Date:

PRACTICAL-1

Aim: Create chat application using TCP protocol.

Code:

Client code:

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.PrintWriter;
import java.net.Socket;
import java.util.Scanner;


public class Client {
 public static void main(String[] args) throws IOException {
  int port=444;
Socket socket = new Socket("127.0.0.1",port);


while(true)
{
System.out.println("started Client");
OutputStream ostream = socket.getOutputStream();
```

```
InputStream istream = socket.getInputStream();


BufferedReader receive = new BufferedReader(new InputStreamReader(istream));
PrintWriter pwrite = new PrintWriter(ostream);
    String send,r;
    Scanner scan= new Scanner(System.in);
     System.out.print("client:");
     send=scan.nextLine();
     pwrite.println(send);
     pwrite.flush();
     while((r= receive.readLine())!=null)
    {
 System.out.println("server:"+r);
  System.out.print("client:");
  send=scan.nextLine();
 pwrite.println(send);
 pwrite.flush();
    }
   }
  }
}
```

Server code:

```
import java.io.*;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Scanner;
public class Server {
    public static void main(String[] args) throws IOException {
int port=444;
ServerSocket serversocket = new ServerSocket(port);
System.out.println("started Server");
while(true)
{
Socket clientsocket = serversocket.accept();
System.out.println("getting data ");
InputStream istream = clientsocket.getInputStream();
OutputStream ostream = clientsocket.getOutputStream();
PrintWriter pwrite = new PrintWriter(ostream);
BufferedReader receive = new BufferedReader(new InputStreamReader(istream));
String r;
 String send;
 Scanner scan= new Scanner(System.in);
while((r= receive.readLine())!=null)
{
 System.out.println("client:"+r);
```

```
 System.out.print("server:");
 send=scan.nextLine();


pwrite.println(send);
   pwrite.flush();
      }
    }
   }
}
```

Output:

Server :

```
run:
started Server
getting data
client:hello!
server:hi
client:i am client
server:i am server
client:exit
```

Client :

```
run:
started Client
client:hello!
server:hi
client:i am client
server:i am server
client:exit
```

Sign:

Date:

PRACTICAL-2

Aim: Implement TCP server for transferring files using Socket and ServerSocket.

Code:

SERVER:

```
import java.io.BufferedReader;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Scanner;
public class Prac2 {
   public static void main(String[] args) throws IOException,
InterruptedException {
    int port=444;
    int byteRead =0;
ServerSocket serversocket = new ServerSocket(port);
if(!serversocket.isBound())
        System.out.println("Sever Socket not Bounded...");
```

```
        else
            System.out.println("Server Socket bounded to Port :
"+serversocket.getLocalPort());
Socket clientsocket = serversocket.accept();
if(!clientsocket.isConnected())
            System.out.println("Client Socket not Connected...");
        else
         System.out.println("Client Socket Connected
:"+clientsocket.getInetAddress());
int x=0;
while(x==0)
{
System.out.println("getting a file");
InputStream istream = clientsocket.getInputStream();
 OutputStream os = new FileOutputStream("Test.txt");
BufferedReader receive = new BufferedReader(new
InputStreamReader(istream));
    byte[] byteArray = new byte[100];
  while((byteRead = istream.read(byteArray, 0, byteArray.length))!= -1 )
            {
               os.write(byteArray, 0, byteRead);
        System.out.println("No. of Bytes Received : "+byteRead);
            }
            synchronized(os){
```

```
            os.wait(100);


            }
             os.close();
            serversocket.close();
            System.out.println("File Received...");
            x=1;
        }
}
}
```

Client:

```
import java.io.BufferedInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.OutputStream;
import java.net.Socket;


public class client {
   public static void main(String[] args)
{
   Socket socket;
   try
   {
```

```
socket = new Socket("127.0.0.1", 444);
if(!socket.isConnected())
   System.out.println("Socket Connection Not established");
else
   System.out.println("Socket Connection established : "+
         socket.getInetAddress());

File myfile = new File("testprgm.txt");



if(!myfile.exists())
   System.out.println("File Not Existing.");
else
   System.out.println("File Existing.");

byte[] byteArray = new byte[1024];

FileInputStream fis = new FileInputStream(myfile);
BufferedInputStream bis = new BufferedInputStream(fis);
OutputStream os = socket.getOutputStream();
int trxBytes =0;
while((trxBytes = bis.read(byteArray, 0, byteArray.length)) !=-1)
{
```

```
            os.write(byteArray, 0, byteArray.length);
            System.out.println("Transfering bytes : "+trxBytes );
            }
            os.flush();
            bis.close();
            socket.close();

            System.out.println("File Transfered...");
        }
        catch(Exception e)
        {
            System.out.println("Client Exception : "+e.getMessage());
        }
    }
    }
```

Output:

Server:

```
run:
Server Socket bounded to Port : 444
Client Socket Connected : /127.0.0.1
getting a file
No. of Bytes Received : 43
File Received...
```

Socket:

```
run:
Socket Connection established : /127.0.0.1
File Existing.
Transfering bytes : 43
File Transfered...
BUILD SUCCESSFUL (total time: 0 seconds)
```

Sign:

Date:

PRACTICAL-3

Aim: Implement any one sorting algorithm using Socket using TCP/UDP on Server application and Give Input on client side and client should sorted output from server and display sorted on client side.

Code:

Server:

```
import java.io.BufferedInputStream;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.OutputStream;
import java.net.ServerSocket;
import java.net.Socket;

public class Prac3 {

   public static void main(String[] args) throws IOException,
ClassNotFoundException {
   ServerSocket sc= new ServerSocket(999);
   Socket s= sc.accept();
   if(s.isConnected())
   System.out.println("Server connected with a client");
   else
       System.out.println("problem to connect with a client");
  if(s.isBound())
  {
      OutputStream so=s.getOutputStream();
      ObjectOutputStream out= new ObjectOutputStream(so);// for sending
object
```

```
    Selectionsort h= new Selectionsort();
    InputStream si = s.getInputStream();
    ObjectInputStream in= new ObjectInputStream(si);//for getting object
    int r[];
    r=(int[]) in.readObject();
  if(r != null)
    {
   h.sort(r);
   out.writeObject(r);
   r=null;
    }
  s.close();
}
}

private static class Selectionsort {

    public Selectionsort() {
    }

    void sort(int arr[])
{
    int n = arr.length;
    for (int i = 0; i < n-1; i++)
    {
    int min_idx = i;
        for (int j = i+1; j < n; j++)
            if (arr[j] < arr[min_idx])
                min_idx = j;

        int temp = arr[min_idx];
        arr[min_idx] = arr[i];
        arr[i] = temp;
    }
```

```
    }
    }

}
```

Client:

```
import java.io.IOException;
import java.io.InputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.OutputStream;
import java.net.Socket;
import java.util.Scanner;

public class client {
  public static void main(String arg[]) throws IOException,
ClassNotFoundException
  {
  Socket sc= new Socket("127.0.0.1",999);
  Scanner scan= new Scanner(System.in);
  if(sc.isConnected())
  System.out.println("Connected to server");
  else
     System.out.println("Something Wrong");

  if(sc.isConnected())
  {

      OutputStream so=sc.getOutputStream();
     ObjectOutputStream out= new ObjectOutputStream(so);

     InputStream si = sc.getInputStream();
     ObjectInputStream in= new ObjectInputStream(si);
//for getting object
```

```
   System.out.println("Enter number of data");
   int n= scan.nextInt();
   int a[]= new int[n];
     System.out.println("Enter the data");
   for(int x=0;x<n;x++)
   {
   a[x]=scan.nextInt();
   }
   out.writeObject(a);

   a=   (int[]) in.readObject();
    System.out.println( "sorted array is:");
    for(int x=0;x<n;x++)
   {
  System.out.print( a[x]);
   }
   sc.close();
   }
   }
}
```

Output:

Sever:

```
run:
Server connected with a client
BUILD SUCCESSFUL (total time: 11 seconds)
```

Client:

```
run:
Connected to server
Enter number of data
5
Enter the data
52
4
2
6
8
sorted array is:
2 4 6 8 52 BUILD SUCCESSFUL (total time: 8 seconds)
```

Sign:

Date:

PRACTICAL-4

Aim: Implement Concurrent TCP Server programming in which more than one client can connect and communicate with Server for sending the string and server returns the reverse of string to each of client.

Code:

Server:

```
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.logging.Level;
import java.util.logging.Logger;

public class Prac4 {


    public static void main(String[] args) throws IOException {
        ServerSocket ss = new ServerSocket(5056);
        while (true)
        {
            Socket s = null;

            try
            {

                s = ss.accept();

                System.out.println("A new client is connected : " + s);

                // obtaining input and out streams
                DataInputStream dis = new DataInputStream(s.getInputStream());
```

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

            System.out.println("Assigning new thread for this client");

            // create a new thread object
            Thread t = new ClientHandler(s, dis, dos);

            // Invoking the start() method
            t.start();

         }
         catch (Exception e){
            s.close();
            e.printStackTrace();
         }
      }
   }
}

// ClientHandler class
class ClientHandler extends Thread
{

   final DataInputStream dis;
   final DataOutputStream dos;
   final Socket s;


   // Constructor
   public ClientHandler(Socket s, DataInputStream dis, DataOutputStream
dos)
   {
      this.s = s;
```

```
        this.dis = dis;

 this.dos = dos;
    }

    @Override
    public void run()
    {
        StringBuffer received;
        String toreturn;
        while (true)
        {
            try {
                // receive the answer from client
                received =new StringBuffer( dis.readUTF());

                if(received.equals("Exit"))
                {
                    System.out.println("Client " + this.s + " sends exit...");
                    System.out.println("Closing this connection.");
                    this.s.close();
                    System.out.println("Connection closed");
                    break;
                }

       toreturn =received.reverse().toString();

                  dos.writeUTF(toreturn);
                // write on output stream based on the
                // answer from the client
                } catch (IOException ex) {

            }
```

```
        try
        {
            // closing resources
            this.dis.close();
            this.dos.close();

        }catch(IOException e){
            e.printStackTrace();
            }

        }
    }
```

Client:

```
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.net.InetAddress;
import java.net.Socket;
import java.util.Scanner;

public class client {
    public static void main(String Arg[])
    {

     try
        {
            Scanner scn = new Scanner(System.in);

            // getting localhost ip
            InetAddress ip = InetAddress.getByName("localhost");

            // establish the connection with server port 5056
            Socket s = new Socket(ip, 5056);
```

```
        // obtaining input and out streams
        DataInputStream dis = new DataInputStream(s.getInputStream());
        DataOutputStream dos = new
DataOutputStream(s.getOutputStream());

        // the following loop performs the exchange of
        // information between client and client handler
        while (true)
        {
         System.out.println("ENter string you want to reverse");
           String tosend = scn.nextLine();
           dos.writeUTF(tosend);


           if(tosend.equals("Exit"))
           {
              System.out.println("Closing this connection : " + s);
              s.close();
              System.out.println("Connection closed");
              break;
           }

           // printing date or time as requested by client
           String received = dis.readUTF();
           System.out.println(received);
        }

        // closing resources
        scn.close();
        dis.close();
        dos.close();
      }catch(Exception e){
        e.printStackTrace();
      }
```

```
    }
}
```

Output:

Server:

```
run:
A new client is connected : Socket[addr=/127.0.0.1,port=8601,localport=5056]
Assigning new thread for this client
```

Client:

```
run:
ENter string you want to reverse
mynameismayank
knayamsiemanym
ENter string you want to reverse
```

Sign:

Date:

PRACTICAL-5

Aim: Write RMI application where client supplies two numbers and server response by summing it. Provide your custom security policy for this application.

Code:

Addc.java (interface):

```
import java.rmi.Remote;

public interface Addc extends Remote {

 public int add(int x,int y) throws Exception;
}
```

Addprog.java (class):

```
import java.rmi.server.UnicastRemoteObject;

public class Addprog extends UnicastRemoteObject implements Addc{

 Addprog() throws Exception
 {
  super();
 }
 public int add(int x,int y)
 {
  return x+y;
 }
}
```

Server:

```
import java.rmi.Naming;
import java.rmi.registry.LocateRegistry;
```

```
public class Prac5RMI {
    public static void main(String[] args) throws Exception {

 Addc obj= new Addprog();
    LocateRegistry.createRegistry(1900);
  Naming.rebind("rmi://localhost:1900/Adder",obj);//skelton
  System.out.println("Server Started");
 }
}
```

Client:

```
import java.rmi.*;
import java.util.Scanner;

public class client
{
 public static void main(String args[]) throws Exception
 {
  Addc obj=(Addc) Naming.lookup("rmi://localhost:1900/Adder");
  Scanner scan=new Scanner(System.in);
  System.out.println("Enter 2 numbers to do addition:");
  int a=scan.nextInt();
  int b=scan.nextInt();
  int sum=obj.add(a,b);
  System.out.println("Addition :\t" + sum);
 }}
```

Output:

Server:

```
run:
Server Started
```

Client:

```
run:
Enter 2 numbers to do addition:
50
40
Addition :      90
BUILD SUCCESSFUL (total time: 6 seconds)
```

Sign:

Date:

PRACTICAL-6

Aim: Implement Student information system using JDBC and RMI. JDBC/Servlet

Code:

StudentI.java:

```
import java.rmi.Remote;
import java.sql.*;
import java.util.*;

public interface StudentI extends Remote {


 public abstract ArrayList insert(int id,String name,String branch,int atd) throws Exception;

}
```

StudentC.java:

```
import java.rmi.server.UnicastRemoteObject;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.ArrayList;

public class StudentC  extends UnicastRemoteObject implements StudentI{

   public StudentC() throws Exception
 {
  super();
```

```
 }
 public ArrayList insert(int id,String name,String branch,int atd) throws
Exception
 {
  ArrayList ar=new ArrayList();
  Class.forName("com.mysql.jdbc.Driver");
  Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/test","root
","");
  Statement stmt=con.createStatement();
  stmt.executeUpdate("insert into student
values("+id+",'"+name+"','"+branch+"',"+atd+")");
  ResultSet rs=stmt.executeQuery("select * from student where id="+id);
  rs.next();
  int Id=rs.getInt(1);
   name=rs.getString(2);
   branch=rs.getString(3);
   atd=rs.getInt(4);

   ar.add(new Integer(Id));
  ar.add(name);
  ar.add(branch);
  ar.add(new Integer(atd));
  con.close();
  return ar;
 }
}
```

Server:

```
package prac6;
    import java.rmi.*;
import java.rmi.registry.LocateRegistry;

public class Prac6 {
```

```
 public static void main(String args[]) throws Exception
 {
  StudentI obj=new StudentC();
     LocateRegistry.createRegistry(1900);
  Naming.rebind("rmi://localhost:1900/stinfo",obj);
  System.out.println("Server Started");
 }
}
```

Client:

```
import java.rmi.*;
import java.sql.*;
import java.util.*;

public class client {

 public static void main(String args[]) throws Exception
 {
  Scanner scan= new Scanner(System.in);
  System.out.println("Enter id,name,branch,attendance for student:");
  int id=scan.nextInt();
  String name=scan.next();
  String branch=scan.next();
  int atd=scan.nextInt();
  StudentI obj=(StudentI)Naming.lookup("rmi://localhost:1900/stinfo");
  ArrayList ar=obj.insert(id,name,branch,atd);
  Iterator it=ar.iterator();
  System.out.println("Id\tName\tBranch\tAttendance");

System.out.println(it.next()+"\t"+it.next()+"\t"+it.next()+"\t"+it.next()+"\t");
 }
}
```

Output:

Server:

```
run:
Server Started
```

Client:

```
run:
Enter id,name,branch,attendance for student:
4
ks
ss
4
Id      Name    Branch  Attendance
4       ks      ss      4
BUILD SUCCESSFUL (total time: 6 seconds)
```

Sign:

Date:

PRACTICAL-7

Aim: Create Servlet file which contains following functions:1. Connect 2. Create Database 3. Create Table 4. Insert Records into respective table 5. Update records of particular table of database 6. Delete Records from table. 7. Delete table and also database.

Code:

Index.html:

```
<!DOCTYPE html>
<html>
<head>
 <title>prac7</title>
  </head>
<body >
<h1 align="center">Welcome To Student Portal</h1>
<div class="main">
<input type="button" class="button" name="insert_user"
onclick="window.location.href='page.html'" value="Insert Student"><br>
<input type="button" class="button green" name="delete_user"
onclick="window.location.href='delete.html'" value="Delete Student"><br>
<input type="button" class="button grey" name="update_user"
onclick="window.location.href='update.html'" value="Update Student"><br>
<input type="button" class="button blue" name="select_user"
onclick="window.location.href='select.html'" value="Select Student">
</div>
</body>
</html>
```

Delete.html

```
<!DOCTYPE html>
<html>
   <head>
      <title>Servlet</title>
      <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
   </head>
   <body >
 <form action="WebServlet" method="post" style="text-align:center;">
<h2 style="color:white;font-family: Times; ">Student Information</h2>
<input class="side" type="text" pattern="[0-9]*" required=""
name="s_enrollno" minlength="12" maxlength="12" placeholder="Enrollment
No" style=" width: 370px;"><br>

<input class="b1" type="submit" value="Delete Data" name="s1">

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

Page.html:

```
<!DOCTYPE html>
<html>
<head>
      <title>Servlet</title>
      <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width, initial-scale=1.0">
      <link rel="stylesheet" type="text/css" href="main.css">

 </head>
<body >
 <form action="WebServlet" method="post" style="text-align:center;">



<h1 style="color:white;font-family: Times; ">Student Information</h1>


<input class="side" type="text"  pattern="[0-9]*" required=""
name="s_enrollno" placeholder="Enrollment No" maxlength="12"><br>
<input class="side a" type="text" pattern="[A-Za-z]+"
required=""  name="s_firstname" placeholder="First Name">

<input class="side b" type="text" pattern="[A-Za-z]+" name="s_lastname"
placeholder="Last Name" required><br>

<input class="side" type="text" pattern="[A-Za-z]+" name="s_branch"
placeholder="Branch" required><br>

<input class="side" type="text"  pattern="[0-9]*" required=""
```

```
name="s_mobileno" placeholder="Mobile No" minlength="10"
maxlength="10"><br>

<input class="b1" type="submit" value="Insert Data" name="s1" >

</form>

</body>
</html>
```

Select.html

```
<!DOCTYPE html>
<html>
   <head>
      <title>Servlet</title>
      <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width, initial-
scale=1.0">       </head>
   <body
 <form action="WebServlet" method="post" style="text-align:center;">
 <h1 style="color:white;font-family: Times; ">Student Information</h1>
<input class="side" type="text" pattern="[0-9]*" required=""
name="s_enrollno" placeholder="Enrollment No" maxlength="12" style="
width: 370px;"><br>

<input class="b1" type="submit" value="View Data" name="s1">

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

Update.html

```
<!DOCTYPE html>
<html>
   <head>
      <title>Servlet</title>
      <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width, initial-scale=1.0">

     </head>
   <body >
 <form action="WebServlet" action="newupdate.html" method="post"
style="text-align:center;">
```

```
<h1 style="color:white;font-family: Times; ">Student Information</h1>
<h2 style="color:white;">Insert Enrollment No To Change Details Of
Student</h2>
<input class="side" type="text" pattern="[0-9]*" required=""
name="s_enrollno" placeholder="Enrollment No" maxlength="12"><br>
<h2 style="color:white;"">Insert New Data</h2>
<input class="side a" type="text" pattern="[A-Za-z]+"
required=""  name="s_firstname" placeholder="First Name">

<input class="side b" type="text" pattern="[A-Za-z]+" name="s_lastname"
placeholder="Last Name" required><br>

<input class="side" type="text" pattern="[A-Za-z]+" name="s_branch"
placeholder="Branch" required><br>

<input class="side" type="text" pattern="[0-9]*" required=""
name="s_mobileno" placeholder="Mobile No" minlength="10"
maxlength="10" ><br>
<input class="b1"  type="submit" value="Update Data" name="s1" >

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

WebServlet.java

```
import java.io.*;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;



import javax.servlet.*;
import javax.servlet.http.*;

 public class WebServlet  extends HttpServlet {


   @Override
   protected void doPost(HttpServletRequest request, HttpServletResponse
response)
```

```
        throws ServletException, IOException {
      response.setContentType("text/html;charset=UTF-8");
      PrintWriter out = response.getWriter();

   try
      {
         Class.forName("com.mysql.jdbc.Driver");
         System.out.println("loaded driver succesfull");
         Connection cn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/mysql","root","birj
u");
         Statement st=cn.createStatement();
         st.executeUpdate("create database student");
         st.executeUpdate("use student");
         st.execute("create table student2 (s_enrollno BIGINT,s_firstname
varchar(20),s_lastname varchar(20),s_branch varchar(10),s_mobileno
BIGINT)");

         ResultSet rs ;


        String s2 = request.getParameter("s1");

        Long eno = Long.valueOf(request.getParameter("s_enrollno"));

        String fname =request.getParameter("s_firstname");
        String lname = request.getParameter("s_lastname");
        String branch =request.getParameter("s_branch");
        String mobile=request.getParameter("s_mobileno");

