1)Echo Client and Server Programs using UDP in Java

Server Side:

```
Import java.io.*;

Import java.net.*;

Public class UDPEchoServer {

  Public static void main(String[] args) {

    Try (DatagramSocket serverSocket = new DatagramSocket(9876)) {

      Byte[] receiveData = new byte[1024];

      While (true) {

        DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);

        serverSocket.receive(receivePacket);

        String sentence = new String(receivePacket.getData(), 0, receivePacket.getLength());

        InetAddress IPAddress = receivePacket.getAddress();

        Int port = receivePacket.getPort();

        Byte[] sendData = sentence.getBytes();
```

```
                DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length, IPAddress,
port);

                serverSocket.send(sendPacket);

            }

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

}
```

Client Side:

```
Import java.io.*;

Import java.net.*;

Public class UDPEchoClient {

    Public static void main(String[] args) {

        Try (DatagramSocket clientSocket = new DatagramSocket()) {

            InetAddress IPAddress = InetAddress.getByName("localhost");
            Byte[] sendData = “Hello, UDP Server!”.getBytes();

            Byte[] receiveData = new byte[1024];

            DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length, IPAddress,
9876);

            clientSocket.send(sendPacket);

            DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);

            clientSocket.receive(receivePacket);

            String modifiedSentence = new String(receivePacket.getData(), 0, receivePacket.getLength());

            System.out.println(“Received: “ + modifiedSentence);

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

}
```

Output:

```
Received: Hello, UDP Server!
```

2. Hello Server Program using UDP in Java

```
Import java.io.*;
Import java.net.*;
Public class UDPHelloServer {
   Public static void main(String[] args) {
      Try (DatagramSocket serverSocket = new DatagramSocket(9876)) {
         Byte[] receiveData = new byte[1024];
         DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);
         serverSocket.receive(receivePacket);
         String sentence = new String(receivePacket.getData(), 0, receivePacket.getLength());
         If (sentence.equals("Hello, world!")) {
            System.out.println("Received: " + sentence);
         } else {
            System.out.println("Invalid message");
         }
      } catch (IOException e) {
         e.printStackTrace();
      }
   }
}
```

Output:

```
Received: Hello, world
```

```
Invalid message.   //If it receives a different message or no message at all
```

3. Simple Client-Server Application using UDP in Java

```
Import java.io.*;

Import java.net.*;

Public class UDPClient {

  Public static void main(String[] args) {

    Try (DatagramSocket clientSocket = new DatagramSocket()) {

      InetAddress serverAddress = InetAddress.getByName("localhost");

      Int serverPort = 9876;

      String messageToSend = "Hello, server!";

      Byte[] sendData = messageToSend.getBytes();

      DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length,
serverAddress, serverPort);

      clientSocket.send(sendPacket);
```

```
            byte[] receiveData = new byte[1024];

            DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);

            clientSocket.receive(receivePacket);

            String modifiedMessage = new String(receivePacket.getData(), 0, receivePacket.getLength());

            System.out.println("Received from server: " + modifiedMessage);

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

}
```

Output:

```
Received from server: HELLO, SERVER!
```

4. HTTP Web Client Program using TCP Sockets in Java

```
Import java.io.*;

Import java.net.*;

Public class HTTPClient {

  Public static void main(String[] args) {

    Try {

      Socket clientSocket = new Socket(www.example.com, 80);

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

      Out.println("GET / HTTP/1.1\r\nHost: www.example.com\r\n\r\n");

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

      String inputLine;

      While ((inputLine = in.readLine()) != null) {

        System.out.println(inputLine);

      }

      clientSocket.close();

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

        }

    }

}
```

5. A) Write the use of the following network configuration commands in respective

Environment – Unix /Windows

i) Tcpdump ii) netstat iii) ifconfig / ipconfig iv) nslookup v) traceroute

i) tcpdump

- Unix: Used for packet sniffing and network analysis.

```
Sudo tcpdump -i <interface>
```

- Windows: Similar functionality with Wireshark or Microsoft Network Monitor.

ii) netstat

Unix: Displays network connections, routing tables, interface statistics, masquerade connections, etc.

```
Netstat -a
```

```

- Windows: Same command, but options might differ.

```
Netstat -a
```

iii) ifconfig / ipconfig
Unix (ifconfig): Configures and displays network interfaces.

```
Ifconfig eth0 up/down
```

- Unix (ipconfig): Displays IP configuration.

```
Ipconfig
```

- Windows: Configures and displays network interfaces.

```
Ipconfig /release /renew
```

iv) **nslookup**

- Unix: Resolves domain names to IP addresses.

```
Nslookup example.com
```

- Windows: Same command.

```
Nslookup example.com
```

v) traceroute
```

- Unix: Shows the route that packets take to reach a destination.

  ```
  Traceroute example.com
  ```

- Windows: Similar functionality with `tracert`.

  ```
  Tracert example.com
  ```

6. DNS Implementation Using UDP Sockets in Java

```
Import java.io.*;
Import java.net.*;

Public class DNSServer {
  Public static void main(String[] args) {
    Try (DatagramSocket serverSocket = new DatagramSocket(53)) {
      Byte[] receiveData = new byte[1024];
      Byte[] sendData;

      While (true) {
        DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);
        serverSocket.receive(receivePacket);
        String queryContent = new String(receivePacket.getData(), 0, receivePacket.getLength());

        sendData = ipAddress.getBytes();

        InetAddress IPAddress = receivePacket.getAddress();
        Int port = receivePacket.getPort();
```

```
                DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length, IPAddress,
port);
                serverSocket.send(sendPacket);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

7. Write a program to implement ARP protocols

```
Import java.util.*;

Public class ARPProtocol {
    Public static void main(String[] args) {
        Map<String, String> arpCache = new HashMap<>();

        // Adding entries to the ARP table (IP to MAC mappings)
        arpCache.put("192.168.0.1", "00:1A:2B:3C:4D:5E");
        arpCache.put("192.168.0.2", "A1:B2:C3:D4:E5:F6");

        // Retrieving MAC address for an IP
        String ipAddressToSearch = "192.168.0.1";
        String macAddress = arpCache.getOrDefault(ipAddressToSearch, "Unknown");
        System.out.println("MAC Address for " + ipAddressToSearch + ": " + macAddress);

        // Trying to retrieve MAC address for an unknown IP
        String unknownIpAddress = "192.168.0.3";
        String unknownMacAddress = arpCache.getOrDefault(unknownIpAddress, “Unknown”);
        System.out.println(“MAC Address for “ + unknownIpAddress + “: “ + unknownMacAddress);
    }
}
```

Output:

```
MAC Address for 192.168.0.1: 00:1A:2B:3C:4D:5E
MAC Address for 192.168.0.3: Unknown
```

8. Write a program to implement CRC algorithm for error detection technique

```
Public class CRCErrorDetection {
    Public static void main(String[] args) {
        String data = “11010011101100”; // Sample data to be checked
        String divisor = “1011”; // Divisor for CRC (4-bit divisor)

        String remainder = performCRC(data, divisor);
        System.out.println(“Remainder after CRC: “ + remainder);
    }

    Public static String performCRC(String data, String divisor) {
        StringBuilder sb = new StringBuilder(data);
        Int divisorLength = divisor.length();

        For (int i = 0; i < data.length() – (divisorLength – 1); i++) {
            If (sb.charAt(i) == ‘1’) {
                For (int j = 0; j < divisorLength; j++) {
```

```
                Sb.setCharAt(i + j, (sb.charAt(i + j) == divisor.charAt(j)) ? '0' : '1');
            }
        }
    }

    Return

Sb.substring(data.length() – (divisorLength – 1));
  }
}
```

Output:

```
Remainder after CRC: 1001
```

11 Write a socket program for simulation of echo server. The client and server pair runs a simple TCP

Sockets program an echo server that allows one or more client to connect the server

Server:

```
Import java.io.*;
Import java.net.*;

Public class EchoServer {
    Public static void main(String[] args) {
        Try (ServerSocket s = new ServerSocket(7777)) {
            While (true) new Thread(() -> {
                Try (Socket c = s.accept();
                     BufferedReader r = new BufferedReader(new InputStreamReader(c.getInputStream()));
                     PrintWriter w = new PrintWriter(c.getOutputStream(), true)) {
                    String m;
                    While ((m = r.readLine()) != null) w.println("Server Echo: " + m);
                } catch (IOException e) { e.printStackTrace(); }
            }).start();
        } catch (IOException e) { e.printStackTrace(); }
    }
}
```

Client:

```
Import java.io.*;
Import java.net.*;

Public class EchoClient {
    Public static void main(String[] args) {
        Try (Socket s = new Socket("localhost", 7777);
             BufferedReader u = new BufferedReader(new InputStreamReader(System.in));
             BufferedReader i = new BufferedReader(new InputStreamReader(s.getInputStream()));
             PrintWriter o = new PrintWriter(s.getOutputStream(), true)) {

            String m;
            While ((m = u.readLine()) != null) {
                o.println(m);
                System.out.println("Server Response: " + i.readLine());
            }
        } catch (IOException e) { e.printStackTrace(); }
    }
}
```

Output:

```
Hello Server!
Server Response: Server Echo: Hello Server!
How are you?
Server Response: Server Echo: How are you?
```

12. Write a program a client-server application for CHAT using TCP in java

Client:

```
Import java.io.*;
Import java.net.*;

Public class ChatClient {
    Public static void main(String[] args) {
        Try (Socket socket = new Socket("localhost", 9876);
             BufferedReader userInput = new BufferedReader(new InputStreamReader(System.in));
             PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
             BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream())))
{

            System.out.println("Connected to server. Start typing your messages:");

            String userMessage;
        While ((userMessage = userInput.readLine()) != null) {
          Out.println(userMessage);
          System.out.println("Server: " + in.readLine());
        }
      } catch (IOException e) {
        e.printStackTrace();
      }
    }
}
```

Server:

```
Import java.io.*;
Import java.net.*;

Public class ChatServer {
  Public static void main(String[] args) {
      Try (ServerSocket serverSocket = new ServerSocket(9876);
           Socket clientSocket = serverSocket.accept();
           PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);
           BufferedReader in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()))) {

        System.out.println("Client connected. Start typing your messages:");

        String clientMessage;
        While ((clientMessage = in.readLine()) != null) {
          System.out.println("Client: " + clientMessage); // Display client's message

          BufferedReader userInput = new BufferedReader(new InputStreamReader(System.in));
```

```
                String serverResponse = userInput.readLine(); // Type server’s response
                Out.println(serverResponse); // Send response to client
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Output:

Client:

```
Connected to server. Start typing your messages:
Hello Server!
Server: Hello Client!
How are you?
Server: I’m good, thank you!
```

Server:

```
Client connected. Start typing your messages:
Hello Server!
Client: Hello Server!
How are you?
Client: How are you?
I’m good, thank you!
```

13 Write a program to perform File Transfer in Client & Server Using TCP. The server sends a reply to the

User with the files. The user specified files needs to be downloading

Client

```
Import java.io.*;
Import java.net.*;
Public class FileTransferServer {
  Public static void main(String[] args) {
    Try (ServerSocket serverSocket = new ServerSocket(9876);
       Socket clientSocket = serverSocket.accept();
       DataInputStream dis = new DataInputStream(clientSocket.getInputStream());
       DataOutputStream dos = new DataOutputStream(clientSocket.getOutputStream())) {

      String requestedFile = dis.readUTF();
      File file = new File(requestedFile);

      If (file.exists()) {
        Byte[] buffer = new byte[1024];
        Int bytesRead;
        BufferedInputStream bis = new BufferedInputStream(new FileInputStream(file));

        While ((bytesRead = bis.read(buffer)) != -1) {
          Dos.write(buffer, 0, bytesRead);
        }
        Bis.close();
      } else {
        Dos.writeUTF("File not found");
```

