Term Project: *Chat Application*

Design Document

Table of Contents

1 Introduction 2

1.1 Purpose and Scope 2

1.2 Target Audience 2

1.3 Terms and Definitions 2

2 Design Considerations 3

2.1 Constraints and Dependencies 3

2.2 Methodology 3

3 System Overview 4

4 System Architecture 5

4.1 Login Client 5

4.1.1 Login Client establishing connection 5

4.1.2 Login Client validating information 5

4.2 Server 5

4.3 Client Handler 5

5 Detailed System Design 6

5.1 Server Object 6

5.1.1 Account Manager Object 6

5.1.2 User Account Object 6

5.2 Client Handler Object 6

5.3 Login Client Object 6

# Introduction

This document is the design plan for the Chat Application program. This chat application consists of a chat server and multiple client users. The server accepts connections from clients and delivers all the message in-between the users. This is a two-way communication tool. These document breakdowns the program into multiple classes and design features.

## Purpose and Scope

The purpose of this document is to give a broad overview of the design going to be used for the Chat Application. It provides visual and brief descriptions of each component in the system. The Chat Application has been split up into subsystems that work together to form the chat. This document shows the control and flow of the program.

## Target Audience

This document directed towards developers and System Architect. It distributes the chat application into multiple subsystems and classes that work together. This document should be used as a guideline to build the Chat Application.

## Terms and Definitions

1. **User-** A client that uses the chat application.
2. **Client**- The chat handler for a user. The user interface and communication
3. **Server**- A server that accepts connections from clients and directs all communication
4. **ChatLog-** A text file record of a user’s chat history
5. **Account**- An account that stores username and password for the user.

7. **Account Manager**- Class that stores all the registered valid accounts

# 2 Design Considerations

The purpose of section is to demonstrate a way of breaking up the chat application into smaller sub systems. This section demonstrates possible ways of breaking up the components into sub classes. These designs can be modified or removed to fit developer and client needs.

## Constraints and Dependencies

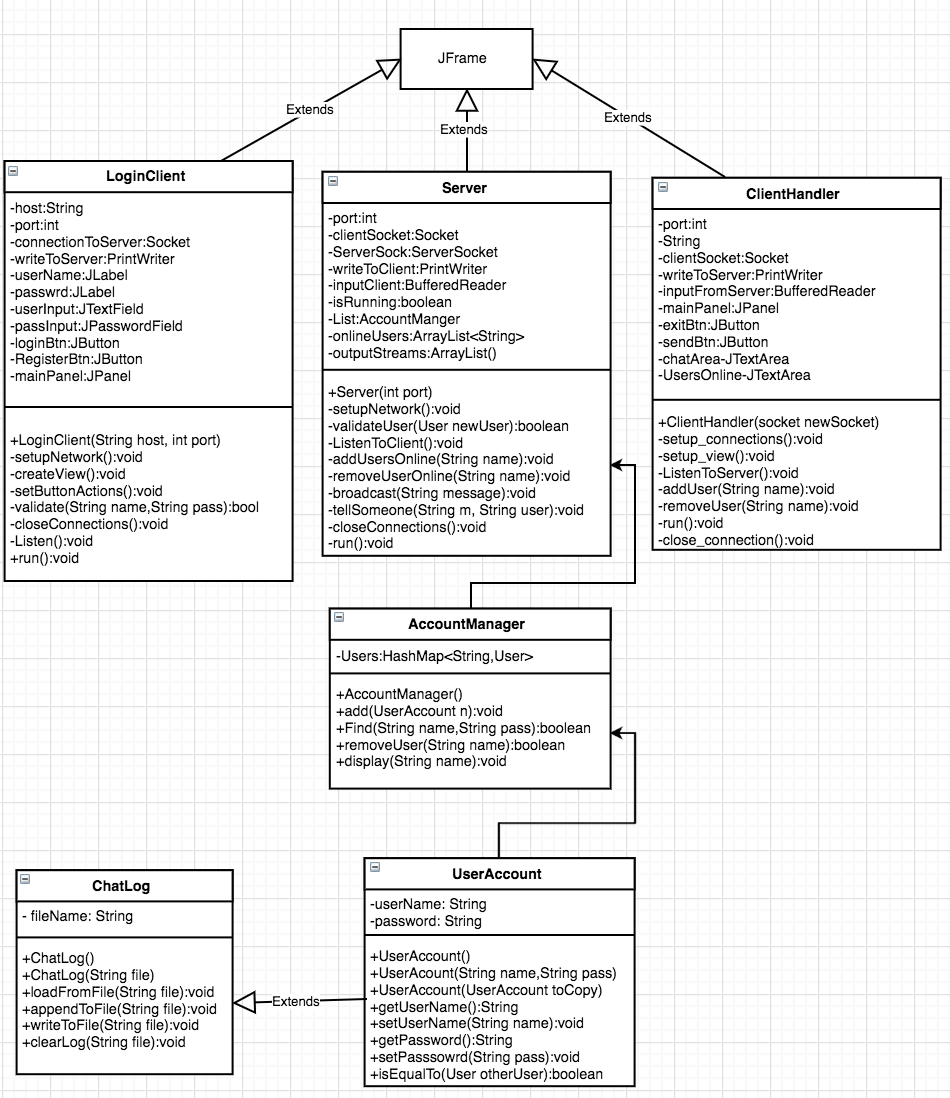
This design what created with the client/server model in mind. The chat application must be able to handle multiple users connecting and discounting from the server simultaneously. The User who logs in must have an account so the Chat Application, a type of validation has to be used in order to verify details and open access.

