# *COMP90015 Distributed Systems*

Assignment 2 Report

Group Member:

ZICHUN ZHU 784145

SHAOBO WANG 935596

YUNXUE CHEN 905136

ZIXUAN LI 811636

Team Name: A Simple Group

Tutor: Minxian Xu (16:15-17:15, Fri)

# 1 INTRODUCTION

In this project, we are required to build a Scrabble Game that allows multiple players to play together. We are following the rules of a traditional Scrabble Game at the same time implementing our own version of the game.

# 2 SYSTEM ARCHITECTURE

We implemented the game architecture using port and sockets for networking, and a TCP connection between the server and clients to ensure a reliable communication. The messages are mainly communicated in JSON format. As the game allows for more than 2 multiple players, a server was built to broadcast the messages with updates to all other clients in a concurrent manner.

## 2.1 Login, Game Membership Pool and Room

When the server is initiated, it allows for multiple clients to connect to the server through an IP address and port number. The user will then login with a unique name and enter the game membership pool. The user becomes a potential player and can see each other in the pool. Any player can invite any other players in the pool, and players accepted the invitation means they are ready and will enter a room. Every player has the choice to start a game or leave the room.

## 2.2 Game Start Management

The game start management allows having at most one game at a time and does not allow people to join the game when playing, but it allows people to exit the game at any time. An instruction of game control will show at the start of a game and the game playing rules will be explained later in this report.

## 2.3 Game Over and Scoring System

Whenever all players pass their turn in one round, or anyone of the last two players leaves the game, the game ends and the Game Over interface will show all players' scores and the winner. Players with the same highest score can win together. The detailed scoring system will be explained later in this report.

# 3 SYSTEM DESIGN

In this report, we implement a Java application that runs on a distributed system. The system is designed to allow several users to play the scrabble game, which is placing letters on a $20 \times 20$-tile chessboard to make words in turns.

## 3.1 Communication Protocols

The system using the mechanism of socket and thread, and provides the TCP protocol for message exchange. It implements the model of thread-per-connection. The reason to use TCP is easy management. All the threads or methods need to be follow the same rules or orders to execute, and TCP is also good for synchronization since every step can be controlled by Server. The order

of the game in turns, thus it is better to use a protocol that can be controlled step by step rather than without order.

## 3.2 Message Formats

In the message exchange, we use JSON as base format to transform information and define several keys in client and server. We implemented a JSON object class for player, the format can be seen in the Fig.0


Fig.0 The JSON format of JSON object class

In the server, the format of message exchange shows in the table 1. Each value of the "Method" key is a different step of game. In each step, the user has been required to send various content. For example, in "join game", the user need to send a JSON object player which is the current user. Since it is a multi-thread game, every time Server needs to broadcast to every thread that make clear every thread can be received same data at the same time.

| KEY(Type) | VALUEs | | | | | |
|---|---|---|---|---|---|---|
| Method(String) | "join game" | "request invite" | "turn" | "vote" | "end" | "update" |
| Players(JsonArray) | | | | | √ | √ |
| Player(JsonObject) | √ | √ | √ | √ | | |
| Potential Players (JsonArray) | | | | | | √ |
| Word(String) | | | | √ | | |
| Map(2D-JsonArray(char)) | | | | | √ | √ |

Table.1 The message exchange format (Server)

In the client, the format of message exchange shows in the table 2 and table3. The game transformation is based on an order, it follows Login, Invite, Start, Turn, Sending word, Voting, Scoring, Turn as a basic order to execute the game.

| KEY(Type) | VALUEs | | | | |
|---|---|---|---|---|---|
| Method (String) | join game | request invite | response invite | leave room | update |

| | | | | | |
|---|---|---|---|---|---|
| Name(String) | √ | | | | |
| Value(int)<br>0 or 1 | | | √ | | |
| ID(long) | | √ | √ | √ | |
| Players<br>(JsonArray) | | √ | | | |
| Player<br>(JsonObject) | √ | | | √ | |

Table.2 The message exchange format (Client-before the game start)

| KEY(Type) | VALUEs | | | |
|---|---|---|---|---|
| Method (String) | turn | vote | end | update |
| Value(int) (0 or 1) | | √ | | |
| Players (JsonArray) | | | √ | √ |
| Word (String) | √ | | | |
| Map (2D-JsonArray (char)) | √ | | √ | √ |

Table.3 The message exchange format (Server – after the game start)

### 3.3 Class Design

This is the UML for Server. Server has Server class, which has main method. In Server class, we will set up socket and port number in order to let client to connect. After running the main method, world class will be executed. World class will generate player arraylist and Player class to save data. The data saved in Player class will be name, ip, socket, port, vote, score and status. Also, World will create listener thread in order to listen the request from Client and transform data with client. Each thread will follow the regular order to listen the request. If no request comes, thread will keep sleeping. For every step, Server will send message to Client every half-second in order to check whether a player is disconnected.


