Program 5: Trivia Game!

Software Design Document

Team Number: 12

Class Section: 11:00am-12:15pm

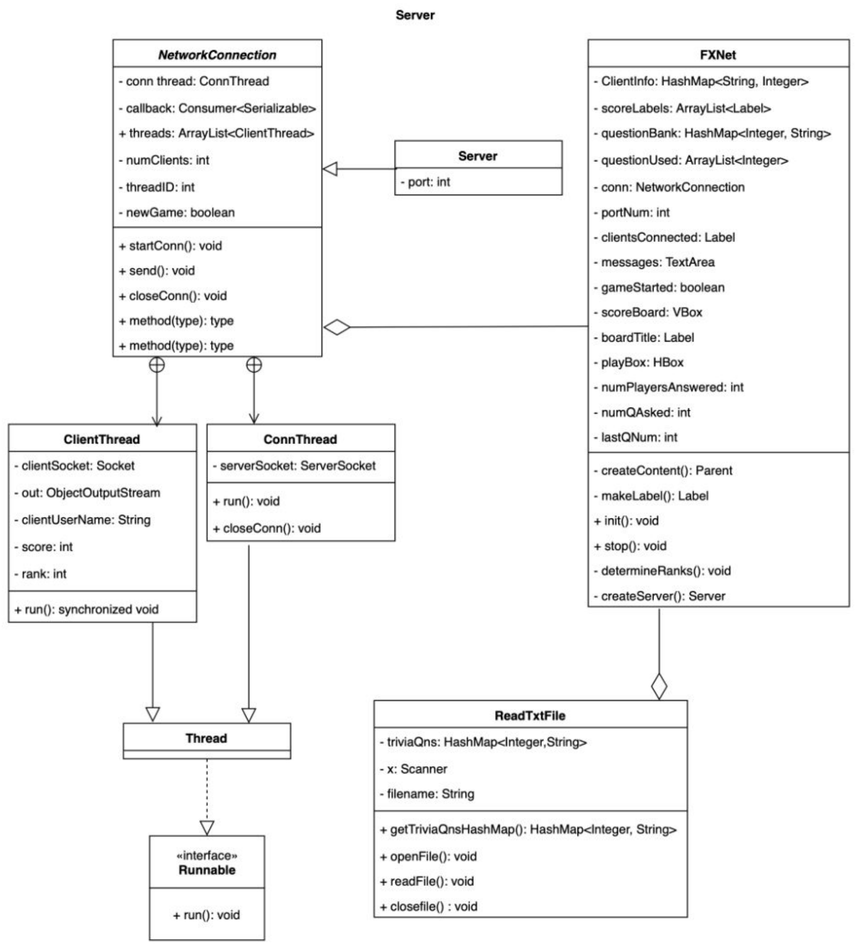
Name	Email
Kirun Haque	khaque3@uic.edu
Michelle Baginski	mbagin3@uic.edu
Saema Ansari	sansar27@uic.edu
James Iorgovan	jiorgo2@uic.edu

Purpose:

The purpose of this project is to create a four player trivia game for our last group project in our Software Design class. The game will consist of 10 questions and each question is answered correctly, the players will earn a point. At the end of the game all of the players' scores will be ranked. The four player game is required to to have server and client communication. The four players will obviously be the clients and server will be used to help compute scores and be the middleman for the clients to be able to communicate to each other. The project is using 2 programs, one for the server and for the client.

High Level Design Description:

-Server



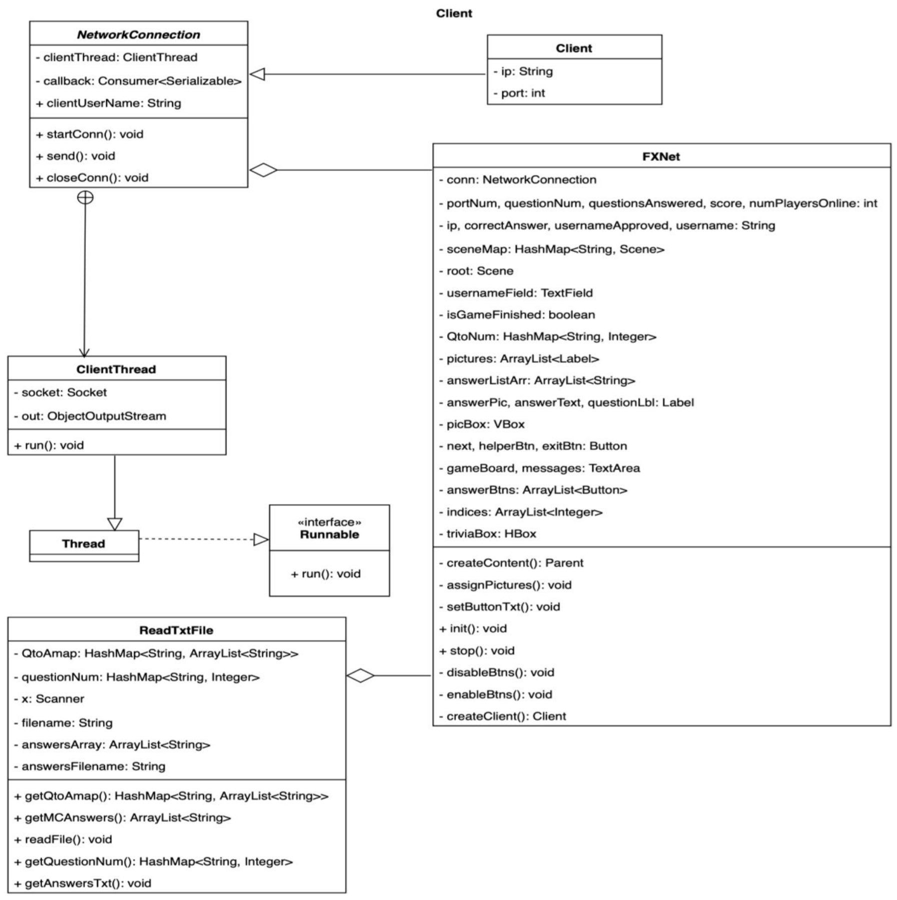
Networkconnection is class with two inner classes. The two inner classes are used to create a thread that will be listening to clients to connect which is the ConnThread class and keep. The other inner class is ClientThread which is will indicate each client that is connected to the server. The Networkconnection class is also responsible of keeping track of all of the clients connected to the server and to send messages to the clients.

Server class extends network connection and it is used to create an instance of a server after the port number is chosen.

ReadTxtFile class is used to extract from a text file which will record the all of the questions that will be used in the game. While extracting the file, the question will be inserted in a HashMap as a value and the key will be a integer value.

FXNet is a class that will create an instance of Networkconnection and server. The class is also responsible of creating the graphical user interface and prompted the user what port number should the server be created with. FXNet also sends a question number to the client program and stores that number so it won't be reused again. The question number indicated which question will be given to the clients. At the end of the game, FXNet class will be used to rank the players scores and sends that information to all of the clients connected to the server.

-Client Program


Client
NetworkConnection
- clientThread: ClientThread
- callback: Consumer<Serializable>
+ clientUserName: String
+ startConn(): void
+ send(): void
+ closeConn(): void
Client
- ip: String
- port: int
FXNet
- conn: NetworkConnection
- portNum, questionNum, questionsAnswered, score, numPlayersOnline: int
- ip, correctAnswer, usernameApproved, username: String
- sceneMap: HashMap<String, Scene>
- root: Scene
- usernameField: TextField
- isGameFinished: boolean
- QtoNum: HashMap<String, Integer>
- pictures: ArrayList<Label>
- answerListArr: ArrayList<String>
- answerPic, answerText, questionLbl: Label
- picBox: VBox
- next, helperBtn, exitBtn: Button
- gameBoard, messages: TextArea
- answerBtns: ArrayList<Button>
- indices: ArrayList<Integer>
- triviaBox: HBox
- createContent(): Parent
- assignPictures(): void
- setButtonTxt(): void
+ init(): void
+ stop(): void
- disableBtns(): void
- enableBtns(): void
- createClient(): Client
ClientThread
- socket: Socket
- out: ObjectOutputStream
+ run(): void
Thread
«interface»
Runnable
+ run(): void
ReadTxtFile
- QtoAmap: HashMap<String, ArrayList<String>>
- questionNum: HashMap<String, Integer>
- x: Scanner
- filename: String
- answersArray: ArrayList<String>
- answersFilename: String
+ getQtoAmap(): HashMap<String, ArrayList<String>>
+ getMCAnswers(): ArrayList<String>
+ readFile(): void
+ getQuestionNum(): HashMap<String, Integer>
+ getAnswersTxt(): void

