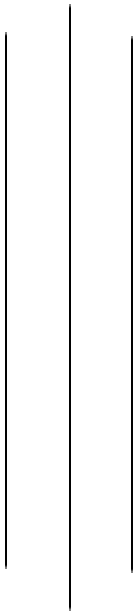


Programme Name: _____BCS (Hons.)_____

Course Code: __CSC1513___

Course Name: _____Programming Fundamentals_____

Assignment / Lab Sheet / **Project** / Case Study No. _1__

Date of Submission: ___2021/05/15_____

Submitted By:

Student Name: Shishir Aryal

IUKL ID: 042003900004

Semester: 2nd

Intake: September, 2020

Submitted To:

Faculty Name: Prakash Chandra

Department: PO (BCS)

CSC 1513 Project Documentation

15.05.2021

A Simple Multiclient Chat Application

Table of Content

Chapter 1: Abstract

A simple cross-platform multi-client single server-based server chat application that can also share files has been implemented in Java. The chat application has various functionalities that can be accessed using different commands which are explained in this documentation.

Background

Implementing a chat server application provides a great opportunity for a beginner to design and implement a network-based system. Lots of free source code is available on the web. Excellent and highly configurable applications are available both as open-source as well as proprietary software. With little experience in network programming as well as a short duration for the projection, my intention was not to match or improve an existing implementation but to create and implement a basic version on my own.

Design

Socket and ServerSocket classes are used for connection-oriented socket programming and DatagramSocket and DatagramPacket classes are used for connection-less socket programming. TCP which is a connection-oriented protocol is used as a transport-layer protocol in this project since it provides reliable delivery which is critical for the given application. TCP is not as fast as UDP but is not an issue since this is a chat application.

Chapter 2: Introduction

In order to create a chat application that execute across multiple devices in a network, we need to use Network Programming. The java.net package contains a collection of classes and interfaces that provide us all the required low-level communication details, allowing us to create this application. The java.net package support two communication protocol TCP and UDP and as discusses previously, we are using TCP which is best for our usecase scenario.

There are two kinds of TCP sockets in Java. One is for servers, and the other is for clients. The ServerSocket class is designed to be a "listener," which waits for clients to connect before doing anything. Thus, ServerSocket is for servers. The Socket class is for clients.

A socket is simply an endpoint for communications between the machines. It provide the communication mechanism between two computers using TCP. The Socket class can be used to create a socket. The ServerSocket class can be used to create a server socket. This object is used to establish communication with the clients.To create the server application, we need to create the instance of ServerSocket class. In this program, we are using port 8818 for the communication between the client and server. You may also choose any other port number (bigger than 1000). The accept() method waits for the client. If clients connects with the given port number, it returns an instance of Socket (clientSocket in this program).

```
try {
    ServerSocket serverSocket = new ServerSocket(serverPort);
    while(true) {
        System.out.println("\nListening on port " + serverPort +"...");
        Thread.sleep(1000);
        System.out.println("Accepting client connection...");
        Socket clientSocket = serverSocket.accept();
        System.out.println("Accepted connection from " + clientSocket);
```

There are various Socket class methods that are very important to know about while creating any application using Socket Programming. Some of them are:

1. public InputStream getInputStream() -> returns the InputStream attached with this socket.
2. public OutputStream getOutputStream() ->returns the OutputStream attached with this socket.
3. public synchronized void close() -> closes this socket

You can gain access to the input and output streams associated with a Socket by use of the getInputStream() and getOuptutStream() methods, as shown here. Each can throw an IOException if the socket has been invalidated by a loss of connection

Several other methods are available, including connect( ), which allows you to specify a new connection; isConnected( ), which returns true if the socket is connected to a server; isBound( ), which returns true if the socket is bound to an address; and isClosed( ), which returns true if the socket is closed. To close a socket, call close( ). Closing a socket also closes the I/O streams associated with the socket. Beginning with JDK 7, Socket also implements AutoCloseable, which means that you can use a try-with-resources block to manage a socket.

For ServerSocket:

1. public Socket accept() -> returns the socket and establish a connection between server and client.
2. public synchronized void close() -> closes the server socket.

This application is made up of 3 class (ServerMain, Server and ServerWorker). ServerMain class starts the server, then the Server class starts listening on a specific port and connects to the clients. When a new client gets connected, an instance of ServerWorker is created to serve that client. Since each connection is processed in a separate thread, the server can handle multiple clients at the same time.

Server

The server is divided into two classes, ServerMain and Server Class where ServerMain is the main class. It specifies the listening port of the server and calls the Server class. Server class extends a thread. Sever class creates a Server Socket and accepts all incoming client connections. Another important task of the Server class is to store all the client information inside an Arraylist (workerList). A ServerWorker object is created which passes the Server and ClientSocket to the ServerWorker class. Various strings like “Listening on port”, “Accepting client connections”, etc are displayed to make the program more user-friendly. The removeWorker is created to remove inactive/offlline users from the workerList.

Client

The client is implemented as a singleton class (ServerWorker). There is one ServerWorker thread for each client connection. ServerWorker handles all the logins, commands, User inputs, and outputs and uses clientSocket (Socket) for connecting with the server. There are various methods inside the ServerWorker class like handleclientSocket, removeWorker, handleLeave,

isMemberofTopic, handleJoin, handleMessage, handleLogoff, and handleLogin which are called using specific commands and they all do tasks similar to their names.

The handleClientSocket method accesses the input and output stream of the clientSocket. A buffer reader stream is also created to read the commands from the user console. A string token is then created to separate the commands from the user message based on the index. The commands should always be delivered at index 0. Different methods are called based on the command user provides. For. eg: login command calls handleLogin method.

The handleLogin method handles user login. It checks whether the given username and password match the given username and password. If the username and password are correct, the user is permitted to log in and the user login information is stored in the login string. Also, the online status of the user is sent to all other users.

The handleMessage method as the name suggests handles one-to-one and group user messages. It is triggered by the msg command. The first word is the msg command then comes the username of whom you want to send the message and the other remaining is the message body that is sent to the required user. In the case of group messaging the second parameter contains the group name (#groupname) instead of the username.

The handleJoin method creates and stores groups and is triggered by the join command. For. eg join #programming_group creates a group named programming_group and adds the user to the group. Other users who use the same command are also added to the group to see the group messages.

The handleLeave method lets users leave the group and is triggered by the leave command. For. eg: leave #programming_group, removes the user from the programming_group.

The handleLogoff method handles user logout. The offline status of the user is then sent to all other users connected to the server.

Now, last but not the list is to download a file from the server. To do it, just write the command “dfile” in the User’s terminal. You’ll then be asked the path of the file you want to download and the target destination and the file will be downloaded from the server and saved in the given destination folder.

