*Kevin Kowalski*
*John Oshana*
*Francisco Ramirez Cruz*
*Alec Thomas*

# Group 21, 11:00 AM

| | |
|---|---|
| Kevin Kowalski | kkowal28@uic.edu |
| John Oshana | joshan3@uic.edu |
| Francisco Ramirez Cruz | framir23@uic.edu |
| Alec Thomas | athoma86@uic.edu |

*Kevin Kowalski*
*John Oshana*
*Francisco Ramirez Cruz*
*Alec Thomas*

**Project Design**:

Our game, Speed Typer, allows a user to compete in a typing race against other users. Once connected, the objective of the game is to type out the word given to you as fast as you can. The goal of the game is to get the highest points possible. Each game will play until someone first reaches five points. A point is awarded to the user that first successfully typed the current round's word before any other client could. This game was created for anyone who wants to test their typing skills or improve them in a fun and competitive way.

Speed Typer consists of both server and client Java applications. The server application creates a server in which clients can connect to using Java sockets. Once connected, clients can join currently hosted lobbies or create their own and once a lobby is full the game will automatically begin. Each application is programmed with an interactive graphical user interface using JavaFX.

Lobby Class

Used as a hub for clients that want to play a new game. A single lobby can hold up to four players max and will start a game as soon as all players have connected. Each lobby is a unique container holding its own name, status, and list of connected clients. The list of clients is in the form of a hashmap where the hashmap is used to store an individual client's name (as key) and score (as value). The status of the lobby will either be "JOIN" or "FULL" which will inform a client whether or not they can join said lobby. If a user attempts to join a full lobby, the client will receive a message

alerting them that they cannot join the selected lobby. Once the lobby is populated with four players, it will automatically create a new Game object.

Game Class

Contains all game logic as well as clients in a current game session. The game object creates a list of strings that have been parsed from a file to use as words to send to users. A random word will be selected every round and that same word will be sent to each client. The client that successfully sends the round word first will be awarded a point. In order to avoid confusion between clients, the game will wait until all clients have successfully typed the round word. Each client is placed into an array list after they have sent the word and the user in the first index will be awarded the round point.

Points are kept track of inside of each client's score value. After the fifth round is completed, the game object will call a function called determineWinner() which will check each client's score. A second function called setGameWinner() will be called and inform the clients of the winner and losers. The clients are shown an alert box declaring whether they have won or not and they will then be returned to the lobby view screen.

Client Class (server-side)

The client class holds all the necessary information that the server needs about a player: user name, connectionID, message to send, score, lobby, and server connection are some of the data stored for a single client. Each client is stored into an array list for easy upkeep and they have multiple functions that allow them to send information back

and forth to the server and other clients. For the most part the client code consists of getters/setters in order to change the private variables in the client class.

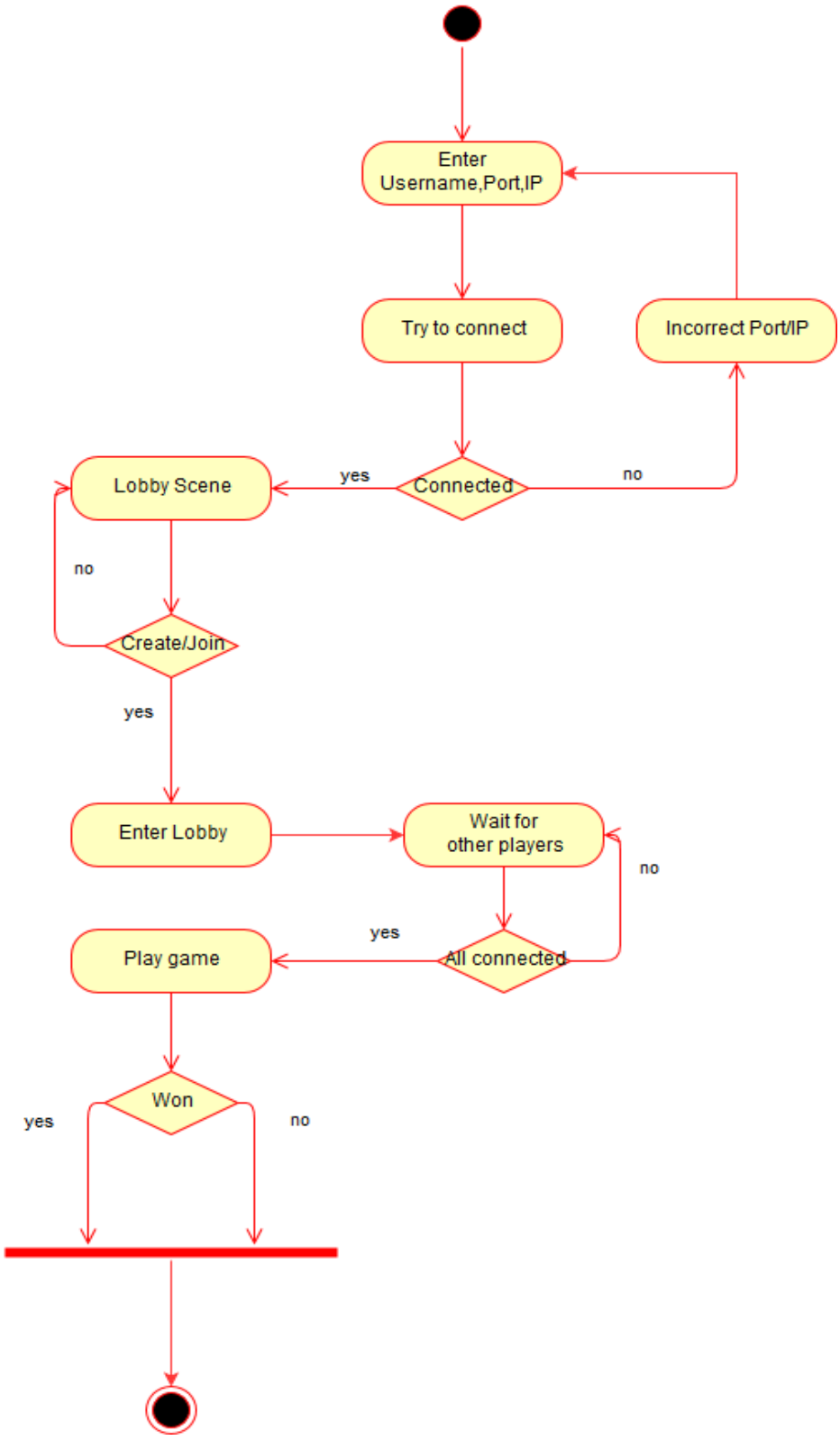
Client Class (client-side)

In addition to what is on the client's class of the server, the client class on the client program has additional functions and features for commutating to the server and other clients. The client class also extends thread so that each client is on its own thread. Once a client thread is created a client is greeted with a login message.

A function for creating a lobby allows a client to create a new lobby which will inform all other connected clients and the server that a new lobby has been made available. A join lobby function allows a client to join a lobby that isn't full and will notify all clients/server that the client has entered selected lobby. There are also functions that update the GUI to represent any changes that the server or client has requested.

*Kevin Kowalski*
*John Oshana*
*Francisco Ramirez Cruz*
*Alec Thomas*

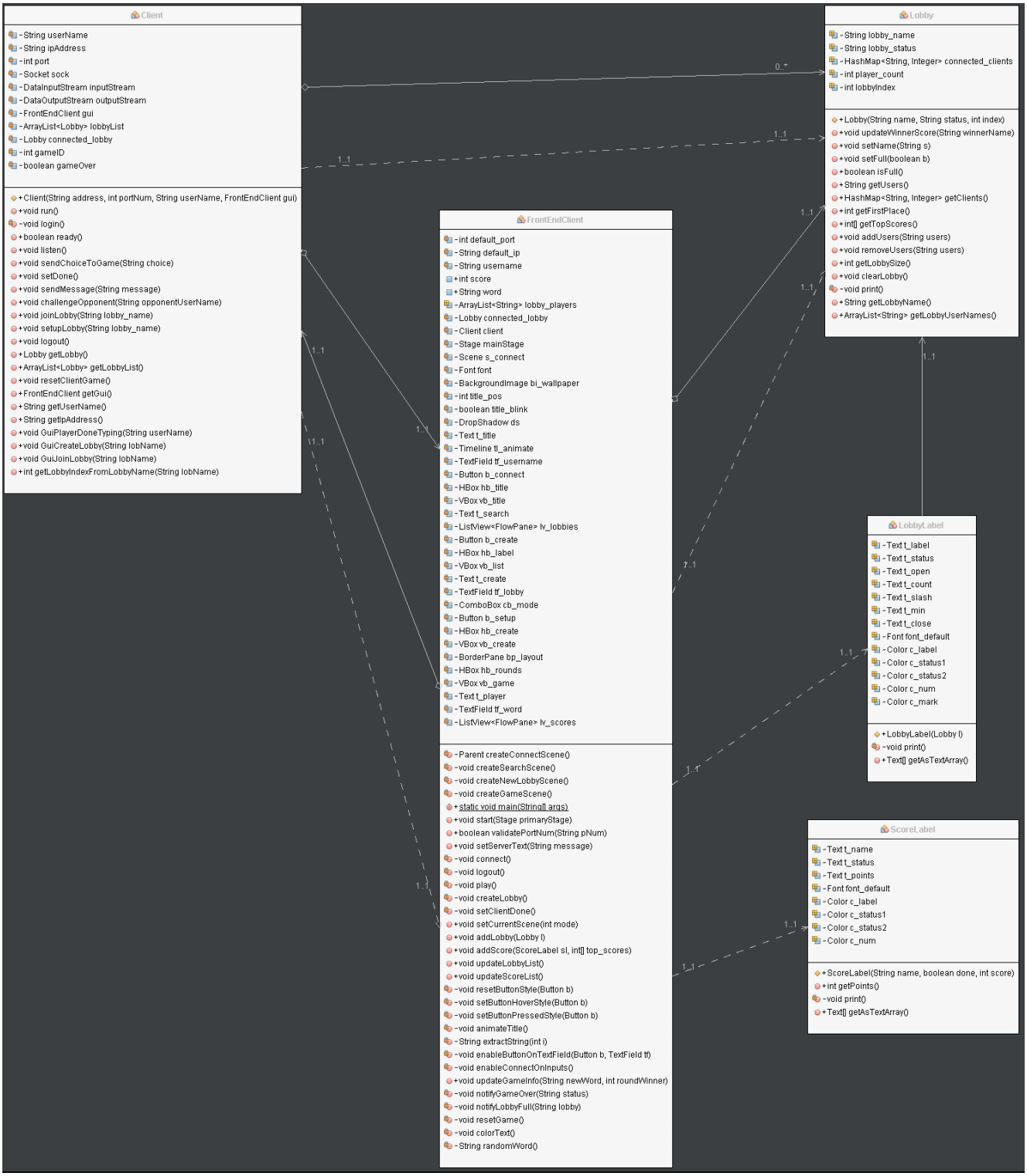
**Activity Diagram**:



*Kevin Kowalski*
*John Oshana*
*Francisco Ramirez Cruz*
*Alec Thomas*

**UML Diagrams**:

## Client UML



*Kevin Kowalski*
*John Oshana*
*Francisco Ramirez Cruz*
*Alec Thomas*

# Server UML