# Index

# Index

**Practical No. 1**

**Aim:** Write a java program to implement Chatting Application

**BroadcastClient:**

```
import java.net.*;

import java.io.*;

public class BroadcastClient {

 public static final int PORT = 1234;

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

 String ip = "localhost";

 MulticastSocket  socket;

 DatagramPacket packet;

 InetAddress address;

 address = InetAddress.getByName(ip);

 socket = new MulticastSocket(PORT);

 // join a multicast group

 //socket.joinGroup(address);

 byte[] data = new byte[100];

 packet = new DatagramPacket(data, data.length);

 for (;;) {

 // recive the packet

 socket.receive(packet);

 String str = new String(packet.getData());

System.out.println("Message Recived from" + packet.getAddress() +

"Message is :- " + str);

 }

 }

}
```

**BroadcastServer :**

```
import java.net.*;
import java.io.*;
import java.util.*;
public class BroadcastServer {
 public static final int PORT = 1234;
 public static void main(String args[]) throws Exception {
 MulticastSocket socket;
 DatagramPacket packet;
 InetAddress address;
 address = InetAddress.getByName("localhost");
 socket = new MulticastSocket();
 // join a multicast group and send the group message
 //socket.joinGroup(address);
 byte[] data = null;
 for (;;) {
 Thread.sleep(10 * 1000);
System.out.println("Sending ");
String str = "This is Rv calling...";
data = str.getBytes();
 packet = new DatagramPacket(data, str.length(), address, PORT);
 // send the packet
 socket.send(packet);
 }
 }
}
```

OUTPUT:

```
C:\Users\User11\Desktop>java BroadcastServer
Sending
Sending
Sending
Sending
Sending
```

```
C:\Users\User11\Desktop>java BroadcastClient
Message Recived from/127.0.0.1Message is :- This is Rv calling...

Message Recived from/127.0.0.1Message is :- This is Rv calling...

Message Recived from/127.0.0.1Message is :- This is Rv calling...
```

## Practical No.2

**Aim:** Write an RPC code for implementing simple calculator toperform basiccalculation.

**Calculator server:**

```
import java.rmi.*;
import java.rmi.server.*;
public class CalculatorServer extends UnicastRemoteObject implements Calculator
{
	public CalculatorServer() throws RemoteException
	{
		System.out.println("Server is Instantiated");
	}
	public int sum(int first, int Second) throws RemoteException
	{
		return first + Second;
	}
	public int sub(int first, int Second) throws RemoteException
	{
		return first - Second;
	}
	public int mul(int first, int Second) throws RemoteException
	{
		return first * Second;
	}
	public static void main(String arg[])
	{
		try
		{
			CalculatorServer p = new CalculatorServer();
```

```
                Naming.rebind("Cal", p);
            }
            catch (Exception e)
            {
                System.out.println("Exception occurred : " + e.getMessage());
            }
        }
}
```

**Calculator Client:**

```
import java.rmi.*;
import java.io.*;
public class CalculatorClient
{
    public static void main(String args[])
    {
        try
        {
            BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
            Calculator p = (Calculator) Naming.lookup("Cal");
            System.out.println("Enter fisrt number");
            String strFirst = br.readLine();
            System.out.println("Enter second number");
            String strSecond = br.readLine();
            System.out.println("Sum :" + p.sum(Integer.parseInt(strFirst),
            Integer.parseInt(strSecond)));
            System.out.println("Sub :" + p.sub(Integer.parseInt(strFirst),
            Integer.parseInt(strSecond)));
            System.out.println("Mul :" + p.mul(Integer.parseInt(strFirst),
```

```
                Integer.parseInt(strSecond)));
        }
        catch (Exception e)
        {
                System.out.println("Exception occurred : " + e.getMessage());
        }
    }
}
```

**Calculator:**

```
import java.rmi.*;
public interface Calculator extends Remote
{
    public int sum(int a, int b) throws RemoteException;
    public int sub(int a, int b) throws RemoteException;
    public int mul(int a, int b) throws RemoteException;
}
```

OUTPUT:

```
C:\Users\User11\Desktop\pract2>java CalculatorServer
Server is Instantiated
```

```
C:\Users\User11\Desktop\pract2>java CalculatorClient
Enter fisrt number
12
Enter second number
22
Sum :34
Sub :-10
Mul :264
```

```
C:\Users\User11\Desktop\pract2>rmiregistry
```

