It is a java application that allows communication between two or more java application running on different JREs.

Components:

IP Address

→ It is a numerical value that is assigned to a computer in a network.

→ It is used for Network Interface Identification and location addressing.

→ IP address is of 2 types:
IPv4 and IPv6.

IPv4

→ It is 32 bits long and is divided into 4 octates.

IPv6

→ It is 128 bits long

Port

→ It is a logical construct

→ It is a number that uniquely identifyies a process running in computer.

→ It is 16 bits long unsigned integer. ranging from 0 to 65535

Types of ports

1) Well known ports (0-1023)

→ Port numbers running from 0-1023 is well known ports

→ These are assigned to commonly used applications eg (HTTP 80, HTTPS: 443, SMTP: 25.

2) Registered Ports (1024-49151)

→ These numbers are assigned to vendor applications.

→ e.g. (MySQL: 3306, Oracle: 1521)

3) Dynamic Ports (49152-65535)

→ Ephemeral

→ Used by applications which runs for short period of time.

Socket

- It is defined as end point of two way application.

→ combination of IP address and port nu

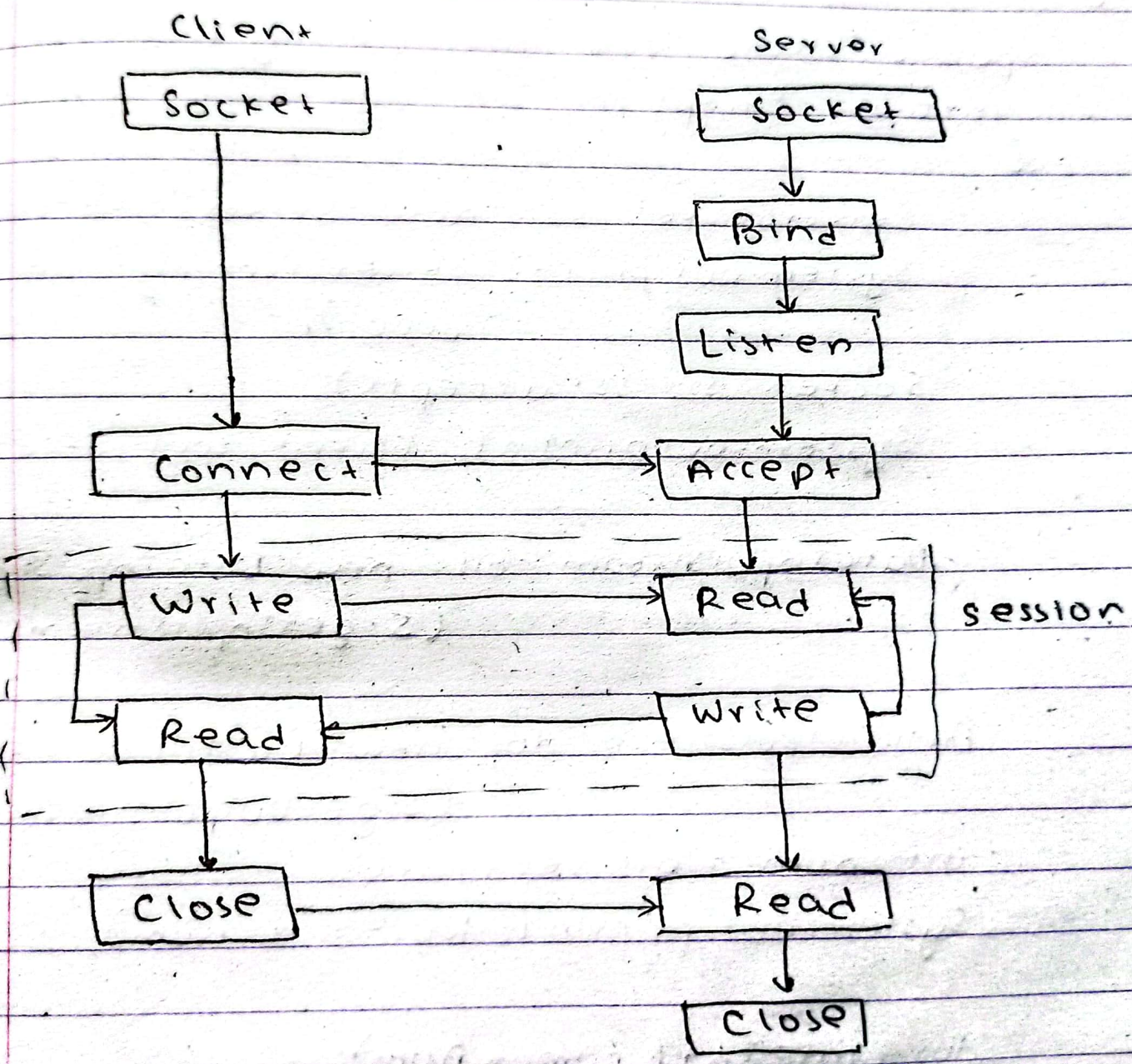
Protocol

→ It is defined as the set of rules followed by receiver and transmitter in

data communication.

→ Protocol can be either connection oriented (TCP) or connectionless (UDP).

TCP Architecture



Q) Create a TCP application where client sends a number (integer) and server responds by sending its square.

=>

```
import java.net.*;
import java.io.*;
class Server
{
    public static void main(String [] args)
    throws Exception
    {
        ServerSocket ss = new ServerSocket(60
        System.out.println("Server running on
                                6000");
        Socket s = ss.accept();
        System.out.println("Client connected

        DataInputStream dis = new DataInputSt
                              (s.getInputStream

        DataOutputStream dos = new DataOutputSt
                               (s.getOutputStrea

        int num = dis.readInt();
        System.out.println("Num = "+num);

        dos.writeInt(num *num);

        dos.close();
        dis.close();
```

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

Client side

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

class Client
{
    public static void main(String [] args)
    throws Exception
    {
        Scanner scan = new Scanner(System.in
        System.out.println("Enter num:");
        int num = scan.nextInt();

        Socket s=new Socket("localhost", 6000)

        DataInputStream dis = new DataInput
            -Stream(s.getInputStream());
        DataOutpurStream dos = new DataOutput
            -Stream(s.getOutputStream())
        dos.writeInt(num);
        int square = ~~do~~ dis.readInt();
```

```
        System.out.println (" square :"+ square);

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

Create a TCP architecture where client sends a string and server echos the same string in uppercase.

```
import    java.net.*;
import    java.io.*;
class  server
{
    public  static void main( String []a
    throws  Exception
    {
        ServerSocket  ss = new serverSo
                                (5000);

        System.out.println(" server is ru
            on  port  5000 ");

        Socket  s = ~~ss~~ ss.accept();
        System.out.println(" client con
```

```
DataInputStream dis = new DataInputStr-
        eam(s.getInputStream());
DataOutputStream dos = new DataOutputStre-
        am(s.getOutputStream());
String st= dis.readUTF();
System.out.println("String : "+ st);
dos.writeUTF(st.toUpperCase());

dos.close();
dis.close();
s.close();
ss.close();
}
}
```

Client Side

```
import java.net.*; import java.util.Scan
import java.io.*;
class Client
{
    public static void main(String[] args
    throws Exception
    {
        Scanner scan = new Scanner(Syste
        String st = new scan.next();
        System.out.println("Enter string:
        String st = new scan.next();
```

```
Socket s = new Socket("localhost",80

DataInputStream dis = new DataInputSt
        -am(s.getInputStream());
DataOutputStream dos = new DataOutputSt
        (s.getOutputStream());

dos.writeUTF(st);
String str = dis.readUTF();

System.out.println("uppercase =" +str

dos.close();
dis.close();
s.close();

    }
}
```

09/05 Friday

s.getPort();

6) create a TCP client server application where client sends string and server replies whether the string is palindrome or not

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

class Server
{
    public static void main (String [] args)
    throws Exception
    {
        ServerSocket ss = new ServerSocket(5000);
        System.out.println("Server running on 5000");

        Socket s = ss.accept();
        System.out.println("Client connected");
    DataInputStream dis = new DataInputStre
                          am(s.getInputStream());

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

    String str = dis.readUTF();
    String ans = isPalindrome()? "Palindrome":
                 "Not Palindrome";
    dos.writeUTF(ans);
    dos.close();
```

```
            dis.close();
            s.close();
            ss.close();

        }
        public static boolean isPalindrome(String s
        {
            String rev = "";
            for(int i = ~~rev~~str.length()-1 ; i >=0 ; i--
            {
                rev = rev + str.charAt(i);
            }
            if( rev.equalsIgnoreCase(str))
            {
                return true;
            }
            else
            {
                return false;
            }

    }
}
```

```
import java.net.*; import java.util.*;
import java.io.*;
class Client
{
    public static void main(String [] args)
    throws Exception
    {
```

```
Scanner scan = new Scanner(System.in);

Socket s = new Socket("localhost", 5000);

DataInputStream dis = new DataInputStre-
                      am(s.getInputStream());

DataOutputStream dos = new DataOutputStr-
                       eam(s.getOutputStream());

System.out.println("Enter a string:");
String msg = scan.next();
dos.writeUTF(msg);
dis.writeUTF(msg);
String res = dos.readUTF();
System.out.println("msg + is " + res);

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

Q) Create TCP client server app. where client sends string and server says palindrome or not. The communication continues until the client says 'bye'.

→ Client side

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

class Client
{
    public static void main (String [] args)
    throws Exception
    {
        Scanner scan = new Scanner(System

        Socket s = new Socket("localhost",

DataInputStream dis = new DataInputStrea
                      (s.getInputStream());
DataOutputStream dos = new DataOutputStre
                      (s.getOutputStream
```

```
        dos.writeUTF(msg);

        if(msg.equalsIgnoreCase("Bye"))
        {
            break;
        }

        String  res = dis.readUTF();

        System.out.println(msg+" is "+res);
    }
    dos.close();
    dis.close();
    s.close();
}
}
```

Server Side

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

class  Server
{
    public static void main(String []args)
    throws Exception
    {
        ServerSocket ss = new ServerSocket
                          (5000);
```

```
System.out.println("Server running");

Socket s = ss.accept();
System.out.println("Client connected");

DataInputStream dis = new DataInput
        Stream(s.getInputStream());
DataOutputStream dos = new DataOutpu
        -Stream(s.getOutputStream());

while(true)
{
    String msg = dis.readUTF();
    if(msg.equalsIgnoreCase("Bye"))
    {
        break;
    }

    String rep = isPalindrome(msg)?"Pa
                -rome":"Not Palindrom

    dos.writeUTF(rep);

}
dos.close();
dis.close();
s.close();
ss.close();
}
}
```

Socket — ServerSocket (TCP)
DatagramSocket (UDP)

UDP (User Datagram Protocol)

Create a UDP application where client sends a string and server responds by echoing in uppercase

```
import java.net.*;

class Client
{
    public static void main(String []args)
    throws Exception
    {
        DatagramSocket client = new Datagram
                                Socket();
        String msg = "Hello";
        byte [] snBuffer = msg.getBytes();
        int serverPort = 6000;
        InetAddress serverIp = InetAddress.getLo-
                               -calHost();
        DatagramPacket snPacket = new Data-
                -gramPacket(snBuffer,
                snBuffer.length, serverIp, serverP
                -ort);

        client.send(snPacket);
```

```
// Receiving
byte [] rxBuffer = new byte [1024];

DatagramPacket rxPacket = new Datagram-
                Packet( rxBuffer, rxBuffer.length
client.receive( rxPacket );

String reply = new String( rx.getData() );
System.out.println( reply );

client.close();
    }
}


//Server Code..
import java.net.*;

class Server
{
    public static void main (String [] args)
    throws Exception
    {
        DatagramSocket server = new
                DatagramSocket (6000);

        byte [] rxBuffer = new byte [1024];
```

```
DatagramPacket rxPacket = new DatagramPack-
                ·et(rxBuffer, rxBuffer.length);

    server.receive(rxPacket);
    String msg = new String(rxPacket.
    String reply = msg.to getData());

    String reply = msg.toUpperCase();

// sending
    byte [] snBuffer = reply.getBytes();

    int clientPort = rxPacket.getPort();
InetAddress clientIp = rxPacket.getAddress()

DatagramPacket snPacket = new
            DatagramPacket(snBuffer,
                snBuffer.length, clientIp,
                    clientPort);
/
server.send(snPacket);

    server.close();
}
}
```

TCP	UDP
1) Transmission control Protocol	1) User Datagram Protocol.
2) It is connection oriented protocol.	2) It is connectionless protocol.
3) It is reliable; it guarantees the deli-very of data.	3) It is not reliable; it doesn't guarantee of delivery of data
4) It has the provision of retransmitting in case of data loss.	4) It doesn't retransm-it data.
5) Acknowledge segment is present.	5) It doesn't have ackn-wledge segment.
6) It has variable header length of 20 to 60 bytes.	6) It has header length of 8 bytes.
7) It has extensive error detection and error correction mechanism.	7) It only implements basic error detectio mechanism.
It is slower.	8) It is faster.
It doesn't support broadcasting.	9) It supports broadca-ting

VoIP (Illegal).

10) It is used by HTTP, HTTPS, SMTP, etc

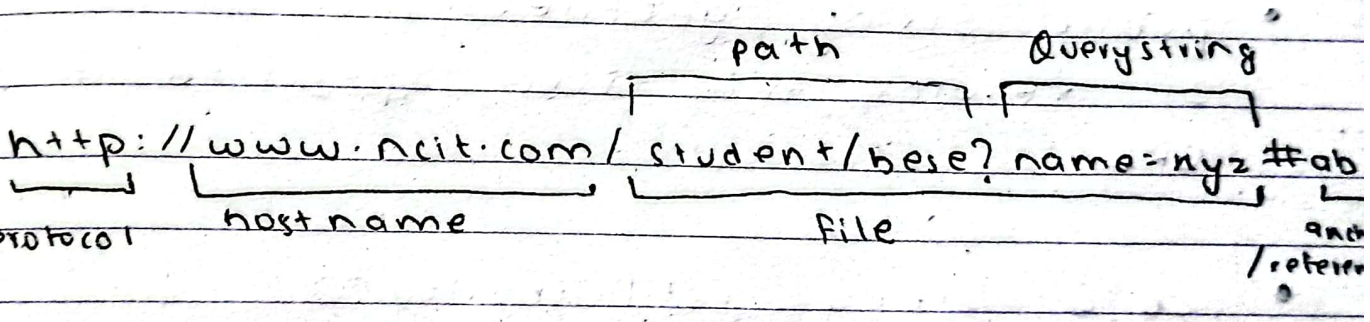
10) It is used by DNS, VoIP, live telecast, etc

2081/09/19
Friday

URL

→ It specifies the location of resource in a computer network.

→ In Java, URL is a class which is a part of java.net package that allows us to work with URLs.



Methods

- public String getProtocol()

→ It returns the protocol defined in the URL.

- public String getHost().

→ It returns the hostname of the URL.

- public int getPort()

→ It returns the port number specified in URL

→ If the port no is not specified, it returns -1.

- public int getDefaultPort()
→ It returns the default port no. based on the protocol eg for Http. returns 80.

- public String getPath()
→ It returns the path specified in the URL.
→ If path is not specified it returns null.

- public String getFile()
→ It returns the file specified in the URL.

- public String getAuthority()
→ It returns the authority of URL (combination of hostname and port number)
e.g. localhost:3000.

- public String getQuery()
→ It returns the query string of URL

- public String getRef()
→ It returns the reference / anchor (after #)

to use MalformedURLException we have to import java.io.*;
so simply (throws Exception).

Q WAP in Java to show URL processing.

```
import java.net.*;
class Example
{
    public static void main(String [] args)
    throws MalformedURLException
    { URL url = new URL("http://ncit.com
                /student?q=abc#smth");
      System.out.println("Protocol:" + url.getProtocol())
      System.out.println("Host :" + url.getHost());
      System.out.println("Port:" + url.getPort());
      System.out.println("Default Port:" + url.getDe-
                -faultPort());

      System.out.println("Authority:" + url.getAutho-
                -rity());
      System.out.println("Path:" + url.getPath());
      System.out.println("Query:" + url.getQuery());

      System.out.println("File:" + url.getFile());
      System.out.println("Ref:" + url.getRef());

    }
}
```

http
ncit.com
-1
443
ncit.com
/student
=abc
dent?q=abc
#smth

URLConnection class

This class provides us method that allows us to read or write from the resource specified by URL.

Q. WAP to display HTML code of "https://ncit.edu.np".

```
import java.io.*;
import java.net.*;
class Example
{
  public static void main(String []args)
  throws Exception
  {
    URL url = new URL("https://ncit.edu.np");

    URLConnection cn = url.openConnection();

    InputStream in = new InputStream(cn.getInputStream());
    int value;
    while((value = in.read()) != -1)
    {
      System.out.print((char) value);
    }
    in.close();
    cn.close();
  }
}
```

2081/09/25
Thursday

InetAddress

- It is a part of java.net package and it allows us to work with IP Addresses.
- It has two sub-classes:
  1) Inet4Address and 2) Inet6Address to work with IPV4 and IPV6 Address.

Q) WAP to print IP Address of ncit.edu.np.

```
import java.net.*;
class Example
{
    public static void main(String [] args)
    throws Exception
    {
        InetAddress inet = InetAddress.getByName
                ("www.ncit.edu.np");
        System.out.println("host:" + inet.getHostName
        System.out.println("IP Address:" + inet.getH
                -stAddress());

    }
}
```

6) WAP to all the IP Address associated with netflix.com. Also print the IP Address of localhost

```
import java.net.*;
class Example
{
  public static void main(String [] args)
 throws Exception
 {
   InetAddress [] inet = InetAddress.
                  getAllByName("www.netflix.com
   for(InetAddress i :inet)
   {
      System.out.println(i.getHostAddress()
   }

 // Localhost
   InetAddress local = InetAddress.getLoca
                        Host();
   System.out.println(" Name = " + local.getHo
                        -Name());
   System.out.println(" IP = " + local.getHostAddres
 }
}
```

RMI (Remote Method Invocation)

→ It is a java based technology that is used to create distributed application.

→ It allows an object on a client side to invoke methods of remote object (server)

Communication through (in) ~~RTM~~ RMI takes place through stub and skeleton.

Stub

It is an object at the client side that acts as an gateway for the client. All the requests from client are routed through stub.

⇒ The main responsibilities of stub are:

- initiate ~~remote~~ connection with remote JVM
- It bundles all the arguments and parameters required to invoke remote method which is known as marshalling.
- If a required argument of primitive data type, it adds header in the data. But if the argument is an object it searializes it.
- It is also responsible for receiving the response from the invoked method. (remote)

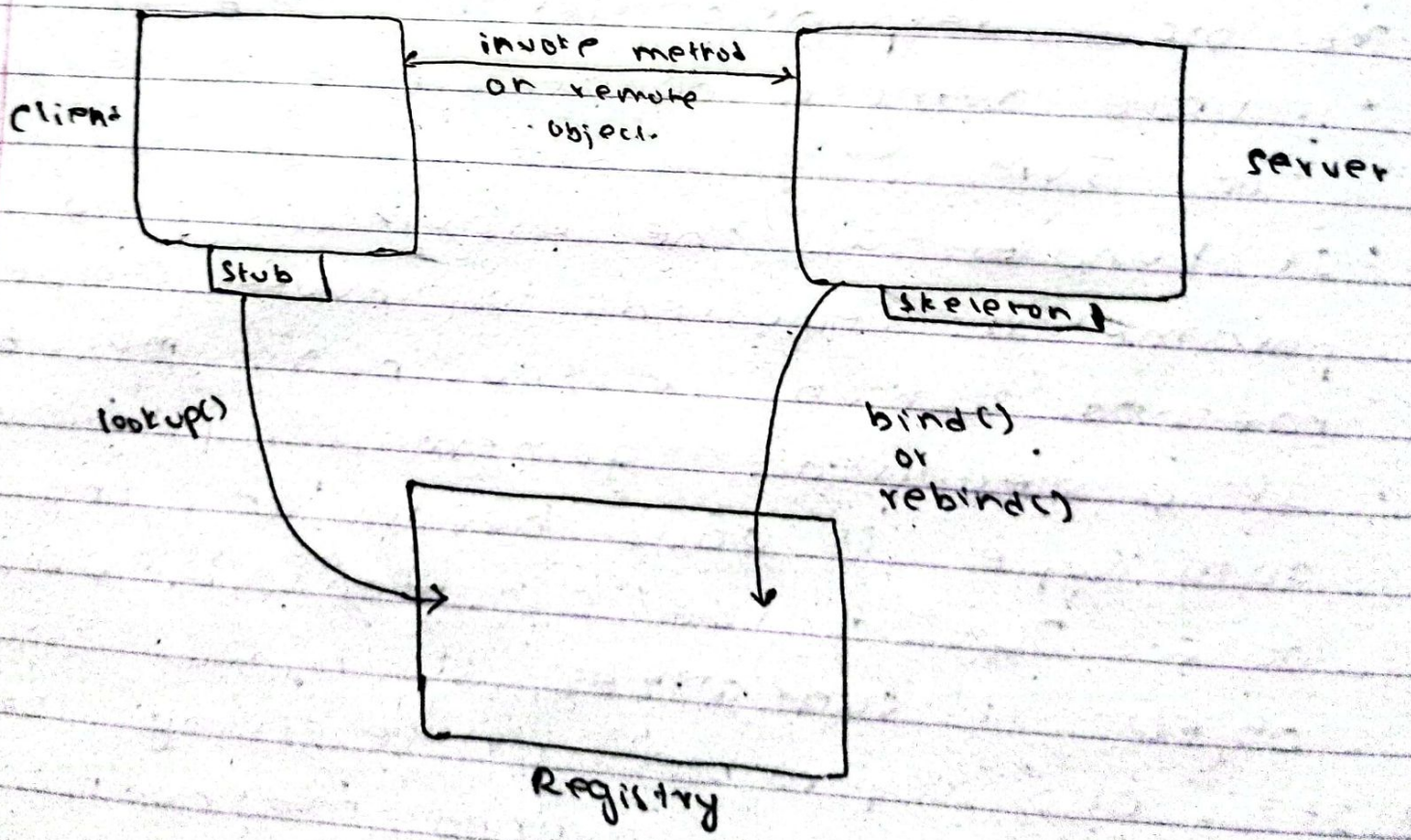
Skeleton

→ Skeleton is an object that acts a gateway for server side.

