CS 342 Project 5
Guess a Number

Team 19 (11am section)
Levi Herrera - levih2@uic.edu
Ellen Kidane - ekidan2@uic.edu
Trishla Shah - tshah28@uic.edu
Mohammad Ramahi - mramah4@uic.edu

Purpose of the Project
This project is designed to fulfill the requirements of project 5 in CS 342, Software Design. These requirements consist of creating a multiplayer game, with at least four players, using Java and JavaFX, making use of sockets and threads, or Node.js with some open source frameworks. The game must be run on a server, with each player as a client, and the game chosen must be an original implementation.

The rules of the game we decided upon are described below. This game may be played by individuals who particularly enjoy guessing numbers. Since the game is a desktop application, developed using Java and JavaFX, a user with programming skills may be able to configure the game to his/her liking. For instance, a user may change the background color or font on the GUI windows of the client and/or the server, or perhaps the user would like to have only one winner in the game, so as to allow multiple people to come to a decision about who goes first in a game or which restaurant they would like to go to for lunch. Due to this, this game is easily customizable and adaptable.

Rules of the Game
Server randomly generates a positive number in some range like 1 - 100. All clients try to guess what it is.The server gives one hint about the number to start with. When a client guesses incorrectly the server will give them information about the number. No client can guess twice. If they guess correct number with at most all clients guessing then the clients win; otherwise they lose. Losing takes clients to a losing screen and then winning takes them to a victory screen. After a game the server makes up a new number.

Design Chosen
Languages and Frameworks - The languages and frameworks used will be Java, for the logic of the game, and JavaFX, for the GUI programming and presenting the screens that the user sees. We will be making use of networking with sockets and threading within Java to allow for communication between the server and all of the clients.

Client/Server Relationship - As mentioned previously, the client and server will communicate via networking with sockets in Java. The server will control the flow of the game and will be the primary source of the game logic. As such, the server will be the one choosing the number at the outset of the game; it is this number that the clients have to try to guess correctly and which will be hidden from the clients (only the server will know the value of the number). Each client's guess will be sent to the server, which will compare it to the correct number and notify the client if they need to guess a different number. In the case that the client guesses the number correctly, the server

will notify that client that they won and also will notify the other clients that this one has guessed the number correctly. The clients are on a team together and once one of them guesses the number correctly they all win.

User Interface - The user interface will be similar to ones used in previous projects, with the server and each client all having their own GUI window. The GUI for the server will have a way for the user to enter the port number, which will be used to start the server and establish the connection. At the end of the game, it will also include a way to display which player or players won and what the actual number was. The GUI for the client will include some elements that allow the user to enter the port number and IP address that allow it to connect to the server. The window will also include a text area in which messages from the server will be displayed, and a means for the client to enter their number and send it to the server. After each guess, the GUI will display the number that the client chose and whether the server said to go higher or lower. If the client won, this will also be displayed on the GUI window.

Details of the Design

ClientGUI.java - This file contains the code for the GUI elements of the client window. It inherits from the Java Application class; this allows the class to be able to access the data members and methods of the Application class directly, while still adding its own data members and functions to allow elements from the JavaFX library to be used in the window and to allow for communication with the server. The window itself is quite simple; it displays a message based on what the server sent, as to whether the client won or lost, and includes a way for the client to guess a number. This is done by means of a text field, in which the user may enter the number they would like a guess, and a button which, when pressed, sends this number to the server.

Client.java - This file sets up the thread that is used for the client to communicate with the server. It defines methods to send information to the server, open and close the connection to the server, and receive information from the server. The class that controls the GUI window of the client is thus able to use this connection so that, when the button is pressed, the number is sent to the server. It also makes use of this file to update the GUI with the information sent from the server.

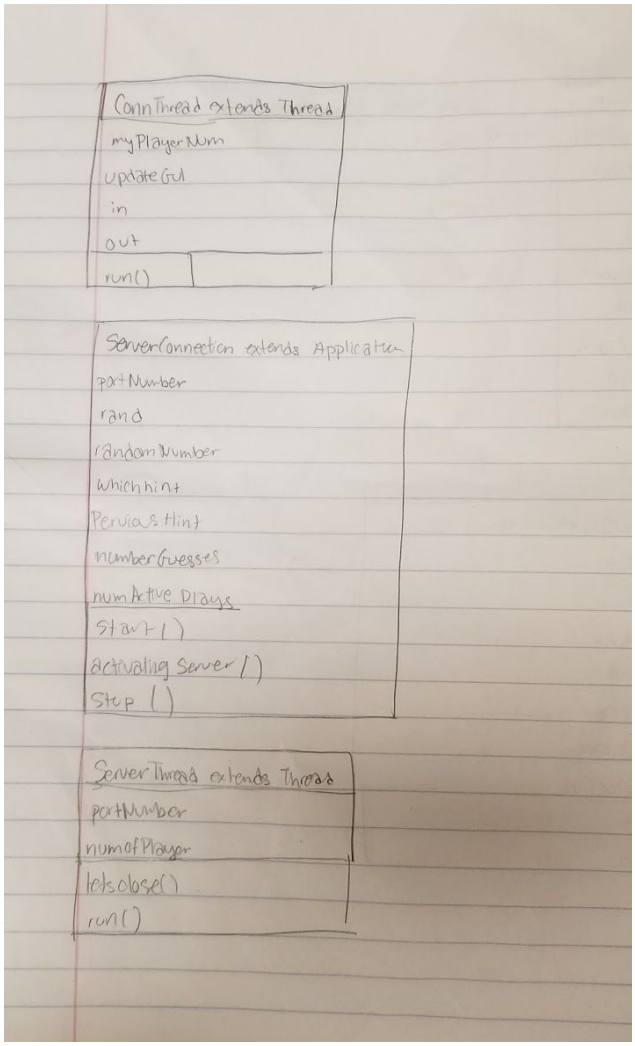
ServerConnection.java- This file contains the code for the GUI elements of the server window. There are 3 items - one button to start the game, one text field to show the number of clients, and the last text field to show what information will be sent out to the

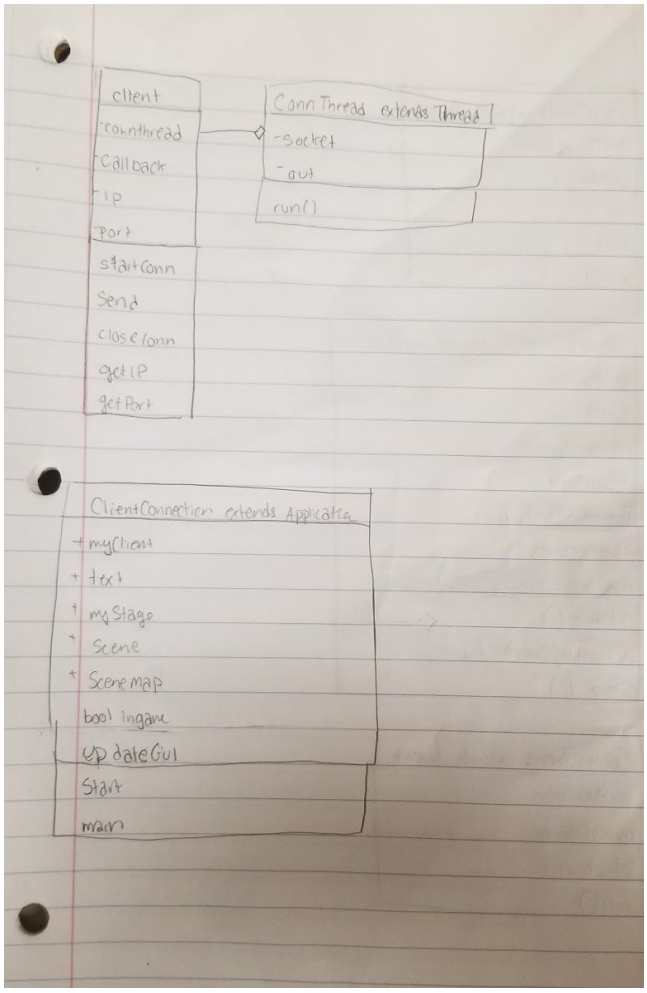
clients. While the GUI is quite simple, so the basis of the game. It allows for teamwork over competition. It also has data members to keep track of the random number that is generated, the number of guesses made, the number of players, the number of clients connected, and what hint to give to the client. This allows the server to control the logic of the game.

