*Text

Description automatically generated*

Graphical user interface

Description automatically generated

Karan Chavare

RJ COLLEGE

CLOUD COMPUTING

GENERAL

Table of Contents

[*Java Networking* 2](#_Toc82178263)

[Advantage of Java Networking 2](#_Toc82178264)

[*Java Networking Terminology* 2](#_Toc82178265)

[1) IP Address 2](#_Toc82178266)

[2) Protocol 3](#_Toc82178267)

[3) Port Number 3](#_Toc82178268)

[4) MAC Address 3](#_Toc82178269)

[5) Connection-oriented and connection-less protocol 3](#_Toc82178270)

[6) Socket 3](#_Toc82178271)

[*Java.Net Package* 4](#_Toc82178272)

[*Java Socket Programming* 6](#_Toc82178273)

## *Java Networking*

Java Networking is a concept of connecting two or more computing devices together so that we can share resources.

Java socket programming provides facility to share data between different computing devices.

Advantage of Java Networking

1. Sharing resources
2. Centralize software management

The java.net package supports two protocols,

1. **TCP:** Transmission Control Protocol provides reliable communication between the sender and receiver. TCP is used along with the Internet Protocol referred as TCP/IP.
2. **UDP:** User Datagram Protocol provides a connection-less protocol service by allowing packet of data to be transferred along two or more nodes

## *Java Networking Terminology*

The widely used Java networking terminologies are given below:

1. IP Address
2. Protocol
3. Port Number
4. MAC Address
5. Connection-oriented and connection-less protocol
6. Socket

1) IP Address

IP address is a unique number assigned to a node of a network e.g. 192.168.0.1 . It is composed of octets that range from 0 to 255.

It is a logical address that can be changed.

2) Protocol

A protocol is a set of rules basically that is followed for communication. For example:

* TCP
* FTP
* Telnet
* SMTP
* POP etc.

3) Port Number

The port number is used to uniquely identify different applications. It acts as a communication endpoint between applications.

The port number is associated with the IP address for communication between two applications.

4) MAC Address

MAC (Media Access Control) address is a unique identifier of NIC (Network Interface Controller). A network node can have multiple NIC but each with unique MAC address.

For example, an ethernet card may have a **MAC** address of 00:0d:83::b1:c0:8e.

5) Connection-oriented and connection-less protocol

In connection-oriented protocol, acknowledgement is sent by the receiver. So it is reliable but slow. The example of connection-oriented protocol is TCP.

But, in connection-less protocol, acknowledgement is not sent by the receiver. So it is not reliable but fast. The example of connection-less protocol is UDP.

6) Socket

A socket is an endpoint between two-way communications.

## *java.net package*

The java.net package can be divided into two sections:

1. **A Low-Level API:** It deals with the abstractions of addresses i.e. networking identifiers, Sockets i.e. bidirectional data communication mechanism and Interfaces i.e. network interfaces.
2. **A High-Level API:** It deals with the abstraction of URIs i.e. Universal Resource Identifier, URLs i.e. Universal Resource Locator, and Connections i.e. connections to the resource pointed by URLs.

The java.net package provides many classes to deal with networking applications in Java. A list of these classes is given below:

* Authenticator
* CacheRequest
* CacheResponse
* ContentHandler
* CookieHandler
* CookieManager
* DatagramPacket
* DatagramSocket
* DatagramSocketImpl
* InterfaceAddress
* JarURLConnection
* MulticastSocket
* InetSocketAddress
* InetAddress
* Inet4Address
* Inet6Address
* IDN
* HttpURLConnection
* HttpCookie
* NetPermission
* NetworkInterface
* PasswordAuthentication
* Proxy
* ProxySelector
* ResponseCache
* SecureCacheResponse
* ServerSocket
* Socket
* SocketAddress
* SocketImpl
* SocketPermission
* StandardSocketOptions
* URI
* URL
* URLClassLoader
* URLConnection
* URLDecoder
* URLEncoder
* URLStreamHandler

**List of interfaces available in java.net package:**

* ContentHandlerFactory
* CookiePolicy
* CookieStore
* DatagramSocketImplFactory
* FileNameMap
* SocketOption<T>
* SocketOptions
* SocketImplFactory
* URLStreamHandlerFactory
* ProtocolFamily

## *Java Socket Programming*

Java Socket programming is used for communication between the applications running on different JRE.

Java Socket programming can be connection-oriented or connection-less.

Socket and ServerSocket classes are used for connection-oriented socket programming and DatagramSocket and DatagramPacket classes are used for connection-less socket programming.

The client in socket programming must know two information:

1. IP Address of Server, and
2. Port number.

Here, we are going to make one-way client and server communication. In this application, client sends a message to the server, server reads the message and prints it. Here, two classes are being used: Socket and ServerSocket. The Socket class is used to communicate client and server. Through this class, we can read and write message. The ServerSocket class is used at server-side. The accept() method of ServerSocket class blocks the console until the client is connected. After the successful connection of client, it returns the instance of Socket at server-side.

Diagram

Description automatically generated

*PRACTICAL - 1*

* ***Implementation of following program using TCP/IP protocol***
* Client program to check prime or not.

*Client.java*

import java.net.\*;  
import java.io.\*;  
  
public class Client  
{  
 public static void main(String args[])  
 {  
 try {  
 Socket s = new Socket("LocalHost", 8001);  
 BufferedReader br = new BufferedReader(new InputStreamReader(System.*in*));  
 System.*out*.println("Enter a Number:");  
 int number=Integer.*parseInt*(br.readLine());  
 DataOutputStream out = new DataOutputStream(s.getOutputStream());  
 out.writeInt(number);  
 DataInputStream in = new DataInputStream(s.getInputStream());  
 System.*out*.println(in.readUTF());  
 s.close();  
 }  
 catch (Exception e){  
 System.*out*.println(e.toString());  
 }  
 }  
}

Text

Description automatically generated

*Server.java*

import java.net.\*;  
import java.io.\*;  
  
public class Server {  
 public static void main(String args[]){  
 try  
 {  
 ServerSocket ss = new ServerSocket(8001);  
 System.*out*.println("Server Started...............");  
 Socket s = ss.accept();  
 DataInputStream in = new DataInputStream(s.getInputStream());  
 int x = in.readInt();  
 DataOutputStream otc = new DataOutputStream(s.getOutputStream());  
 int y = x / 2;  
 if (x == 1 || x == 2 || x == 3) {  
 otc.writeUTF(x + "is Prime");  
 System.*exit*(0);  
 }  
 for (int i = 2; i <= y; i++) {  
 if (x % i != 0) {  
 otc.writeUTF(x + " is Prime");  
 } else {  
 otc.writeUTF(x + " is not Prime");  
 }  
 }  
 }  
 catch (Exception e)  
 {  
 System.*out*.println(e.toString());  
 }  
 }  
}

Text

Description automatically generated

* Chatbot program using TCP/IP protocol:

*ChatClient.java*

import java.net.\*;  
import java.io.\*;  
public class ChatClient  
{  
 public static void main(String args[])  
 {  
 try  
 {  
 Socket s = new Socket("Localhost",8000);  
 BufferedReader br = new BufferedReader(new InputStreamReader(System.*in*));  
 DataOutputStream out = new DataOutputStream(s.getOutputStream());  
 BufferedReader in = new BufferedReader(new InputStreamReader(s.getInputStream()));  
 String msg;  
 System.*out*.println("To stop chatting with server type STOP");  
 System.*out*.print("Client Says: ");  
 while((msg = br.readLine()) != null)  
 {  
 out.writeBytes(msg+"\n");  
 if(msg.equals("STOP"))  
 break;  
 System.*out*.println("Server Says : "+in.readLine());  
 System.*out*.print("Client Says : ");  
 }  
 br.close();  
 in.close();  
 out.close();  
 s.close();  
 }  
 catch(Exception e)  
 {  
 e.printStackTrace();  
 }  
 }  
}

Text

Description automatically generated

*ChatServer.java*

import java.net.\*;  
import java.io.\*;  
class ChatServer  
{  
 public static void main(String args[])  
 {  
 try  
 {  
 ServerSocket ss = new ServerSocket(8000);  
 System.*out*.println("Waiting for client to connect..");  
 Socket s = ss.accept();  
 BufferedReader br = new BufferedReader(new InputStreamReader(System.*in*));  
 DataOutputStream out = new DataOutputStream(s.getOutputStream());  
 BufferedReader in = new BufferedReader(new InputStreamReader(s.getInputStream()));  
 String receive, send;  
 while((receive = in.readLine()) != null)  
 {  
 if(receive.equals("STOP"))  
 break;  
 System.*out*.println("Client Says : "+receive);  
 System.*out*.print("Server Says : ");  
 send = br.readLine();  
 out.writeBytes(send+"\n");  
 }  
 br.close();  
 in.close();  
 out.close();  
 s.close();  
 }  
 catch(Exception e)  
 {  
 e.printStackTrace();  
 }  
 }  
}

Text

Description automatically generated

*PRACTICAL – 2*

* ***Implementation of following program using UDP protocol***
* Program using UDP to find if the number entered is even or odd.

*udpClientEO.java*

import java.io.\*;  
import java.net.\*;  
  
public class udpClientEO  
{  
 public static void main(String args[])  
 {  
 try  
 {  
 DatagramSocket ds = new DatagramSocket(1000);  
 BufferedReader br = new BufferedReader(new InputStreamReader(System.*in*));  
 System.*out*.println("Enter a number : ");  
 String num = br.readLine();  
 byte b[] = new byte[1024];  
 b=num.getBytes();  
 DatagramPacket dp = new DatagramPacket(b,b.length,InetAddress.*getLocalHost*(),2000);  
 ds.send(dp);  
 byte b1[] = new byte[1024];  
 DatagramPacket dp1 = new DatagramPacket(b1,b1.length);  
 ds.receive(dp1);  
 String str = new String(dp1.getData(),0,dp1.getLength());  
 System.*out*.println(str);  
 }  
 catch(Exception e)  
 {  
 e.printStackTrace();  
 }  
 }  
}

Text

Description automatically generated

*udpServerEO.java*

import java.io.\*;  
import java.net.\*;  
  
public class udpServerEO  
{  
 public static void main(String args[])  
 {  
 try  
 {  
 DatagramSocket ds = new DatagramSocket(2000);  
 byte b[] = new byte[1024];  
 DatagramPacket dp = new DatagramPacket(b,b.length);  
 ds.receive(dp);  
 String str = new String(dp.getData(),0,dp.getLength());  
 System.*out*.println(str);  
 int a= Integer.*parseInt*(str);  
 String s= new String();  
 if (a%2 == 0)  
 s = "Number is even";  
 else  
 s = "Number is odd";  
 byte b1[] = new byte[1024];  
 b1 = s.getBytes();  
 DatagramPacket dp1 = new DatagramPacket(b1,b1.length,InetAddress.*getLocalHost*(),1000);  
 ds.send(dp1);  
 }  
 catch(Exception e)  
 {  
 e.printStackTrace();  
 }  
 }  
}

Graphical user interface, text

Description automatically generated

* A program to implement simple calculator operations like addition, subtraction, multiplication, and division using UDP protocol.

*udpClientCAL.java*

import java.io.\*;  
import java.net.\*;  
  
class udpClientCAL  
{  
 udpClientCAL()  
 {  
 try  
 {  
 InetAddress ia = InetAddress.*getLocalHost*();  
 DatagramSocket ds = new DatagramSocket();  
 DatagramSocket ds1 = new DatagramSocket(1300);  
 System.*out*.println("\nudp Client\n");  
 System.*out*.println("Enter method name {add/sub/div/mul} and parameter like {5 6}");  
 while (true)  
 {  
 BufferedReader br = new BufferedReader(new InputStreamReader(System.*in*));  
 String str = br.readLine();  
 byte b[] = str.getBytes();  
 DatagramPacket dp = new  
 DatagramPacket(b,b.length,ia,1200);  
 ds.send(dp);  
 dp = new DatagramPacket(b,b.length);  
 ds1.receive(dp);  
 String s = new String(dp.getData(),0,dp.getLength()); System.*out*.println("\nResult = " + s + "\n");  
 }  
 }  
 catch (Exception e)  
 {  
 e.printStackTrace();  
 }  
 }  
 public static void main(String[] args)  
 {  
 new udpClientCAL();  
 }  
}

*udpServerCAL.java*

import java.util.\*;  
import java.net.\*;  
  
class udpServerCAL  
{  
 DatagramSocket ds;  
 DatagramPacket dp;  
 String str,methodName,result;  
 int val1,val2;  
 udpServerCAL()  
 {  
 try  
 {  
 ds=new DatagramSocket(1200);  
 byte b[]=new byte[4096];  
 while(true)  
 {  
 dp=new DatagramPacket(b,b.length);  
 ds.receive(dp);  
 str=new String(dp.getData(),0,dp.getLength());  
 if(str.equalsIgnoreCase("q"))  
 {  
 System.*exit*(1);  
 }  
 else  
 {  
 StringTokenizer st = new StringTokenizer(str," "); int i=0;  
 while(st.hasMoreTokens())  
  
 {  
 String token = st.nextToken();  
 methodName = token;  
 val1 = Integer.*parseInt*(st.nextToken());  
 val2 = Integer.*parseInt*(st.nextToken());  
 }  
 }  
 System.*out*.println(str);  
 InetAddress ia = InetAddress.*getLocalHost*();  
 if(methodName.equalsIgnoreCase("add"))  
 {  
 result= "" + add(val1,val2);  
 }  
 else if(methodName.equalsIgnoreCase("sub"))  
 {  
 result= "" + sub(val1,val2);  
 }  
 else if(methodName.equalsIgnoreCase("mul"))  
 {  
 result= "" + mul(val1,val2);  
 }  
 else if(methodName.equalsIgnoreCase("div"))  
 {  
 result= "" + div(val1,val2);  
 }  
 byte b1[]=result.getBytes();  
 DatagramSocket ds1 = new DatagramSocket(); DatagramPacket dp1 = new  
 DatagramPacket(b1,b1.length,InetAddress.*getLocalHost*(), 1300); System.*out*.println("result : "+result+"\n"); ds1.send(dp1);  
 }  
 }  
 catch (Exception e)  
 {  
 e.printStackTrace();  
 }  
 }  
 public int add(int val1, int val2)  
 {  
 return val1+val2;  
 }  
 public int sub(int val3, int val4)  
 {  
 return val3-val4;  
 }  
 public int mul(int val3, int val4)  
 {  
 return val3\*val4;  
 }  
 public int div(int val3, int val4)  
 {  
 return val3/val4;  
 }  
 public static void main(String[] args)  
 {  
 new udpServerCAL();  
 }  
}

Text

Description automatically generated

*PRACTICAL – 3*

* ***Implementation of following program using TCP protocol***
* Write a client program to enter the number and server program to calculate the square of the entered number using TCP Communication.

*TCP\_ClientSquare.java*

Graphical user interface, text, application, email

Description automatically generated

A picture containing shape

Description automatically generated

*TCP\_ServerSquare.java*

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

* Write a client program to enter the number and server program to calculate the square root of the entered number using TCP Communication.

*TCP\_ClientSquareRoot.java*

Graphical user interface, text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

*TCP\_ServerSquareRoot.java*

Graphical user interface, text, application

Description automatically generated

A picture containing application

Description automatically generated

* Write a client program to enter the number and server program to calculate the Cube of the entered number using TCP Communication.

*TCP\_ServerCube.java*

Text

Description automatically generated

Graphical user interface, application, Word

Description automatically generated

*TCP\_ClientCube.java*

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

* Write a client program to enter the number and server program to calculate the Cube Root of the entered number using TCP Communication.

*TCP\_ServerCubeRoot.java*

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

*TCP\_ClientCubeRoot.java*

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

*PRACTICAL – 4*

* ***Implementation of following program using UDP protocol***
* Write a client program to enter the number and server program to calculate the factorial of the entered number using UDP Communication.

*UDP\_ServerFactorial.java*

Graphical user interface, text, application

Description automatically generated

*UDP\_ClientFactorial.java*

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated