DNS LOOKUP

```
import java.util.*;
import java.net.*;
public class dnslookup {

    public static void main(String[] args){
        String host;
        Scanner ch = new Scanner(System.in);
        System.out.print("1.Enter Host Name \n2.Enter IP address \nChoice=");
        int choice = ch.nextInt();
        if(choice==1)
        {
            Scanner input = new Scanner(System.in);
            System.out.print("\n Enter host name: ");
            host = input.nextLine();
            try {
                InetAddress address = InetAddress.getByName(host);
                System.out.println("IP address: " + address.getHostAddress());
                System.out.println("Host name : " + address.getHostName());
                System.out.println("Host name and IP address: " + address.toString());
            }
            catch (UnknownHostException ex) {
                System.out.println("Could not find " + host);
            }
        }
        else
        {
            Scanner input = new Scanner(System.in);
            System.out.print("\n Enter IP address: ");
            host = input.nextLine();
            try {
                InetAddress address = InetAddress.getByName(host);
                System.out.println("Host name : " + address.getHostName());
                System.out.println("IP address: " + address.getHostAddress());
                System.out.println("Host name and IP address: " + address.toString());

            }
            catch (UnknownHostException ex) {
                System.out.println("Could not find " + host);
            }
        }}}
```

OUTPUT

```
1.Enter Host Name
2.Enter IP address
Choice=1

 Enter host name: www.google.com
IP address: 142.250.67.164
Host name : www.google.com
Host name and IP address: www.google.com/142.250.67.164

Process finished with exit code 0


1.Enter Host Name
2.Enter IP address
Choice=2

 Enter IP address: 205.251.250.0
Host name : server-205-251-250-0.jfk5.r.cloudfront.net
IP address: 205.251.250.0
Host name and IP address: server-205-251-250-0.jfk5.r.cloudfront.net/205.251.250.0
```

SOCKET PROGRAMMING 2 WAY:-

SERVER SIDE

```
import java.awt.image.DataBuffer;
import java.io.*;
import java.net.*;
public class serverside2way {
public static void main(String [] args) throws Exception{
   ServerSocket serverobject=new ServerSocket(6666); //creating server
   Socket server=serverobject.accept(); //establishing connection
   DataOutputStream ds=new DataOutputStream(server.getOutputStream()); //response
   DataInputStream dis=new DataInputStream(server.getInputStream()); // request
   BufferedReader br=new BufferedReader(new InputStreamReader(System.in)); //combine

   String inp="",outp="";
   while(!inp.equals("stop")){
      inp=dis.readUTF();
      System.out.println("Client says:"+inp);
      outp=br.readLine();
      ds.writeUTF(outp);
      System.out.println("Server says:"+outp);
      ds.flush();
   }
   dis.close();
   server.close();
   serverobject.close();

}
}
```

CLIENT SIDE

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

public class clientside2way {
public static void main(String [] args) throws Exception{
   Socket client=new Socket("localhost",6666);
   DataInputStream dis=new DataInputStream(client.getInputStream());
   DataOutputStream ds=new DataOutputStream(client.getOutputStream());
```

```
    BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
    String inp="",outp="";
    while(!inp.equals("stop")){
        inp=br.readLine();
        ds.writeUTF(outp);
        ds.flush();
        outp=dis.readUTF();
        System.out.println("Server says "+outp);

    }
    dis.close();
    client.close();
}
}
```

OUTPUT

SERVER SIDE

```
Client says:
hi
Server says:hi
Client says:hi
hello
Server says:hello
Client says:hello
server bye
Server says:server bye
stop
Client says:server bye
Server says:stop
```

CLIENT SIDE

```
hi
hello
Server says hi
Server says hello
client bye
Server says server bye
stop
Server says stop
```

FILE TRANSFER

Server side

```
import java.io.*;
import java.net.*;
public class filetransfer {
    public static void main(String[] args) throws Exception{
        ServerSocket sock=new ServerSocket(6666); // create server socket
        Socket s=sock.accept(); //create / establish connection with the client
        File file=new File("C:\\Users\\Prathamesh
Patil\\IdeaProjects\\Firstjavaproject\\src\\chimes-7035.mp3"); //create instance of file
        FileInputStream fis=new FileInputStream(file);//store the instance of file in input
stream
        BufferedInputStream bis=new BufferedInputStream(fis); //store the input of input
stream in buffer memory
        OutputStream os=s.getOutputStream();
        byte[] contents;
        long fileLength = file.length();
        long current = 0;
        while(current!=fileLength){
            int size = 10000;
            if(fileLength - current >= size)
                current += size;
            else{
                size = (int)(fileLength - current);
                current = fileLength;
            }
            contents = new byte[size];
            bis.read(contents, 0, size);
            os.write(contents);
            System.out.println("Sending file ... "+(current*100)/fileLength+"% complete!");
        }
        os.flush();
        s.close();
        sock.close();

    }
}
```

CLIENT SIDE

```
import java.io.*;
import java.net.*;
public class filetransferclient {
public static void main(String [] args) throws Exception{
    Socket s=new Socket("localhost",6666);
    byte[] contents = new byte[10000];
    //Initialize the FileOutputStream to the output file's full path.
    FileOutputStream fos = new FileOutputStream("C:\\Users\\Prathamesh
Patil\\IdeaProjects\\Firstjavaproject\\src\\chimes-7038.mp3");
    BufferedOutputStream bos = new BufferedOutputStream(fos);
    InputStream is = s.getInputStream();
    //No of bytes read in one read() call
    int bytesRead = 0;
    while((bytesRead=is.read(contents))!=-1)
        bos.write(contents, 0, bytesRead);
    bos.flush();
    s.close();
    System.out.println("File saved successfully!");
}
}
```

OUTPUT

```
Sending file ... 2% complete!
Sending file ... 4% complete!
Sending file ... 7% complete!
Sending file ... 9% complete!
Sending file ... 11% complete!
Sending file ... 14% complete!
Sending file ... 16% complete!
Sending file ... 18% complete!
Sending file ... 21% complete!
Sending file ... 23% complete!
Sending file ... 25% complete!
Sending file ... 28% complete!
Sending file ... 30% complete!
Sending file ... 32% complete!
Sending file ... 35% complete!
Sending file ... 37% complete!
Sending file ... 39% complete!
```

Sending file ... 42% complete!
Sending file ... 44% complete!
Sending file ... 46% complete!
Sending file ... 49% complete!
Sending file ... 51% complete!
Sending file ... 53% complete!
Sending file ... 56% complete!
Sending file ... 58% complete!
Sending file ... 60% complete!
Sending file ... 63% complete!
Sending file ... 65% complete!
Sending file ... 68% complete!
Sending file ... 70% complete!
Sending file ... 72% complete!
Sending file ... 75% complete!
Sending file ... 77% complete!
Sending file ... 79% complete!
Sending file ... 82% complete!
Sending file ... 84% complete!
Sending file ... 86% complete!
Sending file ... 89% complete!
Sending file ... 91% complete!
Sending file ... 93% complete!
Sending file ... 96% complete!
Sending file ... 98% complete!
Sending file ... 100% complete!

Process finished with exit code 0

SOCKET PROGRAMMING :- 1 way communication

TCP

SERVER SIDE:-

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

public class ABC {
    public static void main(String[] args) {
        try {
            ServerSocket ss = new ServerSocket(6666);
            Socket s=ss.accept();
            System.out.println("Server Started");
            DataInputStream di=new DataInputStream(s.getInputStream());
            String str=di.readUTF();
            System.out.println("Message from client "+str);
            ss.close();
        }
        catch (Exception e){
            System.out.println(e);
        }
    }
}
```

CLIENT SIDE:-

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

public class PQR {
    public static void main(String[] args){
        try {
            Socket s = new Socket("localhost", 6666);
            DataOutputStream ds = new DataOutputStream(s.getOutputStream());
            ds.writeUTF("Hi!!! Hello");
            ds.flush();
            ds.close();
            s.close();
        }
```

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

OUTPUT:-

```
Server Started
Message from client Hi!!! Hello
```

UDP

SERVER SIDE:-

```
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.SocketException;
import java.util.*;

public class udpserver {
    public static void main(String [] args) throws SocketException{
        DatagramPacket pack;
        DatagramSocket sock=new DatagramSocket();

        try{
            String time=new Date().toString();
            byte[] b=time.getBytes();
            pack=new DatagramPacket(b,b.length,
InetAddress.getByName("localhost"),7777);
            sock.send(pack);
        }
        catch(Exception e){
            System.out.println(e);
        }
        sock.close();
    }
```

```
}
```

CLIENT SIDE:-

```
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.SocketException;

public class udpclient {
    public static void main(String[] args) throws SocketException {
        DatagramSocket sock=new DatagramSocket(7777);
        DatagramPacket pack;
        byte [] b=new byte[30];
        String data="No data";
        try {
            while (true) {
                pack = new DatagramPacket(b, b.length);
                sock.receive(pack);
                data = new String(pack.getData());
                System.out.println("Server sent " + data);
            }
        }
        catch(Exception e){
            System.out.println(e);
        }
        sock.close();
    }

}
```

OUTPUT:-

```
"C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:C:\Program
Files\JetBrains\IntelliJ IDEA Community Edition
2022.2.2\lib\idea_rt.jar=59801:C:\Program Files\JetBrains\IntelliJ IDEA Community
Edition 2022.2.2\bin" -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8
```

Client side:-

```
Server sent Wed Oct 12 08:40:45 IST 2022
```

Subnetting

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

public class subnetting {
public static void main(String []args) throws Exception{
   Scanner sc= new Scanner(System.in);
   System.out.println("Enter ip address seperated by into different inputs");
   System.out.println("First : ");
   int ip1=sc.nextInt();
   System.out.println("Second : ");
   int ip2=sc.nextInt();
   System.out.println("Third : ");
   int ip3=sc.nextInt();
   System.out.println("Fourth : ");
   int ip4=sc.nextInt();
   System.out.println("Enter CIDR(Classless Inter Domain Routing) value:");
   int cidr=sc.nextInt();

   switch(cidr){
      case 1: System.out.println("Subnet mask is "+"128.0.0.0"); break;
      case 2: System.out.println("Subnet mask is "+"192.0.0.0"); break;
      case 3: System.out.println("Subnet mask is "+"224.0.0.0"); break;
      case 4: System.out.println("Subnet mask is "+"240.0.0.0"); break;
      case 5: System.out.println("Subnet mask is "+"252.0.0.0"); break;
      case 7: System.out.println("Subnet mask is "+"254.0.0.0"); break;
      case 8: System.out.println("Subnet mask is "+"255.0.0.0"); break;
      case 9: System.out.println("Subnet mask is "+"255.128.0.0"); break;
      case 10: System.out.println("Subnet mask is "+"255.192.0.0"); break;
      case 11: System.out.println("Subnet mask is "+"255.224.0.0"); break;
      case 12: System.out.println("Subnet mask is "+"255.240.0.0"); break;
      case 13: System.out.println("Subnet mask is "+"255.248.0.0"); break;
      case 14: System.out.println("Subnet mask is "+"255.252.0.0"); break;
      case 15: System.out.println("Subnet mask is "+"255.254.0.0"); break;
      case 16: System.out.println("Subnet mask is "+"255.255.0.0"); break;
      case 17: System.out.println("Subnet mask is "+"255.255.128.0"); break;
      case 18: System.out.println("Subnet mask is "+"255.255.192.0"); break;
```

```
    case 19: System.out.println("Subnet mask is "+"255.255.224.0"); break;
    case 20: System.out.println("Subnet mask is "+"255.255.240.0"); break;
    case 21: System.out.println("Subnet mask is "+"255.255.248.0"); break;
    case 22: System.out.println("Subnet mask is "+"255.255.252.0"); break;
    case 23: System.out.println("Subnet mask is "+"255.255.254.0"); break;
    case 24: System.out.println("Subnet mask is "+"255.255.255.0"); break;
    case 25: System.out.println("Subnet mask is "+"255.255.255.128"); break;
    case 26: System.out.println("Subnet mask is "+"255.255.255.192"); break;
    case 27: System.out.println("Subnet mask is "+"255.255.255.224"); break;
    case 28: System.out.println("Subnet mask is "+"255.255.255.240"); break;
    case 29: System.out.println("Subnet mask is "+"255.255.255.248"); break;
    case 30: System.out.println("Subnet mask is "+"255.255.255.252"); break;
    case 31: System.out.println("Subnet mask is "+"255.255.255.254"); break;
    case 32: System.out.println("Subnet mask is "+"255.255.255.255"); break;

    default: System.out.println("Invalid value");
}
String ips1=Integer.toString(ip1);
String ips2=Integer.toString(ip2);
String ips3=Integer.toString(ip3);
if(ip1>=1 && ip1<=127){
    System.out.println("Network class is A");
    System.out.println("Network Address is "+ips1+"0.0.0");

}
else if(ip1>=128 && ip1<=191){
    System.out.println("Network class is B");
    System.out.println("Network Address is "+ips1+"."+ips2+"0.0");
}
else if(ip1>=192 && ip1<=223){
    System.out.println("Network class is C");
    System.out.println("Network Address is "+ips1+"."+ips2+"."+ips3+".0");

}
else if(ip1>=224 && ip1<=255){
    System.out.println("Network class is D and E");
}
System.out.println("Broadcast Address is "+ips1+"."+ips2+"."+ips3+".255");

int noofhostssubnets=32-cidr;
```

```
    int noofhostssubnetsv=(int)Math.pow(2,noofhostssubnets);
    int noofnetworks=noofhostssubnetsv/cidr;
    System.out.println("No of networks are "+noofnetworks);
    System.out.println("No of hosts for ip addressing per subnet1 are
"+(noofhostssubnetsv-2));

}
}
```

OUTPUT

```
Enter ip address seperated by into different inputs
First :
192
Second :
168
Third :
1
Fourth :
0
Enter CIDR(Classless Inter Domain Routing) value:
26
Subnet mask is 255.255.255.192
Network class is C
Network Address is 192.168.1.0
Broadcast Address is 192.168.1.255
No of networks are 2
No of hosts for ip addressing per subnet1 are 62

Process finished with exit code 0
```