1. METADATA

CIS 675 INFORMATION SECURITY ASSIGNMENT 2 PATR1 (Multi-Threaded Chat Program)
HACI KARAHASANOGLU
CSUID 2502553

2. SUMMARY

Making this chat program for multiple clients was challenging for me. It took long time to figure multiple thread programming and apply it into my codes. Another challenging part was getting port number and hostname. I need to start server first, and server starts with port number, and then client starts communication through that port. So I spent too much time to figure it out at the end I add default port and hostname in the program. Closing pipes for each thread was important otherwise when I was typing in client2' s console message was shown in client1's account. I don't think there is any missing function, but limited client connection which I defined in the program as 4 user, however you may increase that number in the code.

3. Specification

Program has two side, server and client side. First server must run, and user run the client side. When client start it ask username which will display on console to point out which user type which message. Whenever a new client connect program ask user name. Let say there are 4 user in the chat room, when one type a message server distribute that message to everyone. There are 4 user limit but it may be increased in code as many user as you want. I thought 4 will be enough, if not we can change it in the code. Program uses default hostname "localhost" and port number "2500". I did this justification because of I couldn't figure it out running server without port number that is why I defined hostname and port number default in the program.

4. Design and implementation

Server.java
Public class Server has a main method, and main method create defaultt serverSocket with port number 2500, and create 4 socket, thread for each client.Chat client thread open input and output stream for specific client, ask that client user name and broadcast that client's message to every connected clients.

Client.java

Open socket by using default host and port number, open input and output stream, create a thread to read from the server

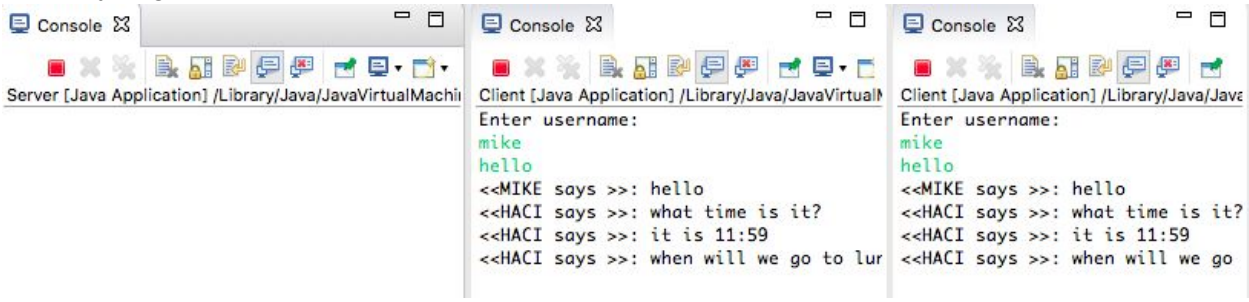
I used try catch while opening a socket, input output stream, this helps to see the error, and as a result of my research and readings I found this is common practice for this kind of programs

There is only one kind of data structured used which is 1 dimensional array on server side to create client threads

