EXPERIMENT 2

CLIENT PROGRAM

```
import java.net.*;
import java.io.*;
import java.awt.image.*;
import javax.imageio.*;

public class Client {
   public static void main(String[] args) {
      Socket soc;
      BufferedImage img = null;

      try {

         img = ImageIO.read(new File("C:/Users/HP/Documents/digital_image_processing.jpg"));

         soc = new Socket("localhost", 4000);
         System.out.println("Client is running.");

         ByteArrayOutputStream baos = new ByteArrayOutputStream();
         ImageIO.write(img, "jpg", baos);
         baos.flush();
         byte[] bytes = baos.toByteArray();
         baos.close();

         OutputStream out = soc.getOutputStream();
         DataOutputStream dos = new DataOutputStream(out);
         dos.writeInt(bytes.length);
         dos.write(bytes, 0, bytes.length);

         System.out.println("Image sent to server.");
         dos.close();
         out.close();
         soc.close();
      } catch (IOException e) {
         System.out.println("Exception: " + e.getMessage());
      }
   }
}
```

SERVER PROGRAM

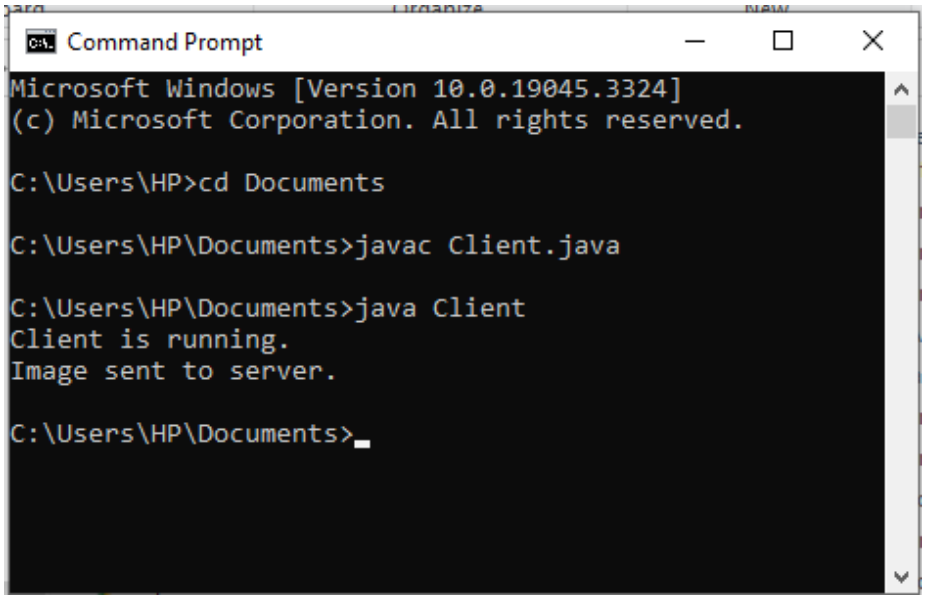
```
import java.net.*;
import java.io.*;
import java.awt.image.*;
import javax.imageio.*;
import javax.swing.*;
```

```
class Server {
   public static void main(String args[]) throws Exception {
      ServerSocket server = null;
      Socket socket = null;

      try {
         // Create a server socket listening on port 4000
         server = new ServerSocket(4000);
         System.out.println("Server waiting for image...");

         // Accept the connection from the client
         socket = server.accept();
         System.out.println("Client connected.");

         // Create input streams to receive the image
         InputStream in = socket.getInputStream();
         DataInputStream dis = new DataInputStream(in);

         // Read the size of the incoming image data
         int len = dis.readInt();
         System.out.println("Image Size: " + len / 1024 + "KB");

         // Create a byte array to hold the image data and read it from the input stream
         byte[] data = new byte[len];
         dis.readFully(data);

         // Convert the byte array back into a BufferedImage
         InputStream ian = new ByteArrayInputStream(data);
         BufferedImage bImage = ImageIO.read(ian);

         // Display the received image in a JFrame
         JFrame f = new JFrame("Server");
         ImageIcon icon = new ImageIcon(bImage);
         JLabel l = new JLabel();
         l.setIcon(icon);
         f.add(l);
         f.pack();
         f.setVisible(true);

         // Close the input streams
         dis.close();
         in.close();
      } catch (Exception e) {
         System.out.println("Exception: " + e.getMessage());
      } finally {
         // Close the socket and server socket
         if (socket != null) {
            socket.close();

         }
```

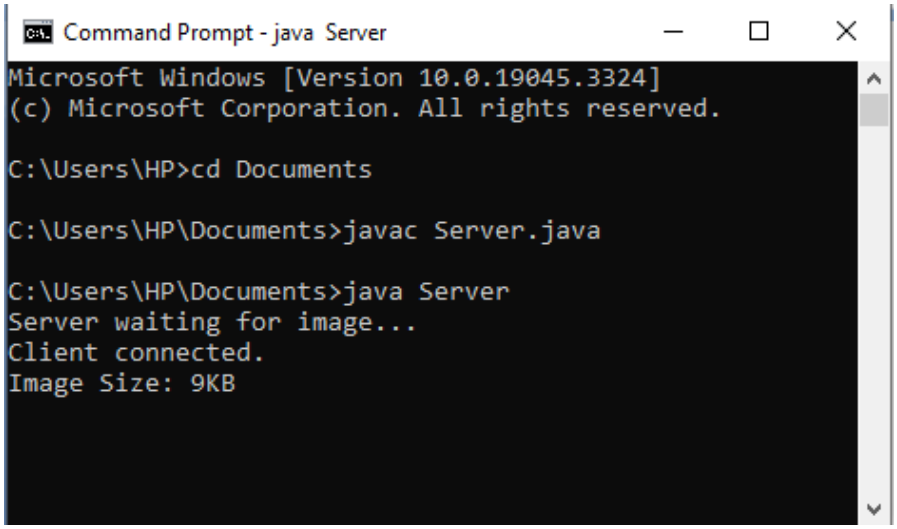
```
        if (server != null) {
            server.close();
        }
    }
  }
}
```