## Methodology

The methodology that has been selected for this design is Object Oriented with a mixture of data driven design. Object Oriented allows for the application to be split up into multiple classes and responsibilities. This makes the partition of the work easier to manage. A chat application can be split into three main components. A server that stores and writes information. A client which in this case would be a User, that communicates with the server and can read and write information to it. Lastly a Login validation component that every user has to use. These main components can easily be created into classes where each class has a specified task to do. This makes management easier and allows for use of inheritance to re-use code.

# System Overview

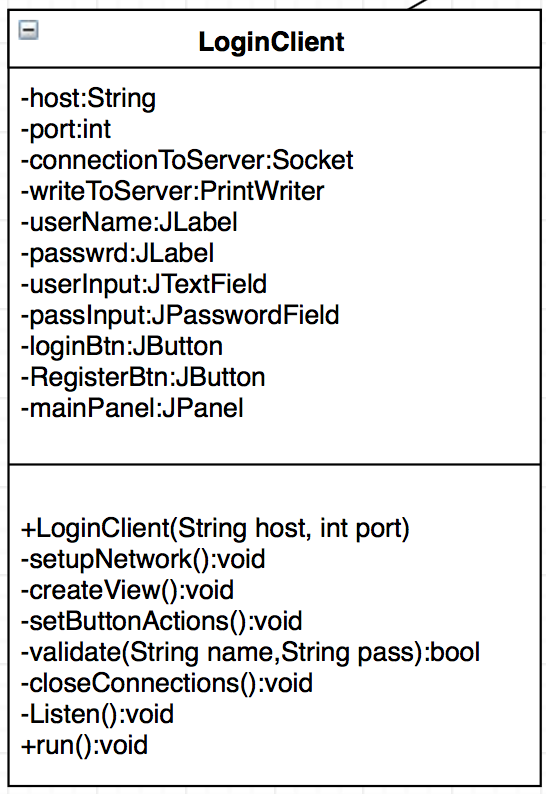
The Chat Application can be partitioned into these following classes. This will allow for easier modifications and allow for inheritance to improve design.



# System Architecture

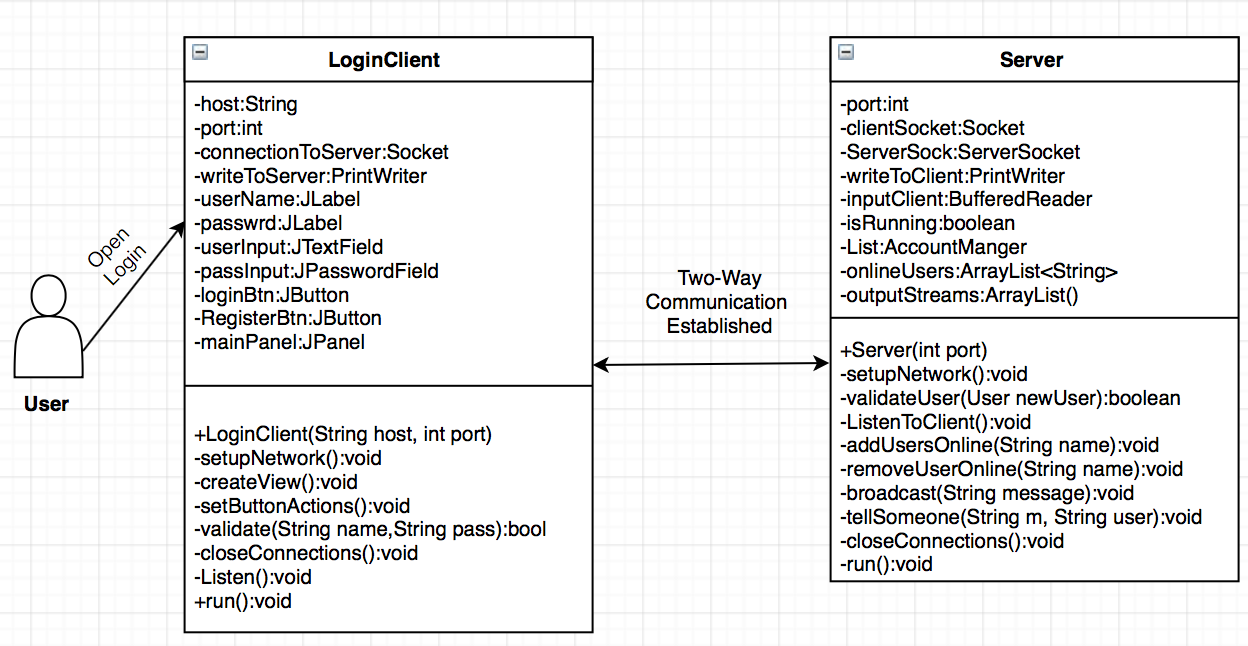
The Chat Application can be split up into the following main classes. A Login class to handle user login. Server class to handle all the networking and two-way communication. A client class that allows the user to connect to the chat server and interact in a group chat. These classes are the building blocks of the application. Smaller sub classes have been added to manager the work easier and partition certain tasks even more.

## Login Client Object

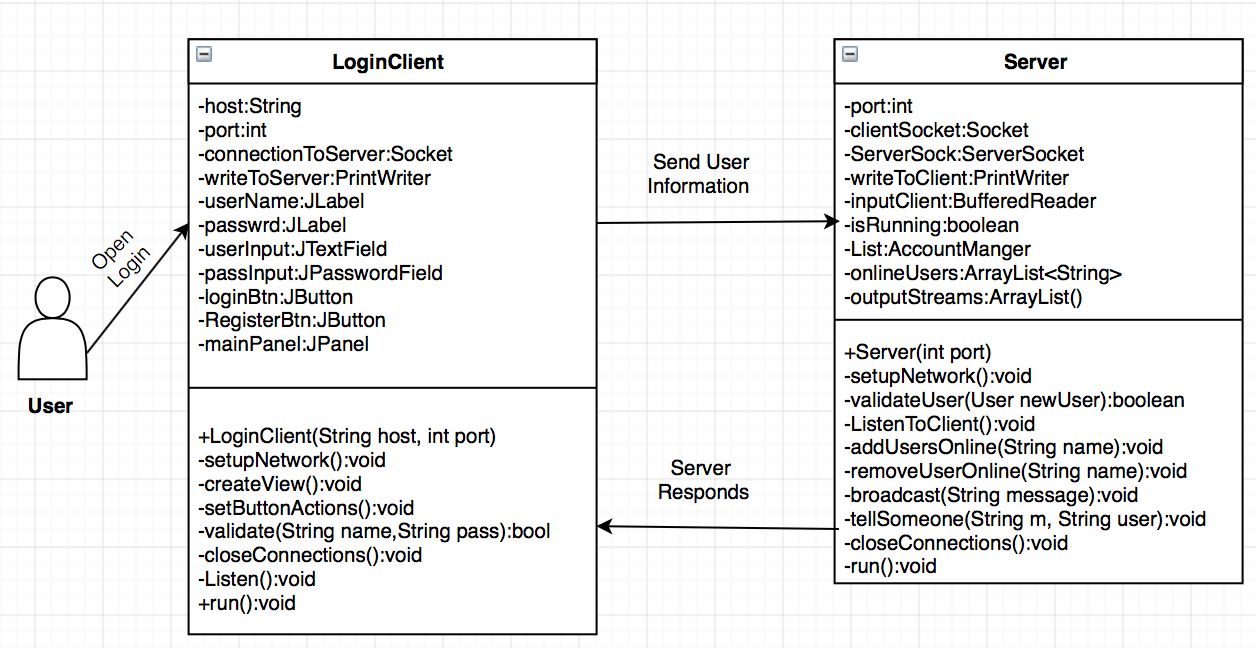


The Login Client object is one of the fundamental base classes of the Chat Application System. It is the first window the User sees when launching the Chat Application. This Login client is the Graphical User Interface the User will interact with and enter its personal details to gain access to the chat. The Login Client will first draw and create the user interface. It will consist of two input text forms for the username and password. Two buttons one for login and one for registering. The Login Client will then establish a connection with the server so it can connect to its user database. The Login client will then wait for username and password. It will then take this information and send it to the server to validate for a valid account. If the server returns valid, access will be granted and a new chat window will appear. If not valid user will have to try again or register.