Networkconnection has an inner class called ClientThread which creates a socket and ObjectOutputStream so the client can communicate to the server. The methods in the Networkconnection class are used to start the ClientThread instance, close the connection, and to send a message directly to the server.

The client class extends the Networkconnection and it is used to get the ip address and port number so the client can connect to the server.

The ReadTxtFile class is used to extract text from a text file and record the questions and list of multiple choice answers and insert into a hashmap. The class is also used to extract from answers.txt file so it can record background information on the correct answer into a string ArrayList.

FXNet class is used to to create the graphical user interface (GUI) and to create a instance of Networkconnection so the client can communicate to the server. FXNet also creates an instance of ReadTxtFile so it can have access to what question and answers to display on the GUI). The class is also responsible to see if the client answered the question correctly and records the client's score. Overall, FXNet is used to create the GUI and to display the correct question, multiple choice answers, and to calculate the client's scores.

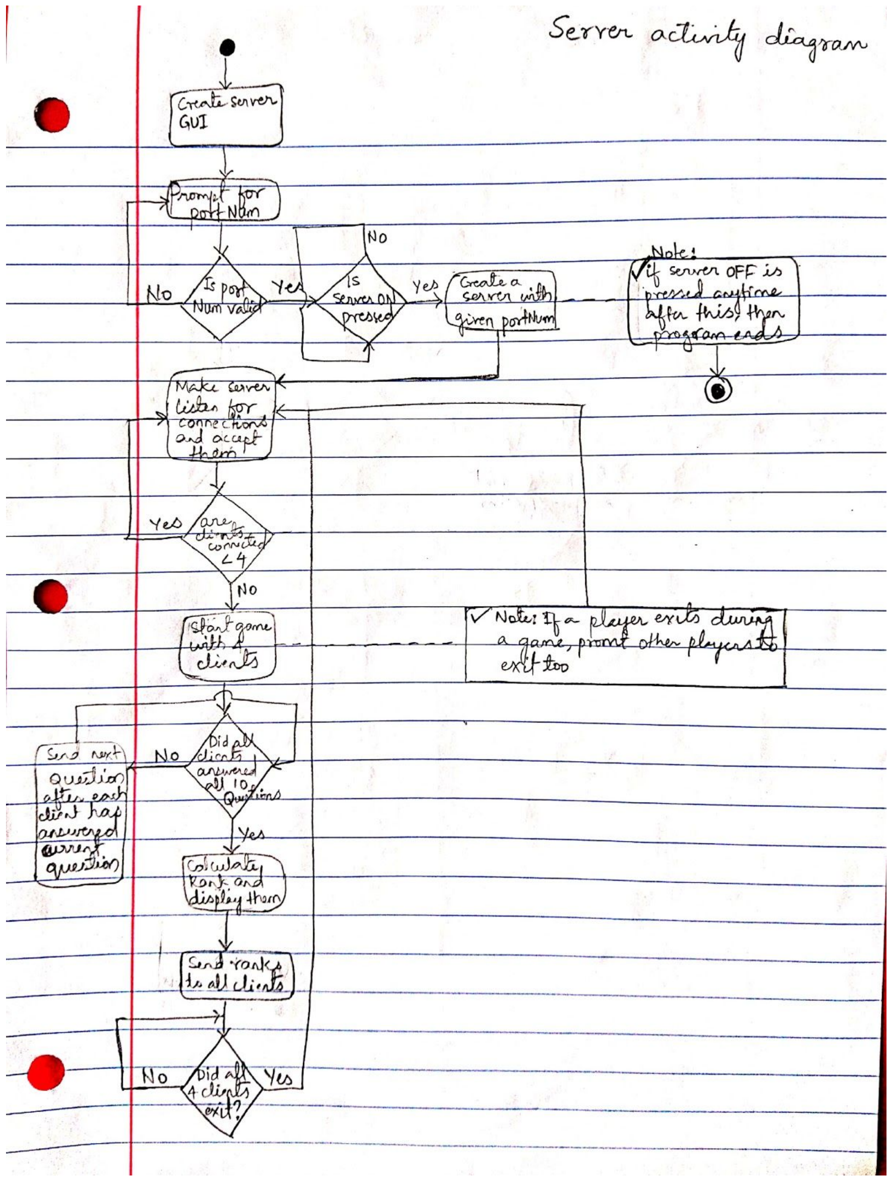
Low Level Design:

The client program will extract from a text file to get the questions and list of multiple choice answers that corresponds to each question and insert it into a hashmap. The hashmap *key* will be the question and the *value* will be the ArrayList<String> of multiple choice answers. The client will receive a question from the server program, so client program will know which question to display on the GUI and to retrieve the ArrayList from the hashmap. There will be an ArrayList<Integer> that will contain 0,1, and 2 in the list. That list will be shuffled every time the server sends a question. After retrieving the ArrayList from the hashmap, the program will traverse the ArrayList<integer to retrieve the multiple choice answer to set the button texts. Once the client chooses an answer, the client program will see if the client answered the question correctly to give the client their score and then their score to the server. Thus, Client program will only communicate with the server when the client chooses a username to check if its not taken and to send their score to the server.

The server program will extract a questions from a text file and insert it into a hashmap as the *value*, the key value will be a integer. The server will randomly choose a number 1 through the number of trivia questions to retrieve the question where the random number is mapped to on the hashmap. The server will record each random number that is used in some sort of collection, so no questions are repeated in one game. Once the question is retrieved, the server will send it to the client program. The server will receive data from all of the clients once all of the clients answered the question to keep track of everyone's scores. Once the game is over, the server will send the players ranks to all of the clients. Thus, Server program will only communicate with client program, to check if the username is unique, to choose a trivia question , to keep track of all of the player's scores when the client sends data after they've answered a question, and send players ranking once the game is over.

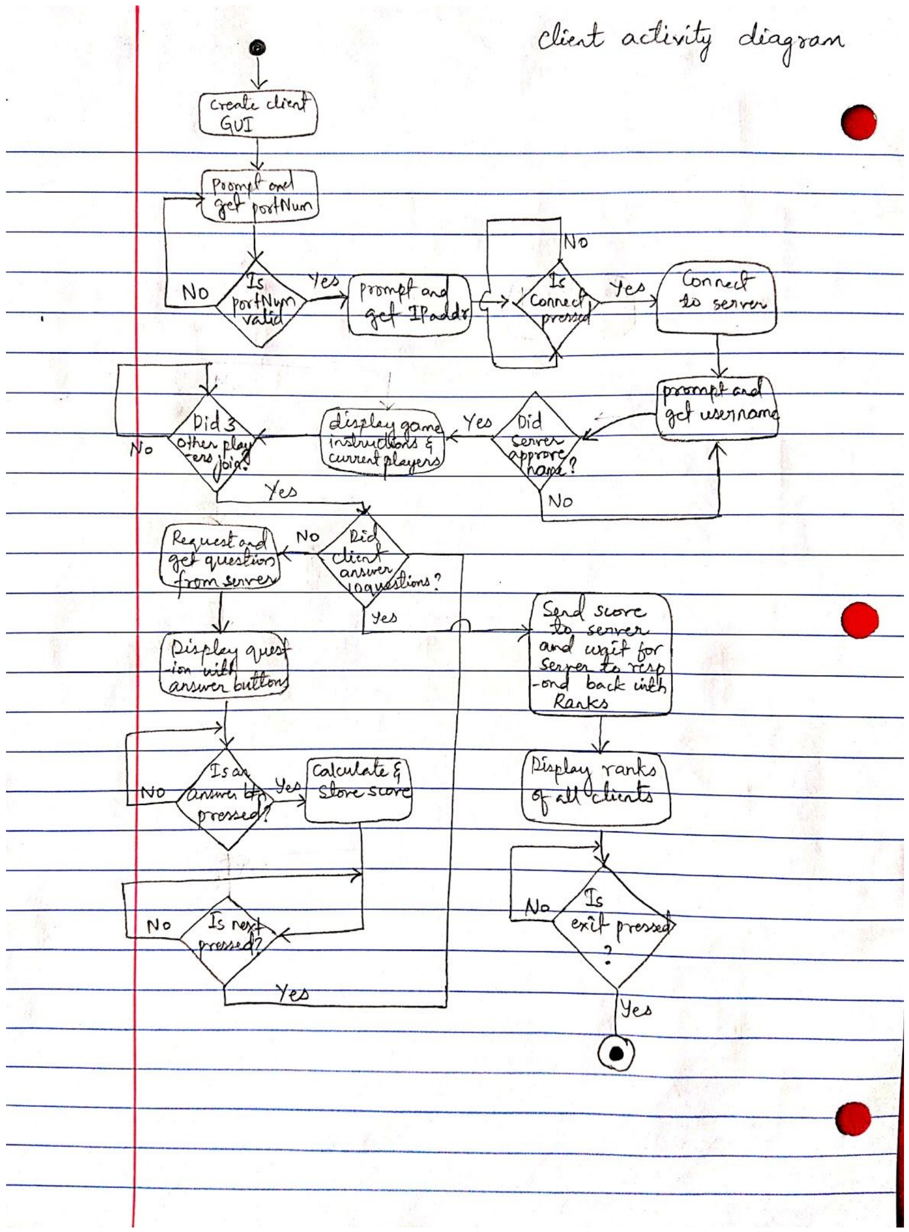
The figure below shows the flow of the server program

