**Networking Lab**

**Programs**

**Program No 1: Write a networking program in Java to implement a TCP server that provides services for a TCP Client.**

**TCPclient.java**

import java.io.\*;

import java.net.\*;

class TCPClient

{

public static void main(String[] args)

{

Socket client;

InputStream inputStream;

DataInputStream dataInputStream;

try

{

client = new Socket("localhost", 8090);

inputStream = client.getInputStream();

dataInputStream = new DataInputStream(inputStream);

System.out.println(dataInputStream.readUTF());

System.out.println(dataInputStream.readUTF());

client.close();

}

catch(IOException e)

{

System.out.println(e);

}

}

}

**TCPserver.java**

import java.io.\*;

import java.net.\*;

import java.util.Calendar;

class TCPServer

{

public static void main(String[] args)

{

ServerSocket server;

Socket client;

OutputStream outputStream;

DataOutputStream dataOutputStream;

Calendar calendar;

try

{

server = new ServerSocket(8090);

System.out.println("Server started...");

client = server.accept();

System.out.println("Connected: " + client.getInetAddress());

outputStream = client.getOutputStream();

dataOutputStream = new DataOutputStream(outputStream);

dataOutputStream.writeUTF("Hi from server. We provide time service.");

calendar = Calendar.getInstance();

dataOutputStream.writeUTF("Time: " + calendar.get(Calendar.HOUR\_OF\_DAY) + ":" + calendar.get(Calendar.MINUTE)

+ ":" + calendar.get(Calendar.SECOND));

server.close();

}

catch(IOException e)

{

System.out.println(e);

}

}

}

**OUTPUT:**

**TCPserver:**

run:

Server started...

**TCPclient:**

run:

Hi from server. We provide time service.

Time: 14:28:12

BUILD SUCCESSFUL (total time: 0 seconds)

**Program No 2: Write a networking program to implement socket programming using User datagram Protocol in Java.**

**UDPclient.java**

import java.net.\*;

import java.io.\*;

import java.util.\*;

public class UDPClient

{

public static void main(String[] args)

{

DatagramSocket datagramSocket;

DatagramPacket datagramPacket;

String userInput;

InetAddress ipAddr;

Scanner scanner = new Scanner(System.in);

byte[] bytes = new byte[1024];

try

{

datagramSocket = new DatagramSocket(7314); //created client on different port

ipAddr = InetAddress.getByName("localhost"); //if server is at different machine you can write it's ip address instead of localhost

System.out.println("Write msg to send");

userInput = scanner.nextLine();

bytes = userInput.getBytes();

datagramPacket = new DatagramPacket(bytes, bytes.length, ipAddr, 7313);

datagramSocket.send(datagramPacket);

scanner.close();

}

catch (SocketException ex)

{

System.out.println(ex);

}

catch (UnknownHostException ex)

{

System.out.println(ex);

}

catch (IOException ex)

{

System.out.println(ex);

}

}

}

**UDPserver.java**

import java.net.\*;

import java.io.\*;

public class UDPServer

{

public static void main(String[] args)

{

DatagramSocket datagramSocket;

DatagramPacket datagramPacket;

byte[] bytes;

String msg;

try

{

datagramSocket = new DatagramSocket(7313);

System.out.println("UDP Server started...");

bytes = new byte[1024];

datagramPacket = new DatagramPacket(bytes, 0, bytes.length);

datagramSocket.receive(datagramPacket);

msg = new String(bytes);

System.out.println(msg);

}

catch (SocketException ex)

{

System.out.println(ex);

}

catch (IOException ex)

{

System.out.println(ex);

}

}

}

**OUTPUT:**

**UDPserver:**

run:

UDP Server started...

**UDPclient:**

run:

Write msg to send

Welcome to Networking Lab

BUILD SUCCESSFUL (total time: 22 seconds)

**UDPserver:**

run:

UDP Server started...

Welcome to Networking Lab

**Program No 3: Implement an FTP server using socket programming.**

**FTPclient.java**

import java.net.\*;

import java.io.\*;

class FTPClient

{

public static void main(String[] args)

{

String fileName;

File file;

FileOutputStream fileOutputStream = null;

InputStream inputStream = null;

DataInputStream dataInputStream = null;

Socket socket = null;

byte[] byteArray;

try

{

socket = new Socket("localhost", 1000);

System.out.println("Connected...");

inputStream = socket.getInputStream();

dataInputStream = new DataInputStream(inputStream);

fileName = dataInputStream.readUTF();

file = new File(fileName);

fileOutputStream = new FileOutputStream(file);

int bytesRead = 0;

byteArray = new byte[1024 \* 10];

while((bytesRead = inputStream.read(byteArray)) > -1)

{

fileOutputStream.write(byteArray, 0, bytesRead);

}

System.out.println("Received Successfully...");

fileOutputStream.close();

socket.close();

}

catch (IOException e)

{

System.err.println("Error 1");

e.printStackTrace();

}

}

}

**FTPserver.java**

import java.io.\*;

import static java.lang.ProcessBuilder.Redirect.to;

import static java.lang.System.out;

import java.net.\*;

import java.util.\*;

class FTPServer

{

public static void main(String[] args)

{

String fileName = null;

FileInputStream fileInputStream = null;

OutputStream outputStream = null;

DataOutputStream dataOutputStream = null;

ServerSocket serverSocket = null;

Socket client = null;

Scanner scan = new Scanner(System.in);

File file;

byte[] bytes;

try

{

serverSocket = new ServerSocket(1000); //creating server

System.out.println("FTP Server started...");

client = serverSocket.accept(); //waiting until client get's connected

System.out.println("Client connected...");

outputStream = client.getOutputStream(); //get the output stream to send data to client

dataOutputStream = new DataOutputStream(outputStream); //DataOutputStream is usedto work with strings unlike output .stream = null

System.out.println("Enter file name to send");

fileName = scan.nextLine(); //get the file name or it's absolute path

file = new File(fileName);

fileInputStream = new FileInputStream(file); //fileInputStream to read the content of file

int fileLength = (int) file.length();

bytes = new byte[fileLength];

fileInputStream.read(bytes, 0, bytes.length); //read the content of file and save it's bytes to byte Array

dataOutputStream.writeUTF(file.getName()); //send user the file name on DataOutputStream

outputStream.write(bytes, 0, bytes.length); //send file bytes to client

outputStream.close(); //closing the stream

scan.close(); //closing the stream

System.out.println("Sent Successfully");

serverSocket.close(); //closing the connection

}

catch (IOException e)

{

e.printStackTrace();

}

}

}

**OUTPUT:**

**FTPserver:**

run:

FTP Server started...

**FTPclient:**

run:

Connected...

**FTPserver:**

run:

FTP Server started...

Client connected...

Enter file name to send

C:\Users\cs25\Desktop\FTP.txt

Sent Successfully

BUILD SUCCESSFUL (total time: 17 seconds)

**FTPclient:**

run:

Connected...

Received Successfully...

BUILD SUCCESSFUL (total time: 13 seconds)

**Program No 4: Implement a chat server using socket programming.**

**Chatclient.java**

import java.net.\*;

import java.io.\*;

import java.util.\*;

class ChatClient

{

public static void main(String[] args)

{

Socket socket = null;

Scanner scan = new Scanner(System.in);

String yourMsg, serverMsg;

InputStream inputStream = null;

OutputStream outputStream = null;

DataInputStream dataInputStream = null;

DataOutputStream dataOutputStream = null;

try

{

socket = new Socket("localhost", 7313);

System.out.println("Connected...");

inputStream = socket.getInputStream();

outputStream = socket.getOutputStream();

dataInputStream = new DataInputStream(inputStream);

dataOutputStream = new DataOutputStream(outputStream);

while(true)

{

serverMsg = dataInputStream.readUTF();

if(serverMsg.equals("exit"))

{

break; // exit the chat if server sends "exit"

}

System.out.println("Server: " + serverMsg);

System.out.print("You: ");

yourMsg = scan.nextLine();

dataOutputStream.writeUTF(yourMsg);

if(yourMsg.equals("exit"))

{

break;

}

}

socket.close();

}

catch(IOException e)

{

System.out.println(e);

}

}

}

**Chatserver.java**

import java.net.\*;

import java.io.\*;

import java.util.\*;

class ChatServer

{

public static void main(String[] args)

{

ServerSocket serverSocket = null;

Socket client = null;

Scanner scan = new Scanner(System.in);

String yourMsg, clientMsg;

OutputStream outputStream = null;

InputStream inputStream = null;

DataOutputStream dataOutputStream = null;

DataInputStream dataInputStream = null;

try

{

serverSocket = new ServerSocket(7313);

System.out.println("Server started...");

client = serverSocket.accept();

System.out.println("Client connected...");

outputStream = client.getOutputStream();

inputStream = client.getInputStream();

dataOutputStream = new DataOutputStream(outputStream);

dataInputStream = new DataInputStream(inputStream);

dataOutputStream.writeUTF("Hi from server");

while(true)

{

clientMsg = dataInputStream.readUTF();

if(clientMsg.equals("exit"))

{

break;

}

System.out.println("Client: " + clientMsg);

System.out.print("You: ");

yourMsg = scan.nextLine();

dataOutputStream.writeUTF(yourMsg);

if(yourMsg.equals("exit"))

{

break;

}

}

serverSocket.close();

}

catch(IOException e)

{

System.out.println(e);

}

}

}

**OUTPUT:**

**Chatserver:**

run:

Server started...

Client connected...

**Chatclient:**

run:

Connected...

Server: Hi from server

You: hello

**Chatserver:**

run:

Server started...

Client connected...

Client: hello

You: welcome to networking lab

**Chatclient:**

run:

Connected...

Server: Hi from server

You: hello

Server: welcome to networking lab

You: thank you

**Chatserver:**

Server started...

Client connected...

Client: hello

You: welcome to networking lab

Client: thank you

You: exit

BUILD SUCCESSFUL (total time: 5 minutes 13 seconds)

**Program No 5: Implement an ECHO server using socket programming.**

**ECHOclient.java**

import java.io.\*;

import java.net.\*;

public class ECHOClient

{

public static void main(String[] args)

{

try

{

Socket s = new Socket("127.0.0.1", 9999);

BufferedReader r = new BufferedReader(new InputStreamReader(s.getInputStream()));

PrintWriter w = new PrintWriter(s.getOutputStream(), true);

BufferedReader con = new BufferedReader(new InputStreamReader(System.in));

String line;

do

{

line = r.readLine();

if ( line != null )

System.out.println(line);

line = con.readLine();

w.println(line);

}

while ( !line.trim().equals("bye") );

}

catch (Exception err)

{

System.err.println(err);

}

}

}

**ECHOserver.java**

import java.io.\*;

import java.net.\*;

public class EchoServer

{

public EchoServer(int portnum)

{

try

{

server = new ServerSocket(portnum);

}

catch (Exception err)

{

System.out.println(err);

}

}

public void serve()

{

try

{

while (true)

{

Socket client = server.accept();

BufferedReader r = new BufferedReader(new

InputStreamReader(client.getInputStream()));

PrintWriter w = new PrintWriter(client.getOutputStream(), true);

w.println("Welcome to the Java EchoServer. Type 'bye' to close.");

String line;

do

{

line = r.readLine();

if ( line != null )

w.println("Got: "+ line);

}

while ( !line.trim().equals("bye") );

client.close();

}

}

catch (Exception err)

{

System.err.println(err);

}

}

public static void main(String[] args)

{

EchoServer s = new EchoServer(9999);

s.serve();

}

private ServerSocket server;

}

**OUTPUT:**

**Chatserver:**

run:

**Chatclient:**

run:

Welcome to the Java EchoServer. Type 'bye' to close.

hi

Got: hi

hello

Got: hello

bye

BUILD SUCCESSFUL (total time: 23 seconds)

**Program No 6: Implement Address Resolution Protocol using socket programming.**

**ARPDemo.java**

import java.util.\*;

import java.io.\*;

import java.net.\*;

public class ARPDemo

{

public static void main(String[] args)

{

String ip;

Scanner scan = new Scanner(System.in);

ProcessBuilder processBuilder = new ProcessBuilder();

Process process;

System.out.println("Enter the ip address");

ip = scan.nextLine();

InputStream is;

try

{

InetAddress inet = InetAddress.getByName(ip);

if(inet.isReachable(5000))

{

process = processBuilder.command("arp", "-a").start();

is = process.getInputStream();

BufferedReader buff = new BufferedReader(new InputStreamReader(is));

String res;

while((res = buff.readLine()) != null)

{

if(res.contains(ip))

{

res = res.trim();

res = res.replaceAll(" +", " ");

String[] array = res.split(" ");

System.out.println(array[0] + " ==> " + array[1]);

}

}

}

else

{

System.out.println("Host is not present");

}

}

catch(Exception e)

{

System.out.println(e);

}

}

}

**OUTPUT1:**

run:

Enter the ip address

172.16.16.16:8090

java.net.UnknownHostException: 172.16.16.16:8090

BUILD SUCCESSFUL (total time: 14 seconds)

**OUTPUT2:**

run:

Enter the ip address

192.168.250.21

192.168.250.21 ==> 6c-3b-e5-2e-66-24

BUILD SUCCESSFUL (total time: 8 seconds)

**Program No 7: Implement a program to retrieve the data for the specified URL**.

import java.io.\*;

import java.net.\*;

import java.util.\*;

public class RetrieveData

{

public static void main(String[] args)

{

Scanner scan = new Scanner(System.in);

HttpURLConnection connection;

URL url;

String input;

try

{

System.out.println("Enter the URL");

input = scan.nextLine();

url = new URL(input);

connection = (HttpURLConnection) url.openConnection();

System.out.println("Request Method: " +

connection.getRequestMethod());

System.out.println("Response Code: " + connection.getResponseCode());

System.out.println("Response Message: " +

connection.getResponseMessage());

Map<String, List<String>> headerFields = connection.getHeaderFields();

Set<String> headerKeys = headerFields.keySet();

for(String key: headerKeys){

System.out.println("Key: " + key + " : " + "Value: " +

headerFields.get(key));

}

connection.disconnect();

scan.close();

}

catch (Exception ex)

{

System.out.println(ex);

}

}

}

**OUTPUT:**

run:

Enter the URL

[https://google.com](https://google.com/)

Request Method: GET

java.net.UnknownHostException: google.com

BUILD SUCCESSFUL (total time: 1 minute 19 seconds)