Program 1: Write a program for one-way client and server communication using java Socket where client sends a message to the server, then the server reads the message and print it.

First run server then client

Client.java

```
package dscc1;

import java.net.*;
import java.io.*;
import java.net.Socket;
public class Client
{
public static void main(String args[]) throws Exception
{
Socket s=new Socket("localhost",1245);
DataOutputStream dos=new DataOutputStream(s.getOutputStream());
dos.writeUTF("Hello!!How are you"); s.close();
}
}
```

Server.java

```
package dscc1;

import java.net.*;
import java.io.*;
public class Server {
public static void main(String[] args) throws IOException {
ServerSocket ss=new ServerSocket(1245);
Socket s=ss.accept();
DataInputStream dis=new DataInputStream(s.getInputStream());
String str=dis.readUTF();
System.out.println("Message from client: "+str);

}
}
```

Program 2: Write a program for client server chat(Two-way communication) using java socket.

First run server then client

Client.java

```
package dscc2;
```

```
import java.net.*;
import java.io.*;
public class Client
{
public static void main(String args[]) throws Exception
{
Socket s=new Socket("localhost",1245);
DataInputStream dis=new DataInputStream(s.getInputStream());
DataOutputStream dos=new DataOutputStream(s.getOutputStream());
dos.writeUTF("Hello!!How are you I am client");
String str1=dis.readUTF();
System.out.println("Message from server: "+str1);

s.close();
}
}
```

Server.java

```
package dscc2;
import java.net.*;
import java.io.*;
public class Server
{
public static void main(String args[]) throws Exception
{
ServerSocket ss=new ServerSocket(1245);
Socket s=ss.accept();
DataInputStream dis=new DataInputStream(s.getInputStream());
DataOutputStream dos=new DataOutputStream(s.getOutputStream());
dos.writeUTF("Hello!!I am server");
String str=dis.readUTF();
System.out.println("Message from client: "+str);

}
}
```

Program 3: A program for client server GUI chat using java Socket.

First run server then client

Client.java

```
package dscc3;
```

```
import java.io.*;
import java.net.*;
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
public class Client extends JFrame implements ActionListener,Runnable  {
JButton b;
JTextField tf;
JTextArea ta;
Socket s;
PrintWriter pw;
BufferedReader br;
Thread th;
public Client()
{
b=new JButton("Send");
b.addActionListener(this);
tf=new JTextField(20);
ta=new JTextArea(19,30);
add(ta);
add(tf);
add(b);
try
{
s=new Socket("localhost",1234);
br=new BufferedReader(new InputStreamReader(s.getInputStream()));  pw=new
PrintWriter(s.getOutputStream(),true);
}
catch(Exception e) { }
th=new Thread(this);
th.start();
}
public void actionPerformed(ActionEvent ae)
{
pw.println(tf.getText());
ta.append("Client says : "+tf.getText()+"\n");
tf.setText("");
}
public void run()
{
while(true)
{
try
{
```

```
ta.append("Server says : "+br.readLine()+"\n");
}
catch(Exception e) {}
}
}
public static void main(String args[])
{
Client c = new Client();
c.setLayout(new FlowLayout());
c.setSize(400,400);
c.setTitle("Client");
c.setVisible(true);
}
}
```

Server.java

```
package dscc3;
import java.io.*;
import java.net.*;
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
public class Server extends JFrame implements ActionListener,Runnable  {
JButton b;
JTextField tf;
JTextArea ta;
ServerSocket ss;
Socket s;
PrintWriter pw;
BufferedReader br;
Thread th;
public Server()
{
b=new JButton("Send");
b.addActionListener(this);
tf=new JTextField(20);
ta=new JTextArea(19,30);
add(ta);
add(tf);
add(b);
try
{
```

```
ss=new ServerSocket(1234);
s=ss.accept();
br=new BufferedReader(new InputStreamReader(s.getInputStream()));  pw=new
PrintWriter(s.getOutputStream(),true);
}
catch(Exception e) { }
th=new Thread(this);
th.start();
}
public void actionPerformed(ActionEvent ae)  {
pw.println(tf.getText());
ta.append("Server says : "+tf.getText()+"\n");  tf.setText("");
}
public void run()
{
while(true)
{
try
{
ta.append("Client says : "+br.readLine()+"\n");  }
catch(Exception e) {}
}
}
public static void main(String args[])  {
Server as = new Server();
as.setLayout(new FlowLayout());
as.setSize(400,400);
as.setTitle("Server");
as.setVisible(true);
}
}
```

Program No. 4 Implement a Program for multi-client chat server.