## Practical No.3

**Aim:** Write a java program to find the current date and time using RPC

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

class DateServer{
    public static void main(String args[]) throws Exception{
        ServerSocket s = new ServerSocket(5217);

        while(true){
            System.out.println("Waiting for connection... ");
            Socket soc = s.accept();
            DataOutputStream out = new DataOutputStream(soc.getOutputStream());
            out.writeBytes("Server Date :"+(new Date()).toString()+"\n");
            out.close();
            soc.close();
        }
    }
}
```

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

class DateClient
{
    public static void main(String args[]) throws Exception
    {
        Socket soc = new Socket(InetAddress.getLocalHost(), 5217);
        BufferedReader in = new BufferedReader(new
InputStreamReader(soc.getInputStream()));
        System.out.println(in.readLine());
    }
}
```

dtss — java DateServer — 150×39

~/Desktop/dtss — rmiregistry | ~/Desktop/dtss — java DateServer | ~/Desktop/dtss — -zsh

```
rv@Rvs-iMac dtss % java DateServer
Waiting for connection....
Waiting for connection....
```

dtss — -zsh — 150×39

~/Desktop/dtss — rmiregistry | ~/Desktop/dtss — java DateServer | ~/Desktop/dtss — -zsh

```
rv@Rvs-iMac dtss % java DateClient
Server Date :Wed Mar 29 11:38:21 IST 2023
rv@Rvs-iMac dtss %
```

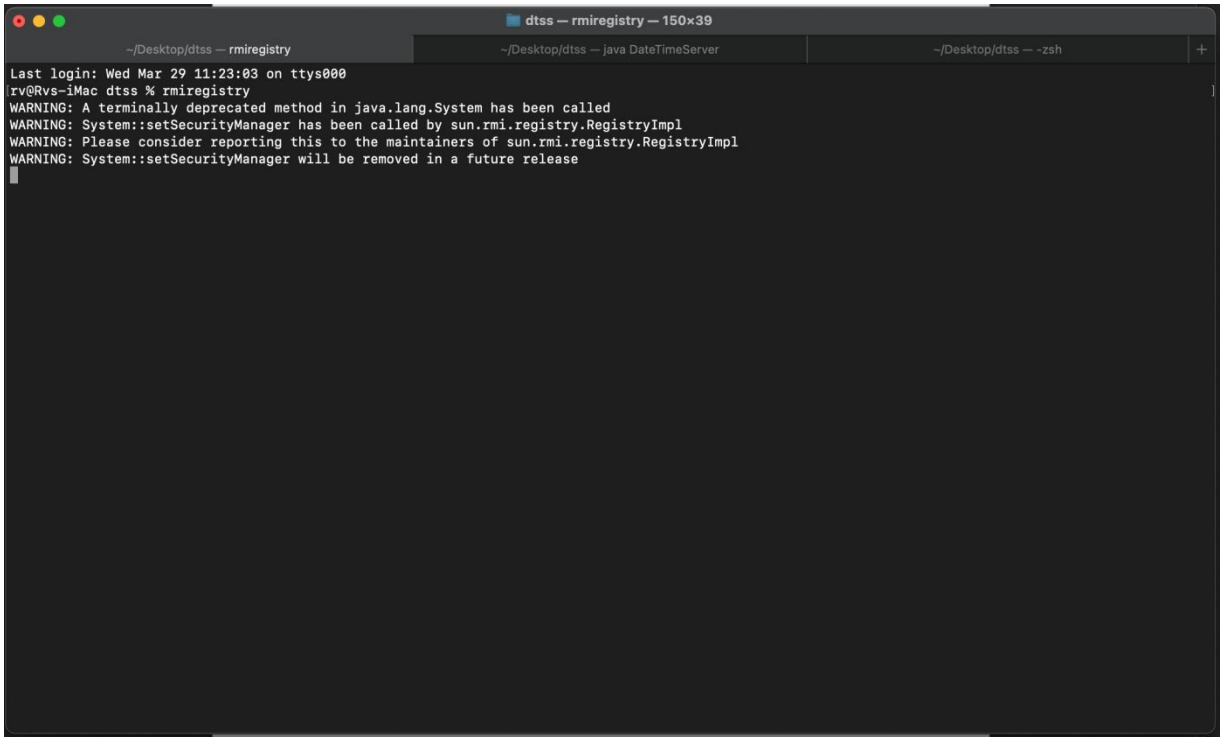
**Practical No.4**

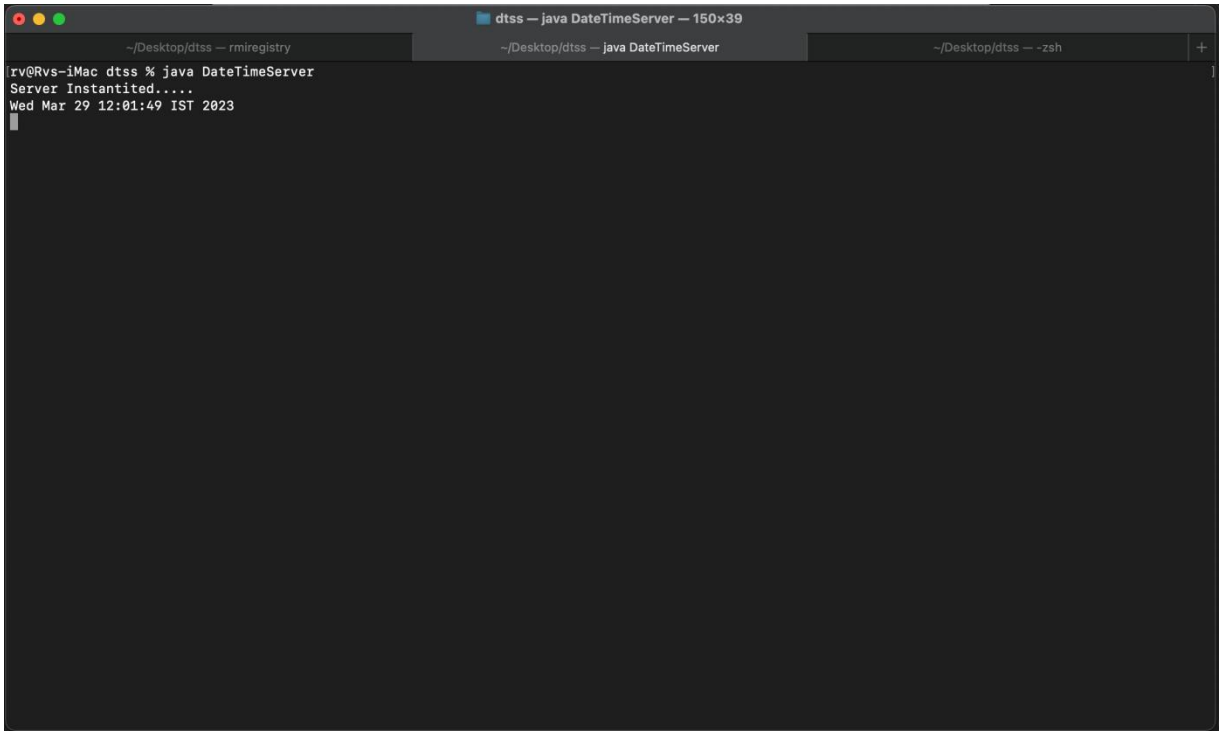
**Aim:** Write an RMI code to Retrieve time and date function from server to client.