→ All the incoming request are routed through it

→ When the skeleton receives an incoming request it does the following:

- it reads the parameter for the remote method.
- it invokes the ~~obj~~ method on the remote object.
- It transmits or writes data back to the client.



Registry

- Registry is a namespace.
- All the objects from the server are placed in this namespace.
- Every object in the registry must be bound to a unique name which is known as binding name.
- Client can locate an object in registry using lookup() method.

Q Create an RMI application where client invokes a remote method that returns the sum of two integers.

Steps

- create a common interface that defines the methods.
- In server side provide the implementation of interface
- Bind the object in the registry.
- In client lookup() for the object and invoke the method.

2)

```
// common Interface
import java.rmi.*;
interface Maths extends Remote
{
    int sum(int x, int y) throws
        RemoteException;
}
```

```
import java.rmi.*;
import java.rmi.server.*;
class RemoteMaths extends UnicastRemoteO
 -ject implements Maths
{
    public RemoteMaths throws RemoteExce
    {
        super();
    }
    @Override
    public int sum(int x, int y)
    {
        return x+y;
    }
}
```

start rmiregistry 5000

```
//server
import java.rmi.*;
import java.rmi.registry.*;

class Server
{
    public static void main(String []args)
    throws Exception
    {
        Maths obj = new RemoteMaths();

        Naming.rebind("rmi://localhost:5000/smth",
                      obj);
        System.out.println("Server running");
    }
}
```

```
// Client
import java.rmi.*;

class Client
{
    public static void main(String []args)
    throws Exception
    {
        Maths stub = (Maths) Naming.lookup
                     ("rmi://localhost:5000/smth");
        int ans = stub.sum(9,10);
        System.out.println(ans);
    }
}
```

2021/09/26
Friday

Q) Create an RMI application where client can invoke following methods
(i) returns string in uppercase
(ii) returns length of string.
(iii) returns hello world

A)

```
// Common interface
import java.rmi.*;
interface Common extends Remote
{
    String strUp(String str) throws
    RemoteException;
    int strLength(String str) throws
    RemoteException;
    String sayHello() throws
    RemoteException;
}

// Implementation
Import java.rmi.*;
import java.rmi.server.*;

class RemoteCommon implements
Common extends UniCastRemoteObject
{
    public RemoteCommon()
    {
        super();
    }
```

```
public String strUp(String str)
{
        remo
        return str.toUpperCase();
}

public int strLength(String str)
{
        return str.length();
}

public String sayHello()
{
        return "Hello World";
}

}

// Server Side.

import java.rmi.*;
import java.rmi.registry.*;
class Server
{
    public static void main(String []args)
    throws Exception
    {
        Common obj = new RemoteCommon(
        Naming.rebind("rmi://localhost:5000/
                        sth", obj);
        System.out.println("Server is running");
    }
}
```

```
// Client side
    import java.rmi.*;
class class Client
{
    public static void main(String [] args)
    throws Exception
    {
        Client stub = new Stub();
        Common stub = (Common). Naming.
                    lookup("rmi://localhost:5000
        String
        System.out.println("Upper =" +
                    stub.strUp("rajan"));
        System.out.println("Length =" +
                    stub.strLength("Rajan"
        System.out.println("Say " + stub.say-
                    -Hello());

    }
}
```

CORBA (Common Object Request Broker Application).

- It is a middleware that provides mechanism to create distributed application.
- It is developed and managed by OMG (Open Management Group).

stub I skeleton I I

- It is platform & language independent.
- It provides IDL (~~It provi~~ Interface Definition Language) to provide implementation of common interface.

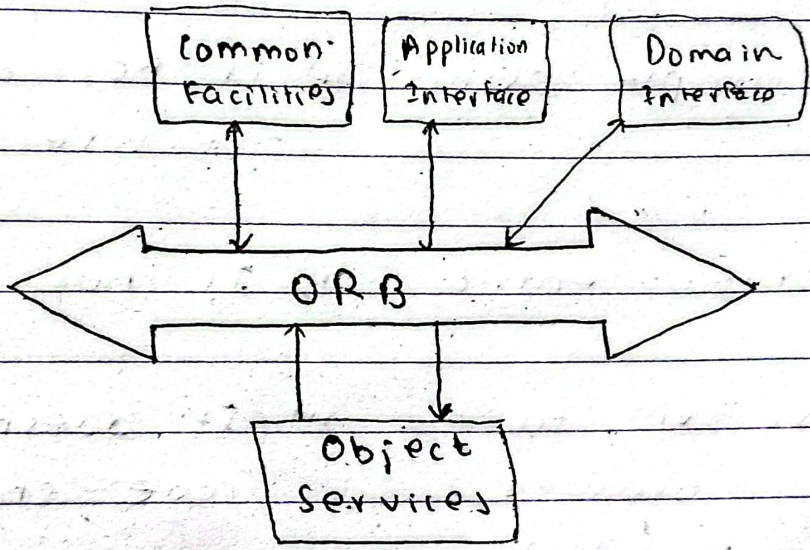


fig: OMG reference model.

Object Services

These are Domain independent interfaces that are used by distributed object programs, such as; the naming service - allows client to find object based on name

P2P

MT2

6) The trading services - Allows client to find object based on properties.

Comparision Between RMI and CORBA:

RMI	CORBA
i) It is language dependent i.e it can be implemented only using java.	i) It is language independent.
ii) It has simple architecture.	ii) It has complex architecture.
iii) It is server-centric.	iii) It has P2P archi
iv) It allows JVM to download libraries from remote JVM.	iv) It doesn't support code sharing.
v) It supports automatic Garbage collection.	v) It doesn't have Garbage collector.
vi) Interface must be defined using Java.	vi) common interface written in IDL.
vii) It doesn't have any sp. security mechanism.	vii) It has sp. security mechanism

viii) JVM is responsible for locating an object.

viii) Object Adapter are used for locating object.

ix) It uses JRMP (Java Remote Method Protocal).

ix) It uses Internet Inter-ORB Protocol.

Java Mail API

→ It is platform independent as well as protocol independent java framework that allows us to write, compose, read electronic mails.

→ All the classes of this interface are in javax.mail and javax.mail.activation package.

→ Protocols for Email.

1) SMTP (Simple Mail Transfer Protocol)
It provides mechanism to deliver email.

2081/10/03
Thursday

Servlet / JSP

Q) Create a web application using servelet where user sends a number and servlet should responds whether it is even or odd.

index.html

```
<html>
<body>
<form action = "check">
    Enter num: <input type = "text" name
                = "num">
    <br>
    <input type = "submit" value = "check">
</form>
</body>
</html>
```

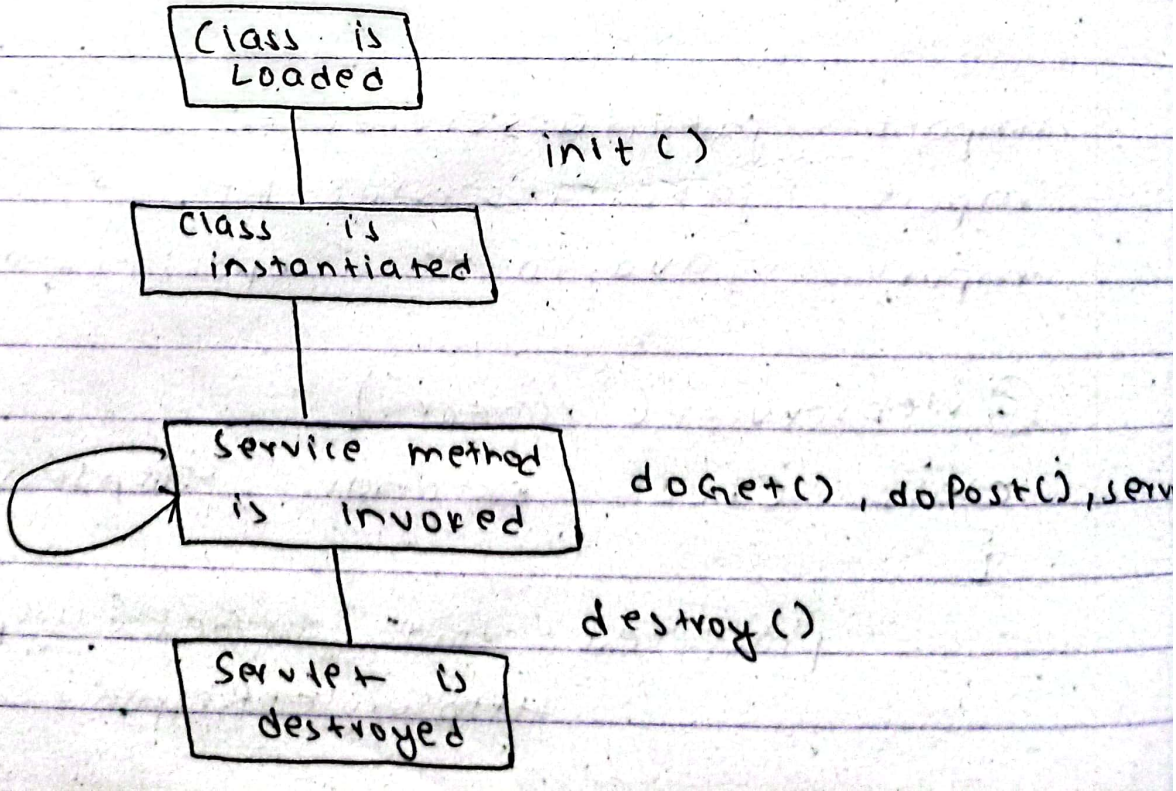
```
import jakarta.servlet.*;
import jakarta.servlet.http.*;
import java.io.*; import jakarta.servlet.annotation
```

```
@WebServlet("/check")
class Check extends Httpservlet
{
    protected void doGet(HttpServletRequest req
            HttpServletResponse res) throws
```

```
ServletException, IOException
{
    int n = Integer.parseInt(req.getParameter("num"));

    String ans;
    if (n%2==0)
    { ans = "Even"; }
    else
    { ans = "Odd"; }
    PrintWriter out = res.getWriter();
    out.print("<p>" + ans + "</p>");
  }
}
```

Servlet

Servlet is a java technology that is used to create dynamic web application.



6) Create a servlet that displays total number of visitors (hit counter)

```
import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
import java.io.*;
@WebServlet("/")
class Counter extends HttpServlet
{
    private int count;
    public void init()
    {
        count = 0;
    }
    protected void doGet(HttpServletRequest req
        HttpServletResponse res) throws
        IOException, ServletException
    {
        res.setContentType("text/html");
        count ++;

        PrintWriter out = res.getWriter();
        out.println("<b> Visit : " + count +
                    "</b>");
    }
    public void destroy()
    { }
}
```

Servlet runs inside a container in a web server and it handles requests and response

HTTP Request

1) GET Request

This request is used to fetch a single resource or group of resource from a server.

A suriessful get request is responded with 200 → 2xx statuscode.

We can also send data using GET request but such data are visible in URL as a parameter or query string.

2) POST Request

This request is used to send data to the server. It accepts request body. i.e data are send through request body.

It is used to create resource in the backend.

DELETE Request

It is used to remove resource or group of resource from backend.

4) PUT Request

-> It is used to update an existing resource.

-> It completely replaces the existing resource with request body.

5) PATCH Request

- It is also used to update an existing resource.

-> It accepts request body and partially replaces the existing resource.

Beside these, there are other http (ve requests such as: HEAD, TRACE.

HTTP response code

1) 1xx (Informational Response)
Response code starting from '1xx' is sent when server is in processing state.
e.g 100 (continue)
It means client should continue the request
- 101 (Switching Protocol)

2) 2xx (Success Response)
When a user request is successful the response is accompained by 2xx.
e.g. 200 (OK): The GET req. is successful.
201 (Created): POST req is successful and new resource is created.

3xx (Redirection Message)
It indicates redirection.
e.g. 300 (Multiple Choice)
→ indicates that request has more than one possible response.

301 (Moved Permanently)
→ it means the URL of the requested resource has been changed permanent

→ 307 (Temporary Redirection)
→ 308 (Permanent Redirection).

4) 4xx (Client Error Response)
→ indicates there has been error while processing the req. and the error is due to client.

Eg. 400 (Bad Request)
→ It means a server cannot or will not process the req. due to client error.

401 (Unauthorized)
→ The resource requested by the user is protected and client doesn't have required authorization

402 (Payment Required)
403 (Forbidden)
404 (Page Not Found)

5) 5xx (Server Error Response)
→ Indicates error due to server's inability to process request.

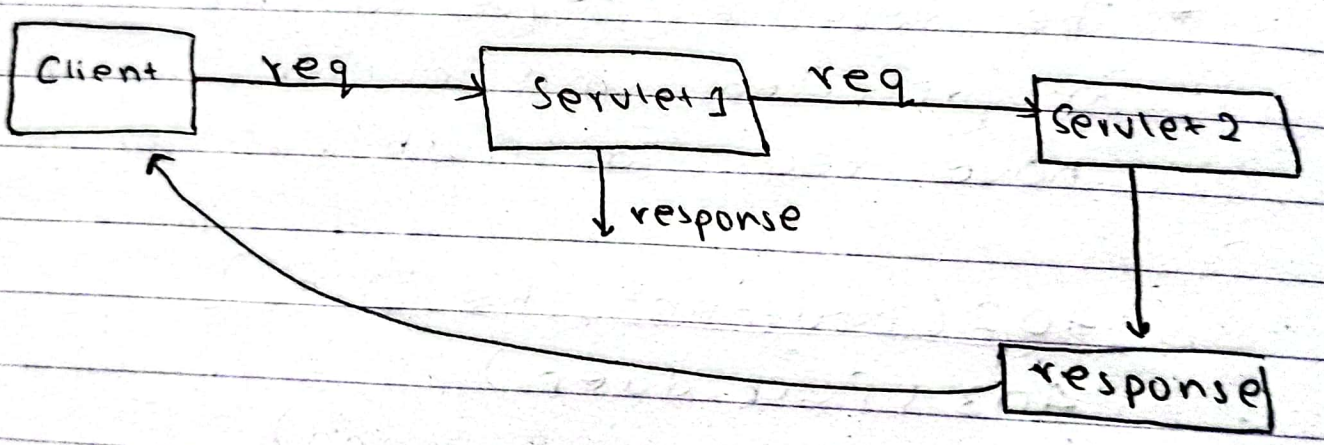
500 (Internal Server Error)
504 (

Request Dispatcher

It is an interface that allows us to dispatch a request to another resource which may be a servlet, JSP or HTML

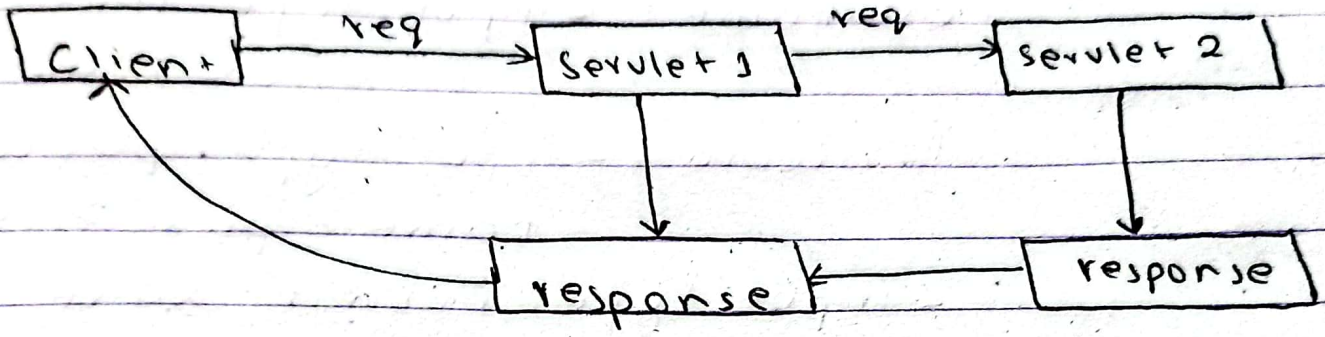
Methods

1) public void forward(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException



Using this method, a request can be forwarded into another method but the response of initial servlet is ignored. and only the response of last servlet is sent to the client

2) public void include (HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException.



This method also allows us to dispatch request from one resource to another but unlike the forward method it also allows us to include response of ~~servlet~~ One servlet with another.

```
<html>
<body>
    <form action: /register method: "post" >

        Username : <input type: text" name
                   = "username" > <br>
        Password : < input type = "password"
                   name = "pwd" > <br>
        < input type = "submit" value = "signup">

    </ form>
</ body>
</ html>


//Database. Conn & query.
import java.sql.*;
class Database
{
    public static boolean insert (string
                uname, string pwd) throws
    Exception
    {
        Class.forName ("com.msql.cj.jdbc.Driver");
        Connection cn = DriverManager.getConnectio
                ("jdbc:mysql://localhost:3306/
                    ncit", "root", "");
        PreparedStatement ps = cn.preparestatement
```

```
"Select * from student where
        uname = ? " );
ps.setString (1, uname);

ResultSet rs = ps.executeQuery();
if (rs.next())
{   return false;
}

Prepo
ps = cn.prepareStatement(" Insert into
        student values (?, ?) " );

int
ps.setString (1, uname);
ps.setString (2, pwd);

int a = ps.executeUpdate();
if(i==1)
{
    return true;
}
return false;

}
}
```

```
import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
import java.io.*;

@WebServlet("/register")
class Register extends HttpServlet
{
    protected void doPost(HttpServletRequest
            req, HttpServletResponse res)
    throws ServletException, IOException
    {
        res.setContentType("text/html");
        String uname = req.getParameter("usern
        String pwd = req.getParameter("pwd");

        PrintWriter out = res.getWriter();

        try
        {
            if( Database.insert(uname, pwd))
            {
                RequestDispatcher rd = req.getReque
                                    Dispatcher("/dashbo

                rd.forward(req, res);
            }
            else
            {
```

```
out.print("<p style='color:red;'>
            User already exists</p>");

RequestDispatcher rd = req.getRequestDis-
                -patcher("/index.html");
rd.include(req, res);
    }
} catch(Exception e)
{
 }
}
}
// Dashboard

@WebServlet("/dashboard");
Class Dashboard extends HttpServlet
{
protected void service(HttpServletRequest
      req, HttpServletResponse res)
throws ServletException, IOException
{
    res.setContentType("text/html");
    PrintWriter out = res.getWriter();
    out.print("<p> Welcome User </p>");
}
}
```

2081/10/07

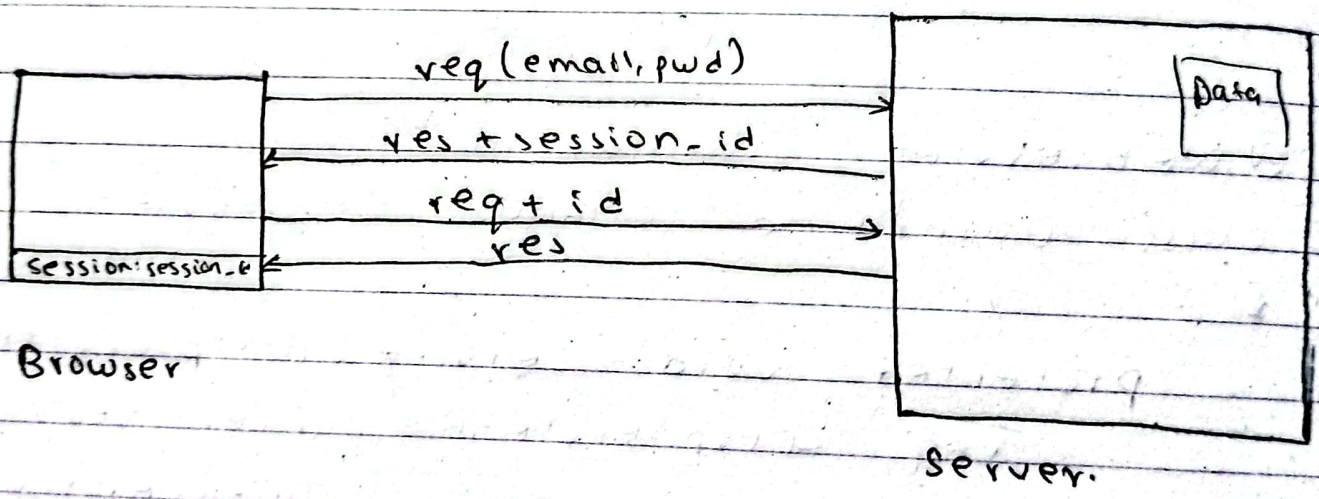
Session

→ Session is a session management technique used by web application where user data are stored in server.

→ When a session is created for user, unique session id is generated which is sent to the client through response.

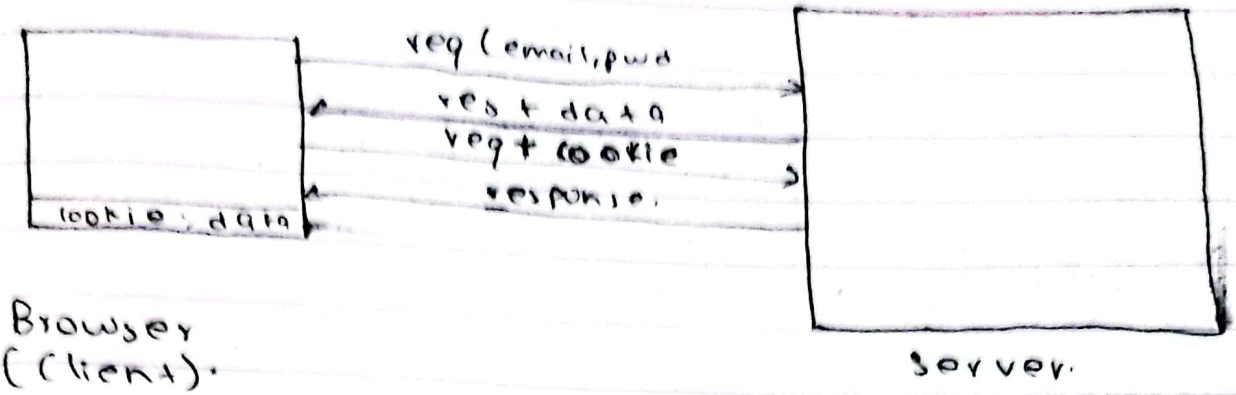
→ This session id is stored in session cookie in the user's device.

→ Every successive request is attached with session id which helps server to track the user's session.



Cookie

→ Cookie is a small text file that is stored in user's browser or device which helps to track user's activity in a web application.



Session	Cookie
1) Data are stored in server side.	1) Data are stored in client side.
2) It can store any kind of data.	2) It can only store text data
3) It is safer.	3) It can be easily tempered, relatively unsafe.
4) Theoritically there is no limit to session storage capacity. But due to limitation of scripting language, the max. session storage is 128 MB.	4) It's max. storage limit is 4KB
5) Session storage is destroyed when user logs out or closes the browser or tab.	5) Cookie is destroyed by the time specified ~~during~~ while making the cookie.
6) Sensitive data is stored in session.	6) Less sensitive data is stored in cookie.

Q) Create a servlet application using session that displays the date and time of first visit, last visit and the ~~no~~ total visit for each user.

A)
```
import java.util.*;
import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
import java.io.*;

@WebServlet("/")
public class visit extends HttpServlet
{
    protected void doGet( HttpServletRequest req,
        HttpServletResponse res) throws IOException,
        ServletException
    {
        res.setContentType("text/html");
        Http Session ssn = req.getSession();
        // Creates new session if session doesn't already exist
        int count;
        Date creation = new Date(ssn.getCreation
                        Time());
        Date lastAccess = new Date(ssn.get
                        LastAccessedTime() );
        if( ssn.isNew())
        {   count = 1;
            ssn.setAttribute("visit", count);
        }
```

```
else
{
    count = (Integer) ssn.getAttribute("visit");
    count++;
    ssn.setAttribute("visit", count);
}
PrintWriter out = res.getWriter();
out.println("<p> First visit" + creation +
                "</p>");
out.print("<p> Last visit" + lastAccess +
                "</p>");
out.print("<p> Total visit" + count + "</p>");
}
}
```

2081/10/10
Thursday

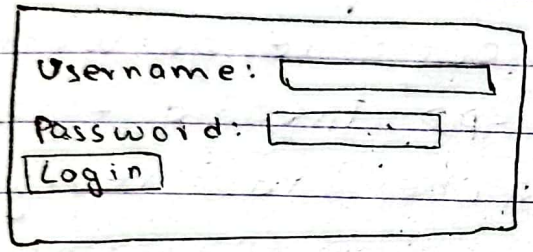
Q) Create a login application using servlet.

⇒ If user is authenticated, store the username in session. and redirect the user to dashboard. In dashboard display the username in dashboard and also display a logout button in dashboard.

→ Destroy the session when user logout.

→ Also prevent the user from directly visiting dashboard without authentication.

⇒



For auth

```
import
.
.
.
@WebServlet("/auth")
public class Auth extends HttpServlet
{
    protected void doPost(HttpServletRequest req,
                          HttpServletResponse res) throws
    ServletException, IOException
    {
        res.setContentType("text/html");
        String uname = req.getParameter("uname");
        String pwd = req.getParameter("pwd");
```

```
        if( user
         if ( uname.equals("abc") && pwd.equals("123"))
        {
            HttpSession ssn = req.getSession();
            ssn.setAttribute("user", uname);

            res.sendRedirect("/dashboard");
        }
        else
        {
            PrintWriter out = res.getWriter();
            out.print("<p> Invalid </p>");
           RequestDispatcher rd = req.getRequest-
                                  -Dispatcher("/index.html");
            rd.include(req, res);
        }
    }
}

// Dashboard.
import
@WebServlet("/dashboard")
public class Dashboard extends HttpServlet
{
    protected void service( HttpServletRequest req,
        HttpServletResponse res) throws IOException,
        ServletException
    {
        HttpSession ssn = req.getSession(false);
```

```
PrintWriter out = res.getWriter();
if (ssn==null)
{
    RequestDispatcher rd = req.getRequest
            Dispatcher("/index.html");
    out.print("<p> Auth required </p>");
    rd.include(req, res);
}
out.println("<p> Hello " + ssn.getAttribute
        ("user") + "</p>");

out.println("<form action='logout'>
        <input type='submit' value=
            'Logout'>
        </form>");
    }
}

// Logout
import
:
@WebServlet("/logout")

public class Logout extends HttpServlet
{
    protected void service(HttpServletRequest req,
        HttpServletResponse res) throws IOException,
    ServletException
    {
        HttpSession ssn= req.getSession();
```

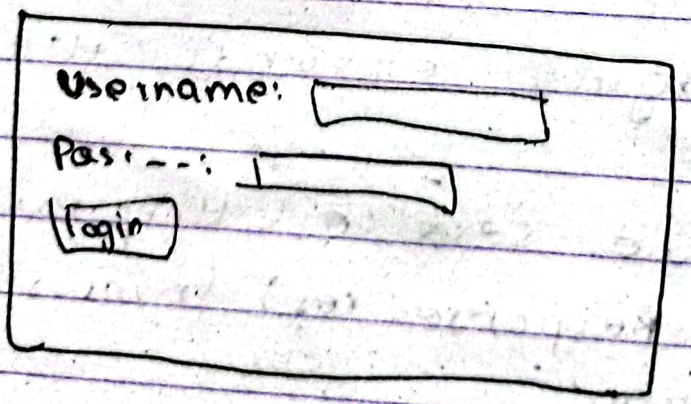
```
PrintWriter out = res.getWriter();
ssn.invalidate(); // destroy session.

out.println("<p> You have logged out
            </p>");

    }

}
```

Cookie

→ It is also a technique used for session management.

→ A cookie is a small text file that is stored in a user's browser or device.

Login Using Cookie (similar as prev. program)



```
// Auth
import .....
;

@WebServlet ("/auth")
public class Auth extends HttpServlet
{
    protected void service( ..... ) throws .....
    {
        String user = req.getParameter("uname");
        String pwd = req.                    ("pwd");
        if( user.equals("abc") && pwd.equals("123")
        {
            Cookie ck = new Cookie("user", user);
            res.addCookie(ck);
            res.sendRedirect("/dashboard");
        }
        else
        {
            same as अगाडि को program
        }
    }
```

```
for(Cookie c : ck)
{
    sop(c.getName());
    sop(c.getValue());
}
```

// Dashboard ["user" : abc]

import

@WebServlet("/dashboard")

public class Dashboard extends HttpServlet
{

protected void service(....) throws ....
{

Cookie [] ck = req.getCookies();
String name = ck[0].getValue();

if(name==null || name="")
{
//Redirection code same as [illegible]
}

Print out = res.getWriter();
out.print(" Hello" + name + " ")
out.print("<form action='logout'>
<input type='submit' value
'logout'>
</form>");

}

}

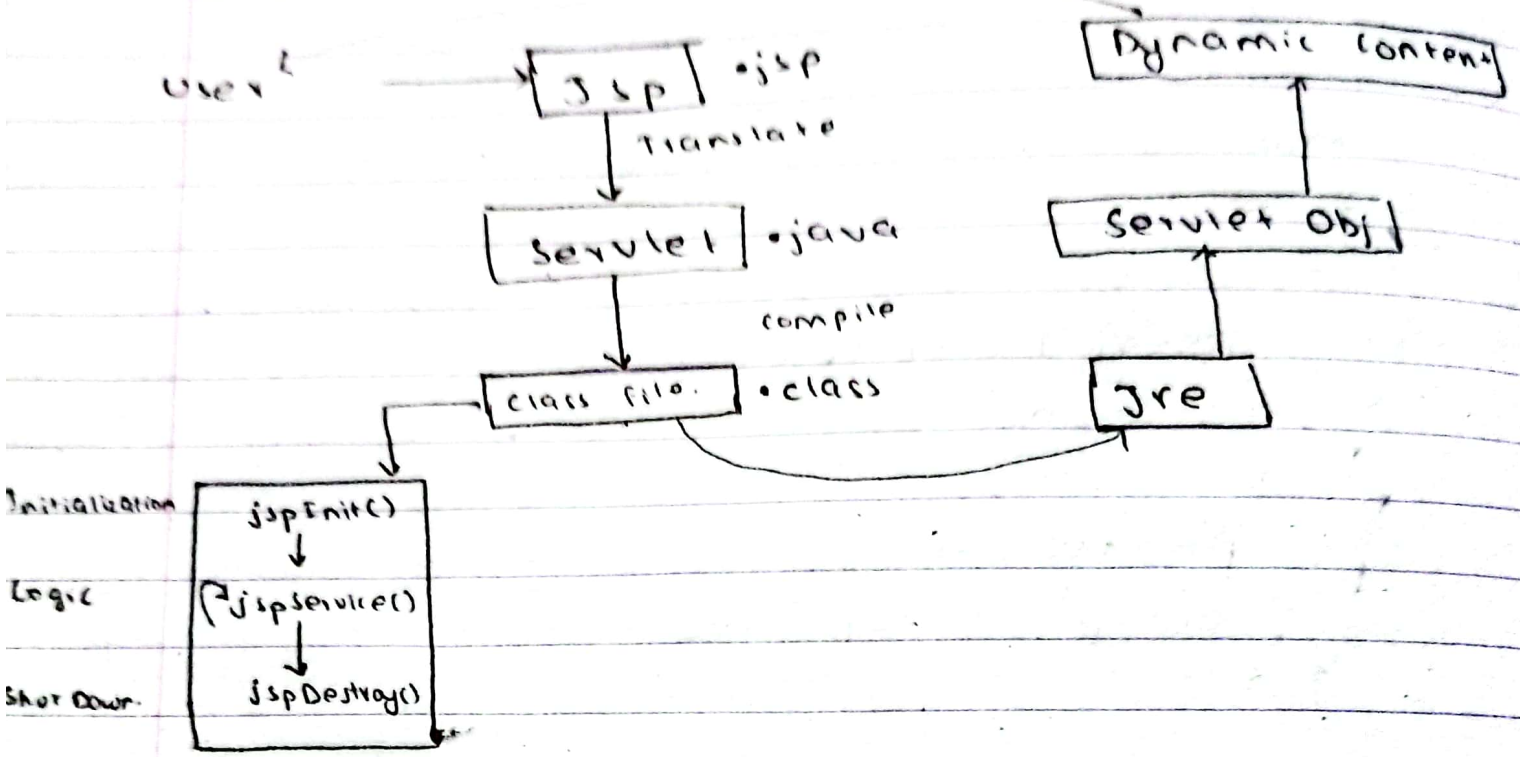
// logout

```
import

@webservlet("/logout")
public class Logout extends HttpServlet
{
    protected void service(....) throws ....
    {
        Cookie ck = new Cookie("user", "");
        ck.setMaxAge(0);
        res.addCookie(ck);

        PrintWriter out = res.getWriter();
        out.print("<p> You have logged out </p>");
    }
}
```

2081/10/13
Friday

JSP

- Java server page and new name is Jarata server page.
- JSP is a java code embedded in html.



JSP Tags

1) Scriplet Tags <% %>
2) <%= %> Expression
3) <%! %> Declaration
4) <%@ %> Directive.

Implicit Objects

- request
- response
- out
- session.

Q Using JSP display the following
1) Date and Time of first access
2) Date and Time of last visit
3) Total no of visit for each user. (using session)

index.jsp

```
<html>
<head> <title> JSP </title> </head>
<body>
    <%  if (session.isNew()){
            session.setAttribute("visit", 1);
        }else{
        int count = (Integer) session.get-
                    -Attribute("visit");
        count++;
        session.setAttribute("visit", count);
        } %>
    <%@ page import="java.util.*" %>

    <%
        Date first = new Date(session.getCrea-
                    -tion.Time());
        Date last = new Date(session.getLast-
                    -Accessed Time());
    %>
    <p> First visit = <%= first %> </p>
    <p> Last access = <%= last %> </p>
    <p> Visits = <%= session.getAttribute("visit"); %> </p>
</body>
</html>
```

Difference between Servlet and JSP

Servlet	JSP
i) It consists of pure java code only.	i) It is java code embedded in HTML.
ii) Servlet is faster.	ii) It is slower since it translated into servlet.
iii) We can override service().	iii) We cannot override service() method.
iv) Session management is not initialized by default.	iv) Session management is created beforehand (by default).
v) There is no provision for tags.	v) JSP has scriplet, declaration, expression and directive tags.
vi) It is ideal for web application that needs extensive data processing.	vi) It is not ideal for web application that needs extensive data processing.
vii) In MVC pattern, servlet acts as controller.	vii) In MVC pattern, JSP acts as view.
viii) There are no implicit objects in servlet.	viii) JSP has 9 implicit objects that can be used without declaration e.g. request, response, etc.

iii) any changes requires recompilation.

iv) Changes in html doesn't require recompilation.

Design Pattern

Singleton

Eager Instantiation

```
class Singleton
{
    private static final singleton instance =
                        new singleton();

    private singleton() { }

    public static singleton getInstance()
    {
        return instance;
    }
}

class Test
{
    public static void main(String [] args)
    {
        Singleton s1 = Singleton.getInstance();
        Singleton s2 = Singleton.getInstance();
        if(s1.hashCode() == s2.hashCode())
        {
            System.out.println(" same");
        }
    }
}
```

Lazy Instantiation

```
private static Singleton instance;
:
public static Singleton getInstance()
{
    if (instance == null)
    {
        instance = new Singleton();
    }
    return instance;
}
```