Fig. 1 UML Class diagram of Server

This is UML for client. Since the UML picture is too big for Word, we delete some private attributes such like panel and label in Client's UML picture. Client has a class Main, which has main method in it. In main, user needs to input socket and port in order to connect with server.

After connected, Login class will be executed. There will be a panel appeared and asked users to input user name in order to join the game, and the user name will be saved into Player class. User name cannot be the same with others, if it happens, Server will tell the player that the user name has been exists. For each player, there will be a thread created.

Then, Room class will be executed. Room class can create room to invite players. Room will create thread for each player who has connected with server. In room, there are three buttons, Scrabble Invite and leave room. Player will create room automatically if the player invites someone else to join the game. If all the players come into the room, the creator can click Scrabble and ChessBoard class will be executed.

In ChessBoard class, it will create thread in order to send request to server and receive reply from server, and the game starts. Server will tell the order and let player to input letters and choose the word by using direction keys. After that, the word will be sent to

server and server will ask player to vote. If all the votes are “yes”, then player will gain scores depend on the length of word. There will be a button called “Pass”, if the player does not want to input any word, then the player can use “pass” to change the order. However, if everyone clicks “Pass”, then the game will be over. Any disconnection or one of the last two players quitting the game will call the game over method. Then, a new screen will be put out and print out the winner(s). Player class will be used in ChessBoard, Room and Login classes in order to save player’s data. There is a picture below summarizing the working flow path.


Fig. 2 UML Class diagram of Client

## 3.4 INTERACTION

This is the picture of how the system working. As it mentioned above, both server and client need to create thread to transform information including request and reply. The game needs to be synchronized and thread will be refreshed many times during the game in order to check if any places are not same.

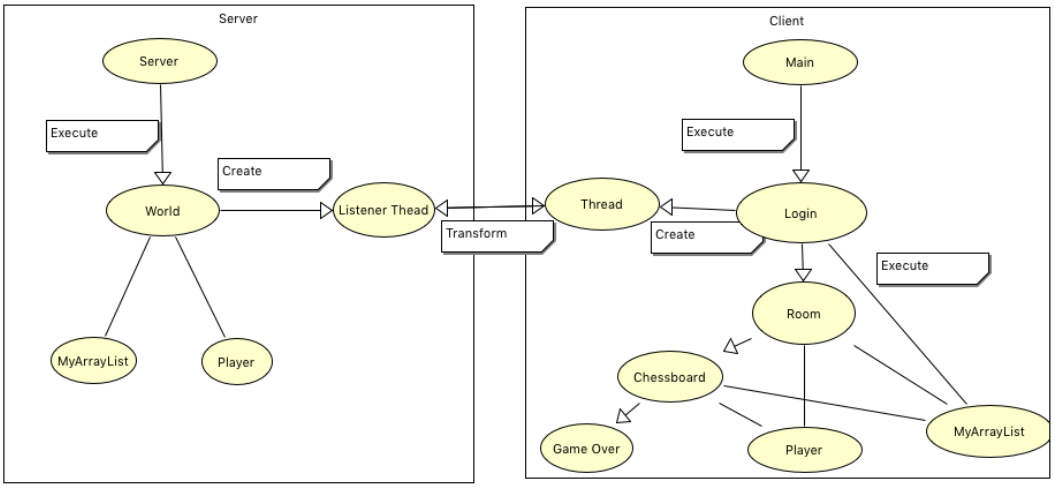


Fig.3 Interaction diagram

# 4 IMPLEMENTATION

## 4.1 USER INTERFACE

At the beginning, the user need to login to the game application (fig.5). There is a game membership pool where users come in and see other potential players (fig.6).

