Quiz Quest

Documentation

Team #6 – TrashCan Productions

Filipa Ivanković

Ivo Ivanković

Saelda Rrahi

Date: 3th May 2019

Revision: 1.0

**Table of Contents**

Revision History 3

Executive overview 4

Audience 4

Application intentions 5

Audience 5

Assumptions made for this project 5

*Gantt chart* 6

Visual chart – made by Saelda | Table Gantt chart – made by Filipa 6

*Class and method overview* 7

Quiz Quest UML 8

Simple UML – made by Saelda 8

Client GUI 9

Server GUI 11

Networking connections & Protocols 12

Communication class 12

Data used 14

Data files 14

Unresolved Issues 15

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason for Change** | **Version** |
| Filipa | 10.4.2019. | - Design Document Initial form | 0.0 |
| Team | 24.4.2019. | - Explaining the game’s code and starting | 0.1 |
| Ivo | 25.4.2019. | - Combining Quiz Quest game files with Server/Client files that were created prior to this project | 0.2 |
| Filipa & Saelda | 26.4.2019. | - Recreating the Server & Client due to a problem with using Print Writer as the main stream for sending messages (replaced with ObjectOutputStream) | 0.3 |
| Ivo | 27.-28. 4. 2019. | - Combining new Chat(Client) with Quiz Quest game  - Server correctly reads all Client & Game inputs  - Testing  – Game is able to be played | 1.0 |
| Filipa | 29.4.2019 | - Testing game capabilities for in-class demonstration  - Minor coding and comments | 1.0.1 |
| Filipa | 30.4.2019. | - Created Server GUI  - Credits menu option on all frames  - Server prints out every necessary game/client status and winner into server GUI  - JavaDocs and comments on all classes  - Lots of testing | 1.0.2 |
| Filipa | 1.5.2019. | - Implemented a login system into the Client to replace the necessity of using args & java.util.Scanner, and to simplify the start-up process  - Client Login functionality and menu options  - Server.jar & Client.Jar File  - Instructions txt file & on every GUI  - Testing  - GAME IS FINISHED & UPLOADED | 1.0.3 |

# Executive overview

Quiz Quest is a small, Java, 2 player, text-based board game where the goal is to reach the end before the other player, while answering a bunch of silly, nonsensical, bizarre and downright absurd questions. Questions can range from something as simple as “Where did I put my keys?” to “How do I commit a crime and get away with it?” The game relies on humor, references, absurdity and stupidity to make the experience enjoyable and fun.

The board consists of 2 player pieces (characters) and 16 spaces. Once a player reaches the last, 16th space, the game is over and the player, who got there first, wins. In the game’s menu, players are able to customize the board’s appearance by choosing between having a colored board, or a simple non-colored black and white board. The menu also offers credits, the option to reset the game (start over from space 1) and rules on how to play the game.

The game is played by pressing the “Move” button. Pressing it will move the current player’s piece by 1 space. After moving, a randomly generated question will pop up. Each question comes either in the form of a question, or a short story on which the player will decide its outcome. On each question, a player will be presented with 4 different options (a, b, c, d choices) with each of them resulting in a different event. Some choices will be correct, while others, incorrect. If the player answers the question correctly, they will be moved by 2 additional spaces on the board; on the other hand, answering incorrectly will send them back by 1 space, placing their piece back where they started before moving. After a question is answered, it is the other player’s turn to move and answer a new question. As of writing this document, there are 12 different questions, meaning there are in total, 48 different outcomes.

In the upcoming improvements, the game will be going online, making the game playable over different computers on a network. This new version will feature a chat where players will be able to communicate with each other and receive some additional information from the game.

# Audience

## Application intentions

The main goal of this project is to create a silly multiplayer board game that cracks jokes and references which will make the players laugh, or simply frustrate them. Some questions are simple and fun, while others are made to be annoying and difficult. Quiz Quest is basically a double edged sword of a game that looks to achieve both the good and the bad feedback. It is a game that is best enjoyed with a couple of friends to minimize frustrations and maximize the fun.

## Audience

Because Quiz Quest relies heavily on its jokes and references, someone who might not get the humor, or simply doesn’t enjoy the type of humor Quiz Quest provides, we tried to incorporate different types of humor into the questions, so at last one question will be able to make the players crack a smile or frustrate them. The game fails, if the player doesn’t like/love the game, or dislikes/hates it.

The comedy in Quiz Quest is based around popular culture, existentialism, absurdity, stupidity, irony, memes, dark humor and shock value; if you enjoy that type of comedy, Quiz Quest might be for you. As a text-based game, Quiz Quest will require a good amount of reading to be experienced to its fullest. So, if you are the type of person who considers reading a drag and boring, you won’t get much from this game.

