Group 13

Mohammed Khan – mkhan263

Qasim Mir – qmir2

Shalin Patel – spate423

Pat Kolakowski - pkolak3

Min-Max code

The Min/Max code was used to make up our medium and expert levels for tic-tac-toe. So what we did was get the state list of the moves by using the findmoves() method and if it ended up returning a 10 we would then go in an if statement to check out that the win condition was met. Otherwise if you found a state that ended giving you a 0 (i.e ending game at worst a tie) we would grab that state list and put it in an ArrayList alongside the other possible states which generate a 0. Then a random generator was used to pick out a move from those 0 states and this process was repeated for the second move taken by server in tic-tac-toe. This was the case for expert mode, in medium we ended up deciding to use min-max for the first two moves taken by the server and afterwards it would randomly place an X on a tile, so it wasn't too challenging or felt too easygoing. No other changes were made to the min max code that the professor provided.

Server and Client Logic

The server and client logic in terms of sending information back and forth was similar to project 4. But the main difference between project 4 and project 5 was obviously the different game but also the clients plays against the server rather than two clients playing. Our approach to designing and implementing the logic was straightforward. What we first did was created the gui for both the server and client with all the necessary information and TicTacToe board. Which then led us to send information to the server once the client has selected one of the boxes on the board. The server takes the arraylist of string and uses the minmax algorithm depending on the difficulty chosen and updates the client with the new arraylist of strings. This arraylist of strings is located in the gameinfo. So, once the server chose the move, the gui prints that move on the board which is done in the platform runlater. And this way it goes back and forth between the client and server choosing and displaying their move. Once the game has come to an end with client winning or server winning or tie then it would disable all the buttons on the board and display who won on the gui. This logic between server and client was instant so once client chooses a move the server then chooses and gui updates it. One of the main algorithms that was implemented was for the level of difficulty. It has been discussed above in the minmax code description but the overall idea behind that was simple. The easy difficulty was pure random, the server chose a tile that had a "b" and updated client with that. Then the expert difficulty was completely depending on the minmax algorithm, the server chose the best move to make sure that the client can't win. Finally, the medium difficulty was a mixture of both easy and expert, we made the first two moves play in expert mode and then the other moves were done randomly. This was the basic overview of the server and client logic that we implemented in this project.

Project Contribution

Mohammed Khan: Contributed towards implementing the array list functionalities. Properly adding the client in the array lists and game info passing through the array lists were added. Ensuring that each game had their gameinfo objects separate which helped avoid randomly assigning the gameinfo objects to clients in different games. If a game is decided, then also removing that game from the games arraylist and updating the client. Provided insight with logic and overall implementation of the project by working on code dealing with the client sending information to the server and vice versa. Dealt with scenarios where the tic-tac-toe would be full and the design choices that came along with it. That is scenarios where the board is full, and no player has won, or player wins/loses on last move. Responsible for debugging and correcting mistakes in the GUI/overall implementation. Completed 1/3 of the documentation.

Qasim Mir: Contributed towards the GUI for server and clients. Was responsible for the wireframe and the implementation of it for the project. That is creating the layout for the tic-tac-toe board and creating it, so it gives X and O's once clicked. Was also responsible for coming up with algorithms for how to implement easy, medium and expert mode. That is how to go about approaching how to use Min/Max code. Gave input on creating an arraylist to store the state lists that generate 0 so you can use a random generator to pick out the next move for the server. For medium mode coming with possible solutions in such that you use the first two moves to act as expert and the rest as easy mode, so the level is fair difficulty. Completed 1/3 of the documentation.

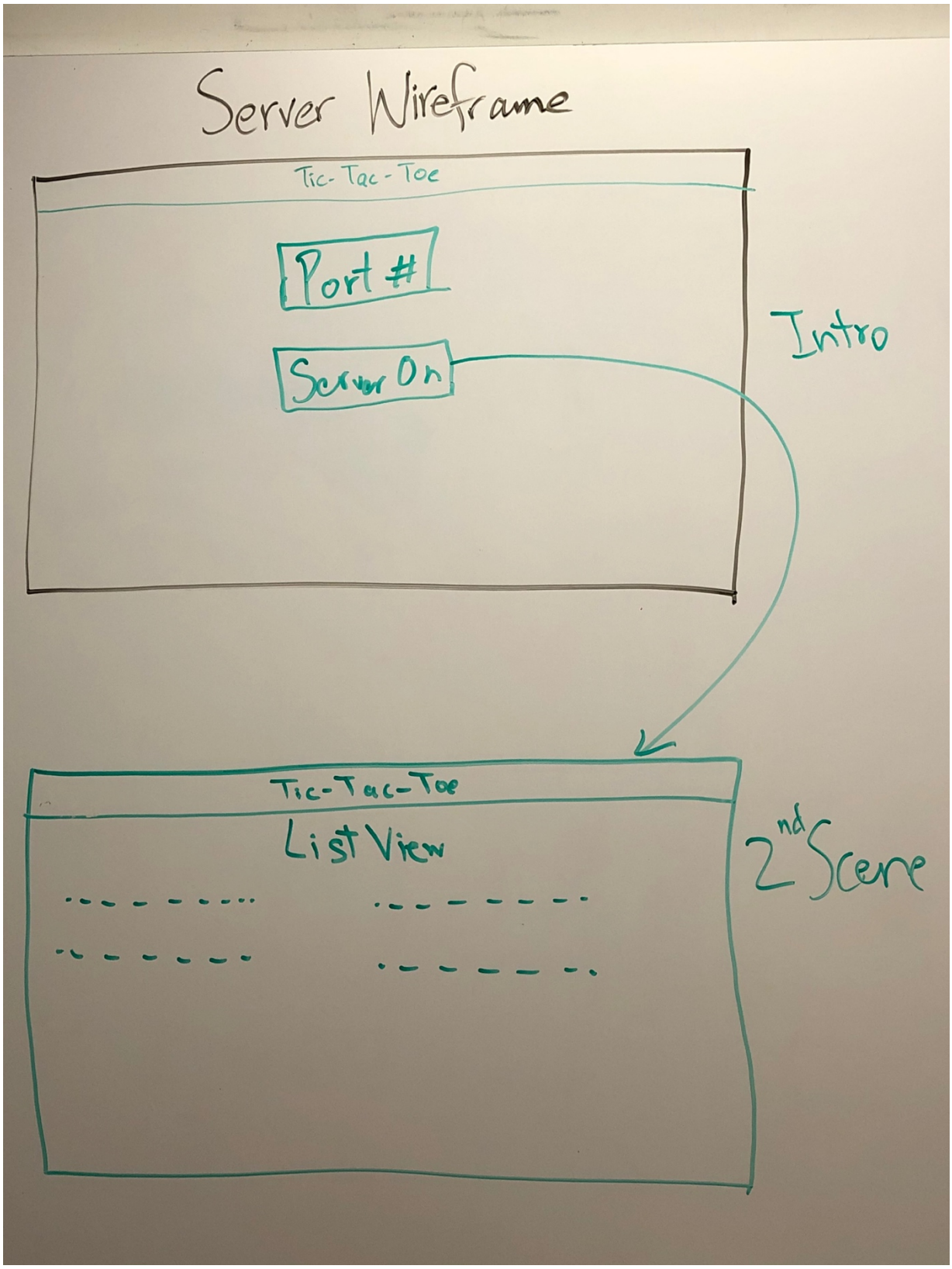
Shalin Patel: Contributed towards the keeping track of the leaderboard and having it updated smoothly. Implemented the idea of a client entering a username and sending the information to the server so it can keep track of each individual client and its score so it can then send it back to all clients. Implemented the refresh button as well so a client can keep track of the top 3 leaders in a lobby accurately. Responsible for sorting the arraylist as well so the ranking comes out in descending order. Implemented the whoWon function as well. This way it was easier to have the user's scores updated accurately by incrementing when player won. Kept track of how many games were played for a given client as well. Also responsible for coming up with the required testcases for Min/Max algorithm. Completed 1/3 of the documentation.

Pat Kolakowski: Did not contribute.

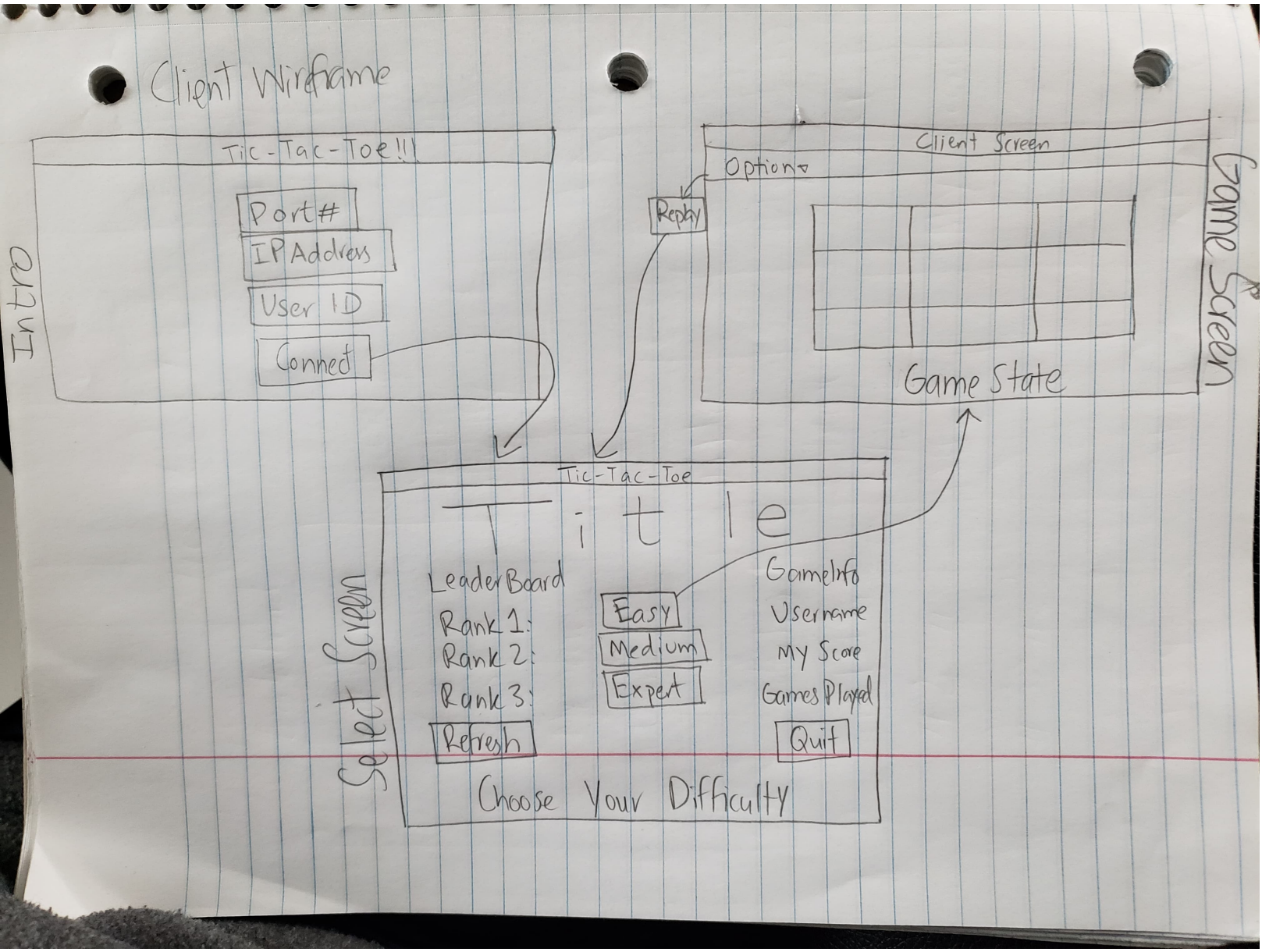
Changes in the diagrams and wireframe from part 1

- No change in the server wireframe. And for client wireframe there were minor changes such as adding a refresh button for leaderboard and textfield for client username.
- No major change in the server and client activity diagram apart from making sure that the client doesn't enter wrong port and ip address, if wrong port or ip address then it doesn't let you connect. And also, the refresh button for the leaderboard was added which can be pressed in the second scene and that refreshes the scoreboard.
- Main change in both the server and client UML class was adding another class called ScoreNode which was used by gameinfo and TicTacToe class. Also added different variables and methods inside the gameinfo class.
- The updated and final versions of the diagrams/wireframes are shown below.

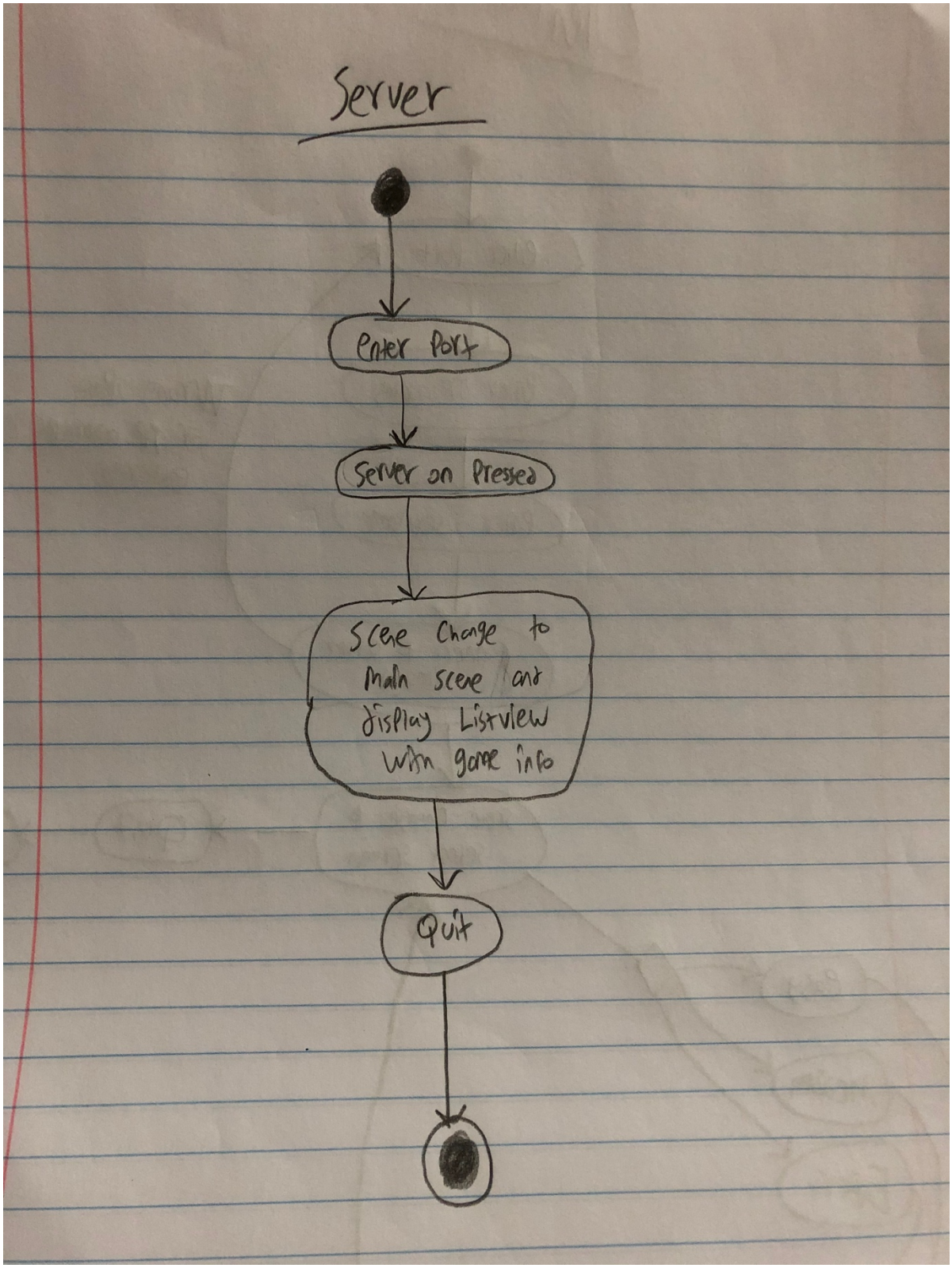
Final Version of Server Wireframe



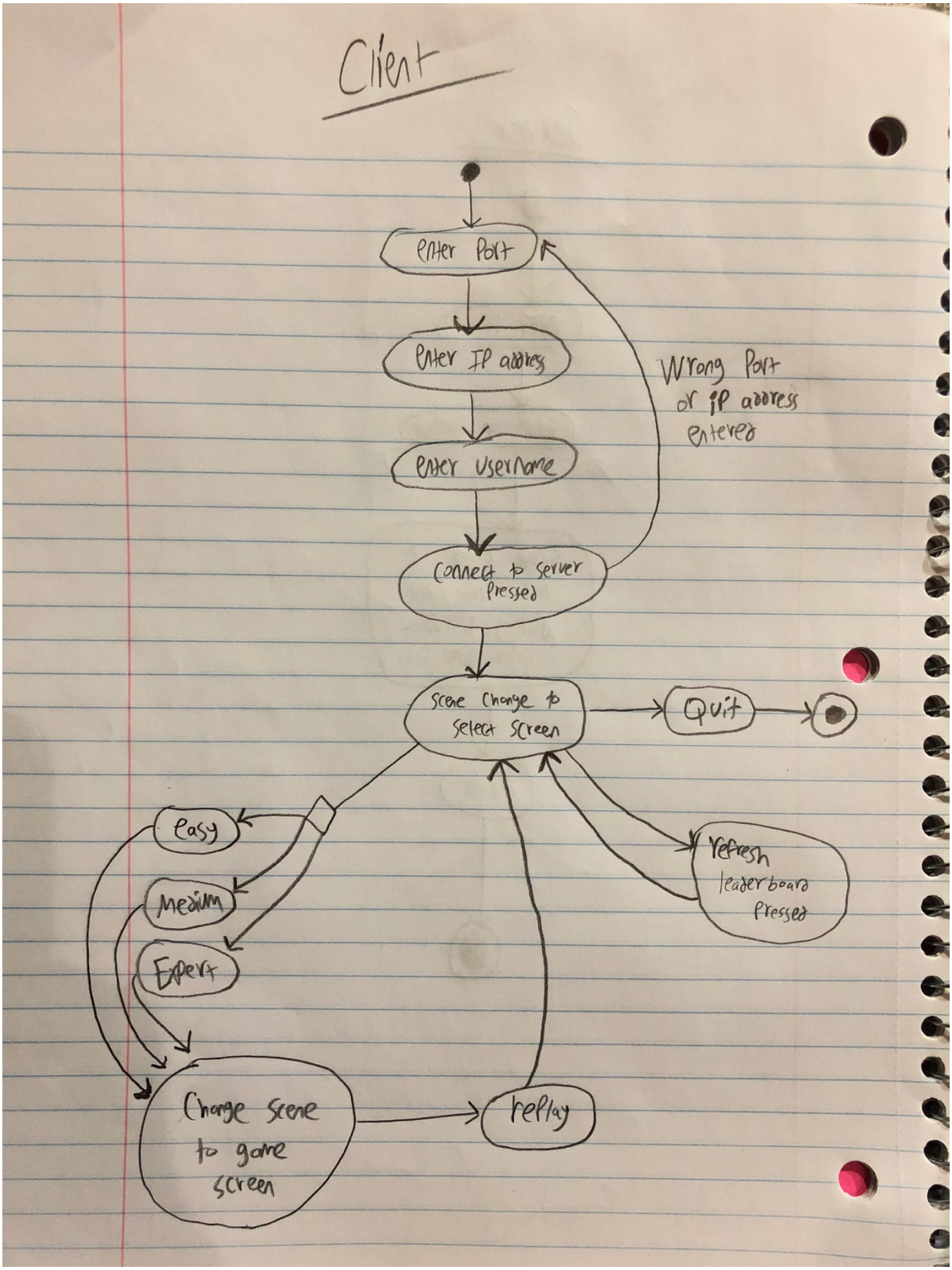
Final Version of Client Wireframe



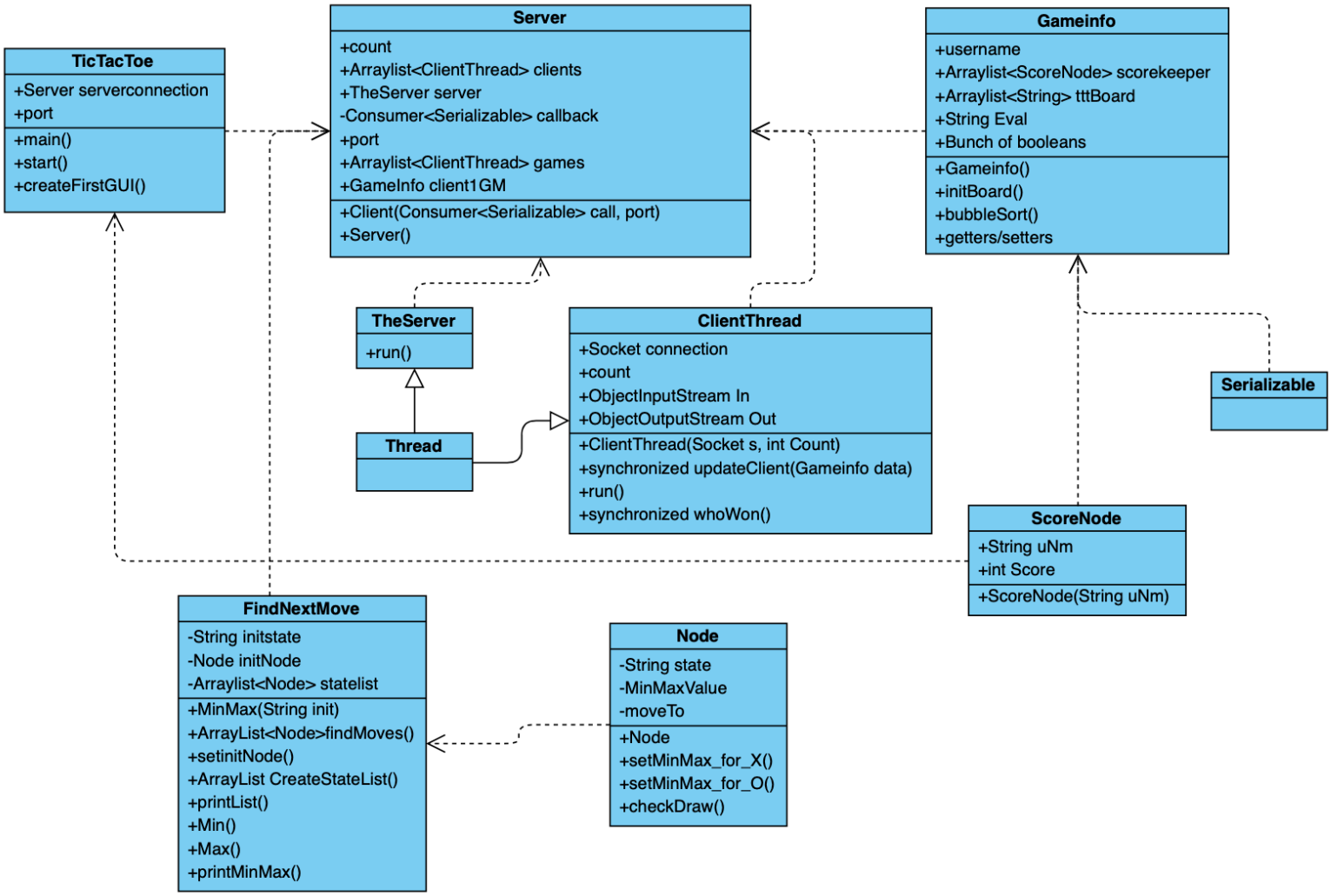
Final Version of Server Activity Diagram



Final Version of Client Activity Diagram



Final Version of Server UML Class



Final Version of Client UML Class