CS342 Project 5: TicTacToe w/ Min-Max

Team 43

Wael Mobeirek - wmobei2@uic.edu

Enoc Carranza - ecarra6@uic.edu

Taha Khomusi - tkhomu2@uic.edu

Hasan Sehwail - hsehwa2@uic.edu

- **How we incorporated the Min-Max code (including any changes made to the original code).**
    - We create FindNextMove class, that runs on its own thread in Server class
    - In FindNextMove, there is an instance of AI_MinMax
    - AI_MinMax's constructor was changed, so that it takes a board as an argument instead
    - In FindNextMove, there are 4 ArrayLists in which contain all moves, winning moves, draw moves, and losing moves.
    - In FindNextMove, there is a method called getServerNextMove(), which takes a board and a difficulty level as arguments, and does the following:
        - Initializes an instance of AI_MinMax, and gets all possible moves.
        - Sorts all moves into winning, drawing and losing.
        - Depending on the difficulty level, the method decides what move to return using Math.random() and our sorted ArrayLists

- **A general description of the server and client logic.**
    - Server
        - Each client has their own Game object, which has all game information
        - Clients can ask the server to do the following:
            - Challenge, which means that they select a difficulty and start a game
            - Play again, which means that they're wish to play another game after their current game is over
            - Play a move, which means that they're currently in a game, and they wish to play. The server validates their move, adds it to the board, gets a move from AI_MinMax as discussed above, checks if the game is won, then sends all the information back to the client.

- Client
  - The backend has a send and receive functionality to/from the server
  - The client's GUI takes the game information sent by the server, and updates the GUI to the correct scene with the correct information
  - If the player is in a game, the client's GUI is also responsible for drawing the board and calling the send() function when the player presses a button in the board grid
  - The client GUI also updates top 3 high scores.

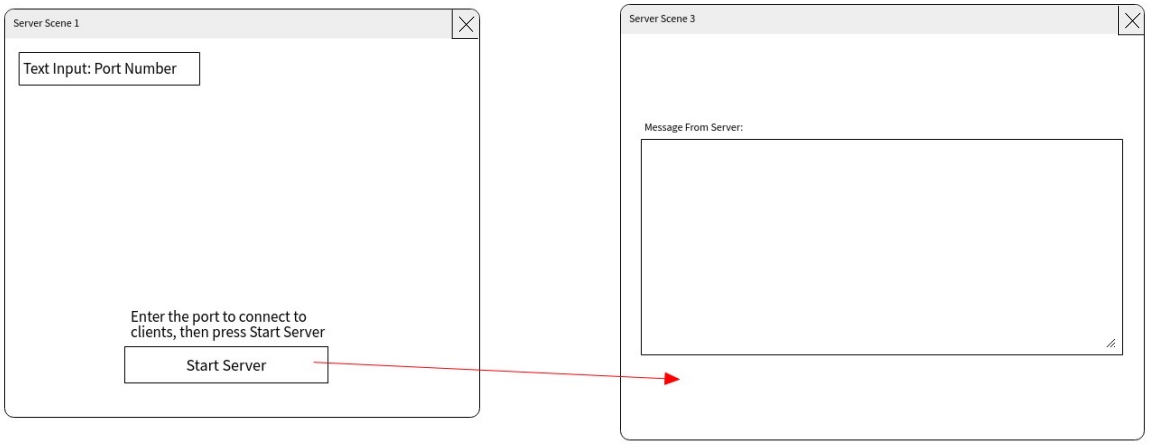
- **Description of what each teammate contributed to this project**
  - Wael Mobeirek - wmobei2@uic.edu
    - Organizer/planner
    - Server and Client Threading and logic
    - Incorporated AI_MiniMax
    - UML class diagrams
    - Updated Server Activity Diagram
    - JUnit testing
    - Collaboration Document
    - Debugging
  - Enoc Carranza - ecarra6@uic.edu
    - Wireframes
    - Client GUI design and logic
    - Debugging
  - Taha Khomusi - tkhomu2@uic.edu
    - Created base code from previous projects
    - Client GUI design and logic
    - Debugging
  - Hasan Sehwail - hsehwa2@uic.edu
    - Incorporated difficulty level into the server's next move decision

        - Implemented isGameWon(), which checks whether a game is won/over or not, and if so by whom or if its a draw.
        - Activity Diagrams
        - JUnit testing
        - Debugging

