A distributed scrabble game

Tutor: Shashikant Ilager

Team: all for one

Member:

Name	Student ID	Username
Yishan Shi	883166	SYishan
Huiya Chen	894933	huiyac
Tong He	867488	the2
Yao Wang	869992	yaow15

1. Introduction

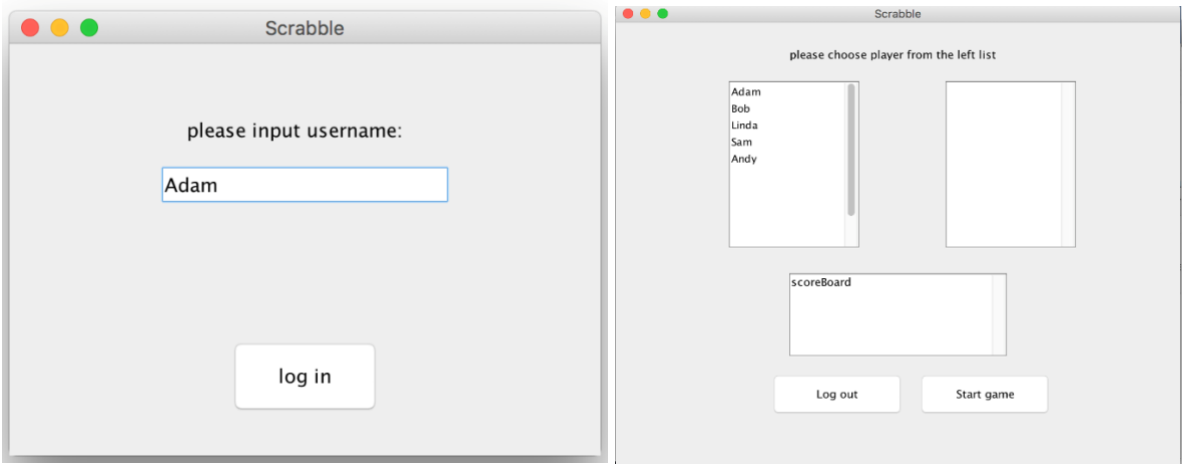
In this project, we designed and achieved a distributed scrabble game. When the client user connects with the game server, he can see the other users, if he wants to start a new game with other users, he can invite them by choosing usernames in left membership list. Once he clicks the start game button, a scrabble game will start. During your own turn, you can only insert letter in same row or same column. If you think you finished a right word, you can apply for vote, after all users agree, you will get marks. When all users pass without successfully applying for vote, or when any user exists, this game will end and all users can see the results. The two fundamental technologies used in the distributed game are sockets and threads.

2. Game architecture

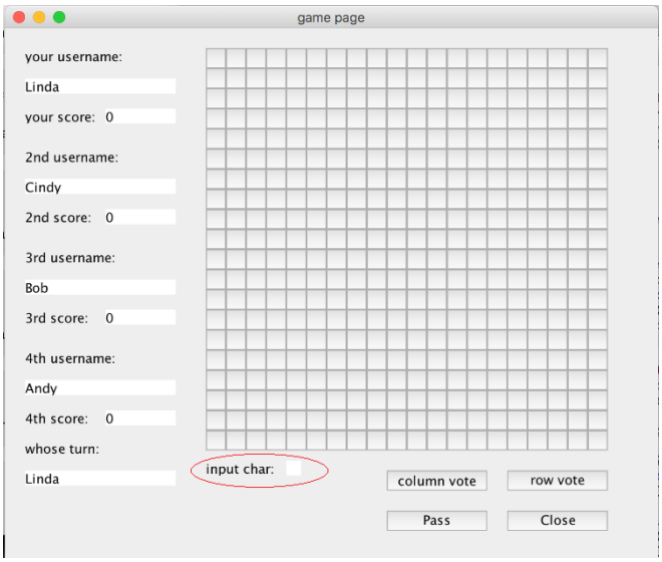
2.1 Interface

There are three interfaces for players.

- Login interface (figure 1a): This is the first interface when start a game. User are asked to input a username to login. The username is unique identifier for them. After this, press “log in” button, invite player interface is displayed.
- Invite_player interface (figure 1b): Online player is shown in the left user pool. Players are free to select at most three other players to start game. Of course, they can start game directly and play single-player game. There are three boards in this interface. The left board shows all online players. Choose any item in this list and the selected player will be listed in the right board. The bottom board record player’s score. Every time a game is over, the score will be added in this board.
- Game interface(figure 1c): Users see this interface when they start a game or are invited to participate a game. This interface shows a 20 * 20 grids in which users place words. Other information of this round such as usernames and scores for each player is also shown in this interface. The text in left bottom corner shows whose turn to place word. Users can not input characters if it is not their turn.



a. login b. invite player



c. game

Figure 1. interface

2.2 Game process

1. The game interface is shown in the figure 1c. Players take turns to play game.
2. In your turn, you can input a char in the place that is circled, and click button where you want to put the character in the grid. User are allowed to input alphabet several times until they think it is a word. However, they can just input word in a row or column. And except the first character, others have to be adjacent to an existing alphabet.
3. When you finished input, if you think word in row or column (or both) is a word, you can press “column vote” or “row vote” button(or both). If you cannot get a word, please press “Pass” button to give up this turn. Each button can only be pressed once. After press vote button, it is not allowed to input character again. The vote response is shown in figure 2:

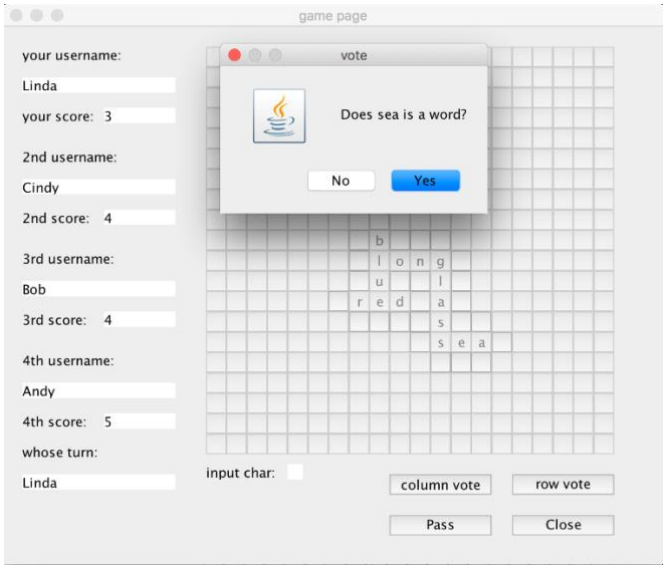


Figure 2. Vote

4. The vote window will be broadcasted to every player, if they think this is a word, they press “Yes” button. If all users press “Yes” button, the user can get score same as the length of word. The score of users will also be updated as shown in figure 2. After vote, users press “pass” button to show his(her) operation has finished, another user can input characters.
5. If one player presses “close” button, then this round of game will end. Current grade will determine the winner(Figure 3). Same thing happens if all player press “pass” button.

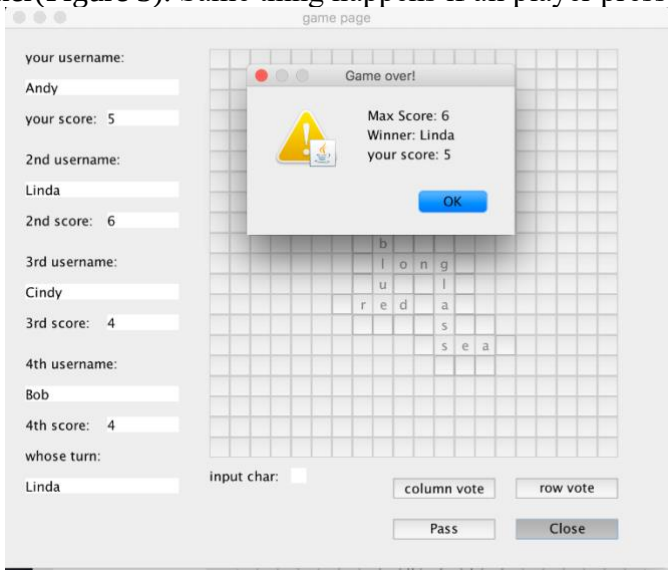


Figure 3. Game Over

6. After finish a round of game, the grade will be returned and record in the scoreboard as Figure 4 shows.

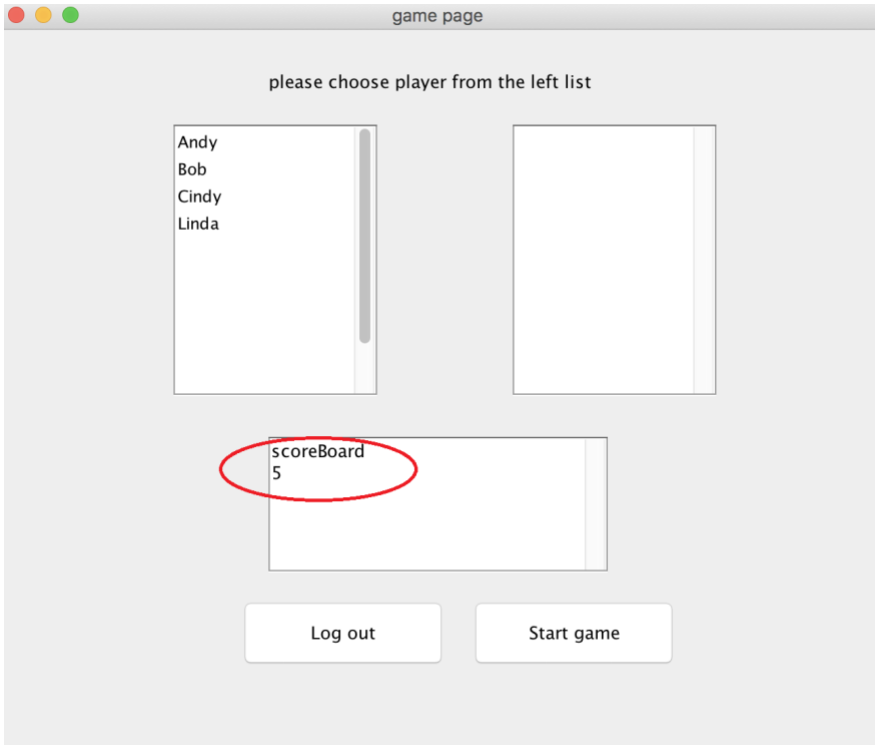


Figure 4. updated scoreBoard

3. Class design

3.1 UML diagram

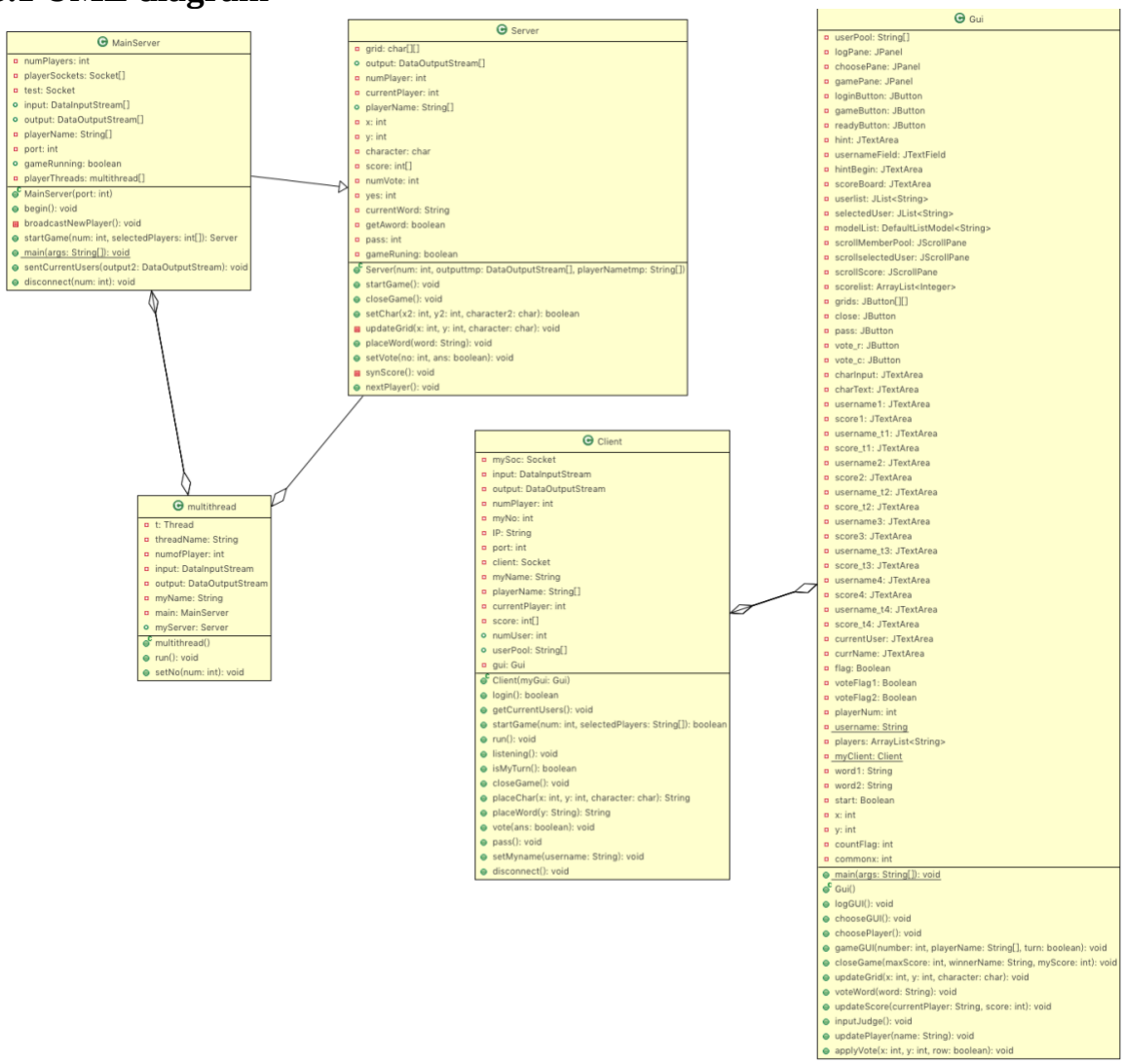


Figure 1 Class UML

3.2 class explanation

There are 5 classes in our project .
MainServer ,Server ,multithread, Client , Gui.

Client part include Client & Gui:
The **Gui** class is responsible for interacting with the user. Handle new, display, and modify different windows. At the same time, the user's input is restricted and processed, and the communication with the server is realized through the call of the Client class method.

The **Client** class handles communication-related functions, such as establishing a TCP connection with the server and sending and receiving messages. In order to maintain both input and output, the received message is monitored by a separate thread.

The two classes handle the user interface and background interaction logic separately. The user starts with the Gui class, generates its one-to-one Client class, connects to the server, and

starts communication. When the user enters from the GUI interface and calls the Client class function to handle.

Server part include MainServer ,Server ,multithread:
MainServer:
This is the only primary server that maintains the entire user pool. All users connect to the primary server first, and then the main server creates a new thread multithread class to receive user input. When you start a new game, the Server class is built to handle the game logic.
multithread:
Each logged in user has a corresponding multithread thread listening for his input. According to the different instructions read in, multithread goes to MainServer or Server to call the corresponding method.
Server:
The Server class is in charge of all the logic of a game server like checking the voting of words. It is called by multithread to get the information and broadcast it to synchronize all players when appropriate.

4. protocol and message description

This game system uses java Socket for network comunication. Each player establishes a TCP connection to the server, which guarantees the reliability of the communication. When communicating, both parties will first send a UTF to indicate which type this message is. The receiver receives the relevant parameters in turn according to UTF. Each user has a separate thread multithread to receive the data, and then call the corresponding functions in other classes.

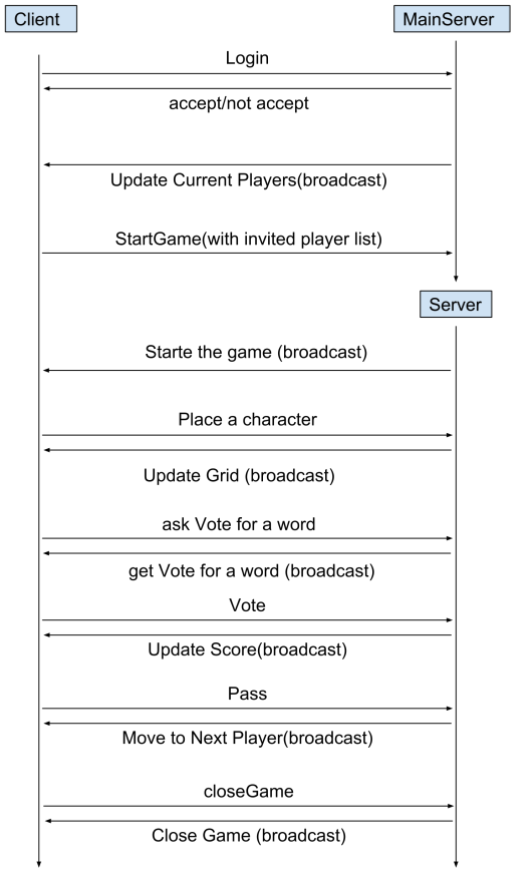


Figure 2 Communication flow chart

5. Exception handling

5.1 client exception handling

a. Unique username for log in

when a user runs the client, the log in page is shown. The user needs to input a username as the unique identifier. If the username sent has been in server's user pool, a warning page will be shown, the user needs to resend a different username for log in.

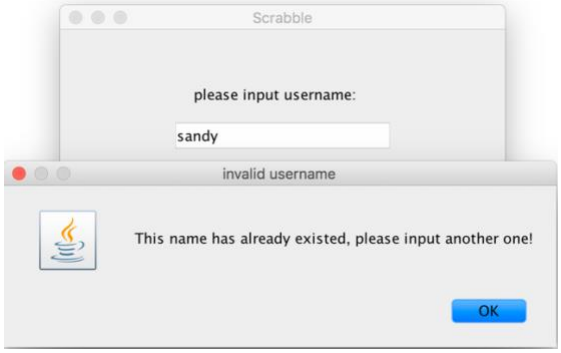


Figure 1. handle repeated username

b. Invite other users to play together

when logging successfully, the invite player page is shown. If the user wants to play with other users, the user can click the username from left user pool list, the username chosen will be shown in right list. If you want to choose yourself, an error warning will appear.

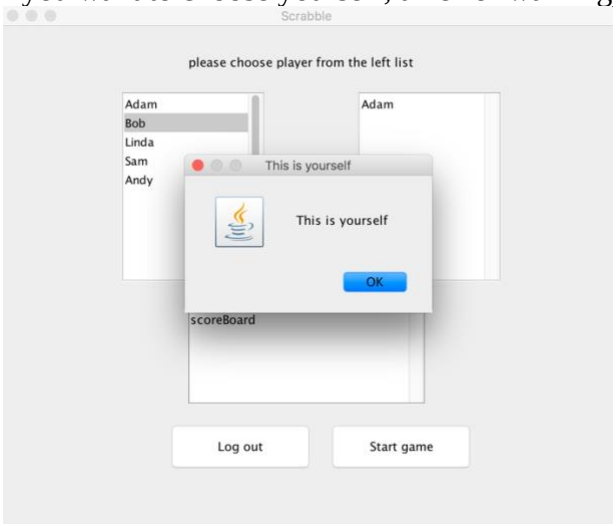


Figure 2. handle choose self

In our game design, the maximum number of users in one game is four. Hence if the user invites more than three other users, an error warning will be shown.

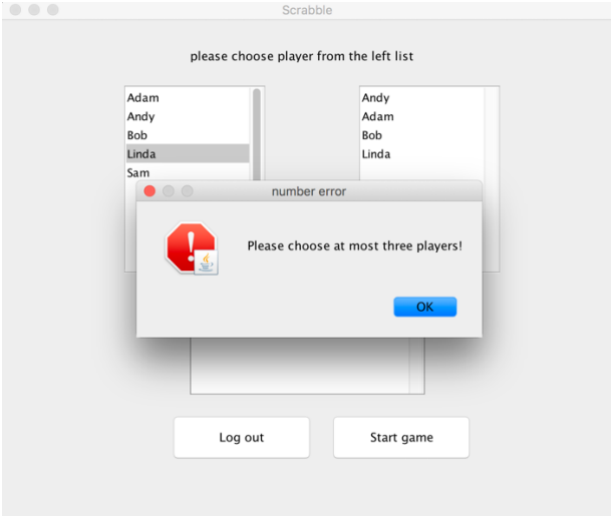


Figure 3. handle max user number

c. game page on your turn

when the user starts a game or is invited, the scrabble game page will be shown. Users can only have actions in their own turn. Before they click the button to insert on the grid, if they don't type a char, a warning will be shown.

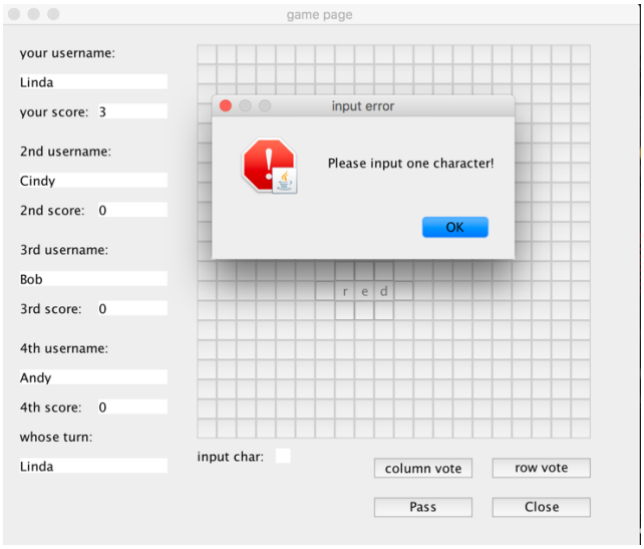


Figure 4. handle empty input

Users can only insert into the same row or same column, if they want to insert into error position, a warning will appear.

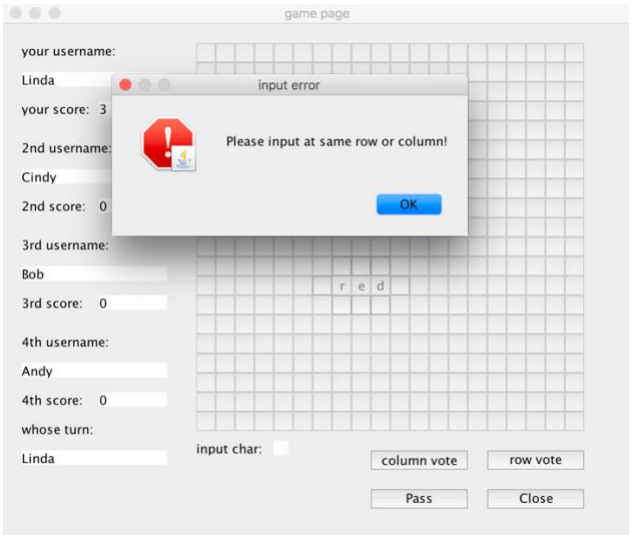


Figure 5. handle invalid input

In one turn, the column button and row button can only be clicked one time. If the user wants to click again, the application for vote will not work, an error warning appears.

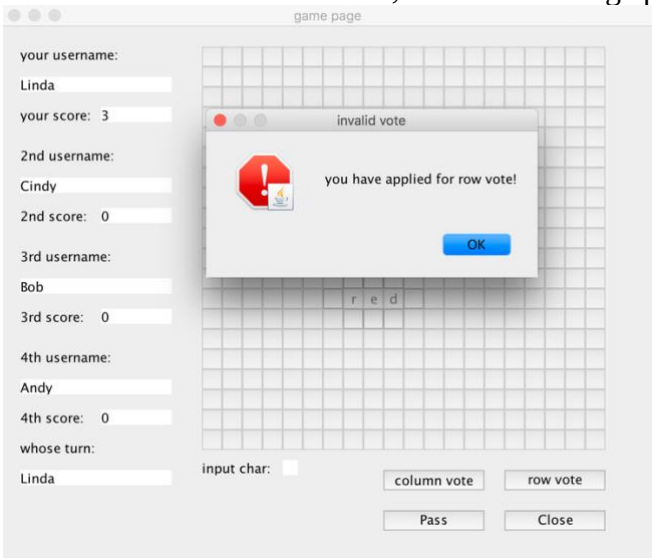


Figure 6. handle repeated vote

Once users have applied for vote, they are not allowed to insert any chars. A warning will appear for this.

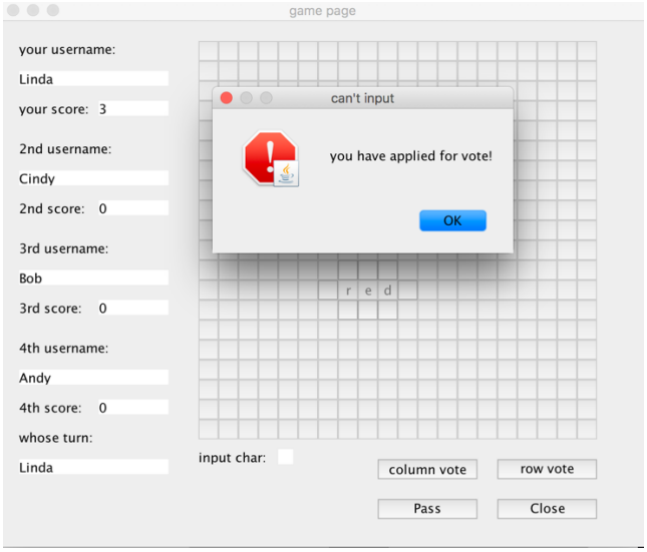


Figure 7. handle invalid input after applying for vote

d. game page not on your turn

if current turn is not your turn, any actions (insert, apply for vote, pass) except close are allowed. If you have invalid actions, an error message will appear.

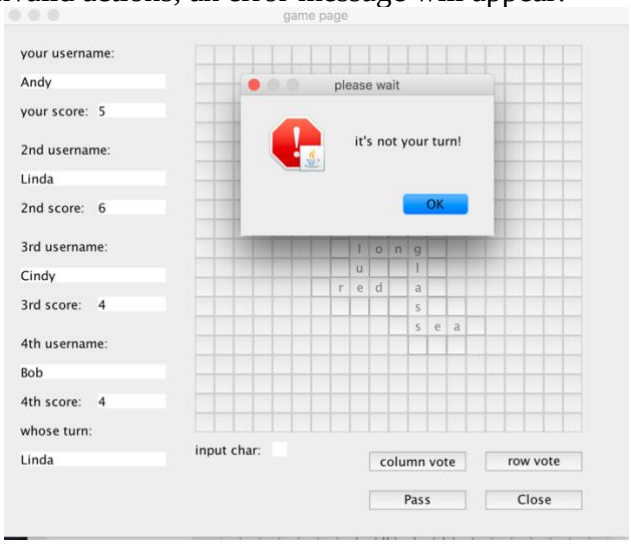


Figure 8. handle invalid actions when not your turn

e. invalid input (number, data type)

f. connection fail

5.2 server exception handling

a. invalid input (number, data type)
b. invalid port when building
c. one game is playing

6. Distribution section

In the beginning, we determined the GUI structure and layout used in this game and the architecture of the distributed server-client model through group meetings. In the first phase of development, our goal was to achieve a successful connection between the server and the client and achieve a single-version game with the reasonable logic, then the client can be connected to the server for single player games. After the message protocol was designed and identified by group, He and Wang were responsible for the development of the server, Shi and Chen were responsible for the development of the client and game logic.

In the second phase of development, through the group meetings, we decided to use multi-threading to achieve communication between the server and clients, so that the client can send an application to the server while listening to the server message. He and Wang were responsible for the game control and broadcast functions on the server side. Shi and Chen were responsible for the message listening processing and message sending functions on the client side.

Finally, through the group meetings, we perfected the logic of the game together, for example, a letter can only be inserted around the letters on the grid, a letter can only be inserted in the same row or the same column on one turn, etc., therefore the distributed game was more real and friendly. The exceptions were handled well during the testing phase.

We had an equal contribution to this project and actively worked together on design, development and test.