Server Activity Diagram


Server activity diagram
Create server GUI
Prompt for portNum
Is port Num valid
No
Yes
Is server ON pressed
No
Yes
Create a server with given portNum
Note: if server OFF is pressed anytime after this, then program ends
Make server listen for connections and accept them
are clients connected < 4
Yes
No
Start game with 4 clients
Note: If a player exits during a game, prompt other players to exit too
Did all clients answered all 10 Questions
No
Send next question after each client has answered current question
Yes
Calculate Rank and display them
Send ranks to all clients
Did all 4 clients exit?
No
Yes

The figure below shows the flow of the client program

Client Activity Diagram


client activity diagram
Create client GUI
Prompt and get portNum
Is portNum valid
No
Yes
prompt and get IPaddr
Is Connect pressed
No
Yes
Connect to server
prompt and get username
Did server approve name?
No
Yes
display game instructions & current players
Did 3 other players join?
No
Yes
Did client answer 10 questions?
No
Yes
Request and get question from server
Display question with answer button
Is an answer btn pressed?
No
Yes
Calculate & Store score
Is next pressed?
No
Yes
Send score to server and wait for server to respond back with Ranks
Display ranks of all clients
Is exit pressed?
No
Yes

Benefits, assumptions, risks/issues

The risks of this program is having the text file of the answers in the client program, because it’s possible for a client to figure out that the answer key is on their program. Also having two of the same text files in the client and server is a risk because if one text file is incorrect it will affect the other program.

There was risk when this project did not have any sort of action when a client exited a program. The issue had been resolved when we notified all of the clients that are online that player has dropped from the game and prompted them to exit the game.

However, the issues regarding when a clients exits the program is when they exit after connecting and before the client enters a username, it can cause issues to the server. The server works with 0 issues after the clients has connected to the server and entered a username. That is the only issue with exiting the program.

Benefits of this project, it has 4 clients that can connect to the server and nothing more per game, which meets the requirements of the project. There's a healthy competition between 2 players and it is able to rank players fairly. The questions that are being asking in this program is a good learning experience because the questions our based on fun facts and tricky answers.

Conclusion

This project uses a server and client communication to create a trivia game between 4 players with 2 different programs. The players scores will be ranked in the server program while their scores are being tracked in the client program. Both program will have access to the questions and answers to the game in order for this program to function properly. The game will be played 4 players at a time, there are only slight issues with exiting. Other than that, this is full functioning project that can have 4 clients playing a trivia game communicating through a server.