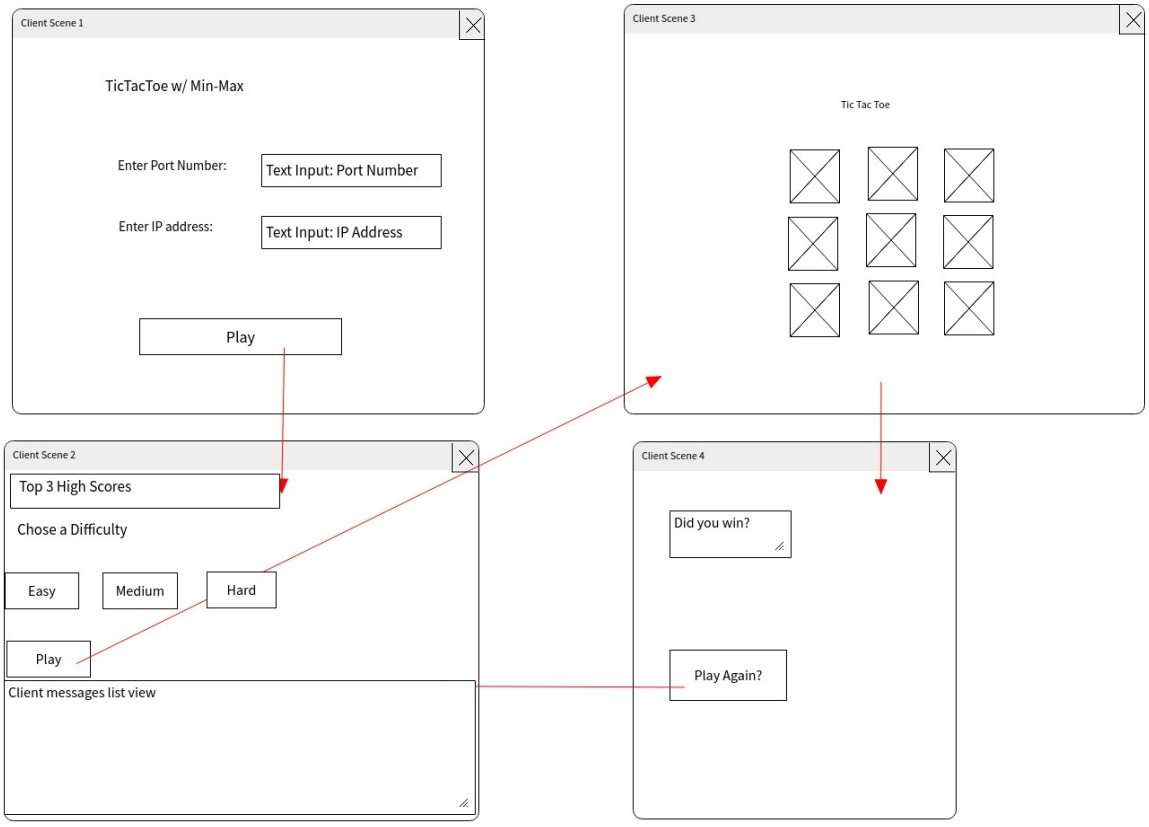
- **A discussion of changes in the diagrams.**
    - UML class diagrams
        - Added FindNextMove class
        - updated previously existing classes to the current status
    - Redid the server activity diagram to have two parallel processes with more details
    - Updated wireframes to match the current GUI

- **Final versions of diagrams: below.**

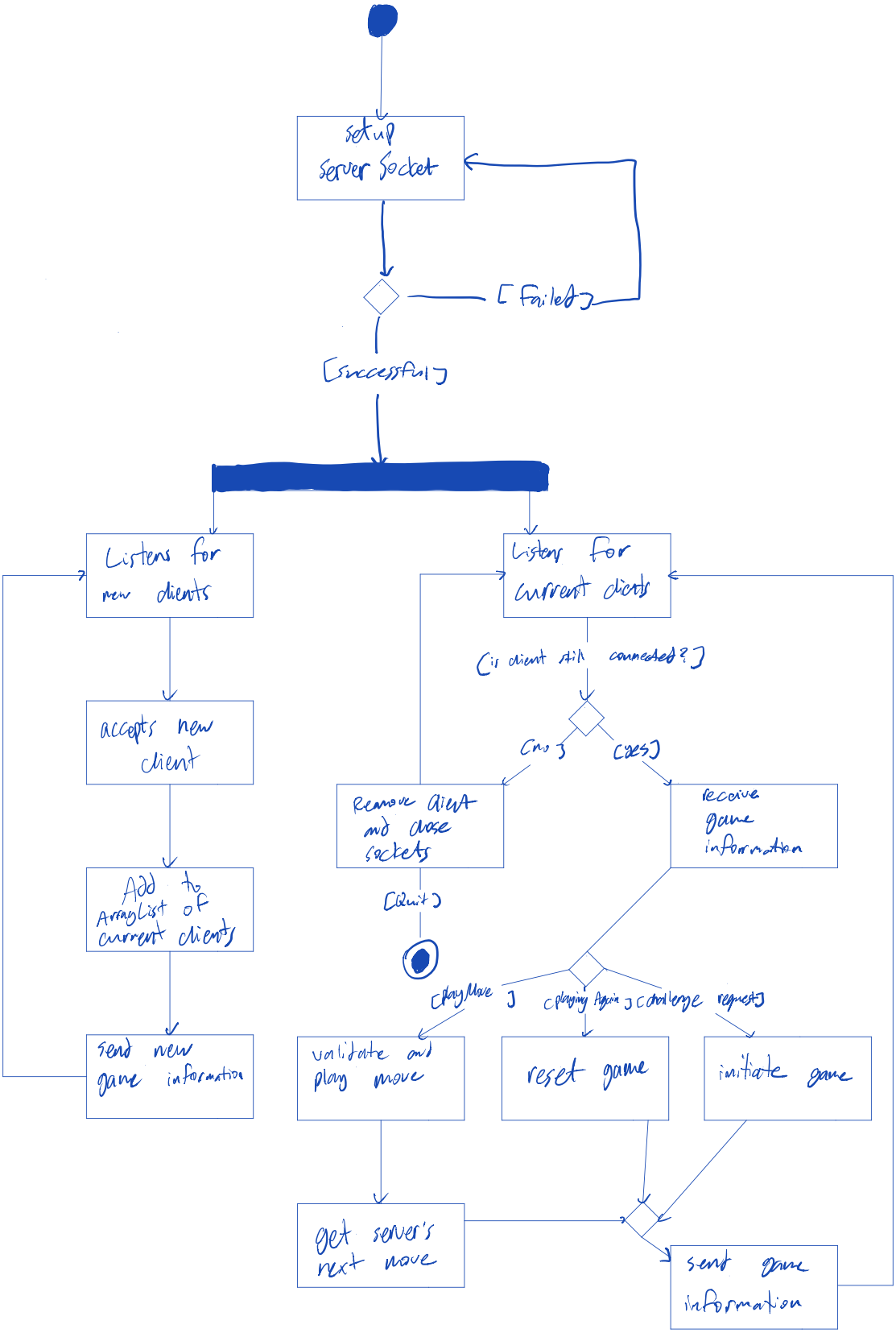
Server Wireframe



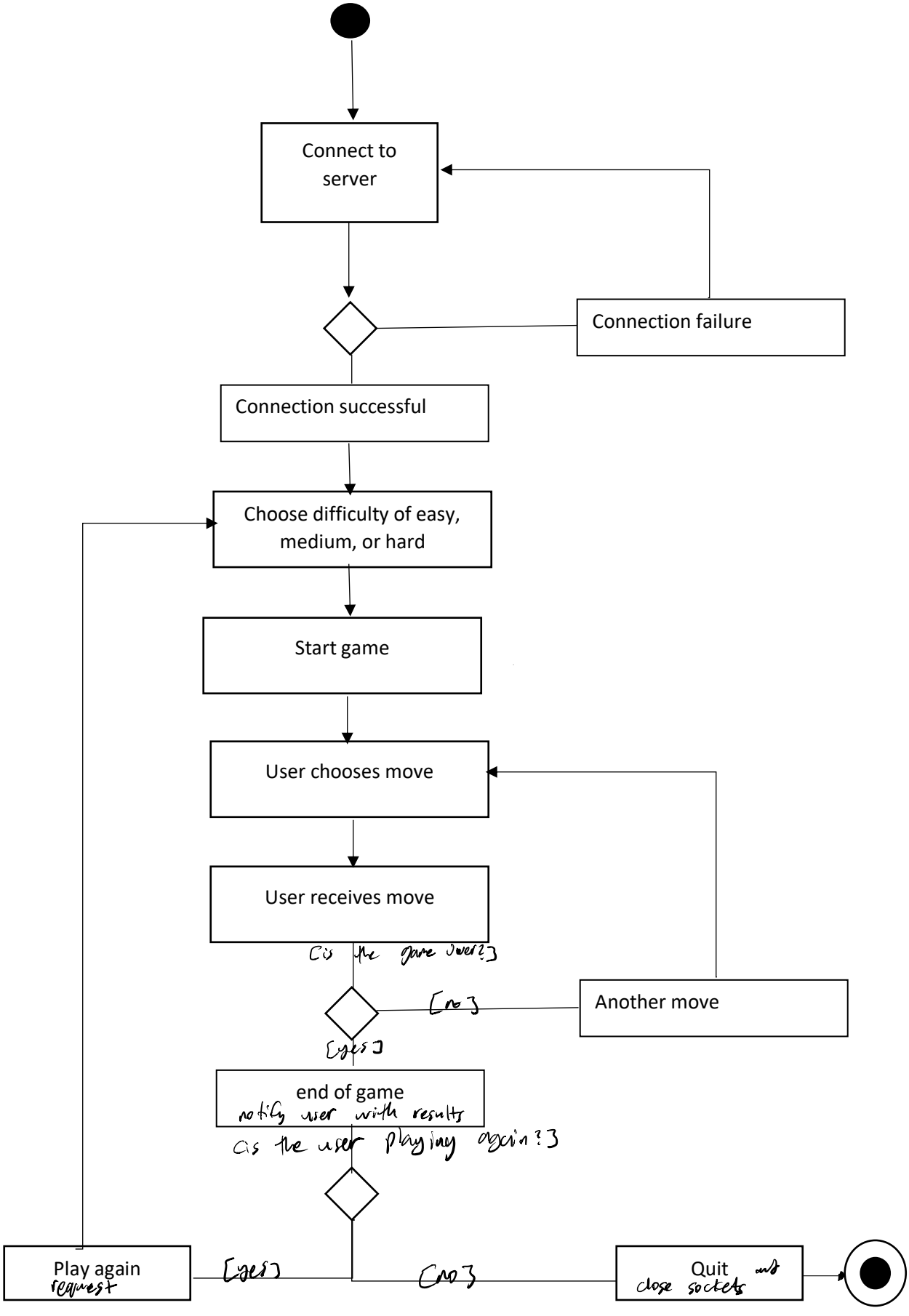
Client Wireframe



Server Activity Diagram



Client Activity Diagram

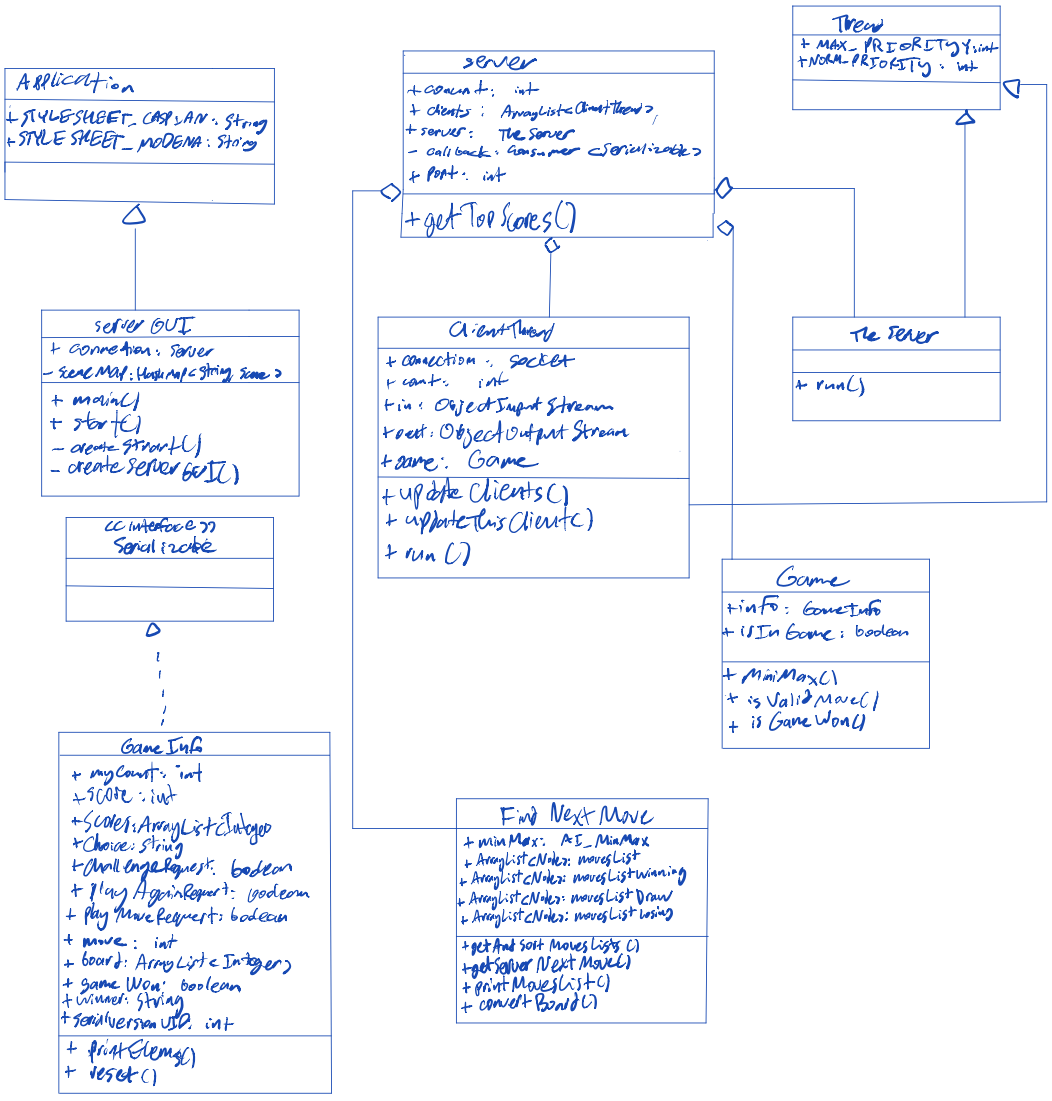


Wael Mobeirek wmobeir2@uic.edu
Enoc Carranza ecarra1@uic.edu
Taha Khomusi tkhomuz@uic.edu
Hasan Selmani hselmaz@uic.edu

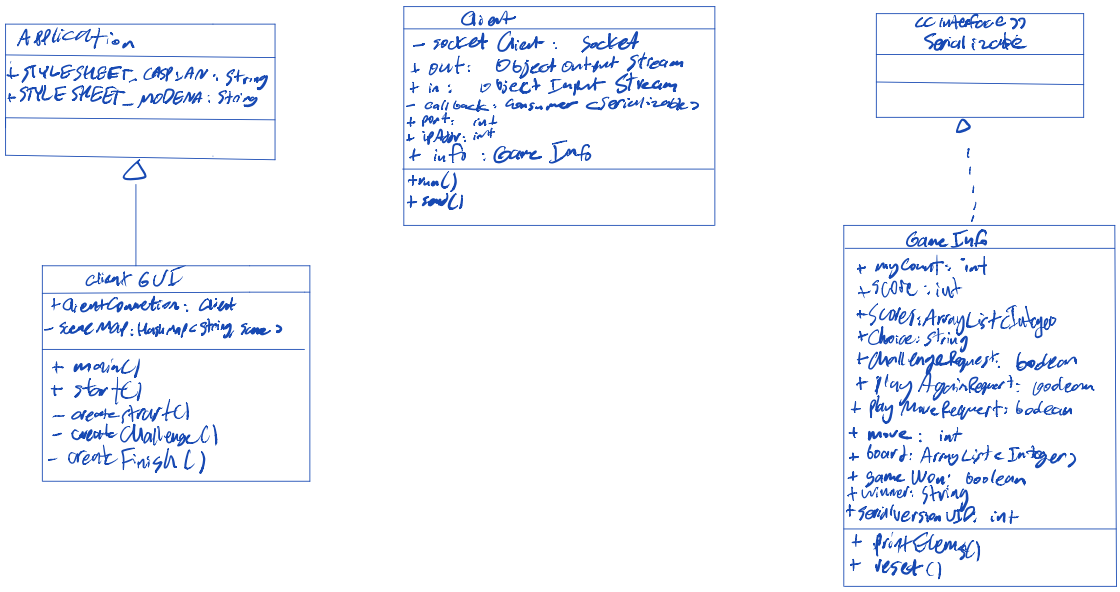
Team 43

Project 5 : UML Class Diagram

- Server



- Client


Application
+STYLESHEET_CASPIAN : String
+STYLESHEET_MODENA : String
clientGUI
+clientConnection : Client
- sceneMap : HashMap<String, Scene>
+ main()
+ start()
- createStartC()
- createChallengeC()
- createFinishC()
Client
- socketClient : Socket
+ out : ObjectOutputStream
+ in : ObjectInputStream
- callback : Consumer<Serializable>
+ port : int
+ ipAddr : int
+ info : GameInfo
+main()
+send()
<<interface>>
Serializable
GameInfo
+ myCount : int
+SCORE : int
+Scores : ArrayList<Integer>
+Choice : String
+ChallengeRequest : boolean
+ playAgainRequest : boolean
+ playMoveRequest : boolean
+ move : int
+ board : ArrayList<Integer>
+ gameWon : boolean
+winner : String
+serialVersionUID : int
+ printElements()
+ reset()

** The End **