9869 Course Project Report

Chatting Application in Java

Author: Xiaofei Wang

Student ID: 201294147

Date: Friday, July 31, 2015

1. Introduction

In a networked world, it is common used to share resources among multiple users. Therefore applications which designed to be deployed in a network environment should be considered and designed to serve multiple users requests at the same time or simultaneously. For example, it is common to see even desktop and laptop computers with an ability to process multiple tasks concurrently. To meet these requirements, the programming languages and the operating system are designed to provide an ability which can support the development of applications containing multiple activities that can be processed concurrently. In Java, this ability is realized by using multithreading techniques and these techniques include two significant concepts which are process and thread.

A process is an execution of a program and a thread is a single execution of work within the process. One process can obtain one or more threads and a thread is also called light weight process. In Java, a process is run independently from other processes in a JVM and threads in a JVM share the heap which belongs to that process. That is the reason for why several threads access the same object in the heap. Threads share the heap and have their own stack space. Therefore, an invocation of a method and its local variables are kept thread safe from other threads, while heap is not thread safe and must be synchronized for multithread programs. The major difference between a thread and a process is a process could have many threads and a thread may be assumed as a subset of a process. The following picture shows this difference and also indicates how the thread and process to work in the Java environment.

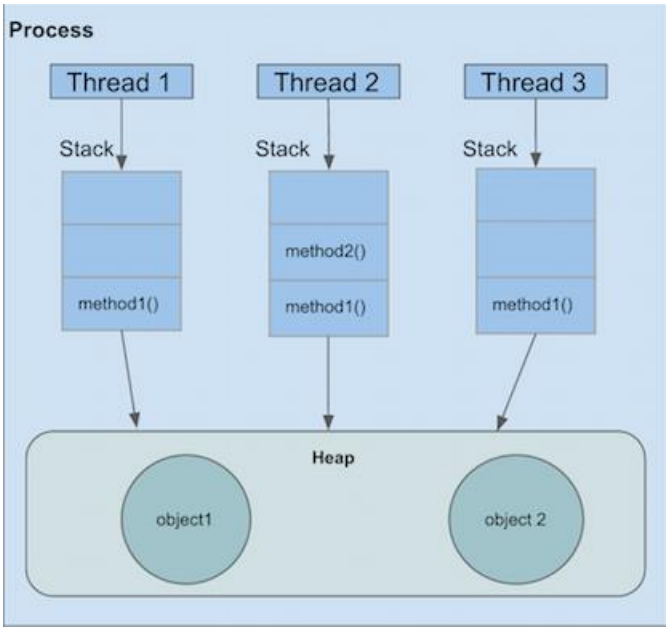

Figure 1: Details of the data flow of thread and process in Java environment [1]

For instance, if two application are run on a computer like Word and Facebook, two processes are created and multitasking of two or more processes is called process based multitasking. Similarly, multitasking of two or more threads is called thread based

multitasking. The difference between these two kind of multitasking is the process based multitasking is common controlled by the operating system and the thread based multitasking is controlled by the programmer. In programming area, thread based multitasking is often refers to multithreading.

In this project, I choose a chatting room application which contains a server to simulate process multiple requests at the same time which means the server can connect many clients and process requests come from these clients at the same time. The programming language for this application is Java and this application provides a friendly graphical user interface, either.

2. Background knowledge of Java multiple thread programming

2.1 Thread in Java

2.1.1 Thread states

First of all, a thread have different states in its life cycle. The following picture displays these states in details.

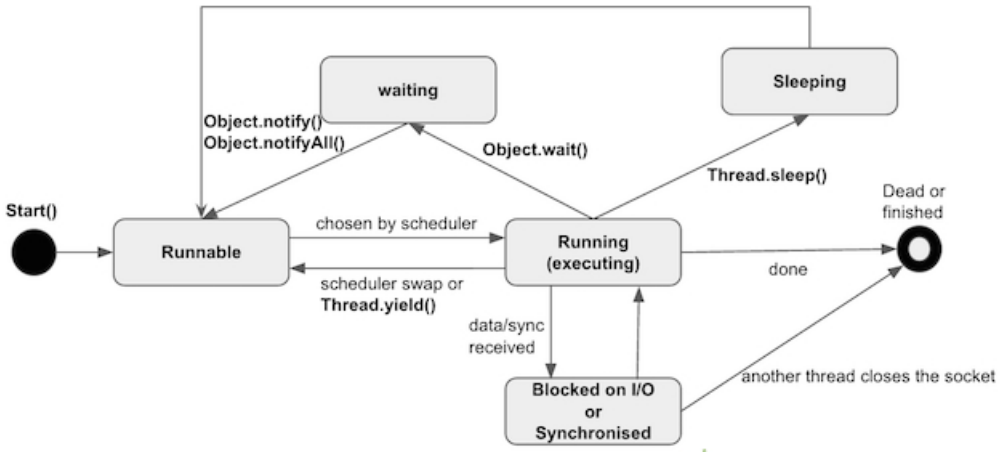

Figure 2: Flow chart for a thread in its life cycle [1]

