كلية علوم الحاسب والمعلومات
College of Computer and Information Sciences

# CS330–Computer Networks Project

# TCP-based client server application


## Students Names:

Abdulelah Alhomaidhi - 441015575
Abdulaziz Almuharib - 441015350
Salman Alyahya - 441017071
Abdulmalik Bawajeeh - 441014266

**Section:** 171-172-173

## 1-Setting up the Programming Environment

We choose the java language because we have knowledge and experience of it.

## We used the following libraries:

1- java.net: which provides the classes necessary to create an applet and to run the programs on the network.

2- java.io: which provides a set of input streams and a set of output streams used to read and write data to files or other input and output services.

3- java.util.Scanner: which is provided to user entering data, then receive user input and parse them into primitive data types.

## 2-Steps for TCP socket programming for client-server connection

We took a code that start the TCP client-server connection using java and we made the changes that needed for the project like (The secure mode and the open mode… ) And we created a main menu for the program to make it more suitable for the user and more friendly.

1- The server will create the socket through these lines:

```
ServerSocket ss = new ServerSocket(3333);
Socket s = ss.accept();
DataInputStream din = new DataInputStream(s.getInputStream());
DataOutputStream dout = new DataOutputStream(s.getOutputStream());
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
```

2- The client will connect to the server's ip and port through these lines:
(Note): you should put the ip of the server inside the "localhost" parameter.

```
Socket s = new Socket("localhost", 3333);
DataInputStream din = new DataInputStream(s.getInputStream());
DataOutputStream dout = new DataOutputStream(s.getOutputStream());
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
Scanner in = new Scanner(System.in);
```

## 3-Steps for setting up the network

1- We get the IP address of the server.
2- Establish Socket by the unique port numbers.
3- Run the server and wait for the client to connect with the same port number.
4- The server will accept the client request, and finally the connection is established.

**We used separated laptops by wireless connections (LAN).**

**(At the testing Phase the client's code and the server code were on a separated hosts (as required), But on the demo video we ran the application on the same host to make the outputs more clear to the viewer.)**

# 4- Codes and comments:

## Code of server' side:

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

public class MyServer {

    public static void main(String args[]) throws Exception {

        ServerSocket ss = new ServerSocket(3333);   // starts server and waits for a connection
        Socket s = ss.accept();                     // accept requist from the right client
        DataInputStream din = new DataInputStream(s.getInputStream());  // initialize socket input stream
        DataOutputStream dout = new DataOutputStream(s.getOutputStream());  //initialize socket output stream
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

        // read message from client
        String choice;
        do {
            choice = din.readUTF();
            switch (choice) {
                case "1":
                    openMode(ss, s, din, dout, br);
                    break;
                case "2":
                    secureMode(ss, s, din, dout, br);
                    break;
                case "3":  // quite application and close connection
                    din.close();
                    s.close();
                    ss.close();
                    System.out.println("\nCnnection is closed\n");
                    break;
            }
        } while (!choice.equals("3"));
    }
```

```java
// open mode methdos that will be send back a copy of the word sent by the user

public static void openMode(ServerSocket ss, Socket s, DataInputStream din, DataOutputStream dout, BufferedReader br) throws Exception {

    String str = "";

    while (!str.equals("back")) {
        str = din.readUTF(); // read from client
        if (str.equals("back")) {
            break;
        }
        System.out.println("Client's Message:( " + str + " )\n_____\n");
        dout.writeUTF(str);
        dout.flush();
    }
}
```

```java
    // secure mode methdod that word is encrypted using a shared encryption key.
    public static void secureMode(ServerSocket ss, Socket s, DataInputStream din, DataOutputStream dout, BufferedReader br) throws Exception {

        String str = "";

        while (!str.equals("back")) {
            str = din.readUTF(); // read from client
            if (str.equals("back")) {
                break;
            }
            System.out.println("Client's Message:( " + str + " )\n_____\n");
            str = encrypt(str, 4);
            dout.writeUTF(str);
            dout.flush();
        }
    }
    // encrybtion technique (Caesar Cipher in Cryptography)

    public static String encrypt(String text, int s) {

        StringBuffer result = new StringBuffer();

        for (int i = 0; i < text.length(); i++) {
            if (Character.isUpperCase(text.charAt(i))) {
                char ch = (char) (((int) text.charAt(i)
                        + s - 65) % 26 + 65);
                result.append(ch);
            } else {
                char ch = (char) (((int) text.charAt(i)
                        + s - 97) % 26 + 97);
                result.append(ch);
            }
        }
        return result.toString();
    }
}
```

**Code for client side:**

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

public class MyClient {

    public static void main(String args[]) throws Exception {

        try {
            Socket s = new Socket("localhost", 3333);  // establish a socket connection
            DataInputStream din = new DataInputStream(s.getInputStream());  // initialize socket input stream
            DataOutputStream dout = new DataOutputStream(s.getOutputStream()); // initialize socket output stream
            BufferedReader br = new BufferedReader(new InputStreamReader(System.in)); //
            Scanner in = new Scanner(System.in);

            String Mode;
            do {
                System.out.println("\nMAIN MENU :");
                System.out.println("1) OPEN MODE");
                System.out.println("2) SECURE MODE");
                System.out.println("3) Quit");
                System.out.println("");
                System.out.print("Enter the number of choice: ");
                Mode = in.next();
                switch (Mode) {

                    case "1":
                        dout.writeUTF("1");
                        dout.flush();
                        call(s, din, dout, br, "1");
                        break;
                    case "2":
                        dout.writeUTF("2");
                        dout.flush();
                        call(s, din, dout, br, "2");
                        break;
                    case "3":
                        dout.writeUTF("3");
                        dout.flush(); // tell the server that the client wants to Quit
                        dout.close(); // close clints connection
                        s.close();// close client connection
                        System.out.println("\nCnnection is closed\n");
                        break;
                    default:
                        System.out.println("**please enter one of the valid number of the options**");
                }
            } while (!Mode.equals("3"));

        } catch (Exception e) {
            System.out.println("\n\nserver is down ):");
        }

    }
```

```java
    public static void call(Socket s, DataInputStream din, DataOutputStream dout, BufferedReader br, String choice) throws Exception {

        String str = "", str2 = "";
        while (!str.equals("back")) {
            System.out.print("Enter your text: ");
            //System.out.println("_____");
            str = br.readLine();
            dout.writeUTF(str);
            dout.flush();
            if (str.equals("back")) {
                break;
            } // return to the menu to choose another mode
            str2 = din.readUTF();

            if (choice.equals("1")) {
                System.out.println("_____");
                System.out.println("\n[Server]'s Message: " + str2);
            } // server messages in open mode
            else if (choice.equals("2")) {
                System.out.println("_____");
                System.out.println("\n[Server]'s Message(encrypted): " + str2);
            } // server messages on secure mode

            System.out.println("_____");
            System.out.println("\n(Enter the word 'back' to return to the main menu...) \n");
        }
    }
}
```

## 5- Snapshots of the application outputs.

## Output of Client's side:

```
MAIN MENU :
1) OPEN MODE
2) SECURE MODE
3) Quit

Enter the number of choice: 1
Enter your text: Hello
_____________________________________________________________

[Server]'s Message: Hello
_____________________________________________________________

(Enter the word 'back' to return to the main menu...)

Enter your text: back

MAIN MENU :
1) OPEN MODE
2) SECURE MODE
3) Quit

Enter the number of choice: 2
Enter your text: hello
_____________________________________________________________

[Server]'s Message(encrypted): lipps
_____________________________________________________________
```

**After choosing option number 3 to quit the program :**

```
(Enter the word 'back' to return to the main menu...)

Enter your text: back

MAIN MENU :
1) OPEN MODE
2) SECURE MODE
3) Quit

Enter the number of choice: 3

Cnnection is closed
```

## Output of Server's side:

```
Client's Message:( Hello )
_____________________________________

Client's Message:( hello )
_____________________________________


Cnnection is closed
```

## 6- Problems and solutions:

**Problem 1:** firstly, we faced the lack of knowledge of how the methods and classes of the libraries that we used. (e.g., java.net and java.io.)

**Solution:** we solved this problem by reading the documentation of the classes.

**Problem 2:** the second problem was when we made the main menu using the switch in java and we tried to connect the user choice (client) to the server and depend on the choice the server should call the secureMode() method or the openMode() method or quit the app.

**Solution:** we solved it by creating another switch on the server's side that reads the clients choice.

**Problem 3:** the TCP client-server connection.

**Solution:** creating a socket at the server first and specifying a unique port for it then connecting the client to the socket using the servers IP and port.

(**We mentioned the reference that helped us with this problem at References section**)

**Problem 4:** how to encrypt messages from client to server.

**Solution:** we used Caesar cipher in cryptography.
(**We mentioned the reference that helped us with this problem at References section**)

**References:**

https://www.javatpoint.com/socket-programming

https://www.geeksforgeeks.org/caesar-cipher-in-cryptography/?ref=leftbar-rightbar

## Link to the demo video:

## https://www.youtube.com/watch?v=bgD43bIVWOc