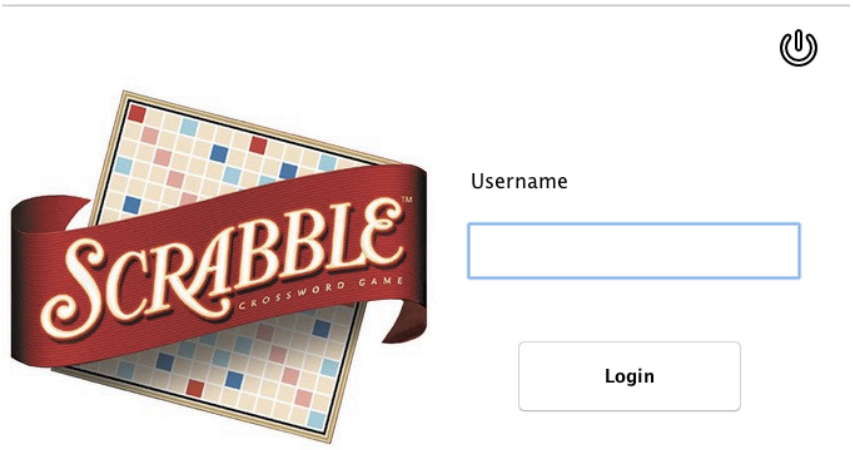


Fig.4 User login Interface.

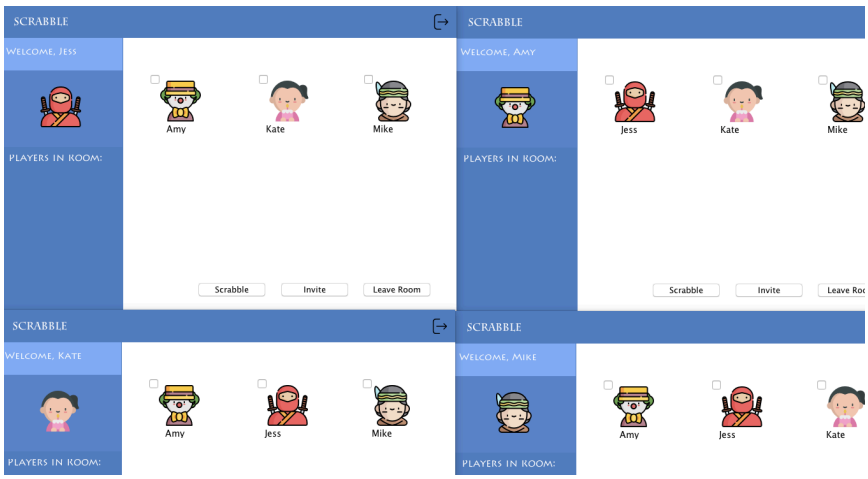


Fig. 5 The game membership pool.

In the membership pool, there are three buttons for user: Scrabble, Invite, and Leave Room. User can invite any other users who display on the membership pool. After the invitees accept the inviting request, these users and the inviter will join in the room, which display on the left side of the interface (fig. 7). Those players who display in the list of room will not show in the membership pool (fig. 8). The button of Leave Room is for the user who already in the room and want to leave the room.

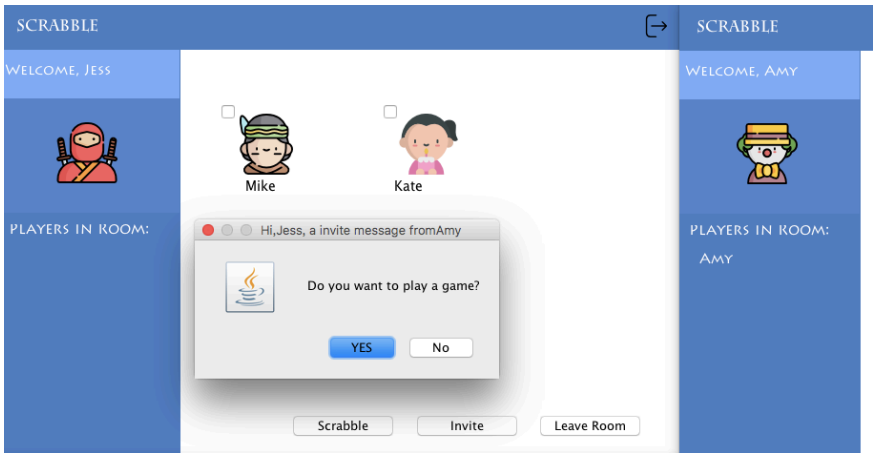


Fig.6 The inviting message.

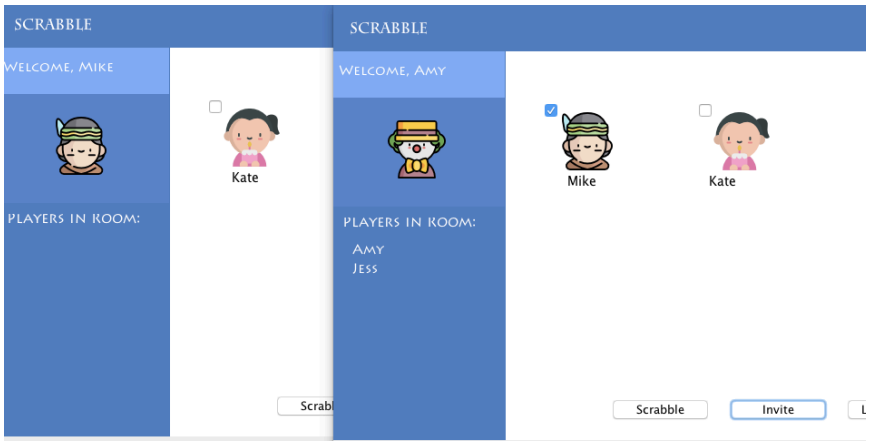


Fig. 7 After invitee accept invites

When a player presses the button of Scrabble, all the players of player list will open the chessboard interface (fig.9) with an instructions (fig.8). There are three values of each player, the first one is the name of the player, the next one is the score of this player, and the last one is the voting result of them. The player need to waiting for other players all vote the result.

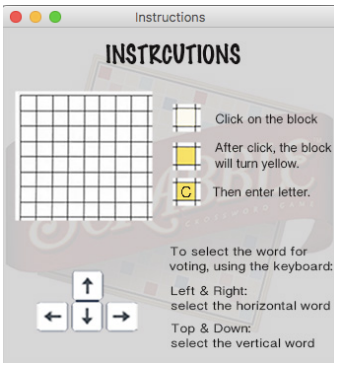


Fig.8 Instructions interface.

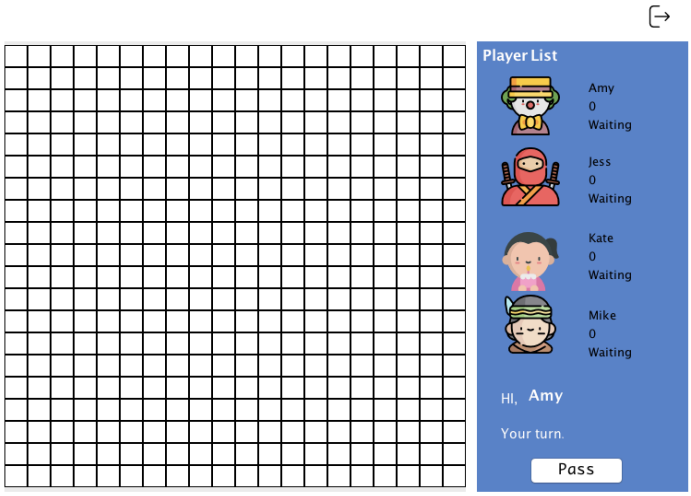


Fig.9 Chessboard interface.

When the player places a letter, and uses the keyboard (up or down, left or right arrow key) to decide which voting word to send for other users (fig.10).