ConnThread.java - This file is essentially the same as Client.java; it exists so that the server may create a new thread every time a client connects, although the server is in a separate project from the client.

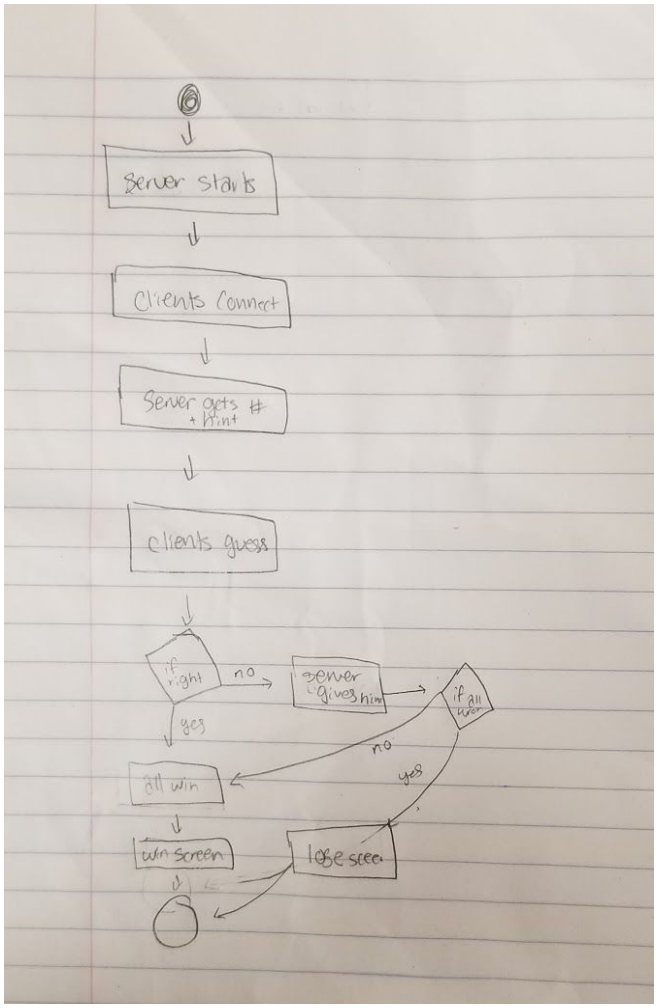
ServerThread.java - This file sets up the thread that is used for server to communicate with all of the clients. It defines methods to send information to the client, open and close its connection, and receive information from the clients. Once the information is received from the clients, this class also interprets this information and notifies the clients with the updates accordingly. The class that controls the GUI window of the server is thus able to use this connection so that it may be updated with information received from the clients.

UML Diagram (Model)





Activity Diagram (Interaction)



Benefits, Risks, and Assumptions Made

Because this game is based on teamwork, if one client guesses correctly, all the clients win. This fosters a positive environment for the users and may be encouraging to someone who is very bad at playing games - since they can all win in this game, he/she can now say that they won a game.

One assumption made is that this game is played in a way where all the players cannot physically see each other. If they can see the exact number guessed by the previous person then it gives them a advantage but this is okay since this game is built on teamwork. If one person wins we all win. Another drawback is that you only get one chance. If they guess the number but too late, then they lose.

We are also operating under the assumption that the users all know to use integers only rather than including floats. While we randomly choose a number, there is nothing that

is truly random, we must operate under the assumption that it is as random as it can get under the laws of this universe. Overall, the risks of playing this game are pretty minimal at best. At worst, you might feel sad for losing.