## Lab Assignment – 6

### **Name:** Srujan Patwardhan

### **STD:** TY-D

### **Roll no.:** 02

### **PRN:** 12210847

### **Subject:** Computer Networks

#### Q1. Write a program using TCP Berkeley socket primitives for wired /wireless network for following a. Say Hello to Each other b. File transfer c. Calculator (Arithmetic) d. Calculator (Trigonometry) Demonstrate the packets captured traces using Wireshark/Fiddler for traffic analysis tool in peer-to-peer mode.

### Arithmetic and Trigonometry Calculator:

### CODE:

#### CalculatorServer.java:

import *java.io.\**;

import *java.net.\**;

import *java.util.StringTokenizer*;

*class* CalculatorServer {

*public* *static* void main(String[] args) {

        int port = 5000;

        try (ServerSocket serverSocket = new ServerSocket(port)) {

            System.out.println("Server started and listening on port " + port);

            Socket clientSocket = serverSocket.accept();

            System.out.println("Client connected");

            BufferedReader in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

            PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);

            String clientMessage;

            while ((clientMessage = in.readLine()) != null) {

                System.out.println("Received from client: " + clientMessage);

*// Process the client's request*

                String serverResponse = processRequest(clientMessage);

                out.println(serverResponse);  *// Send the result back to the client*

            }

            clientSocket.close();

            System.out.println("Client disconnected");

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

*// Process client request and return the result*

*private* *static* String processRequest(String request) {

        StringTokenizer tokenizer = new StringTokenizer(request);

        String command = tokenizer.nextToken();

        switch (command.toLowerCase()) {

            case "add":

                double sum = Double.parseDouble(tokenizer.nextToken()) + Double.parseDouble(tokenizer.nextToken());

                return "Result: " + sum;

            case "sub":

                double difference = Double.parseDouble(tokenizer.nextToken()) - Double.parseDouble(tokenizer.nextToken());

                return "Result: " + difference;

            case "mul":

                double product = Double.parseDouble(tokenizer.nextToken()) \* Double.parseDouble(tokenizer.nextToken());

                return "Result: " + product;

            case "div":

                double divisor = Double.parseDouble(tokenizer.nextToken());

                if (divisor == 0) return "Error: Division by zero!";

                double quotient = Double.parseDouble(tokenizer.nextToken()) / divisor;

                return "Result: " + quotient;

            case "sin":

                double sinValue = Math.sin(Math.toRadians(Double.parseDouble(tokenizer.nextToken())));

                return "Result: " + sinValue;

            case "cos":

                double cosValue = Math.cos(Math.toRadians(Double.parseDouble(tokenizer.nextToken())));

                return "Result: " + cosValue;

            case "tan":

                double tanValue = Math.tan(Math.toRadians(Double.parseDouble(tokenizer.nextToken())));

                return "Result: " + tanValue;

            default:

                return "Error: Unsupported operation!";

        }

    }

}

#### CalculatorClient.java:

import *java.io.\**;

import *java.net.\**;

*class* CalculatorClient {

*public* *static* void main(String[] args) {

        String serverAddress = "192.168.1.4"; *// Server address*

        int port = 5000; *// Port to connect*

        try (Socket socket = new Socket(serverAddress, port)) {

            BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

            PrintWriter out = new PrintWriter(socket.getOutputStream(), true);

            BufferedReader stdIn = new BufferedReader(new InputStreamReader(System.in));

            System.out.println("Connected to server");

            String userInput;

            while (true) {

                displayMenu();

*// Get input from user*

                System.out.print("Enter command: ");

                userInput = stdIn.readLine();

*// If user types 'exit', break the loop*

                if (userInput.equalsIgnoreCase("exit")) {

                    break;

                }

*// Send user input to the server*

                out.println(userInput);

*// Receive and display response from the server*

                String response = in.readLine();

                System.out.println("Server response: " + response);

            }

            socket.close();

            System.out.println("Disconnected from server");

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

*// Method to display available arithmetic and trigonometric functions*

*private* *static* void displayMenu() {

        System.out.println("\nAvailable commands:");

        System.out.println("   - add <num1> <num2>  ");

        System.out.println("   - sub <num1> <num2>  ");

        System.out.println("   - mul <num1> <num2>  ");

        System.out.println("   - div <num1> <num2>  ");

        System.out.println("   - sin <angle>       ");

        System.out.println("   - cos <angle>       ");

        System.out.println("   - tan <angle>      ");

        System.out.println("   - exit  ");

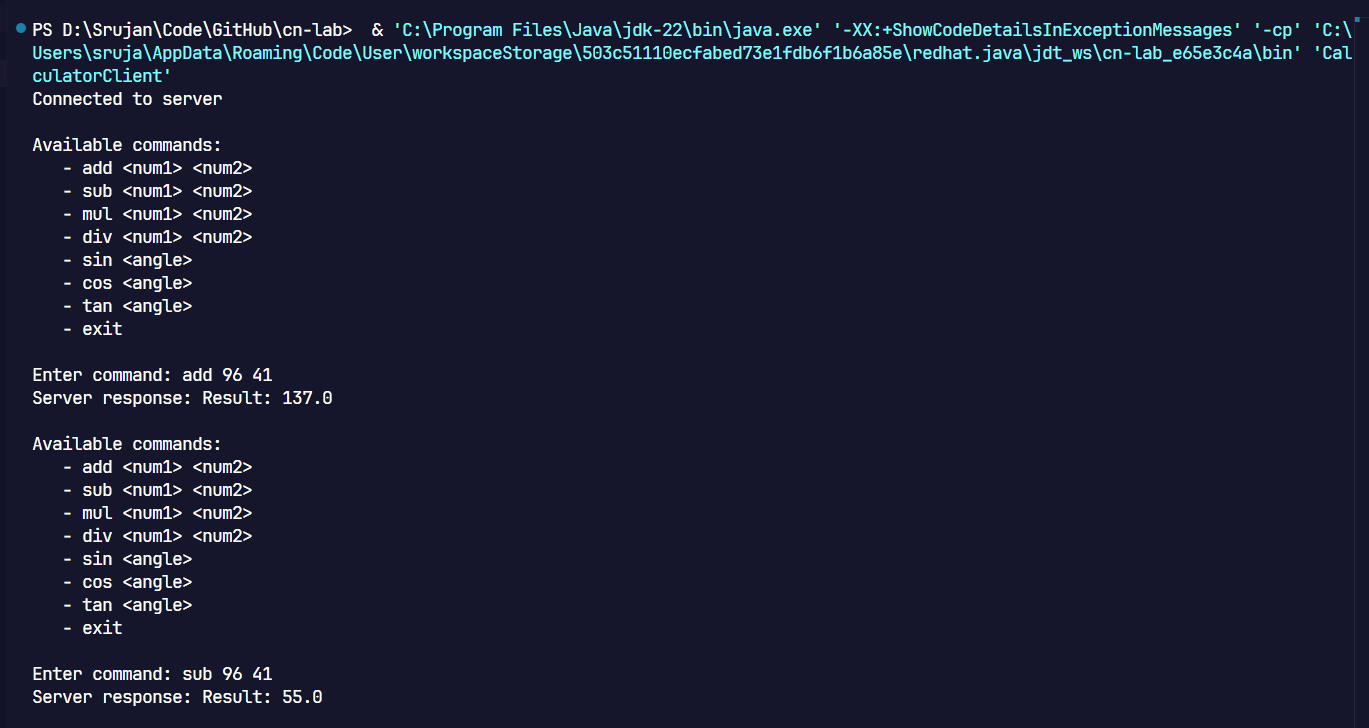
        System.out.println();

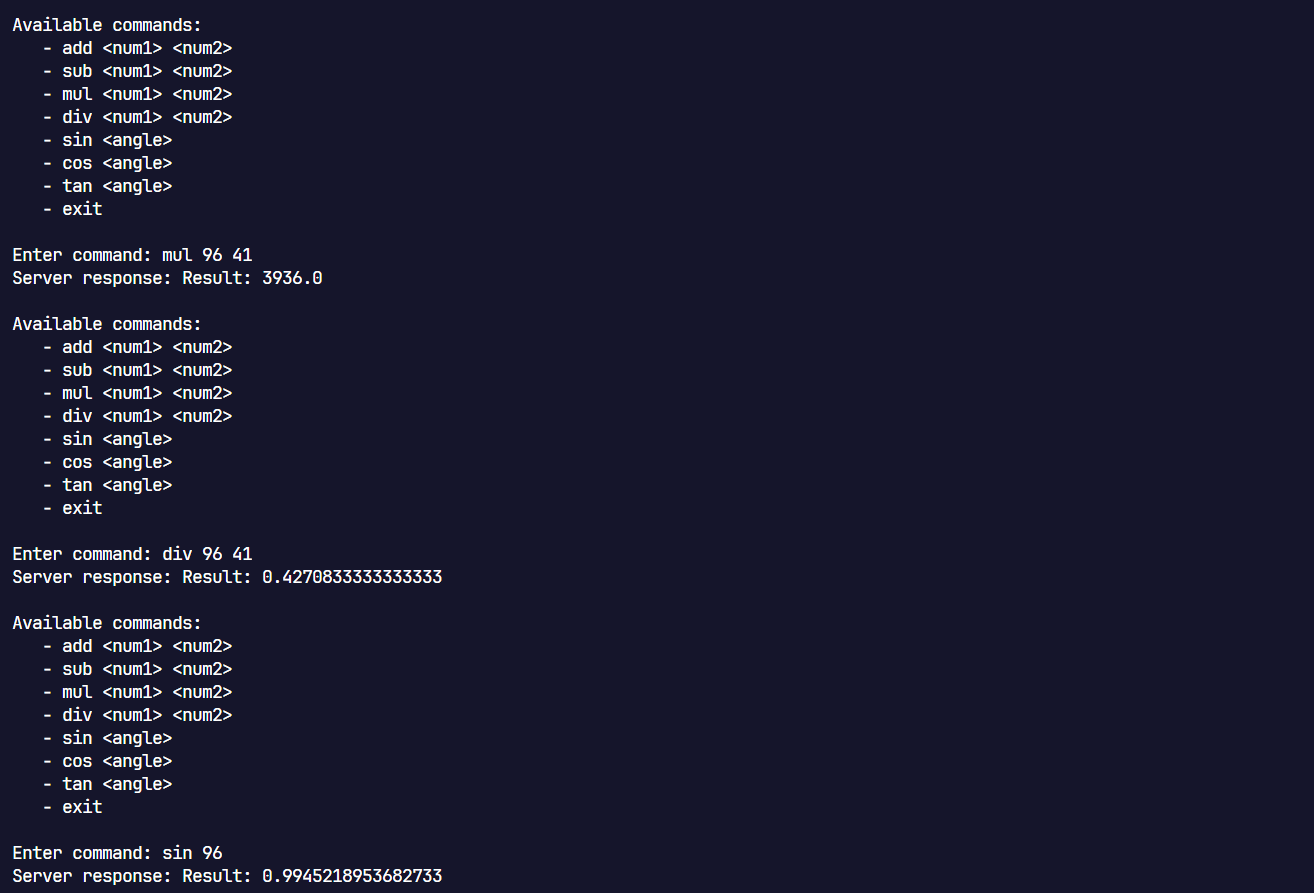
    }

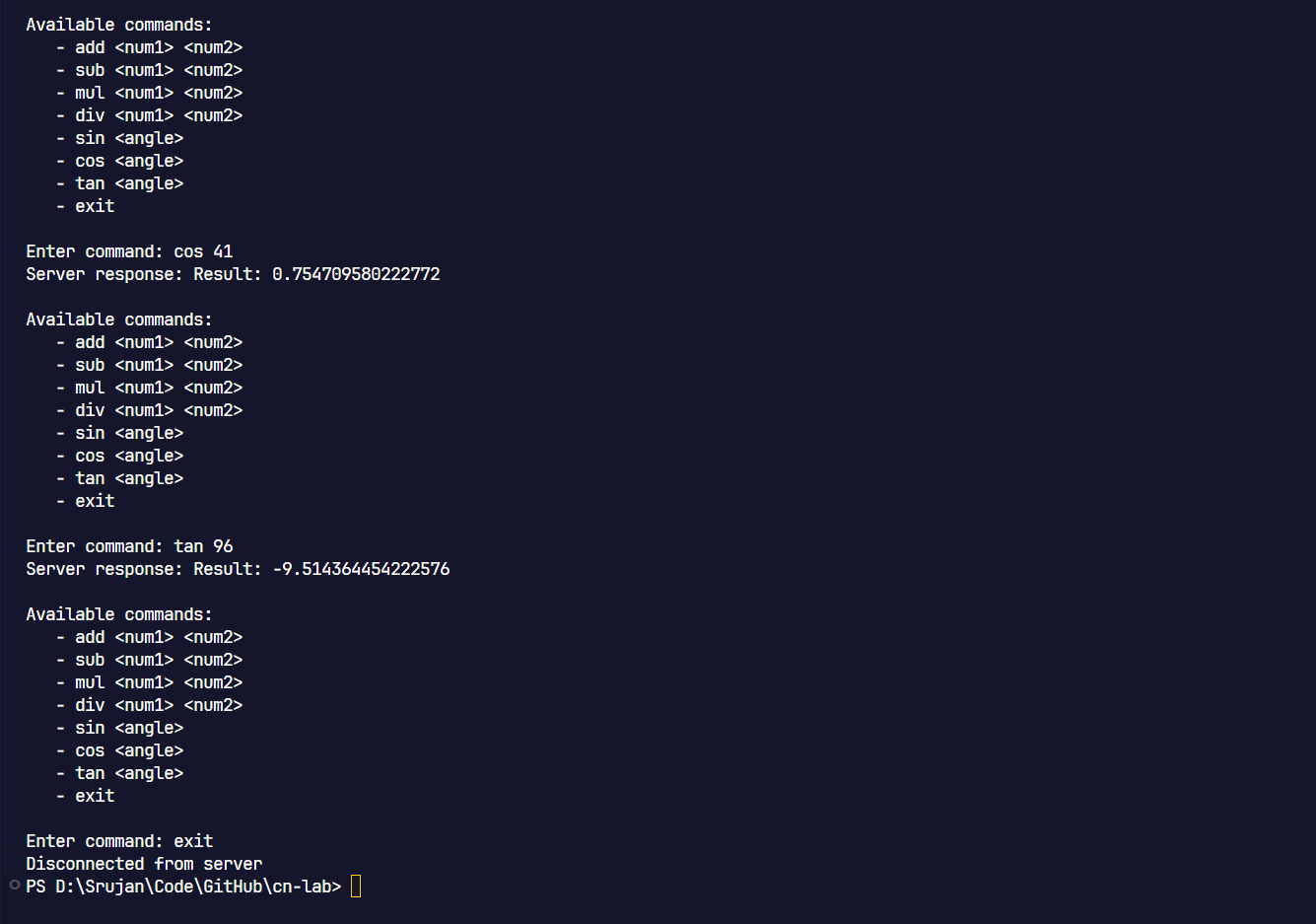
}

### OUTPUT:

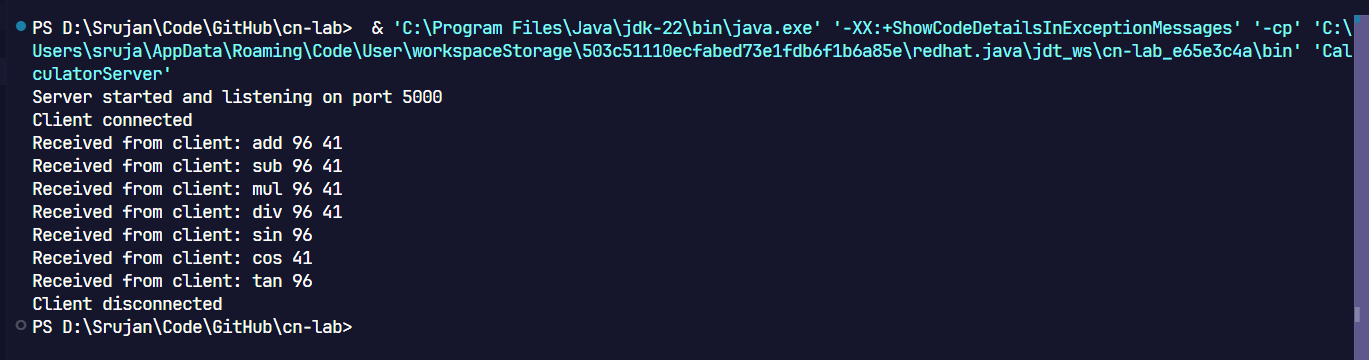
### Client



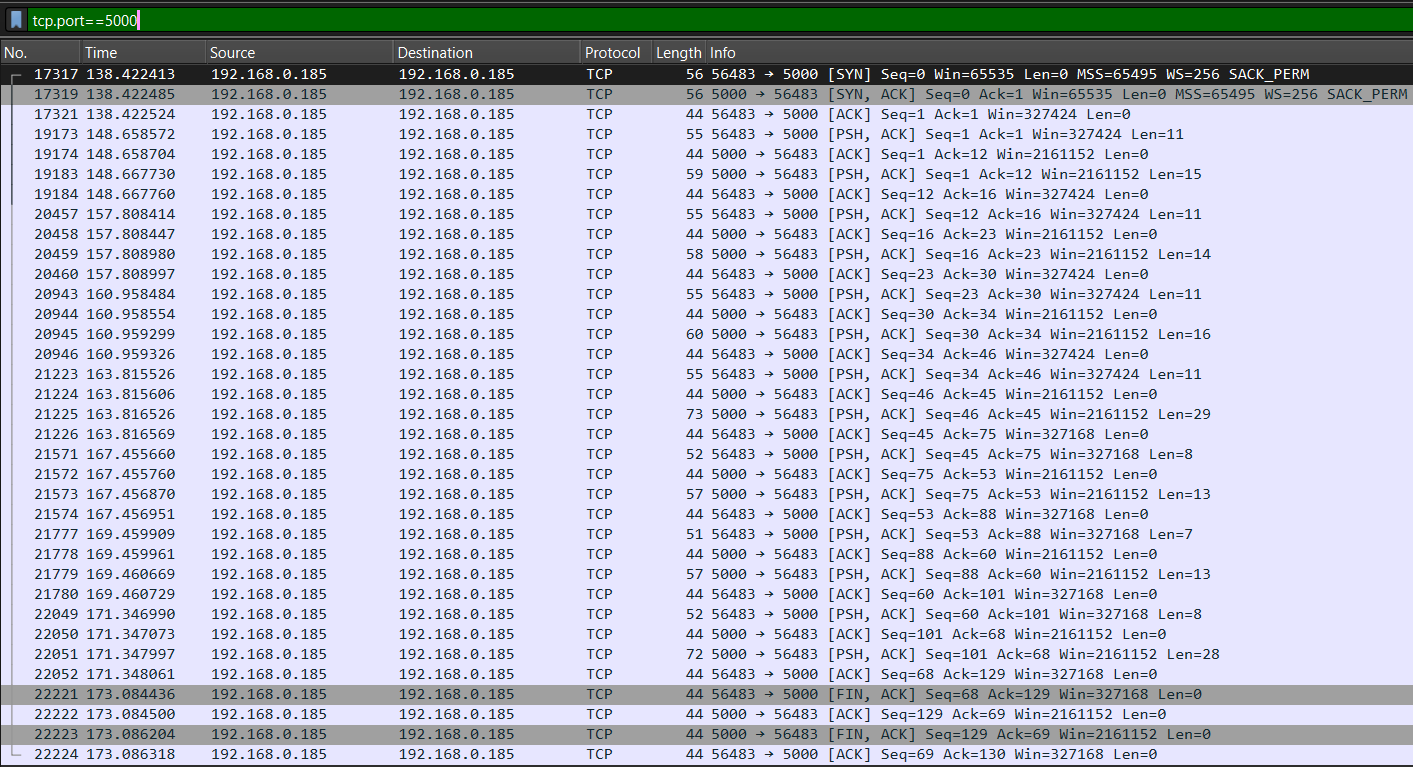
****

****

## Server:



Wireshark:

****

## File Sharing and Saying Hello:

## CODE:

## TCPServer.java:

import *java.io.\**;

import *java.net.\**;

*public* *class* TCPServer {

*public* *static* void main(String[] args) {

        try (ServerSocket serverSocket = new ServerSocket(12345)) {

            System.out.println("Server is listening on port 12345");

            while (true) {

                Socket socket = serverSocket.accept();

                System.out.println("Client connected");

                new ClientHandler(socket).start();

            }

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

}

*class* ClientHandler *extends* Thread {

*private* Socket socket;

*public* ClientHandler(Socket socket) {

        this.socket = socket;

    }

*public* void run() {

        try (BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

             PrintWriter out = new PrintWriter(socket.getOutputStream(), true)) {

            String clientMessage;

            while ((clientMessage = in.readLine()) != null) {

                System.out.println("Client: " + clientMessage);

                if (clientMessage.equalsIgnoreCase("hello")) {

                    out.println("Hello from server!");

                } else if (clientMessage.equalsIgnoreCase("file")) {

                    receiveFile(socket);

                } else {

                    out.println("Unknown command");

                }

            }

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

*private* void receiveFile(Socket socket) *throws* IOException {

        try (InputStream inputStream = socket.getInputStream();

             DataInputStream dataInputStream = new DataInputStream(inputStream)) {

            String fileName = dataInputStream.readUTF(); *// Read file name*

            File file = new File(fileName);

            try (FileOutputStream fos = new FileOutputStream(file);

                 BufferedOutputStream bos = new BufferedOutputStream(fos)) {

                byte[] buffer = new byte[4096];

                int bytesRead;

                while ((bytesRead = dataInputStream.read(buffer)) != -1) {

                    bos.write(buffer, 0, bytesRead);

                }

                System.out.println("File " + fileName + " received.");

            }

        }

    }

}

TCPClient.java:

import java.io.\*;

import java.net.\*;

import java.util.Scanner;

public class TCPClient {

    public static void main(String[] args) {

        try (Socket socket = new Socket("localhost", 12345);

             PrintWriter out = new PrintWriter(socket.getOutputStream(), true);

             BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

             Scanner scanner = new Scanner(System.in)) {

            String userInput;

            while (true) {

                System.out.println("Enter command (hello, file, exit): ");

                userInput = scanner.nextLine();

                if (userInput.equalsIgnoreCase("exit")) {

                    break;

                }

                out.println(userInput);

                if (userInput.equalsIgnoreCase("file")) {

                    sendFile(socket, scanner);

                } else {

                    String response = in.readLine();

                    System.out.println("Server: " + response);

                }

            }

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

    private static void sendFile(Socket socket, Scanner scanner) throws IOException {

        System.out.print("Enter file name to send: ");

        String fileName = scanner.nextLine();

        // Send the file name first

        try (DataOutputStream dos = new DataOutputStream(socket.getOutputStream())) {

            dos.writeUTF(fileName); // Send file name to server

            File file = new File(fileName);

            try (FileInputStream fis = new FileInputStream(file);

                 BufferedInputStream bis = new BufferedInputStream(fis)) {

                byte[] buffer = new byte[4096];

                int bytesRead;

                while ((bytesRead = bis.read(buffer)) != -1) {

                    dos.write(buffer, 0, bytesRead);

                }

                System.out.println("File sent.");

            }

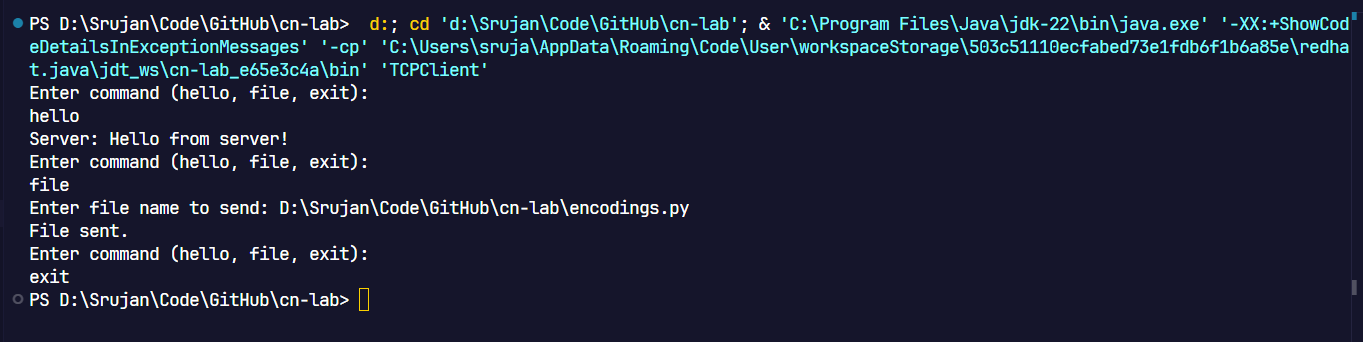
        }

    }

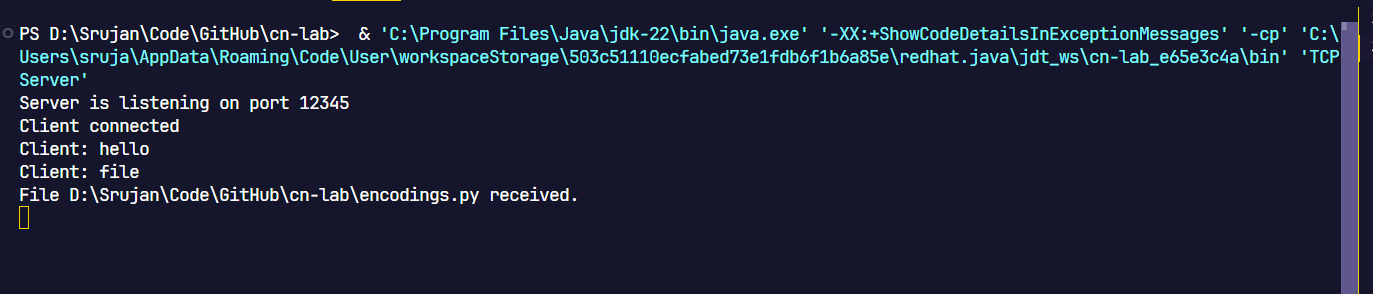
}

## OUTPUT:

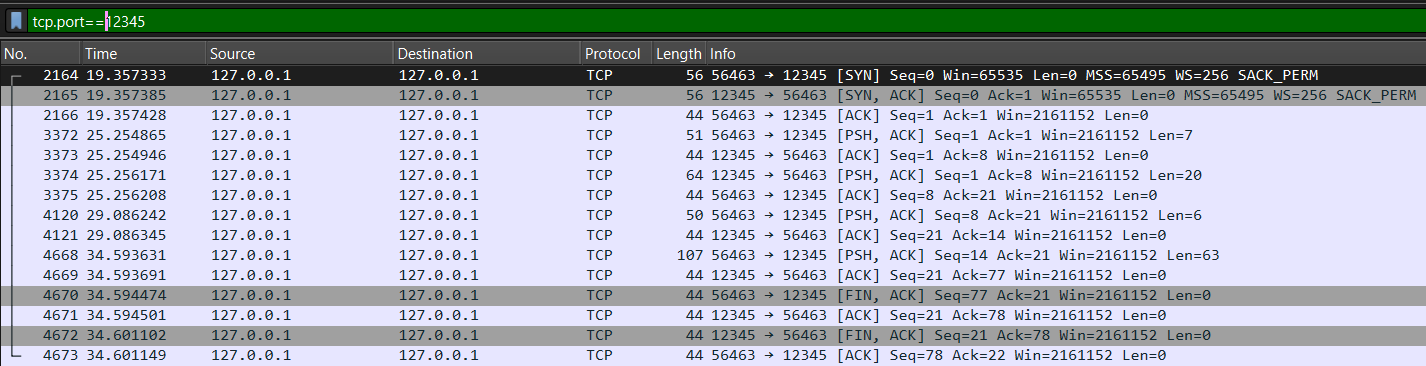
## Client:

****

## Server:

****

## Wireshark:

****