CSEE 5110 NETWORK ARCHITECTURE
PROJECT-2 REPORT

GROUP CHAT APPLICATION PROGRAM

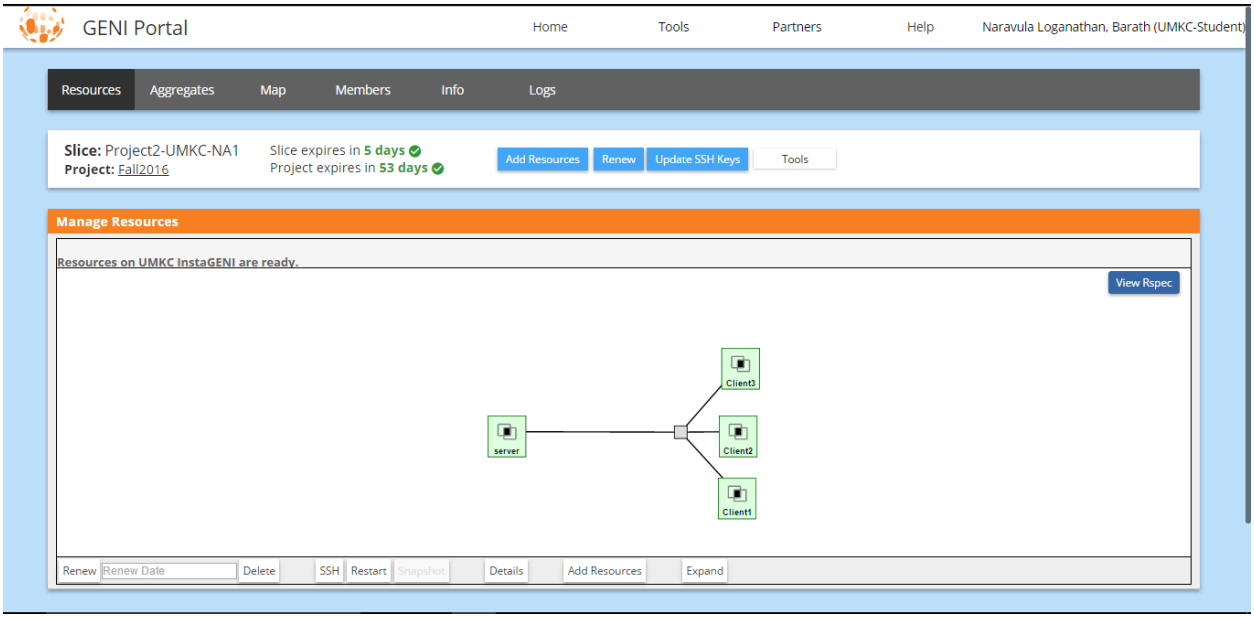
PROJECT TEAM:

Barath Naravula Loganathan – 16228941

Venkata Raghava Kundavajjala-16209348

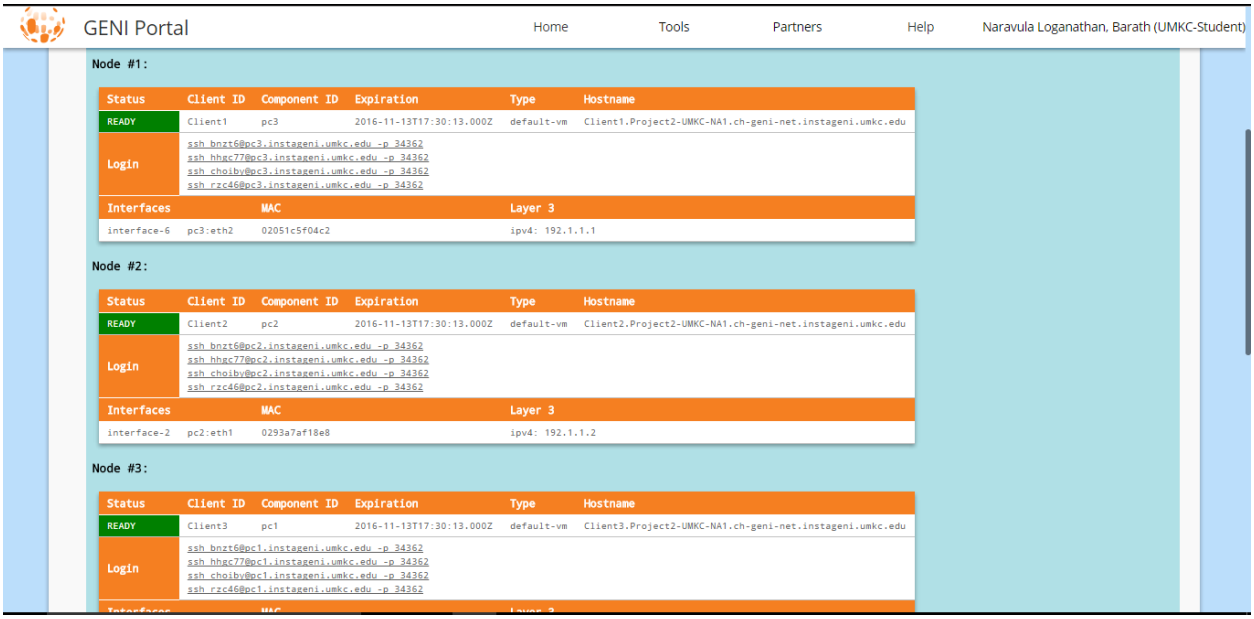
Sangarshan Reddy Karra-16223081

The project started off with creation of a new slice in Geni, **Project 2-UMKC-NA1** under Fall 2016 project. A server with multiple clients(nodes) are created. Specifically, **3 clients** are connected to **one server** for this project.



These are the details of the clients and servers

- Clients: Node#1 is Client1, Node#2 is Client2 and Node#3 is Client3
- Server: Node#4 is Server



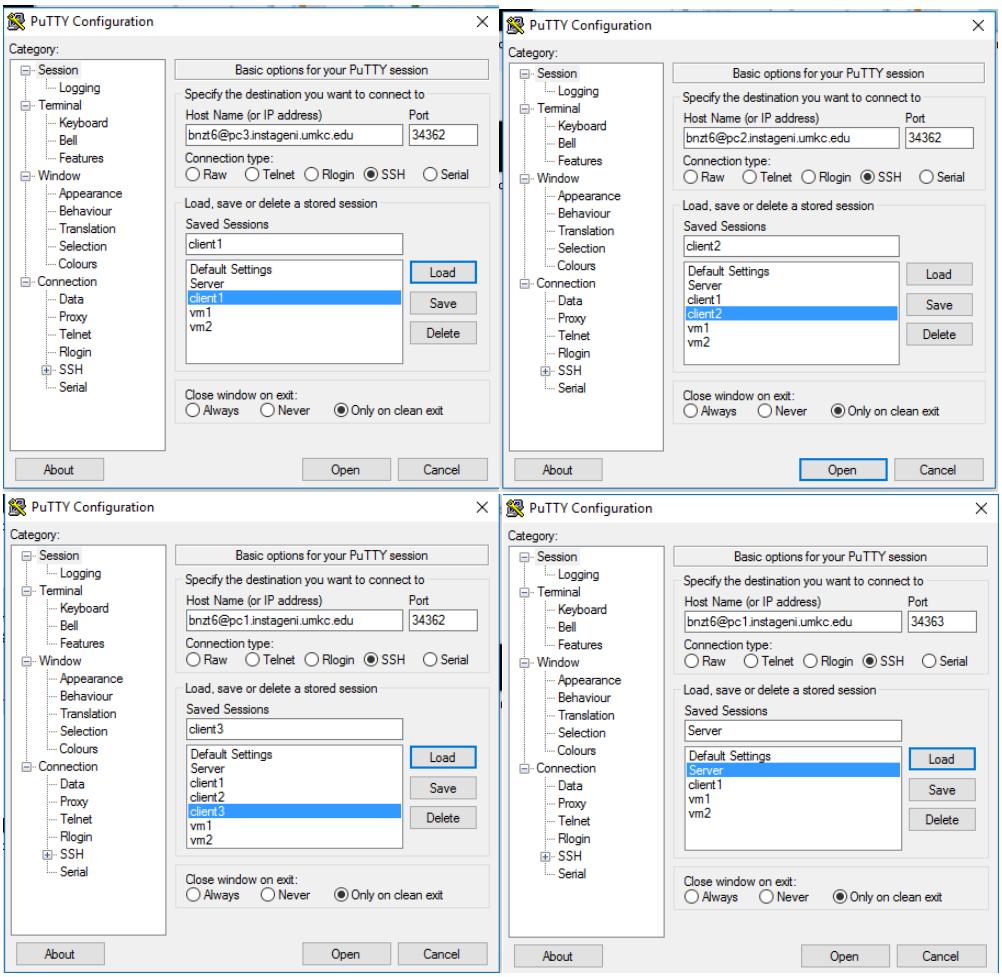
Node #4:

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	server	pc1	2016-11-13T17:30:13.000Z	default-vm	server.Project2-UMKC-NA1.ch-geni-net.instageni.umkc.edu
Login	ssh bnzt6@pc1.instageni.umkc.edu -p 34363 ssh hhgc77@pc1.instageni.umkc.edu -p 34363 ssh choiby@pc1.instageni.umkc.edu -p 34363 ssh rzc46@pc1.instageni.umkc.edu -p 34363				

Interfaces		MAC	Layer 3
interface-3	pc1:eth1	02a1f1b92333	ipv4: 192.1.1.10

Link #1:

Client ID	Endpoint #0	Endpoint #1	Endpoint #2	Endpoint #3
link-1	interface-2	interface-3	interface-6	interface-7



Now we execute the program, multiple client server chat application. We execute the all the A, B, C, D section using a Client and Server programs into single piece of code respectively.

Part A:

For part A, “A chat server will accept a single client connection and display everything the client types. If the client user types ‘exit’, both client and server will end the program.”

```
bnzt6@client1:~$ vi Client1D.java
bnzt6@client1:~$ javac Client1D.java
Note: Client1D.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
bnzt6@client1:~$ java Client1D
------------------------------------------------------------
**************** CHAT Application - Client **************
------------------------------------------------------------
** Usage: host=192.1.1.10, portNumber=33666 **
------------------------------------------------------------
Enter your name:
raghava
 /// Hello raghava to chat room.
 /// To leave enter /exit in a new line
hello how are you
 >> raghava >>hello how are you
/exit
 /// Client raghava exit. Type Bye to close connection///
Bye
bnzt6@client1:~$
```

```
bnzt6@server:~$ vi Server1D.java
bnzt6@server:~$ javac Server1D.java
Note: Server1D.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
bnzt6@server:~$ java Server1D
----------------------------------------
************ Server Running ************
----------------------------------------
** Usage: Port number=33666 **
----------------------------------------
>> raghava : hello how are you
>> raghava : /exit
```

The Client joins the chat by typing the name first. The messages are displayed in the server. The chat is exited by typing ‘/exit’ . We can see the alert “///Client raghava<user name> exit. Type Bye to close connection”. The connection is cut of typing Bye on client side.

PartB:

For partB, “A server now remains ‘open’ for additional connection once a client quits. The server can handle at most one connection at a time.”

```
bnzt6@server:~$ vi Server1D.java
bnzt6@server:~$ javac Server1D.java
Note: Server1D.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
bnzt6@server:~$ java Server1D
----------------------------------------
************ Server Running ************
----------------------------------------
** Usage: Port number=33666 **
----------------------------------------
>> raghava : hello how are you
>> raghava : /exit
```

The Server remains open even after the client is exited from above screenshot.

PartC:

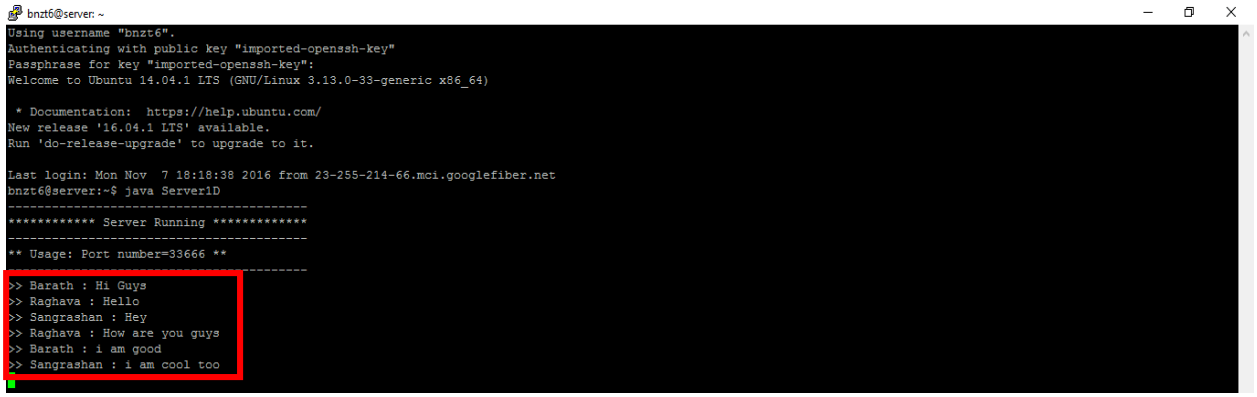
For partC, “A server now can handle multiple clients at the same time. The output from all the connected clients will appear on the server’s screen.”

Client1:

Client2:

Client3:

Server:



All the messages from all the three clients are there on the server side as shown in above highlighted box. We can see multiple clients connected.

PartD:

We can see even part D in the screenshots above, the client messages are echoed everywhere, on every client. From the above screenshots, we can see that for partD, that is echoing the client messages to every client connected "A server now echoes all the text received from any of the connected clients to all".

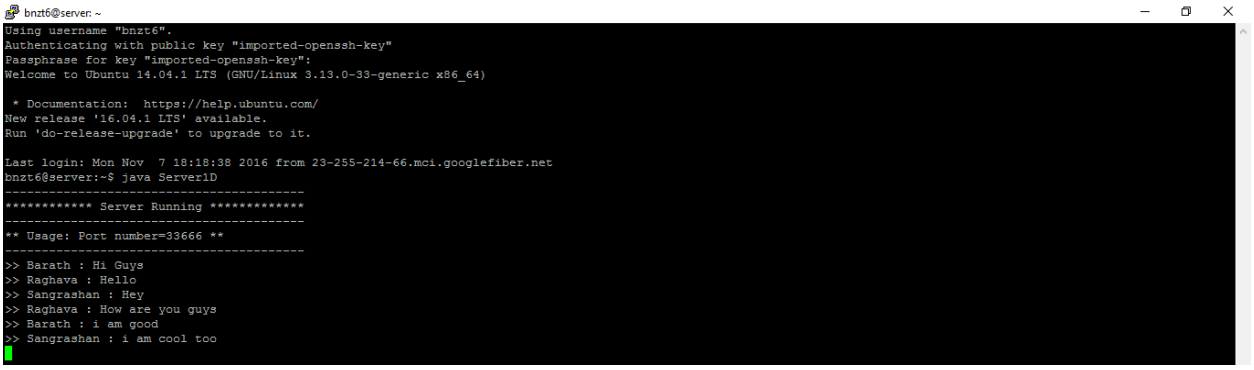
Legend : Client 1(🟧) Client 2(🟦) Client 3(🟩)

Client1:

Client2:

Client3:

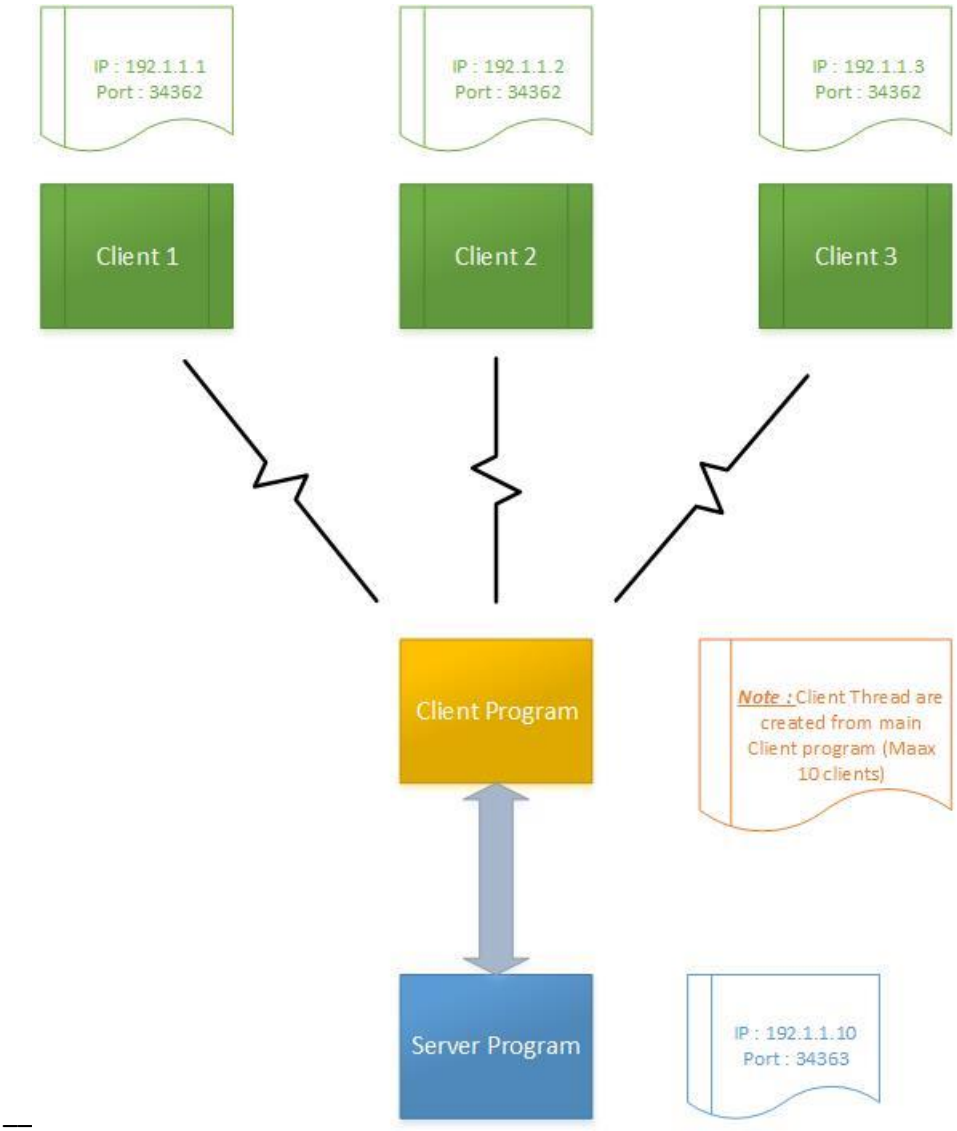
Server:



We can see for messages from client1 and client2 and alert actions such as joining and exiting are shown in client3.

More on Multi-Client single Server chat application program:

The programs for all the three clients are the same, so we created a single file **Client1D.java** and creates individual threads when we run this client program from different resource which act as separate client and **Server1D.java** act as server program.



----------Client1D.java program----------

```
import java.io.DataInputStream;
import java.io.PrintStream;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;
import java.net.Socket;
import java.net.UnknownHostException;

public class Client1D implements Runnable {

 // The client socket
 private static Socket clientSocket = null;
 // The output stream
 private static PrintStream os = null;
 // The input stream
 private static DataInputStream is = null;

 private static BufferedReader inputLine = null;
 private static boolean closed = false;

 public static void main(String[] args) {

  // The default port.
  int portNumber = 33666;
  // The default host.
  String host = "192.1.1.10";

  System.out.println("------------------------------------------------------------");
        System.out.println("****************** CHAT Application - Client ***************");
        System.out.println("------------------------------------------------------------");
        System.out.println("** Usage: host=" + host + ", portNumber=" + portNumber+" **");
        System.out.println("------------------------------------------------------------");


  /*
   * Open a socket on a given host and port. Open input and output streams.
   */
  try {
   clientSocket = new Socket(host, portNumber);
   inputLine = new BufferedReader(new InputStreamReader(System.in));
   os = new PrintStream(clientSocket.getOutputStream());
   is = new DataInputStream(clientSocket.getInputStream());
  } catch (UnknownHostException e) {
```

```
      System.err.println("Unknown host " + host);
    } catch (IOException e) {
      System.err.println("Connection Error to "+ host);
    }
    /*
     * If everything has been initialized then we want to write some data to the
     * socket we have opened a connection to on the port portNumber.
     */
    if (clientSocket != null && os != null && is != null) {
      try {
        /* Create a thread to read from the server. */
        new Thread(new Client1D()).start();
        while (!closed) {
          os.println(inputLine.readLine().trim());
        }
        /*
         * Close the output stream, close the input stream, close the socket.
         */
        os.close();
        is.close();
        clientSocket.close();
      } catch (IOException e) {
        System.err.println("IOException:  " + e);
      }
    }
  }
  public void run() {
    /*
     * Keep on reading from the socket till we receive "Bye" from the
     * server. Once we received that then we want to break.
     */
    String responseLine;
    try {
      while ((responseLine = is.readLine()) != null) {
        System.out.println(responseLine);
        if (responseLine.indexOf("*** Bye") != -1)
          break;
      }
      closed = true;
    } catch (IOException e) {
      System.err.println("IOException:  " + e);
    }
  }
}
```

----------Server1D.java program----------

```
import java.io.DataInputStream;
import java.io.PrintStream;
import java.io.IOException;
import java.net.Socket;
import java.net.ServerSocket;

/*
 * A chat server that delivers public and private messages.
 */
public class Server1D {

  // The server socket.
  private static ServerSocket serverSocket = null;
  // The client socket.
  private static Socket clientSocket = null;

  // This chat server can accept up to maxClientsCount clients' connections.
  private static final int maxClientsCount = 10;
  private static final clientThread[] threads = new clientThread[maxClientsCount];

  public static void main(String args[]) {

   // The default port number.
   int portNumber = 33666;

          System.out.println("----------------------------------------");
          System.out.println("************ Server Running **************");
          System.out.println("----------------------------------------");
          System.out.println("** Usage: Port number=" + portNumber+" **");
          System.out.println("----------------------------------------");
    /*
    * Open a server socket on the portNumber 33666
    */
   try {
    serverSocket = new ServerSocket(portNumber);
   } catch (IOException e) {
    System.out.println(e);
   }
   /*
    * Create a client socket for each connection and pass it to a new client
    * thread.
    */
   while (true) {
```

```
      try {
        clientSocket = serverSocket.accept();
        int i = 0;
        for (i = 0; i < maxClientsCount; i++) {
          if (threads[i] == null) {
            (threads[i] = new clientThread(clientSocket, threads)).start();
            break;
          }
        }
        if (i == maxClientsCount) {
          PrintStream os = new PrintStream(clientSocket.getOutputStream());
          os.println("Server reached Maximum clients. Try later.....");
          os.close();
          clientSocket.close();
        }
      } catch (IOException e) {
        System.out.println(e);
      }
    }
  }
}
class clientThread extends Thread {

  private DataInputStream is = null;
  private PrintStream os = null;
  private Socket clientSocket = null;
  private final clientThread[] threads;
  private int maxClientsCount;

  public clientThread(Socket clientSocket, clientThread[] threads) {
    this.clientSocket = clientSocket;
    this.threads = threads;
    maxClientsCount = threads.length;
  }
  public void run() {
    int maxClientsCount = this.maxClientsCount;
    clientThread[] threads = this.threads;
    try {
      /*
       * Create input and output streams for this client.
       */
      is = new DataInputStream(clientSocket.getInputStream());
      os = new PrintStream(clientSocket.getOutputStream());
      os.println("Enter your name:");
```

```
    String name = is.readLine().trim();
    os.println(" /// Hello " + name+ " to chat room.\n /// To leave enter /exit in a new line");
    for (int i = 0; i < maxClientsCount; i++) {
      if (threads[i] != null && threads[i] != this) {
        threads[i].os.println(" /// Notice ::  A new user " + name+ " entered the chat room !!! ///");
      }
    }
    while (true) {
      String line = is.readLine();
                    System.out.println(">> "+name+" : "+line);
      if (line.startsWith("/exit")) {
        break;
      }
      for (int i = 0; i < maxClientsCount; i++) {
        if (threads[i] != null) {
          threads[i].os.println(" >> " + name + " >>" + line);
        }
      }
    }
    for (int i = 0; i < maxClientsCount; i++) {
      if (threads[i] != null && threads[i] != this) {
        threads[i].os.println(" /// Notice :: The user " + name + " has left the chat room !!! ///");
      }
    }
    os.println(" /// Client " + name + " exit. Type Bye to close connection///");
    /*
     * Clean up. Set the current thread variable to null so that a new client
     * could be accepted by the server.
     */
    for (int i = 0; i < maxClientsCount; i++) {
      if (threads[i] == this) {
        threads[i] = null;
      }
    }
    /*
     * Close the output stream, close the input stream, close the socket.
     */
    is.close();
    os.close();
    clientSocket.close();
   } catch (IOException e) {
   }
  }
}
```

REFERENCES:

http://makemobiapps.blogspot.com/p/multiple-client-server-chat-programming.html

http://stackoverflow.com/questions/10131377/socket-programming-multiple-client-to-one-server

http://stackoverflow.com/questions/27823167/multi-threading-client-server-chat-application-in-java