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

Server:

```
Import java.io.*;
Import java.net.*;

Public class FileTransferClient {
    Public static void main(String[] args) {
        Try (Socket socket = new Socket("localhost", 9876);
             BufferedReader userInput = new BufferedReader(new InputStreamReader(System.in));
             BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
             OutputStream out = socket.getOutputStream()) {

            System.out.print("Enter file name to download: ");
            String requestedFile = userInput.readLine();
            Out.write(requestedFile.getBytes());

            Byte[] buffer = new byte[1024];
            Int bytesRead;
            FileOutputStream fileOutputStream = new FileOutputStream("received_file.txt");

            While ((bytesRead = in.read(buffer)) != -1) {
                fileOutputStream.write(buffer, 0, bytesRead);
            }
            fileOutputStream.close();

            System.out.println("File received as 'received_file.txt'");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Output:

```
File received as 'received_file.txt' // if file transfer successful

'File not found' // if req file doesn't exist
```

17 Simulation of Distance Vector Routing algorithm for finding the shortest-path routing

```
Import java.util.Arrays;

Public class DistanceVectorRouting {
    Static final int INF = 9999; // Infinity value

    Public static void main(String[] args) {
        Int[][] matrix = {
            {0, 2, INF, 1},
          {2, 0, 5, INF},
          {INF, 5, 0, 1},
          {1, INF, 1, 0}
        };

        distanceVectorRouting(matrix);
    }

    Static void distanceVectorRouting(int[][] matrix) {
        Int size = matrix.length;

        // Update distance matrix based on shortest paths
        For (int k = 0; k < size; k++) {
          For (int i = 0; i < size; i++) {
            For (int j = 0; j < size; j++) {
              Matrix[i][j] = Math.min(matrix[i][j], matrix[i][k] + matrix[k][j]);
            }
          }
        }

        // Print the shortest paths
        For (int i = 0; i < size; i++) {
          System.out.println(Arrays.toString(matrix[i]));
        }
    }
}
```

18 Write a code simulating ARP protocols for client and server in java program.

```
Import java.util.HashMap;

Class ARPServer {
   Private static HashMap<String, String> arpTable = new HashMap<>();

   Public static void main(String[] args) {
      arpTable.put(“192.168.1.2”, “AA:BB:CC:DD:EE:FF”); // Simulated ARP table

      String ipAddressToResolve = “192.168.1.2”; // IP to resolve
      String macAddress = resolveIP(ipAddressToResolve);
      System.out.println(“Resolved MAC address for “ + ipAddressToResolve + “: “ + macAddress);
   }

   Private static String resolveIP(String ipAddress) {
      Return arpTable.getOrDefault(ipAddress, “MAC address not found”);
   }
}

Class ARPClient {
   Public static void main(String[] args) {
      String ipAddress = “192.168.1.2”; // IP to request

      // Simulate ARP request from client
      System.out.println(“ARP Request: Who has “ + ipAddress + “?”);
      // Simulated ARP response from server
      System.out.println(“ARP Response: “ + ipAddress + “ is at AA:BB:CC:DD:EE:FF”);
   }
}
```

Output:

ARPServer:

```
Resolved MAC address for 192.168.1.2: AA:BB:CC:DD:EE:FF
```

ARPClient:

```
ARP Request: Who has 192.168.1.2?
ARP Response: 192.168.1.2 is at AA:BB:CC:DD:EE:FF
```

19 Implement and check the error detection/error correction techniques in networks and identify the errors

```
Public class ErrorDetection {
  Public static void main(String[] args) {
    String data = “1010101010”; // Original data
    String receivedData = introduceError(data); // Simulating an error

    Boolean errorDetected = verifyChecksum(receivedData); // Check for error

    If (errorDetected) {
      System.out.println(“Error Detected: true”);
    } else {
      System.out.println(“No Error Detected: false”);
    }
  }

