HTTP WEB CLIENTS TO DOWNLOAD WEBPAGE USING TCP SOCKETS

CLIENT

```
import java.net.*; import java.awt.image.*; import javax.imageio.*; import java.io.*;
public class Client {
   public static void main(String[] args) {
      try (Socket socket = new Socket("localhost", 4000)) {
         System.out.println("Client is running.");
         BufferedImage img = ImageIO.read(new File("digital_image_processing.jpg"));
         ByteArrayOutputStream baos = new ByteArrayOutputStream();
         ImageIO.write(img, "jpg", baos);
         byte[] imageBytes = baos.toByteArray();

         DataOutputStream dos = new DataOutputStream(socket.getOutputStream());
         dos.writeInt(imageBytes.length);
         dos.write(imageBytes);
         System.out.println("Image sent to server.");
      } catch (IOException e) {
         System.out.println("Exception: " + e.getMessage()); }}}
```

SERVER

```
import java.net.*; import java.awt.image.*; import javax.imageio.*; import java.io.*;
public class Server {
   public static void main(String[] args) {
      try (ServerSocket serverSocket = new ServerSocket(4000)) {
         System.out.println("Server Waiting for image");
         try (Socket socket = serverSocket.accept()) {
            System.out.println("Client connected.");
            DataInputStream dis = new DataInputStream(socket.getInputStream());
            int length = dis.readInt();
            byte[] imageBytes = new byte[length];
            dis.readFully(imageBytes);

            BufferedImage img = ImageIO.read(new ByteArrayInputStream(imageBytes));
            displayImage(img);
         }
      } catch (IOException e) {
         System.out.println("Exception: " + e.getMessage()); }}

   private static void displayImage(BufferedImage img) {
      JFrame frame = new JFrame("Server");
      frame.add(new JLabel(new ImageIcon(img)));
      frame.pack();
      frame.setVisible(true); }}
```

SOCKET PROGRAM FOR ECHO
CLIENT
```
import java.io.*; import java.net.*;
public class EchoClient {
    public static void main(String[] args) {
        try (Socket socket = new Socket("localhost", 8080);
             BufferedReader consoleReader = new BufferedReader(new
InputStreamReader(System.in));
             PrintWriter serverWriter = new PrintWriter(socket.getOutputStream(), true);
             BufferedReader serverReader = new BufferedReader(new
InputStreamReader(socket.getInputStream()))) {
            String userInput;
            System.out.println("Client connected. Type 'exit' to quit.");
            while (true) {
                System.out.print("client: ");
                userInput = consoleReader.readLine();
                serverWriter.println(userInput);
                if ("exit".equalsIgnoreCase(userInput)) {
                    break;
                }
                System.out.println("server: " + serverReader.readLine());
            }
        } catch (IOException e) {
            System.out.println("Connection error: " + e.getMessage());  }}}
```

SERVER
```
import java.io.*; import java.net.*;
public class EchoServer {
    public static void main(String[] args) {
        try (ServerSocket serverSocket = new ServerSocket(8080)) {
            System.out.println("Server waiting for client connection...");
            try (Socket clientSocket = serverSocket.accept();
                 BufferedReader clientReader = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
                 PrintWriter clientWriter = new PrintWriter(clientSocket.getOutputStream(), true)) {
                System.out.println("Client connected.");
                String receivedMessage;
                while ((receivedMessage = clientReader.readLine()) != null) {
                    System.out.println("Received from client: " + receivedMessage);
                    clientWriter.println(receivedMessage);
                }
            }
        } catch (IOException e) {
            System.out.println("Server error: " + e.getMessage());  }}}
```

SIMULATION OF DNS USING UDP SOCKETS
CLIENT
```
import java.io.*; import java.net.*;
public class UdpDnsClient {
    public static void main(String[] args) throws IOException {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        DatagramSocket clientSocket = new DatagramSocket();
        InetAddress serverAddress;
        if (args.length == 0) {
            serverAddress = InetAddress.getLocalHost();
        } else {
            serverAddress = InetAddress.getByName(args[0]);
        }
        byte[] sendData;
        byte[] receiveData = new byte[1024];
        int serverPort = 1362;
        System.out.print("Enter the hostname: ");
        String hostName = br.readLine();
        sendData = hostName.getBytes();
        DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length,
serverAddress, serverPort);
        clientSocket.send(sendPacket);
        DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);
        clientSocket.receive(receivePacket);
        String ipAddress = new String(receivePacket.getData()).trim();
        System.out.println("IP Address: " + ipAddress);
        clientSocket.close();  }}
```

SERVER
```
import java.io.*; import java.net.*;
public class UdpDnsServer {
    private static int indexOf(String[] array, String str) {
        str = str.trim();
        for (int i = 0; i < array.length; i++) {
            if (array[i].equals(str)) return i;
        }
        return -1;
    }
    public static void main(String[] args) throws IOException {
        String[] hosts = {"yahoo.com", "gmail.com", "cricinfo.com", "facebook.com"};
        String[] ips = {"68.180.206.184", "209.85.148.19", "80.168.92.140", "69.63.189.16"};
        DatagramSocket serverSocket = new DatagramSocket(1362);
        System.out.println("DNS Server is running. Press Ctrl + C to quit.");
        while (true) {
            byte[] receiveData = new byte[1024];
            DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
            serverSocket.receive(receivePacket);
            String hostName = new String(receivePacket.getData()).trim();
            InetAddress clientAddress = receivePacket.getAddress();
            int clientPort = receivePacket.getPort();
            System.out.println("Request for host: " + hostName);
            String response;
            int index = indexOf(hosts, hostName);
            if (index != -1) {
                response = ips[index];
            } else {
                response = "Host Not Found";
            }
            byte[] sendData = response.getBytes();
            DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length,
clientAddress, clientPort);
            serverSocket.send(sendPacket);  }}}
```
—---

SIMULATION OF ARP PROTOCOL
CLIENT

```
import java.io.*; import java.net.*;
public class ArpClient {
    public static void main(String[] args) {
        try (Socket socket = new Socket(InetAddress.getLocalHost(), 1100);
             PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
             BufferedReader userIn = new BufferedReader(new InputStreamReader(System.in));
             BufferedReader serverIn = new BufferedReader(new
-InputStreamReader(socket.getInputStream()))) {
            System.out.print("Enter the IP address: ");
            String ip = userIn.readLine();
            out.println(ip);
            String response;
            System.out.println("ARP response from server:");
            while ((response = serverIn.readLine()) != null) {
                System.out.println(response);  }
        } catch (IOException e) {
            System.out.println("Error: " + e.getMessage());  }}}
```

SERVER

```
import java.io.*; import java.net.*;
public class ArpServer {
    public static void main(String[] args) {
        try (ServerSocket serverSocket = new ServerSocket(1100)) {
            System.out.println("ARP Server is running...");
            while (true) {
                try (Socket clientSocket = serverSocket.accept();
                     PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);
                     BufferedReader in = new BufferedReader(new
-InputStreamReader(clientSocket.getInputStream()))) {
                    String ip = in.readLine();
                    System.out.println("Received IP address: " + ip);
                    Process process = Runtime.getRuntime().exec("arp -a " + ip);
                    try (BufferedReader arpOutput = new BufferedReader(new
-InputStreamReader(process.getInputStream()))) {
                        String line;
                        while ((line = arpOutput.readLine()) != null) {
                            out.println(line);   }   }
                } catch (IOException e) {
                    System.out.println("Client handling error: " + e.getMessage()); }
            }
        } catch (IOException e) {
            System.out.println("Server error: " + e.getMessage()); }}}
```