OUTPUT

CLIENT



SERVER

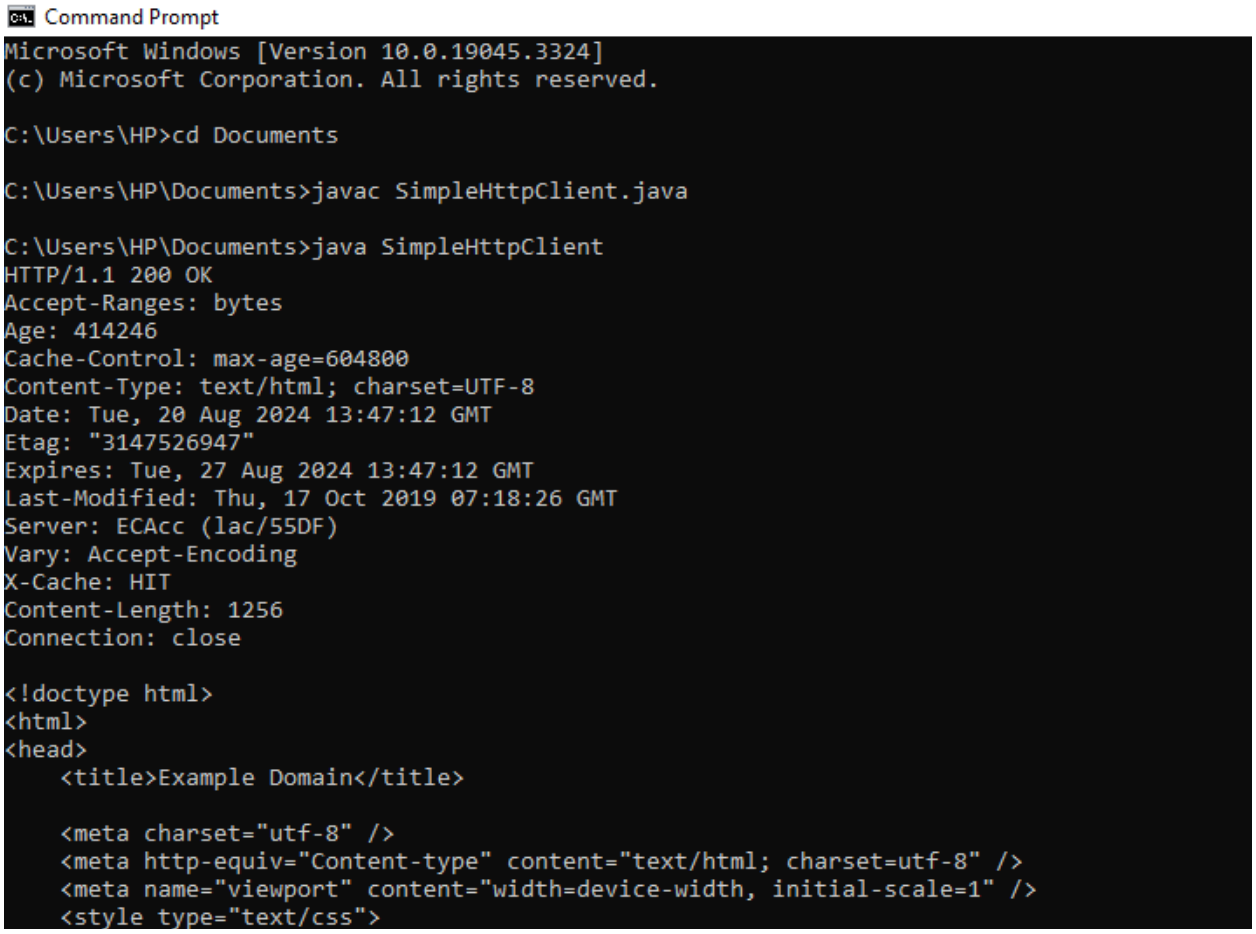


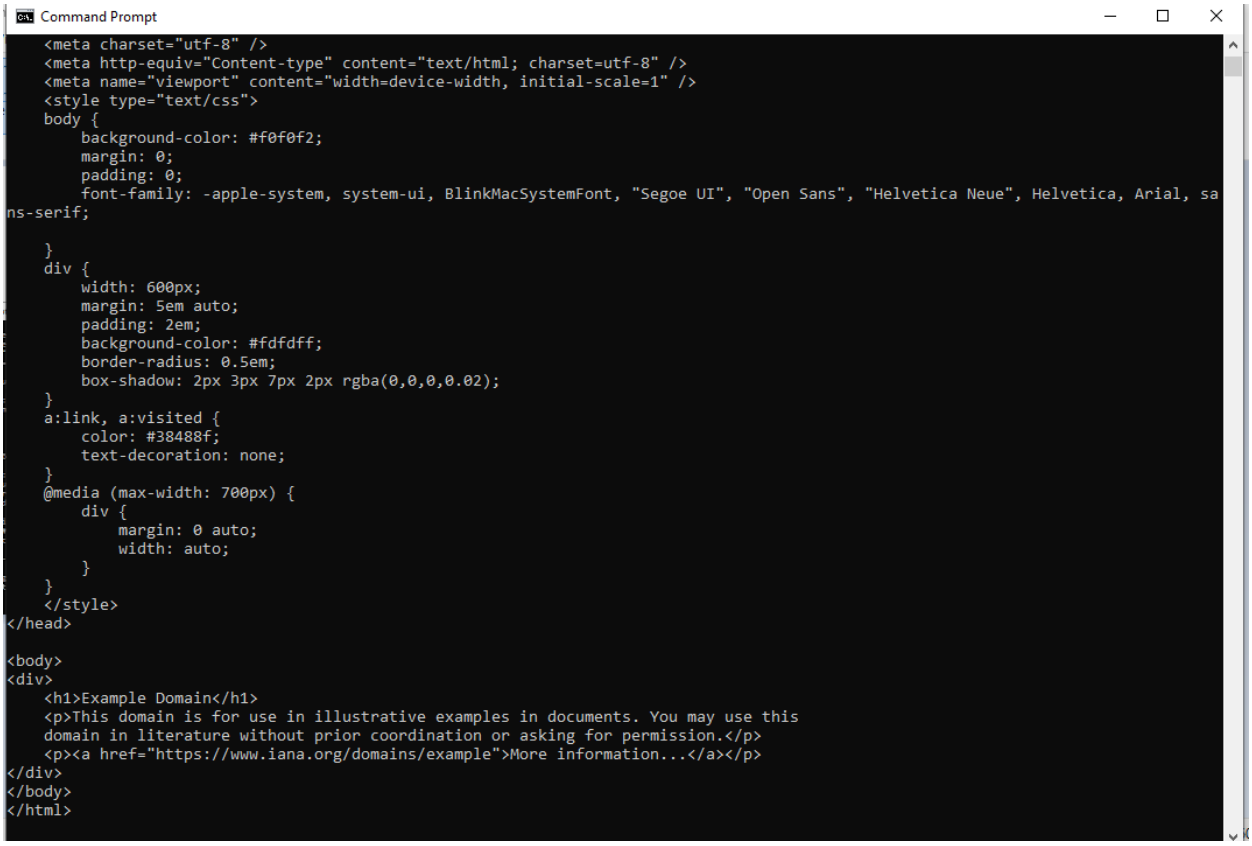
Client program:

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

public class SimpleHttpClient {
   public static void main(String[] args) {
      String host = "example.com"; // The host you're connecting to
      int port = 80; // HTTP port

      try (Socket socket = new Socket(host, port)) {
         // Send HTTP GET request
         PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
         out.println("GET / HTTP/1.1");
         out.println("Host: " + host);
         out.println("Connection: Close");
         out.println(); // Blank line to indicate end of headers

         // Read the response
         BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
         String responseLine;
         while ((responseLine = in.readLine()) != null) {
            System.out.println(responseLine);
         }
```

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

Output:


Command Prompt
Microsoft Windows [Version 10.0.19045.3324]
(c) Microsoft Corporation. All rights reserved.
C:\Users\HP>cd Documents
C:\Users\HP\Documents>javac SimpleHttpClient.java
C:\Users\HP\Documents>java SimpleHttpClient
HTTP/1.1 200 OK
Accept-Ranges: bytes
Age: 414246
Cache-Control: max-age=604800
Content-Type: text/html; charset=UTF-8
Date: Tue, 20 Aug 2024 13:47:12 GMT
Etag: "3147526947"
Expires: Tue, 27 Aug 2024 13:47:12 GMT
Last-Modified: Thu, 17 Oct 2019 07:18:26 GMT
Server: ECAcc (lac/55DF)
Vary: Accept-Encoding
X-Cache: HIT
Content-Length: 1256
Connection: close
<!doctype html>
<html>
<head>
<title>Example Domain</title>
<meta charset="utf-8" />
<meta http-equiv="Content-type" content="text/html; charset=utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1" />
<style type="text/css">
```
```



EXPERIMENT 3(a) echo

CLIENT PROGRAM

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

public class EClient {
   public static void main(String[] args) {
      try {
         Socket socket = new Socket("localhost", 9000);
         BufferedReader userInput = new BufferedReader(new
InputStreamReader(System.in));
         PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
         BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));

         String line;
```