        PreparedStatement sr;

        Statement stmt = cn.createStatement();
         ResultSet res;

         // For Insert Data......
         switch (s2) {
         //For Delete Data......
            case "Insert Data":
               String str = "insert into student2 values(?,?,?,?,?)";
               sr = cn.prepareStatement(str);
               sr.setLong(1,eno);
               sr.setString(2,fname);
               sr.setString(3,lname);
```

```
                sr.setString(4,branch);
                sr.setString(5,mobile);
                int i = sr.executeUpdate();
                out.println("Student Data inserted");
                cn.close();
                break;
             case "Delete Data":
                String str1 = "select s_enrollno from student2 where s_enrollno =
"+eno+" ";
                rs =stmt.executeQuery(str1);
                if(rs.next())
                {
                   str1 = "delete from student2 where s_enrollno = ?";
                   sr = cn.prepareStatement(str1);
                   sr.setLong(1,eno);
                   sr.executeUpdate();
                   out.println("Student Data Deleted");
                }
                else{
                   out.println("Student Data Not Valid");
                }      cn.close();
                break;
             case "Update Data":
                String str3 = "select * from student2 where s_enrollno = "+eno+"";
                rs =stmt.executeQuery(str3);
                if(rs.next())
                {
                   str3= "update student2 set s_enrollno=? ,
s_firstname=?,s_lastname=?,s_branch=?,s_mobileno=? where s_enrollno =
"+eno+"";
                   sr=cn.prepareStatement(str3);
                   sr.setLong(1,eno);
                   sr.setString(2,fname);
                   sr.setString(3,lname);
                   sr.setString(4,branch);
                   sr.setString(5,mobile);

                   sr.executeUpdate();

                   out.println("Update Student2 Data Succesfull");
                   cn.close();
                }
                else{
                   out.println("Student Data Not Valid");
```

```
                }   break;
              case "View Data":
                 res = stmt.executeQuery("select * from student2 where
s_enrollno="+eno+"");
                 if(res.next()){
                    Long enu= res.getLong("s_enrollno");
                    String fname1= res.getString("s_firstname");
                    String lname1= res.getString("s_lastname");
                    String branch1= res.getString("s_branch");
                    String mb1= res.getString("s_mobileno");
                    out.println("<html><body>");
                    out.println("Enrollment No:- ");
out.println(enu);out.println("<br>");
                    out.println("First Name:- ");
out.println(fname1);out.println("<br>");
                    out.println("Last Name:- ");
out.println(lname1);out.println("<br>");
                    out.println("Branch:- "); out.println(branch1);
out.println("<br>");
               out.println("Mobile No:- ");out.println(mb1);out.println("<br>");
              out.println("</html></body>");
            }
              else{
              out.println("Student Data is Not Valid");
              }    break;
            }
       }catch(ClassNotFoundException | SQLException e   )
       { System.out.println(e); }}}
```

web.xml

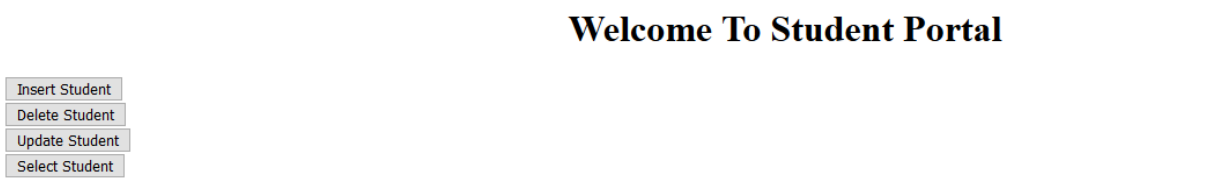
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee">
   <servlet>
      <servlet-name>WebServlet</servlet-name>
      <servlet-class>WebServlet</servlet-class>
   </servlet>
   <welecome-file-list>
      <welcome-file>MainPage.html</welcome-file>
        <welcome-file>index.html</welcome-file>
   </welecome-file-list>

   <servlet-mapping>
      <servlet-name>WebServlet</servlet-name>
      <url-pattern>/WebServlet</url-pattern>
```

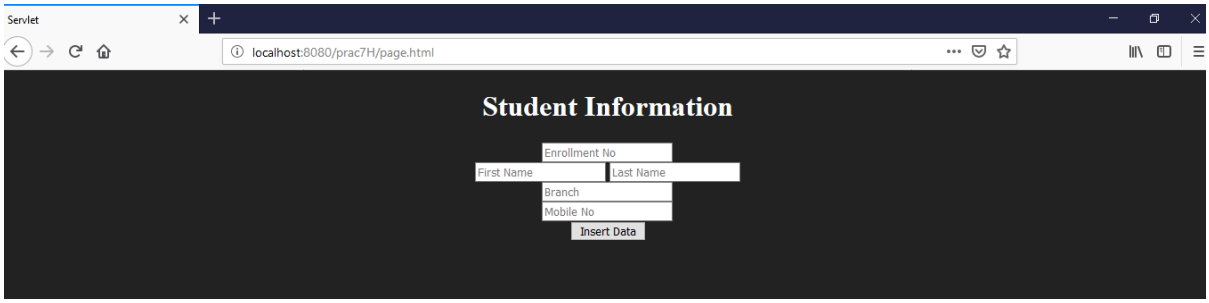
```
    </servlet-mapping>
</web-app>
```

Output:

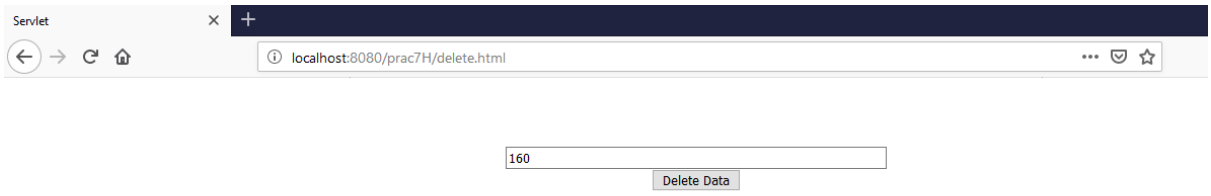
Index.html:



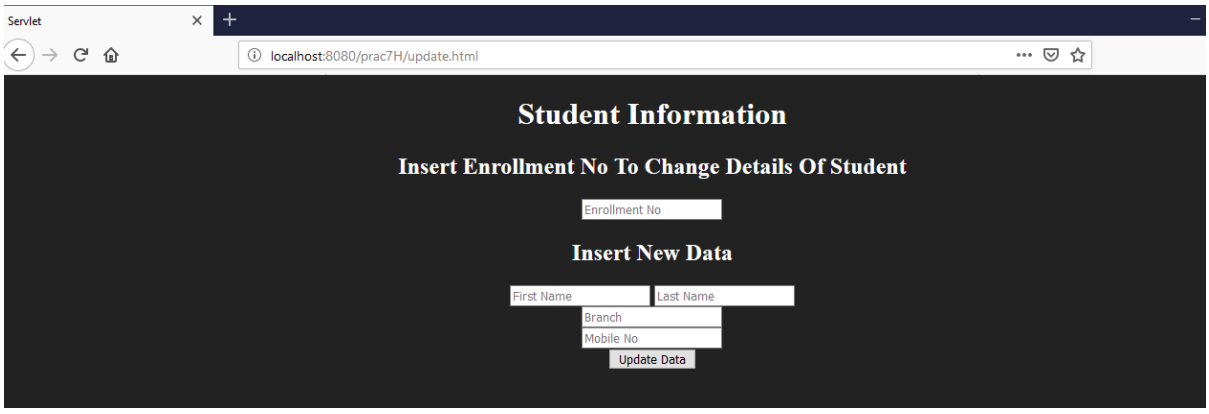
Insert Student:



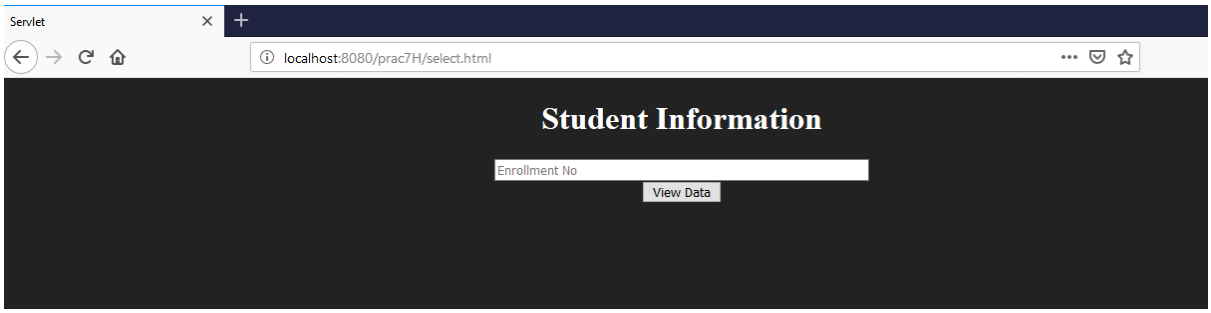
Delete student:



Update student detail:



View data:



Sign:

Date:

PRACTICAL-8

Aim: User can create a new database and also create new table under that database. Once database has been created then user can perform database operation by calling above functions. Use following Java Statement interface to implement program:

1. Statement 2. Prepared statement 3. Callable statement

Code:

```
import java.sql.*;
import java.util.Scanner;

public class SqlCon {
	public static void main(String[] args)throws Exception
	{
		System.out.println("Enter database name to create new database:");
		Scanner scan=new Scanner(System.in);
		String dbname=scan.next();
		Class.forName("com.mysql.jdbc.Driver");
		Connection
		con=DriverManager.getConnection("jdbc:mysql://localhost:3306/"
		,"wordpress","VHXes6LbxbXlXail");
		Statement stmt=con.createStatement();
		stmt.executeUpdate("create database "+dbname);
		System.out.println("Database Created");
		stmt.execute("use "+dbname);
		System.out.println("Enter table name to create table:");
		String tbname=scan.next();
		stmt.executeUpdate("create table "+tbname+" (id int,name
char(20),branch
		char(10),address char(20))");
		System.out.println("Table Created with (id,nam,branch,address)
fields");
		while(true)
		{
			System.out.println("Enter your
			choice:\n1.Insert\t2.Delete\t3.select\t4.Exit");
				int choice=scan.nextInt();
				switch(choice)
				{
			case 1:
			{
```

```
                System.out.println("Enter (id,name,branch,address) of
student:");
                int id=scan.nextInt();
                String name=scan.next();
                String branch=scan.next();
                String add=scan.next();
                PreparedStatement sr;
                String str = "insert into "+tbname+" values(?,?,?,?)";
                sr = con.prepareStatement(str);
                sr.setInt(1,id);
                sr.setString(2,name);
                sr.setString(3,branch);
                sr.setString(4,add);
                sr.executeUpdate();
                System.out.println("Data inserted");
                break;
            }
            case 2:
            {
                System.out.println("Enter id of student:");
                int id=scan.nextInt();
                ResultSet rs=stmt.executeQuery("select * from
"+tbname+"
                where id="+id);
                if(rs.next())
                {
                        stmt.executeUpdate("delete from
"+tbname+" where
                    id="+id);
                        System.out.println("Data deleted");
                }
                else
                {
                        System.out.println("Data with id "+id+"
doesn't exsist");
                }
                break;
            }
            case 3:
            {
                System.out.println("Enter id of student:");
                int id=scan.nextInt();
                stmt.execute("CREATE PROCEDURE
"+dbname+".selector (IN idr
```

```
                                INT) "+
                                "BEGIN "+
                                "SELECT * "+
                                "FROM "+tbname+" "+
                                "WHERE id = idr; "+"END");
                                CallableStatement cs = con.prepareCall("{call
                                "+dbname+".selector(?)}");
                                cs.setInt(1, id);
                                System.out.println("procedure call sucess");
                                cs.execute();
                                ResultSet rs=cs.getResultSet();
                                if(rs.next())
                                {
                                        System.out.println("id\tName\tBranch\tAddress
                                        ");
                                        System.out.println(rs.getInt(1)+"\t"+rs.getStrin
                                        g(2)+"\t"+rs.getString(3)+"\t"+rs.getString(4));
                                }
                                 break;
                        }
                        case 4:
                        {
                                break;
                        }
                        default:
                        {
                                System.out.println("Invalid option");
                        }
                        }
                        if(choice==4)
                        {
                        break;
                        }
        }
        con.close();
    }
}
```

OUTPUT:-

Enter database name to create new database:
STINFO
Database Created
Enter table name to create table:

STRESULT
Table Created with (id,nam,branch,address) fields
Enter your choice:
1.Insert 2.Delete 3.select 4.Exit
1
Enter (id,name,branch,address) of student:
1
Mayank
IT
Surendranagar
Data inserted
Enter your choice:
1.Insert 2.Delete 3.select 4.Exit
1
Enter (id,name,branch,address) of student:
2
hiren
IT
Surat
Data inserted
Enter your choice:
1.Insert 2.Delete 3.select 4.Exit
2
Enter id of student:
2
Data deleted
Enter your choice:
1.Insert 2.Delete 3.select 4.Exit
3
Enter id of student:
1
procedure call success
id Name Branch Address
1 Mayank IT surendranagar
Enter your choice:
1.Insert 2.Delete 3.select 4.Exit
4

Sign:

Date:

PRACTICAL-9

Aim: Create Servlet file and study web descriptor file.

Code:

DemoServlet.java

```
import javax.servlet.http.*;
import javax.servlet.*;
import java.io.*;
public class DemoServlet extends HttpServlet{
public void doGet(HttpServletRequest req,HttpServletResponse res)
throws ServletException,IOException
{
res.setContentType("text/html");//setting the content type
PrintWriter pw=res.getWriter();//get the stream to write the data

//writing html in the stream
pw.println("<html><body>");
pw.println("Welcome to servlet");
pw.println("</body></html>");

pw.close();//closing the stream
}}
```

Create the deployment descriptor (web.xml file)

The deployment descriptor is an xml file, from which Web Container gets the information about the servet to be invoked.

The web container uses the Parser to get the information from the web.xml file. There are many xml parsers such as SAX, DOM and Pull.

There are many elements in the web.xml file. Here is given some necessary elements to run the simple servlet program.

web.xml file

```
<web-app>

<servlet>
<servlet-name>sonoojaiswal</servlet-name>
<servlet-class>DemoServlet</servlet-class>
</servlet>

<servlet-mapping>
<servlet-name>sonoojaiswal</servlet-name>
<url-pattern>/welcome</url-pattern>
</servlet-mapping>

</web-app>
```

Sign:

Date:

PRACTICAL-10

Aim: Create login form and perform state management using Cookies, HttpSession and URL Rewriting.

Code:

index.html

```
<body>
    <div class="box">
    <h1> Mayank's Session Management</h1>
    <form action="FirstServlet" method="post">
        <input class="side" type="text" name="userName" placeholder="Username"/><br/>
        <input class="side" type="password" name="password" placeholder="Password"/><br/>
<input class="b1" type="submit" value="Login"/>
</form>
    </div>
  </body>
```

FirstServler.java

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.Cookie;
```

```
import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;


public class FirstServlet extends HttpServlet {

   protected void service(HttpServletRequest request, HttpServletResponse
response)

          throws ServletException, IOException {

       response.setContentType("text/html;charset=UTF-8");

       try{

 response.setContentType("text/html");

 PrintWriter out = response.getWriter();

 String name=request.getParameter("userName");

       String pwd=request.getParameter("password");

       if(pwd.equals("admin") && name.equals("admin"))

       {

 out.print("<h1>Welcome "+name+"</h1>");

 Cookie ck=new Cookie("uname",name);//creating cookie object

       response.addCookie(ck);//adding cookie in the response

       String n=name;

       out.print("Welcome "+n);

       HttpSession session=request.getSession();
```

```
        session.setAttribute("uname",n);

        //appending the username in the query string

        out.print("<a href='SecondServlet?uname="+n+"'>Click here for URL Rewrite session tracking method</a>");

       out.print("<b>Cookie has been generated for this session<br></b>");

        out.print("<b>Click on button to view Cookie...</b>");

 //creating submit button

 out.print("<form action='SecondServlet' method='post'>");

 out.print("<br><input type='submit' value='go'>");

 out.print("</form>");

        }

        else

        {

            out.println("Incorrect Username or Password!!!");

        }

 out.close();

        }catch(Exception e){System.out.println(e);}

  }

}
```

SecondServlet.java

```
import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;
```

```
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
public class SecondServlet extends HttpServlet {
   protected void service(HttpServletRequest request, HttpServletResponse
response)
           throws ServletException, IOException {
      try{
  response.setContentType("text/html");
 PrintWriter out = response.getWriter();
  String n=request.getParameter("uname");
if(n!=null)
{
  out.println("Value stored in URL : "+n+"\n");
}
    HttpSession session=request.getSession(false);
        String x=(String)session.getAttribute("uname");
        out.println("\n this is from current session: "+x);
 Cookie ck[]=request.getCookies();
 out.println("Value stored in Cookie : "+ck[0].getValue());
```

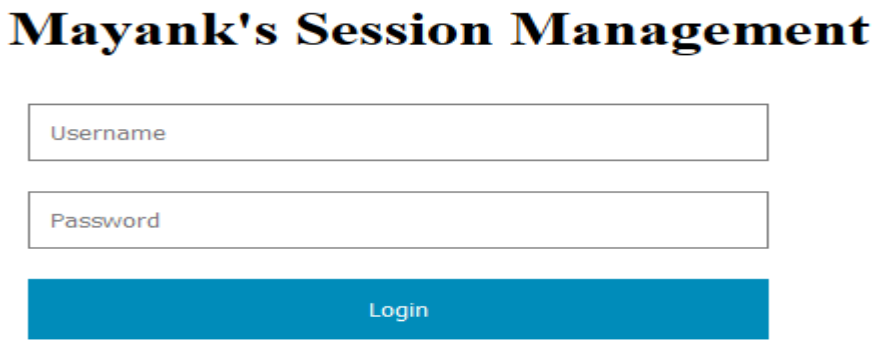
```
 out.close();
        }catch(Exception e){System.out.println(e);}
       }
   }
```

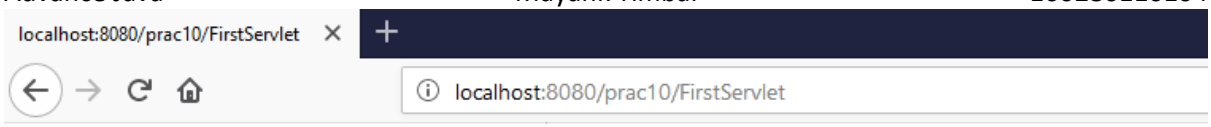
Web.xml

```
<web-app >
   <servlet>
      <servlet-name>FirstServlet</servlet-name>
      <servlet-class>FirstServlet</servlet-class>
   </servlet>
   <servlet>
      <servlet-name>java</servlet-name>
      <servlet-class>SecondServlet.java</servlet-class>
   </servlet>
   <servlet>
      <servlet-name>SecondServlet</servlet-name>
      <servlet-class>SecondServlet.SecondServlet</servlet-class>
   </servlet>
   <servlet-mapping>
      <servlet-name>FirstServlet</servlet-name>
      <url-pattern>/FirstServlet</url-pattern>
   </servlet-mapping>
```

```
    <servlet-mapping>
        <servlet-name>java</servlet-name>
        <url-pattern>/java</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>SecondServlet</servlet-name>
        <url-pattern>/SecondServlet</url-pattern>
    </servlet-mapping>
    <session-config>
        <session-timeout>
            30
        </session-timeout>
    </session-config>
</web-app>
```

Output:

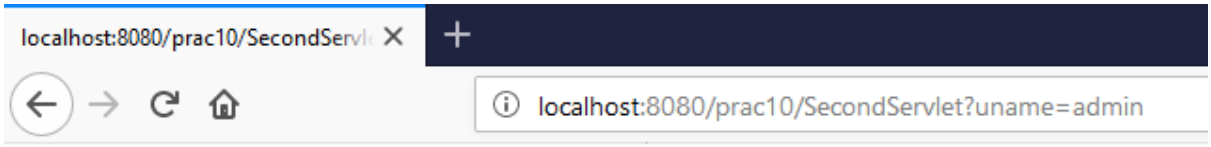




Welcome admin

Welcome adminClick here for URL Rewrite session tracking method**Cookie has been generated for this session**
Click on button to view Cookie...

go



Value stored in URL : admin this is from current session: admin Value stored in Cookie : admin

Date:

PRACTICAL-11

Aim: Implement Authentication filter using filter API.

Code:

index.html

```
<form action="servlet1">
Name:<input type="text" name="name"/><br/>
Password:<input type="password" name="password"/><br/>
<input type="submit" value="login">
</form>
```

MyFilter.java

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.*;

public class MyFilter implements Filter{

public void init(FilterConfig arg0) throws ServletException {}

public void doFilter(ServletRequest req, ServletResponse resp,
      FilterChain chain) throws IOException, ServletException {

   PrintWriter out=resp.getWriter();

   String password=req.getParameter("password");
   if(password.equals("admin")){
   chain.doFilter(req, resp);//sends request to next resource
   }
   else{
   out.print("username or password error!");
   }

}
   public void destroy() {}

}
```

AdminServlet.java

```
import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.*;

public class AdminServlet extends HttpServlet {
public void doGet(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {

      response.setContentType("text/html");
      PrintWriter out = response.getWriter();

      out.print("welcome ADMIN");
      out.close();
   }
}
```

Web.xml

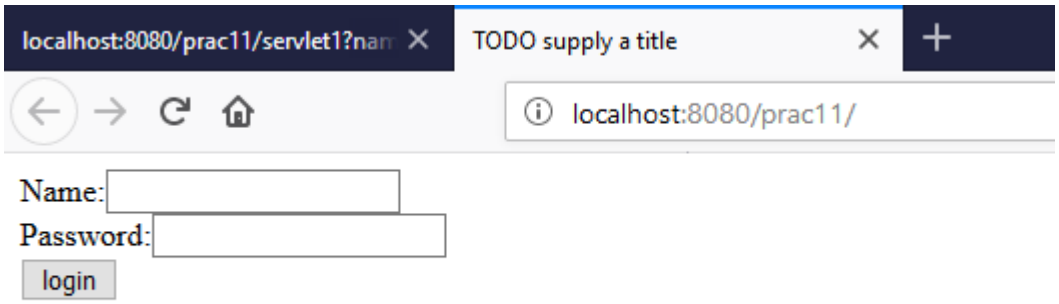
```
<web-app>
 <servlet>
   <servlet-name>AdminServlet</servlet-name>
   <servlet-class>AdminServlet</servlet-class>
  </servlet>

  <servlet-mapping>
   <servlet-name>AdminServlet</servlet-name>
   <url-pattern>/servlet1</url-pattern>
  </servlet-mapping>

 <filter>
  <filter-name>f1</filter-name>
  <filter-class>MyFilter</filter-class>
  </filter>
  <filter-mapping>
  <filter-name>f1</filter-name>
  <url-pattern>/servlet1</url-pattern>
  </filter-mapping>

</web-app>
```

Output:



For wrong password:

```
username or password error!
```

For right password:

welcome ADMIN

Sign:

Date:

PRACTICAL-12

Aim: Create database of student subject-wise data and retrieve all data using JSP and generate xml structure along with DTD and XML Schema definition.

Code:

JSP file:

```
<%@ taglib prefix = "c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix = "x" uri="http://java.sun.com/jsp/jstl/xml" %>
<html>
  <head>
     <title></title>
  </head>
  <body>
    <h3>Books Info:</h3>
    <c:import var = "student" url="http://localhost:8080/practicals/stud.xml"/>
    <x:parse xml = "${student}" var = "output"/>
    <b>student details are as below as per subjects:</b>:
     <x:out select = "$output/sub[1]" />
    <x:out select = "$output/sub/AJ[1]/name" />
    <x:out select = "$output/sub/AJ[1]/marks" /><br>
    <x:out select = "$output/sub[2]]" />
    <x:out select = "$output/sub/AJ[2]/name" />
    <x:out select = "$output/sub/AJ[2]/marks" /><br>
   </body>
</html>
```

xml file:

```
<?xml version="1.0" encoding="UTF-8"?>
<student>
   <sub>
      <AJ>
         <name>abc</name>
         <marks>70</marks>
      </AJ>
      <AJ>
         <name>bcd</name>
         <marks>80</marks>
```

```
      </AJ>
       <AJ>
         <name>def</name>
         <marks>90</marks>
      </AJ>
   </sub>
   <sub>
      <WT>
         <name>abc</name>
         <marks>70</marks>
      </WT>
      <WT>
         <name>bcd</name>
         <marks>80</marks>
      </WT>
       <WT>
         <name>def</name>
         <marks>90</marks>
      </WT>
    </sub>
</student>
```

stud.dtd:

```
<!ELEMENT student (sub,AJ,WT)>
<!ELEMENT sub (#PCDATA)>
<!ELEMENT AJ(#PCDATA)>
<!ELEMENT WT (#PCDATA)>
<?xml version="1.0"?>
```

stud.xsd:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" >
<xs:element name="stud">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="sub" type="xs:string"/>
      <xs:element name="AJ" type="xs:string"/>
      <xs:element name="WT" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

OUTPUT:

student info:

student details are as below as per subjects::
sub:AJ
name:abc
marks:70

sub:AJ
name:bcd
marks:0

Date:

PRACTICAL-13

Aim: Refer Practical 11 and apply XSLT (style) to generated XML document and print you result.

Code:

User.xml:

```
<?xml version = "1.0"?>
<?xml-stylesheet type = "text/xsl" href = "User.xsl"?>
<users>
  <user id = "1">
    <name>admin</name>
   </user>
  <user id = "2">
    <name>Sara</name>
   </user>
  </users>
```

user.xsl:

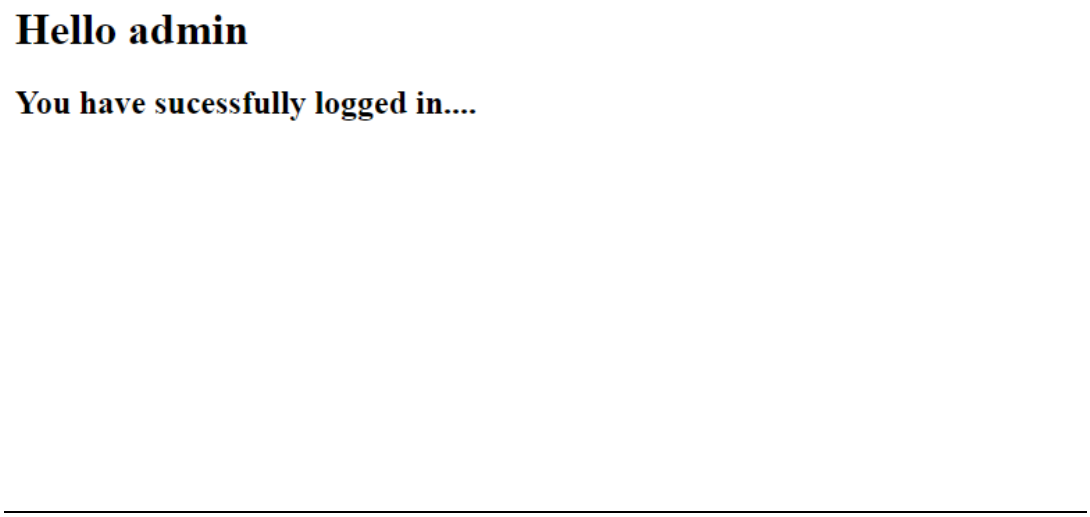
```
<xsl:template match="user/name">
 <fo:block font-weight="bold">
  <xsl:apply-templates/>
 </fo:block>
</xsl:template>
```

index.html:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xsl:stylesheet version = "1.0"
xmlns:xsl = "http://www.w3.org/1999/XSL/Transform">
  <xsl:template match = "/">
<html>
        <body>
        <h1><B>HELLO <xsl:value-of select =”name”/> </b></h1>
```

```
        You have successfully  logged in……
</body>
</html>
```

OUTPUT:



Date:

PRACTICAL-14

Aim: Create Web Service which provides student information.

Code:

Stud_Info.java:

```
package com.javaservdemo;
import javax.jws.WebMethod;
import javax.jws.WebService;
import javax.jws.soap.SOAPBinding;
import javax.jws.soap.SOAPBinding.Style;
//Service Endpoint Interface
@WebService
@SOAPBinding(style = Style.RPC)
public interface Stud_Info{
   @WebMethod String getStudentInfoAsString(String name);
}
```

Stud_Info_Impl.java:

```
package com.javaservdemo;
import javax.jws.WebService;
//Service Implementation
@WebService(endpointInterface = "com.javatpoint.Stud_Info")
public class Stud_Info_Impl implements Stud_Info{
 Array Data[]=new Array[5];
Data[“admin”] =”123456789”;
Data[“abcd”] =”125556789”;
Data[“abde”] =”111111111”;
 @Override
   public String getStudentInfoAsString (String name) {
      return  “enrollment:”+Data[name] ;
   }
}
```

Admin.java:

```
package com.javaservdemo;
```

```
import javax.xml.ws.Endpoint;
public class Admin{
   public static void main(String[] args) {
      Endpoint.publish("http://localhost:7779/ws/admin", new
Stud_Info_Impl());
       }
}
```

Client.java:

```
package com.javaservdemo;
import java.net.URL;
import javax.xml.namespace.QName;
import javax.xml.ws.Service;
public class Client{
   public static void main(String[] args) throws Exception {
   URL url = new URL("http://localhost:7779/ws/hello?wsdl");
       QName qname = new QName("http://localhost:7779",
"Stud_Info_ImplService");
       Service service = Service.create(url, qname);
       Stud_Info data = service.getPort(Stu_Info.class);
       System.out.println(data. getStudentInfoAsString ("admin"));
    }
 }
```

OUTPUT:

enrollment:123456789

Sign:

Date:

PRACTICAL-15

Aim: Create Web Service client which consumes above service and display student data by entering student id.

Code:

Stud_Info.java:

```
package com.javaservdemo;
import javax.jws.WebMethod;
import javax.jws.WebService;
import javax.jws.soap.SOAPBinding;
import javax.jws.soap.SOAPBinding.Style;
@WebService
@SOAPBinding(style = Style.RPC)
public interface Stud_Info{
   @WebMethod String getStudentInfoAsString(String name);
}
```

Stud_Info_Impl.java:

```
package com.javaservdemo;
import javax.jws.WebService;
//Service Implementation
@WebService(endpointInterface = "com.javatpoint.Stud_Info")
public class Stud_Info_Impl implements Stud_Info{
 Array Data[]=new Array[5];
Data[“admin”] =”123456789”;
Data[“abcd”] =”125556789”;
Data[“abde”] =”111111111”;
 @Override
   public String getStudentInfoAsString (String name) {
      return  “enrollment:”+Data[name] ;
   }
}
```

Admin.java:

```
package com.javaservdemo;
import javax.xml.ws.Endpoint;
```

```
public class Admin{
   public static void main(String[] args) {
      Endpoint.publish("http://localhost:7779/ws/admin", new
Stud_Info_Impl());
       }
}
```

Client.java:

```
package com.javaservdemo;
import java.net.URL;
import javax.xml.namespace.QName;
import javax.xml.ws.Service;
public class Client{
   public static void main(String[] args) throws Exception {
   URL url = new URL("http://localhost:7779/ws/hello?wsdl");
       QName qname = new QName("http://localhost:7779",
"Stud_Info_ImplService");
       Service service = Service.create(url, qname);
       Stud_Info data = service.getPort(Stu_Info.class);
       System.out.println(data. getStudentInfoAsString (1));
    }
 }
```

OUTPUT:

```
name:admin
```

Date:

PRACTICAL-16

Aim: Study and implement Hibernate.

Code:

- **t1.java:**

```
package javaapplication1;
public class t1 {
privateint id,mark1,mark2;
private String name;
publicintgetid() {
return id;
}
public void setid(int id) {
this.id = id;
}
public String getname() {
return name;
}
public void setname(String name) {
this.name = name;
}
publicint getmark1() {
return mark1;
}
public void setmark1(int mark1) {
this.mark1 = mark1;
}
publicint getmark2() {
return mark2;
}
public void setmark2(int mark2) {

this.mark2 = mark2;
} }
```

- **t1.hbm.xml :**

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping
DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
```

```
<hibernate-mapping>
<class name=" javaapplication1.t1" table="t1">
<id name="id">
<generator class="assigned"></generator>
</id>
<property name="name"></property>
<property name="mark1"></property>
<property name="mark2"></property>
</class>
</hibernate-mapping>
```

- **hibernate.cfg.xml:**

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
"-//Hibernate/Hibernate Configuration DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
<session-factory>
<property name="hbm2ddl.auto">update</property>
<property name="dialect">org.hibernate.dialect.MySqlDialect</property>
<property
name="connection.url">jdbc:mysql://localhost:3308/test_today</property>
<property name="connection.username">root</property>
<property name="connection.password">ait</property>
<property name="connection.driver_class">com.mysql.jdbc.Driver</property>
<mapping resource="t1.hbm.xml"/>
</session-factory>
</hibernate-configuration>
```

- **Insertdemo.java:**

```
package javaapplication1;
importorg.hibernate.Session;
importorg.hibernate.SessionFactory;
importorg.hibernate.Transaction;
importorg.hibernate.cfg.Configuration;
public class Insertdemo.java
{
public static void main(String[] args) {
Configuration cfg=new Configuration();
cfg.configure("hibernate.cfg.xml");
SessionFactory factory=cfg.buildSessionFactory();
Session session=factory.openSession();
```

```
Transaction t=session.beginTransaction();
t1 e1=new t1();
e1.setid(11);
e1.setname(“abc”);
e1.setmark1(49);
e1.setmark2(43);
session.persist(e1);
t.commit();
session.close();
System.out.println("Inserted Successfully");
}
}
```

Sign:

Date:

PRACTICAL-17

Aim: Study and implement MVC using Spring Framework.

Code:

- **Employee.java**

```
public class Employee
{
privateint id;
private String name;
public Employee()
 {
System.out.println("def cons");
}
public Employee(int id)
{
this.id = id;
}
public Employee(String name)
 {
this.name = name;
}
public Employee(int id, String name)
{
this.id = id;
this.name = name;
}
void show()
{
System.out.println(id+" "+name);
}
}
```

- **applicationContext.xml**

```
<beans>
<bean id="e" class="Employee">
<constructor-arg value="10" type="int">
</constructor-arg>
</bean>
</beans>
```

- **Test.java**

```
importorg.springframework.beans.factory.BeanFactory;
importorg.springframework.beans.factory.xml.XmlBeanFactory;
import org.springframework.core.io.*;
public class Test
{
public static void main(String[] args)
 {
Resource r=new ClassPathResource("applicationContext.xml");
BeanFactory factory=new XmlBeanFactory(r);
Employee s=(Employee)factory.getBean("e");
s.show();
}
}
```

Output:10 null

Sign: