UNIVERSITY OF MELBOURNE

COMP90015 DISTRIBUTED SYSTEMS

ASSIGNMENT 2

DISTRIBUTED SCRABBLE

Group: Infinity Monkey Theorem
Team member: Yunfan Yang
Xu Wang
Jun Wang
Yuming Shen
Swapnil Shailee
Tutor: Xunyun Liu

1. Introduction

Scrabble is a very popular crossword puzzle, multiple players place letters on one board to form a word to earn points, and the player with the highest total score will win the game. The purpose of this project is to develop a multiplayer online Scrabble game include the server and client by using Java programming language. The player opens the client to connect server and plays the game with other online players. Our design allowed multiplayer play multigame at the same time. The system will automatically handle the player's actions and push the game's progress.

The application was developed by using the concept of sockets for communication between the different clients and the game server. TCP Sockets are used for communication. The concurrency in the game is achieved through multithreading. The client has two thread and the server has multithread which is the thread-per-connection structure. Threads help in executing one or more tasks concurrently in a java program. The message passing between different clients and the game server is IO objects stream, implemented by sending objects through sockets using the concept of serialization. The error handling is implemented through Exception handling in Java. And the Graphical User Interface is implemented using JavaFX and Cascading Style Sheets.

2. Architecture

In this project, the system architecture is Client-Server architecture and used the classic two-tier model. The Client has two thread one thread is for GUI and catches user's operation then send to the Server, the other one is listening to the response from the Server and represent on the GUI. The Server used multithread, the server threading architectures is thread-per-connection. The main thread will always listen for the connection request, once the client connects to the server, the server will create a new thread for the client. The client will interact with the thread for it on the server. The broadcast will be used on Server to transfer message to Client.

3. Protocols and Message

In this project, the commutation is socket-base and we used TCP protocol to exchange the message between server and client. Socket-Base communication is programming language independent, it can improve the heterogeneity, openness, and scalability. TCP can provide reliability of the message exchange. The transmission is orderly, which can perform ordered operations well. The message format is serialized object when sending the message, we first do the serialization which is the process to convert the object a byte stream, and then write the byte stream the data stream. We will use the IO object stream to transfer the data stream, when the object is received, to subsequently deserialized then a copy identical to the original will be created. When the server received the message from the client, it will broadcast to the other users if needed.

The message class in server and client are in the same format and register with the same serialVersionUID. The message object contains all the information that needs to be transmitted, including name, word, operation, etc. al. The interaction diagram is shown below.

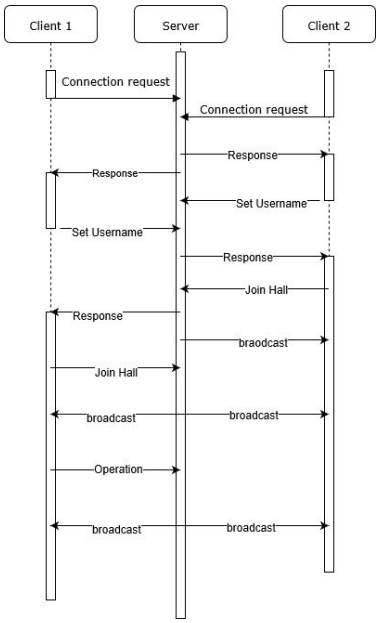

Figure 1: Interaction diagram

4. Game Processing and GUI

The client application has 5 stages, login, set username, hall, game preparation, and game. The system diagram is shown below.

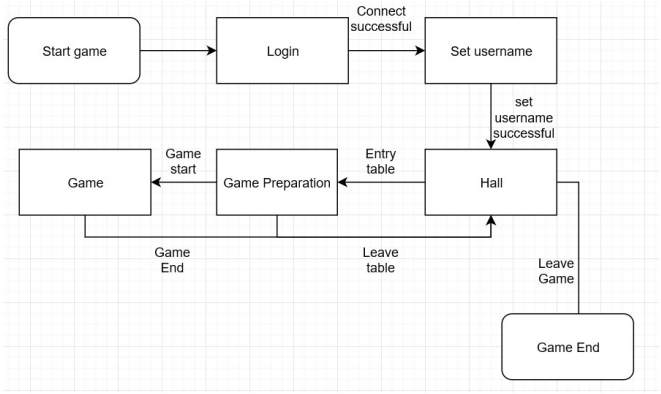

Figure 2: System diagram

4.1 Login

User input the host address and port to connect the server. The user can click on the Connect button to connect to the Server. Alerts are implemented in case of the incorrect host or port number, login failure or if the server is not working. If the connection is successful, the application will go set username stage.