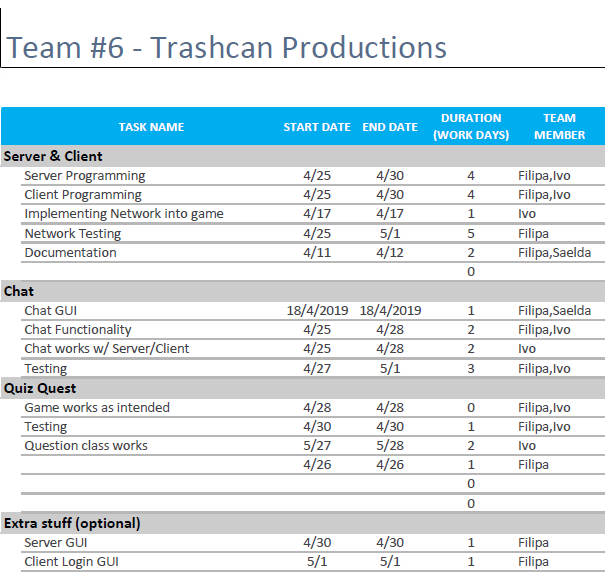
We are assuming that the game’s primary audience will be young adults, adults and older teens. Children likely won’t have the patience to play this game and the game isn’t visually stimulating enough for them. Quiz Quest’s humor will most likely be a miss with senior generations.

# Assumptions made for this project

Quiz Quest is a game anyone able to read and use a computer mouse can play. However, the computer on which the game will be run on, requires to be able to run Java and be able to connect to an online network. No previous video game experiences is necessary to be able to play Quiz Quest.

# *Gantt chart*

## Visual chart – made by Saelda | Table Gantt chart – made by Filipa



# *Class and method overview*

Overview of the classes and functionality.

Client

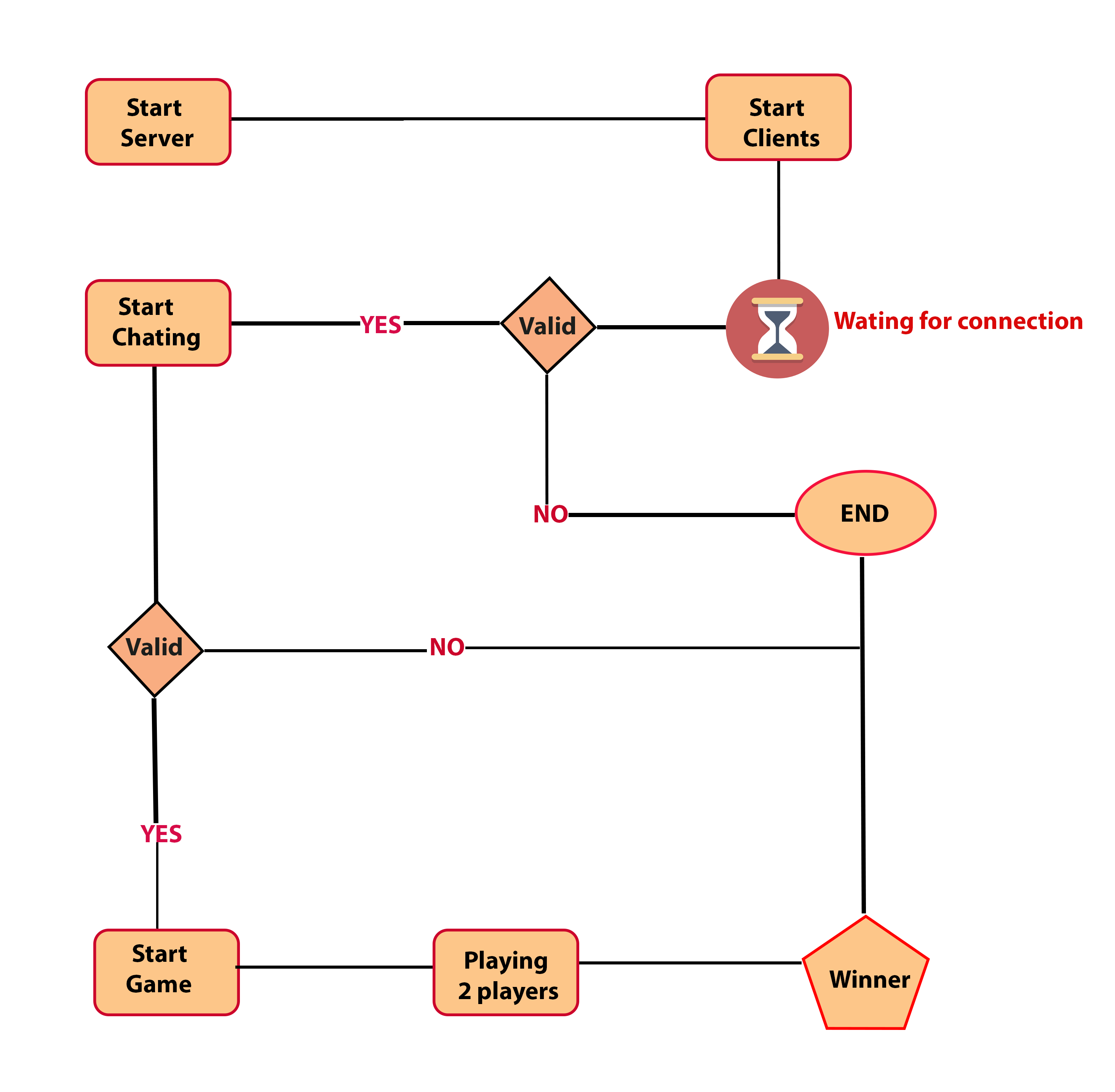
* Main programm
  + Opens up Login GUI – 2 JTextfields for user inputs (ip,username)
  + After valid entry or using entry Login GUI is disposed & the Chat GUI opens
  + Menu options – Exit, Info/Help, Credits/Authors
* Chat
  + Connects to the Server via multithreaded inner class
  + Enables communication with other player
  + Menu Options – Exit, Info/Help, Credits/Authors
  + Start game button which creates an instance of Quiz Quest game
* Quiz Quest class
  + GUI that contains the board, player characters & the game itself
  + Menu option - Exit, Help, Credits/Authors, Reset game, Options – Turn board color on/off
  + Move button which moves the player characters on the board
  + Pressing button opens Question class
* Question
  + Supporting class for Quiz Quest class
  + Contains all the questions and scenarios for the questions
  + Creates a temporary GUI with 4 JRadioButtons that play out scenarios and have fixed true/false values – choices affect player character movement and game progression

Server

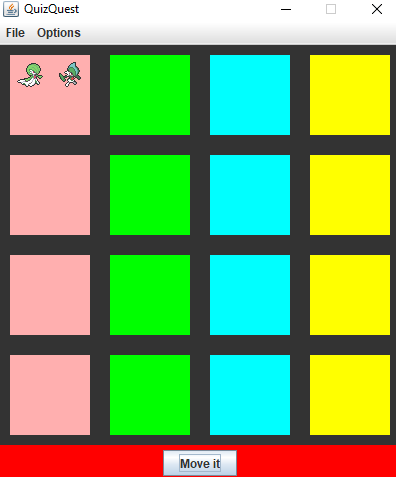
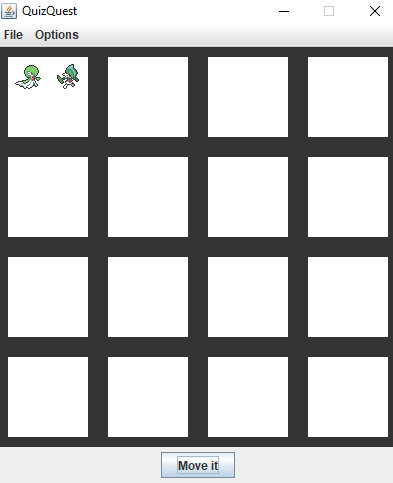
* Main method
  + Opens up Server GUI which contains a JTextArea, JTextFields & Menu options
  + Starts the Server, allowing Clients to connect to the server
  + Menu Options – Exit, Info/Help, Credits/Authors
  + JTextFields – one displays current number of clients connected to server, the other whether the Quiz Quest game instance is running or not
  + JTextArea – will display clients joining/leaving server, game status & progression, some client actions

# Quiz Quest UML

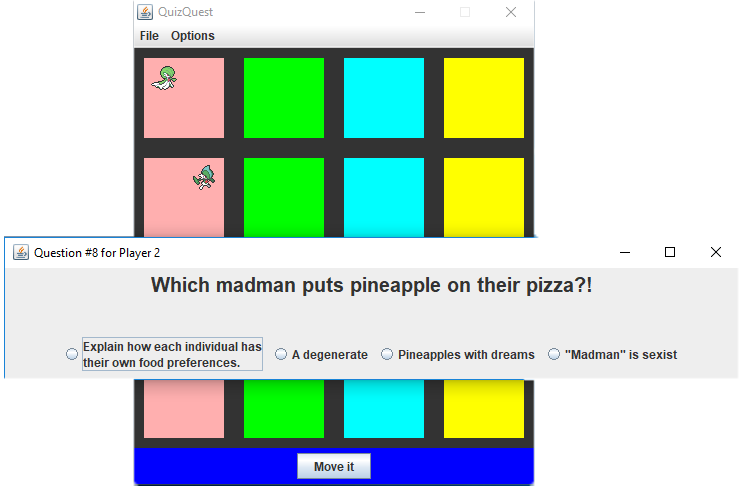
### Simple UML – made by Saelda



# Client GUI



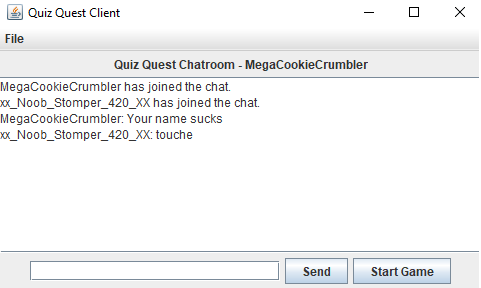
Quiz Quest GUI without colors

Quiz Quest GUI w/ colored board

# 

Example of a question

Victory for Player 1

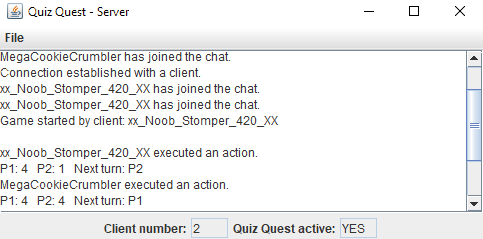


Chat

# 

Login

# Server GUI



Server GUI

# Networking connections & Protocols

Client connection to server:

In the Client class, main program, the Login GUI, the user can specify the ip address by entering it into a JTextField. If the user does not specify an ip, the default “localhost” value for the ip will be selected. On this Login GUI, users also specify their username. If not specified, they will be assigned a random numeric values as their username.

Port number(s): 16789

This port number was selected, simply because it was most commonly used by us during other works. Other than that, there isn’t much significance to that particular number.

Protocol interface code for both client and server:

None.

# Communication class

**Chat interaction between Clients and Server.**

| **Client** | **Communication** | **Server** |
| --- | --- | --- |
|  |  | Startup – opens Server GUI & allows connections |
| Startup – prompt user with Login GUI |  | Waits for client to connect |
| User enters values for ip & username |  |  |
| Stores ip & username – opens Chat GUI | Ip & Username are sent to Server | Server receives ip & username – detects new Client – displays in GUI |
| Client 1 is successfully connected |  | Server runs normally |
| Repeat steps for Client 2 | Again, values sent to Server | Server detecs second Client – displays in GUI |
| Clients can enter & send messages | String (message) is sent | Server filters String – sent filtered String to all Client’s Chat GUI |
| One Client presses Start game button | Game startup event | Server displays Quiz Quest game object to Client who started the game |
| After Client 1 finishes turn – game object is sent to Server | Sending Quiz Quest instance to Server | Server sends the game object to Client 2 – this goes back and forth as long the game is running. |
| Client 1 change board color | Boolean colored is switched to false & sent to server | Client 2 will have the game non-colored if Boolean was checked |
| Each game move is sent as String back to Server | Sending various types of Strings depending on action | Server receives the Strings & changes the game accordingly – these values are continuously sent back and forth to the Client which will use Quiz Quest class to update the game |

# Data used

The sprites used in the game are Pokemon sprites drawn after the in-game sprites by fans for the public domain.

Sprites were taken from: <https://archives.bulbagarden.net/wiki/Category:Generation_VI_menu_sprites>



# Data files

# None.

# Unresolved Issues

As the Server/Client connection is built to support only two players, some issues become apparent when a third, or fourth, etc. Client joins the Server. The third client, and onwards, cannot participate in chatting; however, they can start a new game and listen to the conversation between the two players playing the game. The two players will be aware of the third clients existence and connection, but they won’t be able to communicate with them, as the third Client is unable to send message back to the two players. If the third client starts a game, that game object will be sent to the two players currently playing the game. What happens then is that the two players will have multiple instances of the game open at the same time.

The game uses the client’s name as a way to send message back & forth between the Server & Client. The clientName works as a sort of identification key for a player. The game won’t play properly, if two clients have the exact same names. They can chat and join the Chat room, but the game will stop working. There are some tools in the programm that try to minimize this from happening.