Team 17:

Angel Naranjo: anaran7

Eivydas Raulynaitis: erauly2

Luis Villalobos: Lvilla48

Jim Li: jli263

1) Discussing how you incorporated the Min-Max code into your overall design including any changes made to the original code.

The min max code was incorporated by having FindNextMove class that extended thread and its run method will run the MINMAX algorithm each time the clientthread does the run method. It is set up so that FindNextMove class has a string array that store the current board and an int that stores the difficulty. These values are changed in the server when a client makes a move and right before the run method is called

We incorporated different difficulties by allowing the player to select a difficulty. Based on that difficulty the FindNextMove class finds the next move for the AI by either randomly picking a spot (easy), picking the MINMAX spot (hard), or randomly picking between the two (medium).

2) Give a general description of the server and client logic. This description should highlight and discuss algorithms and design choices made.

The client starts of by clicking on a difficulty. This is saved in a string. The client then clicks on the tic tac toe board which updates its board and sends a gameinfo object containing the client's current board, the client's id, and the difficulty he is playing in. The server who receives the board runs the MINMAX algorithm to get a move list which it then defaults to picking the recommendation. Then the server checks the difficulty if it is in hard it does nothing, if it is medium it randomly decides with 50% chance of picking the recommended move or a random move. If it is easy it randomly decides with 25% chance of picking the recommended move or a random move. Then it checks if the game has ended and sets the appropriate variable in the gameinfo object and sends it back to the client. The client receives it and checks if the game ended if it has it goes to the select difficulty screen. If the game did not end it picks its next move and the cycle continues until the game ends.

3) *A discussion of changes in the diagrams and wireframes from the ones you made*

Added the server activity diagram. The other change that was made was adding a menu bar that had a button you click on the view the leaderboards. The UML for the server and diagrams were changed to include the updated GameInfo with the new variables

4) Description of what each teammate contributed to this project

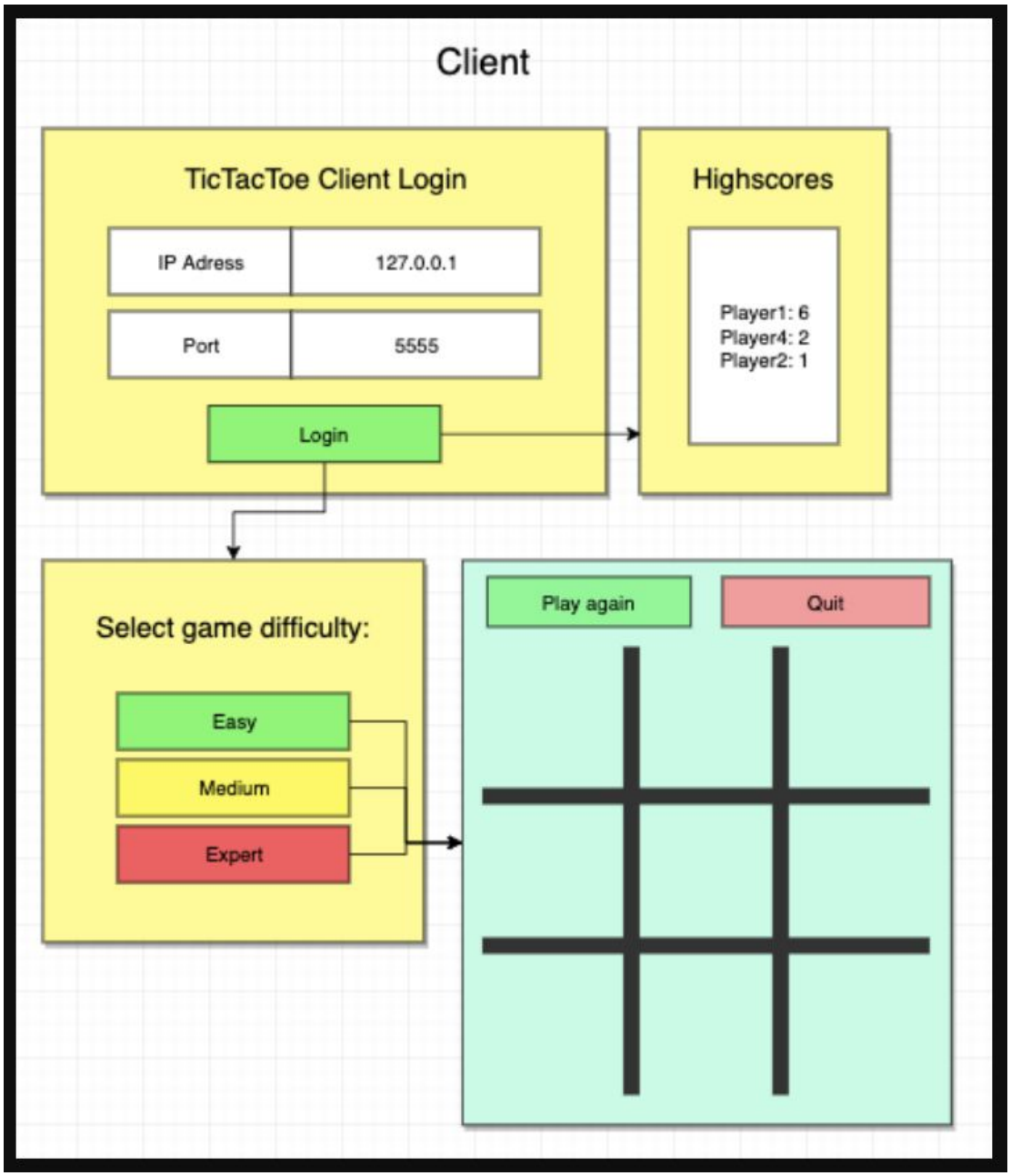
Angel - Added the MinMax algorithm in its own thread. Added the ability to play against the server 1 player at a time. Worked on having multiple players play against the server at the same time. Created the UML for client and server code. Worked on the final documentation for the project.

Jim - Created the client GUI along with event handlers, allowed for players to play again and quit by re-enabling the buttons, worked on game difficulty, worked on player vs server, and worked on the listViews. Created the UML client class diagram.

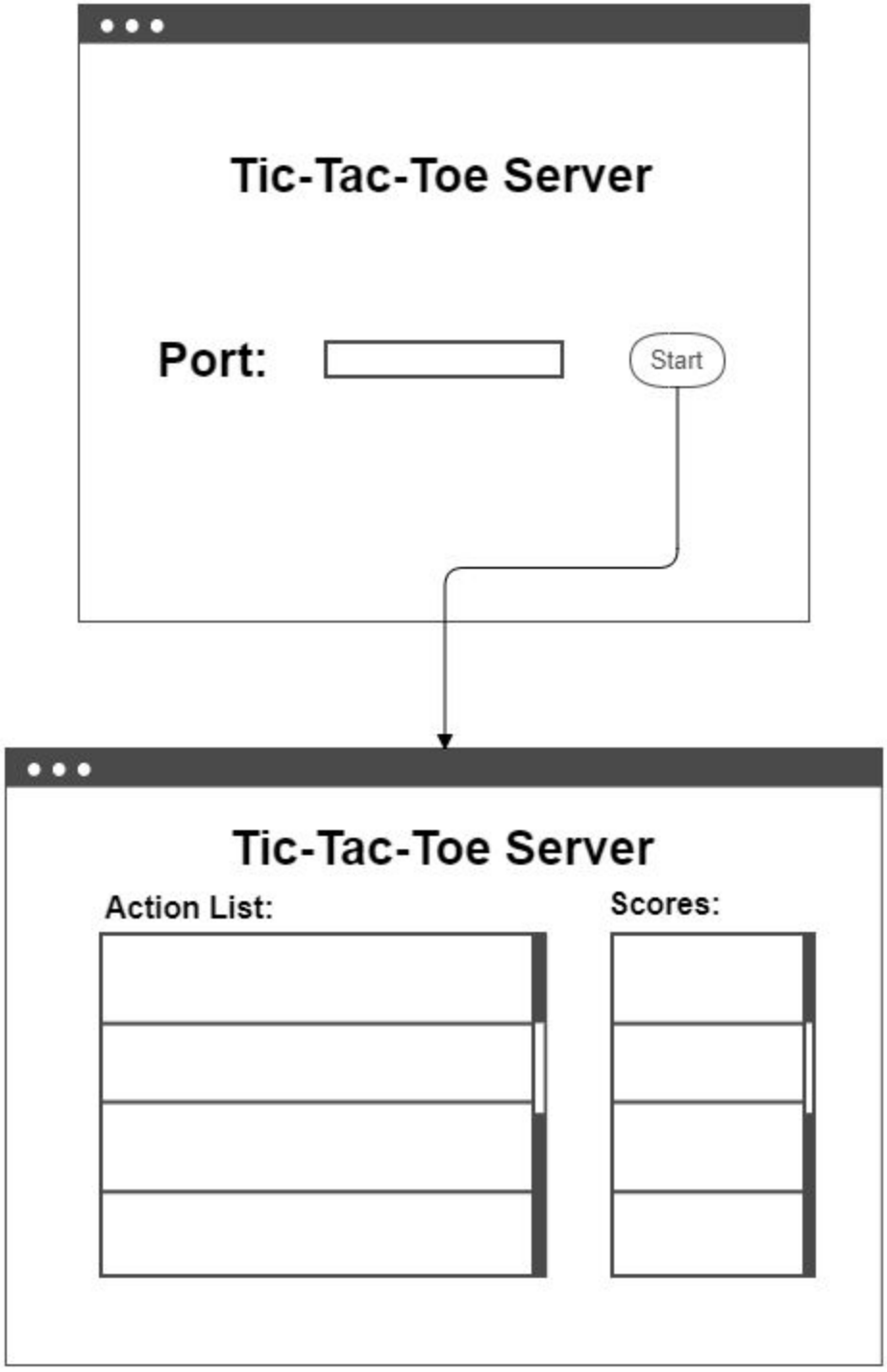
Luis - Created the second scene for the Server GUI. Allowed for the server list view to update and display whatever actions were being done. The same thing was done for the score list view on the server side. Did the server wireframe.

Eivydas - Created port screen Server GUI, polished up overall server GUI w/ consistent design. Implemented difficulties selection between server and client and created easy and medium difficulties for MINIMAX algorithm. Implemented leaderboard logic for up to 3 players.

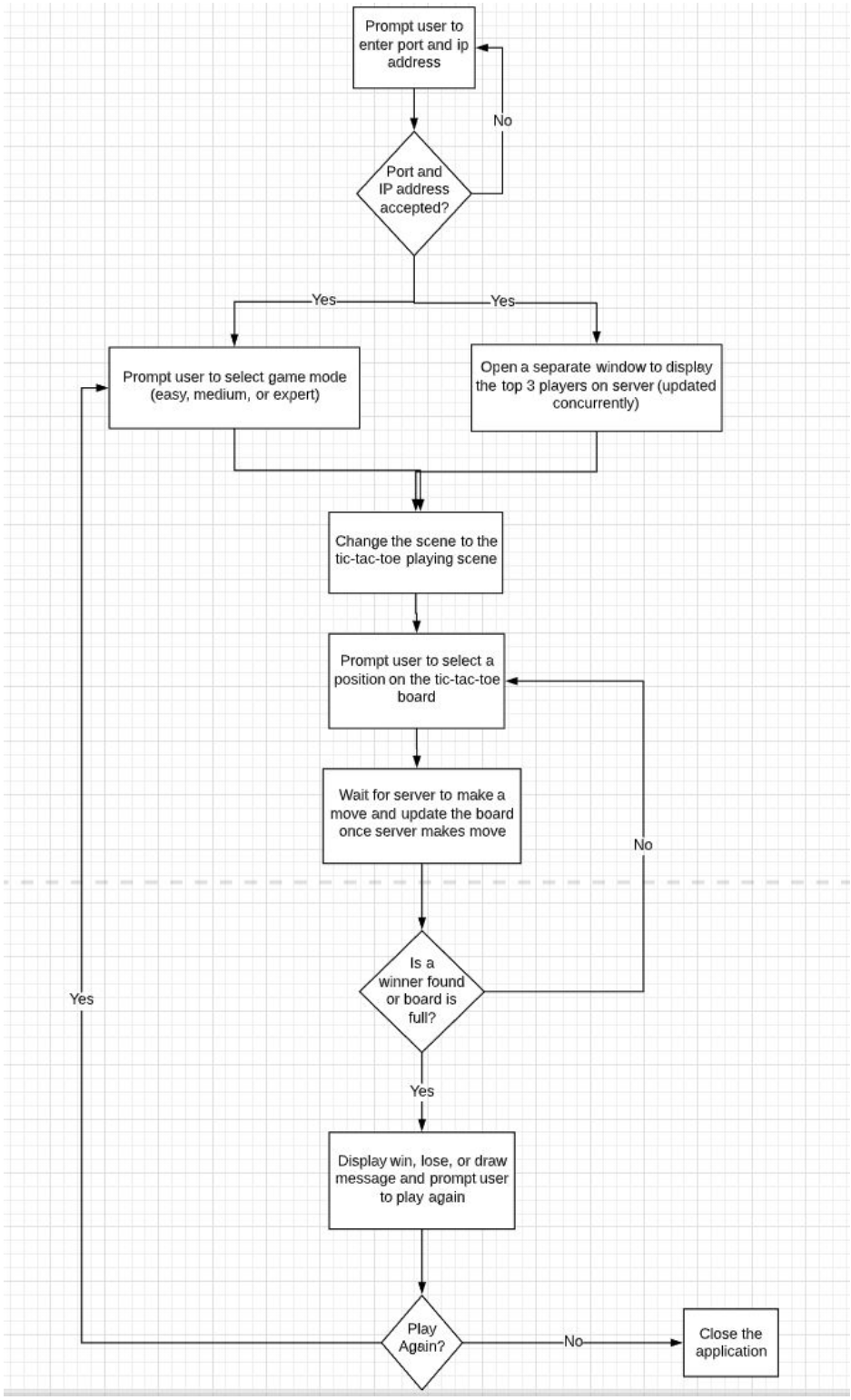
Wireframe(Client):



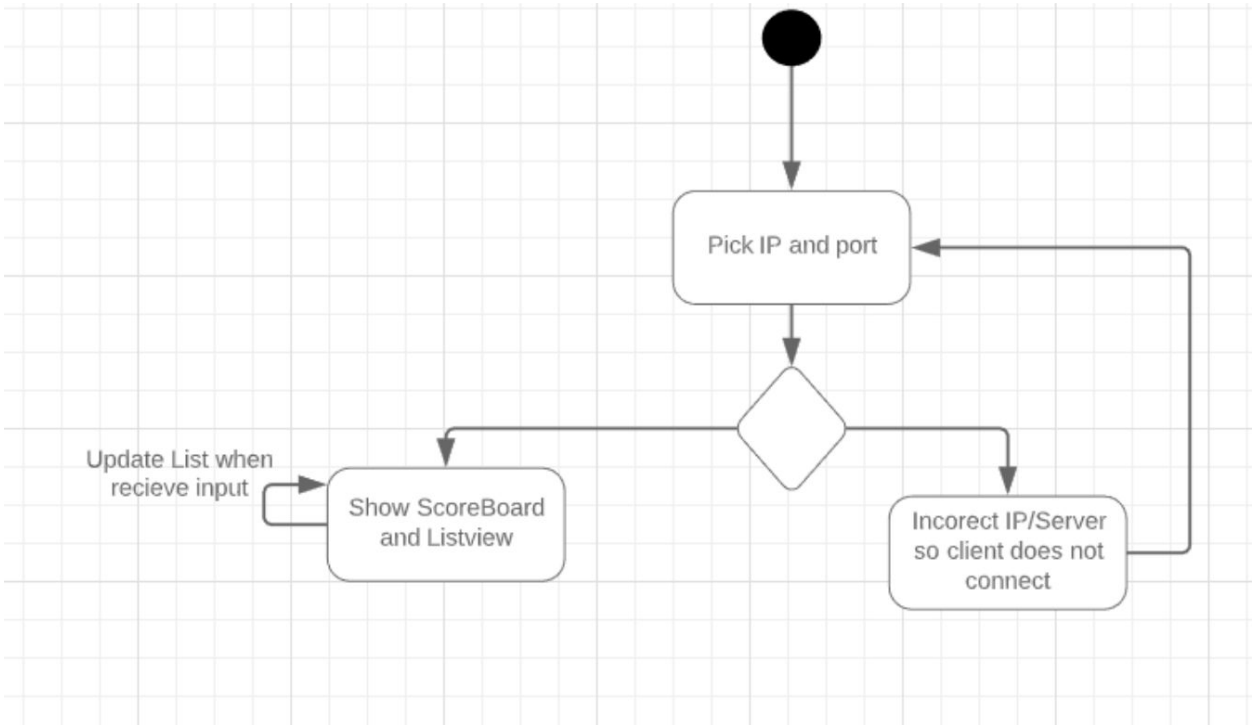
Wireframe(Server):



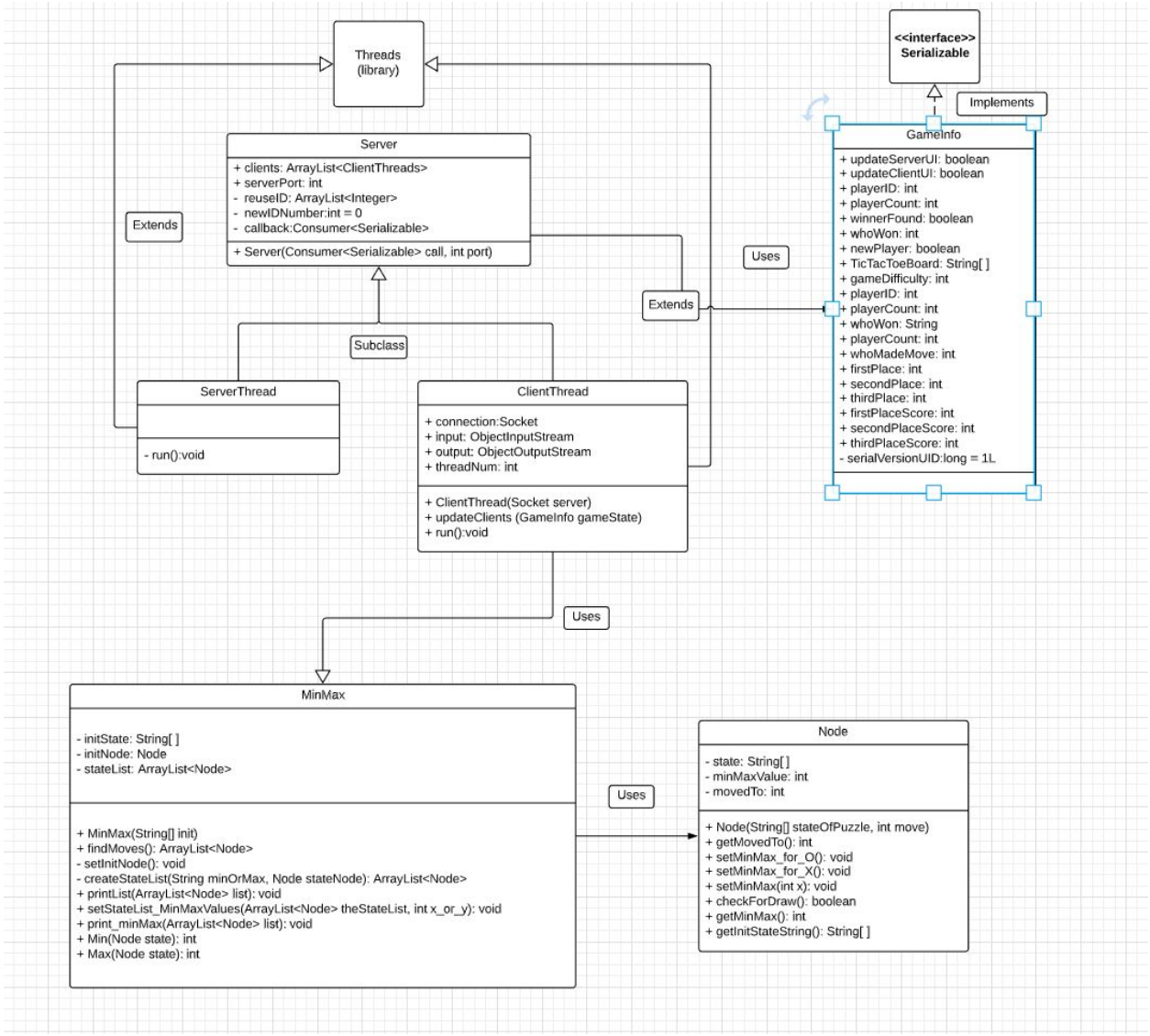
Activity Diagram(Client):



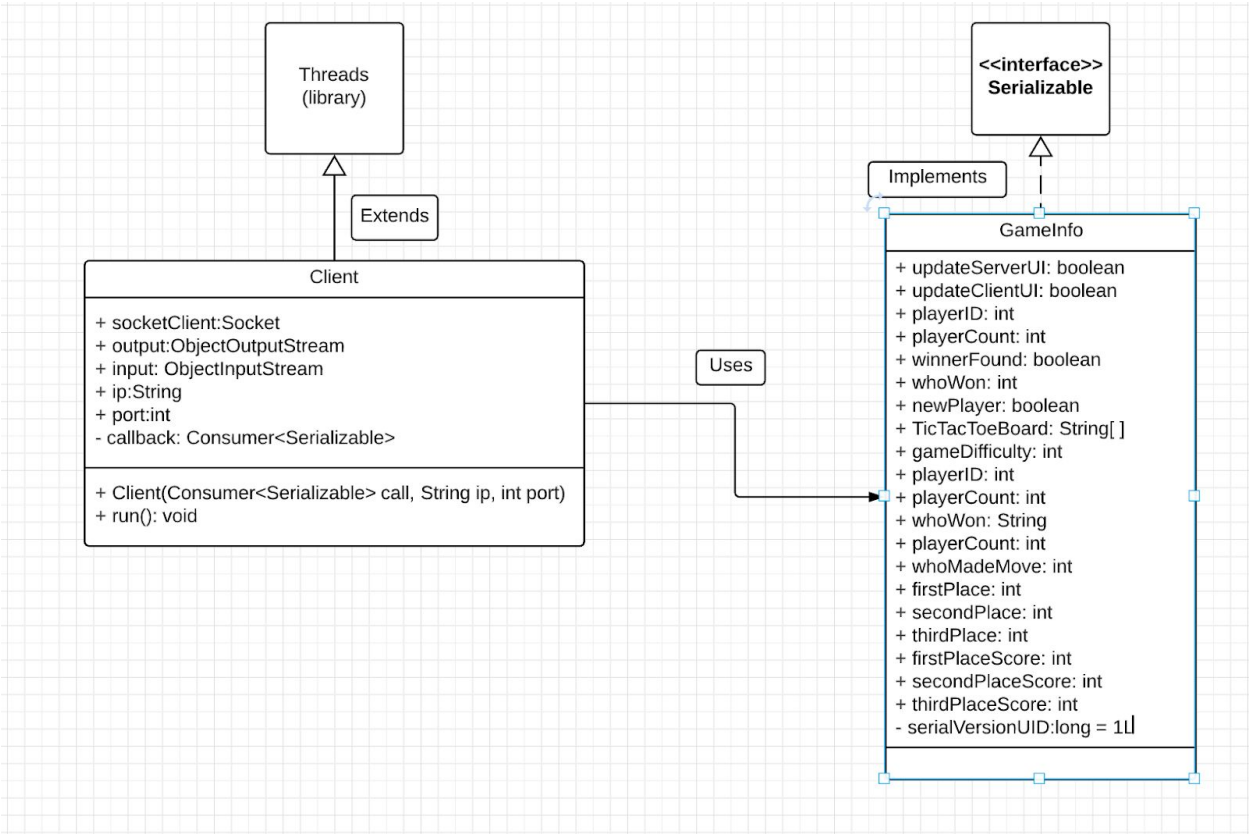
Activity Diagram(Server):



UML(SERVER):



UML(Client):


Threads
(library)
Extends
Client
+ socketClient:Socket
+ output:ObjectOutputStream
+ input: ObjectInputStream
+ ip:String
+ port:int
- callback: Consumer<Serializable>
+ Client(Consumer<Serializable> call, String ip, int port)
+ run(): void
Uses
<<interface>>
Serializable
Implements
GameInfo
+ updateServerUI: boolean
+ updateClientUI: boolean
+ playerID: int
+ playerCount: int
+ winnerFound: boolean
+ whoWon: int
+ newPlayer: boolean
+ TicTacToeBoard: String[ ]
+ gameDifficulty: int
+ playerID: int
+ playerCount: int
+ whoWon: String
+ playerCount: int
+ whoMadeMove: int
+ firstPlace: int
+ secondPlace: int
+ thirdPlace: int
+ firstPlaceScore: int
+ secondPlaceScore: int
+ thirdPlaceScore: int
- serialVersionUID:long = 1L