IP addressing & Blocks

A MINI PROJECT REPORT

Submitted by

SAHIL BASHIR (200905046)

ADITI ANAND (200905062)

In partial fulfillment for the award of the degree of

B. Tech

IN

Computer Science and Engineering

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
A Constituent Institution of Manipal University

Department of Computer Science and Engineering

Department of Computer Science & Engineering

BONAFIDE CERTIFICATE

Certified that this project report “IP Addressing and Blocks” is the bonafide work of

1. **Sahil Bashir**
2. **Aditi Anand**

who carried out the mini project work under my supervision.

Abstract

Our project deals with IP addressing, where our main goal was to design a tool that takes the IP address along with the subnet mask as input and, then, gives the validity of the IP, along with the first address, last address, and the number of addresses in the block. Keeping the size of this block constant, it should give at least other three blocks available of the same size with their network address.

Acknowledgement

First and foremost, I would like to thank our computer network teacher, Ms. Tanuja S who guided us in doing these projects. She provided us with invaluable knowledge through our regular semester classes.Her motivation and help contributed tremendously to the successful completion of the project.

Besides, we would like to thank all the teachers who helped us by giving us advice and providing the equipment which we needed.

At last but not least, we would like to thank everyone who helped and motivated us to work on this project.

Table of Contents

List of Tables

List of Figures

List of Symbols, Abbreviations and Nomenclature

IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
ipconfig/ ifconfig	Discover IP address of the device
ISP	Internet Service Provider
CIDR	Classless Inter-Domain Routing

Chapter 1

Theory

What is an IP address?

IP (Internet Protocol) addresses are used to identify hardware devices on a network. The addresses allow these devices to connect to one another and transfer data on a local network or over the internet.

Each address is a string of numbers separated by periods. There are four numbers in total and each number can range between 0 and 255. An example of an IP address would be: 506.457.14.512

We need billions of IP addresses to identify every computer, router and website on the internet. One day we'll run out of unique addresses and a new IPv6 protocol has been designed to meet this need.

How Do I Find my IP Address?

If your computer is connected to both your local network and the internet, then it will have two IP addresses. You'll have a private IP address locally, and a public IP address on the internet.

A **private IP address** is used to connect your computer or device to your home or business network. This address is normally assigned by your network router.

Private IP addresses are in the range 40.xxx.xxx.xxx or 192.168.xxx.xxx. An example of a private IP address is 192.168.1.1.

There are a few ways to discover your private IP address. For example, on Windows, you can type *ipconfig* on the command prompt. Similarly, Mac users can type the command *ifconfig* in the Terminal app.

Your **public IP address** is used to connect your home or business network to the internet. This address is assigned by your internet service provider (ISP).

IP address classes

Classful addressing is a network addressing the Internet's architecture from 1981 till Classless Inter-Domain Routing was introduced in 1993.
This addressing method divides the IP address into five separate classes based on four address bits.
Here, classes A, B, C offers addresses for networks of three distinct network sizes. Class D is only used for multicast, and class E is reserved exclusively for experimental purposes.

Class	Higher bits	Network address bits	Host address bits	No. of networks	No.of hosts per network	Range
A	0	8	24	2^7	2^{24}	0.0.0.0 to 125.255.255.255
B	10	16	16	2^{14}	2^{16}	128.0.0.0 to 191.255.255.255
C	110	24	8	2^{21}	2^8	192.0.0.0 to 223.255.255.255
D	1110	Not defined and reserved for future	Not defined and reserved for future	Not defined and reserved for future	Not defined and reserved for future	224.0.0.0 to 239.255.255.255

Table 1.1

Let's see each of the network classes in detail:

Class A Network

This IP address class is used when there are a large number of hosts. In a Class A type of network, the first 8 bits (also called the first octet) identify the network, and the remaining have 24 bits for the host into that network.
An example of a Class A address is 102.168.212.226. Here, “102” helps you identify the network and 168.212.226 identifies the host.
Class A addresses 127.0.0.0 to 127.255.255.255 cannot be used and is reserved for loopback and diagnostic functions.

Class B Network

In a B class IP address, the binary addresses start with 10. In this IP address, the class decimal number can be between 128 to 191. The number 127 is reserved for loopback, which is used for internal testing on the local machine. The first 16 bits (known as two octets) help you identify the network. The other remaining 16 bits indicate the host within the network.
An example of a Class B IP address is 168.212.226.204, where *168 212* identifies the network and *226.204* helps you identify the Hut network host.

Class C Network

Class C is a type of IP address that is used for a small network. In this class, three octets are used to indent the network. This IP ranges between 192 to 223.
In this type of network addressing method, the first two bits are set to be 1, and the third bit is set to 0, which makes the first 24 bits of the address them and the remaining bit the host address. Most local area network used Class C IP address to connect with the network.
Example for a Class C IP address: 192.168.178.1

Class D Network

Class D addresses are only used for multicasting applications. Class D is never used for regular networking operations. This class addresses the first three bits set to “1” and their fourth bit set to use for “0”. Class D addresses are 32-bit network addresses. All the values within the range are used to identify multicast groups uniquely.
Therefore, there is no requirement to extract the host address from the IP address, so Class D does not have any subnet mask.

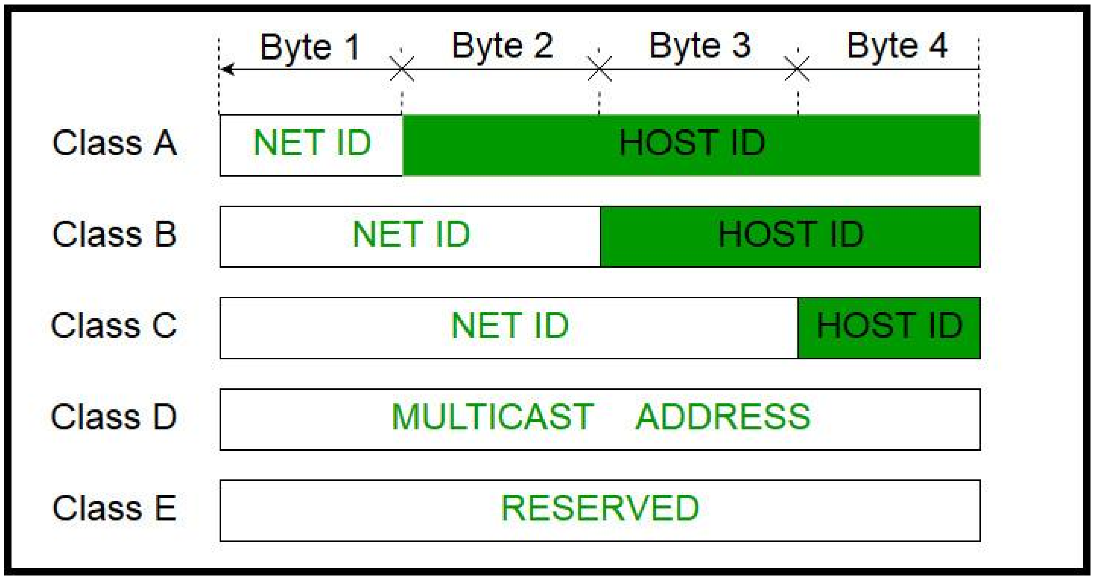


Fig 1.1

Limitations of classful IP addressing

Here are the drawbacks/ cons of the classful IP addressing method:

- Risk of running out of address space soon
- Class boundaries did not encourage efficient allocation of address space

Classless Addressing

To reduce the wastage of IP addresses in a block, we use sub-netting. What we do is that we use host id bits as net id bits of a classful IP address. We give the IP address and define the number of bits for the mask along with it (usually followed by a '/' symbol), like, 192.168.1.1/28. Here, a subnet mask is found by putting the given number of bits out of 32 as 1, like, in the given address, we need to put 28 out of 32 bits as 1 and the rest as 0, and so, the subnet mask would be 255.255.255.240. **Classless Inter-Domain Routing (CIDR)**, eliminates the class model and lets the network designer assign any number of bits to the network prefix, opening up a more extensive choice of the number of hosts per network.

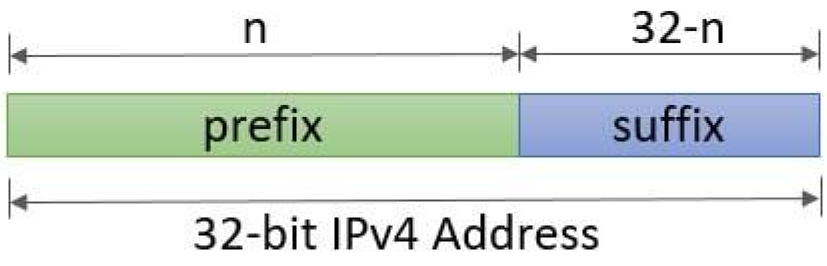


Fig 1.2

What is a subnet?

In an **Internet Protocol** (IP) network, part of the IP address represents the **network**, while the remainder represents the **host** (or **computer** on the network). The act of dividing the IP address is called **subnetting**. Routers use the network part to exchange data between subnets, and the host part to send data to an individual host.

What is a mask?

The network mask is used to determine which portion of the IP address is the network address and which is the host address. This means that the portions of the network to host in an IP address can change. The most common network mask is 255.255.255.0. The simple explanation is that wherever there is a 255, this indicates that it is the network portion. Wherever there is a 0, this indicates the host portion.

Why Use Subnetting?

- A subnet mask allows identification of the host part and network part of an IP address.

- Subnet mask can be used to find if an IP address is present on a subnet or not.

- Subnet mask is utilized for isolating the network id and host ids.

- This is to reduce the broadcast domain or to reduce heavy network traffic.

- A subnet mask helps in separating an IP address into network and host addresses.

- It is a 32-bit number that masks an IP address and divides it into network and host addresses.

Types of Addresses:

Network address

The network IP address is the **first address of the subnet**. A router uses this address to forward traffic to the correct network. **It isn't possible to assign the network address to a host.**

Broadcast address

A host can use the broadcast address to **send data to all the other hosts** on the subnet. It's the **last address** on the subnet.

First and last host addresses

The next address after the network address is the first address available to be assigned to a host. The address just before the broadcast address is the last address that you can allocate to a host.

What are blocks in IP addressing?

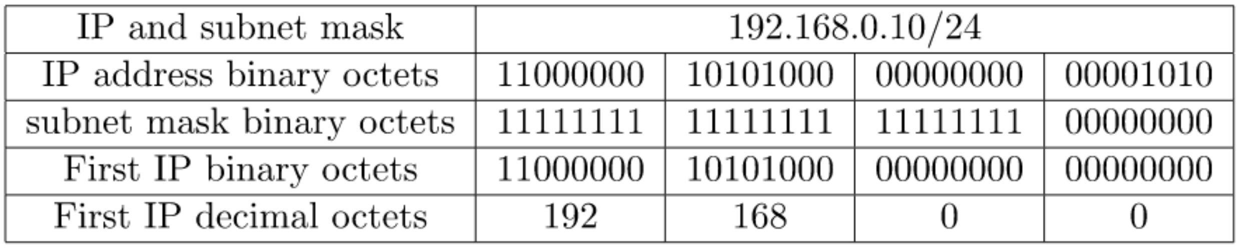
IP addresses are assigned to networks in different sized ‘blocks'. The size of the ‘block' assigned is written after an oblique (/), which shows the number of IP addresses contained in that block.

For example, if an Internet Service Provider (ISP) is assigned a “/16”, they receive around 64,000 IPv4 addresses. A “/26” network provides 64 IPv4 addresses. The lower the number after the oblique, the more addresses contained in that “block”.

Calculating first and last addresses in a block:

First Address:

To find the starting address in the following subnet mask, we simply do binary “and” operation between the IP address and the subnet mask:



IP and subnet mask	192.168.0.10/24			
IP address binary octets	11000000	10101000	00000000	00001010
subnet mask binary octets	11111111	11111111	11111111	00000000
First IP binary octets	11000000	10101000	00000000	00000000
First IP decimal octets	192	168	0	0

Fig 1.3 / Table 1.2

Last Address:

Finally, we calculate the last IP address by applying the “or” operation on it with the bitwise binary inverse of the subnet mask to the first IP address:

IP and subnet mask	192.168.0.10/24			
subnet mask binary octets	11111111	11111111	11111111	00000000
First IP binary octets	11000000	10101000	00000000	00000000

inverse subnet mask binary octets	00000000	00000000	00000000	11111111
Last IP binary octets	11000000	10101000	00000000	11111111
Last IP decimal octets	192	168	0	255

Fig 1.4 / Table 1.3

Chapter 2

The Code

Here is the code for the given question.

Each section will be explained using Comments highlighted in orange.

Note: This code is written in JAVA and uses the java Swing API for the GUI creation.

```
import java.awt.event.*;

import java.util.Scanner;

import java.util.StringTokenizer;

import javax.swing.*;

import java.lang.*;


class text extends JFrame implements ActionListener {

    // JTextField

    static JTextField t,t1;

    // JFrame

    static JFrame f;

    // JButton

    static JButton b;

    // label to display text

    static JLabel l,l1,l2,l3,l4, l5, l6, l7;

    // default constructor

    text()

    {

    }

 //calculate number of addresses in a block

    public static int getNumberofAddress(String msk)
 {
   StringTokenizer t1 = new StringTokenizer(msk, ".");
   int m1 = Integer.parseInt(t1.nextToken());
   int m2 = Integer.parseInt(t1.nextToken());
   int m3 = Integer.parseInt(t1.nextToken());
   int m4 = Integer.parseInt(t1.nextToken());
System.out.println("M4 is "+m4);
   int[] nm1=toBinary(m1);
   int[] nm2=toBinary(m2);
   int[] nm3=toBinary(m3);
   int[] nm4=toBinary(m4);
   int prefix=0;
   int[][] arr=new int[4][8];
   arr[0]=nm1;
   arr[1]=nm2;
   arr[2]=nm3;
   arr[3]=nm4;
   for(int i=0;i<4;i++)
   {
        for(int j=0;j<8;j++)
        {

                  if(arr[i][j]==1)
                  {
```

```
                              System.out.print(arr[i][j]);

                              prefix++;

                    }




          }

          System.out.println();

     }

     int NOA=(int)Math.pow(2,(32-prefix));

     System.out.println(NOA + " " + prefix);


  return NOA;


   }


//check the validity of the ip addresses

   public static boolean validateIpAddress(String ipAddress) {

     boolean b1 = false;


     int ctr=0;


          for(int i=0;i<ipAddress.length();i++)

          {

                    if((ipAddress.charAt(i)=='.'))
                    {
                            ctr++;
                    }
            }

            if(ctr!=3)
            {
                    System.out.println("Entered here");
                    return b1;
            }

        StringTokenizer t = new StringTokenizer(ipAddress, ".");
        int a = Integer.parseInt(t.nextToken());
        int b = Integer.parseInt(t.nextToken());
        int c = Integer.parseInt(t.nextToken());
        int d = Integer.parseInt(t.nextToken());
        if ((a >= 0 && a <= 255) && (b >= 0 && b <= 255) && (c >= 0 && c <= 255) && (d >= 0 && d <=
255))
        {
          b1 = true;
        }
        return b1;
      }

//get the first address in the block of code of the given ip address
```

```
public static String getFirstAddress(String ipAddress, String mask)
{
    StringTokenizer t = new StringTokenizer(ipAddress, ".");
    int i1 = Integer.parseInt(t.nextToken());
    int i2 = Integer.parseInt(t.nextToken());
    int i3 = Integer.parseInt(t.nextToken());
    int i4 = Integer.parseInt(t.nextToken());

    StringTokenizer t1 = new StringTokenizer(mask, ".");
    int m1 = Integer.parseInt(t1.nextToken());
    int m2 = Integer.parseInt(t1.nextToken());
    int m3 = Integer.parseInt(t1.nextToken());
    int m4 = Integer.parseInt(t1.nextToken());

    int[] fai1=toBinary(i1);
    int[] fam1=toBinary(m1);
    int FA1=andIpandMask(fai1,fam1);

    int[] fai2=toBinary(i2);
    int[] fam2=toBinary(m2);
    int FA2=andIpandMask(fai2,fam2);

    int[] fai3=toBinary(i3);
    int[] fam3=toBinary(m3);
```

```
      int FA3=andIpandMask(fai3,fam3);


      int[] fai4=toBinary(i4);

      int[] fam4=toBinary(m4);

      int FA4=andIpandMask(fai4,fam4);

    String str=FA1+"."+FA2+"."+FA3+"."+FA4;


return str;


  }


//get last address of the block of given ip address

  public static String getLastAddress(String ipAddress, String mask)

  {

      StringTokenizer t = new StringTokenizer(ipAddress, ".");

      int i1 = Integer.parseInt(t.nextToken());

      int i2 = Integer.parseInt(t.nextToken());

      int i3 = Integer.parseInt(t.nextToken());

      int i4 = Integer.parseInt(t.nextToken());


      StringTokenizer t1 = new StringTokenizer(mask, ".");

      int m1 = Integer.parseInt(t1.nextToken());

      int m2 = Integer.parseInt(t1.nextToken());

      int m3 = Integer.parseInt(t1.nextToken());
      int m4 = Integer.parseInt(t1.nextToken());


      int[] fai1=toBinary(i1);

      int[] fam1=toBinary(m1);

      int FA1=orIpandMask(fai1,fam1);


      int[] fai2=toBinary(i2);

      int[] fam2=toBinary(m2);

      int FA2=orIpandMask(fai2,fam2);


      int[] fai3=toBinary(i3);

      int[] fam3=toBinary(m3);

      int FA3=orIpandMask(fai3,fam3);


      int[] fai4=toBinary(i4);

      int[] fam4=toBinary(m4);

      int FA4=orIpandMask(fai4,fam4);

      String str=FA1+"."+FA2+"."+FA3+"."+FA4;


      return str;

  }


//generate a random number

  public int getRandomNumber(int min, int max) {
```

```
                return (int) ((Math.random() * (max - min)) + min);

            }


//return a block of address with the same size as that of the given ip address. Return network
//address and range of the block

  public String getRandomAB(String mask)

  {

              int a=getRandomNumber(0,255);

              int b=getRandomNumber(0,255);

              int c=getRandomNumber(0,255);

              int d=getRandomNumber(0,255);


              String str=a+"."+b+"."+c+"."+d;

              String fa = getFirstAddress(str, mask);

              String la = getLastAddress(str, mask);

              String output = "Block" + fa + " to " + la + "\n" + "Network address:"+fa;

              return output;

  }


  // convert from decimal to binary

  public static int[] toBinary(int decimal){

      int binary[] = new int[40];

      int index = 0;

      while(decimal > 0){

        binary[index++] = decimal%2;
        decimal = decimal/2;
      }
     return binary;
  }


//do binary IP and mask and return
  public static int andIpandMask(int[] a,int[] b)
  {
            int n;
            int sum=0,k;
            int[] c=new int[8];
  for(int i=0;i<8;i++)
  {
            c[i]=a[i]&b[i];
             n=i;
             k=(int)Math.pow(2,n);
             sum=sum+(k*c[i]);

  }

  return sum;
  }


//do binary or ip and mask and return
```

```
public static int orIpandMask(int[] a,int[] b)

{

        int n;

        int sum=0,k;

        int vr;

        int[] c=new int[8];

for(int i=0;i<8;i++)

{

        if(b[i]==0)

        {

                vr=1;

        }

        else

        {

                vr=0;


        }


        c[i]=a[i]|vr;

         n=i;

         k=(int)Math.pow(2,n);

         sum=sum+(k*c[i]);


}
```

```
return sum;

}




 // main class

 public static void main(String[] args)

 {



     // create a new frame to store text field and button

     f = new JFrame("textfield");


     // create a label to display text

     l = new JLabel("nothing entered");

     l1 = new JLabel("nothing entered");

     l2 = new JLabel("nothing entered");

     l3 = new JLabel("nothing entered");

     l4= new JLabel("nothing entered");

     l5 = new JLabel("nothing entered");

     l6 = new JLabel("nothing entered");

     l7= new JLabel("nothing entered");
```

```
// create a new button

b = new JButton("CHECK IP");


// create a object of the text class

text te = new text();


// addActionListener to button

b.addActionListener(te);


// create a object of JTextField with 16 columns and a given initial text

t = new JTextField("enter the ip address", 16);

t1 = new JTextField("enter the mask", 16);


// create a panel to add buttons and textfield

JPanel p = new JPanel();


// add buttons and textfield to panel

p.add(t);

 p.add(l);

 p.add(t1);

p.add(l1);

 p.add(l2);
```

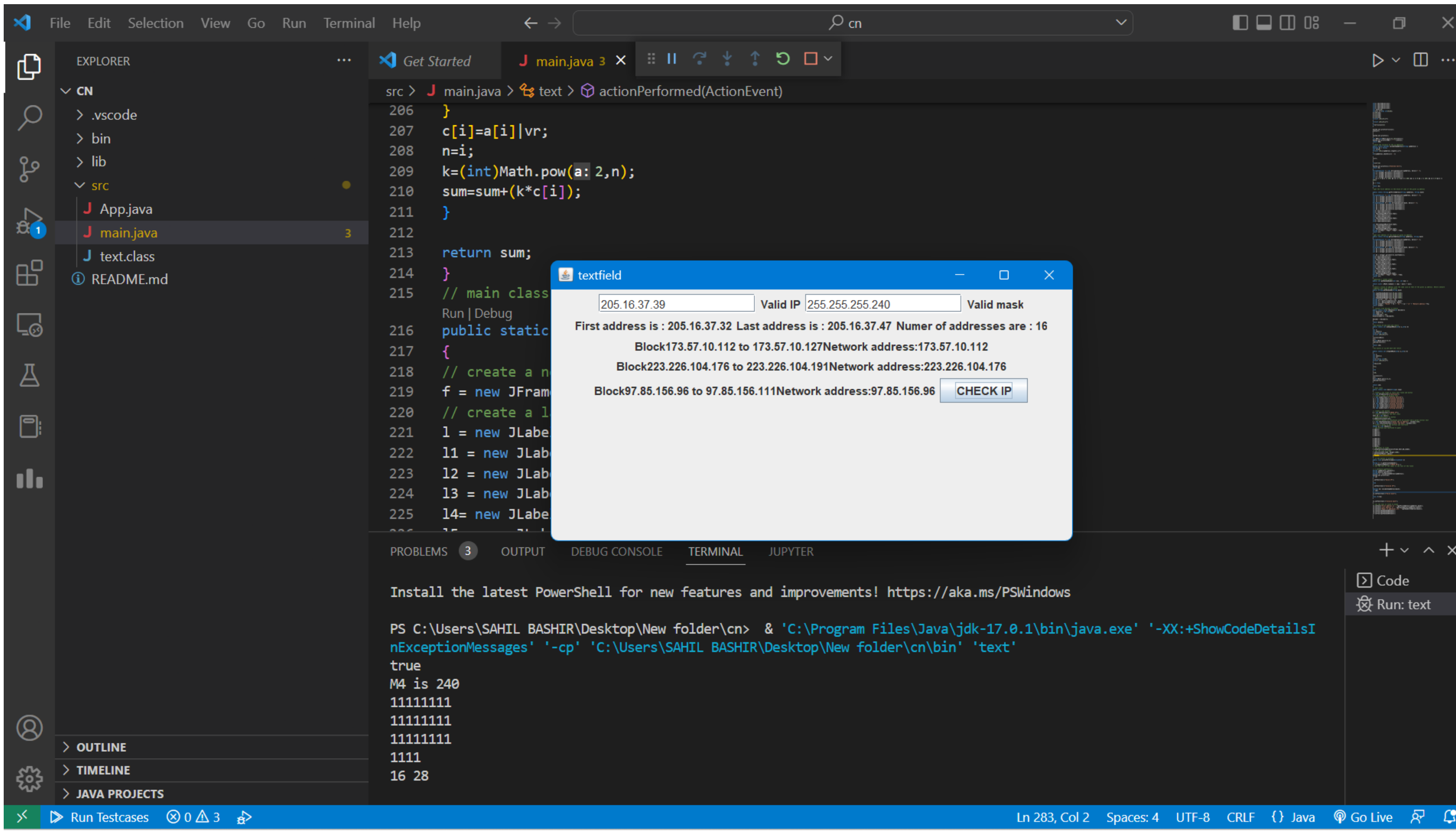
```
    p.add(l3);

   p.add(l4);

   p.add(l5);

   p.add(l6);

   p.add(l7);

    p.add(b);

 // add panel to frame

f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

// set the size of frame

    f.setSize(300, 300);

    f.getContentPane().add(p);


    f.show();




 }


// if the button is pressed

public void actionPerformed(ActionEvent e)

{

    String s = e.getActionCommand();


    if (s.equals("CHECK IP")) {

        // set the text of the label to the text of the field
   // l.setText(t.getText());

    String ipAddress=t.getText();

    String mask=t1.getText();

    boolean b = validateIpAddress(ipAddress);
   System.out.println(b);
    if (b)
    {
      l.setText("Valid IP");
    }
    else
    {
      l.setText("Invalid ip”);
}
    boolean b1= validateIpAddress(mask);
    if (b1)
    {
      l1.setText("Valid mask");
    }
    else if(!b1)
    {
            l1.setText("Invalid mask");

          }

          // set the text of field to blank

        l2.setText("First address is : "+getFirstAddress(ipAddress,mask));


        l3.setText("Last address is : "+getLastAddress(ipAddress,mask);

        l4.setText("Numer of addresses are : "+getNumberofAddress(mask));

         l5.setText(getRandomAB(mask));


         l6.setText(getRandomAB(mask));

          l7.setText(getRandomAB(mask));

      }

   }

}
```

Chapter 3

Testing our code

Input IP Address: 205.16.37.39

Mask: 255.255.255.240

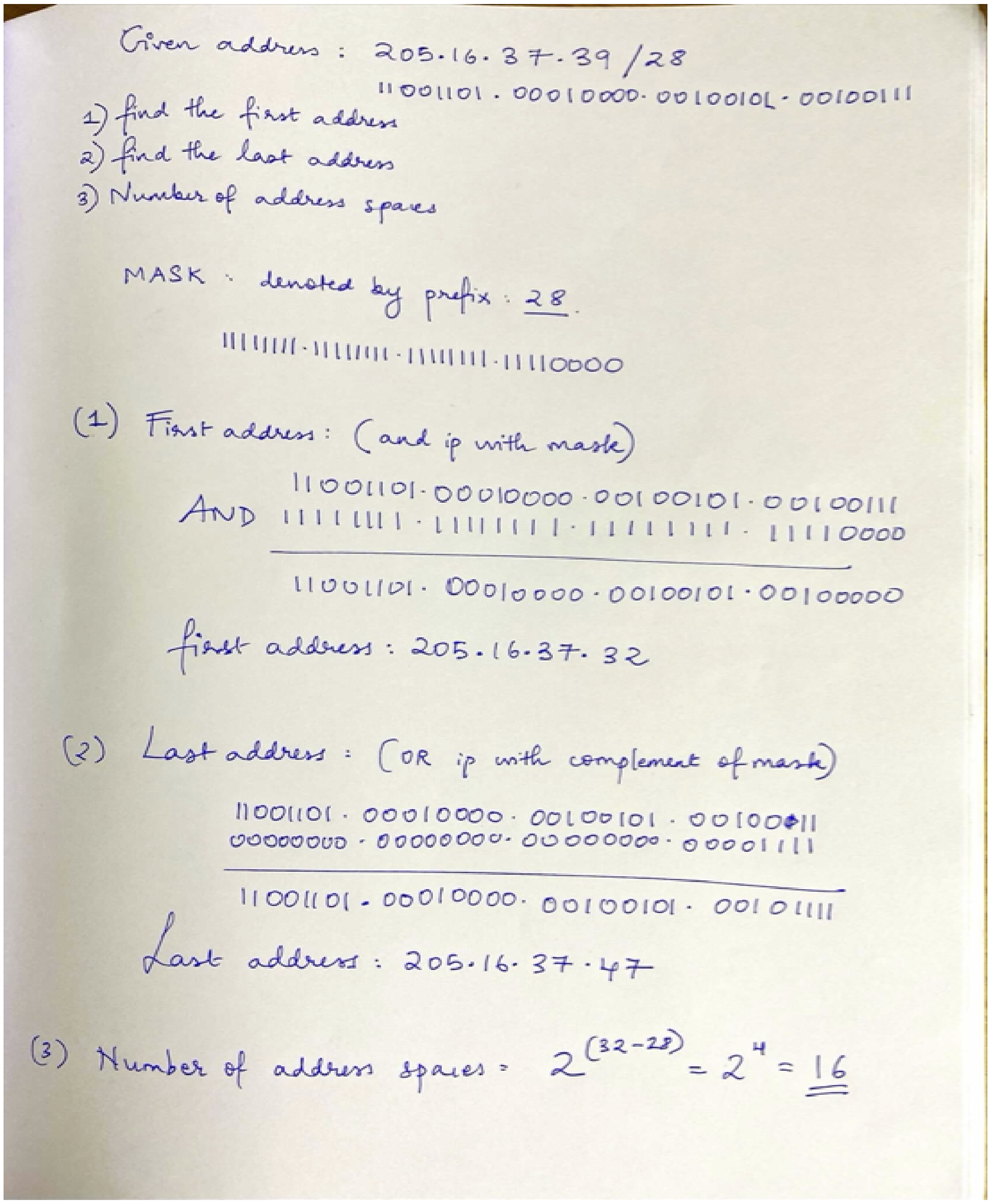


```
}
c[i]=a[i]|vr;
n=i;
k=(int)Math.pow(a: 2,n);
sum=sum+(k*c[i]);
}

return sum;
}
// main class
Run | Debug
public static
{
// create a n
f = new JFram
// create a l
l = new JLabe
l1 = new JLab
l2 = new JLab
l3 = new JLab
l4= new JLabe
```

```
205.16.37.39  Valid IP  255.255.255.240  Valid mask
First address is : 205.16.37.32  Last address is : 205.16.37.47  Numer of addresses are : 16
Block173.57.10.112 to 173.57.10.127Network address:173.57.10.112
Block223.226.104.176 to 223.226.104.191Network address:223.226.104.176
Block97.85.156.96 to 97.85.156.111Network address:97.85.156.96  CHECK IP
```

```
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\SAHIL BASHIR\Desktop\New folder\cn>  & 'C:\Program Files\Java\jdk-17.0.1\bin\java.exe' '-XX:+ShowCodeDetailsI
nExceptionMessages' '-cp' 'C:\Users\SAHIL BASHIR\Desktop\New folder\cn\bin' 'text'
true
M4 is 240
11111111
11111111
11111111
1111
16 28
```

Now calculating the first and last address manually to verify our answer.



Given address : 205.16.37.39 /28

11001101 . 00010000 . 00100101 . 00100111

1) find the first address

2) find the last address

3) Number of address spaces

MASK : denoted by prefix : $\underline{28}$.

11111111 . 11111111 . 11111111 . 11110000

(1) First address : (and ip with mask)

```
     11001101 . 00010000 . 00100101 . 00100111
AND  11111111 . 11111111 . 11111111 . 11110000
     -----------------------------------------
     11001101 . 00010000 . 00100101 . 00100000
```

first address : 205.16.37.32

(2) Last address : (OR ip with complement of mask)

```
11001101 . 00010000 . 00100101 . 00100111
00000000 . 00000000 . 00000000 . 00001111
-----------------------------------------
11001101 . 00010000 . 00100101 . 00101111
```

Last address : 205.16.37.47

(3) Number of address spaces = $2^{(32-28)} = 2^4 = \underline{\underline{16}}$

Hence verified!!

References

1. https://www.youtube.com/watch?v=Tnjdk08z3HM - IP addressing in depth tutorial
2. Computer networking by Jim Kurose
3. https://wiki.teltonika-networks.com/view/What_is_a_Netmask%3F
4. www.wikipedia.org , www.google.com
5. Class lectures - V sem

Thank You :)