

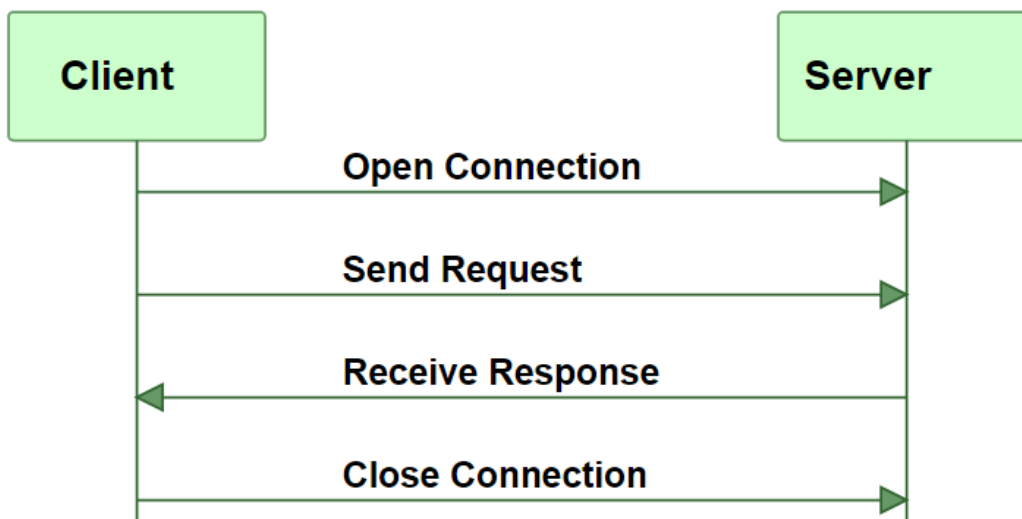
ADVANCED JAVA Unit-1

GTU Syllabus: Java Networking

Network Basics and Socket overview, TCP/IP client sockets, URL, TCP/IP server sockets, Datagrams, java.net package Socket, ServerSocket, InetAddress, URL, URLConnection

Java TCP Networking Basics

- ✓ Typically a client opens a TCP/IP connection to a server. The client then starts to communicate with the server. When the client is finished it closes the connection again. Here is an illustration of that:



- ✓ A client may send more than one request through an open connection. In fact, a client can send as much data as the server is ready to receive. The server can also close the connection if it wants to.

Compiled By,

Prof.Nirali Varnagar

Java Socket's and ServerSocket's

- ✓ When a client wants to open a TCP/IP connection to a server, it does so using a **Java Socket**. The socket is told what IP address and TCP port to connect to and the rest is done by Java.
- ✓ If you want to start a server that listens for incoming connections from clients on some TCP port, you have to use a **Java ServerSocket**. When a client connects via a client socket to a server's ServerSocket, a Socket is assigned on the server to that connection. The client and server now communicates Socket-to-Socket.

Java UDP Networking Basics

- ✓ UDP works a bit differently from TCP. Using UDP there is no connection between the client and server. A client may send data to the server, and the server may (or may not) receive this data. The client will never know if the data was received at the other end. The same is true for the data sent the other way from the server to the client.
- ✓ Because there is no guarantee of data delivery, the UDP protocol has less protocol overhead.
- ✓ In order to connect to a server over the internet (via TCP/IP) in Java, you need to create a `java.net.Socket` and connect it to the server.

UDP vs. TCP

UDP works a bit differently from TCP. When you send data via TCP you first create a connection. Once the TCP connection is established TCP guarantees that your data arrives at the other end, or it will tell you that an error occurred.

With UDP you just send packets of data (datagrams) to some IP address on the network. You have no guarantee that the data will arrive. You also have no guarantee about the order which UDP packets arrive in at the receiver. This means that UDP has less protocol overhead (no stream integrity checking) than TCP.

