter in their turn and then choose “horizontal” or “vertical” as their voting direction. They can also choose “null” option to skip the voting process. Next, all the other users will see a voting interface to evaluate the candidate word. The score table will be updated once all the players vote “Yes”. If a player does not want to place a letter in their round, they could press the “Pass” button to skip.

The game will be over when all players choose to pass or a user has logged out. Then a resulting GUI will show up. All the players will go back to the invitation hall and will be allowed to start a new game.

In addition, once a game has started, no other players are allowed to start a new game. They can only press the “WatchGame” button to watch the game in progress.

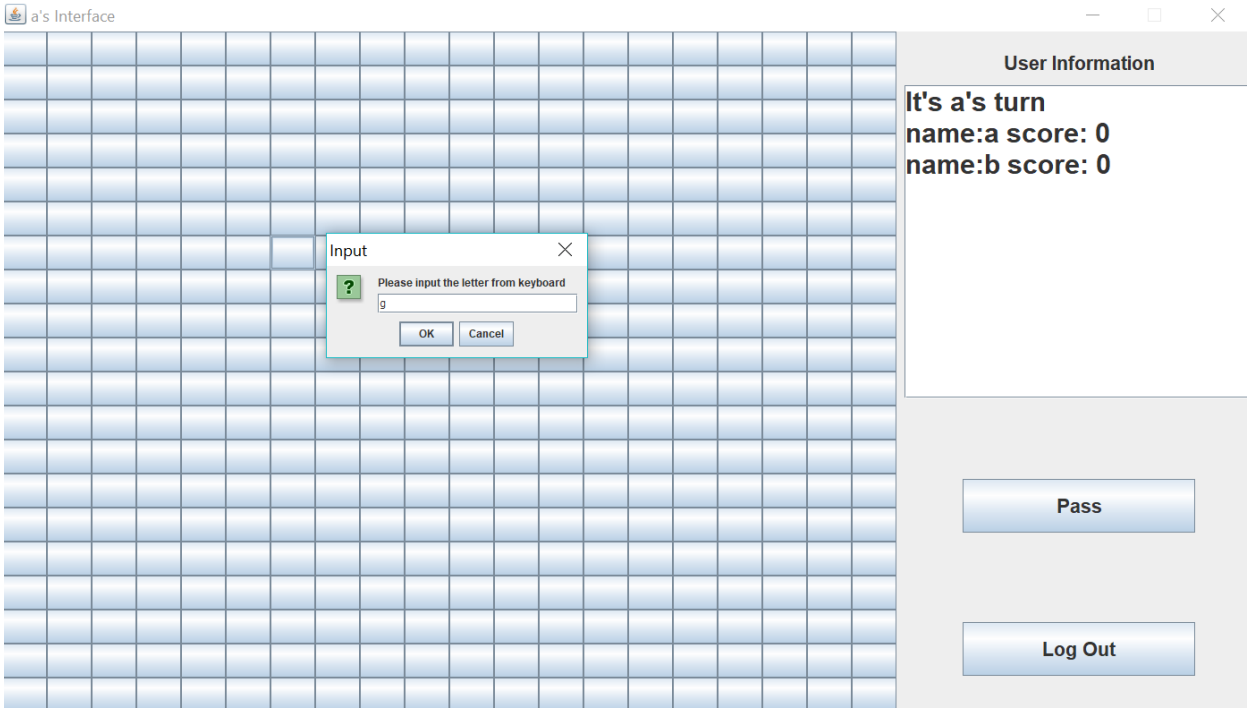

Figure 6. Game interface

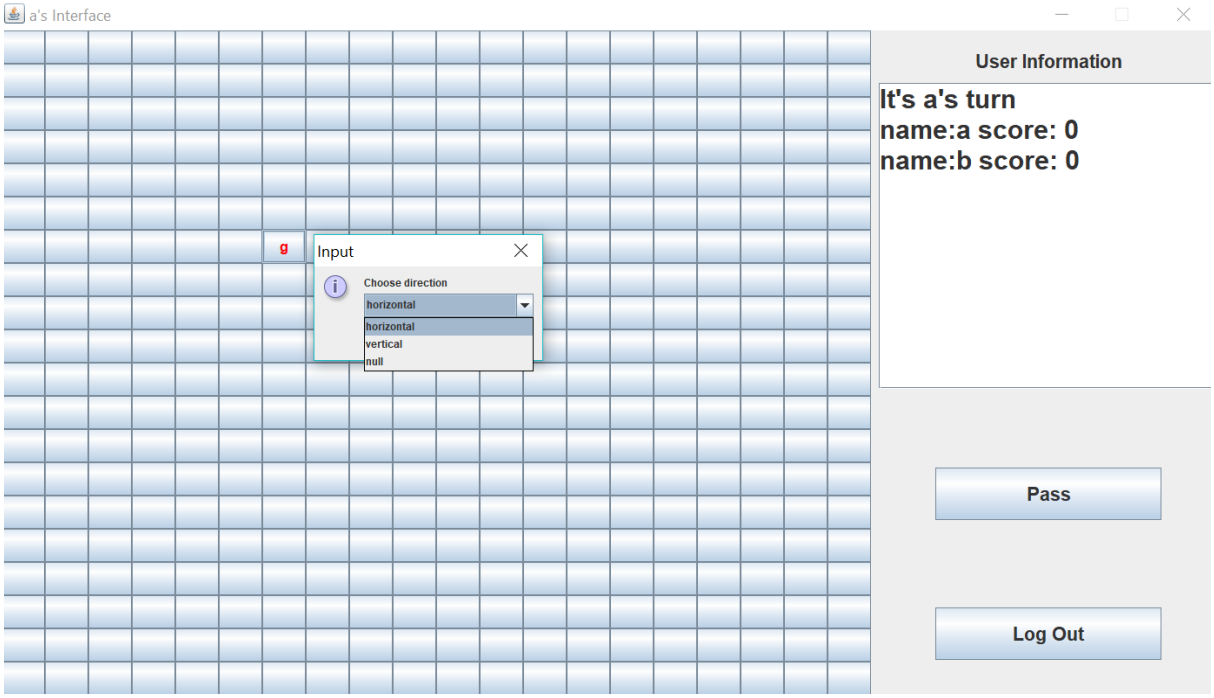

Figure 7. Voting direction choosing

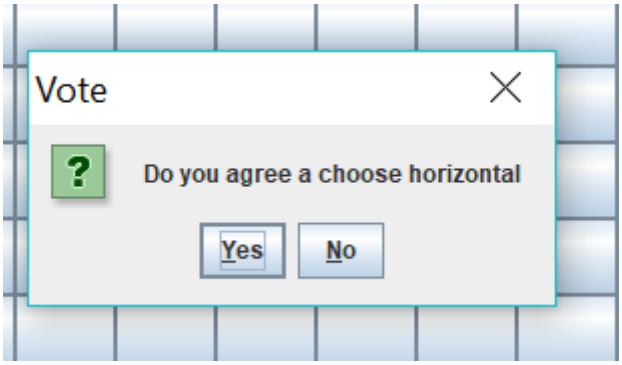

Figure 8. Voting interface

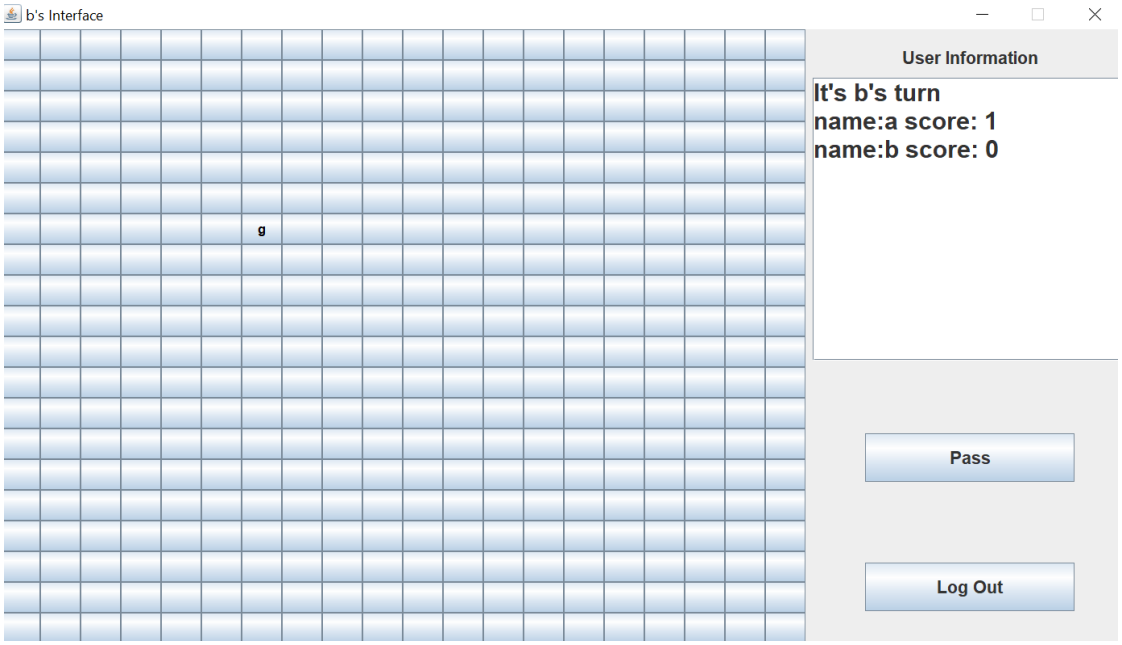

Figure 9. Update score table

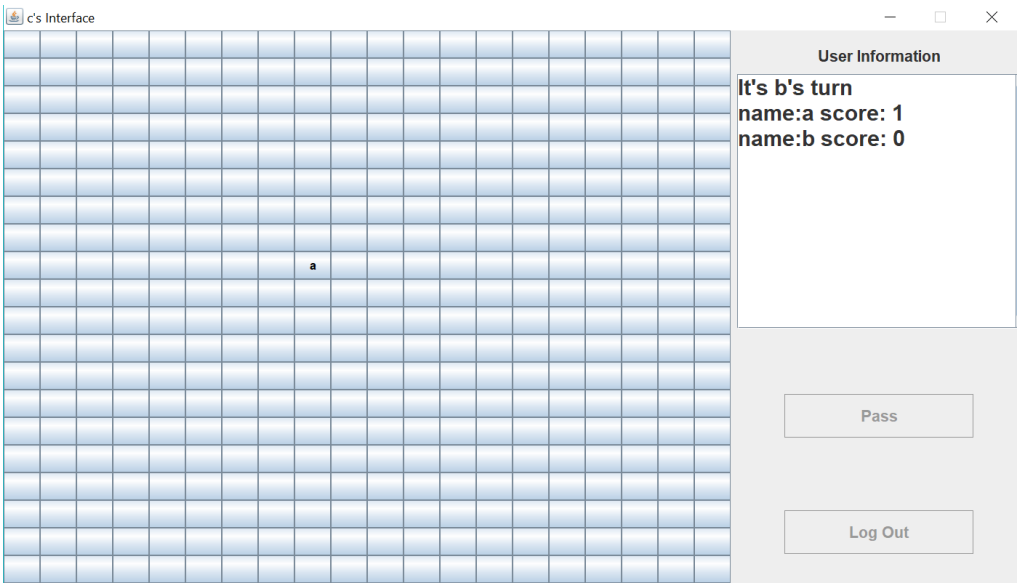

Figure 10. Watch game interface

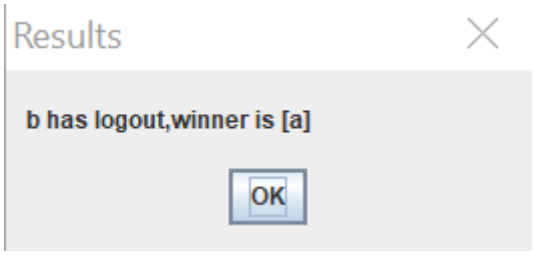

Figure 11. Result interface

3.2.4 Exception Handling

Every player is allowed to place only one valid English letter in each round. The regular expression is applied in this project to check input validity. Thus, all invalid characters such as numbers and punctuation, and more than one or zero-letter input would lead to a warning GUI.

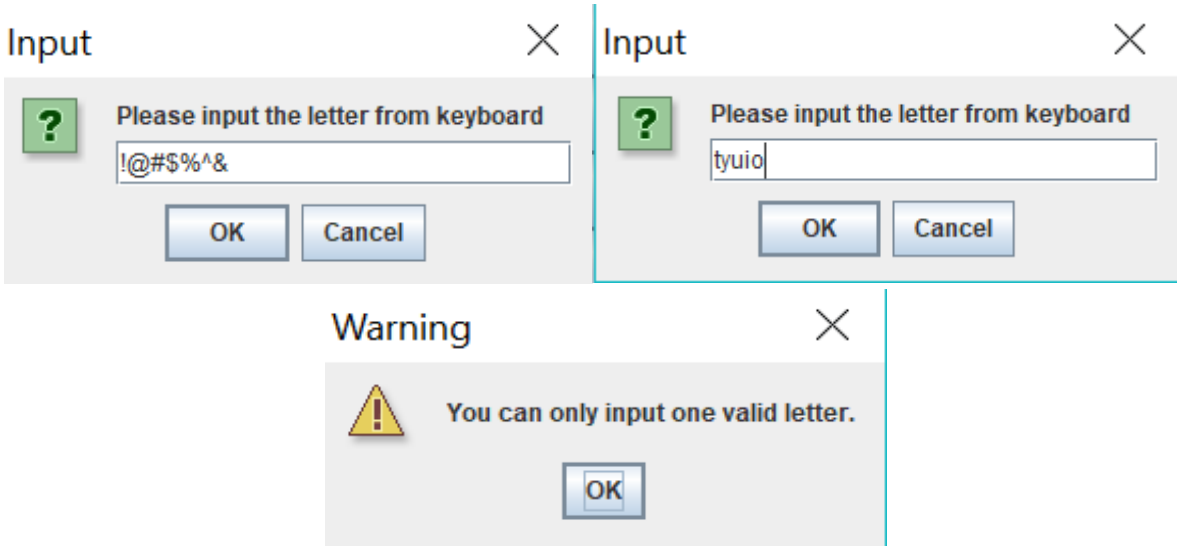

Figure 12. Input validity check

If the client is disconnected accidentally or the server is shut down occasionally, then the game system will display an error message to remind users to restart.

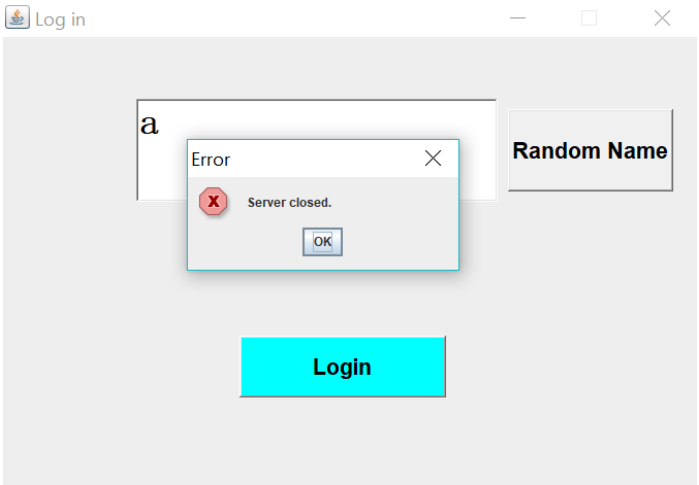

Figure 13. Handle server down exception

If the port number is wrong, there will also be error message.

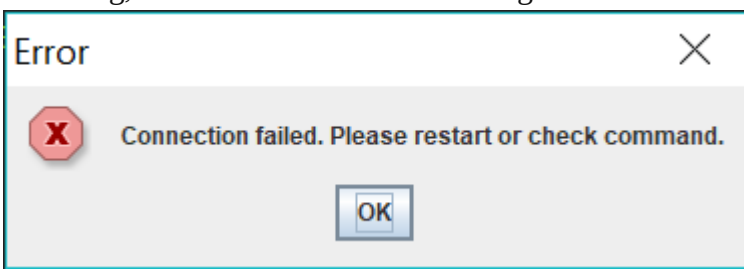

Figure 14. Handle wrong port number exception

4. Critical Analysis

In this project, the key problem is that how to deal with the concurrency. For instance, if a client put a character in the grid, all other players in the game will see the changes at the same time. In the member pool, every player comes to the waiting list first and could choose to join the player list or not. The state of each player is visible by other players in the member pool. When the game is started, the player in the waiting list could watch other players' playing process until someone choose to end the game. There are so many details need to be solved when transferring gaming state to waiting state. In addition, if some new players join the member pool, we need to show the new players' information to the old players. Thus, the structure of data storage and design of concurrency are the most important parts of this project.

5. Conclusion

To sum up, this system successfully realizes a distributed Scrabble Game via TCP socket based on the basic client-server model. Several concise and fully functional GUIs are provided for the players to join the game, invite other players, start game and watch the ongoing game. Although these GUIs has provides sufficient functions, some more powerful functions can be implemented in the future.