Chapter 3: Testing

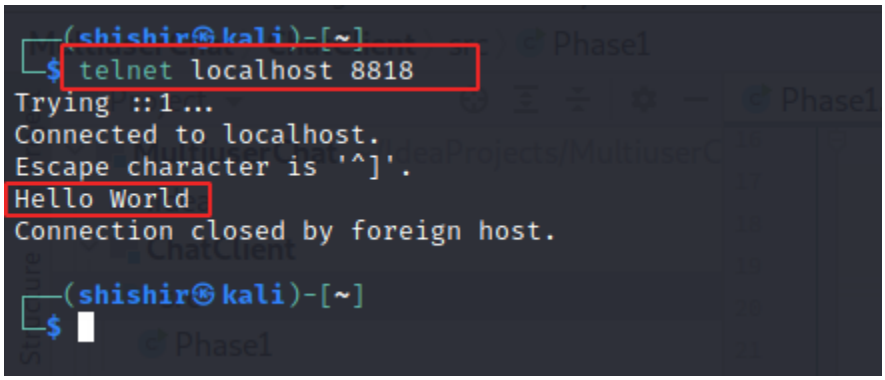
The project is developed in various phases: Phase 1.1, 1.2 1.3, 2.1, 2.2, 2.3, and then the final application. The reason behind creating the project is phases was to simplify the testing phase and make sure that everything worked properly.

In Phase 1 the base version of the application is made which can connect multiple clients to the server where the clients can only do things specified in the server and can not interact with each other.

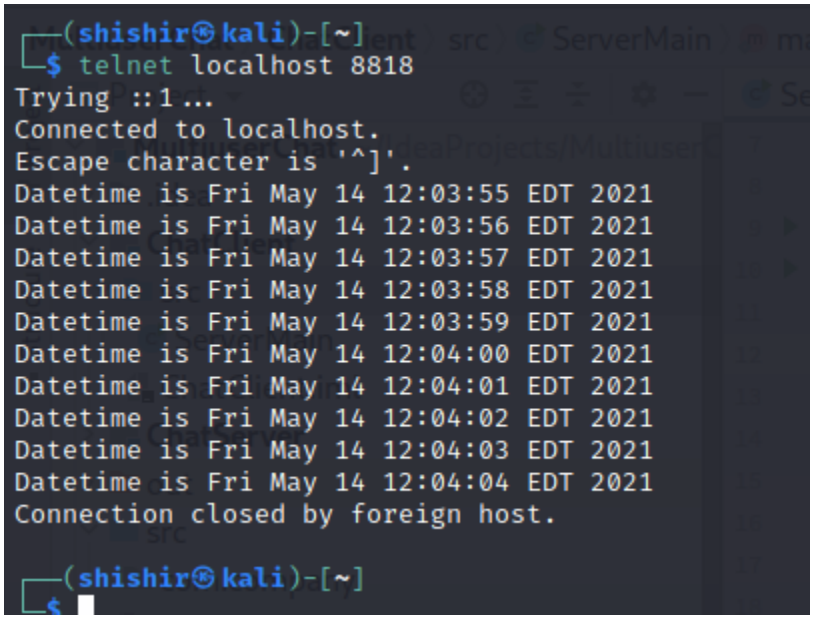
In Phase 2, interactive functionality which includes user authentication, use of commands, direct and group messaging has been added to the clients, and now a user's online status is broadcasted to other users connected with the server. In short, at the end of phase 2, a full chat application without file sharing is created.

In the final phase, file sharing functionality is added and the application is completed.