Compiled By,

Prof.Nirali Varnagar

Java.net Package:

InetAddress class:

- Java has a class `java.net.InetAddress` which abstracts network addresses
- Serves three main purposes:
 - Encapsulates an address
 - Performs name lookup (converting a host name into an IP address)
 - Performs reverse lookup (converting the address into a host name)

Methods:

- **Static construction using a factory method**
 - **`InetAddress getByName(String hostName)`**
 - `hostName` can be “host.domain.com.au”, or
 - `hostName` can be “130.95.72.134”
 - **`InetAddress getLocalHost()`**
- **Some useful methods:**
 - **`String getHostName()`**
 - Gives you the host name (for example “www.sun.com”)
 - **`String getHostAddress()`**
 - Gives you the address (for example “192.18.97.241”)
 - **`InetAddress[] getAllByName(String hostName)`**

Example:

```
import java.io.*;

import java.net.*;

class Address

{
    public static void main(String args[ ]) throws IOException

    {
        //accept name of website from keyboard

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

        System.out.print("Enter a website name: ");

        String site = br.readLine();

        try{

            //getByName() method accepts site name and returns its IP
```

Compiled By,

Prof.Nirali Varnagar

```
InetAddress ip = InetAddress.getByName(site);

System.out.println("The IP Address is: " + ip);

}catch(UnknownHostException ue)

{

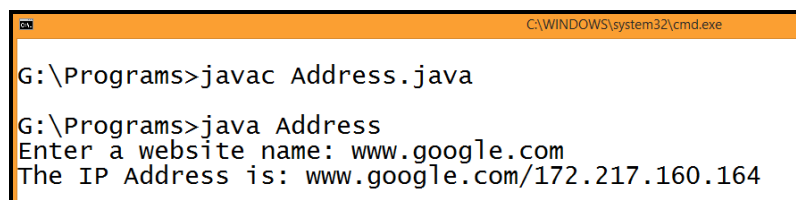
    System.out.println("Website not found");

}

}

}
```

Output:



```
C:\WINDOWS\system32\cmd.exe

G:\Programs>javac Address.java

G:\Programs>java Address
Enter a website name: www.google.com
The IP Address is: www.google.com/172.217.160.164
```

URL class:

- A URL contains many information:
- **Protocol:** In this case, http is the protocol.
- **Server name or IP Address:** In this case, www.javatpoint.com is the server name.
- **Port Number:** It is an optional attribute. If we write http://www.javatpoint.com:80/sonoojaiswal/ , 80 is the port number. If port number is not mentioned in the URL, it returns -1.
- **File Name or directory name:** In this case, index.jsp is the file name.

Constructor: **URL class represent in java.net package**

```
URL obj=new URL(String protocol,string host,int port,string path);
```

```
URL obj=new URL(String site);
```

Compiled By,

Prof.Nirali Varnagar

Methods:

1. public String getProtocol()

➔ it returns the protocol of the URL.

2. public String getHost()

➔ it returns the host name of the URL.

3. public String getPort()

➔ it returns the Port Number of the URL.

4. public String getFile()

➔ it returns the file name of the URL.

5. public URLConnection.openConnection()

➔ it returns the instance of URLConnection i.e. associated with this URL.

6. String toExternalForm()

Program:

```
import java.net.*;

class MyURL
{
    public static void main(String args[ ]) throws Exception{

        URL obj = new URL("http://dreamtechpress.com/index.html");

        System.out.println("Protocol: "+ obj.getProtocol());

        System.out.println("Host: "+ obj.getHost());

        System.out.println("File: "+ obj.getFile());

        System.out.println("Port: "+ obj.getPort());

        System.out.println("Path: "+ obj.getPath());
```

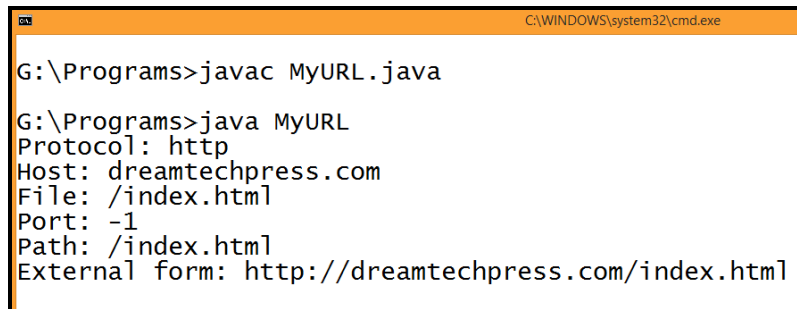
Compiled By,
Prof.Nirali Varnagar

```

        System.out.println("External form: "+ obj.toExternalForm());
    }
}

```

Output:



```

C:\WINDOWS\system32\cmd.exe
G:\Programs>javac MyURL.java
G:\Programs>java MyURL
Protocol: http
Host: dreamtechpress.com
File: /index.html
Port: -1
Path: /index.html
External form: http://dreamtechpress.com/index.html

```

URLConnection class:

- The **Java URLConnection** class represents a communication link between the URL and the application. This class can be used to read and write data to the specified resource referred by the URL.
- How to get the object of URLConnection class
- The openConnection() method of URL class returns the object of URLConnection class.
Syntax:
- **public URLConnection openConnection() throws IOException{ }**

Methods:

1.getDate()

2.getContentType()

3.getExpiration()

4.getLastModified()

5.getContentLength()

Program:

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

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

Compiled By,
Prof.Nirali Varnagar

```
import java.util.*;

class Details

{

    public static void main(String args[ ]) throws Exception

    {

        //pass the site url to URL object

        URL obj = new URL("http://www.google.com");

        //open a connection with the site on Internet

        URLConnection conn = obj.openConnection();

        //display the date

        System.out.println("Date: "+ new Date(conn.getDate()));

        //display the content type whether text or html

        System.out.println("Content-type: "+ conn.getContentType());

        //display expiry date

        System.out.println("Expiry: "+ conn.getExpiration());

        //display last modified date

        System.out.println("Last modified: "+ new

            Date(conn.getLastModified()));

        //display how many bytes the index.html page has

        int l = conn.getContentLength();

        System.out.println("Length of content: "+ l);

        if(l == 0)

        {
```

Compiled By,
Prof.Nirali Varnagar

```

        System.out.println("Content not available");

        return;

    }

    else {

        int ch;

        InputStream in = conn.getInputStream();

        //display the content of the index.html page

        while((ch = in.read())!= -1)

            System.out.print((char)ch);

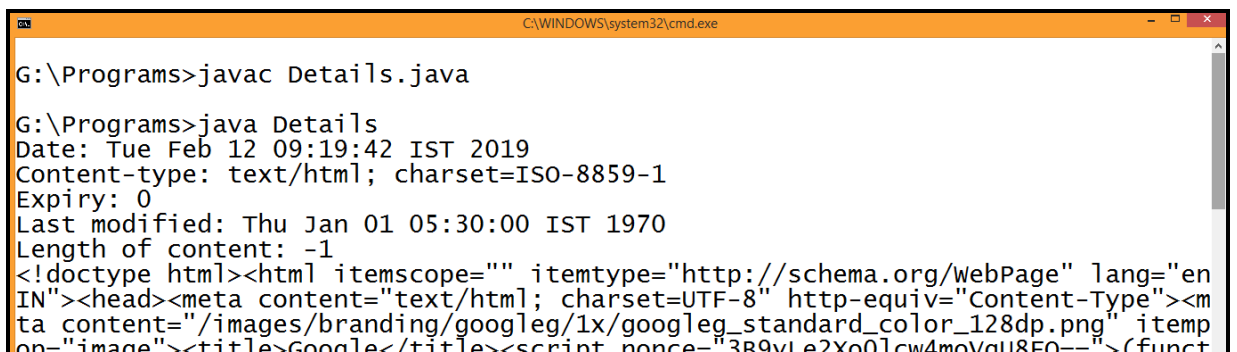
        }

    }

}

```

Output:



```

C:\WINDOWS\system32\cmd.exe
G:\Programs>javac Details.java
G:\Programs>java Details
Date: Tue Feb 12 09:19:42 IST 2019
Content-type: text/html; charset=ISO-8859-1
Expiry: 0
Last modified: Thu Jan 01 05:30:00 IST 1970
Length of content: -1
<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="en
IN"><head><meta content="text/html; charset=UTF-8" http-equiv="Content-Type"><m
ta content="/images/branding/googleg/1x/googleg_standard_color_128dp.png" itemp
on="image"><title>Google</title><script nonce="3B9vle2Xo0lcw4moVouU8EQ=="><funct

```

Socket Programming:

Sockets provide the communication mechanism between two computers using TCP. A client program creates a socket on its end of the communication and attempts to connect that socket to a server.

Compiled By,

Prof.Nirali Varnagar

When the connection is made, the server creates a socket object on its end of the communication. The client and server can now communicate by writing to and reading from the socket.

The `java.net.Socket` class represents a socket, and the `java.net.ServerSocket` class provides a mechanism for the server program to listen for clients and establish connections with them.

The following steps occur when establishing a TCP connection between two computers using sockets:

- The server instantiates a `ServerSocket` object, denoting which port number communication is to occur on.
- The server invokes the `accept()` method of the `ServerSocket` class. This method waits until a client connects to the server on the given port.
- After the server is waiting, a client instantiates a `Socket` object, specifying the server name and port number to connect to.
- The constructor of the `Socket` class attempts to connect the client to the specified server and port number. If communication is established, the client now has a `Socket` object capable of communicating with the server.
- On the server side, the `accept()` method returns a reference to a new socket on the server that is connected to the client's socket.

After the connections are established, communication can occur using I/O streams. Each socket has both an `OutputStream` and an `InputStream`. The client's `OutputStream` is connected to the server's `InputStream`, and the client's `InputStream` is connected to the server's `OutputStream`.

TCP is a twoway communication protocol, so data can be sent across both streams at the same time. There are following usefull classes providing complete set of methods to implement sockets.

Two types of TCP Socket:

- `java.net.ServerSocket` is used by servers so that they can accept incoming TCP/IP connections.

- A server is a piece of software which *advertises* and then provides some service on request
- `java.net.Socket` is used by clients who wish to establish a connection to a (remote) server
 - A client is a piece of software (usually on a different machine) which makes use of some service.

ServerSocket Class Methods:

The **`java.net.ServerSocket`** class is used by server applications to obtain a port and listen for client requests

The `ServerSocket` class has four constructors:

SN	Methods with Description
1	<code>public ServerSocket(int port) throws IOException</code> Attempts to create a server socket bound to the specified port. An exception occurs if the port is already bound by another application.
2	<code>public ServerSocket(int port, int backlog) throws IOException</code> Similar to the previous constructor, the backlog parameter specifies how many incoming clients to store in a wait queue.

Some useful methods:

- **`Socket accept()`**: Blocks waiting for a client to attempt to establish a connection
- **`void close()`**: Called by the server when it is shutting down to ensure that any resources are deallocated.

Socket Class Methods:

The **`java.net.Socket`** class represents the socket that both the client and server use to communicate with each other. The client obtains a `Socket` object by instantiating one, whereas the server obtains a `Socket` object from the return value of the `accept()` method.

Compiled By,

Prof.Nirali Varnagar

The Socket class has five constructors that a client uses to connect to a server:

SN	Methods with Description
1	<p>public Socket(String host, int port) throws UnknownHostException, IOException.</p> <p>This method attempts to connect to the specified server at the specified port. If this constructor does not throw an exception, the connection is successful and the client is connected to the server.</p>

When the Socket constructor returns, it does not simply instantiate a Socket object but it actually attempts to connect to the specified server and port.

Some methods of interest in the Socket class are listed here. Notice that both the client and server have a Socket object, so these methods can be invoked by both the client and server.

SN	Methods with Description
1	<p>public InputStream getInputStream() throws IOException</p> <p>Returns the input stream of the socket. The input stream is connected to the output stream of the remote socket.</p>
2	<p>public OutputStream getOutputStream() throws IOException</p> <p>Returns the output stream of the socket. The output stream is connected to the input stream of the remote socket</p>
3	<p>public void close() throws IOException</p> <p>Closes the socket, which makes this Socket object no longer capable of connecting again to any server</p>

Case-1: Server sends some data which is receive by the client (One way communication)

Server1.java

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

class Server1
{
    public static void main(String args[ ])
        throws Exception
    {
        //Create a server socket with some port number
        ServerSocket ss = new ServerSocket(777);

        //let the server wait till a client accepts connection
        Socket s = ss.accept();

        System.out.println("Connection established");

        //attach output stream to the server socket
        OutputStream obj = s.getOutputStream();

        //attach print stream to send data to the socket
        PrintStream ps = new PrintStream(obj);

        //send 2 strings to the client
        String str = "Hello client";

        ps.println(str);

        ps.println("Bye");

        //close connection by closing the streams and sockets
        ps.close();

        ss.close();

        s.close();
    }
}
```

Compiled By,
Prof.Nirali Varnagar

```
}
```

Client1.java

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

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

```
class Client1
```

```
{
```

```
    public static void main(String args[ ])
```

```
    throws Exception
```

```
    {
```

```
        Socket s = new Socket("localhost", 777);
```

```
        //to read data coming from server, attach InputStream to the socket
```

```
        InputStream obj = s.getInputStream();
```

```
        //to read data from the socket into the client, use BufferedReader
```

```
        BufferedReader br = new BufferedReader(new InputStreamReader(obj));
```

```
        //receive strings
```

```
        String str;
```

```
        while((str = br.readLine()) != null)
```

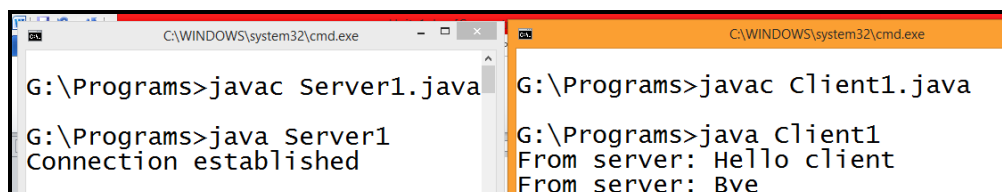
```
            System.out.println("From server: "+str);
```

```
        //close connection by closing the streams and sockets
```

```
        br.close();
```

```
        s.close();}}
```

Output:



```
C:\WINDOWS\system32\cmd.exe
G:\Programs>javac Server1.java
G:\Programs>java Server1
Connection established

C:\WINDOWS\system32\cmd.exe
G:\Programs>javac Client1.java
G:\Programs>java Client1
From server: Hello client
From server: Bye
```

Compiled By,

Prof.Nirali Varnagar

Case-2: Two ways Communication between client and server**Client2.java**

```
import java.io.*;

import java.net.*;

class Client2

{

    public static void main(String args[ ]) throws Exception

    {

        //Create client socket

        Socket s=new Socket("localhost",888);

        //to send data to the server (DOS)

        DataOutputStream dos=new DataOutputStream(s.getOutputStream());

        //to read data coming from the server

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

        //to read data from the key board

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

        String str,str1;

        //repeat as long as exit is not typed at client

        while(!(str=kb.readLine()).equals("exit"))

        {

            dos.writeBytes(str+"\n");

            str1=br.readLine();

            StringBuffer sb=new StringBuffer(str1);

            sb=sb.reverse();

            System.out.println(sb);

        }

    }

}
```

Compiled By,

Prof.Nirali Varnagar

```
s.close();

    }

}

Server2.java

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

class Server2
{
    public static void main(String args[ ])throws Exception
    {
        //Create server socket

        ServerSocket ss=new ServerSocket(888);

        //connect it to client socket

        Socket s=ss.accept();

        System.out.println("connection");

        //to send data to the client printStream

        PrintStream ps=new PrintStream(s.getOutputStream());

        //to read data coming from the client

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

        //to read data from the key board

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

        while(true) //server executes continuously

        {

            String str,str1;
```

Compiled By,
Prof.Nirali Varnagar

```

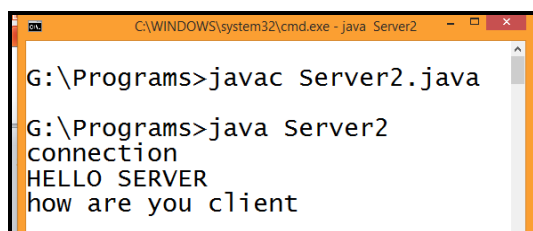
//repeat as long as client does not send null string
while((str=br.readLine())!=null)
{
    System.out.println(str.toUpperCase());
    str1=kb.readLine();
    ps.println(str1);
}

//close connection
ss.close();
s.close();

System.exit(0); //terminate application
} //end of while
}

```

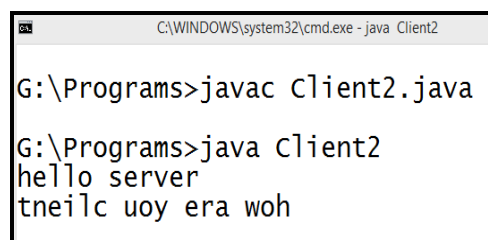
Output:



```

C:\WINDOWS\system32\cmd.exe - java Server2
G:\Programs>javac Server2.java
G:\Programs>java Server2
connection
HELLO SERVER
how are you client

```



```

C:\WINDOWS\system32\cmd.exe - java Client2
G:\Programs>javac Client2.java
G:\Programs>java Client2
hello server
tneilc uoy era woh

```

Case-3 : **Server having some file and which is search by the client and display content of it.**

(Refer from Lab manual)

Java DatagramSocket and DatagramPacket

Java DatagramSocket and DatagramPacket classes are used for connection-less socket programming.

Compiled By,

Prof.Nirali Varnagar

Java DatagramSocket class

Java DatagramSocket class represents a connection-less socket for sending and receiving datagram packets.

A datagram is basically an information but there is no guarantee of its content, arrival or arrival time.

Commonly used Constructors of DatagramSocket class

- **DatagramSocket()** throws **SocketEeption**: it creates a datagram socket and binds it with the available Port Number on the localhost machine.
 - **DatagramSocket(int port)** throws **SocketEeption**: it creates a datagram socket and binds it with the given Port Number.
 - **DatagramSocket(int port, InetAddress address)** throws **SocketEeption**: it creates a datagram socket and binds it with the specified port number and host address.
-

Java DatagramPacket class

Java DatagramPacket is a message that can be sent or received. If you send multiple packet, it may arrive in any order. Additionally, packet delivery is not guaranteed.

Commonly used Constructors of DatagramPacket class

- **DatagramPacket(byte[] barr, int length)**: it creates a datagram packet. This constructor is used to receive the packets.
 - **DatagramPacket(byte[] barr, int length, InetAddress address, int port)**: it creates a datagram packet. This constructor is used to send the packets.
-

Compiled By,

Prof.Nirali Varnagar

Example of Sending DatagramPacket by DatagramSocket

```
//DSender.java

import java.net.*;

public class DSender{

    public static void main(String[] args) throws Exception {

        DatagramSocket ds = new DatagramSocket();

        String str = "Welcome java";

        DatagramPacket dp = new DatagramPacket(str.getBytes(), str.length(), InetAddress.getLocalHost(),
        3000);

        ds.send(dp);

        ds.close();

    }

}
```

Example of Receiving DatagramPacket by DatagramSocket

```
//DReceiver.java

import java.net.*;

public class DReceiver{

    public static void main(String[] args) throws Exception {

        DatagramSocket ds = new DatagramSocket(3000);

        byte[] buf = new byte[1024];

        DatagramPacket dp = new DatagramPacket(buf, 1024);

        ds.receive(dp);

        String str = new String(dp.getData(), 0, dp.getLength());

        System.out.println(str);

        ds.close(); } }
```

Compiled By,

Prof.Nirali Varnagar

GTU asked questions

No	Questions	Marks	Year
1	Discuss the difference between the Socket and ServerSocket class.	3	W-2017 S-2018
2	Write a client-server program using UDP socket. Client send list of N numbers to server and server respond the sum of N numbers.	7	W-2017
3	Write a client-server program using UDP socket. Client send the list of N strings and server responds the concatenation of those strings.	7	W-2017
4	Write a client server program using TCP where client sends a string and server checks whether that string is palindrome or not and responds with appropriate message.	7	S-2017 S-2018
5	Write a client-server program using TCP or UDP where the client sends 10 numbers and server responds with the numbers in sorted order.	7	S-2016
6	Write a Java Client and Server Program to get date time from server on the client request.	7	S-2015,2016
7	What is Server Socket?How it works in java?Explain it with the help of example.	7	S-2014,2016
8	What is Datagram Socket? Explain it with the help of example.	7	S-2014,2016
9	Explain Protocol Handlers	3	S-2014
10	Write an UDP client and server program to do the following: Client send any string and server respond with its capital string.	7	S-2014,2016
11	Write a client server program using TCP where client sends two numbers and server responds with sum of them.	7	S-2014
12	Write a client-server program using TCP sockets to echo the message send by the client.	7	S-2013
13	Use of the URL class.	3	S-2013
14	Discuss the URL and URLConnection class with	7	W-2013

Compiled By,

Prof.Nirali Varnagar

	their use in network programming.		
15	URLConnection class	3	S-2012
16	DatagramSocket and DatagramPacket class	3	S-2012
17	Write a TCP <i>or</i> UDP client and server program to do the following: client> java client localhost/IP Port <enter> Enter text: This is my text to be changed by the SERVER <enter> Response from server: revres EHT YB DEGNAHC EB OT TXET YM SI SIHt client> exit	7	W-2012 S-2016
18	Write a client program to send any string from its standard input to the server program. The server program reads the string, finds number of characters and digits and sends it back to client program. Use connection-oriented or connection-less communication.	7	W-2011
19	Explain Socket, ServerSocket, InetAddress classes. Write a java program to find an IP address of the machine on which the program runs.	7	W-2011, W-2018
20	Write a client-server application using TCP Sockets where client can send message and server respond with the reverse of them.	7	W-2018