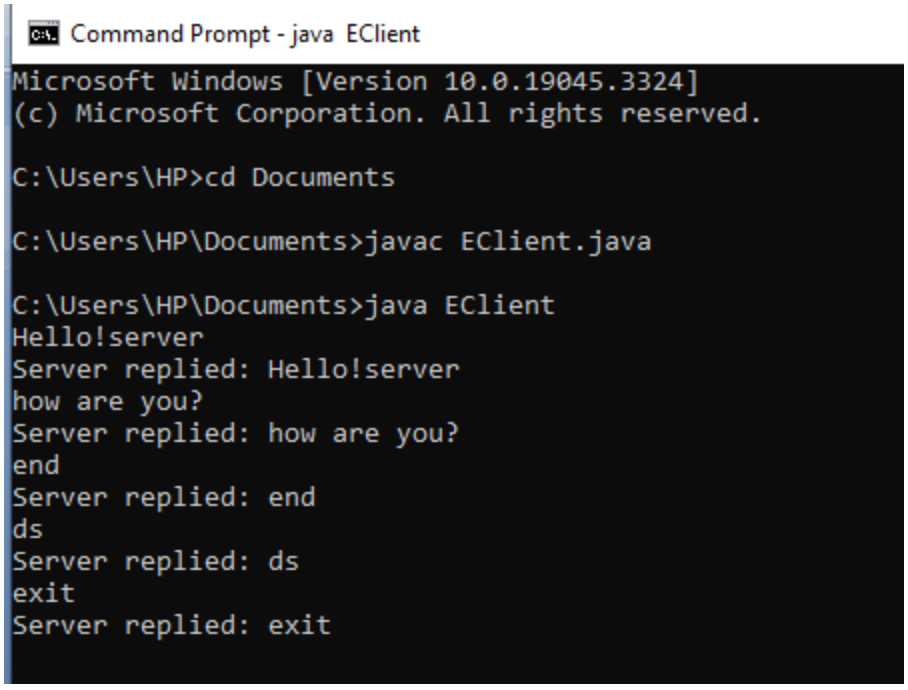
```
        while ((line = userInput.readLine()) != null) {
            out.println(line);
            System.out.println("Server replied: " + in.readLine());
        }

        socket.close();
    } catch (IOException e) {
        System.err.println("IOException: " + e.getMessage());
    }
  }
}
```

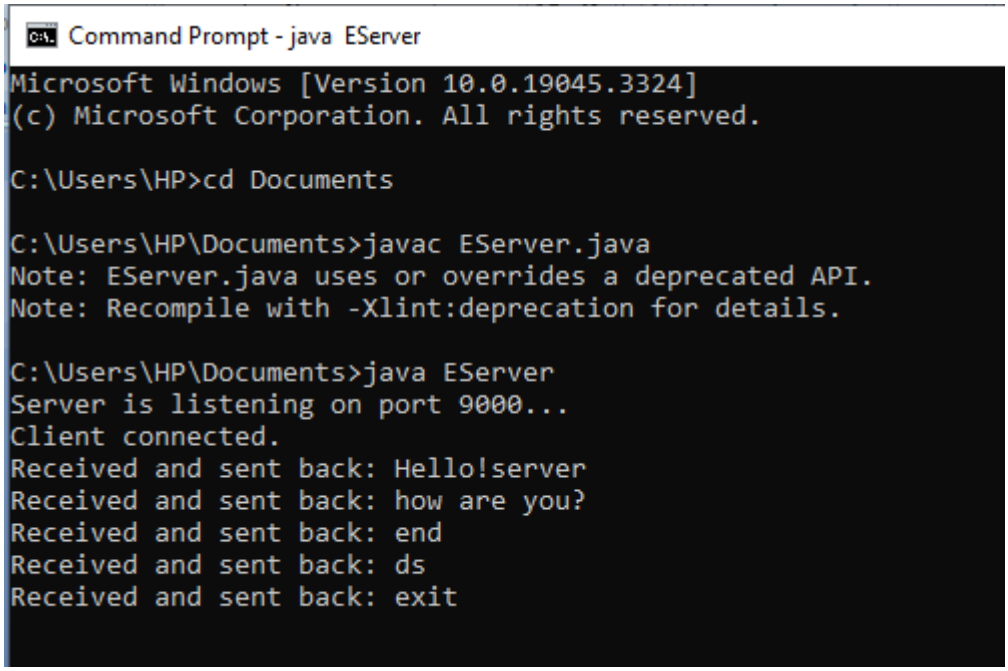
SERVER PROGRAM

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

public class EServer {
  public static void main(String[] args) {
    ServerSocket serverSocket = null;
    Socket clientSocket = null;
    DataInputStream inputStream = null;
    PrintStream outputStream = null;

    try {
      // Create a ServerSocket to listen on port 9000
      serverSocket = new ServerSocket(9000);
      System.out.println("Server is listening on port 9000...");

      // Accept a connection from the client
      clientSocket = serverSocket.accept();
      System.out.println("Client connected.");

      // Create input and output streams
      inputStream = new DataInputStream(clientSocket.getInputStream());
      outputStream = new PrintStream(clientSocket.getOutputStream());

      // Read lines from the client and echo them back
```

```
            String line;
            while ((line = inputStream.readLine()) != null) {
                outputStream.println(line);
                System.out.println("Received and sent back: " + line);
            }
        } catch (IOException e) {
            System.err.println("IOException: " + e.getMessage());
        } finally {
            // Close all resources
            try {
                if (inputStream != null) inputStream.close();
                if (outputStream != null) outputStream.close();
                if (clientSocket != null) clientSocket.close();
                if (serverSocket != null) serverSocket.close();
            } catch (IOException e) {
                System.err.println("Error closing resources: " + e.getMessage());
            }
        }
    }
}
```

OUTPUT





Experiment 3(b)-chat

Client program

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

class UDPclient {
   public static DatagramSocket ds;
   public static int clientport = 789, serverport = 790;

   public static void main(String args[]) throws Exception {
      byte buffer[] = new byte[1024];
      ds = new DatagramSocket(serverport);
      BufferedReader dis = new BufferedReader(new InputStreamReader(System.in));
      System.out.println("Server waiting");
      InetAddress ia = InetAddress.getLocalHost();

      while (true) {
         System.out.print("Client: ");
         String str = dis.readLine();
         if (str.equals("end"))
```

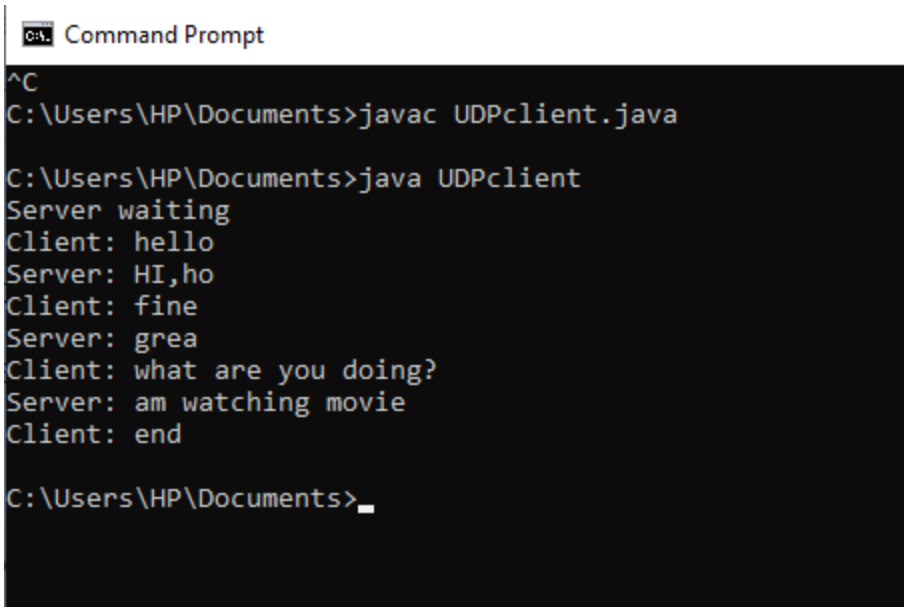
```
            break;
         buffer = str.getBytes();
         ds.send(new DatagramPacket(buffer, str.length(), ia, clientport));
         DatagramPacket p = new DatagramPacket(buffer, buffer.length);
         ds.receive(p);
         String psx = new String(p.getData(), 0, p.getLength());
         System.out.println("Server: " + psx);
      }
      ds.close();
   }
}
```

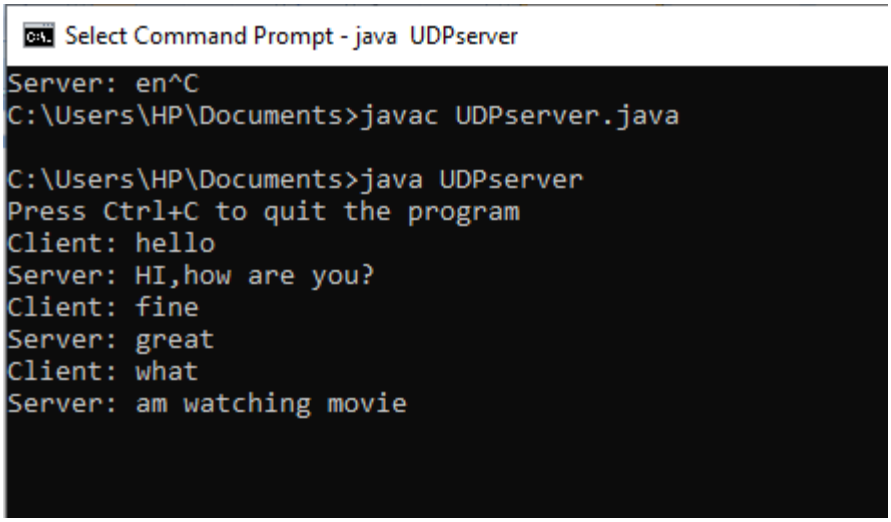
Server program

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

class UDPserver {
   public static DatagramSocket ds;
   public static byte buffer[] = new byte[1024];
   public static int clientport = 789, serverport = 790;

   public static void main(String args[]) throws Exception {
      ds = new DatagramSocket(clientport);
      System.out.println("Press Ctrl+C to quit the program");
      BufferedReader dis = new BufferedReader(new InputStreamReader(System.in));
      InetAddress ia = InetAddress.getLocalHost();

      while (true) {
         DatagramPacket p = new DatagramPacket(buffer, buffer.length);
         ds.receive(p);
         String psx = new String(p.getData(), 0, p.getLength());
         System.out.println("Client: " + psx);
         System.out.print("Server: ");
         String str = dis.readLine();
         if (str.equals("end"))
            break;
```

```
            buffer = str.getBytes();
            ds.send(new DatagramPacket(buffer, str.length(), ia, serverport));
        }
        ds.close();
    }
}
```

output





EXPERIMENT -4
CLIENT PROGRAM

```
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 ipaddress;

      // Set IP address based on command-line argument or default to localhost
      if (args.length == 0) {
         ipaddress = InetAddress.getLocalHost();
      } else {
         ipaddress = InetAddress.getByName(args[0]);
      }

      int portaddr = 1362;
      byte[] senddata = new byte[1024];
      byte[] receivedata = new byte[1024];

      System.out.print("Enter the hostname: ");
      String sentence = br.readLine();
      senddata = sentence.getBytes();

      DatagramPacket pack = new DatagramPacket(senddata, senddata.length, ipaddress,
portaddr);
      clientsocket.send(pack);

      DatagramPacket recvpack = new DatagramPacket(receivedata, receivedata.length);
      clientsocket.receive(recvpack);
      String modified = new String(recvpack.getData(), 0, recvpack.getLength());

      System.out.println("IP Address: " + modified);
      clientsocket.close();
   }
```

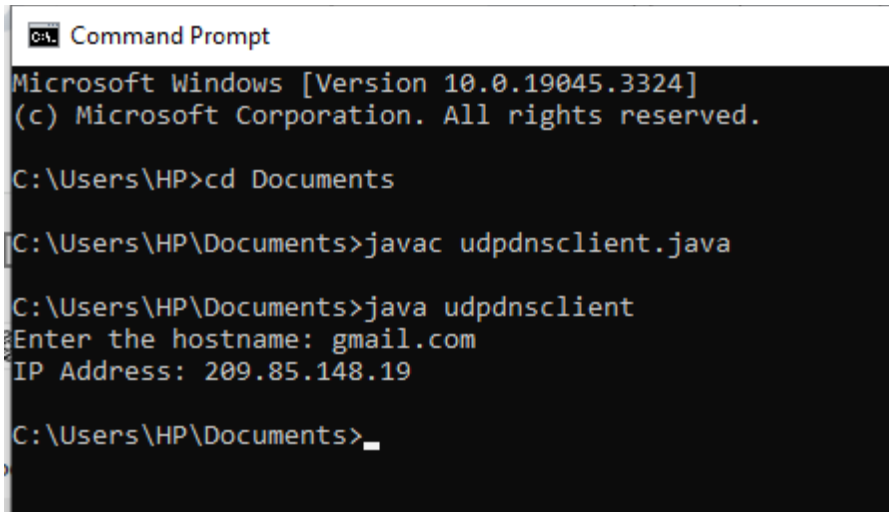
```
}
```

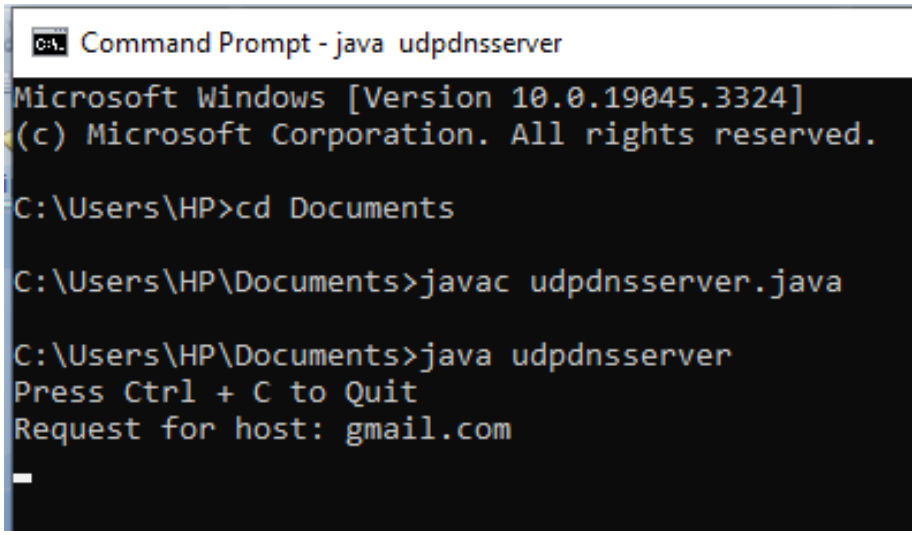
SERVER PROGRAM

```
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[] ip = {"68.180.206.184", "209.85.148.19", "80.168.92.140", "69.63.189.16"};
      System.out.println("Press Ctrl + C to Quit");

      DatagramSocket serversocket = new DatagramSocket(1362);

      while (true) {
         byte[] receivedata = new byte[1024];
         DatagramPacket recvpack = new DatagramPacket(receivedata, receivedata.length);
         serversocket.receive(recvpack);

         String sen = new String(recvpack.getData(), 0, recvpack.getLength()).trim();
         InetAddress ipaddress = recvpack.getAddress();
         int port = recvpack.getPort();

         String capsent;
         System.out.println("Request for host: " + sen);

         if (indexOf(hosts, sen) != -1) {
            capsent = ip[indexOf(hosts, sen)];
         } else {
            capsent = "Host Not Found";
```

```
        }

        byte[] senddata = capsent.getBytes();
        DatagramPacket pack = new DatagramPacket(senddata, senddata.length, ipaddress,
port);
        serversocket.send(pack);
      }
   }
}
```

OUTPUT:





EXPERIMENT-6a-ARP

Client program

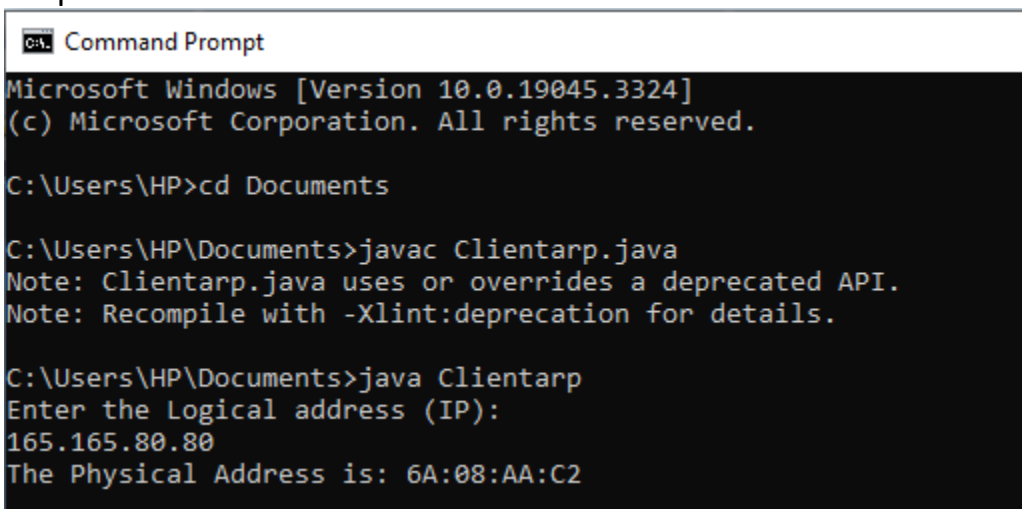
```
import java.io.*;
import java.net.*;

class Clientarp {
   public static void main(String[] args) {
      try {
         BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
         Socket clsct = new Socket("127.0.0.1", 9999); // Connect to localhost on port 139

         DataInputStream din = new DataInputStream(clsct.getInputStream());
         DataOutputStream dout = new DataOutputStream(clsct.getOutputStream());

         System.out.println("Enter the Logical address (IP):");
         String str1 = in.readLine();
         dout.writeBytes(str1 + '\n'); // Send IP address to server

         String str = din.readLine(); // Receive MAC address from server
         System.out.println("The Physical Address is: " + str);

         clsct.close();
      } catch (Exception e) {
         System.out.println(e);
      }
   }
}
```

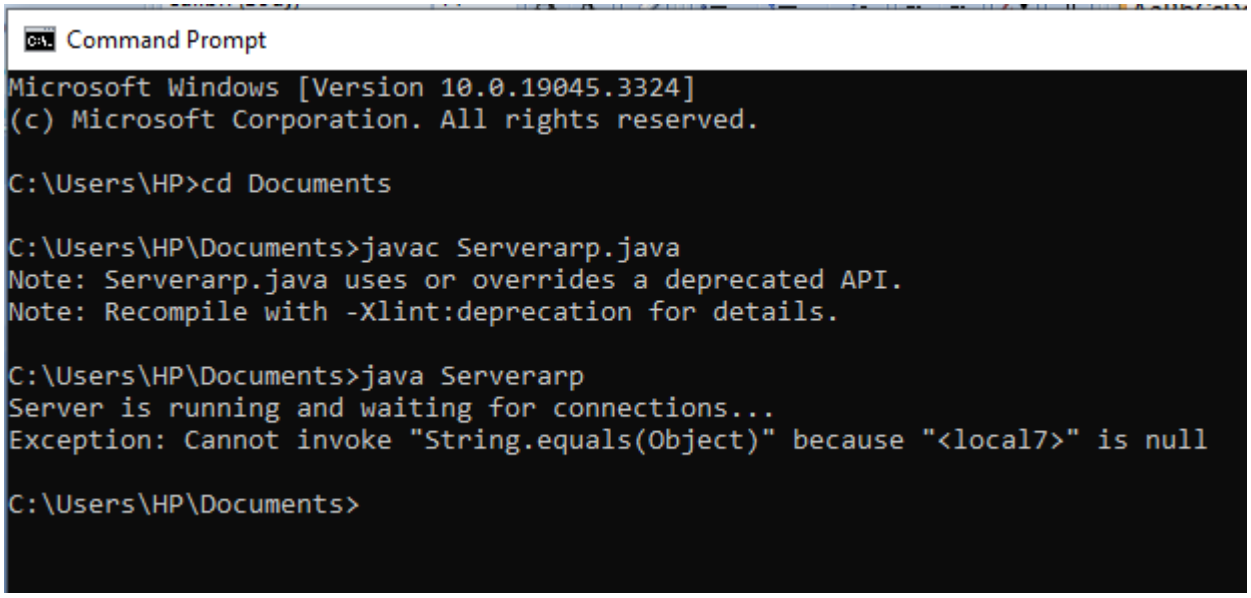
Server program

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

class Serverarp {
   public static void main(String[] args) {
```

```
        try {
            ServerSocket serverSocket = new ServerSocket(9999);
            System.out.println("Server is running and waiting for connections...");

            Socket clientSocket = serverSocket.accept();
            DataInputStream din = new DataInputStream(clientSocket.getInputStream());
            DataOutputStream dout = new DataOutputStream(clientSocket.getOutputStream());

            String[] ip = {"165.165.80.80", "165.165.79.1"};
            String[] mac = {"6A:08:AA:C2", "8A:BC:E3:FA"};

            while (true) {
                String str = din.readLine();

                boolean found = false;
                for (int i = 0; i < ip.length; i++) {
                    if (str.equals(ip[i])) {
                        dout.writeBytes(mac[i] + "\n");
                        found = true;
                        break;
                    }
                }

                if (!found) {
                    dout.writeBytes("IP address not found\n");
                }
            }

            // Close resources (though the server loop runs indefinitely)
            // din.close();
            // dout.close();
            // clientSocket.close();
            // serverSocket.close();

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

Output:





Experiment 6b-RARP

Client program

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

class Clientrarp12 {
   public static void main(String args[]) {
      try {
         // Create a DatagramSocket for sending and receiving UDP packets
```

```
        DatagramSocket client = new DatagramSocket();

        // Get the local host address (server address)
        InetAddress addr = InetAddress.getByName("127.0.0.1");

        // Create byte arrays to send and receive data
        byte[] sendbyte = new byte[1024];
        byte[] receivebyte = new byte[1024];

        // BufferedReader for user input
        BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
        System.out.println("Enter the Physical address (MAC):");

        // Read the MAC address input from the user
        String str = in.readLine();
        sendbyte = str.getBytes();

        // Create a DatagramPacket to send the MAC address to the server
        DatagramPacket sender = new DatagramPacket(sendbyte, sendbyte.length, addr,
1309);
        client.send(sender);

        // Create a DatagramPacket to receive the IP address from the server
        DatagramPacket receiver = new DatagramPacket(receivebyte, receivebyte.length);
        client.receive(receiver);

        // Convert the received data to a string and print it
        String s = new String(receiver.getData());
        System.out.println("The Logical Address is (IP): " + s.trim());

        // Close the client socket
        client.close();
      } catch (Exception e) {
        System.out.println(e);
      }
   }
}
```

Server program

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

class Serverrarp12 {
   public static void main(String args[]) {
      try {
         // Create a DatagramSocket to listen on port 1309
         DatagramSocket server = new DatagramSocket(1309);

         while (true) {
            // Create byte arrays to send and receive data
            byte[] sendbyte = new byte[1024];
            byte[] receivebyte = new byte[1024];

            // Create a DatagramPacket to receive data from the client
            DatagramPacket receiver = new DatagramPacket(receivebyte, receivebyte.length);
            server.receive(receiver);

            // Convert the received data to a string (MAC address)
            String str = new String(receiver.getData());
            String s = str.trim();

            // Get the client's address and port
            InetAddress addr = receiver.getAddress();
            int port = receiver.getPort();

            // Predefined lists of IP addresses and their corresponding MAC addresses
            String ip[] = {"165.165.80.80", "165.165.79.1"};
            String mac[] = {"6A:08:AA:C2", "8A:BC:E3:FA"};

            // Find the IP address corresponding to the received MAC address
            for (int i = 0; i < mac.length; i++) {
               if (s.equals(mac[i])) {
                  sendbyte = ip[i].getBytes();

                  // Create a DatagramPacket to send the IP address back to the client
                  DatagramPacket sender = new DatagramPacket(sendbyte, sendbyte.length,
addr, port);
                  server.send(sender);
```

```
                break;
            }
        }
    }
} catch (Exception e) {
    System.out.println(e);
}
}
}
```

Output: