**PROGRAM:**

Servercho.java

import java.io.\*;

import java.net.\*;

public class serverecho{

public static void main(String[] args) {

try (ServerSocket ssoc = new ServerSocket(3110);

Socket soc = ssoc.accept();

DataInputStream dis = new DataInputStream(soc.getInputStream());

DataOutputStream dos = new DataOutputStream(soc.getOutputStream())

)

{

String rec\_msg;

while ((rec\_msg = dis.readUTF()) != null) {

if (rec\_msg.equals("end")) {

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

break;

} else {

dos.writeUTF(rec\_msg);

}

}

} catch (IOException e) {

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

}

}

}

Clientecho.java

import java.io.\*;

import java.net.\*;

import java.util.Scanner;

public class clientecho{

public static void main(String[] args) {

try (

Socket soc = new Socket("localhost", 3110);

DataOutputStream dos = new DataOutputStream(soc.getOutputStream());

DataInputStream dis = new DataInputStream(soc.getInputStream());

Scanner n = new Scanner(System.in)

) {

System.out.println("Start echoing");

String sent\_msg;

while (true) {

sent\_msg = n.nextLine();

dos.writeUTF(sent\_msg);

if (sent\_msg.equals("end")) {

System.out.println("Disconnected");

break;

} else {

System.out.println("Echoed from server: " + dis.readUTF());

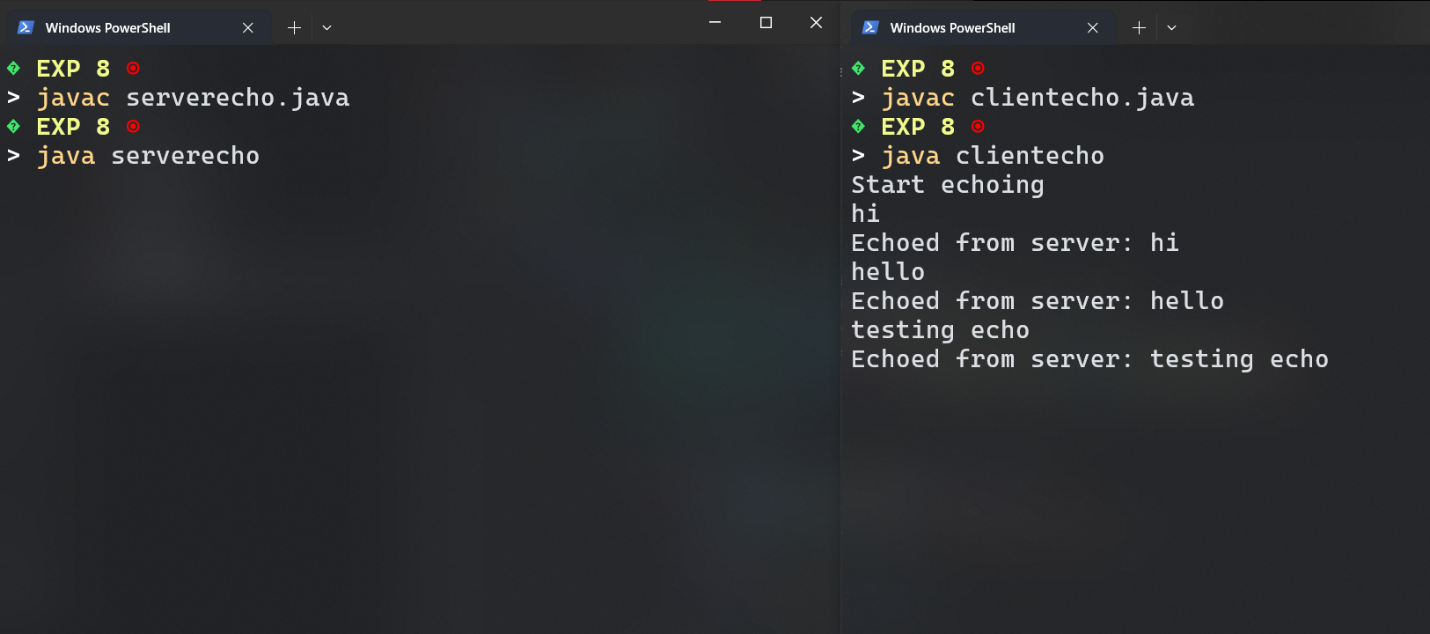
}}}

catch (IOException e) {

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

}}}

**OUTPUT:**

****

**PROGRAM:**

Serverping.java

import java.io.\*;

import java.net.\*;

public class server{

public static void main(String[] args) {

int port = 12345;

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

System.out.println("Server is running on port " + port);

while (true) {

try (Socket clientSocket = serverSocket.accept();

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

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

out.println("Ping from server");

} catch (IOException e) {

System.err.println("Error handling client connection: " + e.getMessage());

}

}

} catch (IOException e) {

System.err.println("Could not start server: " + e.getMessage());

}

}

}

Clientping.java

import java.io.\*;

import java.net.\*;

public class client{

public static void main(String[] args) {

String host = "127.0.0.1";

int port = 12345;

try (Socket socket = new Socket(host, port);

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

System.out.println("Server is reachable.");

String response = in.readLine();

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

} catch (Exception e) {

System.err.println("Server not reachable: " );}}}

**OUTPUT:**

**A screen shot of a computer

Description automatically generated**

**PROGRAM:**

Servertcp.java

import java.io.\*;

import java.net.ServerSocket;

import java.net.Socket;

public class servertcp{

private static DataOutputStream dataOutputStream = null;

private static DataInputStream dataInputStream = null;

private static final String SAVE\_DIR = "C:\\Users\\Public\\";

public static void main(String[] args) {

try (ServerSocket serverSocket = new ServerSocket(5000)) {

System.out.println("Listening on port: 5000");

Socket clientSocket = serverSocket.accept();

System.out.println(clientSocket + " connected.");

dataInputStream = new DataInputStream(clientSocket.getInputStream());

dataOutputStream = new DataOutputStream(clientSocket.getOutputStream());

while (true) {

String fileName = dataInputStream.readUTF();

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

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

break;

}

System.out.println("Receiving file: " + fileName);

receiveFile(fileName);

System.out.println("File received and saved as: " + SAVE\_DIR + fileName);

}

dataInputStream.close();

dataOutputStream.close();

clientSocket.close();

} catch (Exception e) {

e.printStackTrace();

}

} private static void receiveFile(String fileName) throws IOException {

FileOutputStream fileOutputStream = null;

try {

File file = new File(SAVE\_DIR + fileName);

fileOutputStream = new FileOutputStream(file);

long fileSize = dataInputStream.readLong();

System.out.println("Receiving file of size: " + fileSize + " bytes");

byte[] buffer = new byte[4 \* 1024];

int bytesRead;

long bytesRemaining = fileSize;

while (bytesRemaining > 0 && (bytesRead = dataInputStream.read(buffer, 0, (int) Math.min(buffer.length, bytesRemaining))) != -1) {

fileOutputStream.write(buffer, 0, bytesRead);

bytesRemaining -= bytesRead;

}

if (bytesRemaining > 0) {

throw new IOException("File incomplete. Expected " + fileSize + " bytes but received " + (fileSize - bytesRemaining) + " bytes.");

}

System.out.println("Completed receiving file: " + fileName);

} finally {

if (fileOutputStream != null) {

fileOutputStream.close();}}}}

Clienttcp.java

import java.io.\*;

import java.net.Socket;

import java.util.Scanner;

public class clienttcp{

private static DataOutputStream dataOutputStream = null;

private static DataInputStream dataInputStream = null;

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

try (Socket socket = new Socket("localhost", 5000)) {

dataInputStream = new DataInputStream(socket.getInputStream());

dataOutputStream = new DataOutputStream(socket.getOutputStream());

while (true) {

System.out.print("Enter the path of the file to send (or 'exit' to quit): ");

String filePath = scanner.nextLine();

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

dataOutputStream.writeUTF("exit");

break;

}

File file = new File(filePath);

if (file.exists() && !file.isDirectory()) {

String fileName = file.getName();

dataOutputStream.writeUTF(fileName);

sendFile(filePath);

System.out.println("File sent: " + fileName);

} else {

System.out.println("File not found or is a directory. Please enter a valid file path.");

}

}

dataInputStream.close();

dataOutputStream.close();

} catch (Exception e) {

e.printStackTrace();

} finally {

scanner.close();

}

}

private static void sendFile(String path) throws Exception {

int bytes = 0;

File file = new File(path);

FileInputStream fileInputStream = new FileInputStream(file);

dataOutputStream.writeLong(file.length());

byte[] buffer = new byte[4 \* 1024];

while ((bytes = fileInputStream.read(buffer)) != -1) {

dataOutputStream.write(buffer, 0, bytes);

dataOutputStream.flush();

}

fileInputStream.close();}}

**OUTPUT:**