  // Simulate an error by flipping a bit
  Private static String introduceError(String data) {
        Int indexToFlip = 5; // Choose a bit to flip
        Char[] charArray = data.toCharArray();
        charArray[indexToFlip] = (char) ('1' – charArray[indexToFlip]);
        return new String(charArray);
    }

    // Simple parity check for error detection
    Private static boolean verifyChecksum(String receivedData) {
        Int count = receivedData.chars().map(c -> c – '0').sum();
        Return count % 2 != 0; // Parity check for error detection
    }
}
```

Output:

```
Error Detected: true
```

20 a) Write a program and perform File Transfer in Client & Server Using TCP in java

Server side:

```
Import java.io.*;
Import java.net.*;

Public class TCPServer {
    Public static void main(String[] args) {
        Try (ServerSocket serverSocket = new ServerSocket(9876);
             Socket clientSocket = serverSocket.accept();
```

```
            InputStream inputStream = clientSocket.getInputStream();
            FileOutputStream fileOutputStream = new FileOutputStream(“received_file.txt”)) {


           inputStream.transferTo(fileOutputStream);
           System.out.println(“File transfer completed: File received and saved as ‘received_file.txt’”);
        } catch (IOException e) {
           e.printStackTrace();
        }
    }
}
```

Client side:

```
Import java.io.*;
Import java.net.*;

Public class TCPClient {
    Public static void main(String[] args) {
        Try (Socket socket = new Socket(“localhost”, 9876);
            FileInputStream fileInputStream = new FileInputStream(“file_to_send.txt”);
            OutputStream outputStream = socket.getOutputStream()) {


           fileInputStream.transferTo(outputStream);
           System.out.println(“File transfer completed: File sent to server”);
        } catch (IOException e) {
           e.printStackTrace();
        }
    }
}
```

Output:

Server side:

```
File transfer completed: File received and saved as ‘received_file.txt’
```

Client side:

```
File transfer completed: File sent to server
```

14 Write a program for Simulation of DNS using UDP sockets. The DNS have different domain with Corresponding IP address. Identify the error message to the user when domain is not resolved

```
Import java.io.*;
Import java.net.*;

Public class DNSServer {
   Public static void main(String[] args) {
      Try (DatagramSocket serverSocket = new DatagramSocket(53)) {
         System.out.println(“DNS Server is running…”);
         While (true) {
            Byte[] receiveData = new byte[1024];
            Byte[] sendData = new byte[1024];
            DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);
            serverSocket.receive(receivePacket);

            String domain = new String(receivePacket.getData(), 0, receivePacket.getLength());
            String ipAddress = resolveDNS(domain);

            sendData = ipAddress.getBytes();
            DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length,
receivePacket.getAddress(), receivePacket.getPort());
            serverSocket.send(sendPacket);
```

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

    Private static String resolveDNS(String domain) {
        // Simulated DNS resolution with predefined mappings
        Switch (domain) {
          Case “example.com”:
            Return “192.168.1.100”;
          Case “google.com”:
            Return “8.8.8.8”;
          Default:
            Return “Domain not resolved!”;
        }
    }
}
```

15 a) Write a socket program for implementation of client program in c language and server program in

Java language

Client prog in c:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
```

```
#define PORT 7777

#define SERVER_IP “127.0.0.1”

Int main() {

   Int clientSocket;

   Struct sockaddr_in serverAddr;

   Char buffer[1024];


   If ((clientSocket = socket(AF_INET, SOCK_STREAM, 0)) == -1) {

      Perror(“Socket creation failed”);

      Exit(EXIT_FAILURE);

   }

   serverAddr.sin_family = AF_INET;

   serverAddr.sin_port = htons(PORT);

   serverAddr.sin_addr.s_addr = inet_addr(SERVER_IP);

   if (connect(clientSocket, (struct sockaddr *)&serverAddr, sizeof(serverAddr)) == -1) {

      perror(“Connection failed”);

      exit(EXIT_FAILURE);

   }

   Strcpy(buffer, “Hello from C Client”);

   Send(clientSocket, buffer, strlen(buffer), 0);

   Printf(“Message sent to server\n”);

   Close(clientSocket);

   Return 0;

}
```

Server prog in java:

```
import java.io.*;

import java.net.*;


public class JavaServer {

   public static void main(String[] args) {
        try (ServerSocket serverSocket = new ServerSocket(7777)) {
            System.out.println("Java Server is running...");
            while (true) {
                Socket clientSocket = serverSocket.accept();
                BufferedReader in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));

                String message = in.readLine();
                System.out.println("Message received from C Client: " + message);
                clientSocket.close();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

9. a)Write a program to implement distance vector routing algorithm and illustrate the path taken for

Sending the packets from client to server

b) Write a socket program for simulation of CHAT server

A)

```
import java.util.*;
public class DistanceVectorRouting {
    static final int MAX = 20;
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int[][] adjacencyMatrix = new int[MAX][MAX];
        int nodes = scanner.nextInt();
        for (int i = 0; i < nodes; i++)
            for (int j = 0; j < nodes; j++)
                adjacencyMatrix[i][j] = scanner.nextInt();

        bellmanFord(adjacencyMatrix, nodes);
        scanner.close();
    }
    static void bellmanFord(int[][] adjacencyMatrix, int nodes) {
        int[] distance = new int[nodes];
        Arrays.fill(distance, Integer.MAX_VALUE);
        distance[0] = 0;
        for (int i = 0; i < nodes - 1; i++)
            for (int src = 0; src < nodes; src++)
                for (int dest = 0; dest < nodes; dest++)
                    if (adjacencyMatrix[src][dest] != 0 && distance[src] != Integer.MAX_VALUE &&
distance[src] + adjacencyMatrix[src][dest] < distance[dest])
                        distance[dest] = distance[src] + adjacencyMatrix[src][dest];
        for (int i = 0; i < nodes; ++i)
            System.out.println("From node 0 to node " + i + " : " + distance[i]);
    }
}
```

Input:

```
4
0 2 0 5
2 0 4 0
0 4 0 1
5 0 1 0
```

Output:

```
Distance Vector Routing Table:
```

```
From node 0 to node 0 : 0

From node 0 to node 1 : 2

From node 0 to node 2 : 6

From node 0 to node 3 : 3
```

B)

Server :

```
Import java.io.*;
Import java.net.*;
Import java.util.*;
Public class ChatServer {
    Private static final int PORT = 8888;
    Private static final Set<PrintWriter> clientOutputStreams = new HashSet<>();
    Public static void main(String[] args) {
        Try (ServerSocket serverSocket = new ServerSocket(PORT)) {
            System.out.println("Chat Server is running...");
            While (true) {
                Socket clientSocket = serverSocket.accept();
                PrintWriter writer = new PrintWriter(clientSocket.getOutputStream(), true);
                clientOutputStreams.add(writer);
                new Thread(() -> handleClient(clientSocket)).start();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    Private static void handleClient(Socket clientSocket) {
        Try (BufferedReader reader = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()))) {
            String message;
            While ((message = reader.readLine()) != null) {
                System.out.println("Received: " + message);
                clientOutputStreams.forEach(writer -> {
                    writer.println(message);
                    writer.flush();
                });
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Client:

```
import java.io.*;
import java.net.*;
public class ChatClient {
    public static void main(String[] args) {
        final String SERVER_ADDRESS = "127.0.0.1"; // Replace with the server's IP address
        final int SERVER_PORT = 8888;
        try {
            Socket socket = new Socket(SERVER_ADDRESS, SERVER_PORT);
            BufferedReader reader = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
            PrintWriter writer = new PrintWriter(socket.getOutputStream(), true);
            // Send messages from console input
            BufferedReader consoleReader = new BufferedReader(new InputStreamReader(System.in));
            String message;
            while ((message = consoleReader.readLine()) != null) {
                writer.println(message);
            }
            // Close resources
            consoleReader.close();
            socket.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```