Practical - I

```
/* ============================================================
 *
 * Roll No: 30
 *
 * File:      Client10.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

import java.io.*;
import java.net.*;
import java.nio.ByteBuffer;
import java.util.Arrays;

public class Client10 {
  private static String source = "1";
  private static String destination = "2";

  public static void main(String args[]) throws Exception {
    String sentence;
    BufferedReader inFromUser = new BufferedReader(new
InputStreamReader(System.in));
    Socket clientSocket = new Socket(InetAddress.getLocalHost(), 6789);
    DataOutputStream outToServer = new
DataOutputStream(clientSocket.getOutputStream());
    DataInputStream inFromServer = new
DataInputStream(clientSocket.getInputStream());

    sentence = inFromUser.readLine();

    byte[] sourceAddress = Arrays.copyOf(source.getBytes(), 10);
    byte[] destinationAddress = Arrays.copyOf(destination.getBytes(), 10);
    byte[] data = Arrays.copyOf(sentence.getBytes(), 50);
    byte[] packet =
ByteBuffer.allocate(70).put(sourceAddress).put(destinationAddress).put(data).arr
ay();

    outToServer.write(packet);

    clientSocket.close();
  }
}
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      ClientHandler.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
```

```
 * ========================================================== */

 package TCP;

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

public class ClientHandler extends Thread {
    private final Socket socket;
    private final TCP7Server server;
    private PrintWriter writer;
    public String username;

    public ClientHandler(Socket socket, TCP7Server server) {
        this.socket = socket;
        this.server = server;
    }

    @Override
    public void run() {
        try {
            BufferedReader reader = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
            writer = new PrintWriter(socket.getOutputStream(), true);
            username = reader.readLine();
            server.addUser(username, this);
            server.listUsers();
            String serverMessage = "New User Connected: " + username;
            server.sendToEveryone(serverMessage, this);
            String clientMessage;
            do {
                clientMessage = reader.readLine().trim();
                serverMessage = "From " + username + ": " + clientMessage;
                System.out.println(serverMessage);
                server.sendToEveryone(serverMessage, this);
            } while (!clientMessage.equalsIgnoreCase("bye"));
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        } finally {
            try {
                server.removeUser(username);
                socket.close();
                String serverMessage = username + " disconnected...";
                server.sendToEveryone(serverMessage, this);
            } catch (Exception e) {
                System.out.println("Error: " + e.getMessage());
            }
        }
    }

    void sendMessage(String message) {
```

```
        writer.println(message);
    }
}
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      MainServer10.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

 import java.io.*;
import java.net.*;
import java.nio.ByteBuffer;
import java.util.Arrays;

class ms {
    public static void main(String args[]) throws Exception {
        ServerSocket welcomeSocket = new ServerSocket(9999);

        while (true) {
            Socket connectionSocket = welcomeSocket.accept();
            DataInputStream inFromClient = new
DataInputStream(connectionSocket.getInputStream());
            DataOutputStream outToClient = new
DataOutputStream(connectionSocket.getOutputStream());

            byte[] clientData = new byte[70];
            inFromClient.read(clientData);

            String sSourceAddress = new String(Arrays.copyOfRange(clientData, 0,
10));
            String sDestinationAddress = new
String(Arrays.copyOfRange(clientData, 10, 20));
            String receivedData = new String(Arrays.copyOfRange(clientData, 20,
70));

            System.out.println("-- From Client --");
            System.out.println("Sender Source Address: " + sSourceAddress);
            System.out.println("Sender Destination Address: " +
sDestinationAddress);
            System.out.println("Data: " + receivedData);
            break;
        }
    }
}

 /* ============================================================
 *
 * Roll No: 30
```

```
 *
 * File:      Server10.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ========================================================== */

 import java.io.*;
import java.net.*;
import java.nio.ByteBuffer;
import java.util.Arrays;


public class Server10 {
    private static String source = "2";
    private static String destination = "3";

    public static void main(String args[]) throws Exception {
        ServerSocket welcomeSocket = new ServerSocket(6789);

        String sSourceAddress;
        String sDestinationAddress;
        String receivedData;
        while (true) {
            Socket connectionSocket = welcomeSocket.accept();
            DataInputStream inFromClient = new
DataInputStream(connectionSocket.getInputStream());
            DataOutputStream outToClient = new
DataOutputStream(connectionSocket.getOutputStream());

            byte[] clientData = new byte[70];
            inFromClient.read(clientData);

            sSourceAddress = new String(Arrays.copyOfRange(clientData, 0, 10));
            sDestinationAddress = new String(Arrays.copyOfRange(clientData, 10,
20));
            receivedData = new String(Arrays.copyOfRange(clientData, 20, 70));

            System.out.println("-- From Client --");
            System.out.println("Sender Source Address: " + sSourceAddress);
            System.out.println("Sender Destination Address: " +
sDestinationAddress);
            System.out.println("Data: " + receivedData);
            break;
        }
        // Server sends to server
        BufferedReader inFromUser = new BufferedReader(new
InputStreamReader(System.in));
        Socket clientSocket = new Socket(InetAddress.getLocalHost(), 9999);

        DataOutputStream outToServer = new
DataOutputStream(clientSocket.getOutputStream());
```

```
        byte[] sourceAddress = Arrays.copyOf(sSourceAddress.getBytes(), 10);
        byte[] destinationAddress =
Arrays.copyOf(sDestinationAddress.getBytes(), 10);
        byte[] data = Arrays.copyOf(receivedData.getBytes(), 50);
        byte[] packet =
ByteBuffer.allocate(70).put(sourceAddress).put(destinationAddress).put(data).arr
ay();

        outToServer.write(packet);
        clientSocket.close();
    }
}
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      TCP1Client.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class TCP1Client {

    public static void main(String argv[]) throws Exception {
        String sentence;
        while (true) {
            Socket clientSocket = new Socket("localhost", 6789);
            BufferedReader inFromUser = new BufferedReader(new
InputStreamReader(System.in));
            DataOutputStream outToServer = new
DataOutputStream(clientSocket.getOutputStream())
            sentence = inFromUser.readLine();
            outToServer.writeBytes(sentence + '\n');
            if (sentence.equalsIgnoreCase("exit") ||
sentence.equalsIgnoreCase("bye")) {
                clientSocket.close();
                break;
            }
        }
    }
}

 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      TCP1Server.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
```

```
 *
 * ============================================================ */

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

public class TCP1Server {

    public static void main(String argv[]) throws Exception {
        String clientSentence;
        ServerSocket welcomeSocket = new ServerSocket(6789);
        System.out.println("Searching Client.....");
        while (true) {
            Socket connectionSocket = welcomeSocket.accept();
            try (BufferedReader inFromClient = new BufferedReader(
                    new InputStreamReader(connectionSocket.getInputStream()));)
{
                clientSentence = inFromClient.readLine();
                System.out.println("Client String = " + clientSentence);
            }
            if (clientSentence.equalsIgnoreCase("exit") ||
clientSentence.equalsIgnoreCase("bye")) {
                connectionSocket.close();
                break;
            }
        }
    }
}
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      TCP2Client.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class TCP2Client {

    public static void main(String argv[]) throws Exception {
        String sentence;
        String upper;

        try (BufferedReader inFromUser = new BufferedReader(new
InputStreamReader(System.in));
                Socket clientSocket = new Socket(InetAddress.getLocalHost(),
6789);
```

```
                DataOutputStream outToServer = new
DataOutputStream(clientSocket.getOutputStream());
                BufferedReader inFromServer = new BufferedReader(
                        new InputStreamReader(clientSocket.getInputStream()));)
{
            sentence = inFromUser.readLine();

            outToServer.writeBytes(sentence + '\n');

            upper = inFromServer.readLine();

            System.out.println("From Server: " + upper);
        }
    }
}
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      TCP2Server.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class TCP2Server {

    public static void main(String argv[]) throws Exception {
        String clientSentence;

        ServerSocket welcomeSocket = new ServerSocket(6789);

        while (true) {
            Socket connectionSocket = welcomeSocket.accept();
            try (BufferedReader inFromClient = new BufferedReader(
                    new InputStreamReader(connectionSocket.getInputStream()));
                    DataOutputStream outToClient = new
DataOutputStream(connectionSocket.getOutputStream());) {
                clientSentence = inFromClient.readLine();
                String upperCase = clientSentence.toUpperCase();
                System.out.println("Client Request String = " + clientSentence +
" | Response = " + upperCase);
                outToClient.writeBytes(upperCase);
            }
        }
    }
}
 /* ============================================================
 *
```

```
 * Roll No: 30
 *
 * File:      TCP3Client.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class TCP3Client {

    public static void main(String argv[]) throws Exception {
        String sentence;
        int length;

        try (BufferedReader inFromUser = new BufferedReader(new
InputStreamReader(System.in));
                Socket clientSocket = new Socket(InetAddress.getLocalHost(),
6789);
                DataOutputStream outToServer = new
DataOutputStream(clientSocket.getOutputStream());
                BufferedReader inFromServer = new BufferedReader(
                        new InputStreamReader(clientSocket.getInputStream()));)
{
            sentence = inFromUser.readLine();

            outToServer.writeBytes(sentence + '\n');

            length = inFromServer.read();

            System.out.println("From Server: " + length);
        }
    }
}
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      TCP3Server.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class TCP3Server {

    public static void main(String argv[]) throws Exception {
        String clientSentence;
```

```
        int length;

        ServerSocket welcomeSocket = new ServerSocket(6789);

        while (true) {
            Socket connectionSocket = welcomeSocket.accept();
            try (BufferedReader inFromClient = new BufferedReader(
                    new InputStreamReader(connectionSocket.getInputStream()));
                    DataOutputStream outToClient = new
DataOutputStream(connectionSocket.getOutputStream());) {
                clientSentence = inFromClient.readLine();
                length = clientSentence.length();
                System.out.println("Client Request String = " + clientSentence +
" | Response = " + length);
                outToClient.writeByte(length);
            }
        }
    }
}
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      TCP4Client.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class TCP4Client {

    public static void main(String argv[]) throws Exception {
        String sentence;
        String wish;
        System.out.println("Enter Time as per 24 Hours Format (eg 17:50)");
        try (BufferedReader inFromUser = new BufferedReader(new
InputStreamReader(System.in));
                Socket clientSocket = new Socket(InetAddress.getLocalHost(),
6789);
                DataOutputStream outToServer = new
DataOutputStream(clientSocket.getOutputStream());
                BufferedReader inFromServer = new BufferedReader(
                        new InputStreamReader(clientSocket.getInputStream()));)
{
            sentence = inFromUser.readLine();

            outToServer.writeBytes(sentence + '\n');

            wish = inFromServer.readLine();
```

```
            System.out.println("From Server: " + wish);
        }
    }
}
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      TCP4Server.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class TCP4Server {

    public static void main(String argv[]) throws Exception {
        String clientSentence;
        int time = 0;
        int minute = 0;
        ServerSocket welcomeSocket = new ServerSocket(6789);

        while (true) {
            Socket connectionSocket = welcomeSocket.accept();
            try (BufferedReader inFromClient = new BufferedReader(
                    new InputStreamReader(connectionSocket.getInputStream()));
                    DataOutputStream outToClient = new
DataOutputStream(connectionSocket.getOutputStream());) {
                clientSentence = inFromClient.readLine();
                try {
                    time = Integer.parseInt(clientSentence.substring(0,
clientSentence.indexOf(":")));
                    minute = Integer.parseInt(
                            clientSentence.substring(clientSentence.indexOf(":")
+ 1, clientSentence.length()));

                    if ((time < 0 || time > 23) || (minute > 59 || minute < 0))
                        outToClient.writeBytes("Invalid Time");
                    else if (time > 19)
                        outToClient.writeBytes("Good Night");
                    else if (time > 15)
                        outToClient.writeBytes("Good Evening");
                    else if (time > 11)
                        outToClient.writeBytes("Good Afernoon");
                    else if (time > 4)
                        outToClient.writeBytes("Good Morning");
                    else
                        outToClient.writeBytes("Good MidNight");
```

```
                } catch (Exception e) {
                    outToClient.writeBytes("Invalid Time");
                }
                // System.out.println("Client Request String = " +
clientSentence + " | Response
                // = " + length);
            }
        }
    }
}
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      TCP5Client.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class TCP5Client {

    public static void main(String argv[]) throws Exception {
        String sentence;
        String quote;

        try (BufferedReader inFromUser = new BufferedReader(new
InputStreamReader(System.in));
                Socket clientSocket = new Socket(InetAddress.getLocalHost(),
1235);
                DataOutputStream outToServer = new
DataOutputStream(clientSocket.getOutputStream());
                BufferedReader inFromServer = new BufferedReader(
                        new InputStreamReader(clientSocket.getInputStream()));)
{
            sentence = inFromUser.readLine();
            outToServer.writeBytes(sentence + '\n');
            quote = inFromServer.readLine();
            System.out.println("From Server: " + quote);
        }
    }
}
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      TCP5Server.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
```

```
 *
 * ========================================================== */

 import java.io.*;
import java.net.*;
import java.lang.Math;

public class TCP5Server {

    public static void main(String argv[]) throws Exception {
        String clientSentence;
        int random;
        String[] quotes = { "You are nice Person.", "You are good.", "You are
lucky.", "You have good choice.",
                "You can do it.", "You are Awesome.", "You are Smart.", "You are
my friend.", "You are stupid.",
                "Lucky to have you here." };

        ServerSocket welcomeSocket = new ServerSocket(1235);

        while (true) {
            Socket connectionSocket = welcomeSocket.accept();
            try (BufferedReader inFromClient = new BufferedReader(
                    new InputStreamReader(connectionSocket.getInputStream()));
                    DataOutputStream outToClient = new
DataOutputStream(connectionSocket.getOutputStream());) {
                clientSentence = inFromClient.readLine();
                if (clientSentence.equalsIgnoreCase("hello")) {
                    random = (int) (Math.random() * 10);
                    outToClient.writeBytes(quotes[random]);
                } else {
                    outToClient.writeBytes("Hi, say hello for quotes.");
                }
            }
        }
    }
}
 /* ==========================================================
 *
 * Roll No: 30
 *
 * File:      TCP6Client.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ========================================================== */

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

public class TCP6Client {
```

```
    public static void main(String argv[]) throws Exception {
        String sentence;
        String age;

        try (BufferedReader inFromUser = new BufferedReader(new
InputStreamReader(System.in));
                Socket clientSocket = new Socket(InetAddress.getLocalHost(),
6789);
                DataOutputStream outToServer = new
DataOutputStream(clientSocket.getOutputStream());
                BufferedReader inFromServer = new BufferedReader(
                        new InputStreamReader(clientSocket.getInputStream()));)
{
            System.out.print("Enter Birth Date (eg : day/month/year) : ");
            sentence = inFromUser.readLine();
            outToServer.writeBytes(sentence + '\n');
            age = inFromServer.readLine();
            System.out.println("From Server: " + age);
        }
    }
}
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      TCP6Server.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

 import java.io.*;
import java.net.*;
import java.util.Calendar;

public class TCP6Server {

    public static void main(String argv[]) throws Exception {
        String clientSentence;
        int year, month, day;

        ServerSocket welcomeSocket = new ServerSocket(6789);

        while (true) {
            Socket connectionSocket = welcomeSocket.accept();
            try (BufferedReader inFromClient = new BufferedReader(
                    new InputStreamReader(connectionSocket.getInputStream()));
                    DataOutputStream outToClient = new
DataOutputStream(connectionSocket.getOutputStream());) {
                try {
                    clientSentence = inFromClient.readLine();
```

```
                    day = Integer.parseInt(clientSentence.substring(0,
clientSentence.indexOf("/")));
                    month = Integer.parseInt(
                            clientSentence.substring(clientSentence.indexOf("/")
+ 1, clientSentence.lastIndexOf("/")));
                    year = Integer.parseInt(

clientSentence.substring(clientSentence.lastIndexOf("/") + 1,
clientSentence.length()));

                    Calendar cal = Calendar.getInstance();
                    int tday = cal.get(Calendar.DAY_OF_MONTH);
                    int tmonth = cal.get(Calendar.MONTH) + 1;
                    int tyear = cal.get(Calendar.YEAR);

                    GetAge gage = new GetAge();

                    outToClient.writeBytes(gage.age(tday, tmonth, tyear, day,
month, year));

                } catch (Exception e) {
                    outToClient.writeBytes("Invalid Dates");
                }
            }
        }
    }
}

class GetAge {
    private int finalyear;
    private int finalmonth;
    private int finaldays;

    public String age(int tday, int tmonth, int tyear, int day, int month, int
year) {
        finalyear = tyear - year - 1;
        finaldays = tday;
        if (tmonth >= month && tday >= day) {
            finalyear++;
            finalmonth = tmonth - month;
            finaldays = tday - day;
        } else {
            finalmonth = ((12 - month) + tmonth) - 1;
        }
        if (finalmonth == 0 && finaldays == 0)
            return "Happy Birthday your Age = " + finalyear + " years " +
finalmonth + " months " + finaldays
                    + " days ";
        else
            return "Age = " + finalyear + " years " + finalmonth + " months " +
finaldays + " days ";
```

```
    }
}
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      TCP7Client.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class TCP7Client {
    private final String host;
    private final int port;
    private String username;

    public TCP7Client(String host, int port) {
        this.host = host;
        this.port = port;
    }

    public static void main(String[] args) {
        TCP7Client client = new TCP7Client("localhost", 6789);
        client.start();
    }

    public void start() {
        try {
            Socket socket = new Socket(host, port);
            System.out.println("Connected to the server...");
            var writeHandler = new TCP7ClientWriter(socket, this);
            var readHandler = new TCP7ClientReader(socket, this);
            writeHandler.start();
            readHandler.start();
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }
    }

    void setUserName(String username) {
        this.username = username;
    }

    String getUsername() {
        return this.username;
    }
}
 /* ============================================================
```

```
 *
 * Roll No: 30
 *
 * File:      TCP7ClientReader.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ========================================================== */

 package TCP;

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

public class TCP7ClientReader extends Thread {
    private BufferedReader reader;
    private final Socket socket;
    private final TCP7Client client;

    public TCP7ClientReader(Socket socket, TCP7Client client) {
        this.socket = socket;
        this.client = client;
        try {
            reader = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }
    }

    @Override
    public void run() {
        while (true) {
            try {
                String response = reader.readLine();
                System.out.println("\n" + response);
                if (client.getUsername() != null) {
                    System.out.print("From " + client.getUsername() + ":");
                }
            } catch (Exception e) {
                System.out.println("Error: " + e.getMessage());
                break;
            }
        }
    }
}
 /* ==========================================================
 *
 * Roll No: 30
 *
 * File:      TCP7ClientWriter.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
```

```
 *
 * ========================================================== */

 package TCP;

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

public class TCP7ClientWriter extends Thread {
    private PrintWriter writer;
    private final Socket socket;
    private final TCP7Client client;

    public TCP7ClientWriter(Socket socket, TCP7Client client) {
        this.socket = socket;
        this.client = client;
        try {
            OutputStream output = socket.getOutputStream();
            writer = new PrintWriter(output, true);
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }
    }

    @Override
    public void run() {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter your name: ");
        String userName = sc.nextLine();
        client.setUserName(userName);
        writer.println(userName);
        String text;
        do {
            System.out.print("From " + userName + ": ");
            text = sc.nextLine();
            writer.println(text);
        } while (!text.equalsIgnoreCase("bye"));
        try {
            socket.close();
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
} /* ==========================================================
 *
 * Roll No: 30
 *
 * File:      TCP7Server.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
```

```
 * ========================================================== */

 package TCP;

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

public class TCP7Server {
    private final Map<String, ClientHandler> users = new HashMap<>();

    public static void main(String[] args) {
        TCP7Server server = new TCP7Server();
        server.start();
    }

    public void start() {
        try (ServerSocket serverSocket = new ServerSocket(6789)) {
            System.out.println("----- Server -----");
            System.out.println("Listening on port 6789...");
            while (true) {
                Socket socket = serverSocket.accept();
                System.out.println("New User Connected...");
                ClientHandler newUser = new ClientHandler(socket, this);
                newUser.start();
            }
        } catch (IOException e) {
            System.out.println("Error in the server: " + e.getMessage());
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }
    }

    void sendToEveryone(String message, ClientHandler notToUser) {
        try {
            for (Map.Entry<String, ClientHandler> aUser : users.entrySet()) {
                if (aUser.getValue() != notToUser) {
                    aUser.getValue().sendMessage(message);
                }
            }
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }
    }

    void addUser(String userName, ClientHandler userThread) {
        users.put(userName, userThread);
    }

    void removeUser(String userName) {
        users.remove(userName);
```

```
    }

    void listUsers() {
        System.out.println("--- USERS ---");
        users.keySet().stream().forEach(System.out::println);
    }
} /* ============================================================
 *
 * Roll No: 30
 *
 * File:      TCP8Client.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class TCP8Client {
    public static void main(String args[]) throws Exception {
        Socket clientSocket = new Socket(InetAddress.getLocalHost(), 6789);
        DataOutputStream outToServer = new
DataOutputStream(clientSocket.getOutputStream());
        BufferedReader inFromServer = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
        String response = inFromServer.readLine();
        System.out.println("Response from Server: " + response);
        clientSocket.close();
    }
}
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      TCP8Server.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

 import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Arrays;

public class TCP8Server {
    public static void main(String[] args) throws IOException {
        ServerSocket welcomeSocket = new ServerSocket(6789);
        while (true) {
```

```
            Socket connectionSocket = welcomeSocket.accept();
            BufferedReader inFromClient = new BufferedReader(new
InputStreamReader(connectionSocket.getInputStream()));
            DataOutputStream outToClient = new
DataOutputStream(connectionSocket.getOutputStream());
            String ip = connectionSocket.getInetAddress().toString();
            String ipAddress = ip.substring(1, ip.length());
            System.out.println("Request from client: " + ipAddress);
            String response = "Your IP Address is " + ipAddress + ", ";
            response += IPAddress.getIPClass(new IPAddress(ipAddress));
            outToClient.writeBytes(response + "\n");
        }
    }
}

public class IPAddress implements Comparable<IPAddress> {
    int a, b, c, d;

    IPAddress(String ip) {
        if (isValid(ip))
            parse(ip);
        else {
            a = b = c = d = 0;
        }
    }

    private void parse(String ip) {
        int[] data =
Arrays.stream(ip.split("[.]")).mapToInt(Integer::parseInt).toArray();
        a = data[0];
        b = data[1];
        c = data[2];
        d = data[3];
    }

    public static boolean isValid(String ip) {
        String[] data = ip.split("[.]");
        if (data.length < 4 || data.length > 4)
            return false;

        return Arrays.stream(data).allMatch(value -> {
            int val = Integer.parseInt(value);
            return (val >= 0 && val < 256);
        });
    }

    public static String getIPClass(IPAddress ip) {
        if (ip.a >= 0 && ip.a < 128)
            return "Class A";
        if (ip.a >= 128 && ip.a < 192)
            return "Class B";
```

```
        if (ip.a >= 192 && ip.a < 224)
            return "Class C";
        if (ip.a >= 224 && ip.a < 240)
            return "Class D";
        return "Class E";
    }

    public static boolean isSpecial(IPAddress ip) {
        if (ip.compareTo(new IPAddress("0.0.0.0")) >= 0 && ip.compareTo(new
IPAddress("0.255.255.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("10.0.0.0")) >= 0 && ip.compareTo(new
IPAddress("10.255.255.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("100.64.0.0")) >= 0 && ip.compareTo(new
IPAddress("100.127.255.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("127.0.0.0")) >= 0 && ip.compareTo(new
IPAddress("127.255.255.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("169.254.0.0")) >= 0 && ip.compareTo(new
IPAddress("169.254.255.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("172.16.0.0")) >= 0 && ip.compareTo(new
IPAddress("172.31.255.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("192.0.0.0")) >= 0 && ip.compareTo(new
IPAddress("192.0.0.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("192.0.2.0")) >= 0 && ip.compareTo(new
IPAddress("192.0.2.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("192.88.99.0")) >= 0 && ip.compareTo(new
IPAddress("192.88.99.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("192.168.0.0")) >= 0 && ip.compareTo(new
IPAddress("192.168.255.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("198.18.0.0")) >= 0 && ip.compareTo(new
IPAddress("198.19.255.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("198.51.100.0")) >= 0 && ip.compareTo(new
IPAddress("198.51.100.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("203.0.113.0")) >= 0 && ip.compareTo(new
IPAddress("203.0.113.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("224.0.0.0")) >= 0 && ip.compareTo(new
IPAddress("239.255.255.255")) <= 0)
            return true;
```

```
        if (ip.compareTo(new IPAddress("240.0.0.0")) >= 0 && ip.compareTo(new
IPAddress("255.255.255.255")) <= 0)
            return true;
        return false;
    }

    @Override
    public int compareTo(IPAddress anotherIP) {
        if (a > anotherIP.a)
            return 1;
        else if (a < anotherIP.a)
            return -1;
        else {
            if (b > anotherIP.b)
                return 1;
            else if (b < anotherIP.b)
                return -1;
            else {
                if (c > anotherIP.c)
                    return 1;
                else if (c < anotherIP.c)
                    return -1;
                else {
                    if (d > anotherIP.d)
                        return 1;
                    else if (d < anotherIP.d)
                        return -1;
                    else
                        return 0;
                }
            }
        }
    }

    @Override
    public String toString() {
        return a + "." + b + "." + c + "." + d;
    }
} /* ============================================================
 *
 * Roll No: 30
 *
 * File:      TCP9Client.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class TCP9Client {
```

```
    public static void main(String args[]) throws Exception {
        Socket clientSocket = new Socket(InetAddress.getLocalHost(), 6789);
        DataOutputStream outToServer = new
DataOutputStream(clientSocket.getOutputStream());
        BufferedReader inFromServer = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
        String response = inFromServer.readLine();
        System.out.println("From Server: " + response);
        clientSocket.close();
    }
}
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      TCP9Server.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

 import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Arrays;

public class TCP9Server {
    public static void main(String[] args) throws IOException {
        ServerSocket welcomeSocket = new ServerSocket(6789);
        while (true) {
            Socket connectionSocket = welcomeSocket.accept();
            BufferedReader inFromClient = new BufferedReader(new
InputStreamReader(connectionSocket.getInputStream()));
            DataOutputStream outToClient = new
DataOutputStream(connectionSocket.getOutputStream());
            String ip = connectionSocket.getInetAddress().toString();
            String ipAddress = ip.substring(1, ip.length());
            System.out.println("From Client: " + ipAddress);
            String response = "Your IP Address is " + ipAddress;
            if (IPAddress.isSpecial(new IPAddress(ipAddress)))
                response += ", Your IP is special.";
            else
                response += ", Your IP is not special.";
            outToClient.writeBytes(response + "\n");
        }
    }
}

public class IPAddress implements Comparable<IPAddress> {
```

```
    int a, b, c, d;

    IPAddress(String ip) {
        if (isValid(ip))
            parse(ip);
        else {
            a = b = c = d = 0;
        }
    }

    private void parse(String ip) {
        int[] data =
Arrays.stream(ip.split("[.]")).mapToInt(Integer::parseInt).toArray();
        a = data[0];
        b = data[1];
        c = data[2];
        d = data[3];
    }

    public static boolean isValid(String ip) {
        String[] data = ip.split("[.]");
        if (data.length < 4 || data.length > 4)
            return false;

        return Arrays.stream(data).allMatch(value -> {
            int val = Integer.parseInt(value);
            return (val >= 0 && val < 256);
        });
    }

    public static String getIPClass(IPAddress ip) {
        if (ip.a >= 0 && ip.a < 128)
            return "Class A";
        if (ip.a >= 128 && ip.a < 192)
            return "Class B";
        if (ip.a >= 192 && ip.a < 224)
            return "Class C";
        if (ip.a >= 224 && ip.a < 240)
            return "Class D";
        return "Class E";
    }

    public static boolean isSpecial(IPAddress ip) {
        if (ip.compareTo(new IPAddress("0.0.0.0")) >= 0 && ip.compareTo(new
IPAddress("0.255.255.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("10.0.0.0")) >= 0 && ip.compareTo(new
IPAddress("10.255.255.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("100.64.0.0")) >= 0 && ip.compareTo(new
IPAddress("100.127.255.255")) <= 0)
```

```
            return true;
        if (ip.compareTo(new IPAddress("127.0.0.0")) >= 0 && ip.compareTo(new
IPAddress("127.255.255.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("169.254.0.0")) >= 0 && ip.compareTo(new
IPAddress("169.254.255.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("172.16.0.0")) >= 0 && ip.compareTo(new
IPAddress("172.31.255.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("192.0.0.0")) >= 0 && ip.compareTo(new
IPAddress("192.0.0.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("192.0.2.0")) >= 0 && ip.compareTo(new
IPAddress("192.0.2.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("192.88.99.0")) >= 0 && ip.compareTo(new
IPAddress("192.88.99.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("192.168.0.0")) >= 0 && ip.compareTo(new
IPAddress("192.168.255.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("198.18.0.0")) >= 0 && ip.compareTo(new
IPAddress("198.19.255.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("198.51.100.0")) >= 0 && ip.compareTo(new
IPAddress("198.51.100.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("203.0.113.0")) >= 0 && ip.compareTo(new
IPAddress("203.0.113.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("224.0.0.0")) >= 0 && ip.compareTo(new
IPAddress("239.255.255.255")) <= 0)
            return true;
        if (ip.compareTo(new IPAddress("240.0.0.0")) >= 0 && ip.compareTo(new
IPAddress("255.255.255.255")) <= 0)
            return true;
        return false;
    }

    @Override
    public int compareTo(IPAddress anotherIP) {
        if (a > anotherIP.a)
            return 1;
        else if (a < anotherIP.a)
            return -1;
        else {
            if (b > anotherIP.b)
                return 1;
            else if (b < anotherIP.b)
                return -1;
```

```
            else {
                if (c > anotherIP.c)
                    return 1;
                else if (c < anotherIP.c)
                    return -1;
                else {
                    if (d > anotherIP.d)
                        return 1;
                    else if (d < anotherIP.d)
                        return -1;
                    else
                        return 0;
                }
            }
        }
    }

    @Override
    public String toString() {
        return a + "." + b + "." + c + "." + d;
    }
} /* ============================================================
 *
 * Roll No: 30
 *
 * File:      UDP10Client.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

 import java.io.*;
import java.net.*;
import java.nio.ByteBuffer;
import java.util.Arrays;

public class UDP10Client {
    private static String source = "101";
    private static String destination = "102";

    public static void main(String args[]) throws Exception {
        String sentence;
        BufferedReader inFromUser = new BufferedReader(new
InputStreamReader(System.in));
        Socket clientSocket = new Socket(InetAddress.getLocalHost(), 6789);
        DataOutputStream outToServer = new
DataOutputStream(clientSocket.getOutputStream());
        DataInputStream inFromServer = new
DataInputStream(clientSocket.getInputStream());
        sentence = inFromUser.readLine();
        byte[] sourceAddress = Arrays.copyOf(source.getBytes(), 10);
        byte[] destinationAddress = Arrays.copyOf(destination.getBytes(), 10);
```

```
        byte[] data = Arrays.copyOf(sentence.getBytes(), 50);
        byte[] packet =
ByteBuffer.allocate(70).put(sourceAddress).put(destinationAddress).put(data).arr
ay();
        outToServer.write(packet);
        byte[] response = new byte[90];
        inFromServer.read(response);
        String rSourceAddress = new String(Arrays.copyOfRange(response, 0, 10));
        String rDestinationAddress = new String(Arrays.copyOfRange(response, 10,
20));
        String sSourceAddress = new String(Arrays.copyOfRange(response, 20,
30));
        String sDestinationAddress = new String(Arrays.copyOfRange(response, 30,
40));
        String receivedData = new String(Arrays.copyOfRange(response, 40, 90));
        System.out.println("Sender Source Address: " + sSourceAddress);
        System.out.println("Sender Destination Address: " +
sDestinationAddress);
        System.out.println("Receiver Source Address: " + rSourceAddress);
        System.out.println("Receiver Destination Address: " +
rDestinationAddress);
        System.out.println("Data: " + receivedData);
        clientSocket.close();
    }
}
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      UDP10Server.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

 import java.io.*;
import java.net.*;
import java.nio.ByteBuffer;
import java.util.Arrays;

public class UDP10Server {
    private static String source = "102";
    private static String destination = "103";

    public static void main(String args[]) throws Exception {
        ServerSocket welcomeSocket = new ServerSocket(6789);
        while (true) {
            Socket connectionSocket = welcomeSocket.accept();
            DataInputStream inFromClient = new
DataInputStream(connectionSocket.getInputStream());
            DataOutputStream outToClient = new
DataOutputStream(connectionSocket.getOutputStream());
```

```
            byte[] clientData = new byte[70];
            inFromClient.read(clientData);
            String sSourceAddress = new String(Arrays.copyOfRange(clientData, 0,
10));
            String sDestinationAddress = new
String(Arrays.copyOfRange(clientData, 10, 20));
            String receivedData = new String(Arrays.copyOfRange(clientData, 20,
70));
            System.out.println("_____From Client_____");
            System.out.println("Sender Source Address: " + sSourceAddress);
            System.out.println("Sender Destination Address: " +
sDestinationAddress);
            System.out.println("Client - Data: " + receivedData);
            byte[] sourceAddress = Arrays.copyOf(source.getBytes(), 10);
            byte[] destinationAddress = Arrays.copyOf(destination.getBytes(),
10);
            byte[] newPacket =
ByteBuffer.allocate(90).put(sourceAddress).put(destinationAddress).put(clientDat
a)
                    .array();
            outToClient.write(newPacket);
        }
    }
}
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      UDP1Client.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class UDP1Client {
    public static void main(String args[]) throws Exception {
        BufferedReader inFromUser = new BufferedReader(new
InputStreamReader(System.in));
        DatagramSocket clientSocket = new DatagramSocket();
        InetAddress IPAddress = InetAddress.getLocalHost();
        byte[] sendData = new byte[1024];
        String clientSentence;

        do {
            clientSentence = inFromUser.readLine();
            sendData = clientSentence.getBytes();
            DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, IPAddress, 6789);
            clientSocket.send(sendPacket);
```

```
        } while (!clientSentence.equalsIgnoreCase("bye") && !
clientSentence.equalsIgnoreCase("exit"));
        clientSocket.close();
    }
}
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      UDP1Server.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

 import java.net.*;

public class UDP1Server {
    public static void main(String args[]) throws Exception {
        DatagramSocket serverSocket = new DatagramSocket(6789);
        while (true) {
            byte[] receiveData = new byte[1024];
            DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
            serverSocket.receive(receivePacket);
            String clientSentence = (new
String(receivePacket.getData())).trim();
            System.out.println("Client Says : " + clientSentence);
            if (clientSentence.equalsIgnoreCase("bye") ||
clientSentence.equalsIgnoreCase("exit")) {
                serverSocket.close();
                return;
            }
        }
    }
}
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      UDP2Client.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class UDP2Client {
    public static void main(String args[]) throws Exception {
        BufferedReader inFromUser = new BufferedReader(new
InputStreamReader(System.in));
```

```
        DatagramSocket clientSocket = new DatagramSocket();
        InetAddress IPAddress = InetAddress.getLocalHost();
        byte[] sendData = new byte[1024];
        byte[] receiveData = new byte[1024];
        String sentence = inFromUser.readLine();
        sendData = sentence.getBytes();
        DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, IPAddress, 6789);
        clientSocket.send(sendPacket);
        DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
        clientSocket.receive(receivePacket);
        String modifiedSentence = (new String(receivePacket.getData())).trim();
        System.out.println("From Server: " + modifiedSentence);
        clientSocket.close();
    }

}
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      UDP2Server.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

 import java.net.*;

class UDP2Server {
    public static void main(String args[]) throws Exception {
        DatagramSocket serverSocket = new DatagramSocket(6789);
        while (true) {
            byte[] receivedData = new byte[1024];
            byte[] upperCase = new byte[1024];
            DatagramPacket receivedPacket = new DatagramPacket(receivedData,
receivedData.length);
            serverSocket.receive(receivedPacket);
            String sentence = (new String(receivedPacket.getData())).trim();
            // System.out.println("Client Sended : " + sentence);
            InetAddress IPAddress = receivedPacket.getAddress();
            int port = receivedPacket.getPort();
            String capitalizedSentence = sentence.toUpperCase();
            upperCase = capitalizedSentence.getBytes();
            // String upperCase = sentence.toUpperCase();
            System.out.println("Client Request String = " + sentence + " |
Response = " + capitalizedSentence);
            DatagramPacket sendPacket = new DatagramPacket(upperCase,
upperCase.length, IPAddress, port);
            serverSocket.send(sendPacket);
        }
```

```
    }
}
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      UDP3Client.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class UDP3Client {
    public static void main(String args[]) throws Exception {
        BufferedReader inFromUser = new BufferedReader(new
InputStreamReader(System.in));
        DatagramSocket clientSocket = new DatagramSocket();
        InetAddress IPAddress = InetAddress.getLocalHost();
        byte[] sendData = new byte[1024];
        byte[] receiveData = new byte[1024];
        String sentence = inFromUser.readLine();
        sendData = sentence.getBytes();
        DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, IPAddress, 6789);
        clientSocket.send(sendPacket);
        DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
        clientSocket.receive(receivePacket);
        String length = (new String(receivePacket.getData()).trim());
        System.out.println("From Server: " + length);
        clientSocket.close();
    }
}
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      UDP3Server.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class UDP3Server {
    public static void main(String argv[]) throws Exception {
        DatagramSocket serverSocket = new DatagramSocket(6789);
        while (true) {
```

```
            byte[] receivedData = new byte[1024];
            byte[] len = new byte[1024];
            DatagramPacket receivedPacket = new DatagramPacket(receivedData,
receivedData.length);
            serverSocket.receive(receivedPacket);
            String sentence = (new String(receivedPacket.getData())).trim();
            InetAddress IPAddress = receivedPacket.getAddress();
            int port = receivedPacket.getPort();
            len = ("" + sentence.length()).getBytes();
            System.out.println("Client Request String = " + sentence + " |
Response = " + sentence.length());
            DatagramPacket sendPacket = new DatagramPacket(len, len.length,
IPAddress, port);
            serverSocket.send(sendPacket);
        }
    }
}
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      UDP4Client.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class UDP4Client {

    public static void main(String[] args) throws Exception {
        String sentence;
        System.out.println("Enter Time as per 24 Hours Format (eg 17:50)");
        BufferedReader inFromUser = new BufferedReader(new
InputStreamReader(System.in));
        sentence = inFromUser.readLine();
        System.out.println(sendAndReceivePacket(sentence));
    }

    private static String sendAndReceivePacket(String time) throws Exception {
        DatagramSocket clientSocket = new DatagramSocket();
        InetAddress ipAddress = InetAddress.getLocalHost();
        byte[] sendData = time.getBytes();
        DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, ipAddress, 6789);
        clientSocket.send(sendPacket);
        byte[] receiveData = new byte[1024];
        DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
        clientSocket.receive(receivePacket);
```

```
        String greeting = (new String(receivePacket.getData())).trim();
        clientSocket.close();
        return greeting;
    }
} /* ============================================================
 *
 * Roll No: 30
 *
 * File:      UDP4Server.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

 import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;

public class UDP4Server {
    public static void main(String[] args) throws Exception {
        DatagramSocket serverSocket = new DatagramSocket(6789);
        byte[] recieveData = new byte[1024];
        String greeting;
        int time = 0;
        int minute = 0;
        while (true) {
            DatagramPacket receivePacket = new DatagramPacket(recieveData,
recieveData.length);
            serverSocket.receive(receivePacket);
            String clientSentence = new String(receivePacket.getData()).trim();
            try {
                time = Integer.parseInt(clientSentence.substring(0,
clientSentence.indexOf(":")));
                minute = Integer
                        .parseInt(clientSentence.substring(clientSentence.indexO
f(":") + 1, clientSentence.length()));

                if ((time < 0 || time > 23) || (minute > 59 || minute < 0))
                    greeting = "Invalid Time";
                else if (time > 19)
                    greeting = "Good Night";
                else if (time > 15)
                    greeting = "Good Evening";
                else if (time > 11)
                    greeting = "Good Afernoon";
                else if (time > 4)
                    greeting = "Good Morning";
                else
                    greeting = "Good MidNight";

            } catch (Exception e) {
                greeting = "Invalid Time ex";
```

```
            }

            InetAddress ipAddress = receivePacket.getAddress();
            int port = receivePacket.getPort();
            byte[] sendData = greeting.getBytes();
            DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, ipAddress, port);
            serverSocket.send(sendPacket);
        }
    }
} /* ============================================================
 *
 * Roll No: 30
 *
 * File:      UDP5Client.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class UDP5Client {

    public static void main(String[] args) throws Exception {
        String sentence;
        BufferedReader inFromUser = new BufferedReader(new
InputStreamReader(System.in));
        sentence = inFromUser.readLine();
        DatagramSocket clientSocket = new DatagramSocket();
        InetAddress ipAddress = InetAddress.getLocalHost();
        byte[] sendData = sentence.getBytes();
        DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, ipAddress, 6789);
        clientSocket.send(sendPacket);
        byte[] receiveData = new byte[1024];
        DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
        clientSocket.receive(receivePacket);
        String greeting = (new String(receivePacket.getData())).trim();
        System.out.println(greeting);
        clientSocket.close();
    }
} /* ============================================================
 *
 * Roll No: 30
 *
 * File:      UDP5Server.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */
```

```java
 import java.net.*;
import java.lang.Math;

public class UDP5Server {
    public static void main(String[] args) throws Exception {
        DatagramSocket serverSocket = new DatagramSocket(6789);
        byte[] recieveData = new byte[1024];
        byte[] sendData = new byte[1024];
        String selectedData;
        int random;
        String[] quotes = { "You are nice Person.", "You are good.", "You are
lucky.", "You have good choice.",
                "You can do it.", "You are Awesome.", "You are Smart.", "You are
my friend.", "You are stupid.",
                "Lucky to have you here." };
        while (true) {
            DatagramPacket receivePacket = new DatagramPacket(recieveData,
recieveData.length);
            serverSocket.receive(receivePacket);
            String clientSentence = new String(receivePacket.getData()).trim();
            InetAddress ipAddress = receivePacket.getAddress();
            int port = receivePacket.getPort();
            if (clientSentence.equalsIgnoreCase("hello")) {
                random = (int) (Math.random() * 10);
                selectedData = quotes[random];
            } else {
                selectedData = "Hi, say hello for quotes.";
            }
            sendData = selectedData.getBytes();
            DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, ipAddress, port);
            serverSocket.send(sendPacket);
        }
    }
} /* ============================================================
 *
 * Roll No: 30
 *
 * File:      UDP6Client.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class UDP6Client {

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

```
        BufferedReader inFromUser = new BufferedReader(new
InputStreamReader(System.in));
        System.out.print("Enter Birth Date (eg : day/month/year) : ");
        sentence = inFromUser.readLine();
        DatagramSocket clientSocket = new DatagramSocket();
        InetAddress ipAddress = InetAddress.getLocalHost();
        byte[] sendData = sentence.getBytes();
        DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, ipAddress, 6789);
        clientSocket.send(sendPacket);
        byte[] receiveData = new byte[1024];
        DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
        clientSocket.receive(receivePacket);
        String greeting = (new String(receivePacket.getData())).trim();
        System.out.println(greeting);
        clientSocket.close();
    }
} /* ============================================================
 *
 * Roll No: 30
 *
 * File:      UDP6Server.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

 import java.net.*;
import java.util.Calendar;

public class UDP6Server {
    public static void main(String[] args) throws Exception {
        DatagramSocket serverSocket = new DatagramSocket(6789);
        byte[] recieveData = new byte[1024];
        byte[] sendData = new byte[1024];
        int year, month, day;
        String selectedData;
        while (true) {
            DatagramPacket receivePacket = new DatagramPacket(recieveData,
recieveData.length);
            serverSocket.receive(receivePacket);
            String clientSentence = new String(receivePacket.getData()).trim();
            InetAddress ipAddress = receivePacket.getAddress();
            int port = receivePacket.getPort();
            try {
                day = Integer.parseInt(clientSentence.substring(0,
clientSentence.indexOf("/")));
                month = Integer.parseInt(
                        clientSentence.substring(clientSentence.indexOf("/") +
1, clientSentence.lastIndexOf("/")));
                year = Integer.parseInt(
```

```
                        clientSentence.substring(clientSentence.lastIndexOf("/")
+ 1, clientSentence.length()));

                Calendar cal = Calendar.getInstance();
                int tday = cal.get(Calendar.DAY_OF_MONTH);
                int tmonth = cal.get(Calendar.MONTH) + 1;
                int tyear = cal.get(Calendar.YEAR);

                GetAge gage = new GetAge();

                selectedData = gage.age(tday, tmonth, tyear, day, month, year);

            } catch (Exception e) {
                selectedData = "Invalid Dates";
            }
            sendData = selectedData.getBytes();
            DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, ipAddress, port);
            serverSocket.send(sendPacket);
        }
    }
}

class GetAge {
    private int finalyear;
    private int finalmonth;
    private int finaldays;

    public String age(int tday, int tmonth, int tyear, int day, int month, int
year) {
        finalyear = tyear - year - 1;
        finaldays = tday;
        if (tmonth >= month && tday >= day) {
            finalyear++;
            finalmonth = tmonth - month;
            finaldays = tday - day;
        } else {
            finalmonth = ((12 - month) + tmonth) - 1;
        }
        if (finalmonth == 0 && finaldays == 0)
            return "Happy Birthday your Age = " + finalyear + " years " +
finalmonth + " months " + finaldays
                    + " days ";
        else
            return "Age = " + finalyear + " years " + finalmonth + " months " +
finaldays + " days ";
    }
} /* ============================================================


/* ============================================================
```

```
 * ============================================================
 PRACTICAL - II
 * ============================================================
 * ============================================================ */

 *
 * Roll No: 30
 *
 * File:      01-ChatClient.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class ChatClient {

    private final String host;
    private final int port;
    private String username;

    public ChatClient(String host, int port) {
        this.host = host;
        this.port = port;
    }

    public void start() {
        try {
            Socket socket = new Socket(host, port);

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

            var writeHandler = new ClientWriteHandler(socket, this);
            writeHandler.start();
            (new ClientReadHandler(socket, this)).start();

        } catch (UnknownHostException e) {
            System.out.println("Server not found: " + e.getMessage());
        } catch (IOException e) {
            System.out.println("I/O Error: " + e.getMessage());
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }
    }

    void setUserName(String username) {
        this.username = username;
    }
```

```
    String getUsername() {
        return this.username;
    }

    public static void main(String[] args) {
        ChatClient client = new ChatClient("127.0.0.1", 5678);
        client.start();
    }
}

// $ java -cp bin/ ChatClient
// Connected to the server...

// No other users connected
// Enter your name: Nirav
// [Nirav]:
// New User Connected: Milind
// [Nirav]:
// [Milind]: Hey
// [Nirav]: Hello
// [Nirav]:
// New User Connected: Pradip
// [Nirav]:
// [Pradip]: Hello Everyone
// [Nirav]:
// [Pradip]: exir
// [Nirav]:
// [Pradip]: exit
// [Nirav]:
// Pradip disconnected...
// [Nirav]:
// [Milind]: Bye
// [Nirav]:
// [Milind]: exit
// [Nirav]:
// Milind disconnected...
// [Nirav]: bye
// [Nirav]: exit

// $ java -cp bin/ ChatClient
// Connected to the server...

// Connected users: [Nirav, Milind]
// Enter your name: Pradip
// [Pradip]: Hello Everyone
// [Pradip]: exir
// [Pradip]: exit

// $ java -cp bin/ ChatClient
// Connected to the server...
```

```
// // Connected users: [Nirav]
// // Enter your name: Milind
// // [Milind]: Hey
// // [Milind]:
// // [Nirav]: Hello
// // [Milind]:
// // New User Connected: Pradip
// // [Milind]:
// // [Pradip]: Hello Everyone
// // [Milind]:
// // [Pradip]: exir
// // [Milind]:
// // [Pradip]: exit
// // [Milind]:
// // Pradip disconnected...
// // [Milind]: Bye
// // [Milind]: exit
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      01-ChatServer.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class ChatServer {
    private final Map<String, ClientHandler> users = new HashMap<>();
    private final int SERVER_PORT;

    public ChatServer(int port) {
        this.SERVER_PORT = port;
    }

    public void start() {
        try (ServerSocket serverSocket = new ServerSocket(SERVER_PORT)) {

            System.out.println("----- Chat Server -----");
            System.out.println("Listening on port " + SERVER_PORT + "...");

            while (true) {
                Socket socket = serverSocket.accept();
                System.out.println("New User Connected...");
                ClientHandler newUser = new ClientHandler(socket, this);
                newUser.start();
            }
        } catch (IOException e) {
```

```
            System.out.println("Error in the server: " + e.getMessage());
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }
    }

    public static void main(String[] args) {
        int port;
        try {
            port = 5678;
            ChatServer server = new ChatServer(port);
            server.start();
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }
    }

    void sendToEveryone(String message, ClientHandler notToUser) {
        for (Map.Entry<String, ClientHandler> aUser : users.entrySet()) {
            if (aUser.getValue() != notToUser) {
                aUser.getValue().sendMessage(message);
            }
        }
    }

    void sendTo(String message, String username) {
        var user = getUser(username);
        if (user != null) {
            user.sendMessage(message);
        }
    }

    private ClientHandler getUser(String username) {
        for (Map.Entry<String, ClientHandler> aUser : users.entrySet()) {
            if (aUser.getKey().equals(username)) {
                return aUser.getValue();
            }
        }
        return null;
    }

    void addUser(String userName, ClientHandler userThread) {
        users.put(userName, userThread);
    }

    void removeUser(String userName) {
        users.remove(userName);
    }

    Set<String> getUserNames() {
        return this.users.keySet();
```

```
    }

    boolean hasUsers() {
        return !this.users.isEmpty();
    }
}

// $ java -d bin/ *.ajav && java -cp bin/ ChatServer
// ----- Chat Server -----
// Listening on port 5678...
// New User Connected...
// New User Connected...
// New User Connected... /*
============================================================
 *
 * Roll No: 30
 *
 * File:      01-ClientHandler.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class ClientHandler extends Thread {
    private final Socket socket;
    private final ChatServer server;
    private PrintWriter writer;
    public String username;

    public ClientHandler(Socket socket, ChatServer server) {
        this.socket = socket;
        this.server = server;
    }

    @Override
    public void run() {
        try {
            BufferedReader reader = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
            writer = new PrintWriter(socket.getOutputStream(), true);

            listUsers();

            username = reader.readLine();
            server.addUser(username, this);

            String serverMessage = "New User Connected: " + username;
            server.sendToEveryone(serverMessage, this);
```

```
            String clientMessage;

            do {
                clientMessage = reader.readLine().trim();
                serverMessage = "[" + username + "]: " + clientMessage;

                if (clientMessage.startsWith("@")) {
                    int indexOfSpace = clientMessage.indexOf(' ');
                    if (indexOfSpace < 2) {
                        server.sendToEveryone(serverMessage, this);
                    } else {
                        System.out.println(clientMessage.substring(1,
indexOfSpace));
                        server.sendTo(serverMessage, clientMessage.substring(1,
indexOfSpace));
                    }
                } else {
                    server.sendToEveryone(serverMessage, this);
                }
            } while (!clientMessage.equals("exit"));
        } catch (IOException e) {
            System.out.println("Error in ClientHandler: " + e.getMessage());
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        } finally {
            try {
                server.removeUser(username);
                socket.close();
                String serverMessage = username + " disconnected...";
                server.sendToEveryone(serverMessage, this);
            } catch (IOException e) {
                System.out.println("Error: " + e.getMessage());
            }
        }
    }

    void listUsers() {
        if (server.hasUsers()) {
            writer.println("Connected users: " + server.getUserNames());
        } else {
            writer.println("No other users connected");
        }
    }

    void sendMessage(String message) {
        writer.println(message);
    }
}
 /* ============================================================
 *
 * Roll No: 30
```

```
 *
 * File:      01-ClientReadHandler.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ========================================================== */

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

public class ClientReadHandler extends Thread {

    private BufferedReader reader;
    private final Socket socket;
    private final ChatClient client;

    public ClientReadHandler(Socket socket, ChatClient client) {
        this.socket = socket;
        this.client = client;

        try {
            reader = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
        } catch (IOException e) {
            System.out.println("Error getting input stream: " + e.getMessage());
        }
    }

    @Override
    public void run() {
        while (true) {
            try {
                String response = reader.readLine();
                System.out.println("\n" + response);

                // prints the username after displaying the server's message
                if (client.getUsername() != null) {
                    System.out.print("[" + client.getUsername() + "]: ");
                }
            } catch (IOException e) {
                System.out.println("Error reading from server: " +
e.getMessage());
                break;
            } catch (Exception e) {
                System.out.println("Error: " + e.getMessage());
                break;
            }
        }
    }
} /* ==========================================================
 *
 * Roll No: 30
```

```
 *
 * File:      01-ClientWriteHandler.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class ClientWriteHandler extends Thread {
    private PrintWriter writer;
    private final Socket socket;
    private final ChatClient client;

    public ClientWriteHandler(Socket socket, ChatClient client) {
        this.socket = socket;
        this.client = client;

        try {
            OutputStream output = socket.getOutputStream();
            writer = new PrintWriter(output, true);
        } catch (IOException e) {
            System.out.println("Error getting output stream: " +
e.getMessage());
        }
    }

    @Override
    public void run() {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter your name: ");
        String userName = sc.nextLine();
        client.setUserName(userName);
        writer.println(userName);

        String text;

        do {
            System.out.print("[" + userName + "]: ");
            text = sc.nextLine();
            writer.println(text);
        } while (!text.equals("exit"));

        try {
            socket.close();
        } catch (IOException e) {
            System.out.println("Error writing to server: " + e.getMessage());
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
```

```
        }
    }
} /* ============================================================
 *
 * Roll No: 30
 *
 * File:      02-MultiThreadedSocketServer.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

 import java.net.*;

public class MultiThreadedSocketServer {
    public static void main(String[] args) throws Exception {
        try {
            ServerSocket server = new ServerSocket(5678);
            int counter = 0;
            System.out.println("Listening on Port 5678...");
            while (true) {
                counter++;
                Socket serverClient = server.accept();
                System.out.println(">>> " + "Client No: " + counter + "
connected!");
                ServerClientThread sct = new ServerClientThread(serverClient,
counter);
                System.out.println(counter + " clients connected.");
                sct.start();
            }
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      02-ServerClientThread.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

class ServerClientThread extends Thread {
    Socket serverClient;
    int clientNo;

    ServerClientThread(Socket inSocket, int counter) {
```

```
        serverClient = inSocket;
        clientNo = counter;
    }

    public void run() {
        try {
            BufferedReader reader = new BufferedReader(new
InputStreamReader(serverClient.getInputStream()));
            PrintWriter writer = new
PrintWriter(serverClient.getOutputStream());
            String clientMessage = "", serverMessage = "";

            while (!clientMessage.equals("bye")) {
                clientMessage = reader.readLine();
                int n = Integer.parseInt(clientMessage);

                System.out.println("From Client - " + clientNo + ": Number is :
" + clientMessage);
                serverMessage = n + " is " + (n % 2 == 0 ? "Even" : "Odd");
                writer.println(serverMessage);
                writer.flush();
            }

            reader.close();
            writer.close();
            serverClient.close();

        } catch (Exception ex) {
            System.out.println(ex);
        } finally {
            System.out.println("Client - " + clientNo + " disconnected!");
        }
    }
} /* ============================================================
 *
 * Roll No: 30
 *
 * File:      02-TCPThreadClient.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class TCPThreadClient {
    public static void main(String[] args) throws Exception {
        try {
            Socket socket = new Socket("127.0.0.1", 5678);
            BufferedReader reader = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
```

```
            PrintWriter writer = new PrintWriter(socket.getOutputStream());
            BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
            String clientMessage = "", serverMessage = "";

            while (!clientMessage.equals("bye")) {
                System.out.println("Enter number :");
                clientMessage = br.readLine();
                writer.println(clientMessage);
                writer.flush();
                serverMessage = reader.readLine();
                System.out.println(serverMessage);
            }

            writer.close();
            socket.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      03-MultiThreadedSocketServer.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

 import java.net.*;

public class MultiThreadedSocketServer {
    public static void main(String[] args) throws Exception {
        try {
            DatagramSocket serverSocket = new DatagramSocket(5678);
            System.out.println("Listening on Port 5678...");
            ServerClientThread sct = new ServerClientThread(serverSocket);
            sct.start();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
} /* ============================================================
 *
 * Roll No: 30
 *
 * File:      03-ServerClientThread.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */
```

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

class ServerClientThread extends Thread {
    DatagramSocket serverClient;

    ServerClientThread(DatagramSocket inSocket) {
        serverClient = inSocket;
    }

    public void run() {
        try {
            String clientSentence = "", serverMessage = "";
            while (!clientSentence.equals("bye")) {
                byte[] recieveData = new byte[1024];
                byte[] sendData = new byte[1024];
                DatagramPacket receivePacket = new DatagramPacket(recieveData,
recieveData.length);
                serverClient.receive(receivePacket);
                InetAddress ipAddress = receivePacket.getAddress();
                int port = receivePacket.getPort();
                clientSentence = new String(receivePacket.getData()).trim();
                int n = Integer.parseInt(clientSentence);
                serverMessage = n + " is " + (isPrime(n) ? "Prime" : "Not
Prime");
                System.out.println("From Client : " + clientSentence + "
Result : " + serverMessage);
                sendData = serverMessage.getBytes();
                DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, ipAddress, port);
                serverClient.send(sendPacket);
            }

            serverClient.close();

        } catch (Exception ex) {
            System.out.println(ex);
        }
    }

    static boolean isPrime(int n) {
        if (n <= 1)
            return false;

        for (int i = 2; i < n; i++)
            if (n % i == 0)
                return false;

        return true;
    }
```

```java
} /* ============================================================
 *
 * Roll No: 30
 *
 * File:      03-UDPThreadClient.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class UDPThreadClient {
    public static void main(String[] args) throws Exception {
        try {
            DatagramSocket clientSocket = new DatagramSocket();
            InetAddress ipAddress = InetAddress.getLocalHost();
            BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
            String clientMessage = "";

            while (!clientMessage.equals("bye")) {
                System.out.println("Enter number :");
                clientMessage = br.readLine();
                byte[] sendData = clientMessage.getBytes();
                DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, ipAddress, 5678);
                clientSocket.send(sendPacket);
                byte[] receiveData = new byte[1024];
                DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
                clientSocket.receive(receivePacket);
                String result = (new String(receivePacket.getData())).trim();
                System.out.println(result);
            }
            clientSocket.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
} /* ============================================================
 *
 * Roll No: 30
 *
 * File:      04-ChatClient.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

```java
public class ChatClient {

    private final String host;
    private final int port;
    private String username;

    public ChatClient(String host, int port) {
        this.host = host;
        this.port = port;
    }

    public void start() {
        try {
            Socket socket = new Socket(host, port);

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

            var writeHandler = new ClientWriteHandler(socket, this);
            writeHandler.start();
            (new ClientReadHandler(socket, this)).start();

        } catch (UnknownHostException e) {
            System.out.println("Server not found: " + e.getMessage());
        } catch (IOException e) {
            System.out.println("I/O Error: " + e.getMessage());
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }
    }

    void setUserName(String username) {
        this.username = username;
    }

    String getUsername() {
        return this.username;
    }

    public static void main(String[] args) {
        ChatClient client = new ChatClient("127.0.0.1", 5678);
        client.start();
    }
}

// $ java -cp bin/ ChatClient
// Connected to the server...

// No other users connected
// Enter your name: Nirav
// [Nirav]:
```

```
// New User Connected: Milind
// [Nirav]:
// [Milind]: Hey
// [Nirav]: Hello
// [Nirav]:
// New User Connected: Pradip
// [Nirav]:
// [Pradip]: Hello Everyone
// [Nirav]:
// [Pradip]: exir
// [Nirav]:
// [Pradip]: exit
// [Nirav]:
// Pradip disconnected...
// [Nirav]:
// [Milind]: Bye
// [Nirav]:
// [Milind]: exit
// [Nirav]:
// Milind disconnected...
// [Nirav]: bye
// [Nirav]: exit

// $ java -cp bin/ ChatClient
// Connected to the server...

// Connected users: [Nirav, Milind]
// Enter your name: Pradip
// [Pradip]: Hello Everyone
// [Pradip]: exir
// [Pradip]: exit

// $ java -cp bin/ ChatClient
// Connected to the server...

// // Connected users: [Nirav]
// // Enter your name: Milind
// // [Milind]: Hey
// // [Milind]:
// // [Nirav]: Hello
// // [Milind]:
// // New User Connected: Pradip
// // [Milind]:
// // [Pradip]: Hello Everyone
// // [Milind]:
// // [Pradip]: exir
// // [Milind]:
// // [Pradip]: exit
// // [Milind]:
// // Pradip disconnected...
// // [Milind]: Bye
```

```java
// // [Milind]: exit /*
============================================================
 *
 * Roll No: 30
 *
 * File:      04-ChatServer.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class ChatServer {
    private final Map<String, ClientHandler> users = new HashMap<>();
    private final int SERVER_PORT;

    public ChatServer(int port) {
        this.SERVER_PORT = port;
    }

    public void start() {
        try (ServerSocket serverSocket = new ServerSocket(SERVER_PORT)) {

            System.out.println("----- Chat Server -----");
            System.out.println("Listening on port " + SERVER_PORT + "...");

            while (true) {
                Socket socket = serverSocket.accept();
                System.out.println("New User Connected...");
                ClientHandler newUser = new ClientHandler(socket, this);
                newUser.start();
            }
        } catch (IOException e) {
            System.out.println("Error in the server: " + e.getMessage());
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }
    }

    public static void main(String[] args) {
        int port;
        try {
            port = 5678;
            ChatServer server = new ChatServer(port);
            server.start();
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
```

```
    void sendToEveryone(String message, ClientHandler notToUser) {
        for (Map.Entry<String, ClientHandler> aUser : users.entrySet()) {
            if (aUser.getValue() != notToUser) {
                aUser.getValue().sendMessage(message);
            }
        }
    }

    void sendTo(String message, String username) {
        var user = getUser(username);
        if (user != null) {
            user.sendMessage(message);
        }
    }

    private ClientHandler getUser(String username) {
        for (Map.Entry<String, ClientHandler> aUser : users.entrySet()) {
            if (aUser.getKey().equals(username)) {
                return aUser.getValue();
            }
        }
        return null;
    }

    void addUser(String userName, ClientHandler userThread) {
        users.put(userName, userThread);
    }

    void removeUser(String userName) {
        users.remove(userName);
    }

    Set<String> getUserNames() {
        return this.users.keySet();
    }

    boolean hasUsers() {
        return !this.users.isEmpty();
    }
}

// $ javac *.java && java ChatServer
// ----- Chat Server -----
// Listening on port 5678...
// New User Connected...
// 1 clients connected!
// New User Connected...
// 2 clients connected!
// New User Connected...
// 3 clients connected!
```

```
// $ java ChatClient
// Connected to the server...

// No other users connected
// Enter your name: nirav

// 1 clients connected!
// [nirav]: [nirav]:
// New User Connected: milind
// [nirav]:
// 2 clients connected!
// [nirav]:
// New User Connected: pradip
// [nirav]:
// 3 clients connected!
// [nirav]:
// [milind]: bye
// [nirav]:
// [milind]: exit
// [nirav]:
// milind disconnected...
// [nirav]:
// [pradip]: exit
// [nirav]:
// pradip disconnected...
// [nirav]: exit

// $ java ChatClient
// Connected to the server...

// Connected users: [nirav]
// Enter your name: milind

// 2 clients connected!
// [milind]: bye
// [milind]:
// [milind]: exit
// [milind]:
// milind disconnected...
// [milind]: exit

// $ java ChatClient
// Connected to the server...

// Connected users: [nirav, milind]
// Enter your name: pradip

// 3 clients connected!
// [pradip]: [pradip]:
// [milind]: bye
```

```
// [pradip]:
// [milind]: exit
// [pradip]:
// milind disconnected...
// [pradip]: exit /* ============================================================
 *
 * Roll No: 30
 *
 * File:      04-ClientHandler.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class ClientHandler extends Thread {
    private final Socket socket;
    private final ChatServer server;
    private PrintWriter writer;
    public String username;

    public ClientHandler(Socket socket, ChatServer server) {
        this.socket = socket;
        this.server = server;
    }

    @Override
    public void run() {
        try {
            BufferedReader reader = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
            writer = new PrintWriter(socket.getOutputStream(), true);

            listUsers();

            username = reader.readLine();
            server.addUser(username, this);

            String serverMessage = "New User Connected: " + username;
            server.sendToEveryone(serverMessage, this);
            server.sendToEveryone(server.getUserNames().size() + " clients
connected!", null);
            System.out.println(server.getUserNames().size() + " clients
connected!");

            String clientMessage;

            do {
                clientMessage = reader.readLine().trim();
                serverMessage = "[" + username + "]: " + clientMessage;
```

```
                if (clientMessage.startsWith("@")) {
                    int indexOfSpace = clientMessage.indexOf(' ');
                    if (indexOfSpace < 2) {
                        server.sendToEveryone(serverMessage, this);
                    } else {
                        System.out.println(clientMessage.substring(1,
indexOfSpace));
                        server.sendTo(serverMessage, clientMessage.substring(1,
indexOfSpace));
                    }
                } else {
                    server.sendToEveryone(serverMessage, this);
                }
            } while (!clientMessage.equals("exit"));
        } catch (IOException e) {
            System.out.println("Error in ClientHandler: " + e.getMessage());
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        } finally {
            try {
                server.removeUser(username);
                socket.close();
                String serverMessage = username + " disconnected...";
                server.sendToEveryone(serverMessage, this);
            } catch (IOException e) {
                System.out.println("Error: " + e.getMessage());
            }
        }
    }

    void listUsers() {
        if (server.hasUsers()) {
            writer.println("Connected users: " + server.getUserNames());
        } else {
            writer.println("No other users connected");
        }
    }

    void sendMessage(String message) {
        writer.println(message);
    }
}
 /* ============================================================
 *
 * Roll No: 30
 *
 * File:      04-ClientReadHandler.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */
```

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

public class ClientReadHandler extends Thread {

    private BufferedReader reader;
    private final Socket socket;
    private final ChatClient client;

    public ClientReadHandler(Socket socket, ChatClient client) {
        this.socket = socket;
        this.client = client;

        try {
            reader = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
        } catch (IOException e) {
            System.out.println("Error getting input stream: " + e.getMessage());
        }
    }

    @Override
    public void run() {
        while (true) {
            try {
                String response = reader.readLine();
                System.out.println("\n" + response);

                // prints the username after displaying the server's message
                if (client.getUsername() != null) {
                    System.out.print("[" + client.getUsername() + "]: ");
                }
            } catch (IOException e) {
                System.out.println("Error reading from server: " +
e.getMessage());
                break;
            } catch (Exception e) {
                System.out.println("Error: " + e.getMessage());
                break;
            }
        }
    }
} /* ============================================================
 *
 * Roll No: 30
 *
 * File:      04-ClientWriteHandler.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */
```

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

public class ClientWriteHandler extends Thread {
    private PrintWriter writer;
    private final Socket socket;
    private final ChatClient client;

    public ClientWriteHandler(Socket socket, ChatClient client) {
        this.socket = socket;
        this.client = client;

        try {
            OutputStream output = socket.getOutputStream();
            writer = new PrintWriter(output, true);
        } catch (IOException e) {
            System.out.println("Error getting output stream: " +
e.getMessage());
        }
    }

    @Override
    public void run() {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter your name: ");
        String userName = sc.nextLine();
        client.setUserName(userName);
        writer.println(userName);

        String text;

        do {
            System.out.print("[" + userName + "]: ");
            text = sc.nextLine();
            writer.println(text);
        } while (!text.equals("exit"));

        try {
            socket.close();
        } catch (IOException e) {
            System.out.println("Error writing to server: " + e.getMessage());
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
} /* ============================================================
 *
 * Roll No: 30
```

```
 *
 * File:      05-Client.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ========================================================== */

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

public class Client {
    private final String FILENAME = null;
    Connect c = new Connect();
    Socket socket;
    BufferedReader read;
    PrintWriter output;

    public void startClient() throws UnknownHostException, IOException {
        socket = new Socket(c.gethostName(), c.getPort());

        output = new PrintWriter(new
OutputStreamWriter(socket.getOutputStream()));
        BufferedReader in = new BufferedReader(new
InputStreamReader(System.in));

        System.out.println("Enter Username : ");
        String username = in.readLine();

        output.println(username);

        System.out.println("Enter Password : ");
        String password = in.readLine();

        output.println(password);
        output.flush();
        read = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
        String response = read.readLine();
        System.out.println(response);
    }

    public void fileInfo() {

    }

    public static void main(String args[]) {
        Client client = new Client();
        try {
            client.startClient();
        } catch (UnknownHostException e) {
            e.printStackTrace();
        } catch (IOException e) {
```

```
            e.printStackTrace();
        }
    }
} /* ============================================================
 *
 * Roll No: 30
 *
 * File:      05-Connect.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

 public class Connect {
    private String USERNAME = "java";
    private String PASSWORD = "java";
    private int PORT = 9090;
    private String HOSTNAME = "localhost";

    public String getUsername() {
        return this.USERNAME;
    }

    public String getPassword() {

        return this.PASSWORD;
    }

    public int getPort() {
        return this.PORT;
    }

    public String gethostName() {
        return this.HOSTNAME;
    }
} /* ============================================================
 *
 * Roll No: 30
 *
 * File:      05-Server.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class Server {
    private int currentTot;
    ServerSocket serversocket;
    Socket client;
```

```
    int bytesRead;
    Connect c = new Connect();
    BufferedReader input;
    PrintWriter output;

    public void start() throws IOException {
        System.out.println("Connection Starting on port:" + c.getPort());
        serversocket = new ServerSocket(c.getPort());
        client = serversocket.accept();

        System.out.println("Waiting for connection from client");

        try {
            logInfo();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void logInfo() throws Exception {
        input = new BufferedReader(new
InputStreamReader(client.getInputStream()));

        String username = input.readLine();
        System.out.println("Username : " + username);
        String password = input.readLine();
        System.out.println("Password : " + password);

        output = new PrintWriter(new
OutputStreamWriter(client.getOutputStream()));

        if (username.equals(c.getUsername()) &&
password.equals(c.getPassword())) {
            output.println("Welcome, " + username);
        } else {
            output.println("Login Failed");
        }
        output.flush();
        output.close();
    }

    public static void main(String[] args) {
        Server server = new Server();
        try {
            server.start();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
} /* ============================================================
 *
```

```
 * Roll No: 30
 *
 * File:      06-Client.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ========================================================== */

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

class Client {
    public static void main(String ar[]) throws Exception {
        Socket s = new Socket("localhost", 1234);
        PrintWriter p = new PrintWriter(s.getOutputStream());
        BufferedReader in = new BufferedReader(new
InputStreamReader(s.getInputStream()));
        BufferedReader ink = new BufferedReader(new
InputStreamReader(System.in));
        System.out.println("How many numbers to sort? ");
        int num = Integer.parseInt(ink.readLine());
        p.println(num);
        p.flush();
        System.out.println("Enter " + num + " numbers to sort :");
        String sarr[] = new String[num];
        for (int i = 0; i < num; i++) {
            sarr[i] = ink.readLine();
            p.println(sarr[i]);
            p.flush();
        }
        String res;
        System.out.println("\nSorted array::\n");
        while ((res = in.readLine()) != null) {
            System.out.println(res);
        }
        s.close();
    }
}
 /* ==========================================================
 *
 * Roll No: 30
 *
 * File:      06-Server.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ========================================================== */

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

class Server {
    public static void main(String ar[]) throws Exception {
```

```
        ServerSocket s1 = new ServerSocket(1234);
        System.out.println("Server Started");
        Socket s = s1.accept();
        PrintWriter p = new PrintWriter(s.getOutputStream());
        BufferedReader in = new BufferedReader(new
InputStreamReader(s.getInputStream()));
        String num = in.readLine();
        int n = Integer.parseInt(num);
        System.out.println("Client want to sort " + n + " numbers");
        String sarr[] = new String[n];
        int arr[] = new int[n];
        int swap, c, d;
        System.out.println("received numbers::\n");
        for (int i = 0; i < n; i++) {
            sarr[i] = in.readLine();
            arr[i] = Integer.parseInt(sarr[i]);
            System.out.println(arr[i]);

        }

        for (c = 0; c < (n - 1); c++) {
            for (d = 0; d < n - c - 1; d++) {
                if (arr[d] > arr[d + 1]) {
                    swap = arr[d];
                    arr[d] = arr[d + 1];
                    arr[d + 1] = swap;
                }
            }
        }

        System.out.println("\nSorted list of numbers");
        String sendarr = new String();
        for (c = 0; c < n; c++) {
            sendarr += arr[c];
            sendarr += " ";
        }
        System.out.println(sendarr);
        p.println(sendarr);
        p.flush();
        s.close();

    }
} /* ============================================================
 *
 * Roll No: 30
 *
 * File:      07-Client2.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */
```

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

public class Client2 {
    private Socket s = null;
    private DataInputStream dis = null;
    private DataOutputStream dos = null;

    public static void main(String args[]) throws IOException {
        InetAddress ip = InetAddress.getLocalHost();
        Client2 cr = new Client2(ip, 1234);
    }

    public Client2(InetAddress ip, int port) throws IOException {

        s = new Socket(ip, port);
        dis = new DataInputStream(s.getInputStream());
        dos = new DataOutputStream(s.getOutputStream());

        while (true) {
            Scanner sc = new Scanner(System.in);
            int i, l, nob, sum = 0, chk_sum;

            l = dis.readInt();
            int c_data[] = new int[l];
            int data[] = new int[l];

            System.out.println("Data received (alond with checksum) is");

            for (i = 0; i < data.length; i++) {
                data[i] = dis.readInt();
                System.out.println(data[i]);

                nob = (int) (Math.floor(Math.log(data[i]) / Math.log(2))) + 1;
                c_data[i] = ((1 << nob) - 1) ^ data[i];

                sum += c_data[i];
            }
            System.out.println("Sum(in ones complement) is : " + sum);

            nob = (int) (Math.floor(Math.log(sum) / Math.log(2))) + 1;
            sum = ((1 << nob) - 1) ^ sum;
            System.out.println("Calculated Checksum is : " + sum);

            if (sum == 0) {
                dos.writeUTF("success");
                break;
            } else {
                dos.writeUTF("failure");
                break;
```

```
            }
        }

        dis.close();
        dos.close();
        s.close();
    }
} /* ============================================================
 *
 * Roll No: 30
 *
 * File:      07-Client.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class Client {
    private int MAX = 100;
    private Socket socket = null;
    private ServerSocket servsock = null;
    private DataInputStream dis = null;
    private DataOutputStream dos = null;

    public static void main(String args[]) throws IOException {
        Client cs = new Client(1234);
    }

    public Client(int port) throws IOException {
        servsock = new ServerSocket(port);
        socket = servsock.accept();

        dis = new DataInputStream(socket.getInputStream());
        dos = new DataOutputStream(socket.getOutputStream());

        while (true) {
            int i, l, sum = 0, nob;
            Scanner sc = new Scanner(System.in);
            System.out.println("Enter data length");
            l = sc.nextInt();
            int data[] = new int[MAX];
            int c_data[] = new int[MAX];

            System.out.println("Enter data to send");

            for (i = 0; i < l; i++) {
                data[i] = sc.nextInt();
                nob = (int) (Math.floor(Math.log(data[i]) / Math.log(2))) + 1;
```

```
                c_data[i] = ((1 << nob) - 1) ^ data[i];
                sum += c_data[i];
            }
            data[i] = sum;
            l += 1;

            System.out.println("Checksum Calculated is : " + sum);
            System.out.println("Data being sent along with Checkum.....");
            dos.writeInt(l);
            for (int j = 0; j < l; j++)
                dos.writeInt(data[j]);

            if (dis.readUTF().equals("success")) {
                System.out.println("Thanks for the feedback!! Message received
Successfully!");
                break;
            }

            else if (dis.readUTF().equals("failure")) {
                System.out.println("Message was not received successfully!");
                break;
            }
        }

        dis.close();
        dos.close();
        socket.close();
    }
} /* ============================================================
 *
 * Roll No: 30
 *
 * File:      10-SAWClient.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class SAWClient {
    public static void main(String args[]) {
        int p = 9000, i, q = 8000;
        String h = "localhost";
        try {
            Scanner scanner = new Scanner(System.in);
            System.out.print("Enter number of frames : ");
            int number = scanner.nextInt();
            if (number == 0) {
                System.out.println("No frame is sent");
```

```
            } else {
                Socket s2;
                s2 = new Socket(h, q);
                DataOutputStream d1 = new
DataOutputStream(s2.getOutputStream());
                d1.write(number);
                s2.close();
            }
            String str1;
            for (i = 0; i < number; i++) {
                System.out.print("Enter message : ");
                String name = scanner.next();
                System.out.println("Frame " + i + " is sent");
                Socket s1;
                s1 = new Socket(h, p + i);
                DataOutputStream d = new DataOutputStream(s1.getOutputStream());
                d.writeUTF(name);
                DataInputStream dd = new DataInputStream(s1.getInputStream());
                Integer sss1 = dd.read();
                System.out.println("Ack for :" + sss1 + " is  received");
                s1.close();
            }
        } catch (Exception ex) {
            System.out.println("ERROR :" + ex);
        }
    }
} /* ============================================================
 *
 * Roll No: 30
 *
 * File:      10-SAWServer.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

public class SAWServer {
    public static void main(String args[]) {
        String h = "Serverhost";
        int q = 5000;
        int i;
        try {
            ServerSocket ss2;
            ss2 = new ServerSocket(8000);
            Socket s1 = ss2.accept();
            DataInputStream dd1 = new DataInputStream(s1.getInputStream());
            Integer i1 = dd1.read();
            for (i = 0; i < i1; i++) {
```

```
                ServerSocket ss1;
                ss1 = new ServerSocket(9000 + i);
                Socket s = ss1.accept();
                DataInputStream dd = new DataInputStream(s.getInputStream());
                String sss1 = dd.readUTF();
                System.out.println(sss1);
                System.out.println("Frame " + i + " received");
                DataOutputStream d1 = new DataOutputStream(s.getOutputStream());
                d1.write(i);
                System.out.println("ACK sent for " + i);
                ss1.close();
                ss2.close();
            }
        } catch (Exception ex) {
            System.out.println("Error" + ex);
        }
    }
} /* ============================================================
 *
 * Roll No: 30
 *
 * File:      10-SWClient.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

 import java.net.*;
import java.io.*;
import java.rmi.*;

public class SWClient {
    public static void main(String a[]) throws Exception {
        ServerSocket ser = new ServerSocket(10);
        Socket s = ser.accept();
        BufferedReader in = new BufferedReader(new
InputStreamReader(System.in));
        BufferedReader in1 = new BufferedReader(new
InputStreamReader(s.getInputStream()));
        // DataInputStream in1 = new DataInputStream(s.getInputStream());
        String sbuff[] = new String[8];
        PrintStream p;
        int sptr = 0, sws = 8, nf, ano, i;
        String ch;
        do {
            p = new PrintStream(s.getOutputStream());
            System.out.print("Enter the no. of frames : ");
            nf = Integer.parseInt(in.readLine());
            p.println(nf);
            if (nf <= sws - 1) {

                System.out.println("Enter " + nf + " Messages to be send\n");
```

```
                for (i = 1; i <= nf; i++) {
                    sbuff[sptr] = in.readLine();
                    p.println(sbuff[sptr]);
                    sptr = ++sptr % 8;
                }
                sws -= nf;
                System.out.print("Acknowledgment received");
                ano = Integer.parseInt(in1.readLine());
                System.out.println(" for " + ano + " frames");
                sws += nf;
            } else {
                System.out.println("The no. of frames exceeds window size");
                break;
            }
            System.out.print("\nDo you wants to send some more frames : ");
            ch = in.readLine();
            p.println(ch);
        } while (ch.equals("yes"));
        s.close();
    }
} /* ============================================================
 *
 * Roll No: 30
 *
 * File:      10-SWServer.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

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

class SWServer {
    public static void main(String a[]) throws Exception {
        Socket s = new Socket(InetAddress.getLocalHost(), 10);
        BufferedReader in = new BufferedReader(new
InputStreamReader(s.getInputStream()));
        // DataInputStream in = new DataInputStream(s.getInputStream());
        PrintStream p = new PrintStream(s.getOutputStream());
        int i = 0, rptr = -1, nf, rws = 8;
        String rbuf[] = new String[8];
        String ch;
        System.out.println();
        do {
            nf = Integer.parseInt(in.readLine());
            if (nf <= rws - 1) {
                for (i = 1; i <= nf; i++) {
                    rptr = ++rptr % 8;
                    rbuf[rptr] = in.readLine();
                    System.out.println("The received Frame " + rptr + " is : " +
rbuf[rptr]);
```

```
                }
                rws -= nf;
                System.out.println("\nAcknowledgment sent\n");
                p.println(rptr + 1);
                rws += nf;
            } else
                break;
            ch = in.readLine();
        } while (ch.equals("yes"));
    }
} /* ============================================================
 *
 * Roll No: 30
 *
 * File:      11-Dijkstra.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

 public class Dijkstra {

    public static void main(String[] args) {
        int graph[][] = new int[][] { { 0, 0, 1, 2, 0, 0, 0 }, { 0, 0, 2, 0, 0,
3, 0 }, { 1, 2, 0, 1, 3, 0, 0 },
                { 2, 0, 1, 0, 0, 0, 1 }, { 0, 0, 3, 0, 0, 2, 0 }, { 0, 3, 0, 0,
2, 0, 1 }, { 0, 0, 0, 1, 0, 1, 0 } };
        dijkstra(graph, 0);
    }

    public static void dijkstra(int[][] graph, int source) {
        int count = graph.length;
        boolean[] visitedVertex = new boolean[count];
        int[] distance = new int[count];
        for (int i = 0; i < count; i++) {
            visitedVertex[i] = false;
            distance[i] = Integer.MAX_VALUE;
        }

        distance[source] = 0;
        for (int i = 0; i < count; i++) {

            int u = findMinDistance(distance, visitedVertex);
            visitedVertex[u] = true;

            for (int v = 0; v < count; v++) {
                if (!visitedVertex[v] && graph[u][v] != 0 && (distance[u] +
graph[u][v] < distance[v])) {
                    distance[v] = distance[u] + graph[u][v];
                }
            }
        }
```

```
        for (int i = 0; i < distance.length; i++) {
            System.out.println(String.format("Distance from %s to %s is %s",
source, i, distance[i]));
        }

    }

    private static int findMinDistance(int[] distance, boolean[] visitedVertex)
{
        int minDistance = Integer.MAX_VALUE;
        int minDistanceVertex = -1;
        for (int i = 0; i < distance.length; i++) {
            if (!visitedVertex[i] && distance[i] < minDistance) {
                minDistance = distance[i];
                minDistanceVertex = i;
            }
        }
        return minDistanceVertex;
    }
} /* ============================================================
 *
 * Roll No: 30
 *
 * File:      12-Admin.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

 import java.io.File;
import java.io.IOException;
import java.util.HashSet;
import java.util.Scanner;

public class Admin {

    public static void main(String[] args) {

        HashSet<Integer> hashSetPorts = new HashSet<>();
        if (args.length == 0) {
            System.out.println("Please Include Directory Path");
            return;
        } else if (args.length > 1) {
            System.out.println("Incorrect Input Format.! Only one parameter is
required");
            return;
        }

        String PATH = args[0];
        File directory = new File(PATH);

        if (!directory.isDirectory()) {
```

```
            System.out.println("Directoy Path is Incorrect");
            return;
        }
        File data[] = directory.listFiles();
        int size = data.length;
        int[] ports = new int[size];
        String allNodes = "";

        System.out.println("Initilization of Port Number to " + size + "
Routers");
        Scanner scanner = new Scanner(System.in);

        for (int i = 0; i < size; i++) {
            String val = data[i].getName();
            val = val.substring(0, val.indexOf(".dat"));
            System.out.println("Enter Port No: for Router: " + val);

            boolean status = true;

            while (status) {
                try {
                    int num = Integer.parseInt(scanner.nextLine());

                    if (num <= 1024 || num >= 65536) {
                        throw new NumberFormatException();
                    }
                    if (hashSetPorts.contains(num)) {
                        throw new Exception();
                    }
                    ports[i] = num;
                    hashSetPorts.add(num);
                    status = false;
                } catch (NumberFormatException e) {
                    System.out.println("Enter a valid Port Number > 1024 && <
65536");
                    status = true;
                } catch (Exception e) {
                    System.out.println("Address is Already in Use:");
                    status = true;
                }
            }
            allNodes += " " + val + ":" + ports[i];
        }

        scanner.close();

        for (int i = 0; i < size; i++) {
            ProcessBuilder processBuilder = new ProcessBuilder("cmd.exe", "/c",
"start java MainRouter " + (i + 1)
                    + " \"" + data[i].getParent().replace("\\", "/") + "\" " +
size + allNodes);
```

```
            try {
                processBuilder.start();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        System.out.println("DV Algorithm Started");

    }

} /* ============================================================
 *
 * Roll No: 30
 *
 * File:      12-MainRouter.java
 * Copyright: By Ajinkya Rathod(ajinzrathod)
 *
 * ============================================================ */

 import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.SocketException;
import java.util.Arrays;

public class MainRouter {

    public static int router_DisplayCount = 1;
    public static double[][] router_NetworkVectors;
    public static int[] router_Ports;
    public static String[] router_Nodes;
    public static int router_id;
    public static double[] router_MyVector;
    public static String[] router_MyHopList;
    public static DatagramSocket router_Socket;
    public static File router_File;
    public static String[] router_Neighbours;

    public static void main(String args[]) {
        int total = Integer.parseInt(args[2]);
        int currentNum = Integer.parseInt(args[0]);
        String path = args[1];
        setParameters(total, args, currentNum, path);

        MainThread readThread = new MainThread("r");
        readThread.start();
```

```
        MainThread writeThread = new MainThread("w");
        writeThread.start();

        while (true)
            ;
    }

    public static void setParameters(int len, String[] args, int id, String
parent) {

        router_NetworkVectors = new double[len][len];
        router_Ports = new int[len];
        router_Nodes = new String[len];

        for (int i = 0; i < len; i++) {
            Arrays.fill(router_NetworkVectors[i], Double.MAX_VALUE);
            router_NetworkVectors[i][i] = 0.0;
            String[] temp = args[i + 3].split(":");
            router_Nodes[i] = temp[0];
            router_Ports[i] = Integer.parseInt(temp[1]);
        }

        router_id = id;
        router_MyVector = new double[len];
        router_MyHopList = new String[len];
        Arrays.fill(router_MyVector, Double.MAX_VALUE);
        router_MyVector[router_id - 1] = 0.0;
        router_File = new File(parent + "/" + router_Nodes[router_id - 1] +
".dat");
        try {
            router_Socket = new DatagramSocket(router_Ports[router_id - 1]);
        } catch (SocketException e) {

            e.printStackTrace();
        }
        System.out.println("Router " + router_Nodes[router_id - 1] + " is
Working..!");

    }

    public static void distanceAlgorithm() {

        for (int i = 0; i < router_Neighbours.length; i++) {
            int ind = indexFinder(router_Neighbours[i]);
            for (int j = 0; j < router_MyVector.length; j++) {
                if (j == router_id - 1) {
                    continue;
                } else if (i == 0) {

                    router_MyVector[j] = router_NetworkVectors[router_id - 1]
[ind] + router_NetworkVectors[ind][j];
```

```
                    router_MyHopList[j] = router_Neighbours[i];
                } else {

                    if (router_MyVector[j] > router_NetworkVectors[router_id -
1][ind]
                            + router_NetworkVectors[ind][j]) {
                        router_MyHopList[j] = router_Neighbours[i];
                        router_MyVector[j] = router_NetworkVectors[router_id -
1][ind] + router_NetworkVectors[ind][j];
                    }

                }
            }
        }

    }

    public synchronized static void updateNetworkVectors(String[] vector, int
port) {

        int index = 0;
        int ports_Length = router_Ports.length;
        while (index < ports_Length) {
            if (router_Ports[index] == port) {
                break;
            }
            index++;
        }
        if (index == ports_Length) {
            return;
        }
        for (int i = 0; i < vector.length; i++) {
            router_NetworkVectors[index][i] = Double.parseDouble(vector[i]);
        }
    }

    public static void broadCast() {

        try {
            for (int i = 0; i < router_Neighbours.length; i++) {
                String data = "";
                for (int j = 0; j < router_MyVector.length; j++) {
                    if (router_Neighbours[i].equals(router_MyHopList[j])) {
                        data = data + Double.MAX_VALUE + ":";
                    } else {
                        data += router_MyVector[j] + ":";
                    }
                }
                DatagramPacket packet = new DatagramPacket(data.getBytes(),
data.getBytes().length);
                packet.setAddress(InetAddress.getByName("localhost"));
```

```
                packet.setPort(router_Ports[indexFinder(router_Neighbours[i])]);
                router_Socket.send(packet);

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

    }

    public static void output() {

        System.out.println("> output number " + router_DisplayCount++);
        System.out.println();
        String src = router_Nodes[router_id - 1];
        for (int i = 0; i < router_MyVector.length; i++) {
            if (i != (router_id - 1)) {
                String dest = router_Nodes[i];
                if (router_MyVector[i] == Double.MAX_VALUE) {
                    System.out.println("shortest path " + src + "-" + dest + ":
" + " no route found");
                } else {
                    System.out.println("shortest path " + src + "-" + dest + " :
the next hop is " + router_MyHopList[i]
                            + " and the cost is " + router_MyVector[i]);
                }
            }
        }

    }

    public static void readData() {
        boolean status = true;
        while (status) {
            try {
                String type = "u";
                byte[] data = new byte[1024];
                int size = data.length;
                DatagramPacket packet = new DatagramPacket(data, size);
                router_Socket.receive(packet);
                int length = packet.getLength();
                String vector = new String(packet.getData(), 0, length);
                MainThread updateThread = new MainThread(type, vector,
packet.getPort());
                updateThread.start();

            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
```

```
    public static void read() {
        try {
            Arrays.fill(router_NetworkVectors[router_id - 1], Double.MAX_VALUE);
            router_NetworkVectors[router_id - 1][router_id - 1] = 0.0;
            BufferedReader br = new BufferedReader(new FileReader(router_File));
            int length = Integer.parseInt(br.readLine());
            router_Neighbours = new String[length];
            for (int i = 0; i < length; i++) {

                String[] temp = br.readLine().split(" ");
                int ind = indexFinder(temp[0]);
                router_Neighbours[i] = temp[0];

                if (router_DisplayCount == 1) {
                    router_MyHopList[ind] = temp[0];
                    router_MyVector[ind] = Double.parseDouble(temp[1]);
                } else {

                }
                router_NetworkVectors[router_id - 1][ind] =
Double.parseDouble(temp[1]);

            }
            br.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public static int indexFinder(String temp) {
        int pos = -1;
        for (int i = 0; i < router_Nodes.length; i++) {
            if (router_Nodes[i].equals(temp)) {
                pos = i;
                break;
            }
        }
        return pos;
    }

}
```