First run server then client

Client.java

```
package dscc4;

import java.io.*;
import java.net.*;
import java.util.Scanner;
public class Client
{
final static int ServerPort = 1234;
```

```
public static void main(String args[]) throws UnknownHostException,  IOException
{
Scanner scn = new Scanner(System.in);

// getting localhost ip
InetAddress ip = InetAddress.getByName("localhost");
// establish the connection
Socket s = new Socket(ip, ServerPort);
// obtaining input and out streams
DataInputStream dis = new DataInputStream(s.getInputStream()); DataOutputStream dos =
new DataOutputStream(s.getOutputStream());
// sendMessage thread
Thread sendMessage = new Thread(new Runnable()
{
@Override
public void run() {
while (true) {
// read the message to deliver.
String msg = scn.nextLine();
try {
// write on the output stream
dos.writeUTF(msg);
} catch (IOException e) {
e.printStackTrace();
}
}
}
});
// readMessage thread
Thread readMessage = new Thread(new Runnable()
{
@Override
public void run() {
while (true) {
try {
// read the message sent to this client
String msg = dis.readUTF();
System.out.println(msg);
} catch (IOException e) {
e.printStackTrace();
}
}
}
});
```

```
sendMessage.start();
readMessage.start();
}
}
```

Server.java

```
package dscc4;

import java.io.*;
import java.util.*;
import java.net.*;
//Server class
public class Server
{
// Vector to store active clients
static Vector<ClientHandler> ar = new Vector<>();
// counter for clients
static int i = 0;
public static void main(String[] args) throws IOException
{
// server is listening on port 1234
ServerSocket ss = new ServerSocket(1234);
Socket s;
// running infinite loop for getting
// client request
while (true)
{
// Accept the incoming request
s = ss.accept();
System.out.println("New client request received : " + s);
// obtain input and output streams
DataInputStream dis = new DataInputStream(s.getInputStream());
DataOutputStream dos = new DataOutputStream(s.getOutputStream());
System.out.println("Creating a new handler for this client...");
// Create a new handler object for handling this request.
ClientHandler mtch = new ClientHandler(s,"client " + i, dis,
dos);
// Create a new Thread with this object.
Thread t = new Thread(mtch);
System.out.println("Adding this client to active client list");
// add this client to active clients list
ar.add(mtch);
```

```
// start the thread.
t.start();
// increment i for new client.
// i is used for naming only, and can be replaced
// by any naming scheme
i++;
}
}
}
//ClientHandler class
class ClientHandler implements Runnable
{
Scanner scn = new Scanner(System.in);
private String name;
final DataInputStream dis;
final DataOutputStream dos;
Socket s;
boolean isloggedin;
// constructor
public ClientHandler(Socket s, String name,
DataInputStream dis, DataOutputStream
dos) {
this.dis = dis;
this.dos = dos;
this.name = name;
this.s = s;
this.isloggedin=true;
}
@Override
public void run() {
String received;
while (true)
{
try
{
// receive the string
received = dis.readUTF();
System.out.println(received);
if(received.equals("logout")){
this.isloggedin=false;
this.s.close();
break;
}
```

```
// break the string into message and recipient part
StringTokenizer st = new StringTokenizer(received, "#");

String MsgToSend = st.nextToken();
String recipient = st.nextToken();
// search for the recipient in the connected devices list.
// ar is the vector storing client of active users
for (ClientHandler mc : Server.ar)
{
// if the recipient is found, write on its
// output stream
if (mc.name.equals(recipient) &&
mc.isloggedin==true)
{
mc.dos.writeUTF(this.name+" : "+MsgToSend);
break;
}
}
} catch (IOException e) {
e.printStackTrace();
}
}
try
{
// closing resources
this.dis.close();
this.dos.close();
}catch(IOException e){
e.printStackTrace();
}
}
}
```

Program 5:

A. Write a program for one way client and server communication using Datagram Socket.

First run server then client

Client.java

```
package dscc4;
import java.net.*;
import java.io.*;
public class UDPClient
```

```
{
public static void main(String args[]) throws Exception
{
String s="MEOWWWWWWWW??";
DatagramSocket ds=new DatagramSocket();
InetAddress ip=InetAddress.getByName("localhost");
DatagramPacket p=new DatagramPacket(s.getBytes(),s.length(),ip,2222);
ds.send(p);
DatagramSocket ds1=new DatagramSocket(2223);
byte[] b=new byte[1024];
DatagramPacket p1=new DatagramPacket(b,1024);
ds1.receive(p1);
String msg1=new String(p1.getData(),0,p1.getLength());

System.out.println("Message from server: "+msg1);
}}
```

Server.java

```
package dscc4;
import java.net.*;
import java.io.*;
public class UDPServer
{
public static void main(String args[]) throws Exception
{
DatagramSocket ds=new DatagramSocket(2222);
byte[] b=new byte[1024];
DatagramPacket p=new DatagramPacket(b,1024);
ds.receive(p);
String msg=new String(p.getData(),0,p.getLength());

System.out.println("Message from client: "+msg);
String s=("HELLLLLLLOOOOOOO");
DatagramSocket ds1=new DatagramSocket();
InetAddress ip=InetAddress.getByName("localhost");
DatagramPacket p1=new DatagramPacket(s.getBytes(),s.length(),ip,2223);
ds1.send(p1);
}
}
```

B. Implement a Server calculator containing ADD(), MUL(),SUB() etc.

First run server then client

Client.java

```
package dscc5;
import java.io.*;
import java.net.*;
class RPCClient
{
RPCClient()
{
try
{
InetAddress ia = InetAddress.getLocalHost();

DatagramSocket ds = new DatagramSocket();
DatagramSocket ds1 = new DatagramSocket(1300);
System.out.println("\nRPC Client");
System.out.println("----------\n");
System.out.println("Enter Method Name with Parameter like add 3 4\n");
while (true)
{
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
String str = br.readLine();
byte b[] = str.getBytes();
DatagramPacket dp = new DatagramPacket(b,b.length,ia,1200);
ds.send(dp);
dp = new DatagramPacket(b,b.length);
ds1.receive(dp);
String s = new String(dp.getData(),0,dp.getLength());
System.out.println("\nResult = " + s + "\n");
System.out.println("\n\nEnter Method Name with Parameter likeadd 3 4\n");
}
}
catch (Exception e)
{
e.printStackTrace();
}
}
public static void main(String[] args)
{
new RPCClient();
}}
```

Server.java

```
package dscc5;
```

```
import java.net.*;
import java.util.*;
class RPCServer

{
DatagramSocket ds;
DatagramPacket dp;
String str, methodName, result;
int val1, val2;
RPCServer()
{
try
{
ds = new DatagramSocket(1200);
byte b[] = new byte[4096];
while(true)
{
dp = new DatagramPacket(b,b.length);
ds.receive(dp);
str = new String(dp.getData(),0,dp.getLength());
if(str.equalsIgnoreCase("quit"))
System.exit(1);
else
{
StringTokenizer st = new StringTokenizer(str," ");
int i = 0;
while(st.hasMoreTokens()){
String token = st.nextToken();
methodName = token;
val1 = Integer.parseInt(st.nextToken());
val2 = Integer.parseInt(st.nextToken());
}

}
System.out.println("\nClient Selected \""+str+"\" Method :");
System.out.println("\nFirst Value : "+val1);
System.out.println("Second Value : "+val2);
InetAddress ia = InetAddress.getLocalHost();
if(methodName.equalsIgnoreCase("add"))
result= "" + add(val1,val2);
else if(methodName.equalsIgnoreCase("sub"))
result= "" + sub(val1,val2);
else if(methodName.equalsIgnoreCase("mul"))
result= "" + mul(val1,val2);
```

```
else if(methodName.equalsIgnoreCase("div"))
result= "" + div(val1,val2);
byte b1[] = result.getBytes();
DatagramSocket ds1 = new DatagramSocket();
DatagramPacket dp1 = new DatagramPacket(b1,b1.length,InetAddress.getLocalHost(), 1300);
System.out.println("Result : "+result+"\n");
ds1.send(dp1);
}
}
catch (Exception e)
{
e.printStackTrace();
}
}
public int add(int val1, int val2)
{
return val1 + val2;

}
public int sub(int val3, int val4)
{
return val3 - val4;
}
public int mul(int val3, int val4)
{
return val3 * val4;
}
public int div(int val3, int val4)
{
return val3 / val4;
}
public static void main(String[] args)
{
new RPCServer();
}
}
```

Program 6

A. Implement a Date Time Server containing date() and time() using Socket

First run server then client

Client.java

```
import java.net.*;
import java.io.*;
public class DateClient
{
public static void main(String args[]) throws Exception
{
Socket s=new Socket("localhost",4444);
DataInputStream dis=new DataInputStream(s.getInputStream()); String dt=dis.readUTF();
String tm=dis.readUTF();

System.out.println("Date : "+dt);
System.out.println("Time : "+tm);
}
}
```

Server.java

```
import java.net.*;
import java.io.*;
import  java.util.*;
import java.text.*;
public class DateServer
{
DateServer() throws Exception
{
ServerSocket ss=new ServerSocket(4444);
Socket s=ss.accept();
DataOutputStream dos=new DataOutputStream(s.getOutputStream()); dos.writeUTF(date( ));
dos.writeUTF(time( ));
dos.flush();
}
public String date( )
{
return new SimpleDateFormat("dd/mm/yyyy").format(new Date()).toString(); }
public String time( )
{
return new SimpleDateFormat("hh:mm:ss").format(new Date()).toString(); }
public static void main(String args[ ]) throws Exception
{

DateServer d=new DateServer();
}}
```

B. Implement a Date Time Server containing date() and time() using Datagram

First run server then client

Client.java

```
import java.net.*;
import java.io.*;
import java.util.*;
public class DateClient
{
public static void main(String args[]) throws Exception
{
DatagramSocket ds1=new DatagramSocket(2222);
byte[] b1=new byte[1024];
DatagramPacket p1=new DatagramPacket(b1,1024);
ds1.receive(p1);
DatagramSocket ds2=new DatagramSocket(3333);
String dt=new String(p1.getData(),0,p1.getLength());

System.out.println("Date : "+dt);
byte[] b2=new byte[1024];
DatagramPacket p2=new DatagramPacket(b2,1024);
ds2.receive(p2);
String tm=new String(p2.getData(),0,p2.getLength());
System.out.println("Time : "+tm);
}
}
```

Server.java

```
import java.net.*;
import java.io.*;
import java.util.*;
import java.text.*;
public class DateServer
{
DateServer() throws Exception
{
System.out.println("Started");
String d = date( );
DatagramSocket ds=new DatagramSocket();
InetAddress ip=InetAddress.getByName("localhost");
DatagramPacket d1 = new DatagramPacket(d.getBytes(), d.length(), ip, 2222);
ds.send(d1);
String t = time( );
DatagramSocket ds1=new DatagramSocket();
DatagramPacket d2 = new DatagramPacket(t.getBytes(), t.length(), ip, 3333);
```

```
ds.send(d2);
}
public String date( )
{
return new SimpleDateFormat("dd/MM/yyyy").format(new Date()).toString();
}
public String time( )
{
return new SimpleDateFormat("hh:mm:ss").format(new Date()).toString();
}
public static void main(String args[ ]) throws Exception
{
new DateServer();

}
}
```

Program 7: Program to retrieve day, time and date function from server to client. This program should display server day, date and time.(Use RMI)

A.Program to retrieve day, time and date function from server to client. This program should display server day, date and time.(Use RMI)

First run register then client

MyInterface.java

```
import java.rmi.*;
public interface MyInterface extends Remote {
public String getDate() throws RemoteException;



public String getTime() throws RemoteException;
public String getDay() throws RemoteException;
}
```

Register.java

```
import java.rmi.*;
import java.rmi.registry.*; // Registry, LocateRegistry
public class Register {
public static void main(String[] args) {
try {
Registry reg = LocateRegistry.createRegistry(2099);
```

```
MyServer obj = new MyServer();
Naming.rebind("rmi://localhost:2099/dt", obj);
} catch (Exception e) {
}

}
}
```

Client.java

```
import  java.rmi.*; //Naming
public class MyClient {
public static void main(String[] args) {
try {
MyInterface obj = (MyInterface)
Naming.lookup("rmi://localhost:2099/dt");
//It initiates a connection with remote virtual machine(JVM),

System.out.println("Date is : " + obj.getDate());
System.out.println("Time is : " + obj.getTime());
System.out.println("Day is : " + obj.getDay());
} catch (Exception e) {



}
}
}
```

Server.java

```
import java.rmi.*; // RemoteException
import java.util.*; //Date()
import java.text.*; // SimpleDateFormat()
import java.rmi.server.*; // UnicastRemoteObject
public class MyServer extends UnicastRemoteObject implements MyInterface {
MyServer() throws RemoteException {
super();
}
public String getDate() throws RemoteException {
return new SimpleDateFormat("dd/MM/yyyy").format(new

Date()).toString();
```

```
}
public String getTime() throws RemoteException {
return new SimpleDateFormat("hh:mm:ss").format(new

Date()).toString();
}
public String getDay() throws RemoteException {
return new SimpleDateFormat("EEEEEEEE").format(new

Date()).toString();
}
}
```

B.Program to calculate addition,subtraction,multiplication and division from server to client. This program should display addition,subtraction,multiplication and division.(Use RMI)

First run register then client

Myinterface.java

```
import java.rmi.*;
public interface MyInterface extends Remote {

public int add(int a, int b) throws RemoteException;
public int sub(int a, int b) throws RemoteException;
public int mul(int a, int b) throws RemoteException;
public int div(int a, int b) throws RemoteException;
}
```

Myclient.java

```
//Naming
import java.rmi.Naming;
public class MyClient {
public static void main(String[] args)
{
try
{
MyInterface obj=(MyInterface)Naming.lookup("rmi://localhost:2099/dt");
//It initiates a connection with remote Virtual Machine (JVM),
int a=15;
int b=5;
System.out.println("Addition is : "+obj.add(a,b));
System.out.println("Subtraction is : "+obj.sub(a,b));
System.out.println("Multiplication is : "+obj.mul(a,b));
```

```
System.out.println("Division is : "+obj.div(a,b));

}
catch(Exception e) { }
}
}
```

MyServer.java

```
import java.rmi.*; // RemoteException
import java.util.*; //Date()
import java.text.*; // SimpleDateFormat()
import java.rmi.server.*; // UnicastRemoteObject
public class MyServer extends UnicastRemoteObject implements MyInterface {
MyServer() throws RemoteException {
super();
}


public int add(int a, int b) throws RemoteException {
return (a+b);
}
public int sub(int a, int b) throws RemoteException {
return (a-b);
}
public int mul(int a, int b) throws RemoteException {
return (a*b);
}
public int div(int a, int b) throws RemoteException {
return (a/b);
}
}
```

Register.java

```
import java.rmi.*;
import java.rmi.registry.*; // Registry, LocateRegistry
public class Register {
public static void main(String[] args) {
try {
Registry reg = LocateRegistry.createRegistry(2099);
MyServer obj = new MyServer();

Naming.rebind("rmi://localhost:2099/dt", (Remote) obj);
```

```
} catch (Exception e) {
}
}
}
```

Program 8:

A.Design a Graphical User Interface to find greatest of two numbers. Implement using RMI.

First run register then client

MyInterface.java :

```
package dscc8a;
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface MyInterface extends Remote{

	public int gretestno(int a, int b) throws RemoteException;
}
```

MyServer.java :

```
package dscc8a;


import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

public class MyServer extends UnicastRemoteObject implements MyInterface{

	protected MyServer() throws RemoteException {
		super();
		// TODO Auto-generated constructor stub
	}

	/**
	 *
	 */
	private static final long serialVersionUID = 1L;
```

```
	public static void main(String[] args) {
		// TODO Auto-generated method stub

	}

	@Override
	public int gretestno(int a, int b) {
		// TODO Auto-generated method stub
		if(a>b){
			return a;
		}
		else {
			return b;
		}
	}
}
```

Register.java

```
package dscc8a;

import java.net.MalformedURLException;
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;

public class Register {

	public static void main(String[] args) throws RemoteException,
MalformedURLException {
		// TODO Auto-generated method stub
		Registry reg=LocateRegistry.createRegistry(1234);
		MyServer obj=new MyServer();
		Naming.rebind("rmi://localhost:1234/gt", obj);

	}
```

}

MyClient.java

```
package dscc8a;

import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.net.MalformedURLException;
import java.net.Socket;
import java.rmi.Naming;
import java.rmi.NotBoundException;
import java.rmi.RemoteException;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JTextField;

public class myClient extends JFrame implements ActionListener {

	/**
	 *
	 */
	private static final long serialVersionUID = 1L;

	JLabel l1,l2,l3;
	JButton b;
	JTextField t1,t2;

	public myClient(){
		b = new JButton("Calculate");
		l1=new JLabel("Enter 1st no ");
		add(l1);
		t1 = new JTextField(20);
		add(t1);
		l2=new JLabel("Enter 2nd no ");
		add(l2);
```

```
		t2 = new JTextField(20);
		add(t2);
		b.addActionListener(this);
		add(b);
		l3=new JLabel("");
		add(l3);
	}

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		myClient c=new myClient();
		c.setLayout(new GridLayout(6,1));
		c.setSize(300,300);
		c.setVisible(true);
	}

	@Override
	public void actionPerformed(ActionEvent e) {
		// TODO Auto-generated method stub
		try {
			MyInterface
mi=(MyInterface)Naming.lookup("rmi://localhost:1234/gt");
			int a=Integer.parseInt(t1.getText().toString());

			int b=Integer.parseInt(t2.getText().toString());
			int ans=mi.gretestno(a, b);
			l3.setText("Greatest No: "+ans);

		} catch (Exception e1) {
			// TODO Auto-generated catch block
			e1.printStackTrace();
		}
	}
}
```

B.The client should provide an equation to the server through an interface. The server will solve the expression given by the client.(Here take 4 functions to solve 4 different equations)

Equation 1 : (a+b)^2=a^2+2ab+b^2

Equation 2:(a+b)^3=a^3+3a^2b+3ab^2+b^3
Equation 3(for +) and 4(for -) : ax^2+bx+c=0
First run register then client
MyInterface:

```
package dscc8;

import java.rmi.*;
public interface MyInterface extends Remote
{
public double eqOne(int a, int b) throws RemoteException;
public double eqTwo(int a, int b) throws RemoteException;
public String eqThree(int a, int b,int c) throws RemoteException;
}
```

MyServer:

```
package dscc8;
import java.rmi.*;
import java.rmi.server.*;
public class Server extends UnicastRemoteObject implements MyInterface
{
Server() throws RemoteException
{
super( );



}
public double eqOne(int a, int b) throws RemoteException{
double result;
result=Math.pow(a, 2)+(2*a*b)+Math.pow(b, 2);
return result;
}
public double eqTwo(int a, int b) throws RemoteException{
double result;
result=Math.pow(a, 3)+(3*Math.pow(a, 2)*b)+(3*a*Math.pow(b, 2))+Math.pow(b, 3);
return result;
}
public String eqThree(int a, int b,int c) throws RemoteException{
double delta=Math.pow(b,2)-(4*a*c);
if(delta==0) {
double result=(-b+Math.sqrt(delta))/(2*a);
return Double.toString(result);
}
else if(delta>0) {
```

```
double result1=(-b+Math.sqrt(delta))/(2*a);
double result2=(-b-Math.sqrt(delta))/(2*a);
return Double.toString(result1)+" and "+Double.toString(result2);
}
return "Can not determine";
}
}
```

Register:

```
package dscc8;
import java.rmi.*;



import java.rmi.registry.*;
public class Register
{
public static void main(String[] args) {
try
{
Registry reg=LocateRegistry.createRegistry(2099);
Server obj=new Server();
Naming.rebind("rmi://localhost:2099/g",obj); }
catch(Exception e)
{
System.out.println(e);
}
}
}
```

MyClient:

```
package dscc8;

import java.rmi.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class Client extends JFrame implements ActionListener {
JTextField tf1,tf2,tf3;;
JButton btn;
JLabel lb,lb1,lb2,lb3;
Choice ch = new Choice();
Client()
{
```

```
ch.setBounds(100, 100, 75, 75);



ch.add("(a+b)^2");
ch.add("(a+b)^3");
ch.add("ax^2+bx+c=0");
tf1=new JTextField(10);
tf2=new JTextField(10);
tf3=new JTextField("",10);
lb=new JLabel("");
lb1=new JLabel("a:");
lb2=new JLabel("b:");
lb3=new JLabel("c:");
btn=new JButton("Submit");
add(lb1);
add(lb1);
add(tf1);
add(lb2);
add(tf2);
add(lb3);
add(tf3);
add(ch);
add(btn);
add(lb);
btn.addActionListener(this);
}
public void actionPerformed(ActionEvent ae)
{
try
{
MyInterface obj=(MyInterface)Naming.lookup("rmi://localhost:2099/g");
int a=Integer.parseInt(tf1.getText());
int b=Integer.parseInt(tf2.getText());



int c=Integer.parseInt(tf3.getText());
int choice=ch.getSelectedIndex();
switch(choice){
case 0:
lb.setText("Ans: "+obj.eqOne(a,b));
break;
case 1:
```

```
lb.setText("Ans: "+obj.eqTwo(a,b));
break;
case 2:
lb.setText("Ans: "+obj.eqThree(a,b,c));
break;
default:
lb.setText("Please Select correct Option");
break;
}
}
catch(Exception e){}
}
public static void main(String args[])
{
Client c=new Client();
c.setLayout(new GridLayout(6,1));
c.setVisible(true);
c.setSize(300,300);
}
}
```

Program 9 : Using MySQL create Library database. Create table Book (Book_id, Book_name, Book_author) and Retrieve the Book information from the Library database using Remote Object Communication.

Start xammp server start apache and mysql go to admin create database create table insert data provide database and table name in server.java
First run register then client

Client.java

```
package dscc9book;

import java.rmi.*;

public class MyClient {
public static void main(String args[]) throws Exception {
MyInterface obj = (MyInterface)
Naming.lookup("rmi://localhost:2099/db");
String s = obj.getData();

System.out.println(s);
}
}
```

Server.java

```
package dscc9book;
```

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

public class MyServer extends UnicastRemoteObject implements
MyInterface {
String str = " ";

MyServer() throws RemoteException {
super();
}

public String getData() {

try {
Class.forName("com.mysql.jdbc.Driver");
Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3307/library","root", "");
Statement st = con.createStatement();
ResultSet rs = st.executeQuery("select * from book");
while (rs.next()) {
str += rs.getString(1) + " " +
rs.getString(2) + " " + rs.getString(3)+ " \n ";
}
} catch (Exception e) {
System.out.println(e);
}
return str;
}
}
```

Myinterface.java

```
package dscc9book;
import java.rmi.*;

public interface

MyInterface extends Remote
{
public String

getData() throws RemoteException;
}
```

Register.java

```
package dscc9book;
import java.rmi.*;
import java.rmi.registry.*;

public class Register {
public static void main(String[] args) {
try {

Registry reg =LocateRegistry.createRegistry(2099);
MyServer obj = new MyServer();
Naming.rebind("rmi://localhost:2099/db", obj);
} catch (Exception e) {
System.out.println(e);
}
}
}
```

Program 10: Using MySQL create Elecrtic_Bill database. Create table Bill (consumer_name, bill_due_date, bill_amount) and retrieve the Bill information from the Elecrtic_Bill database using Remote Object Communication

Start xammp server start apache and mysql go to admin create database create table insert data provide database and table name in server.java

First run register then client

Client.java

```
package dscc10;
import java.rmi.*;
public class MyClient
{
public static void main(String args[]) throws Exception
{
MyInterface obj=(MyInterface)Naming.lookup("rmi://localhost:2099/db");
String s=obj.getData();
 System.out.println(s);
}
}
```

Server.java

```
package dscc10;
import java.sql.*;
import java.rmi.*;
import java.rmi.server.*;
```

```
public class MyServer extends UnicastRemoteObject implements MyInterface  {
String str=" ";
MyServer() throws RemoteException
{
super();
}
public String getData()
{
try
{
Class.forName("com.mysql.jdbc.Driver");
Connection con=DriverManager.getConnection("jdbc:mysql://localhost/electricity_bill","root","");
Statement st=con.createStatement();
ResultSet rs=st.executeQuery("select * from bill");
while(rs.next())
{
str += rs.getString(1) + " " +
rs.getString(2) + " " + rs.getString(3)+ " \n ";

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

Myinterface.java

```
package dscc10;
import java.rmi.*;
public interface MyInterface extends Remote
{
public String getData() throws RemoteException;
}
```

Register.java

```
package dscc10;
import java.rmi.*;
import java.rmi.registry.*;
public class Register
{
```

```
public static void main(String[] args)
{
try
{
Registry reg=LocateRegistry.createRegistry(2099);
MyServer obj=new MyServer();
Naming.rebind("rmi://localhost:2099/db",obj);
}
catch(Exception e)
{
System.out.println(e);
}
}
}
```

Practical No. 11 Implementation of Mutual Exclusion using Token Ring Technique

First run register then client1 then client2

Client.java

```
package dsccc11;
import java.net.*;
import java.io.*;
class TokenClient1
{
public static DatagramSocket ds;
public static DatagramPacket dp;
public static BufferedReader br;
public static void main(String args[])throws Exception
{

boolean hasToken=true;
ds=new
DatagramSocket(100);

while(true)
{
if(hasToken==true)
{
System.out.println("Do you want to write data...(yes/no)");
br=new BufferedReader(new
InputStreamReader(System.in)); String ans=br.readLine();
if(ans.equalsIgnoreCase("yes"))
{
System.out.println("ready to write");
```

```
System.out.println("enter the data");
br=new BufferedReader(new

InputStreamReader(System.in)); String str="Client-1===>"+br.readLine(); byte buff[]=new
byte[1024];

buff=str.getBytes(); ds.send(new
DatagramPacket(buff,buff.length,InetAddress.getLocalHost(),1000));
System.out.println("now sending");
}
else if(ans.equalsIgnoreCase("no"))
{
System.out.println("I am Busystate");
String msg="token"; byte
bf1[]=new byte[1024];
bf1=msg.getBytes(); ds.send(new
DatagramPacket(bf1,bf1.length,InetAddress.getLocalHost(),200));
hasToken=false;
}
}

else
{
System.out.println("Entering in receiving mode");
byte bf[]=new byte[1024];
ds.receive(dp=new DatagramPacket(bf,bf.length));
String clientmsg=new String(dp.getData(),0,dp.getLength());
System.out.println("The data is "+clientmsg);
if(clientmsg.equals("token"))
{
hasToken=true;
System.out.println("I am leaving busy state");
}
}
}
}
}
```

Server.java

```
package dsccc11;
import java.net.*;
import java.io.*;

class TokenServer
```

```
{
public static DatagramSocket ds;
public static DatagramPacket dp;
public static void main(String[]
args)throws Exception
{
ds=new DatagramSocket(1000);
while(true)
{
byte buff[]=new byte[1024];
ds.receive(dp=new DatagramPacket(buff,buff.length));
String str=new String(dp.getData(),0,dp.getLength());
System.out.println("Message from " +str);

}
}

}
```

Client2.java

```
package dsccc11;
import java.net.*;
import java.io.*;
class  TokenClient2
{
public static DatagramSocket ds;
public static  DatagramPacket dp;
public static BufferedReader  br;
public static void main(String args[])throws  Exception
{
boolean hasToken=false;
ds=new DatagramSocket(200);
System.out.println("158Ankita Tripathi");
while(true)
{
if(hasToken==true)
{
System.out.println("Do you want to write data...(yes/no)");
br=new BufferedReader(new
InputStreamReader(System.in));
String ans=br.readLine();
if(ans.equalsIgnoreCase("yes"))
{
System.out.println("ready to write");
```

```
System.out.println("enter the data");
br=new BufferedReader(new
InputStreamReader(System.in));
String str="Client 2===>"+br.readLine();
byte buff[]=new byte[1024];
buff=str.getBytes();
ds.send(new
DatagramPacket(buff,buff.length,InetAddress.getLocalHost(),1000));
}
else if(ans.equalsIgnoreCase("no"))
{
System.out.println("I am Busy  state");
String msg="token"; byte
bf1[]=new byte[1024];
bf1=msg.getBytes();
ds.send(new
DatagramPacket(bf1,bf1.length,InetAddress.getLocalHost(),100));
hasToken=false;
}
}
else
{
try
{
System.out.println("Entering in receiving mode");
byte bf[]=new byte[1024];
ds.receive(dp=new DatagramPacket(bf,bf.length));
String clientmsg=new String(dp.getData(),0,dp.getLength());
System.out.println("The data is "+clientmsg);
if(clientmsg.equals("token"))
{
hasToken=true;
System.out.println("I am leaving busy state");
}
}
catch(Exception e){}
}
}
}
}
```

Practical No. 12 Implementation of Storage as a Service using Google Docs

Practical No. 13 Develop application for Microsoft Azure using Microsoft Studio 2019

https://docs.google.com/document/d/1sTDgyt76vdptx6THuvFcj_mkd0f3OHxF1G3dq36bf4g/edit?usp=sharing

Practical No. 14 Develop application for Google App Engine Eclipse IDE

https://docs.google.com/document/d/1CX8oqRtStH6m0aM6ktJnl80TixAd6gzS066bvq6ZxkE/edit?usp=sharing

Practical No. 15 Study and implementation of Identity Management

https://docs.google.com/document/d/14j147wzvCX8FWXcQ4Zdh_C5sV8k_HyMckboe6UHEv84/edit?usp=sharing