DS A2_Scrabble Game
Final Report

By group: 404 NOT FOUND

Group Members:

- **Yuanlong Zhang** (772312)
- **Xi Chen** (7983241)
- **Shaoyang Ye** (864716)
- **Pengnan Zhao** (883338)

Executive Summary:
This report is

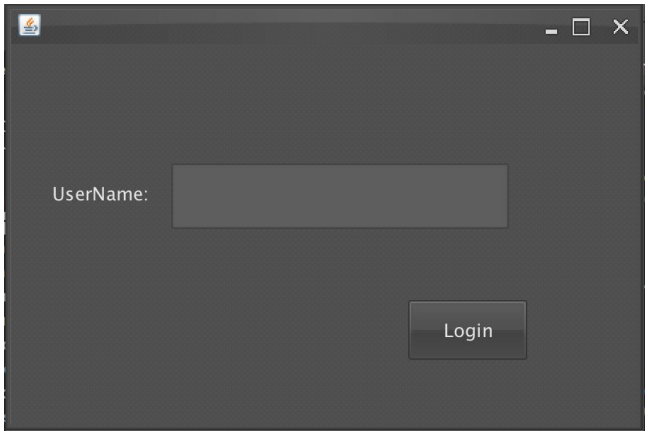
Table of Contents

Scrabble Game Intro:

1. The game is played by one to five players on a square board with a 20×20 grid of cells.
2. A player can create a room and invite friends to play together or play alone.
3. A player can communicate with each other(online players) through a chat window.
4. When the game start, the player in its turn can add a letter or pass its turn.
5. A player can only add one letter in its turn, numbers or more than one letter are invalid.
6. After a letter is added, the voting procedure will start and score according to the vote result.
7. If there is the same word in different place, it still counts.

Game user interface intro:

Login GUI design:



Layout:

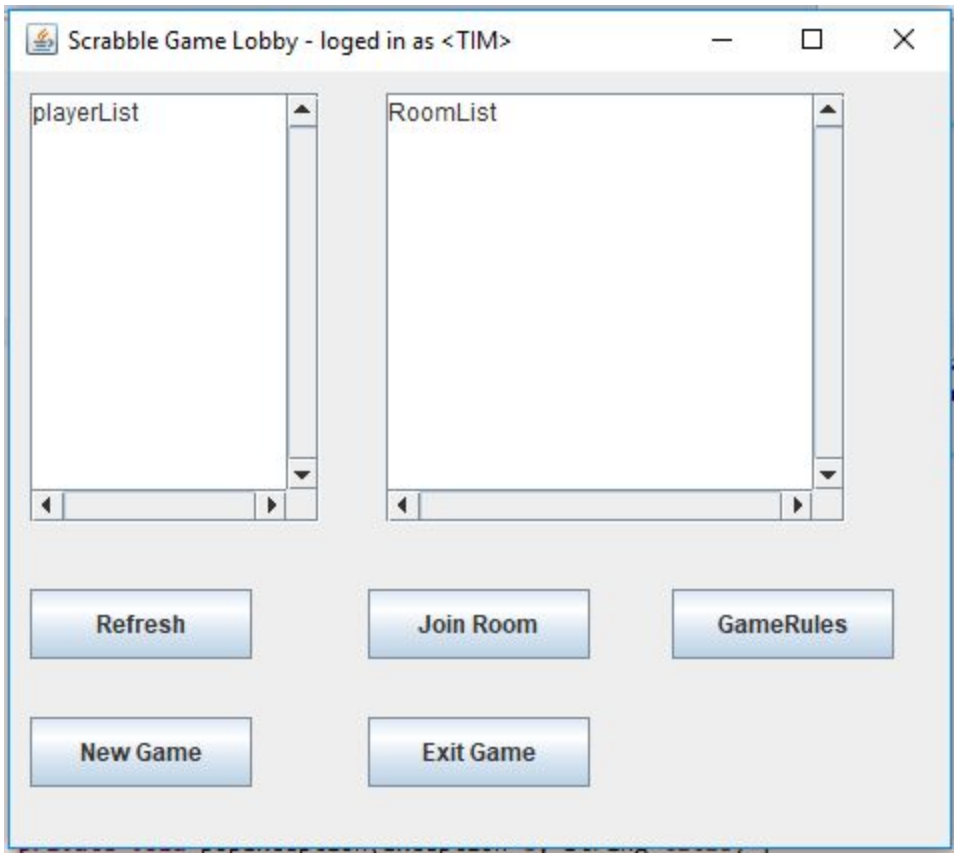
Upper region: One text field(can type in).

Bottom region: One button.

Specific:

8. **A textArea** for typing in the username
9. **A Login button**:
    a. If the username already in use, an error message window will pop up.
    b. If the username is unique, **the game waiting room window will open.**

Game Waiting room design:



Layout:

Upper region: two textArea(cannot edit).

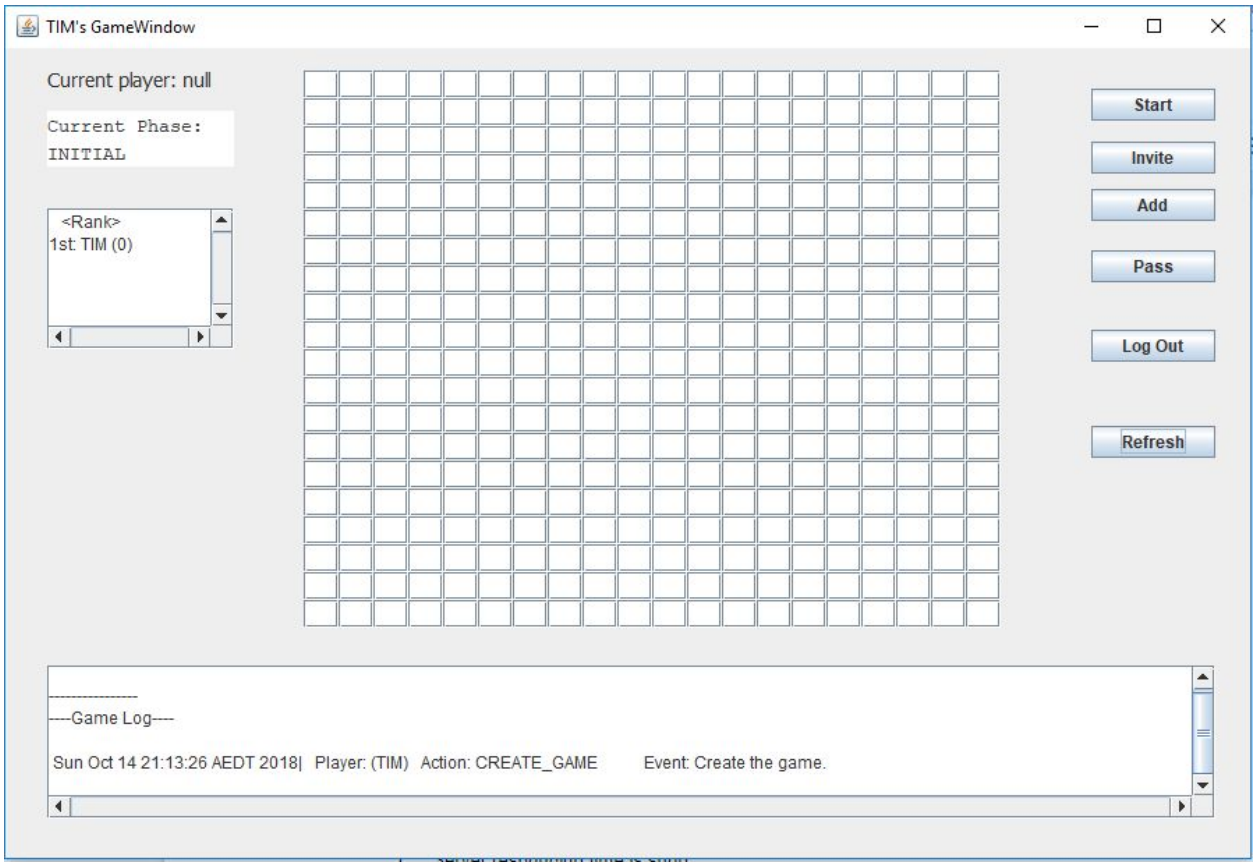
Bottom region: Five buttons.

Specific:

1. **A TextArea** to display all current **online players** with scrollbar
2. **A TextArea** to display all current **online game rooms** with scrollbar
    a. For viewing all existing games.
3. **A Join button** to join an exist (Extra function)
    a. Press the join button to popup a window with a text field.
        i. Type in the room id (4 digits number) to join a game.
            1. If the room id exists and is not created by the player itself, the game window will pop up and the player has joined the game starts the game owner start, the player can start play.
            2. If the room it does not exist, a window will popup show the error message to the user.
4. **A Create Game button**:
    a. Create a game room with the player become the game owner, If the player already creates a game, an error message window will pop up.
    b. If the user has not created any room yet, a **game window will popup**.

5. **An Exit Game** button to log out the server
    a. A popup window that asks for confirmation (Extra function)
6. **A Game Rules** button: game rules (Extra function)
    a. Press the button to popup a new window for our version game rules.

Game room GUI design:



Layout:

Upper region:

Left side: Two textArea(cannot edit), one text field(can type in), one send button.

Middle: The 20*20 game board.

Right side: Six buttons.

Bottom region: One textArea(cannot edit).

Specific:

1. **A textArea** to show game phase, “it is TIM’s turn” for example.
2. **A textArea** to show players list with a scroll bar
    a. The order of player is ranked by their scores.
3. **A textArea** to show chatting info with scrollbar
4. **A text field** to type in the message that the player wants to send

5. **A Send Button**:
    a. If textfield above is not null, send the message to the server.
6. **A Game board** in the middle
    a. The player can choose a grad in his/her turn to type in a letter or pass.
7. **A Start Game button**:
    a. The player who create the game can start the game by pressing this button.
8. **An invite button**:
    a. Press the invite button to popup a window, the window has listed all online player username and there is a text field where a player can type in the username that they want to invite.
        i. If the username is in the online player's list, the player that get invited will have a window pop up to ask for confirmation. If the player accepts the invitation, a game window will pop up and the player has joined the game. Else the player will not join that game.
        ii. If the username is not in the online list, a window will pop up to show the error message.
9. **An Add button**:
    a. If the player chooses a grid and type in one letter and press add button in his/her turn, a window will pop up to ask for confirmation, if the player confirm to add the letter, the game board will update and voting procedure will start, else the game board will return to the state before the player add unexpected input.
    b. If the player chooses a grid and type in more than one letter or numbers then press add button in his/her turn, a window of error message will pop up and the game board will return to the state before the player add invalid input.
10. **A Pass button**:
    a. The player can press the pass button if he/she does not want to add any letter to the game board, and it will go on to the next player's turn.
    b. If all players press the pass button in one round, the game will be over and the player with the highest score will win. And a window with players username and player scores will popup(also ranked according to scores). Then all player will return to their own game waiting window.
11. **A logout button**:
    a. If any player press logout button, the game will be over and the player with the highest score will win. And a window with players username and player scores will popup(also ranked according to scores). Then all player will return to their own game waiting window.
12. **A textArea** for showing **game log** at the bottom

Game System design:

Distributed System Communication Design:

Communication design:

1. Design choices

The important thing to start structure and communication structure:

- Server-Clients Vs Peer-Peer **[Chose: Server-Clients]**
- Socket programming Vs RMI **[Chose: RMI]**

To start with the basic system structure, we need to figure out all the requirements first:

1. All players can play the Scrabble Game in the system
    a. Scrabble game itself is a 2D and almost static game, which means LOW NEED for COMPUTATIONAL POWER!
2. Players can play with other players on the same game board
    a. That means the system needs to allow the player to join and leave at anytime
3. All players share the real-time info
    a. Which means needs to secure the communication quality. I.e. Reask for data if response not received.

So, Compare with Server-Client and P-P:

- S-C
    - Con: Easy to build, resource centralized
    - Pro: Once Server is down, all system stops working
- P-P
    - Con: Service stable and resources are distributed through the system
    - Pro: hard to manage and maintain

Compare the cons&pros we decide to use the **Server-Client** for structure.

There are three reasons for choosing the RMI technique.

1. Firstly, RMI can pass complete objects as parameters and return values, not just predefined data types.
2. Secondly, object delivery capabilities make full use of the power of object-oriented technology in distributed computing, such as two and three-tier systems.

3. Finally, RMI uses Java's built-in security mechanism to secure user systems.

2. Details of the design

The communication process for chatting between players is shown as the following figure:

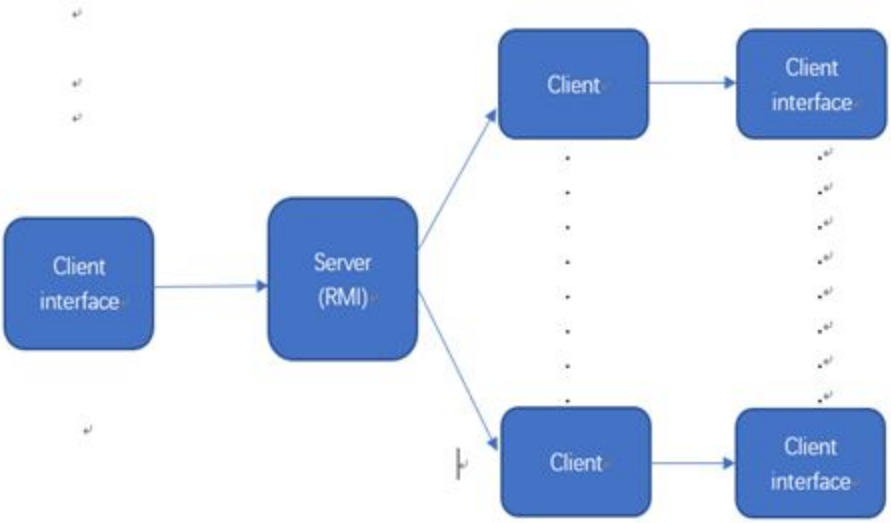


Figure 1: Chat process

As shown, firstly, text is scanned from the client interface and then sends to the server. After that, using the RMI technique to transfer the information to all the clients. Finally, the client interface will read the information and print it on UI.

The process of communication for a multiplayer game is represented as the following figure:

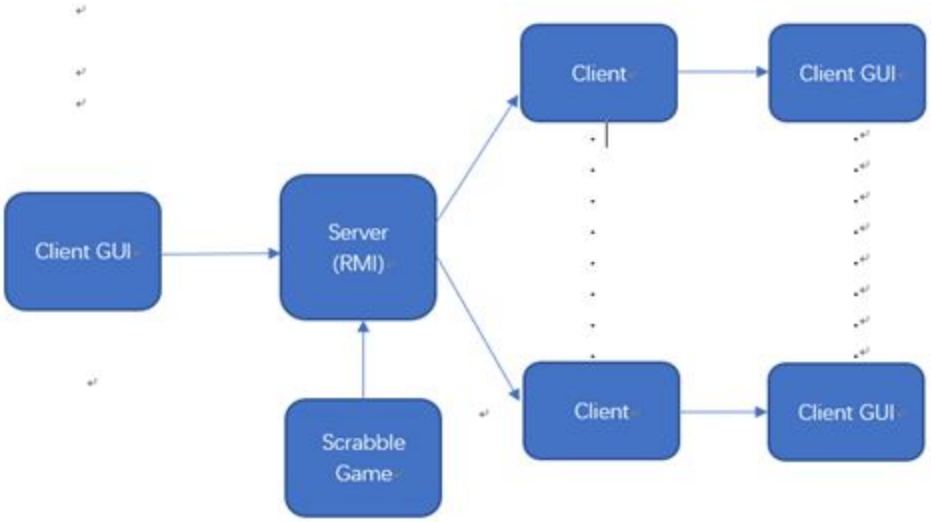


Figure 2: Multiplayer game process

As the chat process, the information needs to be read from the client GUI. Then, the server needs to call a backend method, do some operation to the information and get the game data. Finally, the game data will be sent to all the clients so that the client GUI can read and print it.

For the server structure, there are three main methods: registerclient, broadcast and broadcast_chat.

Ø Registerclient

A hash map is used to store all the clients and their username. Specifically, when a new client comes, a username needs to be entered and the server will determine if the user name is unique, if it is unique, the username will be stored in the map.

Ø Broadcast

This method is used to broadcast game data to all the clients for implementing the multiplayer game. All the client can call this method based on the RMI technique.

Ø Broadcast_chat

This method is used to broadcast chat messages to all the clients. To be more specific, the method uses the RMI technique and all the clients can call it remotely

There are three main methods for the client structure: retrieveMessage_chat, retrieveMessage, and Run.

Ø retrieveMessage_chat

The method is used to receive chat messages from the server and store it on the client side.

Ø retrieveMessage

This method is used to receive game data from the server and store it on the client side.

Ø Run

This is a thread, for listening to the user actions, if there is new action, the action information should be sent to the server.

3. Function

In the communication part, four functions are implemented.

1) Chat function

The function is used for chatting between clients. For example, they can send messages to find players and create games.

2) Determining whether the user name is unique

All the users need to provide a unique username for identifying. The function will remind users that the username they entered is not valid. In addition, the logged in users will also be shown on the game lobby GUI.

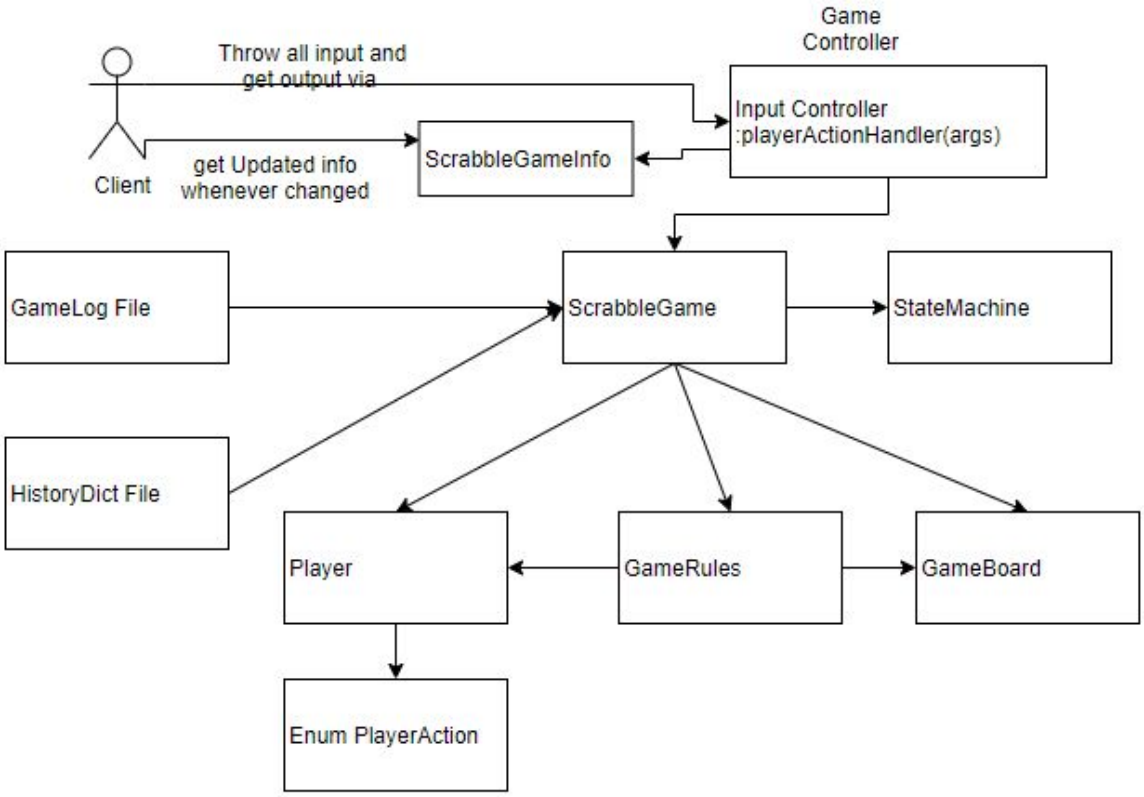
3) Transferring the game data

This function will transfer the game data to all the users for implementing the multiplayer game.

4) Showing clients' information on the game lobby

Much of the information that needs to be displayed on the GUI is transferred through the communication part such as player rank, chat log and game log.

Backend Game Design:

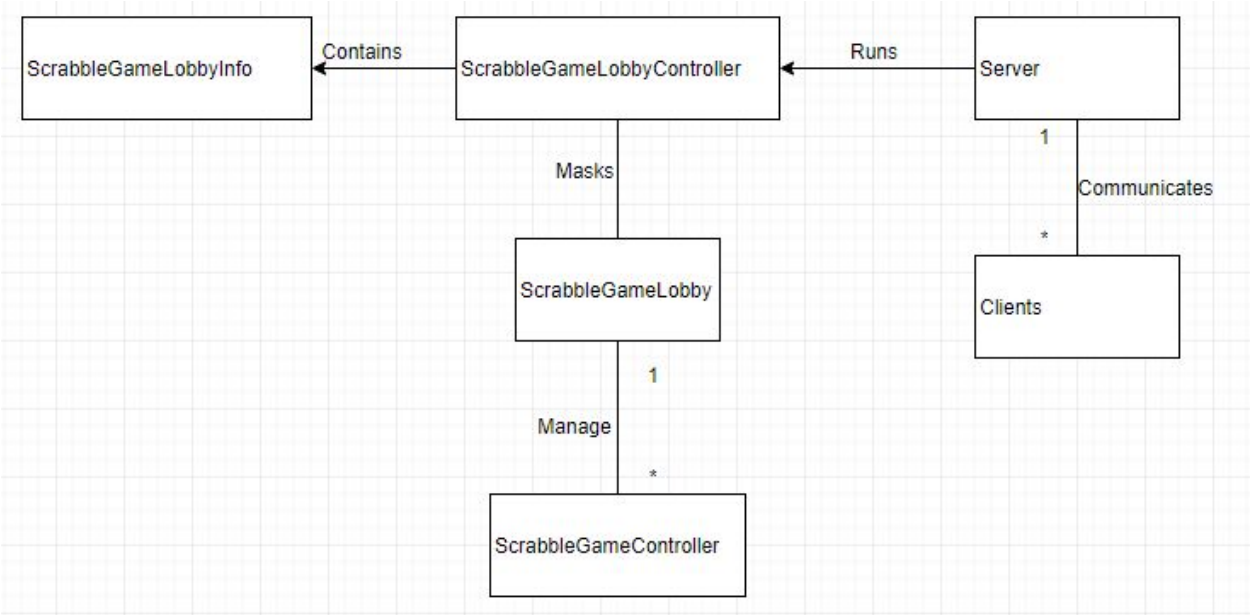


How this works:

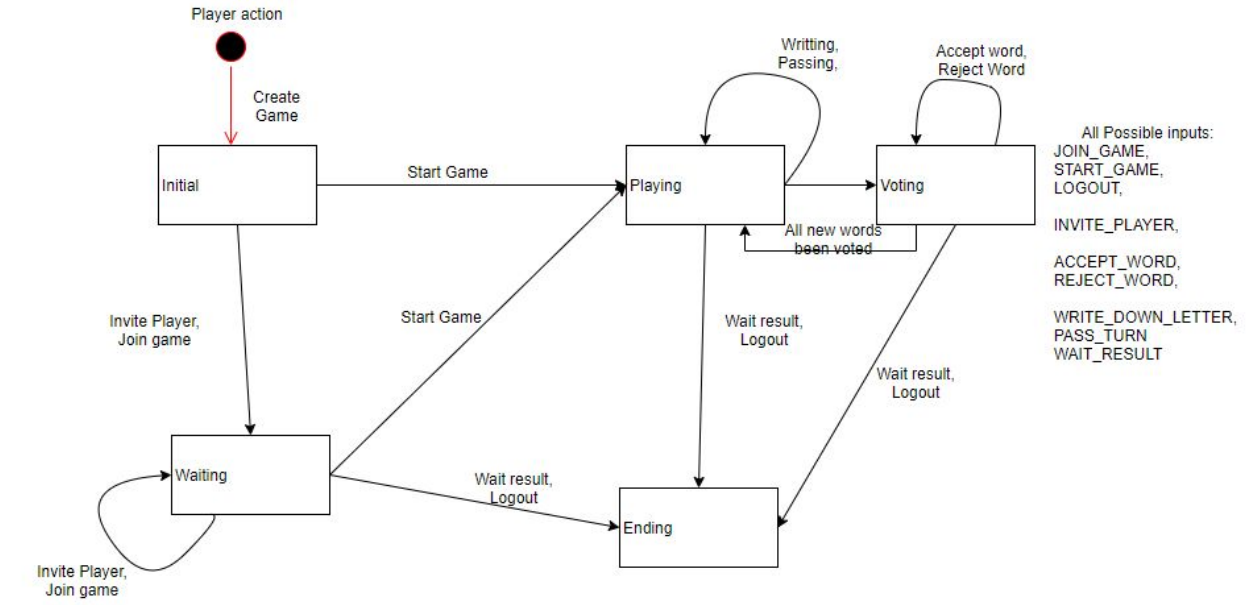
Client access Game Controller via RMI interface. The controller will consume clients' input and change the game data stored on the server.

Then the client can get the updated game data and render it on GUI.

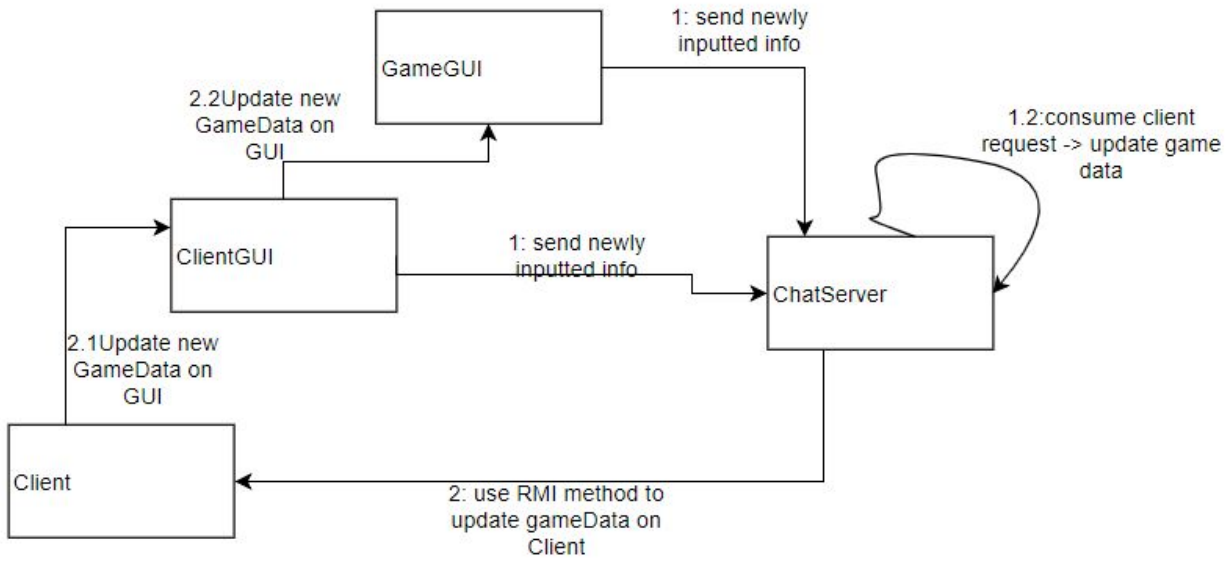
<Table of Contents>



In order to manage multiple games, we create a game lobby to do this. Each game controller represents one individual game, and one game lobby manage all the game controllers.



The state machine for the game. This state machine a easily manage all the behavior of the game, prevent all invalid actions. E.g: after a game start, no more players are allowed to join the game.



This is how the game data is been updated inside the game, it forms a loop, then the once the game data has been changed. All the relevant component will be updated.