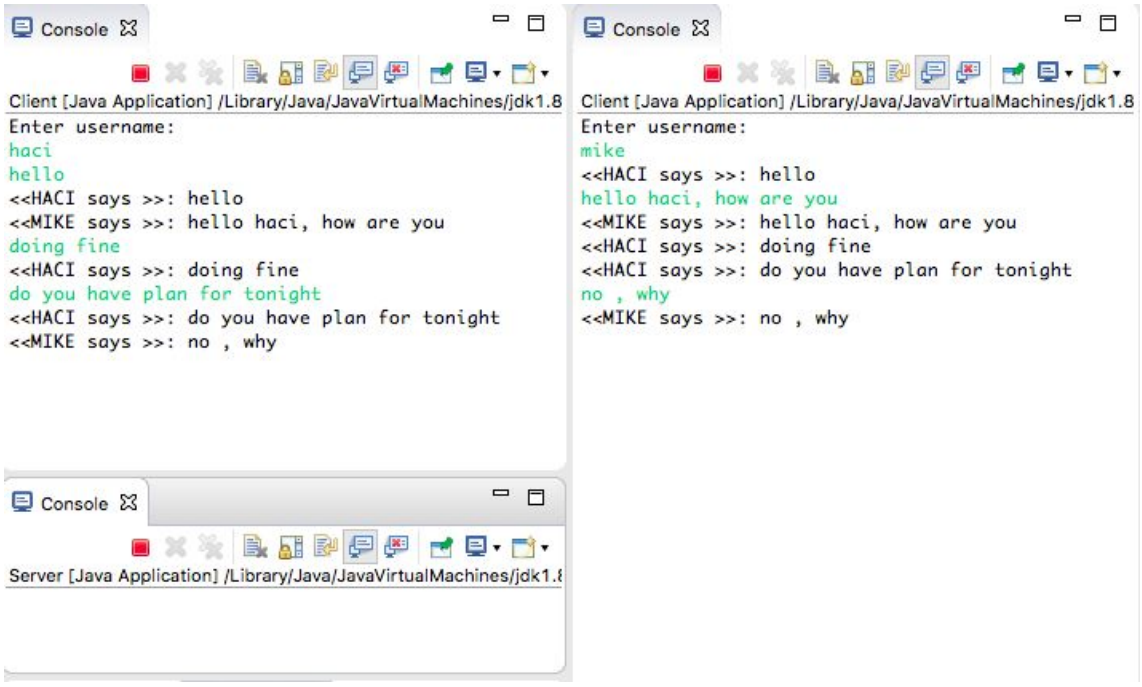
I had problem, eventhough I was typing on client2's console somehow clinet2's message was popping up like Client1's message. This took couple ours to figure, then I realized I need to add close code in below section at Client.java

```
//Reading from server
                try {
                        new Thread(new Client()).start();
                        while (!isClosed) {
                                outputStream.println(input.readLine());
                        }
                } catch (IOException e) {
        System.out.println(e);
                }
          }
```

This keep socket open and broadcast message from the same user no matter other clients typing.



5. Testing



Above working, testing case has two clients, and a sever.

6. Known Problems

I don't think there is any known problem for this assignment. As far as I understood my codes meet with the requirements of the assignment Part1. If there will be any problem that might be misunderstanding.

7. Comments

I spent around total of 20 hours for research articles, design, coding and testing.
Estimated Research articles and learning: 15 hours
Estimated Design Time: 2 hours
Estimated Coding Time: 2 hours
Estimated Testing Time: 1 hours

More details would be better with the examples. Because I have never done any similar assignment before, so i though all communications will be in between server and client, I later learned it will be multi client with threads.

This assignment was hard for me, I couldn't do the time management and that is why I submit it late.

```
/* CIS 675 Assignment#2
 * Part 1
 * Server Side
 * Haci Karahasanoglu
 */
import java.io.IOException;
import java.io.DataInputStream;
import java.io.PrintStream;
import java.net.Socket;
import java.net.ServerSocket;

public class Server {
	// Server soket
	public static ServerSocket serverSocket;
	// Client soket
	public static Socket clientSocket;
	// Client Threads
	public static ClientThread[] clientThreads = new ClientThread[4];

	public static void main(String args[]) {
		// default port 2500
		try {
			serverSocket = new ServerSocket(2500);
		} catch (IOException e) {
			System.out.println(e);
		}

		// Creating sockets and threads foe each client, maximum client number
		// is 4
		do {
			try {
				clientSocket = serverSocket.accept();
				for (int i = 0; i < 4; i++) {
					if (clientThreads[i] == null) {
						(clientThreads[i] = new ClientThread(clientSocket,
clientThreads)).start();
						break;
					}
				}
			} catch (IOException e) {
				System.out.println(e);
			}
		} while (true);
	}
}

// Matching clients for each thread
class ClientThread extends Thread {

	public DataInputStream inputStream;
	public PrintStream outputStream;
	public Socket clientSocket;
```

```
	public ClientThread[] threads;
	public int maxClientNumber;

	public ClientThread(Socket clientSocket, ClientThread[] threads) {
		this.clientSocket = clientSocket;
		this.threads = threads;
		maxClientNumber = threads.length;
	}

	public void run() {
		int maxClientNumber = this.maxClientNumber;
		ClientThread[] threads = this.threads;
		// Creating input and output stream for the client and ask username at
		// first login
		try {
			inputStream = new DataInputStream(clientSocket.getInputStream());
			outputStream = new PrintStream(clientSocket.getOutputStream());
			outputStream.println("Enter username: ");
			String username = inputStream.readLine();
			// Echo client message to all other clients
			do {
				String text = inputStream.readLine();
				for (int i = 0; i < maxClientNumber; i++) {
					if (threads[i] != null) {
						threads[i].outputStream.println("<<" + username.toUpperCase() + "
says >>: " + text);
					}
				}
			} while (true);
		} catch (IOException e) {
		}
	}
}


/*CIS 675 Assignment#2
 * Part 1
 *Server Side
 *Haci Karahasanoglu
 */
import java.io.IOException;
import java.io.DataInputStream;
import java.io.PrintStream;
import java.io.BufferedReader;
import java.net.Socket;
import java.io.InputStreamReader;

public class Client implements Runnable {

	public static Socket clientSocket;
	public static PrintStream outputStream;
	public static DataInputStream inputStream;
	public static BufferedReader input;
```

```
    public static void main(String[] args) {
            // Create a client with default host and port number
            boolean isClosed = false;
            try {
                    clientSocket = new Socket("localhost", 2500);
                    input = new BufferedReader(new InputStreamReader(System.in));
                    outputStream = new PrintStream(clientSocket.getOutputStream());
                    inputStream = new DataInputStream(clientSocket.getInputStream());
            } catch (IOException e) {
System.out.println(e);
            }

            //Reading from server
            try {
                    new Thread(new Client()).start();
                    while (!isClosed) {
                            outputStream.println(input.readLine());
                    }
                    // Closing connections
                    outputStream.close();
                    inputStream.close();
                    clientSocket.close();
            } catch (IOException e) {
System.out.println(e);
            }
    }

    // Thread read from the server side
    public void run() {
            String reply;
            try {
                    while ((reply = inputStream.readLine()) != null) {
                            if (reply.trim().equals("")) {
                        // empty line
                    } else {
                        System.out.println(reply);
                    }
                    }
                    boolean isClosed = true;
            } catch (IOException e) {
System.out.println(e);
            }
    }
}
```

1. METADATA

CIS 675 INFORMATION SECURITY ASSIGNMENT 2 PATR2
HACI KARAHASANOGLU
CSUID 2502553

2. SUMMARY

In addition to Part 1 summary, I did not have much problem, all I did is read couple articles about how to apply DES in Java, check some sample codes, and I added two predefined array in server side of the code for authentication. And add DES code in server side as well. During this process I did not encounter any major error, and it was pretty smooth and in short time.

3. SPECIFICATION

I used Part 1 server and Client code. I only did addition in to Server code, and I guess those additions gave me what I wanted. I used two separate String array, one for username the other for password in server side, if password and username mathes server echoes client's message to all other clients, or client receives other clients' message, if not program give authentication error message to client.

I added a method in server side to do encryption/decryption, this method works after authentication, when client types a message this method encrypt it and decrypt while echoing to other clients. However even though I guess it works I have some doubts on the logic, I don't have much time to work on the logic due to late submission.

4. DESING AND IMPLEMENTATION

Architecture:
I have two seperate program to run, server and client. Server does most of the job. It does authentication and DES encryption. When client type a message server receives it and distribute all the other clients

Code Design:
I used mostly try catch and separate method for each task. Used two string array for authentication.

Programming:
I used part 1 server and client code, and did addition to server side for authentication and des encryption.

Below is the authentication code, get the username from the clinet, check its position in the username array, and get the password, and if password is matching with the password position the authentication is correct and clinet can send /receive message from other clients. If false get error message.

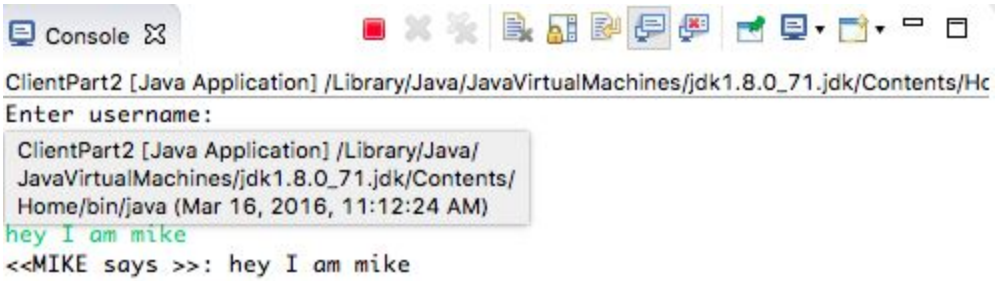
```
String[] usernameArray = { "haci", "mike", "adam", "john" };
```

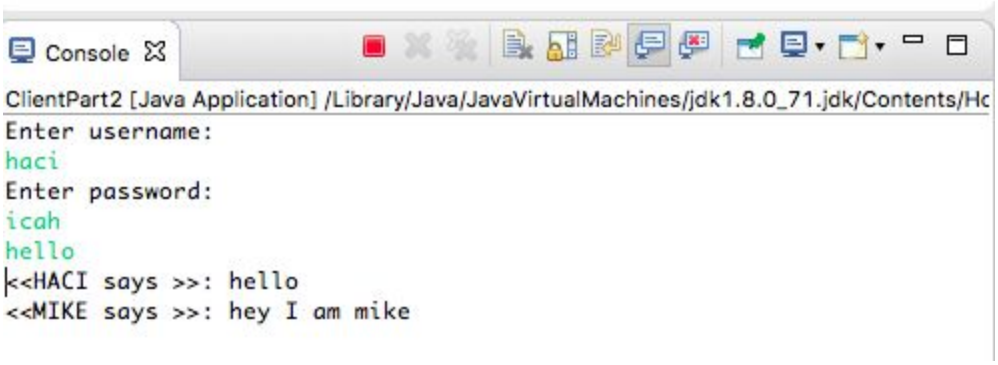
```
                String[] passwordArray = { "icah", "ekim", "mada", "nhoj" };

// Ask username and password
                        outputStream.println("Enter username: ");
                        String username = inputStream.readLine();
                        outputStream.println("Enter password: ");
                        String password = inputStream.readLine();

                        // Find entered username position in usernameArray
                        int index = -1;
                        for (int i = 0; i < usernameArray.length; i++) {
                                if (usernameArray[i].equals(username)) {
                                        index = i;
                                        break;
                                }
                        }

                        // Compered enterd password with the related username password, if
                        // matched continue, else display authentication error message
                        if (passwordArray[index].equals(password)) {
                                // Echo client message to all other clients
                                do {
                                        String text = inputStream.readLine();
                                        for (int i = 0; i < maxClientNumber; i++) {
                                                if (threads[i] != null) {
                                                        threads[i].outputStream.println("<<" +
username.toUpperCase() + " says >>: " + EncDec(text));
                                                }
                                        }
                                } while (true);
                        } else {
                                outputStream.println("Autentitcation Error");
                        }
```

DES enc/dec work on the server side, when a client type a message,m server encrypt it and send it to clients and other clients received decripted message.

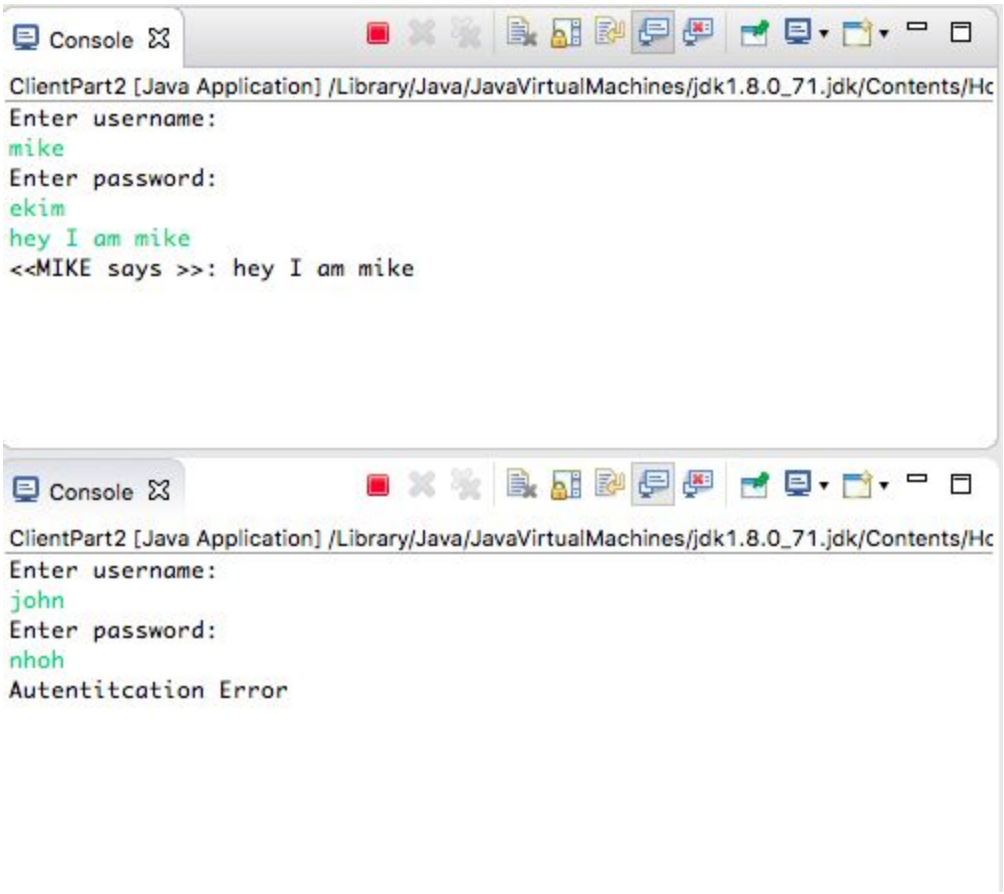
5. TESTING

Authentication testing





Authentication Error Example



I couldn't show the way of my DES algorithm working in visible, I may cancel decryption and show that each client get encrypted message but this may cause misunderstanding from the instructor of not working code.

6. KNOWN PROBLEMS

I don’t see and problem , I guess my program works efficiently, if there is any non working part that is most probably my misunderstanding of the requirement, as far as I know my program meets the requirements of the assignment.

7. COMMENTS

In addition to part 1 I may spent 3 hours, 2 hours for reading DES programming articles to see how it is working, and 1 hour for code implementation and testing.

```
/* CIS 675 Assignment#2
 * Part 1
 * Server Side
 * Haci Karahasanoglu
 */
import java.io.IOException;
import java.io.DataInputStream;
import java.io.PrintStream;
import java.net.Socket;
import java.net.ServerSocket;
//DES
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;

import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.KeyGenerator;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.SecretKey;

public class ServerPart2 {
	// Server soket
	public static ServerSocket serverSocket;
	// Client soket
	public static Socket clientSocket;
	// Client Threads
	public static ClientThread[] clientThreads = new ClientThread[4];

	public static void main(String args[]) {
		// default port 2500
		try {
			serverSocket = new ServerSocket(2500);
		} catch (IOException e) {
			System.out.println(e);
		}

		// Creating sockets and threads foe each client, maximum client number
		// is 4
		do {
			try {
				clientSocket = serverSocket.accept();
				for (int i = 0; i < 4; i++) {
					if (clientThreads[i] == null) {
```

```
                                        (clientThreads[i] = new
ClientThread(clientSocket, clientThreads)).start();
                                        break;
                                    }
                                }
                            } catch (IOException e) {
                                System.out.println(e);
                            }
                        } while (true);
                }
}

// Matching clients for each thread
class ClientThread extends Thread {

        public DataInputStream inputStream;
        public PrintStream outputStream;
        public Socket clientSocket;
        public ClientThread[] threads;
        public int maxClientNumber;

        public ClientThread(Socket clientSocket, ClientThread[] threads) {
                this.clientSocket = clientSocket;
                this.threads = threads;
                maxClientNumber = threads.length;
        }

        public void run() {
                int maxClientNumber = this.maxClientNumber;
                ClientThread[] threads = this.threads;
                // Usernames and password arrays
                String[] usernameArray = { "haci", "mike", "adam", "john" };
                String[] passwordArray = { "icah", "ekim", "mada", "nhoj" };
                // Creating input and output stream for the client and ask username at
                // first login
                try {
                        inputStream = new
DataInputStream(clientSocket.getInputStream());
                        outputStream = new PrintStream(clientSocket.getOutputStream());
                        // Ask username and password
                        outputStream.println("Enter username: ");
                        String username = inputStream.readLine();
                        outputStream.println("Enter password: ");
                        String password = inputStream.readLine();

                        // Find entered username position in usernameArray
                        int index = -1;
```

```
                        for (int i = 0; i < usernameArray.length; i++) {
                                if (usernameArray[i].equals(username)) {
                                        index = i;
                                        break;
                                }
                        }

                        // Compered enterd password with the related username
password, if
                        // matched continue, else display authentication error message
                        if (passwordArray[index].equals(password)) {
                                // Echo client message to all other clients
                                do {
                                        String text = inputStream.readLine();

                                        for (int i = 0; i < maxClientNumber; i++) {
                                                if (threads[i] != null) {
                                                        threads[i].outputStream.println("<<"
+ username.toUpperCase() + " says >>: " + EncDec(text));
                                                }
                                        }
                                } while (true);
                        } else {
                                outputStream.println("Autentitcation Error");
                        }

                } catch (IOException e) {
                }
        }

        // DES Encription/Decription method
        public String EncDec(String a) {
                String s="";
                try {
                        KeyGenerator keygenerator = KeyGenerator.getInstance("DES");
                        SecretKey myKey = keygenerator.generateKey();

                        Cipher desCipher;

                        // Create the cipher
                        desCipher = Cipher.getInstance("DES/ECB/PKCS5Padding");

                        // Initialize the cipher for encryption
                        desCipher.init(Cipher.ENCRYPT_MODE, myKey);

                        // convert parameter to btye array
                        byte[] text = a.getBytes();
```

```
                    // Encrypt the text
                    byte[] textEncrypted = desCipher.doFinal(text);

                    // Initialize the same cipher for decryption
                    desCipher.init(Cipher.DECRYPT_MODE, myKey);

                    // Decrypt the text
                    byte[] textDecrypted = desCipher.doFinal(textEncrypted);

                     s = new String(textDecrypted);
                } catch (NoSuchAlgorithmException e) {
                    e.printStackTrace();
                } catch (NoSuchPaddingException e) {
                    e.printStackTrace();
                } catch (InvalidKeyException e) {
                    e.printStackTrace();
                } catch (IllegalBlockSizeException e) {
                    e.printStackTrace();
                } catch (BadPaddingException e) {
                    e.printStackTrace();
                }
                return s;
        }
}
```

```
/*CIS 675 Assignment#2
 * Part 1
 *Server Side
```

```
 *Haci Karahasanoglu
 */
import java.io.IOException;
import java.io.DataInputStream;
import java.io.PrintStream;
import java.io.BufferedReader;
import java.net.Socket;
import java.io.InputStreamReader;

public class ClientPart2 implements Runnable {

	public static Socket clientSocket;
	public static PrintStream outputStream;
	public static DataInputStream inputStream;
	public static BufferedReader input;


	public static void main(String[] args) {
		// Create a client with default host and port number
		boolean isClosed = false;
		try {
			clientSocket = new Socket("localhost", 2500);
			input = new BufferedReader(new InputStreamReader(System.in));
			outputStream = new PrintStream(clientSocket.getOutputStream());
			inputStream = new DataInputStream(clientSocket.getInputStream());
		} catch (IOException e) {
	System.out.println(e);
		}

		//Reading from server
		try {
			new Thread(new ClientPart2()).start();
			while (!isClosed) {
				outputStream.println(input.readLine());
			}
			// Closing connections
			outputStream.close();
			inputStream.close();
			clientSocket.close();
		} catch (IOException e) {
	System.out.println(e);
		}
	}

	// Thread read from the server side
	public void run() {
		String reply;
		try {
			while ((reply = inputStream.readLine()) != null) {
				if (reply.trim().equals("")) {
			// empty line
			} else {
			System.out.println(reply);
```

```
                    }
                        }
                        boolean isClosed = true;
                } catch (IOException e) {
        System.out.println(e);
                }
          }
}
```