FOXCHAT

Project Milestone

Mark Blankson-Hemans

TABLE OF CONTENTS

ABSTRACT

Foxchat is a Java program developed to be used by Marist students to send messages between one another within the Marist campus. Using java socket programming and the java swing class, users are able to use the lines of code from the ‘back-end’ through a GUI over a server-client connection to send texts between one another. The program was developed to save costs from text-messaging, and to also securely connect students on the Marist campus in a matter of seconds using only their unique CWID’s assigned to them.

INTRODUCTION

Foxchat was developed in order to explore the benefits of Java socket programming. The program utilizes the more expansive and user-friendly adaptation of java sockets: **stream communication**. A stream connection is also known as implementing a Transfer Control Protocol (TCP). The advantages of stream communication are that users only have to establish a connection with the server once to connect with other users. The server listens for connections from users to connect, and clients ask for connections. This enables multiple amounts of users to seamlessly simultaneously establish connections and send messages between one another. This is developed by the utilization of input and output streams, and as such the program becomes very simple and easy to implement on many devices, as only two distinct java classes need to be developed to meet the goal of the project – a server class and a client class.

DETAILED SYSTEM DESCRIPTION

Foxchat is comprised of two distinct classes, the server class to establish a connection and listen for contact, and a client class to request connection and send data to the server for it to be processed.

Foxchat is possible due to the existence of three core packages that are available in the most current java version (Version 8 Update 131). These three packages are but are not limited to:

- *Java IO*
- *Java Net*
- *Java Swing*

JAVA IO

Java IO's main goal is to facilitate the reading/writing of data in the program. This essentially takes the user's data input and transforms it into information as an output after the computer processes it. The input and output streams are features of this package and are essentially the "connection" described throughout this project. It also plays a very important role with its 'PrintWriter' function, which allows complex data manipulations to be categorized by simple textual objects. The program now listens for some key user defined-phrases and manipulates data accordingly. Not only does this make it easier for the developer to visualize the transfer of data, but it also allows an extensive addition of features to be added to the program.

Java Net

The Java Net package outlines the structure of the client-server application with the creation of a port which is the fundamental aspect of the entire program. This allows one class to 'listen' for incoming connections on a real and physical location on a computer. This feature allows a connection between two distinct entities to be established, and the Java IO now comes into play using its input / output streams. Essentially, the server's input stream is connected to the client's output stream and vice versa. This is the basis for data exchange. In this multi-user program, all clients are connected to the same output stream, and all client's outputs are sent to the same input stream in the server which shares the message, allowing a 'chat' to be created.

Java Swing

The java swing class allows the structure of the program to be represented by text boxes, buttons and other graphical assets. This makes the program user-friendly and simple. It's also an abstraction in the sense that it hides the complexity of many data transfers and only presents the textual output to users, which is all they need to see. This also removes the possibility of errors and improper manipulation.

THE SERVER

The server class is used to create the environment for clients to chat with one another. Fundamentally, all clients are, in reality, 'chatting' with the server, but the server is simply displaying these chats to all users also connected to it – it’s a bridge connecting two entities which previously did not have any link between them.

```
ServerSocket listen = new ServerSocket(PORT);
```

Using java socket programing, the server class is able to create a virtual port on the machine which is also specified in the client class. This means that the two entities are now connected by this unique ID.

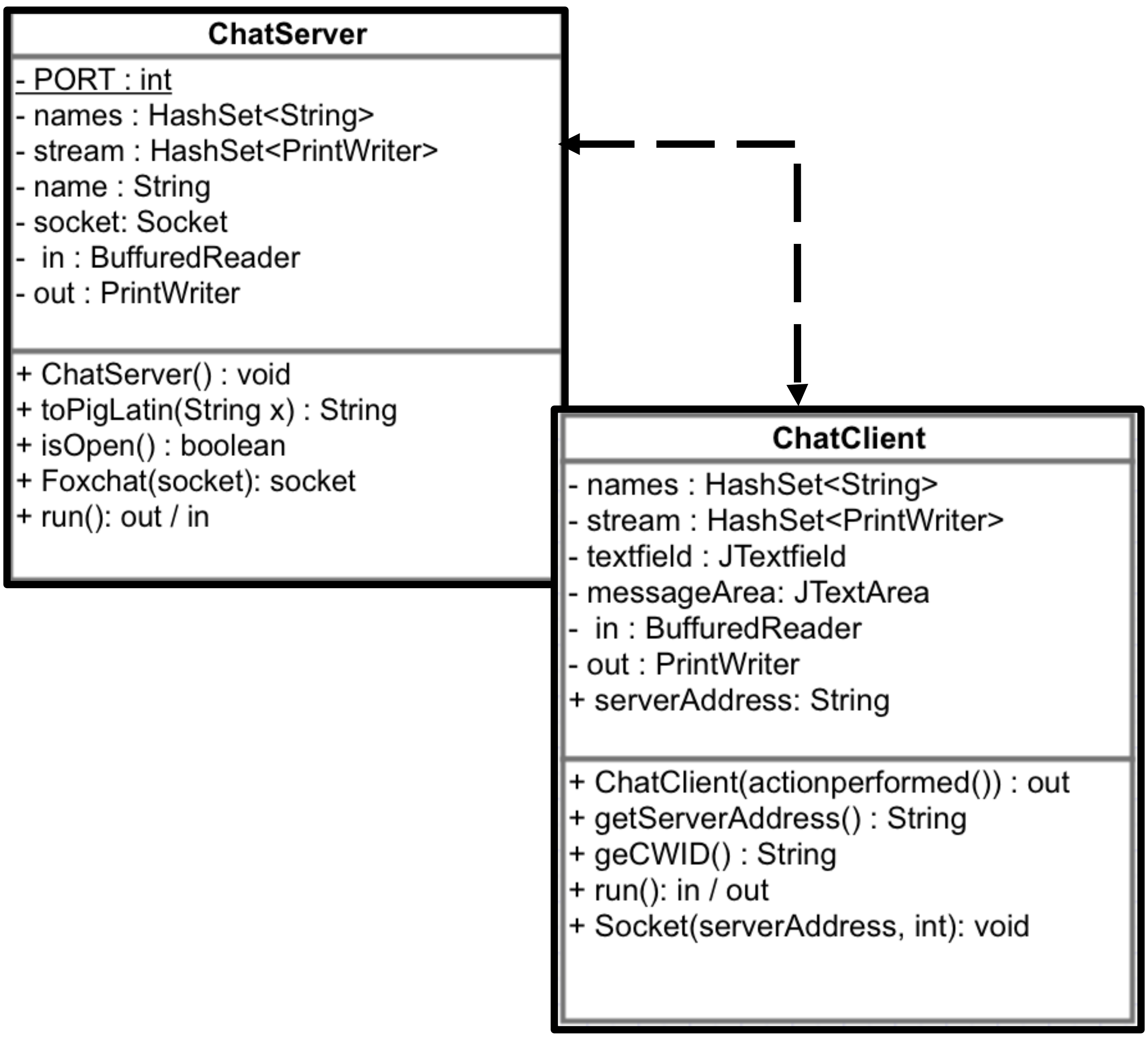
```
try {
            while (true)
            {
                new Foxchat(listen.accept()).start();
            }
        }
        finally
        {
            listen.close();
        }
```

The above code shows how the server listens for new clients to initiate connections and accepts them. In the end, the connection always has to be closed and as such, the server operates using a **try-catch** clause.

THE CLIENT

The client creates a GUI for users to interact with. Using the Java Swing class, various text Areas are generated to handle the user input and send it across the connection.

UML DIAGRAM(S)



REQUIREMENTS

Objectives	**Successes**
The system is supposed to facilitate the sending of messages between students at Marist College without the need of an internet connection.	The system is able to do this.
The ability to have chats between more than one user.	The system is able to do this.
The ability to hold more than one concurrent chat group.	To some extent, the system is able to do this, yet it requires the creation of several server classes instead of one dedicated server to process data.
The system is able to alter messages to add fun.	The system is able to do this on the server side, but implementation from the client interface was not achieved.
The system is able to run various information checks to ensure all sign up info is entered properly.	The system is able to do this, however, failure results in the program reaching an error (frozen state) and does not rollback, due to the presence of the PrintWriter feature.
The system is easy to use and is user friendly.	Achieved.

LITERATURE SURVEY

Similar experiences with the PrintWriter function have sighted the use of the flush() method, however, due to inexperience and lack of time with the project, it was unable to be used to rollback discrepancies found when data was being validated.

In the FoxChat program, the current structure uses a try-catch structure to implement the PrintWriter functions, and the printWriter function was set up to automatically flush code (commiting it to the server) using the **true** java keyword when the predefined method was called.

However, this structure can be abandoned by the implementation of if statements to validate data, with the **flush** method being implemented thereafter. An example is as follows:

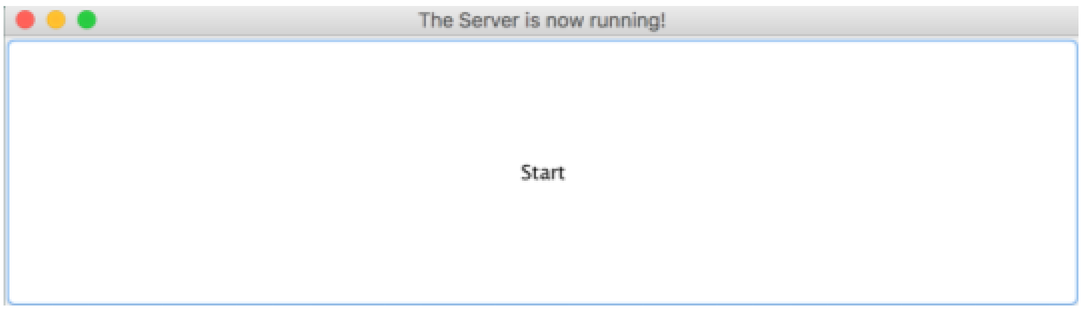
```
  try {
            PrintWriter password = new PrintWriter(pass);
            password.println("hello");
if (password.length >5)
{
            pw.flush();
            pw.close();
}

            } catch(IOException e) { }
```

USER MANUAL

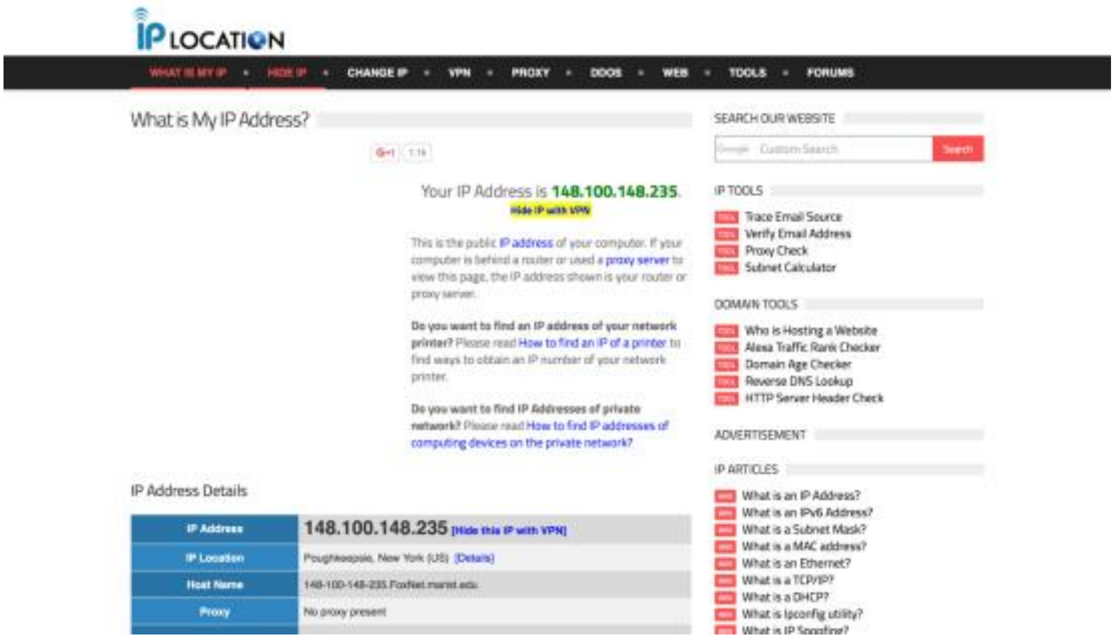
To run the program, the latest Java environment as of May 2017 needs to be installed on the computer. The computer should also have a Java Virtual Machine to process the interfaces as well as run the classes. The computer also needs to have network capabilities, both to run the server class as well as the client class.

For the current extent of this program, a computer shall be dedicated as the "server" for the application, although future developments may prove otherwise. As such, the ChatServer class should be run before any clients connect.



Running the Client program should create the first interface which confirms that the server has been successfully created with a network port predetermined.

Now, the clients who wish to chat should run their program, which can be set up as an icon on the desktop. Running the program should immediately open a browser window which displays the actual IP address of the server.


IP LOCATION
WHAT IS MY IP • HIDE IP • CHANGE IP • VPN • PROXY • DDOS • WEB • TOOLS • FORUMS

What is My IP Address?

Your IP Address is 148.100.148.235.
Hide IP with VPN

This is the public IP address of your computer. If your computer is behind a router or used a proxy server to view this page, the IP address shown is your router or proxy server.

Do you want to find an IP address of your network printer? Please read How to find an IP of a printer to find ways to obtain an IP number of your network printer.

Do you want to find IP Addresses of private network? Please read How to find IP addresses of computing devices on the private network?

IP Address Details

IP Address	148.100.148.235 [Hide this IP with VPN]
IP Location	Poughkeepsie, New York (US) [Details]
Host Name	148-100-148-235.FoxNet.marist.edu
Proxy	No proxy present

SEARCH OUR WEBSITE

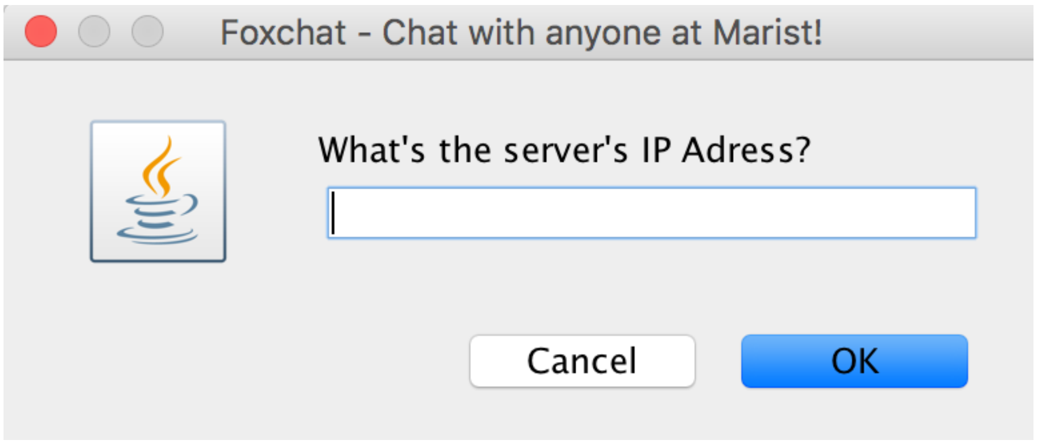
IP TOOLS
- Trace Email Source
- Verify Email Address
- Proxy Check
- Subnet Calculator

DOMAIN TOOLS
- Who is Hosting a Website
- Alexa Traffic Rank Checker
- Domain Age Checker
- Reverse DNS Lookup
- HTTP Server Header Check

ADVERTISEMENT

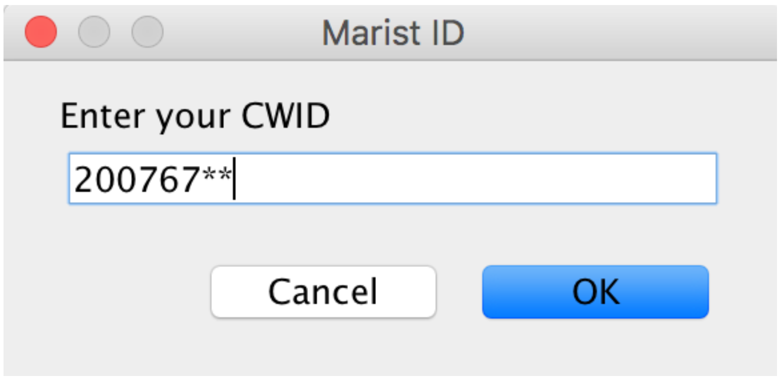
IP ARTICLES
- What is an IP Address?
- What is an IPv6 Address?
- What is a Subnet Mask?
- What is a MAC address?
- What is an Ethernet?
- What is a TCP/IP?
- What is a DHCP?
- What is Ipconfig utility?
- What is IP Spoofing?

This unique IP address should now be copied when the following prompt is displayed.



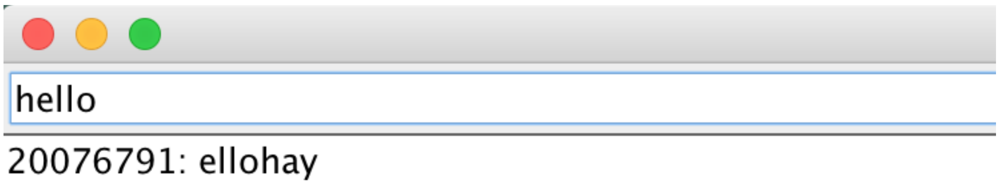
What's the server's IP Adress?

Proper input should lead to the next prompt asking for a CWID number. This would also serve as a client's "Username".



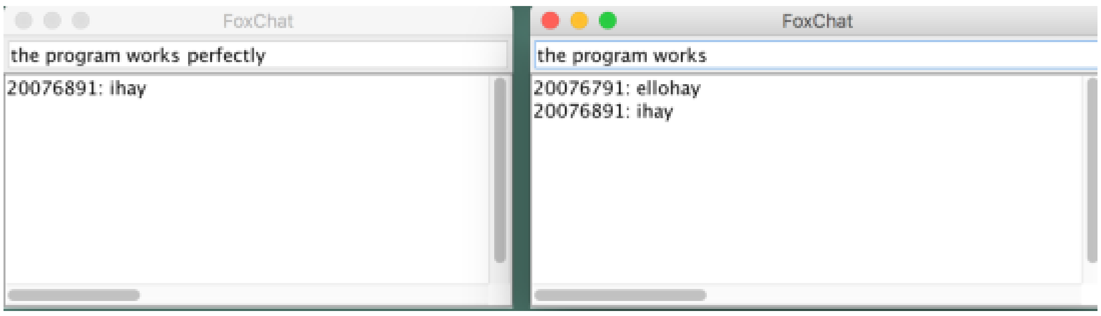
200767**

Correct input should lead to the next interface for users to begin chatting with others. The textbox at the top of the pane is used to collect input, whereas the textarea below displays the messages. Messages are transmitted through the commitment of a line, which is done by pressing the enter key.



```
hello
20076791: ellohay
```

The previous steps should be implemented by another user, or even by another clone class, and users can begin chatting!



```
the program works perfectly
20076891: ihay
```

```
the program works
20076791: ellohay
20076891: ihay
```

CONCLUSION

In conclusion, FoxChat manages to highlight the ability of java socket programming, yet, it does not emulate the complexity of some established freeware chat messengers mostly found on the web (Facebook, MSN etc.) However, the simplicity of the program and local capabilities still make it a viable option to be implemented on the Marist Campus. The program is still able to send messages between a server and a client, and this creates a lot of room for improvement. The system does not achieve the goal of privacy, because all messages are essentially unencrypted and distributed between all clients connected to the same server. Future classes will need to be developed to achieve this goal.

WORKS CITED

Oracle, Java. "Class BufferedReader." *BufferedReader (Java Platform SE 8)*. Java, 12 Jan. 2016. Web. 12 May 2017.

http://www.cs.utexas.edu/users/mitra/csSpring2004/cs313/assgn/BufferedReader.html

https://docs.oracle.com/javase/7/docs/api/java/io/PrintWriter.html

http://www.tutorialspoint.com/java/io/printwriter_flush.htm

https://www.tutorialspoint.com/java/io/index.htm