### Login Client (establishing connection)



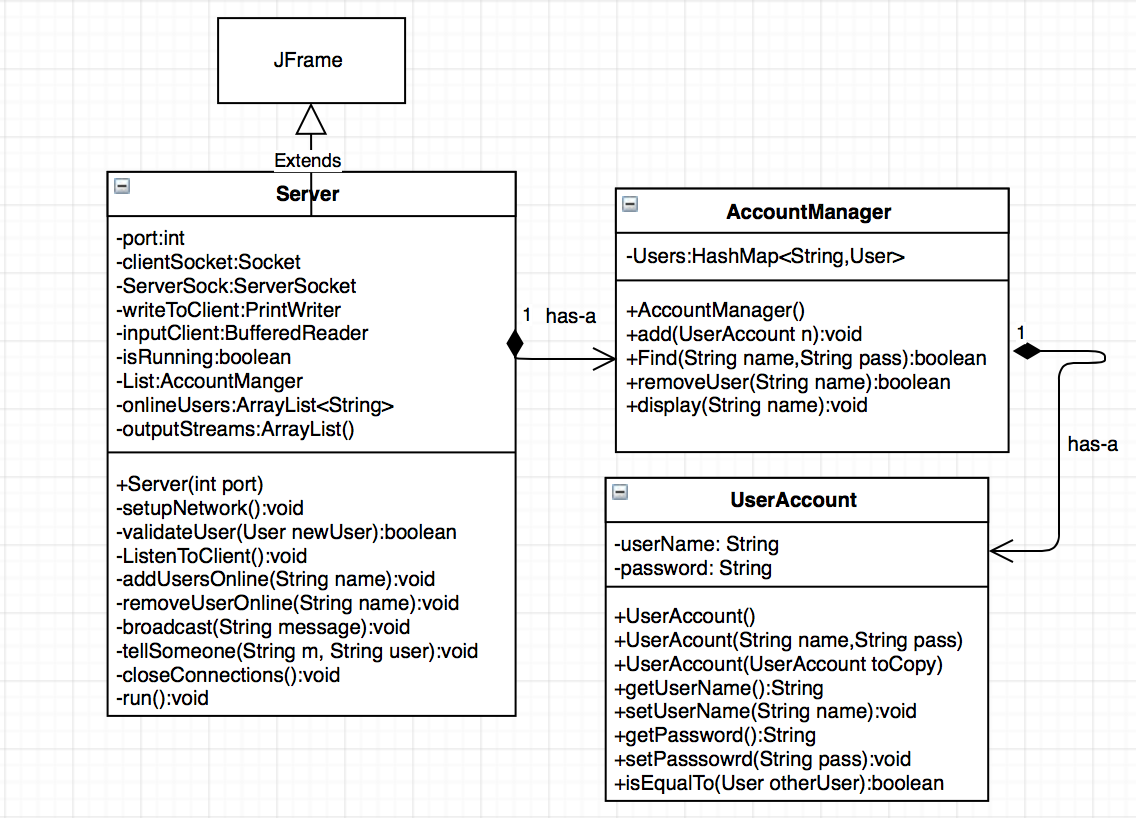
When the Login client runs, it will attempt to create a socket connection with the server. If successful it will create input and output streams that can talk to the server. This will allow the information entered to be sent to the the server for validation. The Login client will establish a connection as long as the user keeps entering information. If a valid login is made, then the Login Client will pass on the information entered and then close all its connections.



### Login Client (validating information)

After a user enters his username and password into the text fields and presses the login button, the Login Client will store that information and then take a pause. It will send the information to the Server and the Server will take charge of validating the details and then send a response back to the Login Client. If the Server returns true, access is granted and the user will be presented with a new Chat Window interface. If the Server returns false, the user will be informed with an error message and the User will have another chance to login or exit the Login Window.

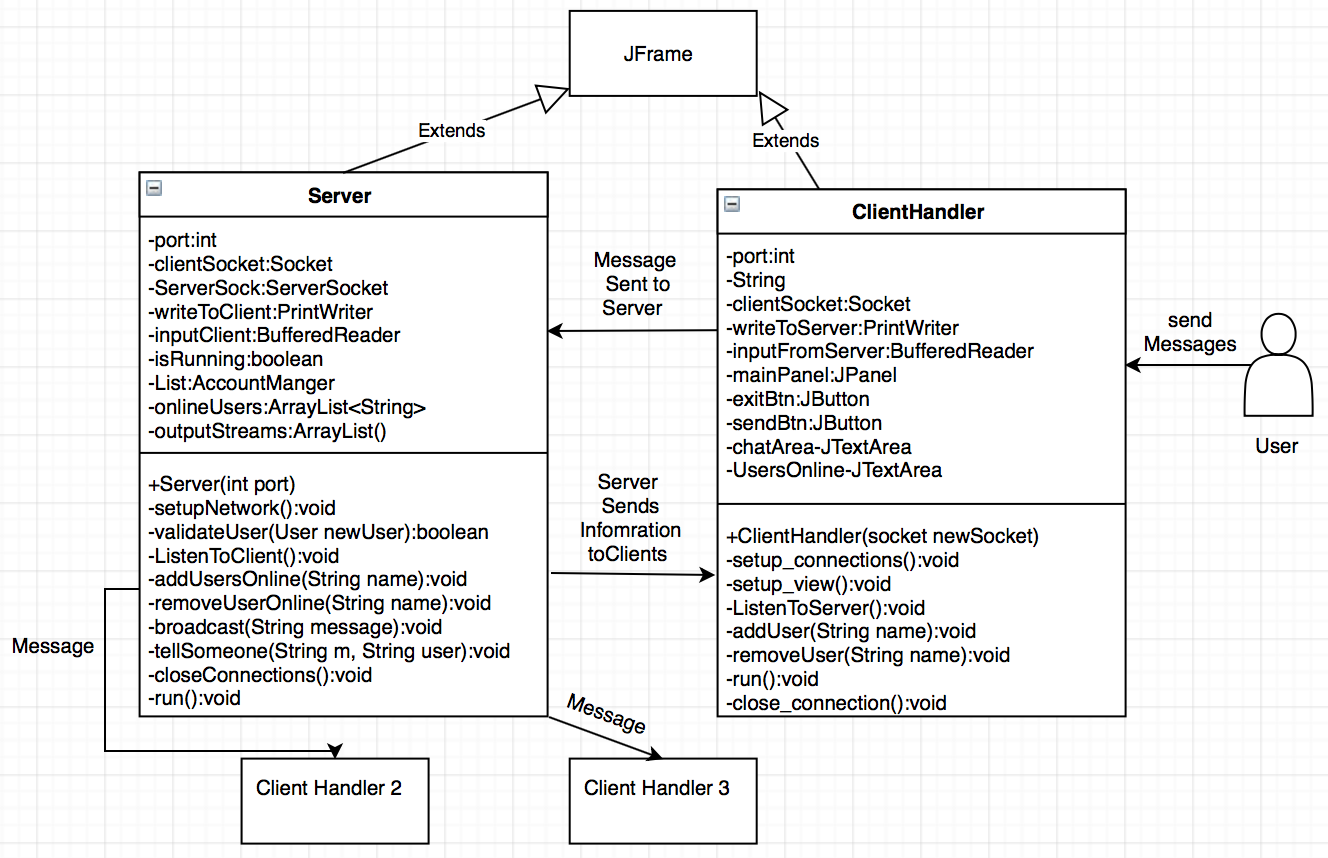
## Server Object



The Server is the center component of the whole Chat Application System. All communication and of reading and writing has to go through the Server. It is in charge of setting up the network and establishing connections with other clients. The Server class will also have containment of other classes in order to manipulate and handle data easier. An Account Manager and User Account class will take charge of all the User tasks and leave the Server doing all the communication controls.

The Server class will create a ServerSocket on a specified host and port number. This will allow the server to accept new connections from User clients that successfully login. The Server will then wait until a User connects and when it does it will take that information and validate it through the Account manger. If the information is valid It will create a new instance of the Chat Handler with the Account information provided. After it will go back to listening for new connections. The Servers second job will be to listen simultaneously for input from the Client Handlers. If a Client sends information to the Server, the Server will than broadcast those messages to all the Clients connected to it.

## Client Chat Handler



The Client Handler class is in charge of interacting with the User who logs on. Each user that logs in will get a new instance of the Chat Handler. This is the only class the User will physically be able to see while conversations take place. The Server class does not have a Graphical User Interface but will be communicating the the Client behind the scenes. The Chat Handler will consist of a window with a Text area that will hold the conversations, a list of online users on the side, a text field where users can enter their message, and two buttons. One will be used to send a message to the Server, the other will be used to access personal chat records.

When the Client Handler it will take a socket that was given from the Server. The socket information will be copied over to able to read and write to the server. A connection will be established and the Server and Client can communicate. The Client Handler will listen and wait for the User to do something. When it receives input, it will send it over to the Server which will then be in charge of distributing it back to everyone. The Server will push all the messages to each Client Handler, then the Client Handler is in charge of reading the message and displaying it in the Chat text area.

# Detailed System Design

## Server Object



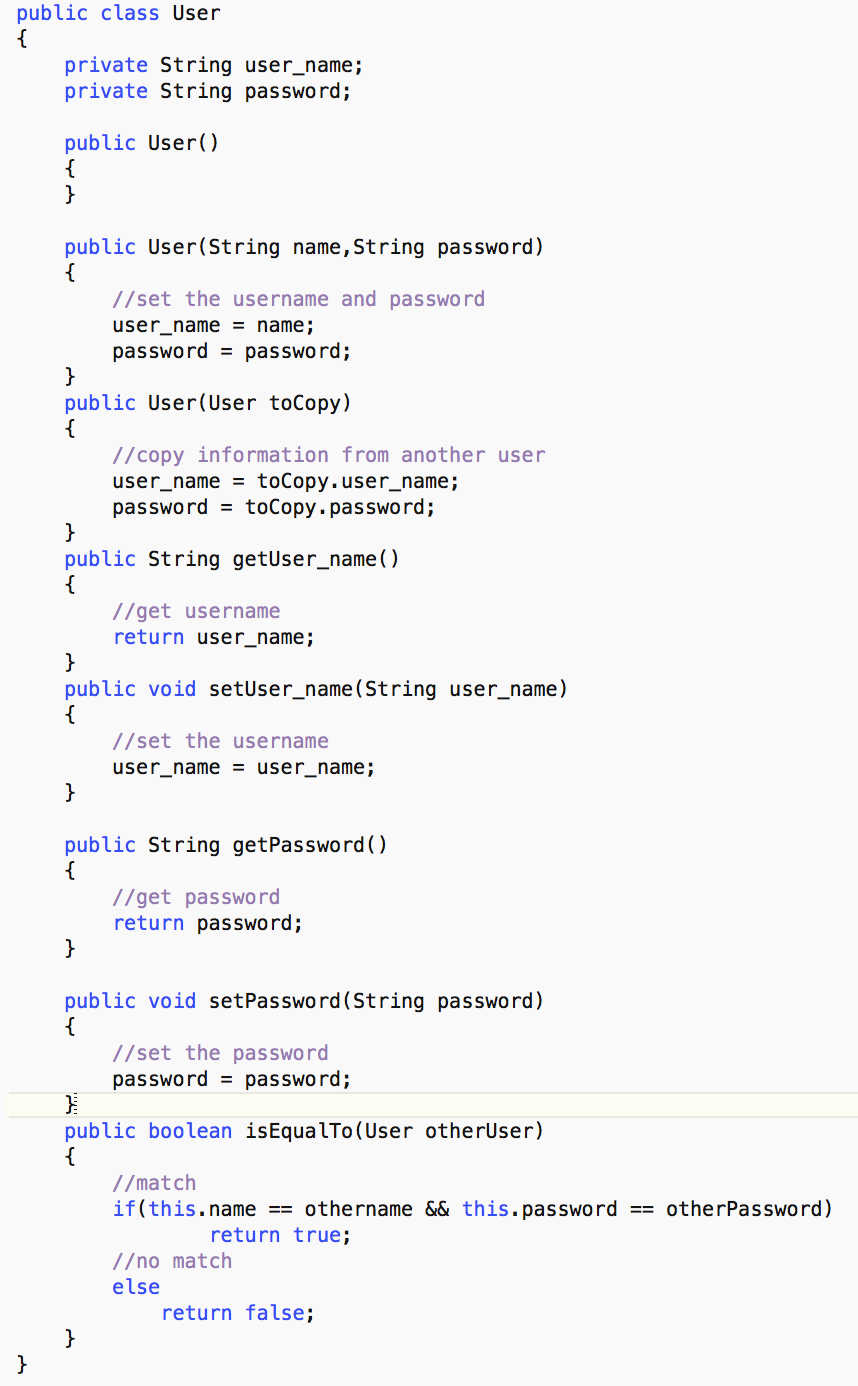
The Server object takes a integer number that will serve as the port number to connect to. The constructor will setup the port and create Account database. The setup Network function will be in charge of creating and starting the ServerSocket to listen to Users who connect. Once it receives a connection it will create a socket and forward the information. The validate function will take the user input and if valid it will run a new instance of ChatHanlder and pass on the User Account and the socket. The listening function will loop over and over again reading input from the clients and then sending them to other users. The broadcast function will loop through all the users that are connected and send the message to all of them. The Server will keep doing this until all clients leave or some connection error happens.

### Account Manager Object (Database for holding user accounts)



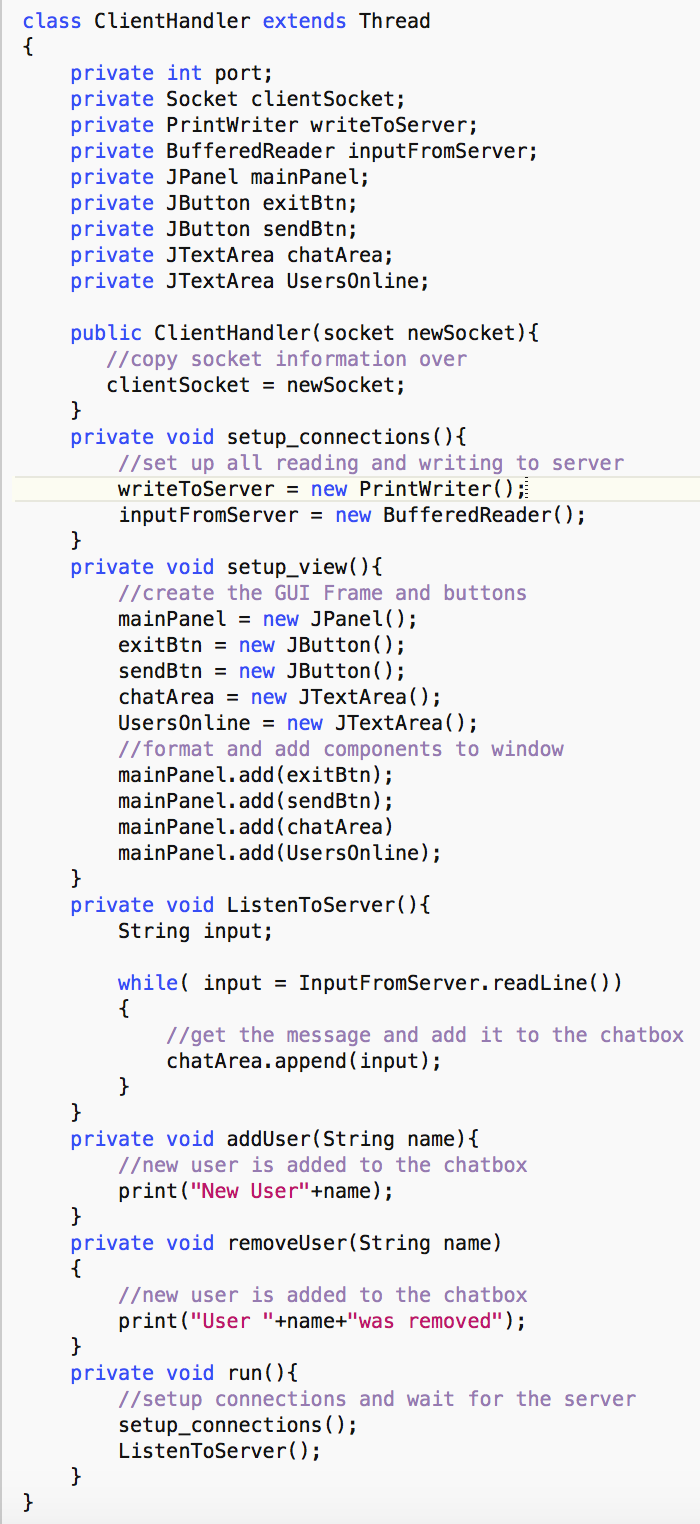
The Account Manager class is designed to store all the User Accounts together in one place. The Server class has an instance of this Account Manager class in order to keep track of all the registered users. This Manager class uses a hash map to store each User account according to a specified username. It has the ability to add, remove, and find user accounts in the system. Anything having to do related with accounts will happen through here first. By having the accounts separate on not directly created individuality on the server, this will improve performance of accessing the accounts because of the hash map data structure. Search should be around O(1) most of the time.

### User Account Object



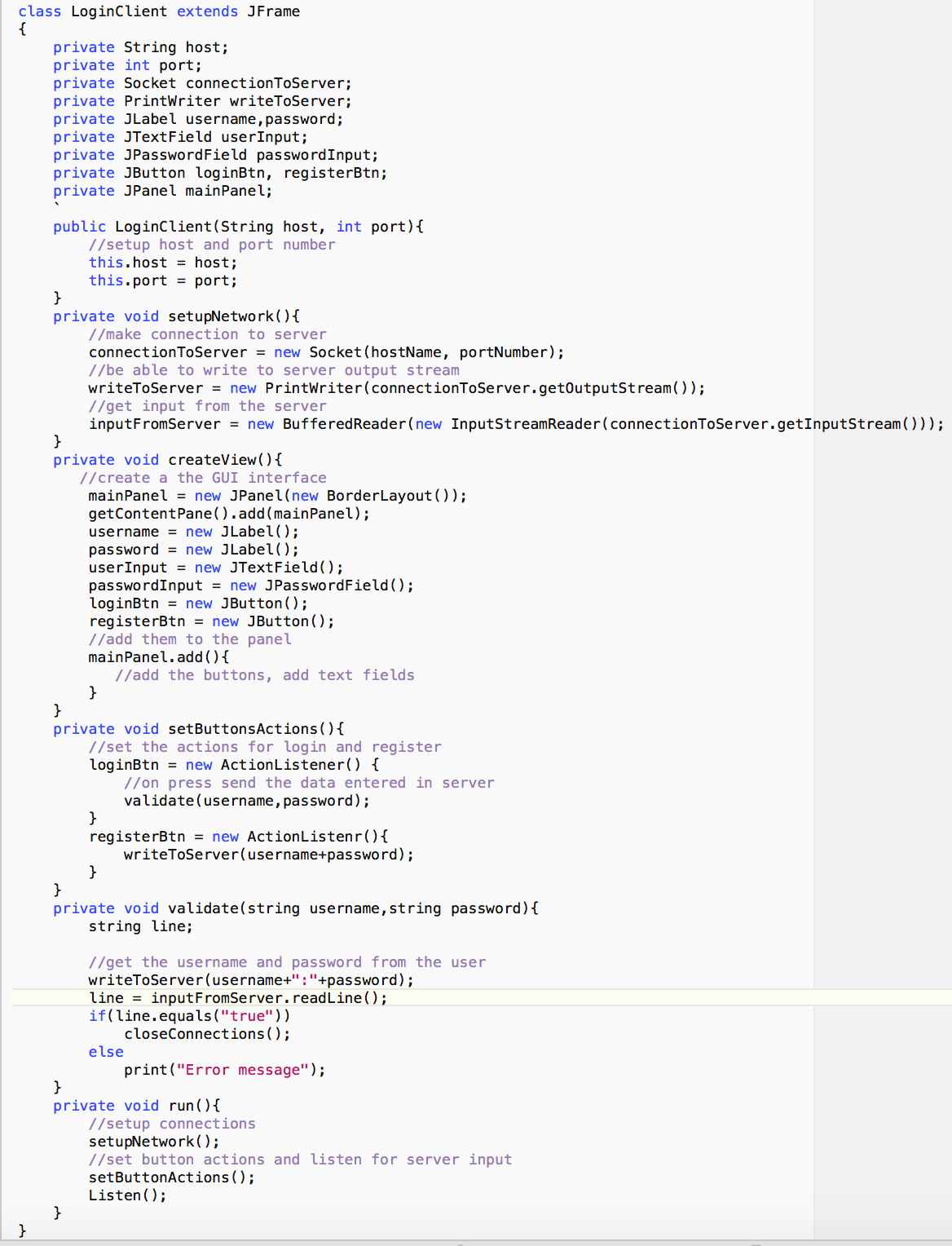
The User Account class has been created to store all the information about a user. The object stores the username and password of the client user. This class handles any changes and manipulations with the user account. It can let the user change or set its password or modify usernames. By Having a dedicated User Account class, the Server does not need to create a bunch of strings to store each user’s information. This way the Server can pass a User Object around and everything will be encapsulated in it. This class works in correlation with the Account Manager class. Together they both form the database of Users for the chat application.

## Client Handler Object



The Client Handler Object is going to be the Chat interface for the user that logs in. This is where they are going to be presented with a chat box and space to enter text. This object is going to be taking messages from the Server and adding them to its chat box. A list of online users will be presented on the side in order keep track of the order of logins. In order to connection to the Server, the client handler will need to take an Socket object as an argument in the constructor. This information will be copied over to another local socket allowing the Client Handler to communicate wit the Server. To build the interface, it will need to have one big main panel, two chat areas and two buttons. These will be position in a way where the User can click and navigate freely without interruption. The Client Handler will have a listening function that is going to constantly be running in the background waiting for new messages from the Server and then displaying them

## Login Client Object



The Login Client object is going to be in charge of logging in the user and connecting to the Server. It will provide a Graphical User Interface with two fields where the User can type in their name and password. At the Bottom of the window it will have a button to login with those details or register as a new User. The Login Client will then have to take that input and send it to the Server for validation. If the response from the Server is valid than a new Client Handler will appear and Login Client will close and save the details. Otherwise Login Client should run and listen constantly until the user attempts to login or register a new account.