Eli Fonseca

JavaIM

5/3/16

## **Functional Overview**

My project will be an instant messenger application. The program will be a standalone application that has a GUI user can interact with. The messenger will be able to connect user on the same local network and also connect users who are not linked by a local network. My programs intended audience is high school students so I will be able to test the user interface and overall functionality of the program with my peers.

My project is in collaboration with Jason Ding’s project. Jason will be working to implement different encryption algorithms that will be incorporated into the instant messenger so all messages will be encrypted.

## **Design Overview**

Program will run like other instant messenger applications. You will have a chat history and a chat box where you can type and send new messages. The program will start off with a login screen where users will sign into or create a profile page. Here users will be able to connect to other users and chat.

The main flow of the program will be as follows. Clients will start up the standalone application which will then establish a connection to a server which will be running my server script. The server will accept the connection and then clients will be able to push messages to the server. The server will then push the messages to the intended receiver who is also connected.

## **Prioritization of Features**

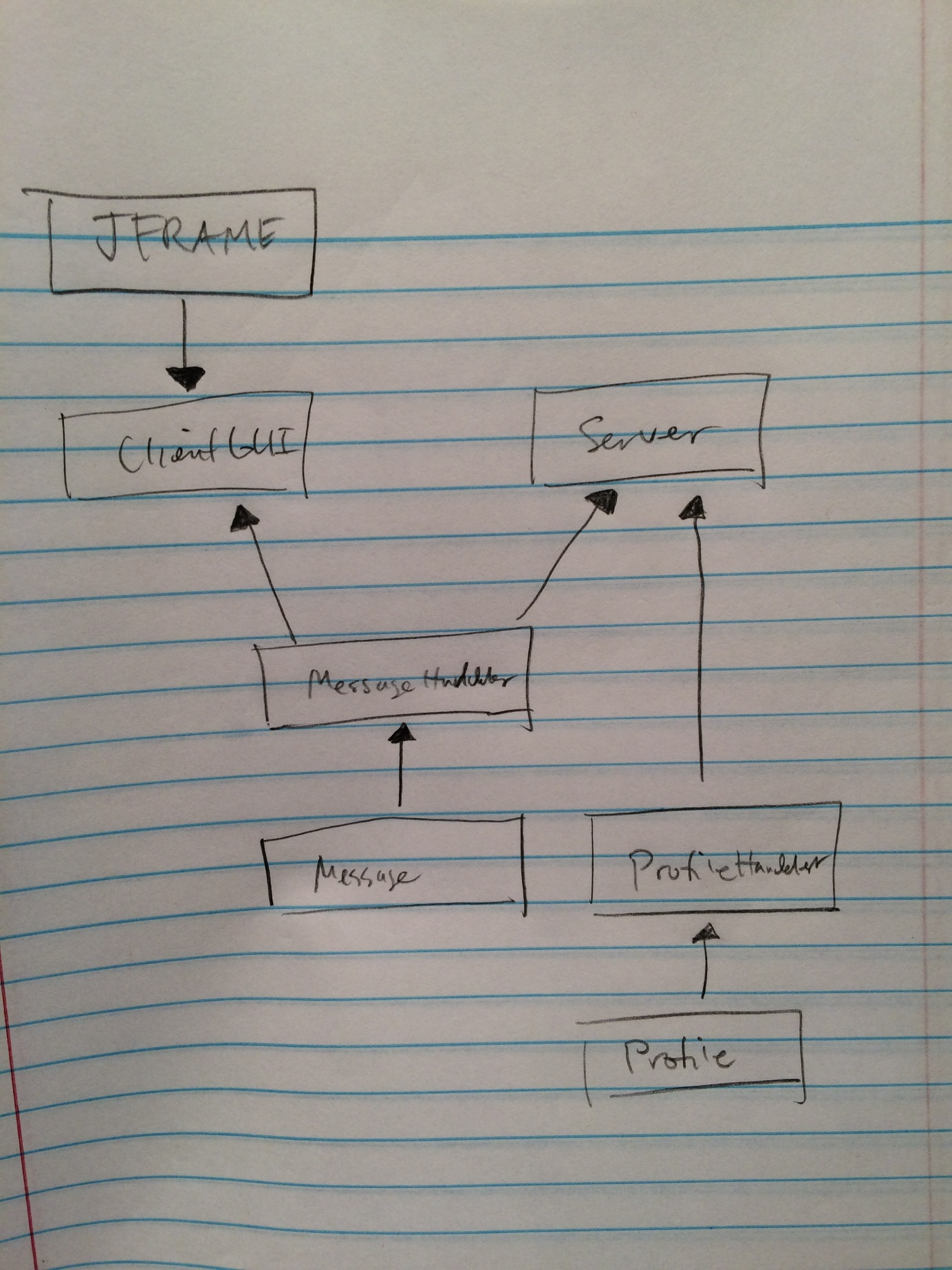
The following is a list of features in descending order of importance:

1. Connection between users not connect to the same local network
2. Allowing multiple people to chat with each other at the same time
3. A beautiful user interface that is easy to use
4. Allowing the creation of user accounts to be stored server side so that people can have profiles as to better access connection between people they had chatted with before

## **Design Details**

The instant messenger will be built out using java. I will be using sublime text and jgrasp for development. Eventually if the project works a server to host the scripts running the backend of the instant messenger will be necessary so that people can actually use the instant messenger. My program will be using TCP protocol, using java Sockets, for connection between clients.

The way the messenger will work is simple in theory. The client sends a message to the server, then the server processes the message and sends it off to the receiving client. I have created a rough outline of what I expect my class hierarchy for this project will look like:



## **Pseudocode/code segments**

The general structure for each class will be as follows:

ClientGUI class

The client gui will be a graphical user interface built with either swing or perhaps javaFX. A picture of what the gui should look like is below.

MessageHandler class Pseudocode:

private ObjectOutputStream output;

private ObjectInputStream input;

private String connectIP;

private Socket connection;

private int port;

public MessageHandler( serverIP, portNum)

assigns fields appropriate values

public void connectToServer()

try to open port to computer hosting the server scripts

public void setUpStreams()

Precondition: Socket connection to server or client must me initiated

sets up output and input streams to the server

or sets up output and input streams to the client who connected to the server

public Object readMessage()

Precondition: streams must be set up

reads a message from the objectstream this handler is connected to

public void send()

Precondition: streams must be set up

sends a message along the steams which have been set up

Message class Pseudocode:

private final InetAddress recipient;

private final InetAddress sender;

private final String message;

public Message(message, senderIP, receiverIP)

assign appropriate values to fields

class also includes associated accessor methods

Server class Pseudocode:

Map<InetAddress, MessageHandler> connections;

public server()

While(true)

Wait for connection from client

Accept connection and create new thread to handle connection

Runnable handling every connection

Add MessageHandler for this connection

Put the messageHandler and the ip address of the client who is connected it a Map of connections

while(this thread is not interrupted)

Push any messages the client sends to their intended recipient

If the client end the connection interrupt the thread handling the connection

Remove the connection this runnable was handing from the Map of connections

## **Testing**

* To test the program, I will create rigorous unit tests for each functional element of each class
  + The most important feature to test will be the MessageHandeler class
  + To test this, I will write several test programs with try to send different objects through the input and output streams checking to ensure data can be passed safely through my MessageHandeler
  + The other important feature to test will be the multiple client case. To test this, I will create several client programs that will attempt to connect to the server and chat at the same time.
* For this program the main issues will most likely be passing data between client and server in an efficient and correct manner.
* How will you test with other users?
  + I will use user testing to test the appearance of the GUI along with how smoothly actually use of the instant messenger is
  + Once the core functionality is finish I can distribute the program to some of my friends for testing. I want make sure that it is easy to use so I will ask them the following questions:
    1. What was the best thing about the GUI? What was the worst?
    2. How do you send a message with this application?
    3. How would you describe what this program can do?
    4. What is one suggestion you have to make this program better?
    5. Would you actually use this program?

## **Grading Rubric**

Write your own grading rubric (out of 40 points) that takes into account whether

1. Functionality
   1. Does the IM allow users to communicate on the local network? 8 points
      1. 0 == More than one major error
      2. 2 == one major error
      3. 4 == Some exceptions not handled, but they were difficult to raise
      4. 6 == a number of minor problems
      5. 8 == no errors
   2. Does the IM allow users to communicate when not connected to a local network? 8 points
      1. 0 == More than one major error
      2. 2 == one major error
      3. 4 == Some exceptions not handled, but they were difficult to raise
      4. 6 == a number of minor problems
      5. 8 == no errors
   3. Does the IM allow multiple users to communicate with each other in a group chat environment? 5 points?
      1. 0 == More than one major error
      2. 3 == Reasonable attempt at this feature
      3. 4 == no error found
2. Program Design
   1. Structure 5 points
      1. 0 == unacceptable structure
      2. 2 == poorly structured
      3. 4 == reasonably well structured but uses many global variables or magic numbers
      4. 5 == all small, coherent, independent modules unless well justified
   2. GUI design 5 points
      1. 0 == unacceptable visual appearance
      2. 2 == simple bland gui that gets the job done some features may not work
      3. 5 == appealing to the eye and all functions work
3. Style and Documentation 9 points
   1. 0 == unacceptable; no comments, no indentation, etc.
   2. 2 == difficult to read and understand, few comments, etc
   3. 4 == generally good, but major comments missing
   4. 6 == generally good, but several imperfections
   5. 8 == one or two small imperfections, improper indentation, a poorly chosen identifier
   6. 9 == perfect

## **Proposed Implementation Schedule**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **M** | **T** | **W** | **T** | **F** |
| 5/2 -5/6 | N/A | AP Exam | Start writing the bare bone of the ClinetGUI class |  | Have the bare bones of the ClientGUI class finished just core functionality at this point nothing the looks the greatest |
| 5/9 -5/13 | Have MessageHandeler class, Message class, and assocaiated unit test finished | Complete last touch ups to MessageHandeler based of the results of the testers | Start learning about multithreading so I can support multiple connections at the same time |  | Start writing the Server class with the knowledge how to support multiple clients |
| 5/16 - 5/20 | Have the Server class finished, and start work on making the ClientGUI look nice | Start user testing with current version of instant messanger while continuing to work on the ClientGUI | Continue user testing | Have the ClientGUI class finished accounting for input from user testing | Work on creating my presentation and demo |
| 5/23 - 5/27 | Be ready to present | Last minute touch ups to code, finishing off javadocs | ·        Presentations | Continue to polish demo and presentation if I have not gone yet | Presentations |
| ·        Code due 5pm |
| 5/30-6/1 | Memorial Day |  | Presentations |  |  |

## **Potential Showstoppers**

* I see a big problem in my project trying to successfully navigate NAT which is employed on almost every private network. To get around this I am creating the serve class so clients behind NAT will be able to establish a connection with the server but I am not sure if I will be able to send messages from the server back to the clients successfully. This will also be hard to test because my home network has NAT and I am not sure how to put my server code in a place where it itself will not be effected by NAT.

## **Open Questions**

* No other questions at this point.

## **Resources**

I am relying heavily on Javadoc to understand TCP protocol and multithreading within java.