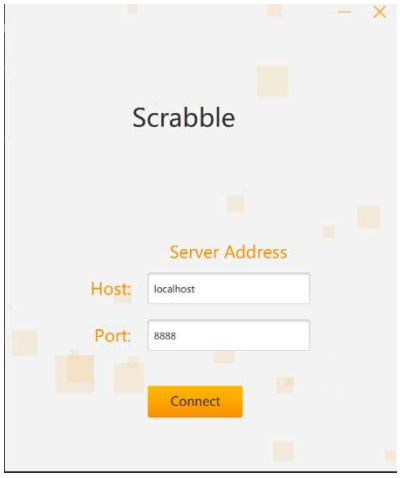

Figure 3: Login UI

4.2 Set username

After successfully connecting to the game server, the user can enter the username or can randomly name pool of usernames by click the dice. The name will send to the server to check the duplication. Alerts are implemented in case of duplicate usernames. If the name is validation, the game will go Hall stage.

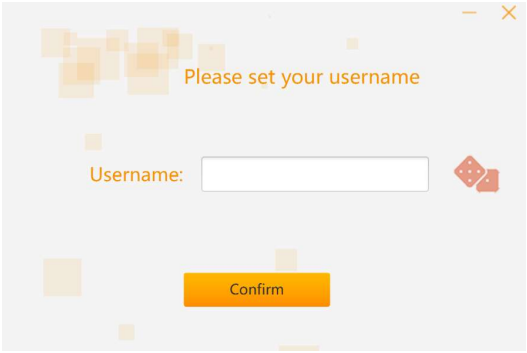

Figure 4: Set Username UI

4.3 Hall

The game hall window shows an advanced feature developed where the user can select from 12 games. Each game can have almost 4 players. The player can view the potential players that is their username and player status. The invitation information will show on the down right. The user can click the table to join the table if the server feeds back that the table is allowed to be joined, it will move to the game preparation stage. The user can also click close to leave the game. Alerts are implemented in case of any failures.

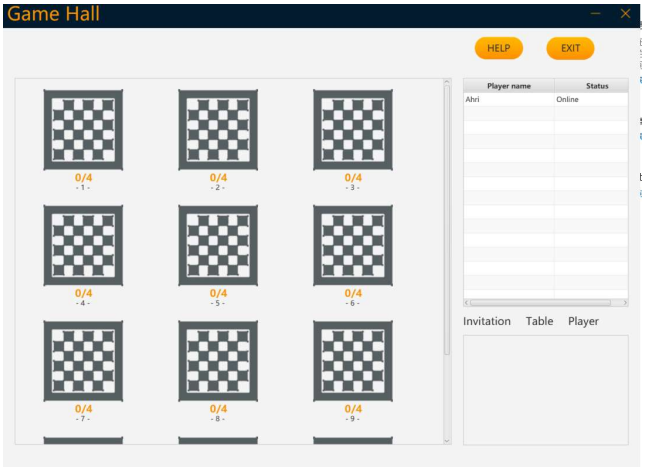

Figure 5: Hall UI

4.4 Game Preparation:

In game preparation, the left shows the player information, the middle grid is the board. The user can click ready to be ready for the game. Once the player number larger than 2 and all the player clicked ready, the game will start and go to the Game stage. The user can also click invite to invite the other users on the Hall stage. The user has been invited can choose to join or reject, and the user who sends the invitation will get the response. The maximum capacity of the table is 4. User can close the table window or give up to exit the table and return to the Hall.

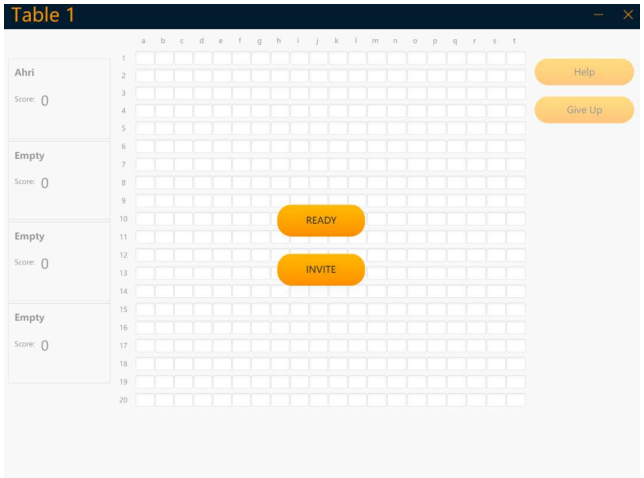

Figure 6: Game Preparation UI

4.5 Game stage:

In game stage, the left side is still the information of the players, the game process uses a turn-based system, and each player's action will be fully displayed to all other players in the same game. The player places a letter on the board every turn and decides whether to vote. If the vote is successful, it can obtain the score. The player can also click Pass to pass the turn. The player score and turn status are updated accordingly. Once the user clicks the Give Up, or close the window, it will be removed from the game and game will keep running if the game remains players more than 2. When the game board is full, or if the player leaves so that only

one player remains, or all players choose PASS, the game will gracefully end and announce the winner.

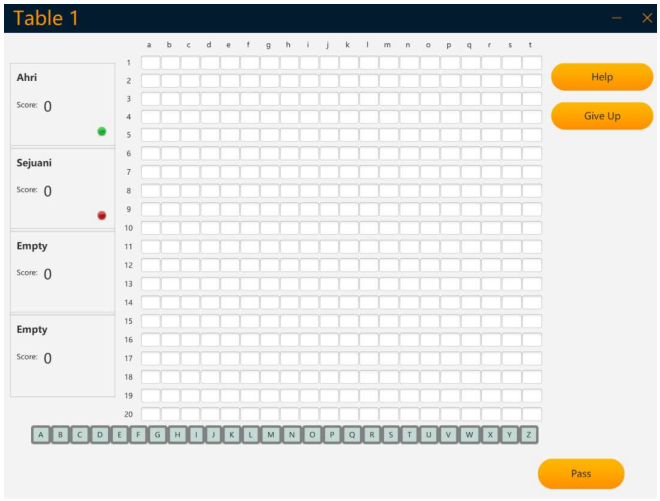

Figure 7: Game UI

5. Class Components

5.1 Client

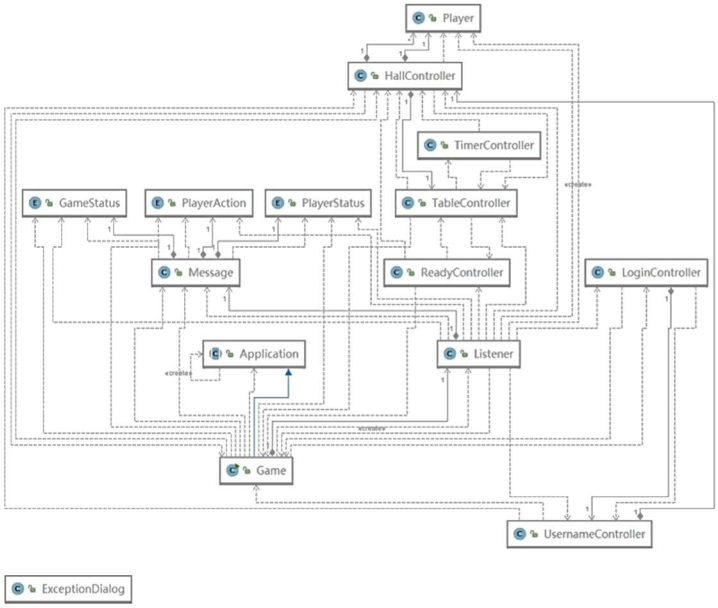

Figure 8: Client Class Diagram

- Game.java: Contains the main method to start the game, connect method to connect to the server through sockets and other important methods for message passing to the game server.
- Listener.java: Extends the Thread class to provide concurrency in the application. Based on game and player status and action, different methods are called.
- Messages: Contains the Message.java which contains the accessor methods to obtain player score, player list, game status, voting result etc. It also contains the enum classes GameAction.java, PlayerAction.java and PlayerStatus.java

- Model: Exception.java is used for handling some errors. Player.java contains accessor methods to obtain and set player status and username.
- View: Consists the hall, table, login and username GUI views along with their controllers.
- Controllers:
  - LoginController.java:
    It contains functions to login to the game, connect to the server, alerts are generated in case of login failure and connection lost. It consists method for animation.
  - UsernameController.java:
    It contains functions to login to the game hall, are generated in case of duplicate username. It consists method for animation.
  - HallController.java:
    It contains functions for entering different game tables, showing tables, table failure, showing player list, invite players to the game, help and exit.
  - ReadyController.java:
    It contains functions for entering the game table, confirm and exit.
  - TableController.java:
    It contains functions for setting up the game board, setting player status, score, turn, setting letter buttons, pass, voting, confirm, clear, help, return to hall.
  - TimerController.java:
    It inherits the thread and counts from 3 to 1 when two or more players join the game. It redirects to the table controller.

5.2 Server

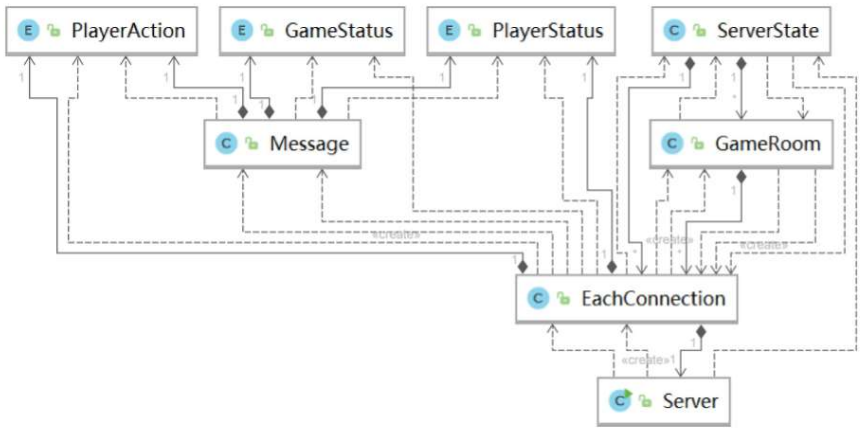

Figure 9: Server Class Diagram

- GameRoom.java: It contains methods for adding players, removing players, game results, voting results, player status and score, pass result and other important accessor methods.

- Messages: Message.java consists of important accessor methods for player score, result, status, voting, game result, game info, player list, game word etc. The enum classes Gamestatus.java, PlayerAction.java and PlayerStatus.java is same as that of client.
- Server: It contains the Server, Server State and Each Connection java files. The server is responsible for creating the server socket to listen for client connections. For each connection, one thread is created.
  The Each Connection class implements the Runnable interface based on player and game status or actions different methods are called from Game Room. The Server state consists of synchronized methods for games and clients lists, connected lists and disconnected lists.

6 Contribution

We have 5 members in the group, every one shows good coding skills and distributed system knowledge, team members all attend meetings on time and help each other, complete their work in time.

Outlines the contribution of each member is as follows

- Architecture design: Yuming Shen & Xu Wang
- Server implementation: Yunfan Yang & Yuming Shen
- Client implementation: Jun Wang & Xu Wang
- Client GUI: Xu Wang & Swapnil Shailee
- Test & Debug: Jun Wang & Yufan Yang
- Report Jun Wang & Swapnil Shailee

7 Conclusion

In this project, we used TCP socket and multithread techniques to develop an online multiplayer Scrabble application. We have a good cooperation, after this project, we are become more familiar with the distributed system knowledge and learned how to work in a team.