==========
NETWORKING PROGRAMS (JAVA) - SIMPLIFIED CODE AND OUTPUT
==========

NOTE: Run the Server program first, then the Client program in a separate terminal.

#1. TCP FILE TRANSFER (Simplified String)

```
// --- FileServerSimple.java ---
import java.io.*;
import java.net.*;
public class FileServerSimple {
public static void main(String[] args) throws IOException {
int port = 65432;
try (ServerSocket server = new ServerSocket(port)) {
System.out.println("Server ready. Waiting for file...");
try (Socket client = server.accept();
BufferedReader in = new BufferedReader(new InputStreamReader(client.getInputStream()))) {
String fileContent = in.readLine();
System.out.println("Client connected: " + client.getInetAddress());
if (fileContent != null) {
System.out.println("--- RECEIVED FILE DATA---");
System.out.println(fileContent);
}
}
}
}
}

// --- FileClientSimple.java ---
import java.io.*;
import java.net.*;
public class FileClientSimple {
public static void main(String[] args) throws IOException {
String host = "127.0.0.1";
int port = 65432;
try (Socket socket = new Socket(host, port);
PrintWriter out = new PrintWriter(socket.getOutputStream(), true)) {
String dataToSend = "This is the very simple file content sent over TCP.";
System.out.println("Connected. Sending data: " + dataToSend);
```

```
out.println(dataToSend);
System.out.println("Data sent successfully.");
}
}
}

// --- SAMPLE OUTPUT ---
// Server Output:
// Server ready. Waiting for file...
// Client connected: /127.0.0.1
// --- RECEIVED FILE DATA---
// This is the very simple file content sent over TCP.
//
// Client Output:
// Connected. Sending data: This is the very simple file content sent over TCP.
// Data sent successfully.

-----------------------------------------------------
#2. TCP ECHO COMMAND
-----------------------------------------------------

// --- EchoServerSimple.java ---
import java.io.*;
import java.net.*;
public class EchoServerSimple {
public static void main(String[] args) throws IOException {
int port = 65433;
try (ServerSocket server = new ServerSocket(port)) {
System.out.println("Echo Server ready. Waiting for client...");
try (Socket client = server.accept();
BufferedReader in = new BufferedReader(new InputStreamReader(client.getInputStream()));
PrintWriter out = new PrintWriter(client.getOutputStream(), true)) {
System.out.println("Client connected: " + client.getInetAddress());
String clientMsg;
while (true) {
clientMsg = in.readLine();
if (clientMsg == null || clientMsg.equalsIgnoreCase("quit")) break;
System.out.println("Received: " + clientMsg);
out.println("Echo: " + clientMsg);
}
System.out.println("Client disconnected.");
}
```

```
}
}
}

// --- EchoClientSimple.java ---
import java.io.*;
import java.net.*;
import java.util.Scanner;
public class EchoClientSimple {
public static void main(String[] args) throws IOException {
String host = "127.0.0.1";
int port = 65433;
try (Socket socket = new Socket(host, port);
PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
Scanner console = new Scanner(System.in)) {
System.out.println("Connected. Type 'quit' to exit.");
String message;
while (true) {
System.out.print("Input > ");
message = console.nextLine();
out.println(message);
if (message.equalsIgnoreCase("quit")) break;
String response = in.readLine();
System.out.println("Output < " + response);
}
}
}
}

// --- SAMPLE OUTPUT (Interactive) ---
// Server Output:
// Echo Server ready. Waiting for client...
// Client connected: /127.0.0.1
// Received: Hello
// Received: test
// Received: quit
// Client disconnected.
//
// Client Output:
// Connected. Type 'quit' to exit.
// Input > Hello
```

```
// Output < Echo: Hello
// Input > test
// Output < Echo: test
// Input > quit

-----------------------------------------------------
#3. TCP CHAT APPLICATION
-----------------------------------------------------

// --- TcpChatServerSimple.java ---
import java.io.*;
import java.net.*;
import java.util.Scanner;
public class TcpChatServerSimple {
public static void main(String[] args) throws IOException {
int port = 65434;
try (ServerSocket server = new ServerSocket(port)) {
System.out.println("Chat Server ready.");
try (Socket client = server.accept();
BufferedReader in = new BufferedReader(new InputStreamReader(client.getInputStream()));
PrintWriter out = new PrintWriter(client.getOutputStream(), true);
Scanner console = new Scanner(System.in)) {
System.out.println("Client connected: " + client.getInetAddress());
new Thread(() -> {
try {
String msg;
while ((msg = in.readLine()) != null && !msg.equalsIgnoreCase("quit")) {
System.out.println("Client > " + msg);
}
} catch (IOException e) { / Closed / }
}).start();
String serverMsg;
while (true) {
serverMsg = console.nextLine();
out.println(serverMsg);
if (serverMsg.equalsIgnoreCase("quit")) break;
}
}
}
}
}
```

```
// --- TcpChatClientSimple.java ---
import java.io.*;
import java.net.*;
import java.util.Scanner;
public class TcpChatClientSimple {
public static void main(String[] args) throws IOException {
String host = "127.0.0.1";
int port = 65434;
try (Socket socket = new Socket(host, port);
BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
Scanner console = new Scanner(System.in)) {
System.out.println("Connected. Start chatting (type 'quit'):");
new Thread(() -> {
try {
String msg;
while ((msg = in.readLine()) != null && !msg.equalsIgnoreCase("quit")) {
System.out.println("Server > " + msg);
}
} catch (IOException e) { / Closed / }
}).start();
String clientMsg;
while (true) {
clientMsg = console.nextLine();
out.println(clientMsg);
if (clientMsg.equalsIgnoreCase("quit")) break;
}
}
}
}

// --- SAMPLE OUTPUT (Interactive - Messages exchange) ---
// Server Output:
// Chat Server ready.
// Client connected: /127.0.0.1
// Client > Hi server!
// I'm fine. (Server types this line)
// Client > How are you?
// quit (Server types this line)
//
// Client Output:
// Connected. Start chatting (type 'quit'):
```

```
// Hi server! (Client types this line)
// Server > Hello client!
// How are you? (Client types this line)
// Server > I'm fine.
// Server > quit (Received from Server)

------------------------------------------------------
#4. UDP CHAT APPLICATION
------------------------------------------------------

// --- UdpChatServerSimple.java ---
import java.io.IOException;
import java.net.*;
import java.util.Scanner;
public class UdpChatServerSimple {
public static void main(String[] args) throws IOException {
int port = 65435;
byte[] buffer = new byte[1024];
try (DatagramSocket socket = new DatagramSocket(port);
Scanner console = new Scanner(System.in)) {
System.out.println("UDP Server ready.");
DatagramPacket receivePacket = new DatagramPacket(buffer, buffer.length);
socket.receive(receivePacket);
InetAddress clientAddr = receivePacket.getAddress();
int clientPort = receivePacket.getPort();
System.out.println("Client discovered: " + clientAddr + ":" + clientPort);
while (true) {
socket.receive(receivePacket);
String receivedMsg = new String(receivePacket.getData(), 0, receivePacket.getLength());
System.out.println("Client > " + receivedMsg);
if (receivedMsg.equalsIgnoreCase("quit")) break;
System.out.print("Input > ");
String serverMsg = console.nextLine();
byte[] sendData = serverMsg.getBytes();
DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length, clientAddr,
clientPort);
socket.send(sendPacket);
if (serverMsg.equalsIgnoreCase("quit")) break;
}
}
}
}
```

```
// --- UdpChatClientSimple.java ---
import java.io.IOException;
import java.net.*;
import java.util.Scanner;
public class UdpChatClientSimple {
public static void main(String[] args) throws IOException {
String host = "127.0.0.1";
int serverPort = 65435;
byte[] buffer = new byte[1024];
try (DatagramSocket socket = new DatagramSocket();
Scanner console = new Scanner(System.in)) {
InetAddress serverAddr = InetAddress.getByName(host);
System.out.println("UDP Client ready. Type 'quit' to exit.");
while (true) {
System.out.print("Input > ");
String clientMsg = console.nextLine();
byte[] sendData = clientMsg.getBytes();
DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length, serverAddr,
serverPort);
socket.send(sendPacket);
if (clientMsg.equalsIgnoreCase("quit")) break;
DatagramPacket receivePacket = new DatagramPacket(buffer, buffer.length);
socket.receive(receivePacket);
String receivedMsg = new String(receivePacket.getData(), 0, receivePacket.getLength());
System.out.println("Server < " + receivedMsg);
if (receivedMsg.equalsIgnoreCase("quit")) break;
}
}
}
}

// --- SAMPLE OUTPUT (Interactive) ---
// Server Output:
// UDP Server ready.
// Client discovered: /127.0.0.1:50000
// Client > Hello UDP
// Input > Server acknowledged
// Client > quit
//
// Client Output:
// UDP Client ready. Type 'quit' to exit.
```

```
// Input > Hello UDP
// Server < Server acknowledged
// Input > quit

------------------------------------------------------
#5. STOP AND WAIT / SLIDING WINDOW PROTOCOL (Simulations)
------------------------------------------------------

// --- StopAndWaitSimulator.java ---
public class StopAndWaitSimulator {
public static void main(String[] args) {
int nextFrameToSend = 0;
int maxFrames = 3;
System.out.println("--- Stop-and-Wait Simulation ---");
while (nextFrameToSend < maxFrames) {
boolean ackReceived = false;
while (!ackReceived) {
System.out.println("Sender: Sending Frame " + nextFrameToSend);
if (Math.random() > 0.3) {
System.out.println("Receiver: Sent ACK " + nextFrameToSend);
ackReceived = true;
} else {
System.out.println("Sender: Timeout! Resending Frame " + nextFrameToSend);
}
}
nextFrameToSend++;
System.out.println("--------------------------------");
}
}
}

// --- SlidingWindowSimulator.java ---
public class SlidingWindowSimulator {
public static void main(String[] args) {
int windowSize = 3;
int nextFrameToSend = 0;
int expectedACK = 0;
int maxFrames = 5;
System.out.println("--- Sliding Window Simulation (Window Size: " + windowSize + ") ---");
while (expectedACK < maxFrames) {
while (nextFrameToSend < expectedACK + windowSize && nextFrameToSend < maxFrames)
{
```

```
System.out.println("Sender: Sending Frame " + nextFrameToSend);
nextFrameToSend++;
}
System.out.println("Receiver: Sending ACK for Frame " + expectedACK);
expectedACK++;
System.out.println("Sender: ACK received. Window Base moves to " + expectedACK + "\n");
}
}
}

// --- SAMPLE OUTPUT (Stop and Wait - Varies due to random loss) ---
// --- Stop-and-Wait Simulation ---
// Sender: Sending Frame 0
// Receiver: Sent ACK 0
// --------------------------------
// Sender: Sending Frame 1
// Sender: Timeout! Resending Frame 1
// Sender: Sending Frame 1
// Receiver: Sent ACK 1
// --------------------------------
// Sender: Sending Frame 2
// Receiver: Sent ACK 2
// --------------------------------
//
// --- SAMPLE OUTPUT (Sliding Window) ---
// --- Sliding Window Simulation (Window Size: 3) ---
// Sender: Sending Frame 0
// Sender: Sending Frame 1
// Sender: Sending Frame 2
// Receiver: Sending ACK for Frame 0
// Sender: ACK received. Window Base moves to 1
//
// Sender: Sending Frame 3
// Receiver: Sending ACK for Frame 1
// Sender: ACK received. Window Base moves to 2
//
// Sender: Sending Frame 4
// Receiver: Sending ACK for Frame 2
// Sender: ACK received. Window Base moves to 3
//
// Receiver: Sending ACK for Frame 3
// Sender: ACK received. Window Base moves to 4
```

```
//
// Receiver: Sending ACK for Frame 4
// Sender: ACK received. Window Base moves to 5

-----------------------------------------------------
#6. DNS/SNMP IMPLEMENTATION
-----------------------------------------------------

// --- DNSClientSimple.java ---
import java.net.*;
public class DNSClientSimple {
public static void main(String[] args) {
String hostname = "www.google.com";
try {
InetAddress ipAddress = InetAddress.getByName(hostname);
System.out.println("Input (Hostname): " + hostname);
System.out.println("Output (IP Address): " + ipAddress.getHostAddress());
} catch (UnknownHostException e) {
System.out.println("Error: Could not find host.");
}
}
}

// --- SimpleSNMPReceiver.java ---
import java.net.*;
public class SimpleSNMPReceiver {
public static void main(String[] args) {
int snmpPort = 162;
byte[] buffer = new byte[1024];
try (DatagramSocket socket = new DatagramSocket(snmpPort)) {
System.out.println("Listening for SNMP Traps on port " + snmpPort + "...");
while (true) {
DatagramPacket packet = new DatagramPacket(buffer, buffer.length);
socket.receive(packet);
String senderInfo = packet.getAddress().getHostAddress() + ":" + packet.getPort();
System.out.println("\n--- Input (TRAP RECEIVED) ---");
System.out.println("From: " + senderInfo);
System.out.println("Output (Data Size): " + packet.getLength() + " bytes");
}
} catch (Exception e) {
System.out.println("Error: " + e.getMessage());
}
```

```
}
}

// --- SAMPLE OUTPUT (DNS) ---
// Input (Hostname): www.google.com
// Output (IP Address): 142.250.67.100 (Note: IP address changes)
//
// --- SAMPLE OUTPUT (SNMP) ---
// Listening for SNMP Traps on port 162...
//
// --- Input (TRAP RECEIVED) ---
// From: 127.0.0.1:41345
// Output (Data Size): 72 bytes

--------------------------------------------------
#7. HTTP WEB PAGE DOWNLOAD
--------------------------------------------------

// --- HttpClientSimple.java ---
import java.io.*;
import java.net.*;
public class HttpClientSimple {
public static void main(String[] args) throws IOException {
String hostname = "example.com";
int port = 80;
try (Socket socket = new Socket(hostname, port);
PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream())))
{
System.out.println("Connected to " + hostname + ".");
out.println("GET / HTTP/1.1");
out.println("Host: " + hostname);
out.println("Connection: close");
out.println();
String line;
System.out.println("\n--- Output (First few lines of HTML) ---");
for(int i=0; i<10; i++) {
line = in.readLine();
if (line == null) break;
System.out.println(line);
}
}
```

```
}
}

// --- SAMPLE OUTPUT ---
// Connected to example.com.
//
// --- Output (First few lines of HTML) ---
// HTTP/1.1 200 OK
// Accept-Ranges: bytes
// Cache-Control: max-age=604800
// Content-Type: text/html; charset=UTF-8
// Date: Wed, 19 Nov 2025 15:50:00 GMT
// Etag: "3147526947"
// Expires: Wed, 26 Nov 2025 15:50:00 GMT

-----------------------------------------------------
#8. DISPLAY CLIENT'S ADDRESS AT SERVER END
-----------------------------------------------------

// --- AddressServerSimple.java ---
import java.io.IOException;
import java.net.*;
public class AddressServerSimple {
public static void main(String[] args) throws IOException {
int port = 65436;
try (ServerSocket server = new ServerSocket(port)) {
System.out.println("Server ready. Waiting for connection...");
try (Socket client = server.accept()) {
System.out.println("\n--- Output (Client Details) ---");
System.out.println("Client Connected!");
System.out.println("IP Address: " + client.getInetAddress().getHostAddress());
System.out.println("Port Number: " + client.getPort());
}
}
}
}

// --- AddressClientSimple.java ---
import java.io.IOException;
import java.net.*;
public class AddressClientSimple {
public static void main(String[] args) throws IOException {
```

```
String host = "127.0.0.1";
int port = 65436;
try (Socket socket = new Socket(host, port)) {
System.out.println("Client Input: Attempting connection...");
System.out.println("Client Output: Connection successful.");
}
}
}

// --- SAMPLE OUTPUT ---
// Server Output:
// Server ready. Waiting for connection...
//
// --- Output (Client Details) ---
// Client Connected!
// IP Address: 127.0.0.1
// Port Number: 50000 (Port will vary)
//
// Client Output:
// Client Input: Attempting connection...
// Client Output: Connection successful.
```