As shown in this diagram, a thread have five states during its whole life cycle. These states include runnable state, running state, waiting state, sleeping state and blocked state. Runnable state means this thread is created and wait to execute. The execution order or schedule for each thread is mainly depend on the priority and mainly decided by the operating system. If a thread is chosen by scheduler and start to execute which means this thread enter a new state which is called running state. During a thread execution time, it may need a value come from another thread. For example, thread A need a value from thread B, however this value needs to execute thread B first. If this situation occurs, the operating system will dispatches thread B into the CPU to replace thread A, at this time thread A enters a new state which called waiting state and thread B enters running state. When the thread B completes execution and generate the value which the thread A request, the operating system will dispatches thread A into the CPU again to replace thread B similarly. Due to the thread A not complete its execution when it replaced by the thread B, therefore, save the process state for thread A is important.

We often calls this process state by context and context exchange will occurs when the operating system dispatch a new thread into CPU to replace the old thread before this thread complete its execution. Because of context switching need to occupy a large number of system resource, there also exist an optimize mechanism which provide by the programmer or the operating system. Sometimes, if a thread need to get a value from user input or read a part of content from a file that exist in the hard disk to continue its execution, we call this situation as wait I/O operation. That means the current thread enters a new state which called blocked state. In this state, the current thread must to wait until the related I/O operation completes. Similarly, the sleeping state for a thread means the programmer forces to pause the execution of a thread for a period of time like serval time slots by use sleep method which provide by java language. The difference between blocked state and sleep state is the blocked state can returns to running state directly but the sleep state only return to runnable state to wait new schedule. These five states constitute the state diagram for a thread and I will introduce how to implement a thread and how to control thread state in Java language.

2.1.2 Advantages of multithreading

Compare with the single-thread application and multiple-thread application, the latter one have some obvious advantages. First of all, multiple-thread application can increases the system performance due to divide a large task into some small tasks and processes these tasks at the same time. Secondly, multiple-thread application can improve the responsiveness which means multiple-thread technique can allow an application to remain responsive to user input. For instance, in a single-thread program, if the main execution thread blocks on a long running task, the whole application can appear to block. Through moving this long running task to another worker thread which runs concurrently with the main execution thread, it is possible to remain responsive to input while execution tasks in the background. Thirdly, a multiple-thread application can process a large number of requests in one clock cycle and this feature is most significant to many famous instant messaging software like Facebook, twitter and Instagram. All these kind of software have a same feature which is need to process a huge number of clients connect requests at the same time and multiple-thread can provide this ability to meet this goal. On the other hand, includes the multiple-thread technique can decreases the cost of system obviously due to context exchange between threads is inexpensive than complete context exchange between processes.

2.1.3 Implement thread in Java

This project is coding by Java and the thread is object in the Java language. Threads can be created through two different methods:

(1) Create a class which extends the Thread class as the Java language have defined
(2) Create a class which implements the runnable interface as the Java language have defined, either.

A thread can be defined through extends the Thread class or implements the runnable interface. In this case, the run method should be overridden and code which contains in this method will be executed by the new thread. The modifier for this method must be

public and the return type must be void either. The following figures indicates how to implements these two mechanisms in the real Java code.

a. Extends the Thread class

```
class SingleThread extends Thread
{
   public void run()
   {
      System.out.println("Single Thread");
   }
}

public static void main(String[] args)
{
     SingleThread thread = new SingleThread();
     thread.start();
}
```

Figure 3: Java code of extends the thread class [1]

When a class extends Thread class, it should be override the run method and provide its own implementation of the run method. The start method use to start the execution of this thread. When start to run the SingleThread, the start method needs to be called and the JVM will calls the run method of this thread which has been overridden by the programmer. This thread will be alive until the execution of the run method is stopped of completed

b. Implements the Runnable interface

```
class RunnableThread implements Runnable
{
    public void run()
    {
       System.out.println("Runnable Thread");
    }
}

public static void main(String[] args)
{
    Thread t = new Thread(new RunnableThread());
    t.start();
}
```

Figure 4: Java code of implements the runnable interface [1]

As shown in this picture, if a class implements runnable interface, it should provide its own implementation of method (override). Due to the runnable interface doesn't have start method, we need to create a new instance of Thread class to start the new thread and execute the run method.

2.1.3 Implement multiple-thread in Java

The following code example indicates how to implement multiple-threads in Java.

```
class RunnableThread implements Runnable
{
   @Override
   public void run() {
       try {
           System.out.println("Runnable Thread:"+
              Thread.currentThread().getName());
           Thread.sleep(5000);
       } catch (InterruptedException e) {
           System.out.println(e.getMessage());
       }
   }
}
public static void main(String[] args)
{
   Thread t1 = new Thread(new RunnableThread());
   Thread t2 = new Thread(new RunnableThread());
   SingleThread t3 = new SingleThread();

   t1.start();
   t2.start();
   try {
       t2.join();
   } catch (InterruptedException e) {
       System.out.println(e.getMessage());
   }
   t3.start();
}
```

Figure 5: Java code of implements multiple-thread [1]

As shown in this picture, there exist three threads. For the RunnableThread, sleep method is called to suspend this thread for a period of time which means force to stop the execution of this period. In this example, thread t2 is started after thread t1 is suspended. Similarly, thread t2 called join method to stop the current thread to not execute until t2 is completed, then, t3 will be executed once thread t2 is finished.