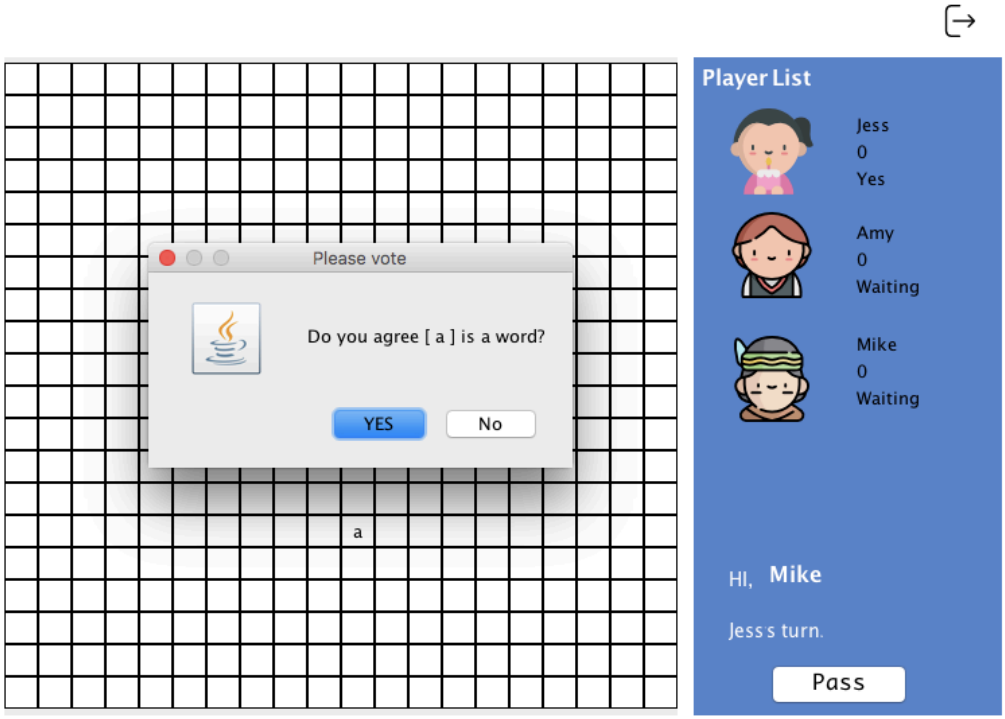


Fig.10 The vote interface

The game ends when all users have a turn and all of them say pass, the result will show in the game over interface (fig.11). If one of the player logout from the game, the socket of this player will close and the rest of player will update the player list.

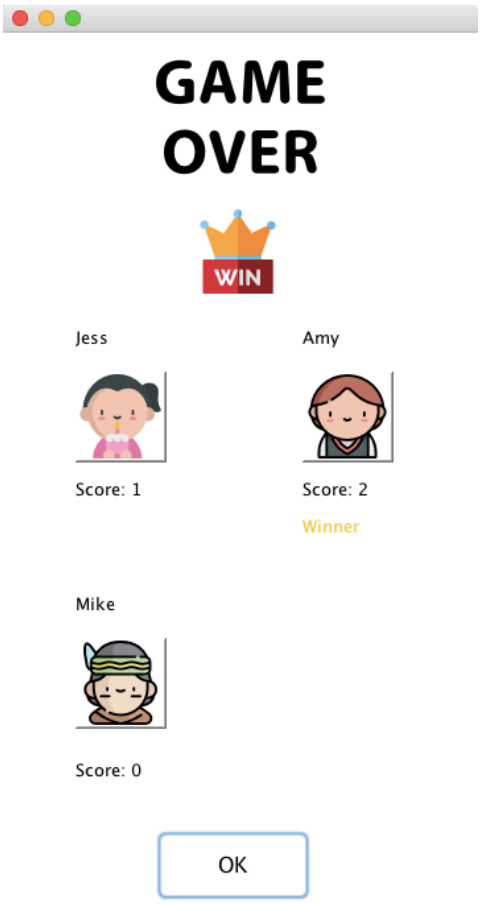


Fig. 11 Game over interface.

### 4.2 FAILURE HANDLING

There are several failures handle by the system. Firstly, the name cannot be duplicate, if the name was repeated, the system will show the error message to user. Then, the server checks the connection of each players in every 500 ms, if any user has loose connection, the server will delete this player from the player list. Moreover, we deal with the concurrency issues of the system. Lastly, each button of this system and each input character has been checked under the rule condition. For example, if there is no player in the room or the user does not choose any player to invite, the user cannot invite. The player can only input alphabet in the game otherwise the error message will pop up.

## 5 CONCLUSIONS

In conclusion, the system using the mechanism of socket and thread, and provides the TCP protocol and uses the JSON format for message exchange.

# Appendix A - Contribution of each member

| Team Members Names | | ZICHUN ZHU | SHAOBO WANG | YUNXUE CHEN | ZIXUAN LI |
|---|---|---|---|---|---|
| General Aspect | Specific Aspect | Team Member 1 | Team Member 2 | Team Member 3 | Team Member 4 |
| Sever | Message transport | √ | | | |
| | The design of Message format | | √ | | |
| | Dealing with concurrency | √ | | | |
| | The design of game logistic | | √ | | |
| Client | GUI | | | √ | √ |
| | Message handle | √ | | √ | |
| | The design of game logistics | | | | √ |