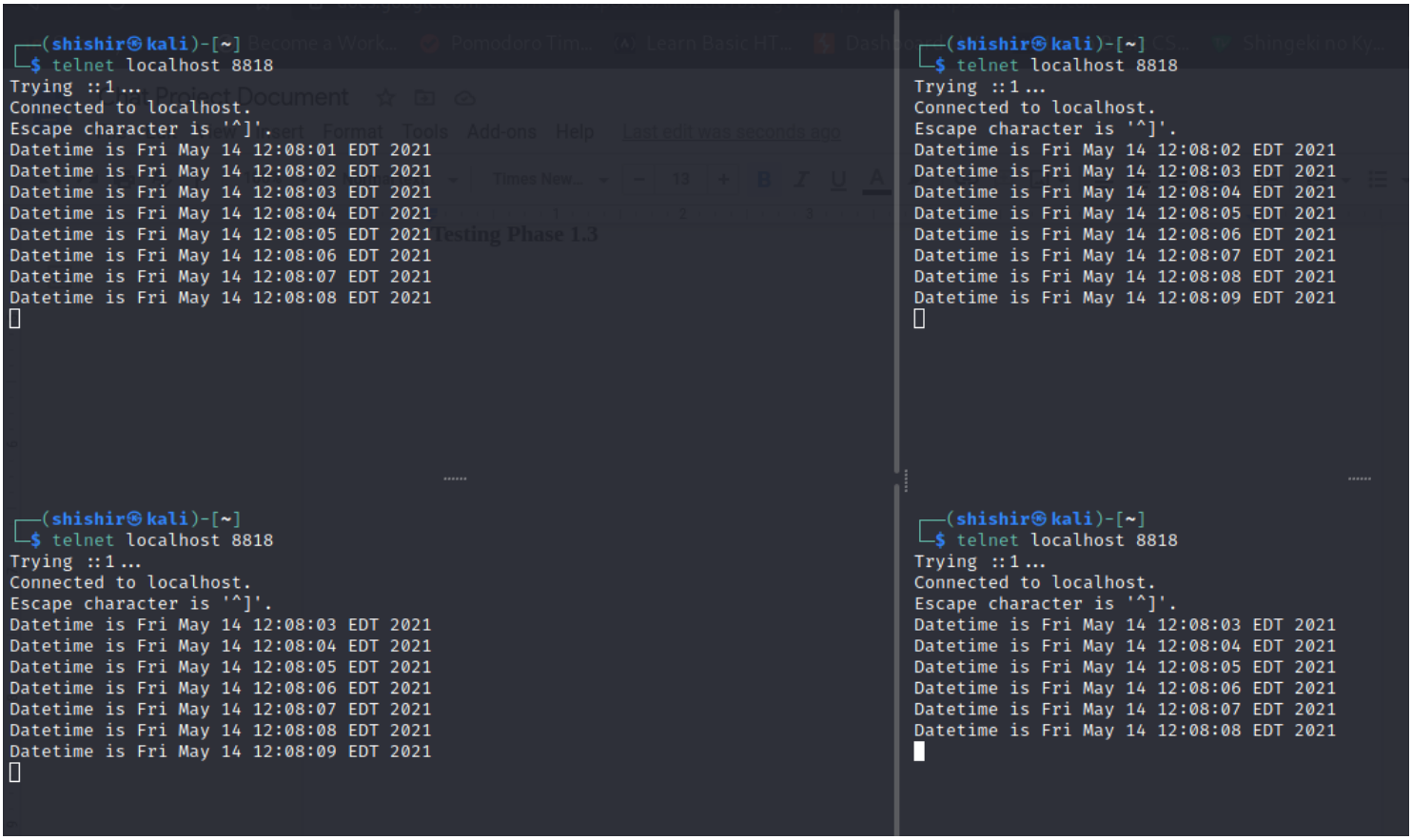
Testing Phase 1.1



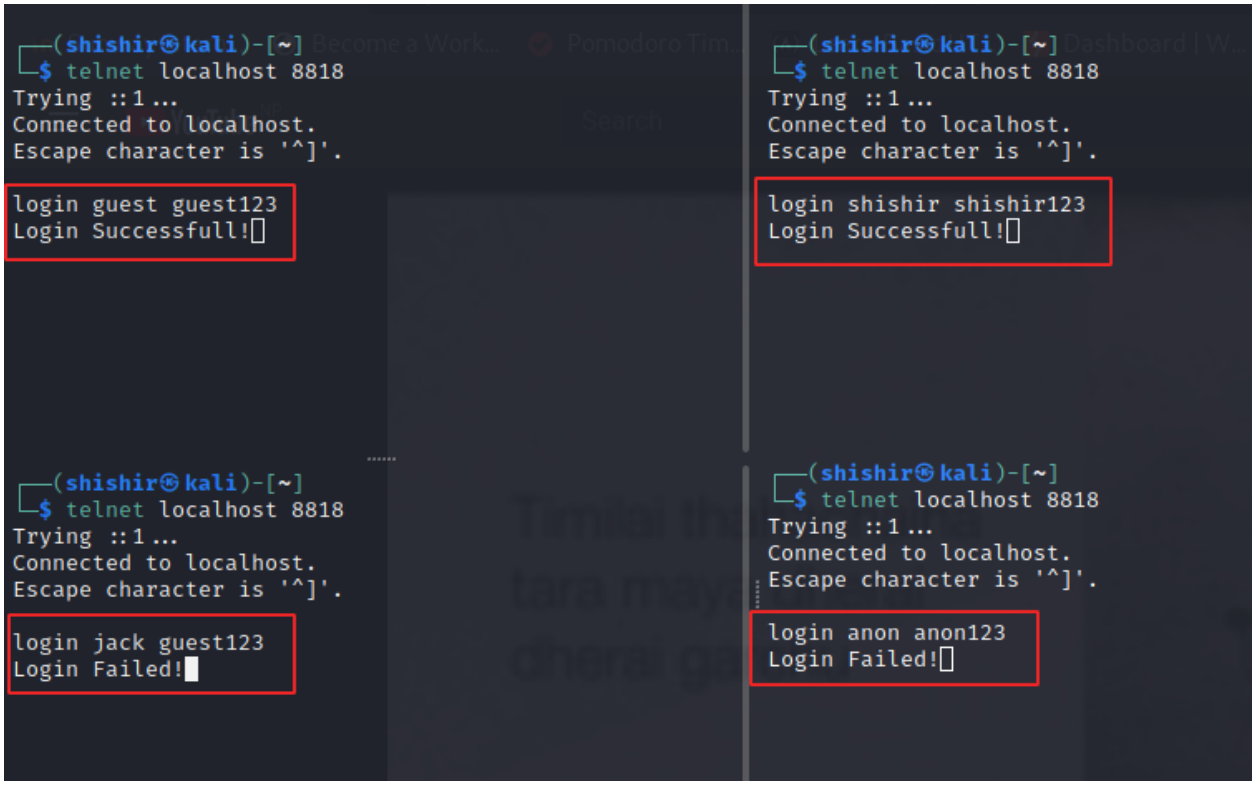
Testing Phase 1.2



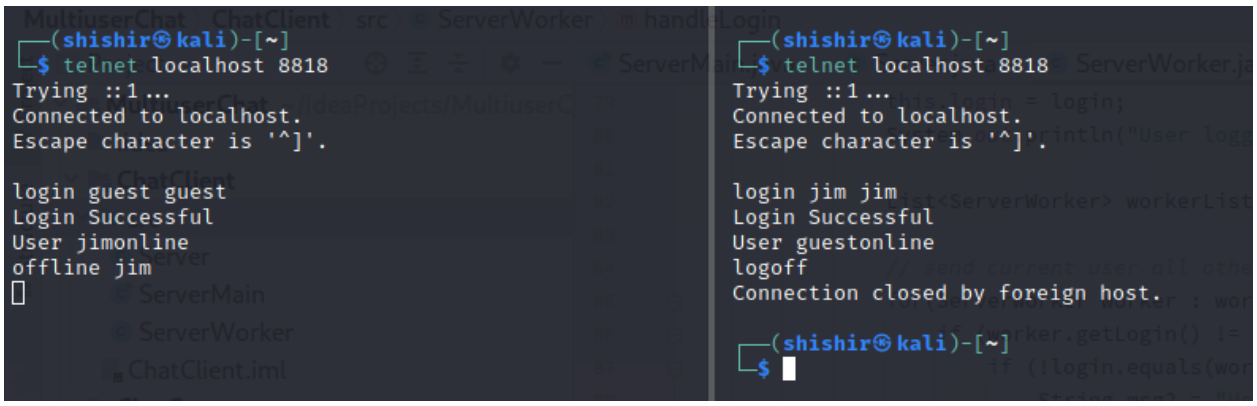
Testing Phase 1.3



Testing Phase 2.1

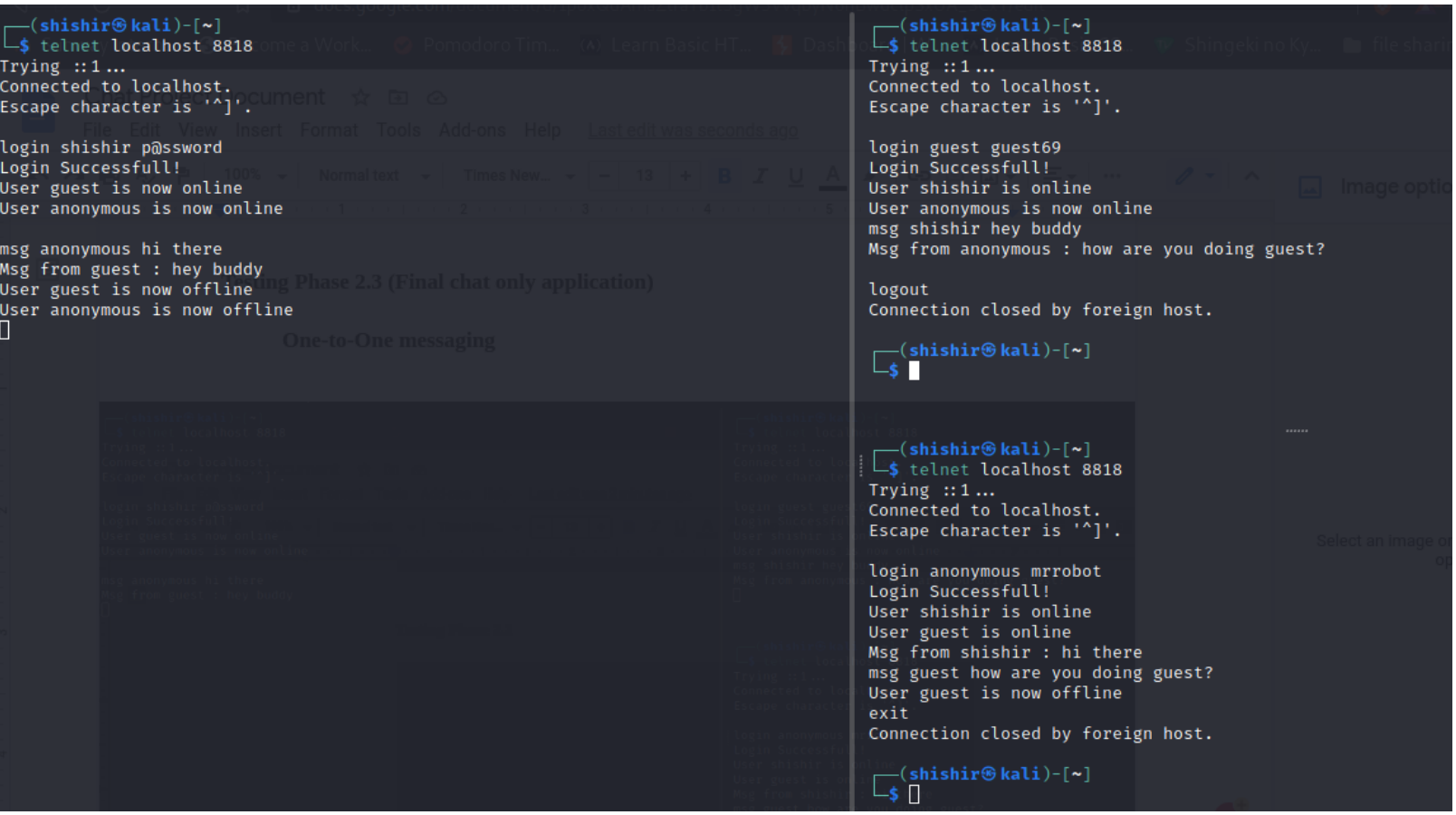


Testing Phase 2.2



Testing Phase 2.3 (Final chat only application)

One-to-One messaging

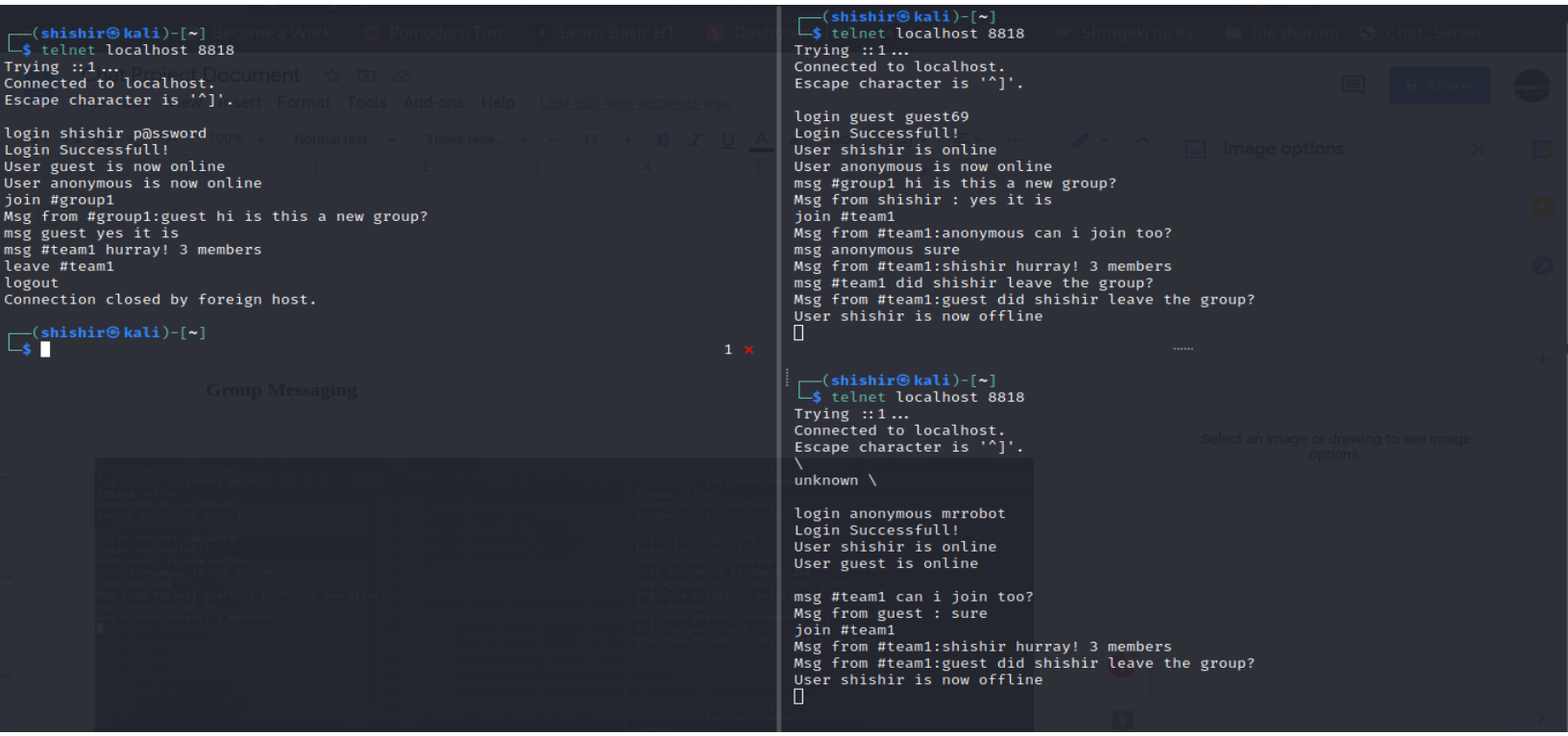


```
┌──(shishir㉿kali)-[~]
└─$ telnet localhost 8818
Trying ::1...
Connected to localhost.
Escape character is '^]'.

login shishir p@ssword
Login Successfull!
User guest is now online
User anonymous is now online

msg anonymous hi there
Msg from guest : hey buddy
User guest is now offline
User anonymous is now offline
```

```
┌──(shishir㉿kali)-[~]
└─$ telnet localhost 8818
Trying ::1...
Connected to localhost.
Escape character is '^]'.

login guest guest69
Login Successfull!
User shishir is online
User anonymous is now online
msg shishir hey buddy
Msg from anonymous : how are you doing guest?

logout
Connection closed by foreign host.

┌──(shishir㉿kali)-[~]
└─$
```

```
┌──(shishir㉿kali)-[~]
└─$ telnet localhost 8818
Trying ::1...
Connected to localhost.
Escape character is '^]'.

login anonymous mrrobot
Login Successfull!
User shishir is online
User guest is online
Msg from shishir : hi there
msg guest how are you doing guest?
User guest is now offline
exit
Connection closed by foreign host.

┌──(shishir㉿kali)-[~]
└─$
```

Group Messaging

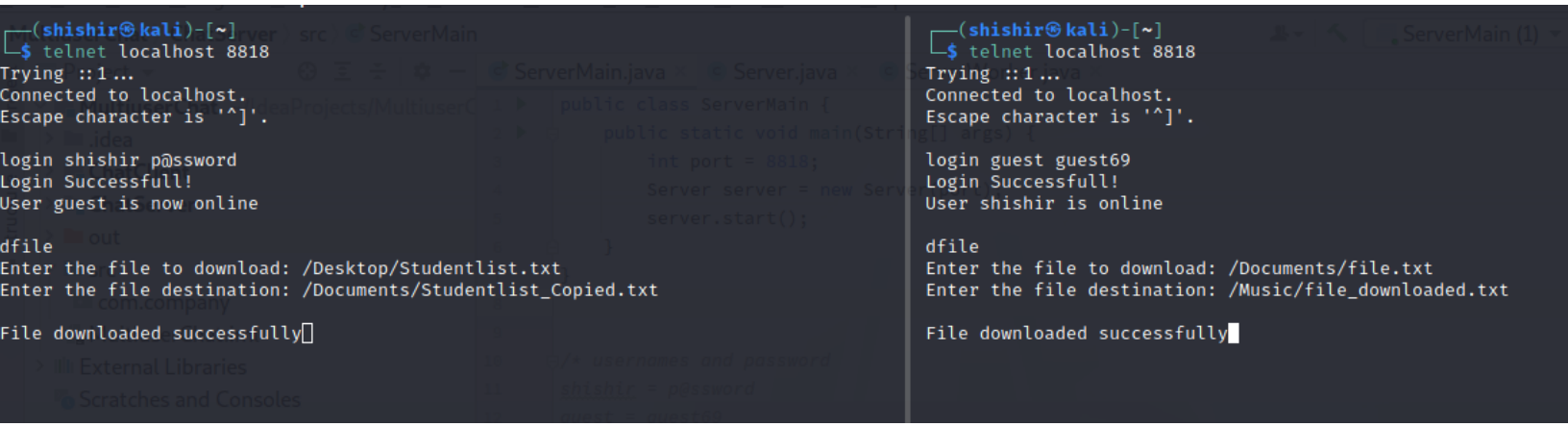


```
┌──(shishir㉿kali)-[~]
└─$ telnet localhost 8818
Trying ::1...
Connected to localhost.
Escape character is '^]'.

login shishir p@ssword
Login Successfull!
User guest is now online
User anonymous is now online
join #group1
Msg from #group1:guest hi is this a new group?
msg guest yes it is
msg #team1 hurray! 3 members
leave #team1
logout
Connection closed by foreign host.

┌──(shishir㉿kali)-[~]
└─$
```

```
┌──(shishir㉿kali)-[~]
└─$ telnet localhost 8818
Trying ::1...
Connected to localhost.
Escape character is '^]'.

login guest guest69
Login Successfull!
User shishir is online
User anonymous is now online
msg #group1 hi is this a new group?
Msg from shishir : yes it is
join #team1
Msg from #team1:anonymous can i join too?
msg anonymous sure
Msg from #team1:shishir hurray! 3 members
msg #team1 did shishir leave the group?
Msg from #team1:guest did shishir leave the group?
User shishir is now offline
```

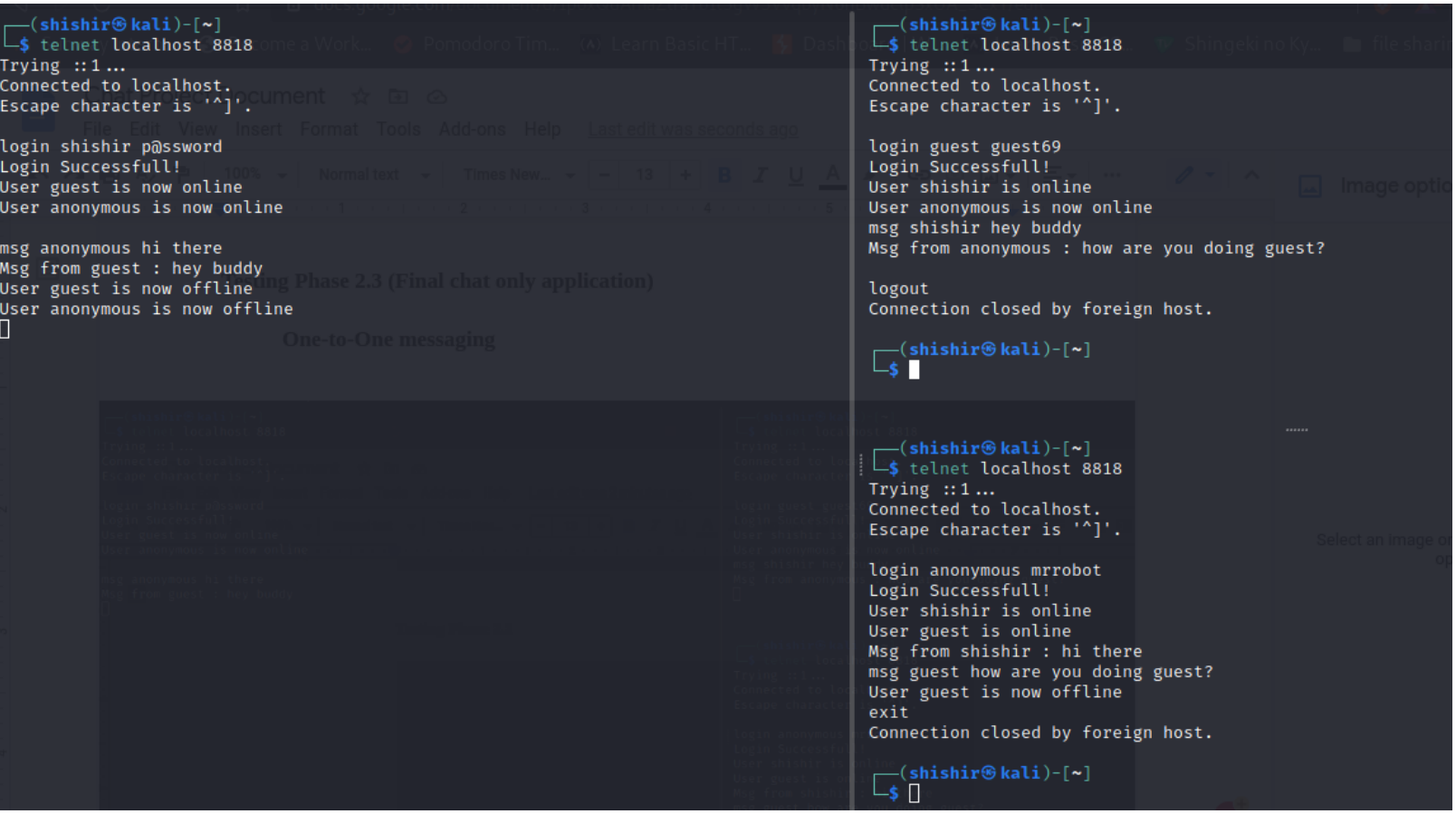
```
┌──(shishir㉿kali)-[~]
└─$ telnet localhost 8818
Trying ::1...
Connected to localhost.
Escape character is '^]'.
\
unknown \

login anonymous mrrobot
Login Successfull!
User shishir is online
User guest is online

msg #team1 can i join too?
Msg from guest : sure
join #team1
Msg from #team1:shishir hurray! 3 members
Msg from #team1:guest did shishir leave the group?
User shishir is now offline
```

Final Application

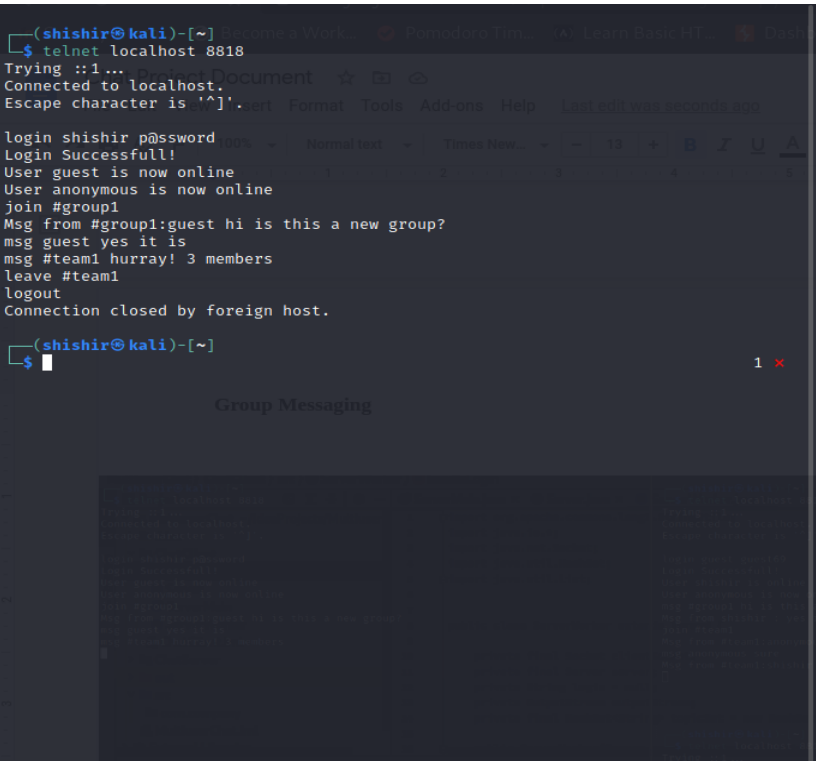


Chapter 4: Output

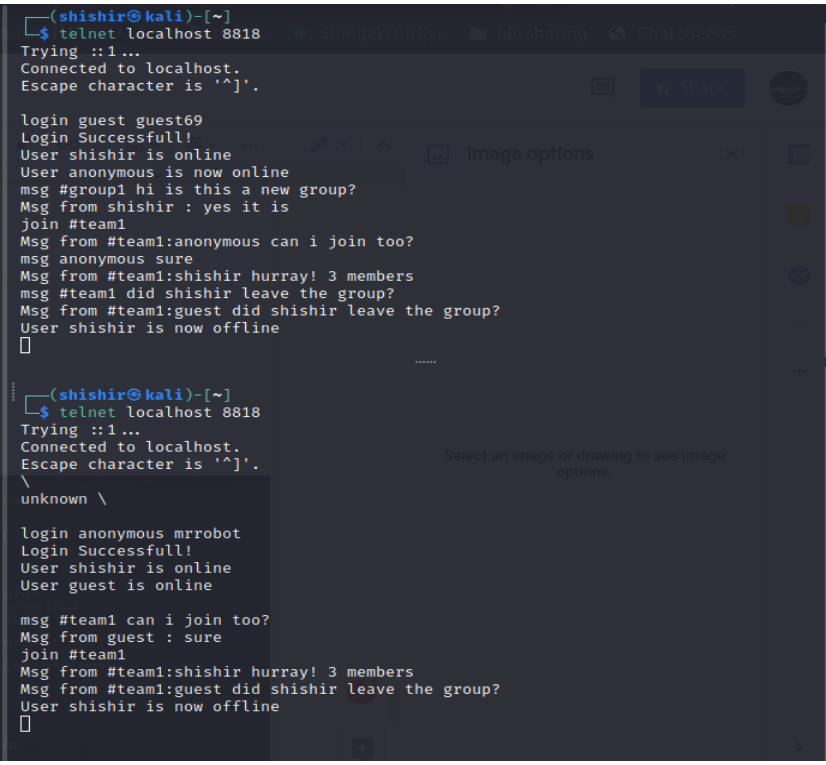
One-to-One messaging



Group Messaging / Broadcasting

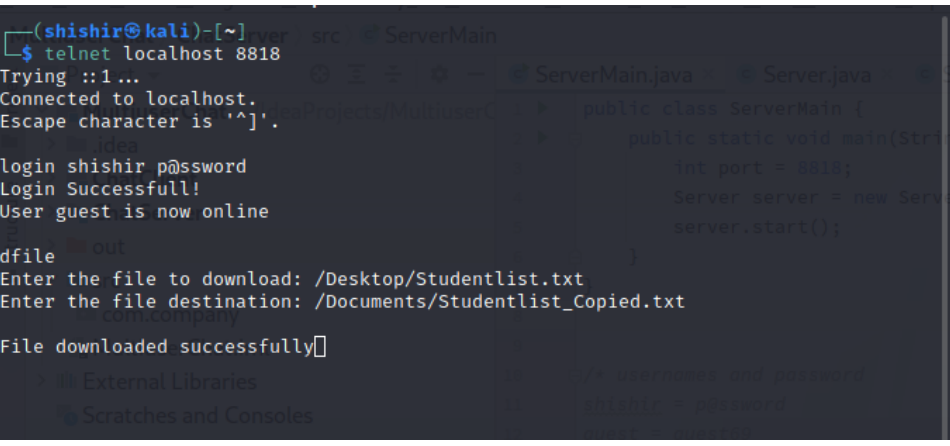


```
┌──(shishir㉿kali)-[~]
└─$ telnet localhost 8818
Trying ::1...
Connected to localhost.
Escape character is '^]'.

login shishir p@ssword
Login Successfull!
User guest is now online
User anonymous is now online
join #group1
Msg from #group1:guest hi is this a new group?
msg guest yes it is
msg #team1 hurray! 3 members
leave #team1
logout
Connection closed by foreign host.

┌──(shishir㉿kali)-[~]
└─$
```

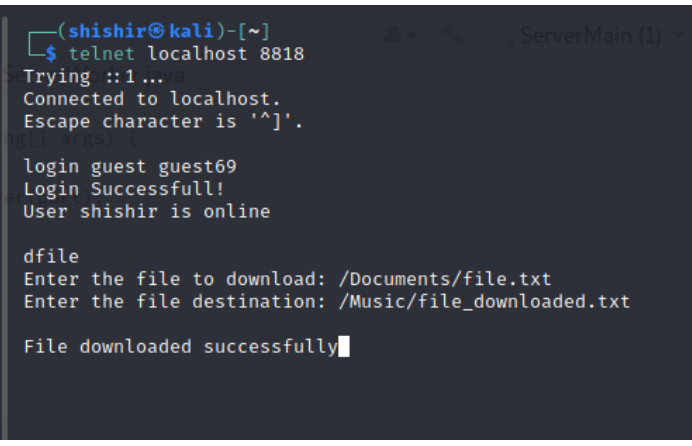


```
┌──(shishir㉿kali)-[~]
└─$ telnet localhost 8818
Trying ::1...
Connected to localhost.
Escape character is '^]'.

login guest guest69
Login Successfull!
User shishir is online
User anonymous is now online
msg #group1 hi is this a new group?
Msg from shishir : yes it is
join #team1
Msg from #team1:anonymous can i join too?
msg anonymous sure
Msg from #team1:shishir hurray! 3 members
msg #team1 did shishir leave the group?
Msg from #team1:guest did shishir leave the group?
User shishir is now offline

┌──(shishir㉿kali)-[~]
└─$ telnet localhost 8818
Trying ::1...
Connected to localhost.
Escape character is '^]'.
\
unknown \

login anonymous mrrobot
Login Successfull!
User shishir is online
User guest is online

msg #team1 can i join too?
Msg from guest : sure
join #team1
Msg from #team1:shishir hurray! 3 members
Msg from #team1:guest did shishir leave the group?
User shishir is now offline
```

File sharing



```
┌──(shishir㉿kali)-[~]
└─$ telnet localhost 8818
Trying ::1...
Connected to localhost.
Escape character is '^]'.

login shishir p@ssword
Login Successfull!
User guest is now online

dfile
Enter the file to download: /Desktop/Studentlist.txt
Enter the file destination: /Documents/Studentlist_Copied.txt

File downloaded successfully
```



```
┌──(shishir㉿kali)-[~]
└─$ telnet localhost 8818
Trying ::1...
Connected to localhost.
Escape character is '^]'.

login guest guest69
Login Successfull!
User shishir is online

dfile
Enter the file to download: /Documents/file.txt
Enter the file destination: /Music/file_downloaded.txt

File downloaded successfully
```

Appendix

ServerMain.java

```
public class ServerMain {
    public static void main(String[] args) {
        int port = 8818;
        Server server = new Server(port);
        server.start();
    }
}


/* usernames and password
shishir = p@ssword
guest = guest69
anonymous = mrrobot
 */
```

Server.java

```
import java.io.*;
import java.net.ServerSocket;
import java.net.Socket;
import java.nio.charset.StandardCharsets;
import java.util.ArrayList;
import java.util.List;

public class Server extends Thread {
    private final int serverPort;

    private ArrayList<ServerWorker> workerList = new ArrayList<>();

    public Server(int serverPort) {
```

```
        this.serverPort = serverPort;
    }

    public List<ServerWorker> getWorkerList() {
        return workerList;
    }

    @Override
    public void run() {
        try {
            ServerSocket serverSocket = new ServerSocket(serverPort);
            while(true) {
                System.out.println("\nListening on port " +
serverPort +"...");
                Thread.sleep(1000);
                System.out.println("Accepting client connection...");
                Socket clientSocket = serverSocket.accept();
                System.out.println("Accepted connection from " +
clientSocket);
                ServerWorker worker = new ServerWorker(this,
clientSocket);
                workerList.add(worker);
                worker.start();









            }
        } catch (IOException | InterruptedException e) {
            e.printStackTrace();
        }
    }

    public void removeWorker(ServerWorker serverWorker) {
        workerList.remove(serverWorker);
    }

    public static void sendFile(Socket clientSocket) throws
```

```
IOException {
        OutputStream os = clientSocket.getOutputStream();

        //file sharing

        //getting the file to download and file destination

        os.write(("Enter the file to download:
").getBytes(StandardCharsets.UTF_8));
        InputStream inputStream = clientSocket.getInputStream();
        BufferedReader reader = new BufferedReader(new
InputStreamReader(inputStream));
        String filetodownload;
        filetodownload = "/home/shishir";
        filetodownload += reader.readLine();

        os.write(("Enter the file destination:
").getBytes(StandardCharsets.UTF_8));
        BufferedReader reader1 = new BufferedReader(new
InputStreamReader(inputStream));
        String filedestination;
        filedestination = "/home/shishir";
        filedestination += reader.readLine();



        //Specify the file
        File file = new File(filetodownload);
        FileInputStream fis = new FileInputStream(file);
        BufferedInputStream bis = new BufferedInputStream(fis);
        //Get socket's output stream
        //Read File Contents into contents array
        byte[] contents;
        long fileLength = file.length();
        long current = 0;
        long start = System.nanoTime();
        while(current!=fileLength){
            int size = 10000;
            if(fileLength - current >= size)
```

```
                current += size;
            else{
                size = (int)(fileLength - current);
                current = fileLength;
            }
            contents = new byte[size];
            bis.read(contents, 0, size);
//            os.write(contents);
            FileOutputStream fos = new
FileOutputStream(filedestination);
            byte[] s = contents;
            fos.write(s);
            os.write(("\nFile downloaded
successfully").getBytes(StandardCharsets.UTF_8));
            System.out.print("Sending file ...
"+(current*100)/fileLength+"% complete!");
        }

        System.out.println("\nFile sent succesfully!");

    }

}
```

ServerWorker.java

```
//handles clients //One ServerWorker for each client

import org.apache.commons.lang3.StringUtils;
import java.io.*;
import java.net.Socket;
import java.util.HashSet;
import java.util.List;


public class ServerWorker extends Thread {

    private final Socket clientSocket;
    private final Server server;
    private String login = null;
    private OutputStream outputStream;
    private final HashSet<String> topicSet = new HashSet<>();

    public ServerWorker(Server server, Socket clientSocket) {
        this.server = server;
        this.clientSocket = clientSocket;
    }

    @Override
    public void run() {
        try {
            handleClientSocket();
        } catch (IOException | InterruptedException e) {
            e.printStackTrace();
        }
    }

    private void handleClientSocket() throws IOException,
InterruptedException {
        InputStream inputStream = clientSocket.getInputStream();
        this.outputStream = clientSocket.getOutputStream();

        BufferedReader reader = new BufferedReader(new
```

```
InputStreamReader(inputStream));
        String line;
        while ((line = reader.readLine()) != null) {
            String[] tokens = StringUtils.split(line);
            if (tokens != null && tokens.length > 0) {
                String cmd = tokens[0];
                if ("logout".equals(cmd) ||
"exit".equalsIgnoreCase(cmd)) {
                    handleLogoff();
                    break;
                } else if ("login".equalsIgnoreCase(cmd)) {
                    handleLogin(outputStream, tokens);
                } else if ("msg".equalsIgnoreCase(cmd)) {
                    String[] tokensMsg = StringUtils.split(line,
null, 3);
                    handleMessage(tokensMsg);
                } else if ("join".equalsIgnoreCase(cmd)) {
                    handleJoin(tokens);
                } else if ("dfile".equalsIgnoreCase(cmd)) {
                    Server.sendFile(clientSocket);

                } else if ("leave".equalsIgnoreCase(cmd)) {
                    handleLeave(tokens);
                } else {
                    String msg = "unknown " + cmd + "\n";
                    outputStream.write(msg.getBytes());
                }
            }
        }

        clientSocket.close();
    }

    private void handleLeave(String[] tokens) {
        if (tokens.length > 1) {
            String topic = tokens[1];
            topicSet.remove(topic);
        }
    }
```

```
    public boolean isMemberOfTopic(String topic) {
        return topicSet.contains(topic);
    }

    private void handleJoin(String[] tokens) {
        if (tokens.length > 1) {
            String topic = tokens[1];
            topicSet.add(topic);
        }
    }

    // format: "msg" "login" body...
    // format: "msg" "#topic" body...
    private void handleMessage(String[] tokens) throws IOException {
        String sendTo = tokens[1];
        String body = tokens[2];

        boolean isTopic = sendTo.charAt(0) == '#';

        List<ServerWorker> workerList = server.getWorkerList();
        for (ServerWorker worker : workerList) {
            if (isTopic) {
                if (worker.isMemberOfTopic(sendTo)) {
                    String outMsg = "Msg from " + sendTo + ":" +
login + " " + body + "\n";
                    worker.send(outMsg);
                }
            } else {
                if (sendTo.equalsIgnoreCase(worker.getLogin())) {
                    String outMsg = "Msg from " + login + " : " +
body + "\n";
                    worker.send(outMsg);
                }
            }
        }
    }

    private void handleLogoff() throws IOException {
```

```
        server.removeWorker(this);
        List<ServerWorker> workerList = server.getWorkerList();

        // send other online users current user's status
        String onlineMsg = "User " + login + " is now offline\n";
        for (ServerWorker worker : workerList) {
            if (!login.equals(worker.getLogin())) {
                worker.send(onlineMsg);
            }
        }
        clientSocket.close();
    }

    public String getLogin() {
        return login;
    }

    private void handleLogin(OutputStream outputStream, String[]
tokens) throws IOException {
        if (tokens.length == 3) {
            String login = tokens[1];
            String password = tokens[2];

            if ((login.equals("guest") && password.equals("guest69"))
|| (login.equals("shishir") && password.equals("p@ssword")) ||
(login.equals("anonymous") && password.equals("mrrobot"))) {
                //|| (login.equals("anonymous") &&
password.equals("mrrobot"))
                String msg = "Login Successfull!\n";
                outputStream.write(msg.getBytes());
                this.login = login;
                System.out.println("User logged in succesfully: " +
login);


                List<ServerWorker> workerList =
server.getWorkerList();

                // send current user all other online logins
```

```
                for (ServerWorker worker : workerList) {
                    if (worker.getLogin() != null) {
                        if (!login.equals(worker.getLogin())) {
                            String msg2 = "User " + worker.getLogin()
+ " is online\n";
                            send(msg2);
                        }
                    }
                }

                // send other online users current user's status
                String onlineMsg = "User " + login + " is now
online\n";
                for (ServerWorker worker : workerList) {
                    if (!login.equals(worker.getLogin())) {
                        worker.send(onlineMsg);
                    }
                }
            } else {
                String msg = "Error login\n";
                outputStream.write(msg.getBytes());
            }
        }
    }

    private void send(String msg) throws IOException {
        if (login != null) {
            outputStream.write(msg.getBytes());
        }
    }
}
```

References

1. https://www.javatpoint.com/socket-programming
2. https://www.geeksforgeeks.org/socket-programming-in-java/
3. https://docs.oracle.com/javase/7/docs/api/java/net/Socket.html
4. https://docs.oracle.com/javase/7/docs/api/java/net/ServerSocket.html
5. https://github.com/ttaomae/Chat
6. https://fullstackmastery.com/page/5/how-build-multiuser-chat-application-in-java
7. https://stackoverflow.com/questions/22916693/java-7-files-copy-copying-empty-file
8. https://github.com/ttaomae/Chat
9. https://www.programmersought.com/article/351126733/
10. https://manikarea.home.blog/2019/12/06/transfer-file-from-client-to-server-using-java-socket-programming-in-localhost/
11. https://programming.vip/docs/java-using-tcp-to-upload-files.html
12. https://www.rgagnon.com/javadetails/java-0542.html
13. https://srikarthiks.files.wordpress.com/2019/07/file-transfer-using-tcp.pdf