2.2 Message passing by socket in Java

2.2.1 Socket in Server-Client model

Socket is a significant concept in thread communication section. Programmer often use the socket to describe IP address and port number. Socket is a handle of communication link and application often use socket to send request to Internet or response request from Internet. Java provide a specialized class library to support communication through socket which called Socket class and ServerSocket class. These class libraries are exist in the Java.net package. In general, programmer use ServerSocket on the server side and use Socket to build a connection. If connects successfully, both the server side and client side will generate a new Socket object and complete related session through this socket object. For a network connection, no matter the socket object exist on the server side or on the client side, they are on the same level for the application and without difference. Socket class and ServerSocket class implement their

work through the SocketImpl class and its subclass. The following list introduce three methods which are used most frequently.

(1) Accept method: this method used to generate ‘block’ until accept a new connection and return a socket object on the client side. Block in this case means the program is stopped to run in this point for a period of time and is started to process again when a new session generate. In general, this kind of block is generate by loop.
(2) GetInputStream method: this method used to obtain the input from network connect input and return an instance of InputStream object.
(3) GetOutputStream method: this method make another side of the connection also obtain an input and also return an instance of OutputStream object.
(4) Both the getInputStream and getOutputStream method will generate an exception which called IOException and this kind of exception must be catch due to the return flow object of these two methods will used in another flow object as usual.

In this project, I imitate a Server-Client model to implement the message passing between the server side and client side through the socket. The transmission in this model is based on the TCP/IP protocol. The following picture shows the structure of this model in details.

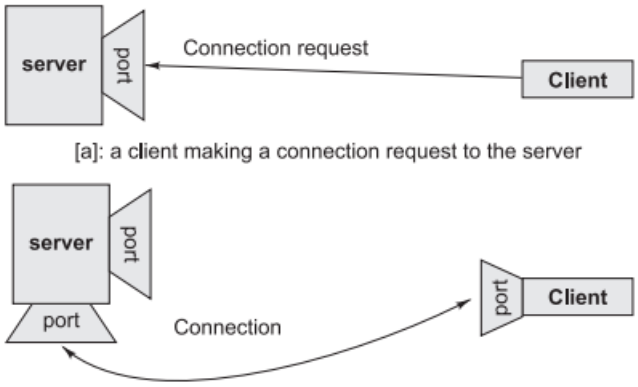

Figure 6: Session established by temporary ports used for two way communication [2]

As shown in this picture, on the server side, server use the ServerSocket to monitor the specific port and wait the request which ask to connect from client side. Due to the port which number smaller than 1024 is the system remain port and this kind of port is not be permitted to use in some operating system, therefore, advice to choose the port which number is larger than 1024. Moreover, on the client side, if client use the Socket to send connect request to one port on the server side and this connect request is completed successfully, a new session will be built and this socket will be closed. In general, client not to choose a specific port for this session and always distributes a port dynamically and temporarily

2.2.2 Socket programming in Java

In this section, I will use Java code to indicate how to implement a serve and a client in Java step by step.

For the server side, there exist five steps to implement a server in Java [2]

Step_1: open the ServerSocket and instantiate a new ServerSocket object
Step_2: wait for the connect request from the client
Step_3: create input and output data flow for communicating to the client
Step_4: process the communication with client
Step_5: close the socket and data flow

```
import java.net.*;
import java.io.*;

public class SimpleServer
{
   public static void main(String args[]) throws IOException
  {
      ServerSocket s = new ServerSocket(1254);
      Socket s1=s.accept();
      OutputStream s1out = s1.getOutputStream();
      DataOutputStream dos = new DataOutputStream (s1out);
      dos.writeUTF("Hi there");
      dos.close();
      s1out.close();
      s1.close();
  }
}
```

Figure 7: A simple server program in Java

This part of program build a server and this server still monitor the port which number is 1254 to wait the user to connect. When the connection is built, the server will send a message to the client which is “Hi there”.

For the client side, there exist four steps to implement a client in Java [2]
Step_1: instantiate a new Socket object
Step_2: create the I/O data streams for communication with the server
Step_3: process the I/O or communication with the server
Step_4: close the socket and data flow when the communication is done

```
import java.net.*;
import java.io.*;
public class SimpleClient
{
   public static void main(String args[]) throws IOException
   {
     Socket s1 = new Socket(“localhost”,1254);
     InputStream s1In = s1.getInputStream();
     DataInputStream dis = new DataInputStream(s1In);
     String st = new String (dis.readUTF());
     System.out.println(st);
     dis.close();
     s1In.close();
     s1.close();
   }
}
```

Figure 8: A simple client program in Java

This part of code build a client and try to connect the server. When the connection is built, the client will receives the message from the server and prints this message in the client side. If required, both the client and server program can run on the same machine. It is important to say that once the server program execution is started, it is not possible for any other server program to run on the same port until the current one is stopped. Port numbers are a mutually exclusive resource and cannot be shared among different processes at the same time.

2.3 Graphical user interface in Java

Swing library is an official Java GUI toolkit and used to create Graphical user interfaces with Java. This toolkit is totally written by Java and not based on a specific platform which make this toolkit can used in different platform such as Windows, Linux and MacOS. Use this toolkit, programmer can provide many visual gadgets for information entry such as text fields, buttons, scroll bars and menus to user and make the user obtain a better user experience. The following list shows the main features of this toolkit: [3]

(1). platform independent
(2). customizable
(3). extensible
(4). configurable
(5). Lightweight

These features means the programmer can use this toolkit to design their own GUI and run this GUI on different platform. The following picture indicates the GUI which involves in this chatting room application.

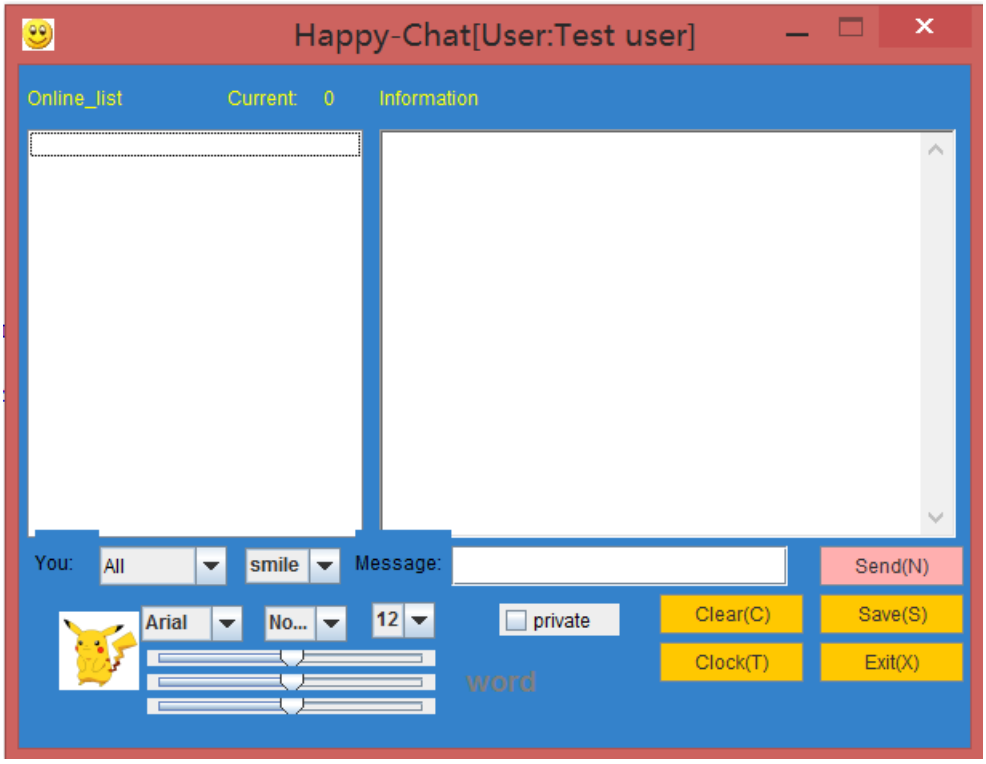

Figure 7: GUI for this project

As shown in this picture, I add many different components in this chatting room frame such as button, scroll bar, check-box, text area and text box. Through these GUI components, user will have a better user experience on the interact section. I will indicates the design and implement details in the later part.

3. Implementation details

3.1 Implementation details of server

In the example of the Java code which I used to show how to implement a server in Java, when that program start to run, the server can only process connect request from one server and can only connect with one client at the same time. The accept method will generate a block is the reason of this situation. That means when the run method is started to execute, it will call a accept method to process the connect process from the client, however, if accept method is occupied, other connect request will not be allowed to process until the current session is terminated. Due to this reason, I overridden the run method and instantiate a new thread which is called "Connection" thread to process the current session after the current client connected to server successfully. Through this mechanism, the server can support many clients and avoid to generate a block state The code details of this part is shown in the following picture.

```
public void run() {
    try {
        while (true)
        {
            Socket client = serverSocket.accept();
            new Connection(sFrame, client, userOnline, v);
        }
    } catch (IOException e)
    {
        fail(e, "Failed to monitor!");
    }
}
```

Figure 8: Code for server part of this project

3.2 Implementation details of client and GUI

3.2.1 Implementation details of client window

In this project, I assume that each user is an independent thread and when the client object is instantiated for one time, a new client thread will be created at the same time. After that, each user can choose the next step by their choice like click the login button to input the username and password to login the chatting room or click the register button to register a new user and then try to login the chatting room again. When the new user start to register, he must input the basic information such as username, password, age, sex and email address. Each item have a related checking mechanism to confirm the user input an acceptable answer for each item like the email address must have a @ character. If the user complete the register step and login the chatting room successfully, a new window will appears and this window is the main frame for each user and main GUI design is contained in this window. The following picture shows

this window and I will introduce the major function for this window such as action performed function and other GUI components.

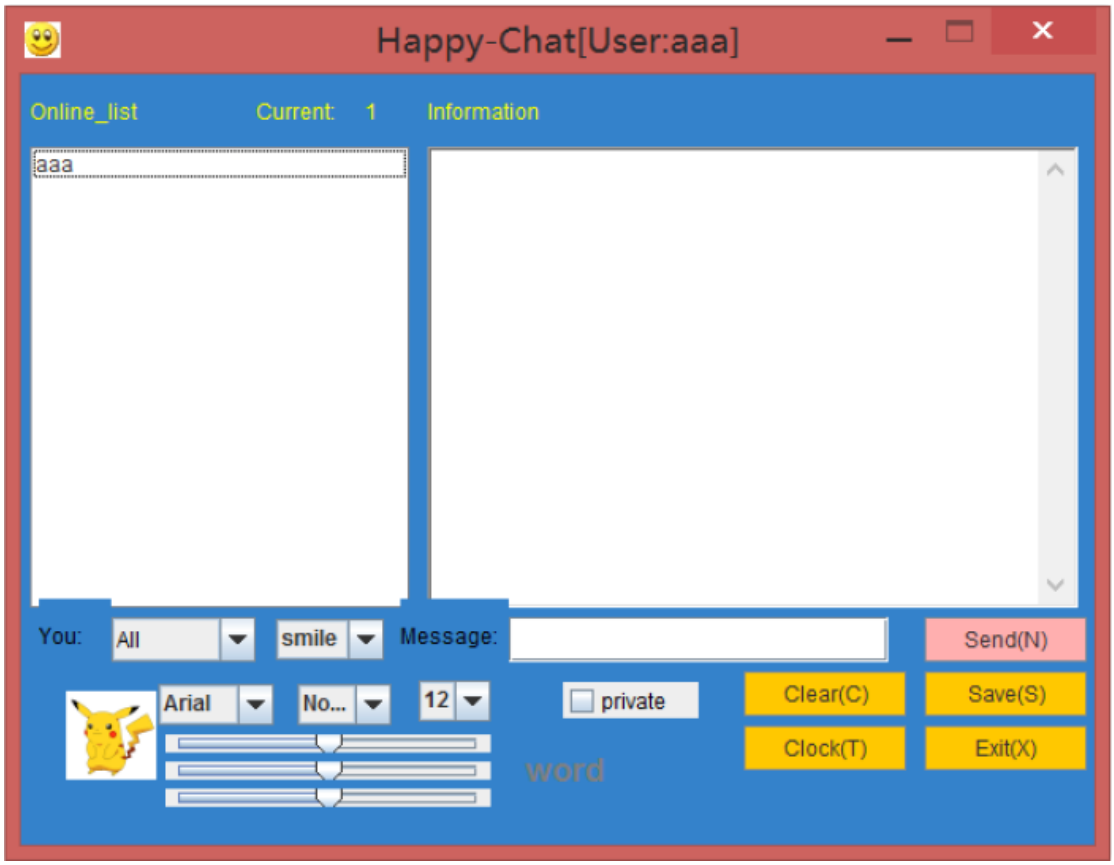

Figure 9: Main frame of this chatting room application

As shown in this picture, you can find that in the middle of this main frame, there exist two sub-frames which is the online user list-box and the other is the information text area. For the online user list-box, once a user login to this chatting room successfully, his or her username will appears in this list-box and the number of current online user which shows after the Current label will also refresh at the same time, in this picture, a user which user name is “aaa” login to this chatting room and the value of this number refresh to 1 at once. Each user can choose the username which listed in this box as a target to send a message and the message content will also appears in the information text area. In the left down corner of this frame, you can find that there exist three combo box and three scroll bars which are used to change the color or font and style of font. For example, you can change the color of font from white to black or change the size of font from 12 to 16. Moreover, you can find that there exist a check box which is labeled by private, this check box is used to define the type of chatting, if choose this check box and make the type change to private, the message which the user send will not record in the server log, otherwise, the content of message will be recorded in the server log. Each user will have a chatting record file which format is txt and the file’s name is the username, all the chatting content which belong to this user will be recorded in this file. Another important part for this frame is the button set which located in the right down corner of this frame and contains five buttons such as clear button, save

button, clock button, exit button and save button. Each button have a response function and I will introduce these response functions in details in the next section.

3.2.2 Implementation details of action performed function for button set

The response function is a major part of this chatting room implementation. When the user click the clear button, all the content in the information text area will be cleared. If the user click the clock button, a new window will appear and draw a clock figure to show the current time. Once the save button is clicked, the chatting content will be saved as a txt format file. The function of the exit button is obviously which means the user will logout the chatting room and the number of current label will be decreased one and refreshed at the same time. The online user list box will be refreshed at the same time either.

The most significant response function of this button set is the "send" part. As we known that, the major task of this part is to communicate with server. The following picture shows the code details of this function and I will explain it in details.

```
public synchronized void sendMessage()
{
    Chat chatobj = new Chat();
    chatobj.chatUser = strLoginName;
    chatobj.chatMessage = txtMessage.getText();
    chatobj.chatToUser = String.valueOf(cmbUser.getSelectedItem());
    chatobj.whisper = chPrivateChat.isSelected() ? true : false;
    chatobj.emote = emote.getSelectedItem().toString();
    // send this message to server
    try {
        Socket toServer = new Socket(strServerIp, 1001);
        ObjectOutputStream outObj = new ObjectOutputStream(toServer
                .getOutputStream());
        outObj.writeObject(chatobj);
        txtMessage.setText(""); // clear text box
        outObj.close();
        toServer.close();
    } catch (Exception e) {
    }
}
```

Figure 10: Code for response function of send button

As shown in this picture, when the user click the send button, a new chat object is created first and the type of this object is Chat. From this code part, you can find that the Chat class contains many details such as the username, content of message, the receiver of this message, the type of this message of chatting and the emote of this message. Then, a new socket object is created either and which means the client start to try connect with the server. In this case, I use the ObjectOutputStream class to change the type of data flow from byte to character and then use the writeObject method to add current data flow. After that, the current message will be sent to server side and the client side will close the current data flow and current transmit socket.

3.2.3 Implementation details of server window

In the above section, I have indicate the code implementation of the server part and I will use the following picture to show the GUI design of the server part.

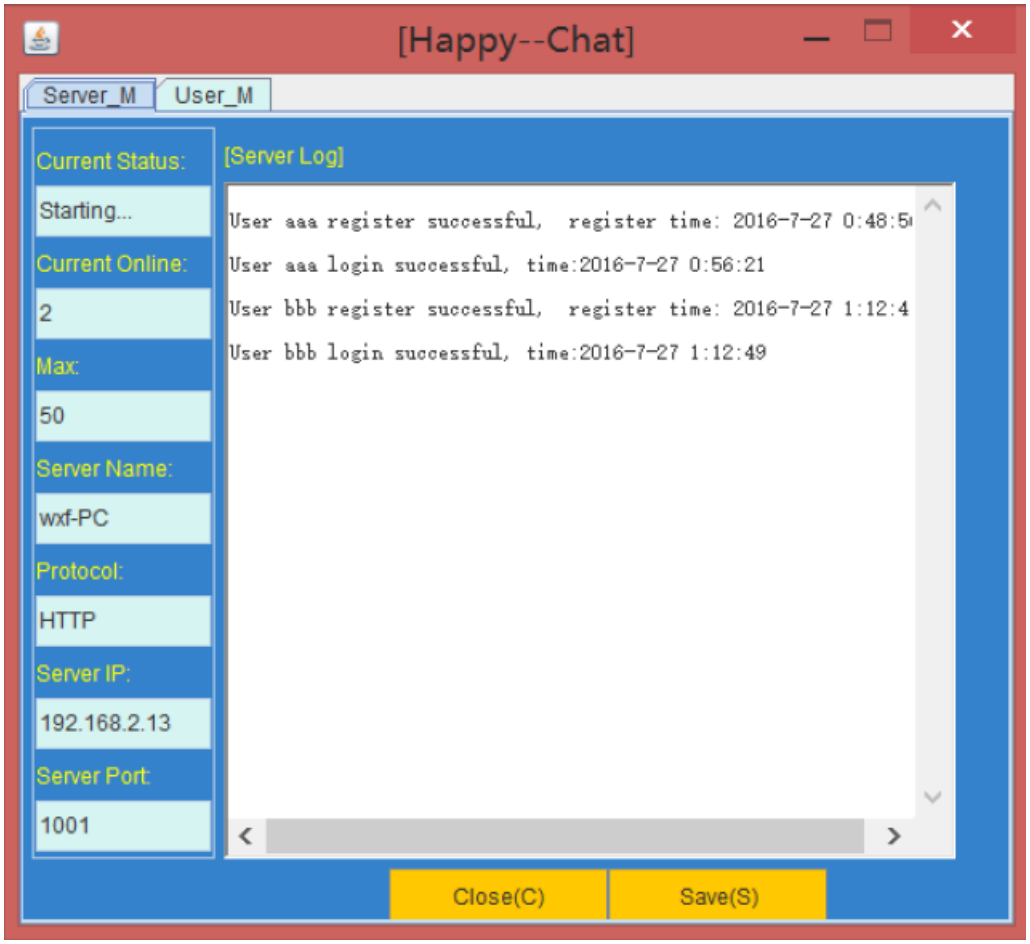

Figure 11: GUI design for server part (server_m) of this project

As shown in this picture, you can find that the server frame contains two label items which are Server_M part and User_M part. In the server_m part, all the basic information of this server will be shown in the left part and the server log text area is located in the right part of this frame. In this example, there are two users which have completed the register steps and login to this chatting room successfully. All the events like this will be recorded in the text area and if the user click the save button, this server log will be saved as a txt file like the chatting content. It is important to say that not to click the close button until you want to exit this chatting completely due to the server will be closed and all chatting room windows will also be closed if user click this button

In the user_m part, there still exist a text area which is located in the left part of the window and is named by user message. All the public chatting content will shows in this area. In the right part, there have an online list box and all online user's name will shows in this box. In the bottom area of this frame, there have a textbox and two buttons which are send button and kick button. If the user input the message in the textbox and click the send button, the server will send this message to all connected clients. If the

user choose a username in the list box and then click the kick button, this client would be forced to logout to this chatting room and this event will be noticed by the other online clients and be recorded in the server log. The following picture shows these details and I will explain it by an example.

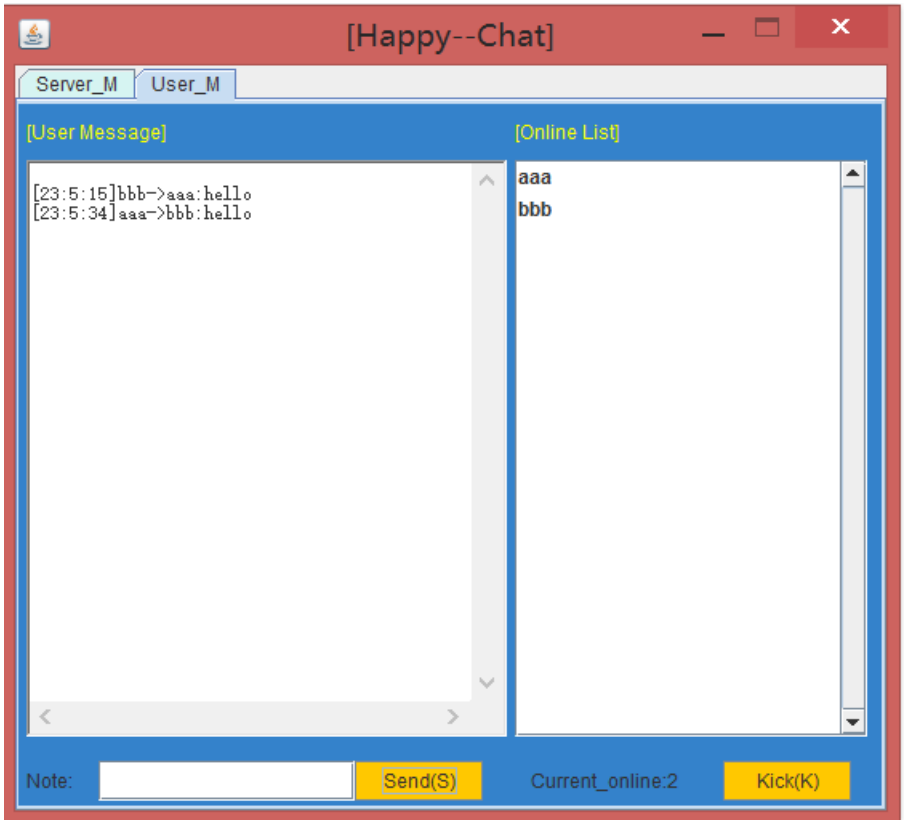

Figure 12: GUI design for server part (user_m frame) of this project

As shown in this picture, you can find that there exist two users and both of them are send a message to another user. In this example, both of them send hello as a message to the other and these two events are recorded in the user message text area. However, if the chatting type is private, that message will not be recorded in this area. If you want to control the number of online user, you just need to choose that username and then click the kick button, that user will be forced to logout. The server_m frame and user_m frame consists of the whole server frame, most of information about server and chatting room status can be found in this frame.

3.3 Implementation details of communication between server and client

This part is the most significant part of the whole project. In general, the communication between server and client involves three steps. The first step is the server still monitor the specific port and wait one client send a connect request. After this, if one client send a request to server, the server will generate a new ServerSocket object to process this request and start to execute the accept method. If this connect request is processed successfully, both the server and client will have a socket instance and start to send and receive message from each other through this socket. In this project, I simulate this process and implement the design goal which is support multiple threads in the server

and client connection section. In the following part, I will explain more implementation details of this part through explain the java codes which used to implement this section in this project.

3.3.1 Run method in the server class

```
public void run() {
    try {
        while (true)
        {
            Socket client = serverSocket.accept();
            new Connection(sFrame, client, userOnline, v);
        }
    } catch (IOException e)
    {
        fail(e, "Failed to monitor!");
    }
}
```

Figure 12: Java code of run method in the server class

As shown in this method, when the server monitor the specific port and receive a connect request from the client, it will instantiate a new ServerSocket object and call accept method to process this request. After that, the server will generate a new thread to continue process this session and avoid generate block state.

3.3.2 Run method in the connection class

```
public void run() {
    try
    {
        obj = (Object) fromClient.readObject();
        if (obj.getClass().getName().equals("Customer")) {
            serverLogin();
        }
        if (obj.getClass().getName().equals("Register_Customer")) {
            serverRegiste();
        }
        if (obj.getClass().getName().equals("Message")) {
            serverMessage();
        }
        if (obj.getClass().getName().equals("Chat")) {
            serverChat();
        }
        if (obj.getClass().getName().equals("Exit")) {
            serverExit();
        }
    } catch (IOException e) {
        System.out.println(e);
    } catch (ClassNotFoundException e1) {
        System.out.println("Failed to read object" + e1);
    } finally {
        try {
            netClient.close();
```

Figure 13: Java code of run method in the connection class

As shown in this picture, each time when the client send a message to server, the server side will read the message from the data flow and obtain the class name of the current object and then call different methods to process these request and close the socket in the end.

3.3.3 Run method in the chatting room class

```
public void run()
{
    int intMessageCounter = 0;
    int intUserTotal = 0;
    boolean isFirstLogin = true;
    boolean isFound;
    Vector user_exit = new Vector();
    try {
        for (;;) {
            Socket toServer;
            toServer = new Socket(strServerIp, 1001);
            messobj = new Message();
            ObjectOutputStream streamtoserver = new ObjectOutputStream(
                    toServer.getOutputStream());
            streamtoserver.writeObject((Message) messobj);
            ObjectInputStream streamfromserver = new ObjectInputStream(
                    toServer.getInputStream());
            messobj = (Message) streamfromserver.readObject();
```

Figure 14: Java code of run method in the chatroom class

As shown in this picture, when a chatting room is created, it will repeat to make a connection with server and send or receive message from server. The chatting room thread will also call different method depends on the return message from the server. For example, if a client exit to chatting room and send a message to server, the server side will obtain the information of this user from that message and then send a new message to other connected clients to indicate which user is exit and refresh the current online user list at the same time.

3.4 Keeping currency in the communication among clients

3.4.1 Synchronized keyword

Synchronized keyword is provided by the Java concurrent package and it is a choice for programmer to ensure the thread safety during the program execution, especially in the multiple-thread environment. Synchronized keyword provides two kinds of implementation which is synchronized method and the other is synchronized block. For the synchronized method, each class instance will related to a lock and each synchronized method must obtain this lock before to call this class instance, otherwise, the thread which belong to this synchronized method will be blocked. If the method obtain this lock and start to execute, this lock will be occupied until this method returned. Other threads which also need to call this class instance will repeat this procedure and this mechanism ensure that this class instance will be used for only one thread at one time. The major difference between the synchronized method and

synchronized block is the synchronized block is more flexible and can be used on any code part and any object which need to add a lock.

3.4.2 Implementation details of synchronized method

In this project, each client can sends message to another client and if multiple clients want to send message to a same client, a block state may occurs and the message data flow may occurs a mistake either due to just one thread is created to process these message at one time. To solve this problem and ensure each message can be processed correctly, I used synchronized method in the chatroom class and ensure that only one client can call the socket object on the receive side and the other clients must to wait. Due to the server and client will run on the same computer in my simulation, therefore, there will exist a little delay when the client send a message to server or another client. This is the result of use the synchronized method because of if two or more messages will be sent to a same receiver, the process order for these message is depend on arrive order and the thread on the receive side will be processed them one by one. In the future, if run the server part and client part on different computers and process speed is quickly enough, this delay will be decreased and the user may not feel it. The following picture shows the code part of response function of the send button which is modified by synchronized keyword.

```
public synchronized void sendMessage()
{
    Chat chatobj = new Chat();
    chatobj.chatUser = strLoginName;
    chatobj.chatMessage = txtMessage.getText();
    chatobj.chatToUser = String.valueOf(cmbUser.getSelectedItem());
    chatobj.whisper = chPrivateChat.isSelected() ? true : false;
    chatobj.emote = emote.getSelectedItem().toString();
    // send this message to server
    try {
        Socket toServer = new Socket(strServerIp, 1001);
        ObjectOutputStream outObj = new ObjectOutputStream(toServer
                .getOutputStream());
        outObj.writeObject(chatobj);
        txtMessage.setText(""); // clear text box
        outObj.close();
        toServer.close();
    } catch (Exception e) {
    }
}
```

Figure 15: Java code of synchronized method in the chatroom class

4. Conclusion

In this project, I use the ServerSocket and Socket to implement the communication between server and client. Also, use the swing toolkit to implement a simple GUI of the chatting room and the whole application can implement common functions as the real chatting application. In the future, I will study the knowledge of Java socket and concurrency more deeply and make this application more perfect.

Reference

[1] https://examples.javacodegeeks.com/core-java/java-multithreading-tutorial/
[2] http://www.buyya.com/java/Chapter13.pdf
[3] "Big Java; Early Objects, 5th Ed." *Reference and Research Book News* 28.4 (2013): Reference and Research Book News, Vol.28 (4). Web.