Computer Networks

Assignment-9

Name: S. Vishwajith

# Register Number: 23BCE1145

## Question-1:

### Client:

import java.net.*\**;

import java.io.*\**;

@SuppressWarnings("unused")

public class Client\_23BCE1145{

    private Socket socket = null;

    private DataInputStream in = null;

    private DataInputStream din = null;

    private DataOutputStream dout = null;

    public Client\_23BCE1145(String *address*, *int* *port*) {

        try{

            socket = **new** Socket(*address*, *port*);

            System.out.println("Connected to Server.\n");

            in = **new** DataInputStream(System.in);

            dout = **new** DataOutputStream(socket.getOutputStream());

            din = **new** DataInputStream(**new** BufferedInputStream(socket.getInputStream()));

        }

        catch(IOException *e*){

            System.out.println(e);

        }

    }

    public *void* Work(){

        try{

            String ip = "14.24.74.0";

*int* cidr = 24;

*int* num = 3;

*int*[] arrIps = {10,60,120};

            StringBuilder request = **new** StringBuilder();

            request.append(ip+" "+cidr+" "+num);

            for (*int* i : arrIps) {

                request.append(" ").append(i);

            }

            dout.writeUTF(request.toString());

            String response = din.readUTF();

            System.out.println("Subnet details: \n" + response);

        }

        catch(IOException *e*){

            System.out.println(e);

        }

        finally{

            System.out.println("\nClosing connection.");

            try{

                socket.close();

                din.close();

                dout.close();

            }

            catch(IOException *t*){

                System.out.println(t);

            }

        }

    }

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

        Client\_23BCE1145 client = **new** Client\_23BCE1145("127.0.0.1", 5000);

        client.Work();

    }

}

### Server:

import java.util.*\**;

import java.net.*\**;

import java.io.*\**;

public class Server\_23BCE1145{

    private Socket socket = null;

    private ServerSocket server = null;

    private DataInputStream din = null;

    private DataOutputStream dout = null;

    public Server\_23BCE1145(*int* *port*){

        try{

            server = **new** ServerSocket(*port*);

            System.out.println("Server started\nWaiting for a client...");

            socket = server.accept();

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

            din = **new** DataInputStream(**new** BufferedInputStream(socket.getInputStream()));

            dout = **new** DataOutputStream(socket.getOutputStream());

        }

        catch(IOException *e*){

            System.out.println(e);

        }

    }

    public *void* Work(){

        try{

            String line = din.readUTF();

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

            String[] arr = line.split(" ");

            String ip = arr[0];

*int* cidr = Integer.parseInt(arr[1]);

*int* num = Integer.parseInt(arr[2]);

*int*[] arrIps = **new** *int*[num];

            for (*int* i=0; i<num; i++) {

                arrIps[i] = Integer.parseInt(arr[3 + i]);

            }

            String line2 = calcSubnets(ip, cidr, arrIps);

            dout.writeUTF(line2);

        }

        catch(IOException *i*){

            System.out.println(i);

        }

        finally{

            System.out.println("\nClosing Connection with the client.");

            try{

                socket.close();

                 din.close();

                dout.close();

            }

            catch(IOException *t*){

                System.out.println(t);

            }

        }

    }

    public static String calcSubnets(String *ip*, *int* *cidr*, *int*[] *arrIps*){

*List*<String> subnetInfo = **new** ArrayList<>();

*int* cIP = ipToInt(*ip*);

*int* n = *arrIps*.length;

        for (*int* i = 0; i < n - 1; i++){

            for (*int* j = 0; j < n - i - 1; j++){

                if (*arrIps*[j] < *arrIps*[j + 1]){

*int* temp = *arrIps*[j];

*arrIps*[j] = *arrIps*[j + 1];

*arrIps*[j + 1] = temp;

                }

            }

        }

        for (*int* i : *arrIps*){

*int* p = (*int*)Math.pow(2,Math.ceil(Math.log(i)/Math.log(2)));

*int* newcidr = 32 - (*int*) (Math.log(p)/Math.log(2));

            String mask = SubnetMask(newcidr);

            String faddr = intToIp(cIP);

            String laddr = intToIp(cIP+p-1);

            subnetInfo.add(

                "\nSubnet: " + faddr + "/" + newcidr +

                "\nMask: " + mask +

                "\nRange: (" + faddr + ", " + laddr + ")" +

                "\nFirst address: " + faddr +

                "\nLast address: " + laddr

                );

            cIP += p;

        }

        return String.join("\n", subnetInfo);

    }

    public static *int* ipToInt(String *ip*){

        String[] arr = *ip*.split("\\.");

*int* res = 0;

        for (*int* i=0; i<4; i++) {

            res |= Integer.parseInt(arr[i]) << ((3-i)\*8);

        }

        return res;

    }

    public static String intToIp(*int* *ip*){

        StringBuilder mask = **new** StringBuilder();

        for (*int* i=3; i>=0; i--) {

*int* s = (*ip* >> (i\*8))&255;

            mask.append(s);

            if (i > 0) {

                mask.append(".");

            }

        }

        return mask.toString();

    }

    public static String SubnetMask(*int* *cidr*){

*int* mask = 0xFFFFFFFF << (32 - *cidr*);

         StringBuilder  smask = **new** StringBuilder();

        for (*int* i=3; i>=0; i--) {

*int* s = (mask >> (i\*8))&255;

            smask.append(s);

            if (i > 0) {

                smask.append(".");

            }

        }

    return smask.toString();

    }

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

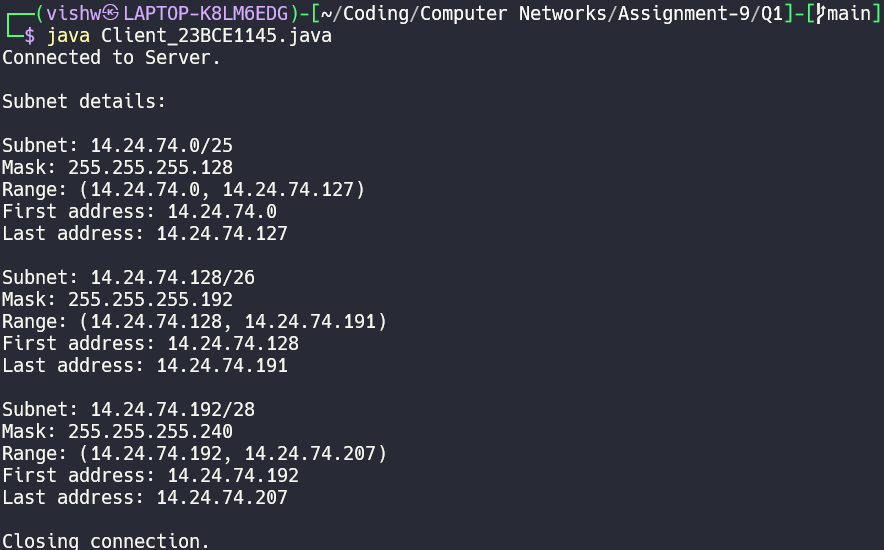
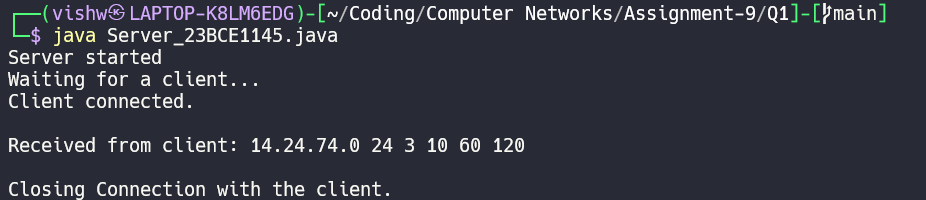
        Server\_23BCE1145 server = **new** Server\_23BCE1145(5000);

        server.Work();

