Ex: 01 **Basic Commands**

Date:

Aim:

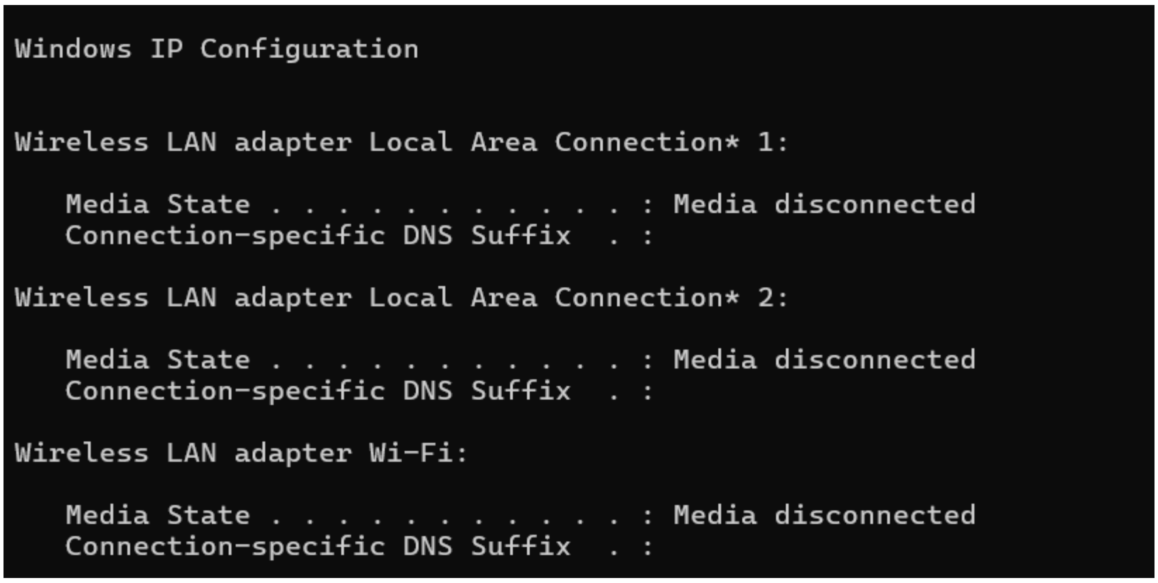
To practice and implement the basic commands in the Command Prompt

Algorithm:

1. Start the cmd and enter the given basic commands

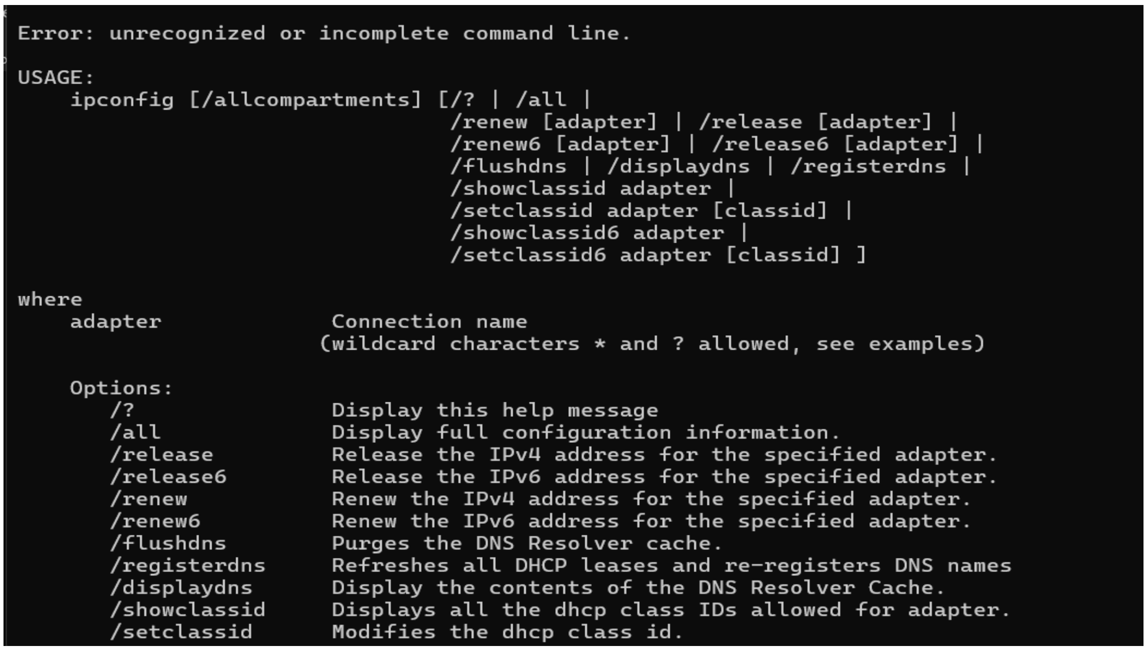
Commands:

1. ipconfig



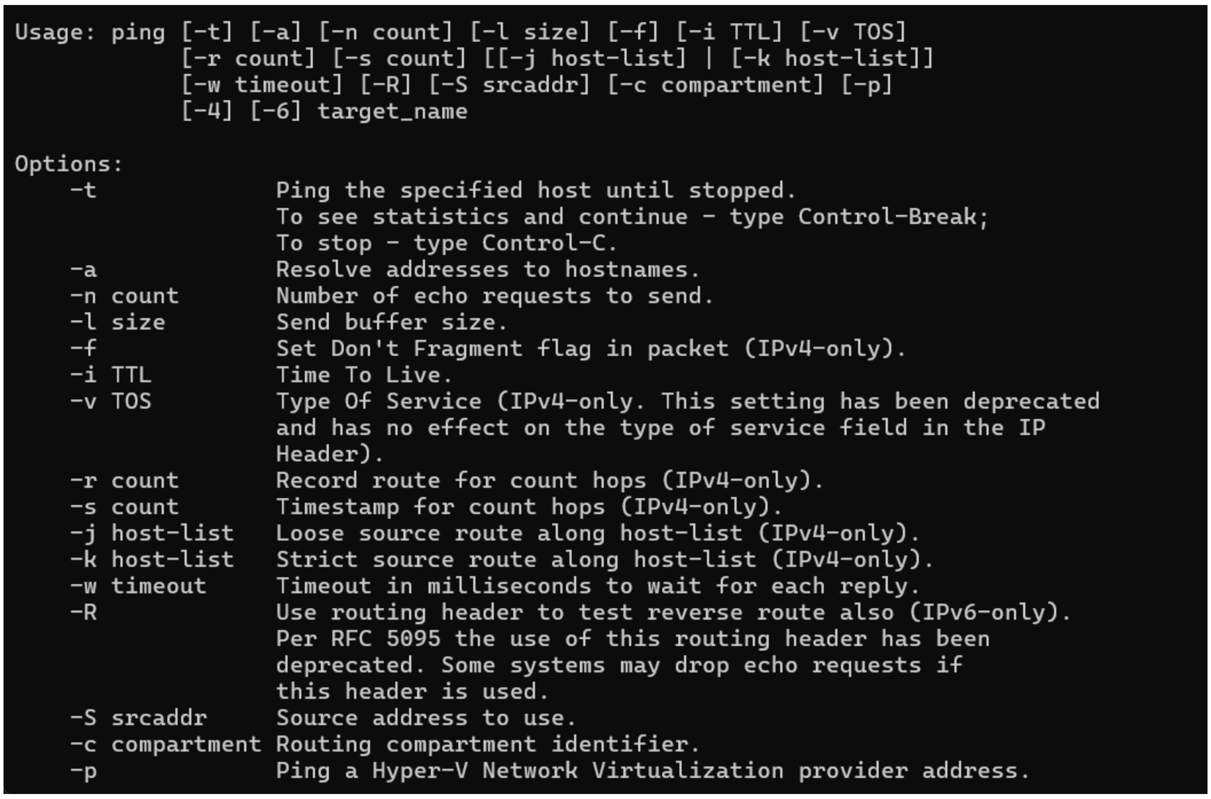
```
Windows IP Configuration


Wireless LAN adapter Local Area Connection* 1:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Local Area Connection* 2:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Wi-Fi:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :
```

2. ipconfig all



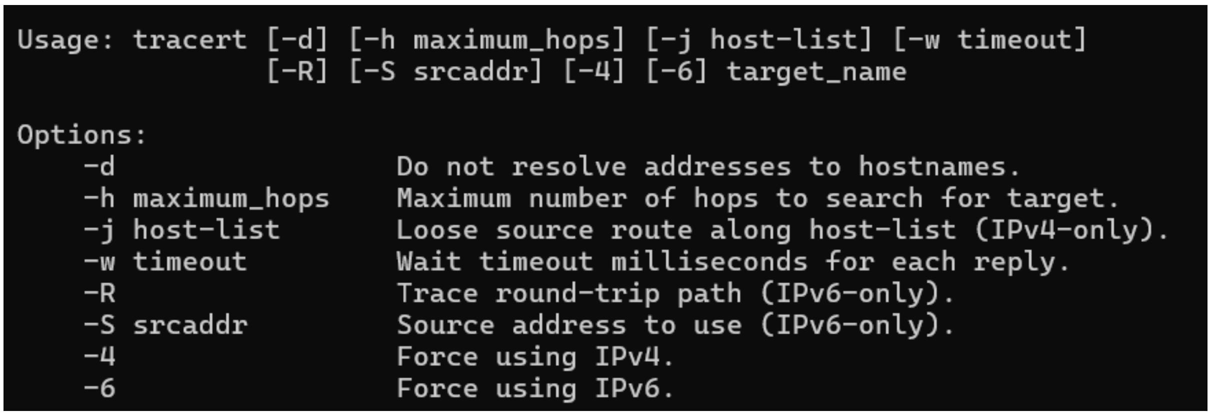
```
Error: unrecognized or incomplete command line.

USAGE:
    ipconfig [/allcompartments] [/? | /all |
                                 /renew [adapter] | /release [adapter] |
                                 /renew6 [adapter] | /release6 [adapter] |
                                 /flushdns | /displaydns | /registerdns |
                                 /showclassid adapter |
                                 /setclassid adapter [classid] |
                                 /showclassid6 adapter |
                                 /setclassid6 adapter [classid] ]

where
    adapter             Connection name
                       (wildcard characters * and ? allowed, see examples)

    Options:
       /?               Display this help message
       /all             Display full configuration information.
       /release         Release the IPv4 address for the specified adapter.
       /release6        Release the IPv6 address for the specified adapter.
       /renew           Renew the IPv4 address for the specified adapter.
       /renew6          Renew the IPv6 address for the specified adapter.
       /flushdns        Purges the DNS Resolver cache.
       /registerdns     Refreshes all DHCP leases and re-registers DNS names
       /displaydns      Display the contents of the DNS Resolver Cache.
       /showclassid     Displays all the dhcp class IDs allowed for adapter.
       /setclassid      Modifies the dhcp class id.
```

3. ping



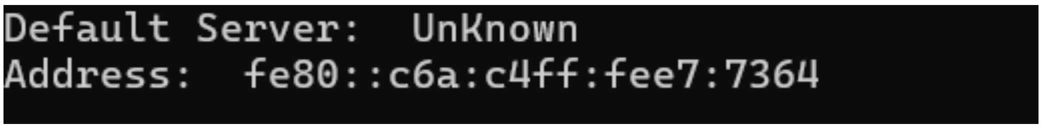
```
Usage: ping [-t] [-a] [-n count] [-l size] [-f] [-i TTL] [-v TOS]
            [-r count] [-s count] [[-j host-list] | [-k host-list]]
            [-w timeout] [-R] [-S srcaddr] [-c compartment] [-p]
            [-4] [-6] target_name

Options:
    -t             Ping the specified host until stopped.
                   To see statistics and continue - type Control-Break;
                   To stop - type Control-C.
    -a             Resolve addresses to hostnames.
    -n count       Number of echo requests to send.
    -l size        Send buffer size.
    -f             Set Don't Fragment flag in packet (IPv4-only).
    -i TTL         Time To Live.
    -v TOS         Type Of Service (IPv4-only. This setting has been deprecated
                   and has no effect on the type of service field in the IP
                   Header).
    -r count       Record route for count hops (IPv4-only).
    -s count       Timestamp for count hops (IPv4-only).
    -j host-list   Loose source route along host-list (IPv4-only).
    -k host-list   Strict source route along host-list (IPv4-only).
    -w timeout     Timeout in milliseconds to wait for each reply.
    -R             Use routing header to test reverse route also (IPv6-only).
                   Per RFC 5095 the use of this routing header has been
                   deprecated. Some systems may drop echo requests if
                   this header is used.
    -S srcaddr     Source address to use.
    -c compartment Routing compartment identifier.
    -p             Ping a Hyper-V Network Virtualization provider address.
```

4. tracert



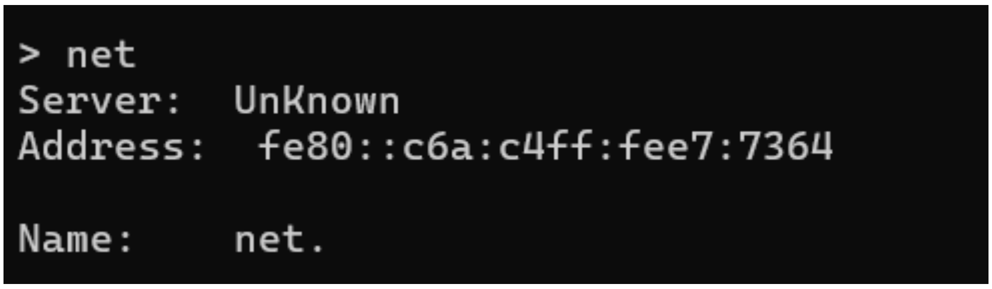
```
Usage: tracert [-d] [-h maximum_hops] [-j host-list] [-w timeout]
               [-R] [-S srcaddr] [-4] [-6] target_name

Options:
    -d                 Do not resolve addresses to hostnames.
    -h maximum_hops    Maximum number of hops to search for target.
    -j host-list       Loose source route along host-list (IPv4-only).
    -w timeout         Wait timeout milliseconds for each reply.
    -R                 Trace round-trip path (IPv6-only).
    -S srcaddr         Source address to use (IPv6-only).
    -4                 Force using IPv4.
    -6                 Force using IPv6.
```

5. nslookup



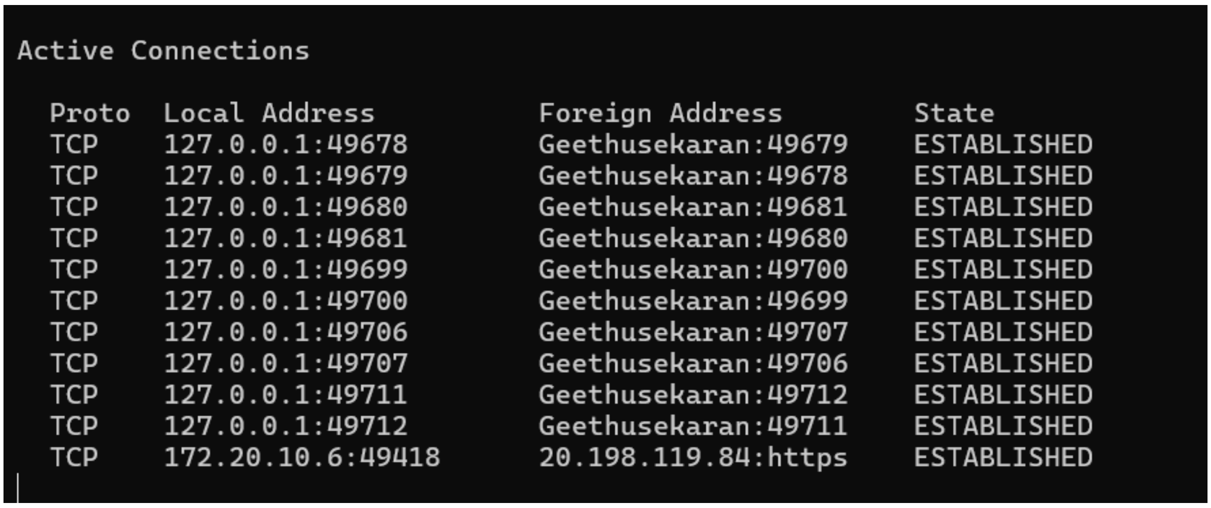
```
Default Server:  UnKnown
Address:  fe80::c6a:c4ff:fee7:7364
```

6. net



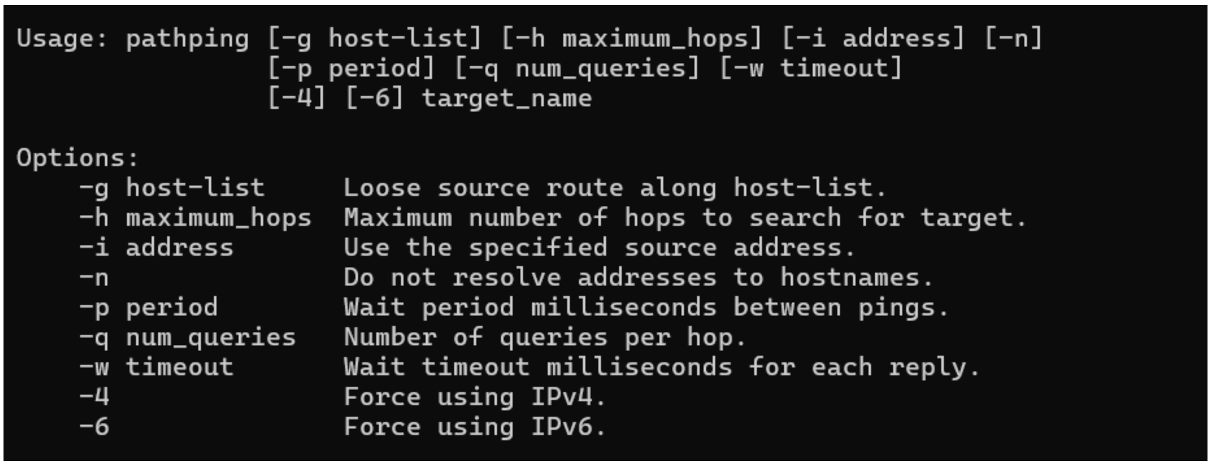
```
> net
Server:  UnKnown
Address:  fe80::c6a:c4ff:fee7:7364

Name:    net.
```

7. netstat



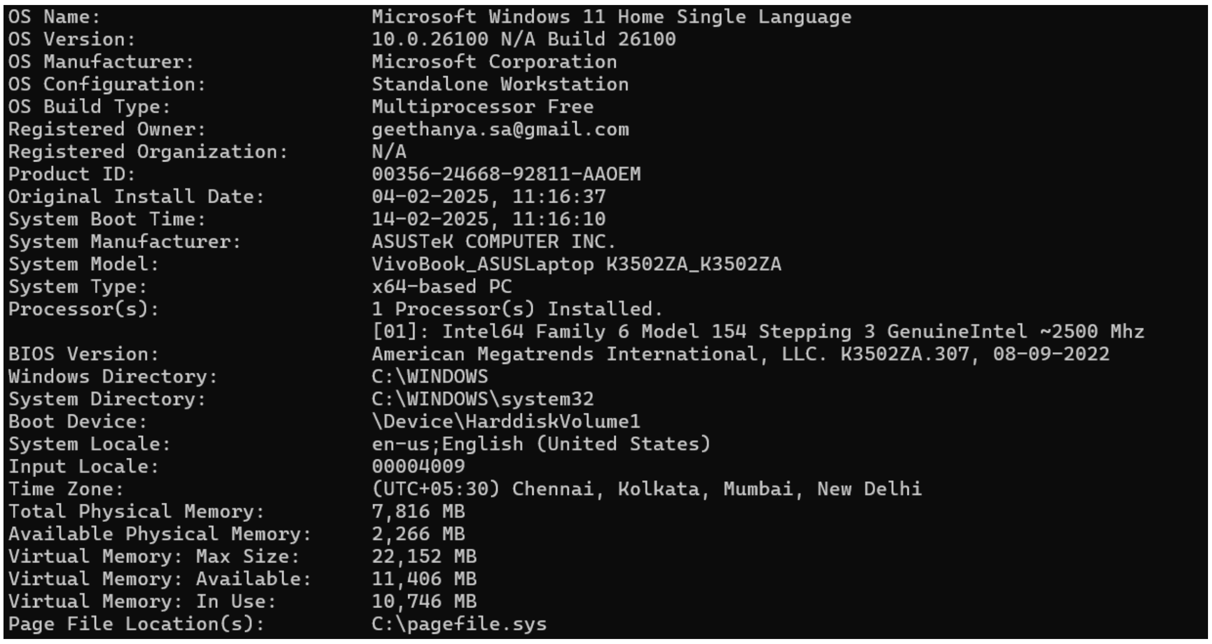
```
Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    127.0.0.1:49678        Geethusekaran:49679    ESTABLISHED
  TCP    127.0.0.1:49679        Geethusekaran:49678    ESTABLISHED
  TCP    127.0.0.1:49680        Geethusekaran:49681    ESTABLISHED
  TCP    127.0.0.1:49681        Geethusekaran:49680    ESTABLISHED
  TCP    127.0.0.1:49699        Geethusekaran:49700    ESTABLISHED
  TCP    127.0.0.1:49700        Geethusekaran:49699    ESTABLISHED
  TCP    127.0.0.1:49706        Geethusekaran:49707    ESTABLISHED
  TCP    127.0.0.1:49707        Geethusekaran:49706    ESTABLISHED
  TCP    127.0.0.1:49711        Geethusekaran:49712    ESTABLISHED
  TCP    127.0.0.1:49712        Geethusekaran:49711    ESTABLISHED
  TCP    172.20.10.6:49418      20.198.119.84:https    ESTABLISHED
```

8. pathping



```
Usage: pathping [-g host-list] [-h maximum_hops] [-i address] [-n]
                [-p period] [-q num_queries] [-w timeout]
                [-4] [-6] target_name

Options:
    -g host-list     Loose source route along host-list.
    -h maximum_hops  Maximum number of hops to search for target.
    -i address       Use the specified source address.
    -n               Do not resolve addresses to hostnames.
    -p period        Wait period milliseconds between pings.
    -q num_queries   Number of queries per hop.
    -w timeout       Wait timeout milliseconds for each reply.
    -4               Force using IPv4.
    -6               Force using IPv6.
```

9. system info



```
OS Name:                   Microsoft Windows 11 Home Single Language
OS Version:                10.0.26100 N/A Build 26100
OS Manufacturer:           Microsoft Corporation
OS Configuration:          Standalone Workstation
OS Build Type:             Multiprocessor Free
Registered Owner:          geethanya.sa@gmail.com
Registered Organization:   N/A
Product ID:                00356-24668-92811-AAOEM
Original Install Date:     04-02-2025, 11:16:37
System Boot Time:          14-02-2025, 11:16:10
System Manufacturer:       ASUSTeK COMPUTER INC.
System Model:              VivoBook_ASUSLaptop K3502ZA_K3502ZA
System Type:               x64-based PC
Processor(s):              1 Processor(s) Installed.
                           [01]: Intel64 Family 6 Model 154 Stepping 3 GenuineIntel ~2500 Mhz
BIOS Version:              American Megatrends International, LLC. K3502ZA.307, 08-09-2022
Windows Directory:         C:\WINDOWS
System Directory:          C:\WINDOWS\system32
Boot Device:               \Device\HarddiskVolume1
System Locale:             en-us;English (United States)
Input Locale:              00004009
Time Zone:                 (UTC+05:30) Chennai, Kolkata, Mumbai, New Delhi
Total Physical Memory:     7,816 MB
Available Physical Memory: 2,266 MB
Virtual Memory: Max Size:  22,152 MB
Virtual Memory: Available: 11,406 MB
Virtual Memory: In Use:    10,746 MB
Page File Location(s):     C:\pagefile.sys
```

Result:

Thus the commands have been implemented successfully

Ex: 02 **Information Retrieval**

Date:

Aim:

To write a program in Java to perform the information retrieval operation

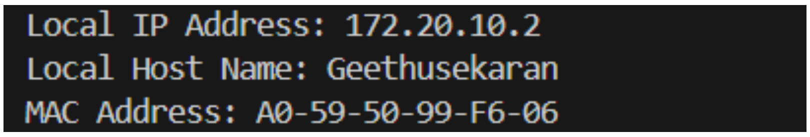
Algorithm:

1. Start the program
2. Import the Inet address, network interface, socket exception and unknownhost
3. Create instance of the NetworkInterface class and use the getByInetAddress (localhost) method
4. Use the getHardwareAddress( ) method to get the mac address
5. Stop the program

Code:

```java
import java.net.InetAddress;
import java.net.NetworkInterface;
import java.net.SocketException;
import java.net.UnknownHostException;

public class Ex2 {
  public static void main(String[] args) {
    try {
      InetAddress localhost = InetAddress.getLocalHost();
      System.out.println("Local IP Address: " + localhost.getHostAddress());
      System.out.println("Local Host name: " + localhost.getHostName());

      NetworkInterface ni = NetworkInterface.getByInetAddress(localhost);
      byte[] mac = ni.getHardwareAddress();
      System.out.print("MAC address: ");

      StringBuilder stringBuilder = new StringBuilder();
      for (int i = 0; i < mac.length; i++) {
          stringBuilder.append(String.format("%02X%s", mac[i], (i < mac.length - 1) ? "-" : ""));
        }
      System.out.println(stringBuilder.toString());

    } catch (UnknownHostException | SocketException ex) {
      ex.printStackTrace();
    }
  }
}
```

Output:



```
Local IP Address: 172.20.10.2
Local Host Name: Geethusekaran
MAC Address: A0-59-50-99-F6-06
```

Result:

Thus the program has been done successfully

Ex: 03 **One-way communication using TCP**

Date:

Aim:

To implement One-way communication using TCP

Algorithm:

1. Start the program
2. In the server code, start the session using ServerSocket and Socket classes.
3. Create an object for DataOutputStream class to send message to the client
4. Create object for BufferedReader class to get message to be sent from the user to the client in the server console
5. Loop to continuously read input from server&#39;s console and send it to client
6. Input is read using readLine( ) and message is sent using writeUTF( )
7. In the client code, instantiate the Socket class using the port number
8. Create object for the DataInputStream to receive data from server
9. Loop to continuously listen for messages from server
10. Messages are scanned using the readUTF( ) method
11. Stop the program

Code:

server:

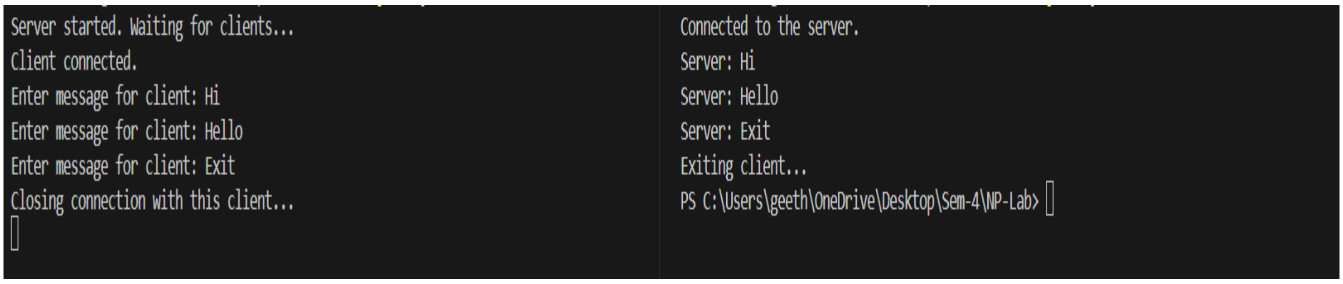
```
import java.io.*;
import java.net.*;
public class Server {
   public static void main(String[] args) {
      try {
         ServerSocket serverSocket = new ServerSocket(6666);
         System.out.println("Server is listening on port 6666...");
         Socket socket = serverSocket.accept();
         System.out.println("Client connected.");
         DataInputStream dis = new DataInputStream(socket.getInputStream());
         String clientMessage = "";
         while (!clientMessage.equals("exit")) {
            clientMessage = dis.readUTF();
            System.out.println("Client: " + clientMessage);
         }
         dis.close();
         socket.close();
         serverSocket.close();
         System.out.println("Server closed.");
      } catch (Exception e) {
         System.out.println("Error: " + e.getMessage());
```

```
        }
    }
}
```

Client:

```
import java.io.*;

import java.net.*;
public class Client {
    public static void main(String[] args) {
        try {
            Socket socket = new Socket("localhost", 6666);
            System.out.println("Connected to server.");
            DataOutputStream dos = new DataOutputStream(socket.getOutputStream());
            BufferedReader clientReader = new BufferedReader(new InputStreamReader(System.in));
            String messageToSend = "";
            while (!messageToSend.equals("exit")) {
                System.out.print("Enter message: ");
                messageToSend = clientReader.readLine();
                dos.writeUTF(messageToSend);
                dos.flush();
            }
            dos.close();
            socket.close();
            System.out.println("Client closed.");
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

Output:



```
Server started. Waiting for clients...
Client connected.
Enter message for client: Hi
Enter message for client: Hello
Enter message for client: Exit
Closing connection with this client...
```

```
Connected to the server.
Server: Hi
Server: Hello
Server: Exit
Exiting client...
PS C:\Users\geeth\OneDrive\Desktop\Sem-4\NP-Lab>
```

Result:

Thus the program has been done successfully

Ex: 04 **Two Way Communication Using TCP**

Date:

Aim:

To implement Two Way Communication using TCP

Algorithm:

1. Start the program
2. Use the function BufferedReader to read the input
3. Use the function DataInputStream to read the information from the client
4. Use the function DataOutputStreat to write the information
5. Run the program
6. Stop the program

Code:

Server:

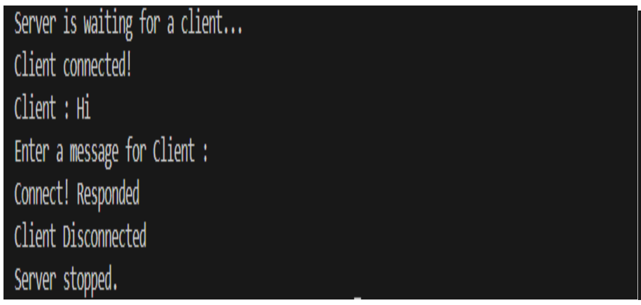
```
import java. io. *;
import java. net. *;
public class MyServer {
    public static void main(String[] args) {
        try {
            ServerSocket ss = new ServerSocket(6663);
            System.out.println("Server is waiting for client...");
            Socket s = ss.accept();
            System.out.println("Client connected!");
            DataOutputStream dos = new DataOutputStream(s.getOutputStream());
            DataInputStream dis = new DataInputStream(s.getInputStream());
            BufferedReader serverReader = new BufferedReader(new InputStreamReader(System.in));
            Thread receiveThread = new Thread(() -> {
                try {
                    String clientMessage;
                    while (true) {

                        clientMessage = dis.readUTF();
                        if (clientMessage.equals("exit")) break;
                        System.out.println("Client: " + clientMessage);

                    }
                } catch (IOException e) {
                    System.out.println("Client disconnected");
                }
            });
            receiveThread.start();
            String serverMessage = "";
            while (!serverMessage.equals("exit")) {
              System.out.print("-");
                serverMessage = serverReader.readLine();
                dos.writeUTF(serverMessage);
                dos.flush();
            }
            dos.close();
            dis.close();
            s.close();
            ss.close();

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

Client:

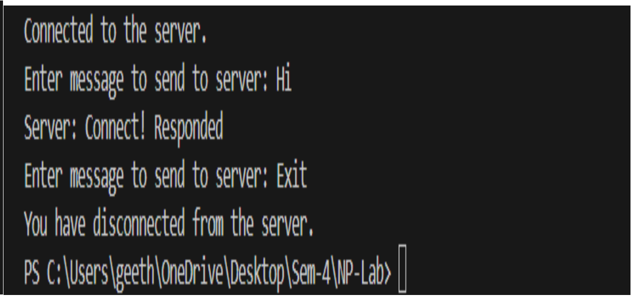
```
import java. io. *;
import java. net. *;
public class MyClient {
    public static void main(String[] args) {
        try {
            Socket s = new Socket("localhost", 6663);
            DataOutputStream dos = new DataOutputStream(s.getOutputStream());
            DataInputStream dis = new DataInputStream(s.getInputStream());
            BufferedReader clientReader = new BufferedReader(new InputStreamReader(System.in));
            Thread receiveThread = new Thread(() -> {
                try {
                    String serverMessage;
                    while (true) {

                        serverMessage = dis.readUTF();
                        if (serverMessage.equals("exit")) break;
                        System.out.println("Server: " + serverMessage);

                    }
                } catch (IOException e) {
                    System.out.println("Disconnected from server");
                }
            });
            receiveThread.start();
            String clientMessage = "";

            while (!clientMessage.equals("exit")) {
                System.out.print("-");
                clientMessage = clientReader.readLine();
```

```
                    dos.writeUTF(clientMessage);
                    dos.flush();
                }
                dos.close();
                dis.close();
                s.close();
            } catch (Exception e) {
                System.out.println(e);
            }
        }
    }
```

Output:



```
Server is waiting for a client...
Client connected!
Client : Hi
Enter a message for Client :
Connect! Responded
Client Disconnected
Server stopped.
```



```
Connected to the server.
Enter message to send to server: Hi
Server: Connect! Responded
Enter message to send to server: Exit
You have disconnected from the server.
PS C:\Users\geeth\OneDrive\Desktop\Sem-4\NP-Lab>
```

Result:

Thus the program has been done successfully

Ex: 05 **One-Way Communication Using UDP**

Date:

Aim:

To implement the UDP One-Way Communication

Algorithm:

1. Start the program
2. Import all the packages for working on UDP
3. For implementing the UDP communication use the DatagramSocket and DatagraPacket
4. DatagramPacket and DatagramSocket is used for Client and Server respectively
5. Run the program
6. Stop the program

Code:

Client:

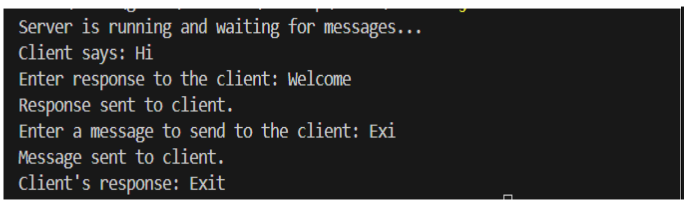
```
import java.io.*;
import java.net.*;
class MyClient {
   public static void main(String args[]) throws Exception {
      DatagramSocket clientSocket = new DatagramSocket();
      byte[] sendData = new byte[500];
      BufferedReader userInput = new BufferedReader(new InputStreamReader(System.in));
      System.out.println("Enter messages to send to the server (type 'exit' to quit):");
      while (true) {
         String message = userInput.readLine();
         if (message.equalsIgnoreCase("exit")) {
            System.out.println("Client exiting...");
            break;
         }
         sendData = message.getBytes();
         InetAddress serverAddress = InetAddress.getByName("localhost");
         DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length,
serverAddress, 9000);
         clientSocket.send(sendPacket);
      }
      clientSocket.close();
   }
}
```

Server:

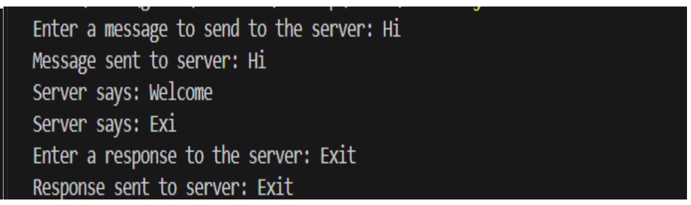
```
import java.io.*;
import java.net.*;
```

```
class MyServer {

   public static void main(String args[]) throws Exception {
      DatagramSocket serverSocket = new DatagramSocket(9000);
      byte[] receiveData = new byte[500];
      System.out.println("Server is running and waiting for messages...");
      while (true) {
         DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);
         serverSocket.receive(receivePacket);
         String message = new String(receivePacket.getData(), 0, receivePacket.getLength());
         System.out.println("Received from client: " + message);
      }
   }
}
```

Output:



```
Server is running and waiting for messages...
Client says: Hi
Enter response to the client: Welcome
Response sent to client.
Enter a message to send to the client: Exi
Message sent to client.
Client's response: Exit
```



```
Enter a message to send to the server: Hi
Message sent to server: Hi
Server says: Welcome
Server says: Exi
Enter a response to the server: Exit
Response sent to server: Exit
```

Result:

Thus the program has been done successfully

Ex: 06 **Two-way Communication using UDP**

Date:

Aim:

To implement the UDP Two-way communication

Algorithm:

1. Start the program
2. Use Datagram as it is for UDP communication
3. Use DatagramPacket and DatagramSocket for communication creation in two-way
4. Run the program
5. Stop the program

Code:

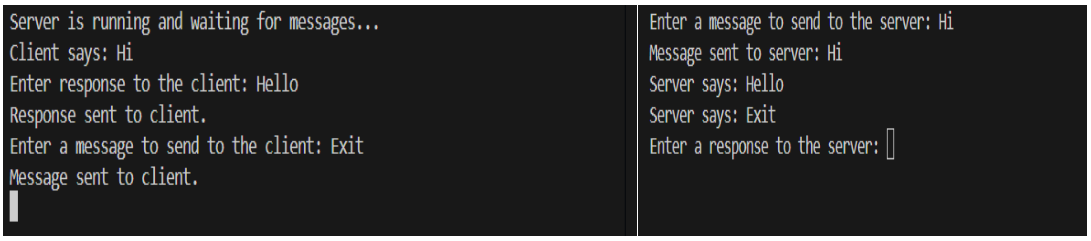
Client:

```
import java. io. *;
import java. net. *;
public class MyClient {
    public static void main(String[] args) {
        try {
            Socket s = new Socket("localhost", 6663);
            DataOutputStream dos = new DataOutputStream(s.getOutputStream());
            DataInputStream dis = new DataInputStream(s.getInputStream());
            BufferedReader clientReader = new BufferedReader(new InputStreamReader(System.in));
            Thread receiveThread = new Thread(() -> {
                try {
                    String serverMessage;
                    while (true) {
                        serverMessage = dis.readUTF();
                        if (serverMessage.equals("exit")) break;
                        System.out.println("Server: " + serverMessage);
                    }
                } catch (IOException e) {
                    System.out.println("Disconnected from server");
                }
            });
            receiveThread.start();
            String clientMessage = "";
            while (!clientMessage.equals("exit")) {
                System.out.print("-");
                clientMessage = clientReader.readLine();
                dos.writeUTF(clientMessage);
                dos.flush();
            }
            dos.close();
            dis.close();
            s.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

Server:

```
import java. io. *;

import java. net. *;
public class MyServer {
    public static void main(String[] args) {
        try {
            ServerSocket ss = new ServerSocket(6663);
            System.out.println("Server is waiting for client...");
            Socket s = ss.accept();
            System.out.println("Client connected!");
            DataOutputStream dos = new DataOutputStream(s.getOutputStream());
            DataInputStream dis = new DataInputStream(s.getInputStream());
            BufferedReader serverReader = new BufferedReader(new InputStreamReader(System.in));
            Thread receiveThread = new Thread(() -> {
                try {
                    String clientMessage;
                    while (true) {
                        clientMessage = dis.readUTF();
                        if (clientMessage.equals("exit")) break;
                        System.out.println("Client: " + clientMessage);

                    }
                } catch (IOException e) {
                    System.out.println("Client disconnected");
                }
            });
            receiveThread.start();
            String serverMessage = "";
            while (!serverMessage.equals("exit")) {
              System.out.print("-");
                serverMessage = serverReader.readLine();
                dos.writeUTF(serverMessage);
                dos.flush();
            }
            dos.close();
            dis.close();
            s.close();
            ss.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

Output:



```
Server is running and waiting for messages...
Client says: Hi
Enter response to the client: Hello
Response sent to client.
Enter a message to send to the client: Exit
Message sent to client.
```

```
Enter a message to send to the server: Hi
Message sent to server: Hi
Server says: Hello
Server says: Exit
Enter a response to the server:
```

Result:

Thus the program has been done successfully

Ex: 07 **Arithmetic Calculator**

Date:

Aim:

To implement the Arithmetic calculator using TCP

Algorithm:

1. Start the program
2. Using TCP the Arithmetic calculator is built
3. Input is given through Client and the Calculation is done at the Server
4. After the server calculates the value the output is displayed in the client
5. Stop the program

Code:

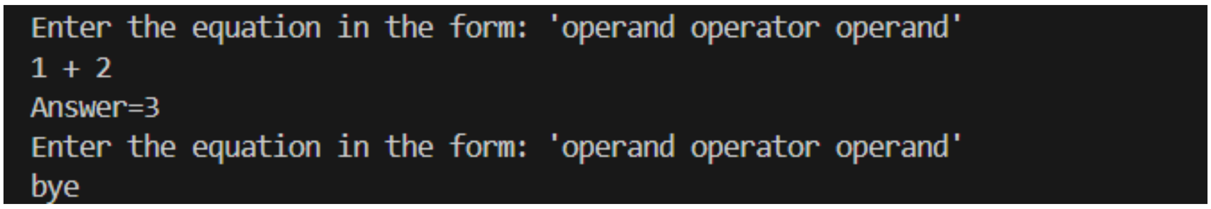
Client:

```
import java.io.DataInputStream;

import java.io.DataOutputStream;
import java.io.IOException;
import java.net.InetAddress;
import java.net.Socket;
import java.util.Scanner;
public class Calc_Client {
    public static void main(String[] args) throws IOException {
        InetAddress ip = InetAddress.getLocalHost();
        int port = 4444;
        Scanner sc = new Scanner(System.in);
        Socket s = new Socket(ip, port);
        DataInputStream dis = new DataInputStream(s.getInputStream());
        DataOutputStream dos = new DataOutputStream(s.getOutputStream());
        while (true) {
            System.out.print("Enter the equation in the form: ");
            System.out.println("'operand operator operand'");
            String inp = sc.nextLine();
            if (inp.equals("bye"))
                break;
            dos.writeUTF(inp);
            String ans = dis.readUTF();
            System.out.println("Answer=" + ans);
        }
    }
}
```

Server:

```java
import java.io.DataInputStream;

import java.io.DataOutputStream;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.StringTokenizer;
public class Calc_Server {
    public static void main(String args[]) throws IOException {
        ServerSocket ss = new ServerSocket(4444);
        Socket s = ss.accept();
        DataInputStream dis = new DataInputStream(s.getInputStream());
        DataOutputStream dos = new DataOutputStream(s.getOutputStream());
        while (true) {
            String input = dis.readUTF();
            if (input.equals("bye"))
                break;
            System.out.println("Equation received:-" + input);
            int result;
            StringTokenizer st = new StringTokenizer(input);
            int oprnd1 = Integer.parseInt(st.nextToken());
            String operation = st.nextToken();
            int oprnd2 = Integer.parseInt(st.nextToken());
            if (operation.equals("+")) {
                result = oprnd1 + oprnd2;
            }
            else if (operation.equals("-")) {
                result = oprnd1 - oprnd2;
            } else if (operation.equals("*")) {
                result = oprnd1 * oprnd2;
            } else {
                result = oprnd1 / oprnd2;
            }
            System.out.println("Sending the result...");
            dos.writeUTF(Integer.toString(result));
        }
    }
}
```

Output:



```
Enter the equation in the form: 'operand operator operand'
1 + 2
Answer=3
Enter the equation in the form: 'operand operator operand'
bye
```

Result:

Thus the program has been done successfully

Ex: 08 **Tic Tac Toe game Using UDP**

Date:

Aim:

To implement the game Tic Tac Toe using the UDP

Algorithm:

1. Start the program
2. Use UDP datagrampacket and datagramsocket for establishing the communication
3. Set the server to play as O and the client to play as X
4. Run the program and get the output
5. Stop the program

Code:

Client:

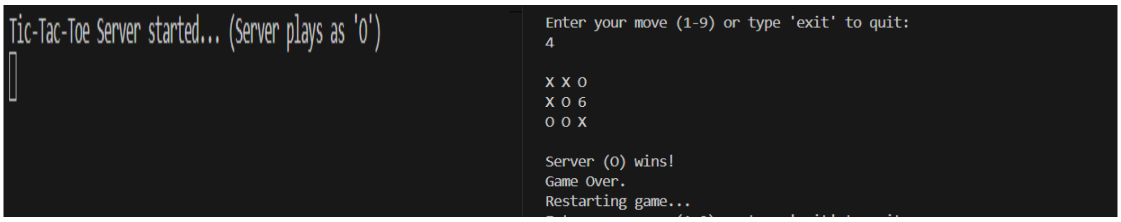
```
import java.net.*;
import java.io.*;
public class TicTacToeUDPClient {
    public static void main(String[] args) {
        try {
            DatagramSocket socket = new DatagramSocket();
            InetAddress serverAddress = InetAddress.getByName("localhost");
            BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
            while (true) {
                System.out.print("Enter position (1-9) or 'exit' to quit: ");
                String move = reader.readLine();
                byte[] buffer = move.getBytes();
                DatagramPacket packet = new DatagramPacket(buffer, buffer.length, serverAddress,
5000);
                socket.send(packet);
                if (move.equalsIgnoreCase("exit")) {
                    System.out.println("Exiting game...");
                    break;
                }
                byte[] responseBuffer = new byte[1024];
                DatagramPacket responsePacket = new DatagramPacket(responseBuffer,
responseBuffer.length);
                socket.receive(responsePacket);
                String response = new String(responsePacket.getData(), 0, responsePacket.getLength());
                System.out.println(response);
                if (response.contains("wins") || response.contains("draw")) {
                    System.out.println("Game Over!");
                    break;
                }
            }
            socket.close();
        } catch (Exception e) {
            e.printStackTrace();
            }
        }
    }
```

Server:

```
import java.net.*;
import java.io.*;
public class TicTacToeUDPServer {
    private static char[][] board = {
            { '1', '2', '3' },
            { '4', '5', '6' },
            { '7', '8', '9' }
    };
    private static DatagramSocket socket;
    public static void main(String[] args) {
        try {
            socket = new DatagramSocket(5000);
            System.out.println("Tic-Tac-Toe AI Server started...");
            while (true) {
                byte[] buffer = new byte[1024];
                DatagramPacket packet = new DatagramPacket(buffer, buffer.length);
                socket.receive(packet);
                String move = new String(packet.getData(), 0, packet.getLength()).trim();
                InetAddress clientAddress = packet.getAddress();
                int clientPort = packet.getPort();
                if (move.equalsIgnoreCase("exit")) {
                    System.out.println("Game over.");
                    socket.close();
                    break;
                }
                String response = processMove(move);
                byte[] responseBytes = response.getBytes();
                DatagramPacket responsePacket = new DatagramPacket(responseBytes,
responseBytes.length, clientAddress,
                        clientPort);
                socket.send(responsePacket);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    private static String processMove(String move) {
        int pos;
        try {
            pos = Integer.parseInt(move);
            if (pos < 1 || pos > 9)
                return "Invalid move! Choose 1-9.";
        } catch (Exception e) {
            return "Invalid input!";
        }
        int row = (pos - 1) / 3;
        int col = (pos - 1) % 3;
        if (board[row][col] == 'X' || board[row][col] == 'O') {
            return "Spot already taken! Try again.";
```

```java
        }
        board[row][col] = 'X'; // Player move
        String status = checkWinner();
        if (!status.isEmpty())
            return displayBoard() + status; // If player wins
        makeAIMove(); // AI move
        status = checkWinner();
        return displayBoard() + (status.isEmpty() ? "Your turn!" : status);
    }
    private static void makeAIMove() {
        int[] bestMove = minimax(board, true);
        board[bestMove[1]][bestMove[2]] = 'O'; // AI plays 'O'
    }
    private static int[] minimax(char[][] board, boolean isAI) {
        int bestScore = isAI ? Integer.MIN_VALUE : Integer.MAX_VALUE;
        int row = -1, col = -1;
        if (!checkWinner().isEmpty())
            return new int[] { evaluateBoard(), -1, -1 };
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                if (board[i][j] != 'X' && board[i][j] != 'O') {
                    char temp = board[i][j];
                    board[i][j] = isAI ? 'O' : 'X';
                    int score = minimax(board, !isAI)[0];
                    board[i][j] = temp;
                    if (isAI && score > bestScore) {
                        bestScore = score;
                        row = i;
                        col = j;
                    } else if (!isAI && score < bestScore) {
                        bestScore = score;
                        row = i;
                        col = j;
                    }
                }
            }
        }
        return new int[] { bestScore, row, col };
    }
    private static int evaluateBoard() {
        String status = checkWinner();
        if (status.contains("Player X wins"))
            return -10;
        if (status.contains("Player O wins"))
            return 10;
        return 0;
    }
    private static String checkWinner() {
        String[] lines = {
                "" + board[0][0] + board[0][1] + board[0][2],
                "" + board[1][0] + board[1][1] + board[1][2],
                "" + board[2][0] + board[2][1] + board[2][2],
                "" + board[0][0] + board[1][0] + board[2][0],
                "" + board[0][1] + board[1][1] + board[2][1],
            "" + board[0][2] + board[1][2] + board[2][2],
            "" + board[0][0] + board[1][1] + board[2][2],
            "" + board[0][2] + board[1][1] + board[2][0]
        };
        for (String line : lines) {
            if (line.equals("XXX"))
                return "Player X wins!";
            if (line.equals("OOO"))
                return "Player O wins!";
        }
        for (char[] row : board) {
            for (char cell : row) {
                if (Character.isDigit(cell))
                    return "";
            }
        }
        return "It's a draw!";
    }
    private static String displayBoard() {
        return "\n " + board[0][0] + " | " + board[0][1] + " | " + board[0][2] +
            "\n---|---|---" +
            "\n " + board[1][0] + " | " + board[1][1] + " | " + board[1][2] +
            "\n---|---|---" +
            "\n " + board[2][0] + " | " + board[2][1] + " | " + board[2][2] + "\n";
    }
}
```

Output:



```
Tic-Tac-Toe Server started... (Server plays as 'O')
```

```
Enter your move (1-9) or type 'exit' to quit:
4

X X O
X O 6
O O X

Server (O) wins!
Game Over.
Restarting game...
```

Result:

Thus the program has been done successfully

Ex: 09 **Payroll Calculation**

Date:

Aim:

To implement the payroll calculation

Algorithm:

1. Start the program
2. In the client side get the input for name of the employee, basic salary, hra, da and pf
3. Enter the formula in the Server side to calculate the net pay of the employee
4. Run the program
5. Stop the program

Code:

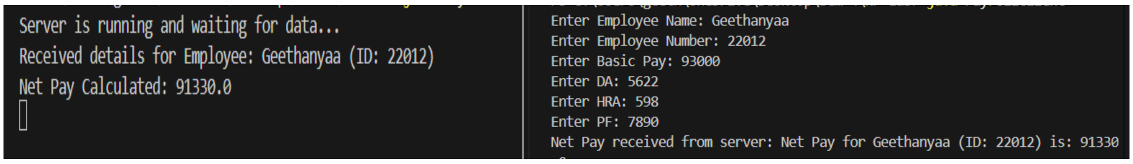
Server:

```
import java.net.*;
import java.io.*;
public class PayrollServer {
    public static void main(String[] args) {
        try {
            DatagramSocket serverSocket = new DatagramSocket(9876);
            byte[] receiveData = new byte[1024];
            byte[] sendData;
            System.out.println("Server is running...");
            while (true) {
                DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
                serverSocket.receive(receivePacket);
                String receivedString = new String(receivePacket.getData(), 0,
receivePacket.getLength());
                String[] details = receivedString.split(",");
                String empName = details[0];
                int empNo = Integer.parseInt(details[1]);
                double basic = Double.parseDouble(details[2]);
                double da = Double.parseDouble(details[3]);
                double hra = Double.parseDouble(details[4]);
                double pf = Double.parseDouble(details[5]);
                double grossPay = basic + da + hra;
                double netPay = grossPay - pf;
                String response = "Emp No: " + empNo + " | Name: " + empName + " | Gross Pay: " +
grossPay
                        + " | Net Pay: " + netPay;
                sendData = response.getBytes();
                InetAddress clientAddress = receivePacket.getAddress();
                int clientPort = receivePacket.getPort();
                DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length,
clientAddress,
                        clientPort);
                serverSocket.send(sendPacket);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Client:

```
import java.net.*;
import java.io.*;
import java.util.Scanner;
public class PayrollClient {
    public static void main(String[] args) {
        try {
            DatagramSocket clientSocket = new DatagramSocket();
            InetAddress serverAddress = InetAddress.getByName("localhost");
            byte[] sendData;
            byte[] receiveData = new byte[1024];
            Scanner scanner = new Scanner(System.in);
            System.out.println("Enter Employee Name: ");
            String empName = scanner.nextLine();
            System.out.println("Enter DA: ");
            double da = scanner.nextDouble();
            System.out.println("Enter HRA: ");
            double hra = scanner.nextDouble();
            System.out.println("Enter PF: ");
            double pf = scanner.nextDouble();
            String message = empName + "," + empNo + "," + basic + "," + da + "," + hra + "," + pf;
            sendData = message.getBytes();
            DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length,
serverAddress, 9876);
            clientSocket.send(sendPacket);
            DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);
            clientSocket.receive(receivePacket);
            String response = new String(receivePacket.getData(), 0, receivePacket.getLength());
            System.out.println("Response from Server: " + response);
            clientSocket.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Output:



```
Server is running and waiting for data...
Received details for Employee: Geethanyaa (ID: 22012)
Net Pay Calculated: 91330.0
```

```
Enter Employee Name: Geethanyaa
Enter Employee Number: 22012
Enter Basic Pay: 93000
Enter DA: 5622
Enter HRA: 598
Enter PF: 7890
Net Pay received from server: Net Pay for Geethanyaa (ID: 22012) is: 91330
```

Result:

Thus the program has been done successfully

Ex: 10 **Concurrent Server Implementation**

Date:

Aim:

To implement the concurrent server and client

Algorithm:

1. Start the program
2. The aim of this program is to show the datetime
3. Create server code for DateTime and a client code for the same
4. Run the program
5. Stop the program

Code:

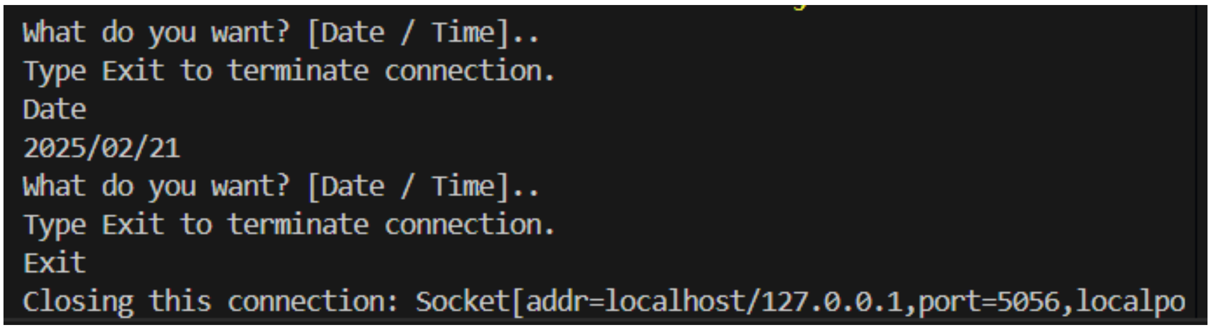
Client:

```
 import java.io.*;

import java.net.*;
import java.util.Scanner;

public class DateTimeClient {
public static void main ( String[ ] args ) throws IOException {
try{
Scanner scn = new Scanner ( System.in ) ;
InetAddress ip = InetAddress.getByName ( &quot;localhost&quot; ) ;
Socket s = new Socket ( ip, 5056 ) ;
DataInputStream dis = new DataInputStream ( s.getInputStream ( ) ) ;
DataOutputStream dos = new DataOutputStream(s.getOutputStream));
while ( true ) {
System.out.println ( dis.readUTF ( ) ) ;
String tosend = scn.nextLine ( ) ;
dos.writeUTF ( tosend ) ;
if ( tosend.equals ( &quot;Exit&quot; ) ) {
System.out.println ( &quot;Closing this connection : &quot; + s ) ;
s.close ( ) ;
System.out.println ( &quot;Connection closed&quot; ) ;

break;
}
String received = dis.readUTF ( ) ;
System.out.println ( received ) ;
}
scn.close ( ) ;
dis.close ( ) ;
dos.close ( ) ;
}catch ( Exception e ) {
e.printStackTrace ( ) ;
}
}
}
```

Sever:

```
import java.io.*;

import java.text.*;
import java.util.*;
import java.net.*;
public class DateTimeServer {
    public static void main(String[] args) throws IOException {
        @SuppressWarnings("resource")
        ServerSocket ss = new ServerSocket(5056);
        while (true) {
            Socket s = null;
            try {
                s = ss.accept();
                System.out.println("A new client is connected : " + s);
                DataInputStream dis = new DataInputStream(s.getInputStream());
                DataOutputStream dos = new DataOutputStream(s.getOutputStream());
                System.out.println("Assigning new thread for this client");
                Thread t = new ClientHandler(s, dis, dos);
                t.start();
            } catch (Exception e) {
                s.close();
                e.printStackTrace();
            }
        }
    }
}

class ClientHandler extends Thread {
    DateFormat fordate = new SimpleDateFormat("yyyy/MM/dd");
    DateFormat fortime = new SimpleDateFormat("hh:mm:ss");
    final DataInputStream dis;
    final DataOutputStream dos;
    final Socket s;

    public ClientHandler(Socket s, DataInputStream dis, DataOutputStream dos) {
        this.s = s;
        this.dis = dis;
        this.dos = dos;
    }

    @Override
    public void run() {
        String received;
        String toreturn;
        while (true) {
            try {
                dos.writeUTF("What do you want?[Date / Time]..\n" + "Type Exit to terminate
connection.");
                received = dis.readUTF();
                if (received.equals("Exit")) {
                    System.out.println("Client " + this.s + " sends exit...");
                    System.out.println("Closing this connection.");
                    this.s.close();
                    System.out.println("Connection closed");
                    break;
                }
                Date date = new Date();
                switch (received) {
                    case "Date":
                        toreturn = fordate.format(date);
                        dos.writeUTF(toreturn);
                        break;
                    case "Time":
                        toreturn = fortime.format(date);
                        dos.writeUTF(toreturn);
                        break;
                    default:
                        dos.writeUTF("Invalid input");
                        break;
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        try {
            this.dis.close();
            this.dos.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Output:



```
What do you want? [Date / Time]..
Type Exit to terminate connection.
Date
2025/02/21
What do you want? [Date / Time]..
Type Exit to terminate connection.
Exit
Closing this connection: Socket[addr=localhost/127.0.0.1,port=5056,localpo
```

Result:

Thus the program has been done successfully