TICTACTOE WITH MIN-MAX

Project 5

Team 25

Ngo, Amy Ngo, William Ocampo, Kent Paglomotan
ango8@uic.edu, wocampo2@uic.edu, kpaglo2@uic.edu

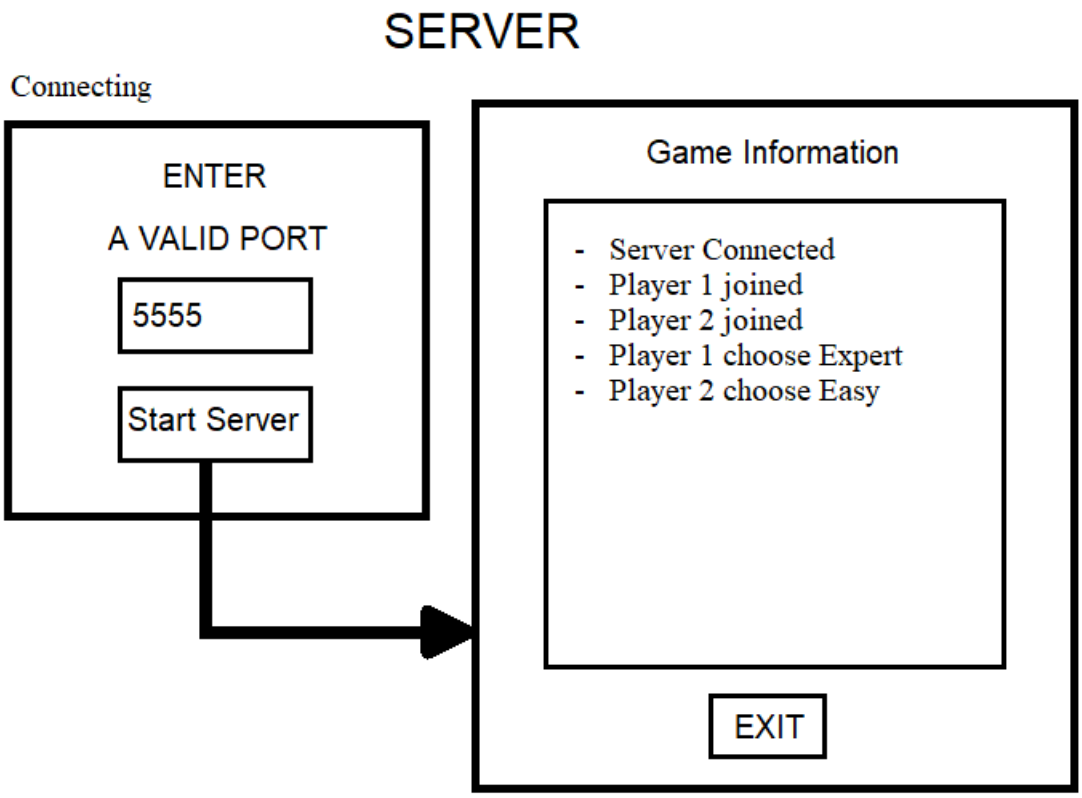
Min – Max Code

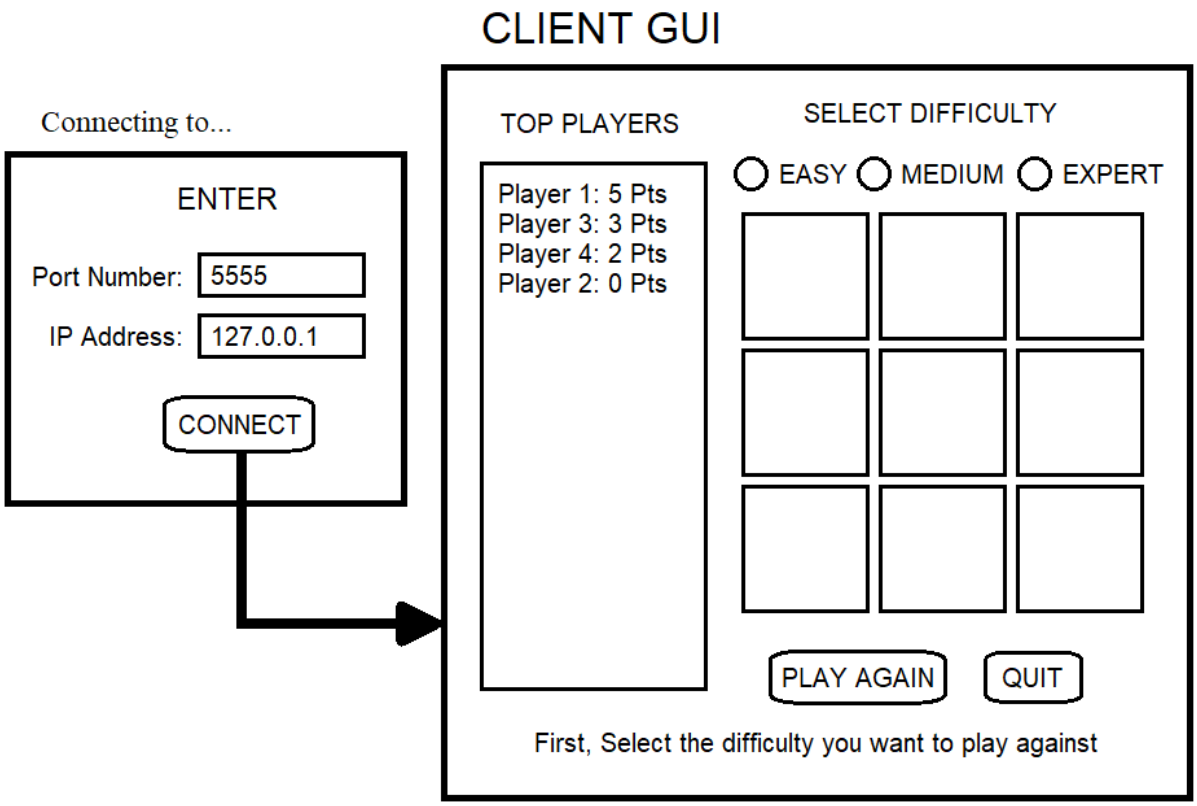
In the AI_MinMax code, an array list of nodes is public so that the Server class could access the code. The constructor in AI_MinMax has a parameter: string puzzle. The same parameter is used in the getBoard method. Many lines of code were commented out, such as the getBoard function. Since the input isn't used, the command line won't be used.

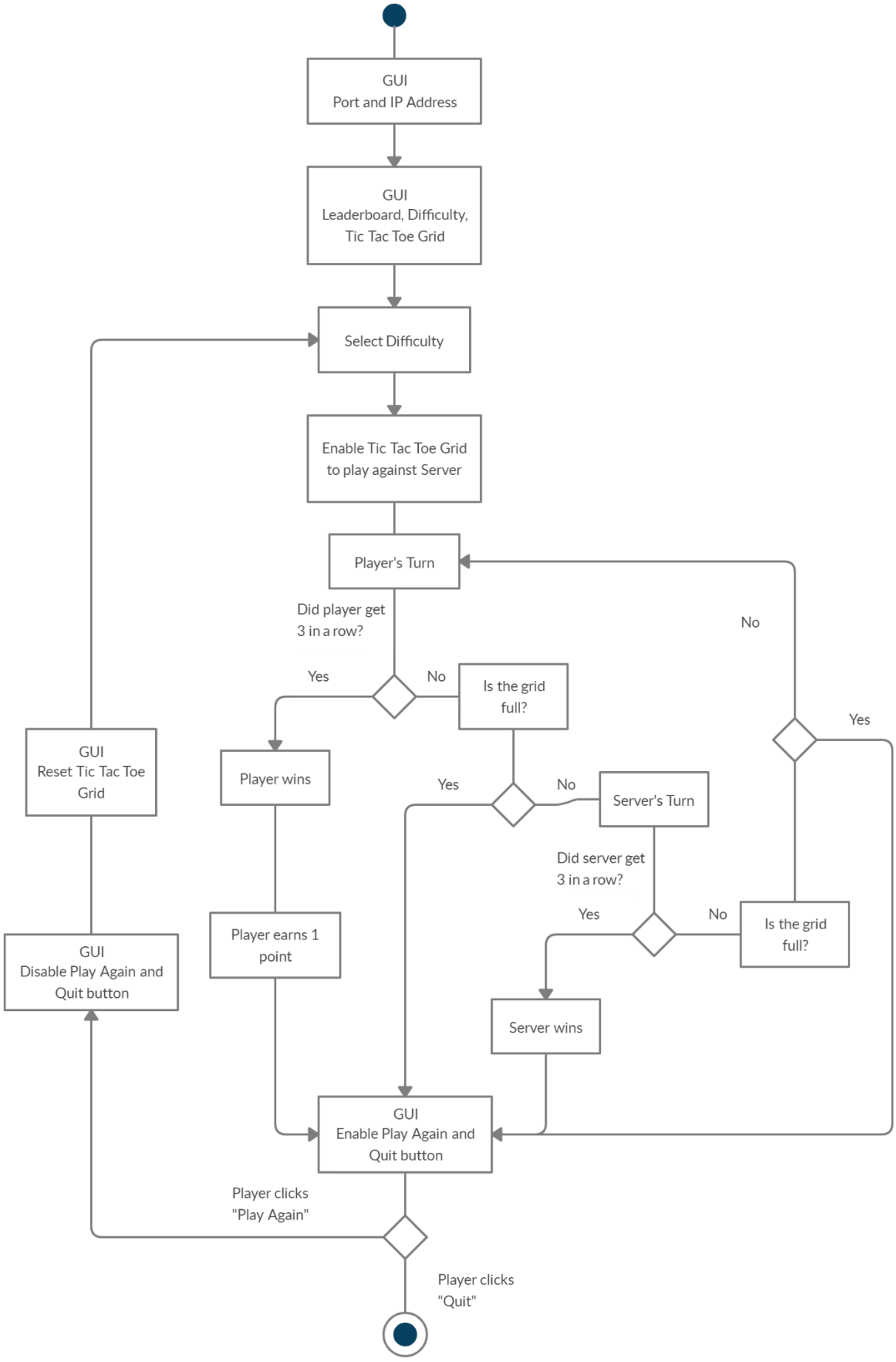
Server and Client Logic

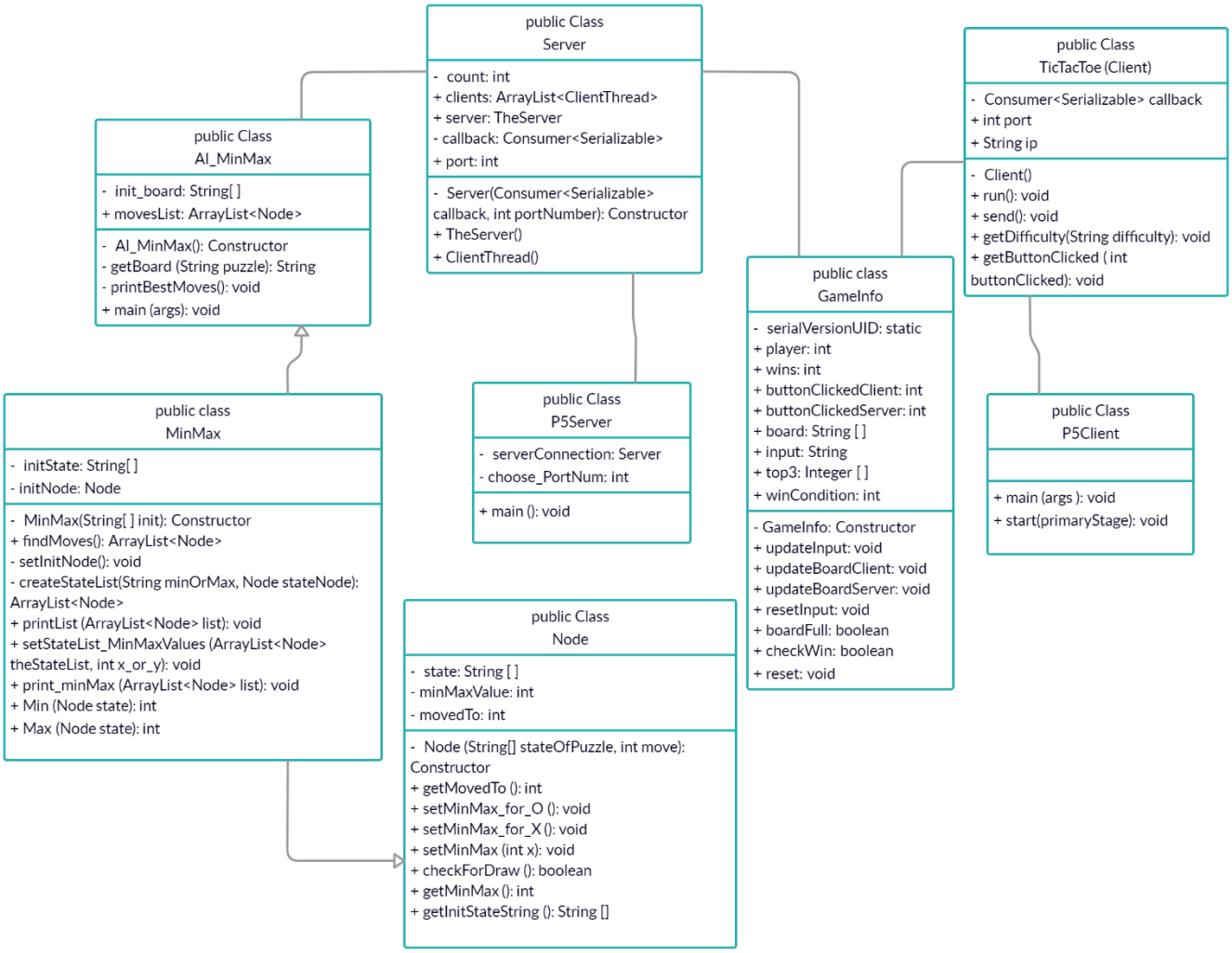
The FindNextMove class was created inside the CliendThread of the Server MinMax, AI_MinMax, and Node. The FindNextMove method is responsible for sending a string of size 9 in the format "b b b b b b b b b". This reads the Tic Tac Toe board from left to right and top to bottom. The string "b" represents an empty slot, "X" represents server's move, and "O" represents the client's move. The input string is sent to the AI_MinMax method and returns an array of integers. For example, if the array of integers is [1, 2] then either 1 or 2 is a good move. The client can choose any other available button on its own turn.

Final Activity and Class Diagrams and Wireframe




CLIENT GUI
Connecting to...
ENTER
Port Number:
5555
IP Address:
127.0.0.1
CONNECT
TOP PLAYERS
Player 1: 5 Pts
Player 3: 3 Pts
Player 4: 2 Pts
Player 2: 0 Pts
SELECT DIFFICULTY
EASY
MEDIUM
EXPERT
PLAY AGAIN
QUIT
First, Select the difficulty you want to play against


GUI
Port and IP Address
GUI
Leaderboard, Difficulty,
Tic Tac Toe Grid
Select Difficulty
Enable Tic Tac Toe Grid
to play against Server
Player's Turn
Did player get
3 in a row?
Yes
No
Is the grid
full?
Player wins
Yes
No
Server's Turn
Did server get
3 in a row?
Yes
No
Is the grid
full?
Server wins
No
Yes
GUI
Reset Tic Tac Toe
Grid
Player earns 1
point
GUI
Disable Play Again and
Quit button
GUI
Enable Play Again and
Quit button
Player clicks
"Play Again"
Player clicks
"Quit"


public Class
Server
- count: int
+ clients: ArrayList<ClientThread>
+ server: TheServer
- callback: Consumer<Serializable>
+ port: int
- Server(Consumer<Serializable> callback, int portNumber): Constructor
+ TheServer()
+ ClientThread()
public Class
TicTacToe (Client)
- Consumer<Serializable> callback
+ int port
+ String ip
- Client()
+ run(): void
+ send(): void
+ getDifficulty(String difficulty): void
+ getButtonClicked (int buttonClicked): void
public Class
AI_MinMax
- init_board: String[ ]
+ movesList: ArrayList<Node>
- AI_MinMax(): Constructor
- getBoard (String puzzle): String
- printBestMoves(): void
+ main (args): void
public class
GameInfo
- serialVersionUID: static
+ player: int
+ wins: int
+ buttonClickedClient: int
+ buttonClickedServer: int
+ board: String [ ]
+ input: String
+ top3: Integer [ ]
+ winCondition: int
- GameInfo: Constructor
+ updateInput: void
+ updateBoardClient: void
+ updateBoardServer: void
+ resetInput: void
+ boardFull: boolean
+ checkWin: boolean
+ reset: void
public Class
P5Server
- serverConnection: Server
- choose_PortNum: int
+ main (): void
public Class
P5Client
+ main (args): void
+ start(primaryStage): void
public class
MinMax
- initState: String[ ]
- initNode: Node
- MinMax(String[ ] init): Constructor
+ findMoves(): ArrayList<Node>
- setInitNode(): void
- createStateList(String minOrMax, Node stateNode): ArrayList<Node>
+ printList (ArrayList<Node> list): void
+ setStateList_MinMaxValues (ArrayList<Node> theStateList, int x_or_y): void
+ print_minMax (ArrayList<Node> list): void
+ Min (Node state): int
+ Max (Node state): int
public Class
Node
- state: String [ ]
- minMaxValue: int
- movedTo: int
- Node (String[] stateOfPuzzle, int move): Constructor
+ getMovedTo (): int
+ setMinMax_for_O (): void
+ setMinMax_for_X (): void
+ setMinMax (int x): void
+ checkForDraw (): boolean
+ getMinMax (): int
+ getInitStateString (): String []

Changes from Initial to Wireframe and Final Diagrams

In the final wireframe of the server GUI, the button to turn on the server was removed. Since the server connects to a port, the button is unnecessary to manually turn the server on. The server also keeps track of a specific player's choice and server's choice in the Tic Tac Toe grid. The result of the match and the player's points are recorded.

The changes in the client GUI were added in the Tic Tac Toe grid. The leaderboard and the buttons to play again and quit are added in with the grid. The player will know other players who has more points in the leaderboard while the player plays against the server. The idea of having separate windows of the leader board and the Tic Tac Toe grid is a hassle to organize on the screen. Thus, the leaderboard and the play again and quit buttons were added with the grid.

In the Activity diagram, the initial diagram had an unclear logic path or missing other possibilities in the game. The final diagram includes more conditional and action paths. It's more fluid and logical with the possibility of having a full grid that may end in a tie match.

In the UML class diagram, there are more public classes are added with changes to the attributes and methods. The AI_MinMax, MinMax and Node were created and connected with each other. Notably, the GameInfo class required more methods and attributes to keep track of the board between the client and server. Some attributes were removed in a few classes from the initial class diagram.

Teammate Contribution

Amy worked the initial and final UML activity diagram. She coded Node class in the Server files and the GameInfo for the Server and Client. She also assisted in research and problem solving. Lastly, she finalized the documentation.

William worked on the initial and final UML class diagram. He coded the AI_MinMax, MinMax, and Server classes. He also seeks for Amy's and Kent's point of view when he is in a bind.

Kent worked on the initial and final Wireframe. He created the GUI on the P5Server and P5Client with visual backgrounds. He coded the Client class. He also gives advice on changes to be added or removed.