    }

}

### Output:



## Question-2:

### Client:

import java.net.*\**;

import java.util.*\**;

import java.io.*\**;

@SuppressWarnings("deprecation")

public class Client\_23BCE1145{

    private Socket socket = null;

    private DataInputStream din = null;

    private DataInputStream in = null;

    private DataOutputStream dout = null;

    private *int* m = 5;

    private *int* ws = (*int*) Math.pow(2, m - 1);

    private HashMap<Integer, String> windows = **new** HashMap<Integer, String>(ws);

    private *Set*<Integer> acks = Collections.synchronizedSet(**new** HashSet<Integer>());

    public Client\_23BCE1145(String *address*, *int* *port*) {

        try {

            socket = **new** Socket(*address*, *port*);

            System.out.println("Connected");

            din = **new** DataInputStream(System.in);

            dout = **new** DataOutputStream(socket.getOutputStream());

            in = **new** DataInputStream(**new** BufferedInputStream(socket.getInputStream()));

        }

        catch (Exception *u*) {

            System.out.println(u);

            return;

        }

    }

    public *void* Work() {

        String line = "";

*int* fc = 0;

        try {

            while (true) {

                System.out.println("\nEnter the message to send: ");

                line = din.readLine();

                System.out.println("\nClient is sending: " + line);

                if (line.equalsIgnoreCase("end")) {

                    System.out.println("Closing Connection");

                    dout.writeUTF("End");

                    dout.flush();

                    Thread.sleep(1000);

                    closeConnections();

                    return;

                }

                String[] arr = line.split(" ");

*int* totalFrames = arr.length;

                for (*int* i = 0; i < totalFrames; i++) {

                    windows.put(i, arr[i]);

                    acks.add(i);

                }

**new** Thread(() *->* receiveACKs()).start();

                while (!acks.isEmpty()) {

                    for (*int* i = 0; i < ws && fc < totalFrames; i++) {

                        if (acks.contains(fc)) {

                            //if (fc!=3)

                            send(fc);

                        }

                        fc++;

                    }

                    Thread.sleep(5000);

                    for (*int* index : **new** ArrayList<>(acks)) {

                        System.out.println("Timeout for Frame " + index + ", resending...");

                        send(index);

                    }

                }

            }

        }

        catch (Exception *t*) {

            System.out.println(t);

        }

        finally {

            closeConnections();

        }

    }

    public *void* send(*int* *i*) {

        System.out.println("Sending Frame " + (*i*+1) + " with Seq No. " + (*i* % (ws \* 2)));

        try {

            dout.writeUTF(windows.get(*i*) + " " + (*i* % (ws \* 2)));

        }

        catch (Exception *e*) {

            System.out.println(e);

        }

    }

    public *void* receiveACKs() {

        try {

            while (!acks.isEmpty()) {

                String ack = in.readUTF();

*int* ackNum = Integer.parseInt(ack);

                if (acks.contains(ackNum)) {

                    System.out.println("Ack " + ackNum + " received");

                    acks.remove(ackNum);

                }

            }

        }

        catch (IOException *e*) {

            System.out.println(e);

        }

    }

    public *void* closeConnections() {

        try {

            socket.close();

            din.close();

            dout.close();

        }

        catch (IOException *t*) {

            System.out.println(t);

        }

    }

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

        Client\_23BCE1145 client = **new** Client\_23BCE1145("127.0.0.1", 5000);

        client.Work();

    }

}

### Server:

import java.net.*\**;

import java.io.*\**;

import java.util.*\**;

@SuppressWarnings("unused")

public class Server\_23BCE1145{

    private Socket socket = null;

    private ServerSocket server = null;

    private DataInputStream din = null;

    private DataOutputStream dout = null;

    private *int* m = 5;

    private *int* ws = (*int*) Math.pow(2, m - 1);

    private HashMap<Integer, String> receivedFrames = **new** HashMap<Integer, String>();

    public Server\_23BCE1145(*int* *port*) {

        try {

            server = **new** ServerSocket(*port*);

            System.out.println("Server started\nWaiting for a client...");

            socket = server.accept();

            System.out.println("Client accepted.\n");

            din = **new** DataInputStream(**new** BufferedInputStream(socket.getInputStream()));

            dout = **new** DataOutputStream(socket.getOutputStream());

            receiveFrames();

            socket.close();

            din.close();

            dout.close();

        }

        catch (IOException *u*) {

            System.out.println(u);

        }

    }

    public *void* receiveFrames() {

        String line = "";

*int* expectedSeq = 0;

        while (true) {

            try {

                line = din.readUTF();

                if (line.equalsIgnoreCase("end")){

                    System.out.println("\nClosing the server.");

                    return;

                }

                String[] parts = line.split(" ");

                String f = parts[0];

*int* seq = Integer.parseInt(parts[1]);

                if (!receivedFrames.containsKey(seq)) {

                    receivedFrames.put(seq, f);

                    System.out.println("Received Frame with Seq No. " + seq + " and message: \"" + f + "\"");

                    if (seq==6){

**new** Thread(()*->*{

                            try{

                                Thread.sleep(6000);

                                dout.writeUTF(seq + "");

                            }

                            catch(Exception *e*){

                                System.out.println(e);

                            }

                        }).start();

                    }

                    else{

                        dout.writeUTF(seq + "");

                    }

                    expectedSeq = (expectedSeq + 1) % (ws \* 2);

                }

                else {

                    System.out.println("Duplicate Frame " + seq + " discarded.");

                }

            }

            catch (EOFException *e*) {

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

                break;

            }

            catch (IOException *e*) {

                System.out.println(e);

                break;

            }

        }

    }

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

        Server\_23BCE1145 server = **new** Server\_23BCE1145(5000);

    }

}

### Output:

A screenshot of a computer program

AI-generated content may be incorrect.A screenshot of a computer program

AI-generated content may be incorrect.

## Question-3:

### Client:

import java.io.*\**;

import java.net.*\**;

@SuppressWarnings("unused")

class Client {

    private Socket socket = null;

    private DataInputStream in = null;

    private DataOutputStream out = null;

    private BufferedReader input = null;

    private String address = "";

    private *int* port = -1;

    public Client(*int* *port*, String *address*) {

*this*.port = *port*;

*this*.address = *address*;

    }

    public *void* Work() {

        try {

            socket = **new** Socket(address, port);

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

            in = **new** DataInputStream(socket.getInputStream());

            out = **new** DataOutputStream(socket.getOutputStream());

            input = **new** BufferedReader(**new** InputStreamReader(System.in));

            System.out.print("Enter the Message: ");

            String Input\_Msg = input.readLine();

            String[] messages = Input\_Msg.split(" ");

*int* Ssize = 4;

            String line = "";

*int* Sf = 0, Sn = 0;

*boolean* flag = false;

*long* currTime = System.currentTimeMillis();

            while(System.currentTimeMillis() - currTime < 2000) {

                while (Sn - Sf + 1 <= Ssize && *Sn* < messages.length) {

                    if(Sn==1 && !flag){

                        System.out.println("Sending packet (Sn) " + Sn + ": " + messages[Sn]+" -- (lost)");

                        Sn++;

                        flag = true;

                        break;

                    }

                    System.out.println("Sending packet (Sn) " + Sn + ": " + messages[Sn]);

                    out.writeUTF(messages[Sn]+": "+Sn);

                    out.flush();

                    Sn++;

                }

                try {

                    socket.setSoTimeout(1000);

                    String ack = in.readUTF();

                    if (ack.equals("ACK: " + (Sf+1))) {

                        System.out.println("Received ACK for packet (Sf)" + Sf);

                        Sf++;

                    }

                } catch (SocketTimeoutException *e*) {

                    System.out.println("Timeout! Resending unacknowledged packets...");

                    for (*int* i = Sf; i < Sn; i++) {

                        System.out.println("Resending packet " + i + ": " + messages[i]);

                        out.writeUTF(messages[i]+": "+i);

                        out.flush();

                    }

                }

            }

            System.out.println("\nAll packets acknowledged. Closing connection...");

            out.writeUTF("end");

            out.flush();

            socket.close();

            in.close();

            out.close();

        } catch (IOException *e*) {

            System.out.println("Client error: " + e);

        }

    }

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

        Client client = **new** Client(5000, "127.0.0.1");

        client.Work();

    }

}

### Server:

import java.io.*\**;

import java.net.*\**;

class Server {

    private Socket socket = null;

    private ServerSocket server = null;

    private DataInputStream in = null;

    private DataOutputStream out = null;

    private BufferedReader input = null;

    private *int* portNum = -1;

    private *int* Rn = 0;

    public Server(*int* *port*) {

*this*.portNum = *port*;

    }

*void* Work() {

        try {

            server = **new** ServerSocket(portNum);

            System.out.println("Server started.\nWaiting for a client...");

            socket = server.accept();

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

            in = **new** DataInputStream(socket.getInputStream());

            out = **new** DataOutputStream(socket.getOutputStream());

            input = **new** BufferedReader(**new** InputStreamReader(System.in));

            String clientMessage;

            while (true) {

                try {

                    clientMessage = in.readUTF();

                    System.out.println("Received: Message = \"" + clientMessage + "\"");

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

                        System.out.println("Client ended the session.");

                        break;

                    }

*int* receivedSeqNum = Integer.parseInt(clientMessage.split(": ")[1]);

                    if (receivedSeqNum == Rn) {

                        System.out.println("Correct packet received. Delivering data...");

                        Rn++;

                        out.writeUTF("ACK: " + Rn);

                        System.out.println("Rn = "+Rn);

                    } else {

                        System.out.println("Out-of-order packet received. Discarding...");

                    }

                } catch (IOException *e*) {

                    System.out.println("Error: " + e);

                    break;

                }

            }

            System.out.println("\nClosing connection...");

            socket.close();

            in.close();

            out.close();

            server.close();

            input.close();

        } catch (IOException *e*) {

            System.out.println("Server error: " + e);

        }

    }

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

        Server server = **new** Server(5000);

        server.Work();

    }

}

### Output: A screenshot of a computer program AI-generated content may be incorrect.A screenshot of a computer program AI-generated content may be incorrect.

## Question-4:

### Client:

import java.io.*\**;

import java.net.*\**;

public class Client\_23BCE1145{

    private static Socket socket;

    private static DataInputStream dataIn;

    private static DataOutputStream dataOut;

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

        String ip = *args*[0];

        String cidr = *args*[1];

        System.out.println("Starting the client...");

        socket = **new** Socket();

        socket.connect(**new** InetSocketAddress("127.0.0.1", 5001), 1000);

        System.out.println("Connected to the server.\n");

        dataIn = **new** DataInputStream(socket.getInputStream());

        dataOut = **new** DataOutputStream(socket.getOutputStream());

        dataOut.writeUTF(ip);

        dataOut.writeUTF(cidr);

        String serverMessage = dataIn.readUTF();

        System.out.println("Subnet information: " + serverMessage);

        dataIn.close();

        dataOut.close();

        socket.close();

        System.out.println("\nConnection closed.");

    }

}

### Server:

import java.io.*\**;

import java.net.*\**;

import java.util.*\**;

@SuppressWarnings("resource")

public class Server\_23BCE1145{

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

        ServerSocket serverSocket = **new** ServerSocket(5001);

        System.out.println("Server started on port 5001...");

*long* startTime = System.currentTimeMillis();

        while(true){

            if(System.currentTimeMillis() - startTime > 30000){

                System.out.println("No longer accepting clients!");

                break;

            }

            Socket connectionSocket = serverSocket.accept();

**new** Thread(() *->* {

                try{

                    DataInputStream dataIn = **new** DataInputStream(connectionSocket.getInputStream());

                    DataOutputStream dataOut = **new** DataOutputStream(connectionSocket.getOutputStream());

                    String ip = dataIn.readUTF();

*int* cidr = Integer.parseInt(dataIn.readUTF());

                    String line2 = calcSubnets(ip, cidr);

                    dataOut.writeUTF(line2);

                    connectionSocket.close();

                }

                catch(IOException *e*){

                e.printStackTrace();

                }

            }).start();

        }

    }

    public static String calcSubnets(String *ip*, *int* *cidr*){

*List*<String> subnetInfo = **new** ArrayList<>();

        String faddr = *ip*;

        String laddr = lastAddr(*ip*,*cidr*);

*int* numHosts = (*int*) Math.pow(2,(32-*cidr*));

        String mask = SubnetMask(*cidr*);

        subnetInfo.add(

            " First address: " + faddr +

            " Last address: " + laddr +

            " Number of Hosts: " + numHosts +

            " Subnet Mask: " + mask

            );

        return String.join("\n", subnetInfo);

    }

    public static *int* ipToInt(String *ip*){

        String[] arr = *ip*.split("\\.");

*int* res = 0;

        for(*int* i=0; i<4; i++){

            res |= Integer.parseInt(arr[i]) << ((3-i)\*8);

        }

        return res;

    }

    public static String intToIp(*int* *ip*){

        StringBuilder mask = **new** StringBuilder();

        for(*int* i=3; i>=0; i--){

*int* s = (*ip* >> (i\*8))&255;

            mask.append(s);

            if (i > 0) {

                mask.append(".");

            }

        }

        return mask.toString();

    }

    public static String lastAddr(String *ip*,*int* *cidr*){

*int* n = ipToInt(*ip*);

*int* mask = 0xFFFFFFFF << (32-*cidr*);

*int* res = n | ~mask;

        StringBuilder  smask = **new** StringBuilder();

        for (*int* i=3; i>=0; i--) {

*int* s = (res >> (i\*8))&255;

            smask.append(s);

            if (i > 0) {

                smask.append(".");

            }

        }

    return smask.toString();

    }

    public static String SubnetMask(*int* *cidr*){

*int* mask = 0xFFFFFFFF << (32 - *cidr*);

        StringBuilder  smask = **new** StringBuilder();

        for (*int* i=3; i>=0; i--) {

*int* s = (mask >> (i\*8))&255;

            smask.append(s);

            if (i > 0) {

                smask.append(".");

            }

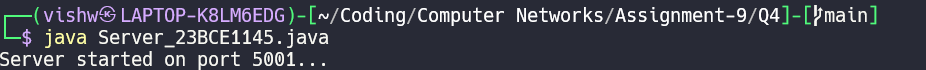
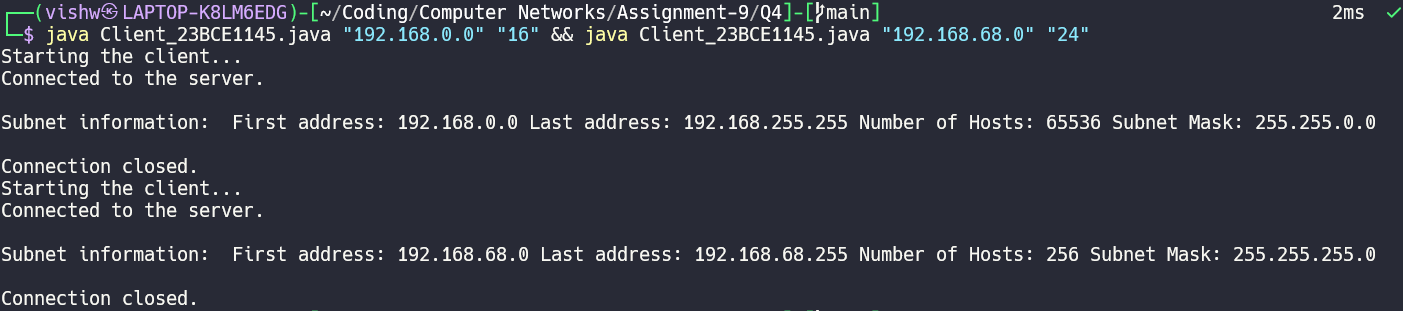
        }

    return smask.toString();

    }

}

### Output:



## Question-5:

### Client:

import java.io.*\**;

import java.net.*\**;

public class Client\_23BCE1145{

    private static Socket socket;

    private static DataInputStream dataIn;

    private static DataOutputStream dataOut;

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

        String ip = "192.17.21.26";

        String packet = *args*[0];

        socket = **new** Socket();

        socket.connect(**new** InetSocketAddress("127.0.0.1", 5001), 1000);

        System.out.println("Connection Successful!\n");

        dataIn = **new** DataInputStream(socket.getInputStream());

        dataOut = **new** DataOutputStream(socket.getOutputStream());

        dataOut.writeUTF(ip);

        dataOut.writeUTF(packet);

        String serverMessage = dataIn.readUTF();

        System.out.println("Subnet information: " + serverMessage);

        dataIn.close();

        dataOut.close();

        socket.close();

        System.out.println("\nConnection Closed!");

    }

}

### Server:

import java.io.*\**;

import java.net.*\**;

@SuppressWarnings({"resource", "unused"})

public class Server\_23BCE1145{

    static final String[][] routing\_table = {

        {"192.17.0.0", "18", "a"},

        {"192.17.20.0", "22", "b"},

        {"0.0.0.0", "0", "Default"}

    };

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

        ServerSocket serverSocket = **new** ServerSocket(5001);

        System.out.println("Server started on port 5001...");

        while (true){

            Socket connectionSocket = serverSocket.accept();

**new** Thread(() *->* {

                try{

                    DataInputStream dataIn = **new** DataInputStream(connectionSocket.getInputStream());

                    DataOutputStream dataOut = **new** DataOutputStream(connectionSocket.getOutputStream());

                    for(*int* i=0;i<3;i++){

                        for(*int* j=i+1;j<3;j++){

                            if(Integer.parseInt(routing\_table[i][1]) < Integer.parseInt(routing\_table[j][1])){

                                String arr[] = routing\_table[i];

                                routing\_table[i] = routing\_table[j];

                                routing\_table[j] = arr;

                            }

                        }

                    }

                    String ip = dataIn.readUTF();

                    String packet = dataIn.readUTF();

                    String line2 = "";

                    for(*int* i=0;i<3;i++){

*int* n = ipToInt(ip);

*int* mask = 0xFFFFFFFF << (32-Integer.parseInt(routing\_table[i][1]));

*int* k = ipToInt(routing\_table[i][0]);

                        if((n & mask)==k){

                            line2 += "The longest prefix match is " + routing\_table[i][0] + "/" + routing\_table[i][1] + " and packet will be sent to interface "+routing\_table[i][2];

                            break;

                        }

                    }

                    dataOut.writeUTF(line2);

                    connectionSocket.close();

                }

                catch(IOException *e*){

                    e.printStackTrace();

                }

            }).start();

        }

    }

    public static *int* ipToInt(String *ip*) {

        String[] arr = *ip*.split("\\.");

*int* res = 0;

        for (*int* i=0; i<4; i++) {

            res |= Integer.parseInt(arr[i]) << ((3-i)\*8);

        }

        return res;

    }

}

### Output:

A screen shot of a computer

AI-generated content may be incorrect.