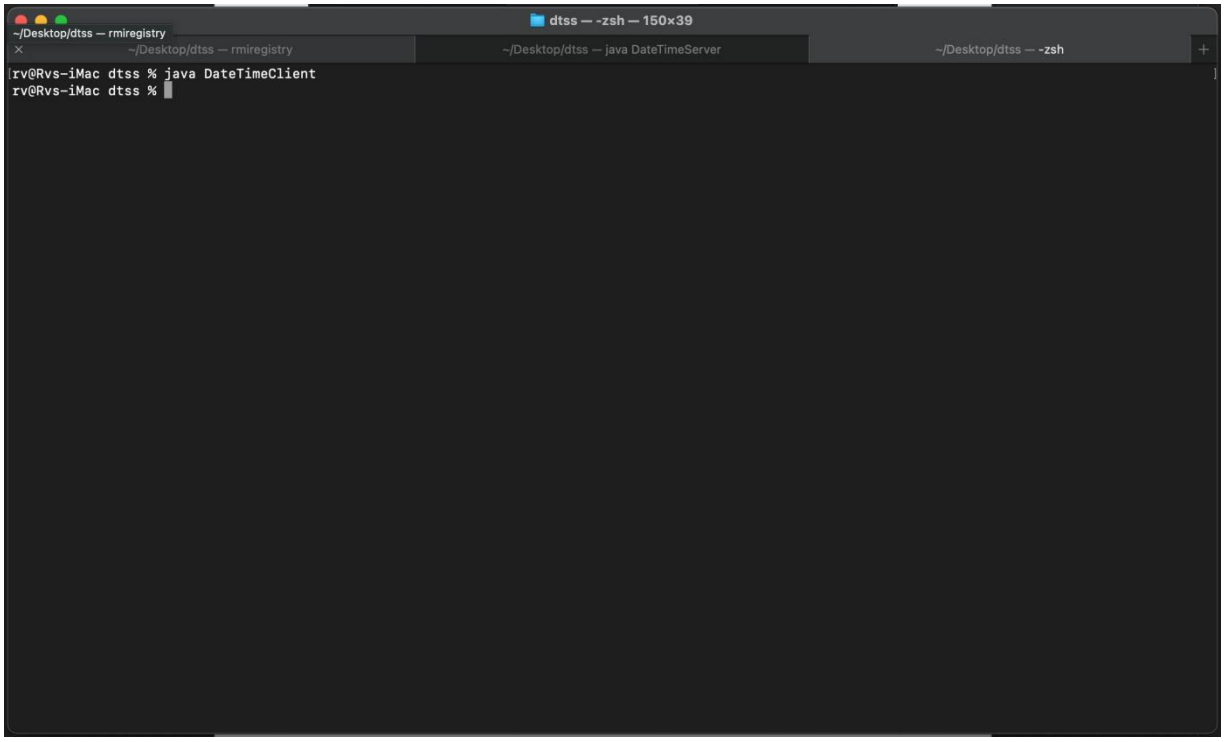
```java
import java.rmi.Remote;

public interface DateTime extends Remote {
    public void printDateTime() throws Exception;
}


import java.rmi.Naming;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.util.Date;

public class DateTimeServer extends UnicastRemoteObject implements
DateTime {
    public DateTimeServer() throws RemoteException {
        System.out.println("Server Instantited. ...");
    }

    @Override
    public void printDateTime() throws Exception {
        System.out.println((new Date()).toString());
    }

    public static void main(String[] args) {
        try {
            DateTimeServer server = new DateTimeServer();
            Naming.rebind("rmi://localhost:1099/DateTimeService", server);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

}


import java.rmi.Naming;

public class DateTimeClient {
    public static void main(String[] args) {
        try {
            DateTime dateTime = (DateTime)
Naming.lookup("rmi://localhost:1099/DateTimeService");
            dateTime.printDateTime();
        } catch (Exception e) {
```

```
            e.printStackTrace();
        }
    }
}
```

Output:




dtss — -zsh — 150×39
~/Desktop/dtss — rmiregistry
~/Desktop/dtss — rmiregistry
~/Desktop/dtss — java DateTimeServer
~/Desktop/dtss — -zsh
rv@Rvs-iMac dtss % java DateTimeClient
rv@Rvs-iMac dtss %

## Practical No.5

**Aim:** Write an RMI code to implement equation solver using RMI((a-b)2=a2-2ab+b2).

**Calculator:**

```
import java.rmi.*;
public interface Calculator extends Remote
{
        public int sum(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 double calculateExpression(double a, double b)  throws RemoteException;

}
```

**CalculatorClient:**

```
import java.rmi.*;
import java.io.*;
public class CalculatorClient
{
        public static void main(String args[])
        {
                try
                {
                        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
                        Calculator p = (Calculator) Naming.lookup("Cal");
                        System.out.println("Enter fisrt number");
                        String strFirst = br.readLine();
                        System.out.println("Enter second number");
                        String strSecond = br.readLine();
```

```
                System.out.println("Sum :" + p.sum(Integer.parseInt(strFirst),
                Integer.parseInt(strSecond)));
                System.out.println("Sub :" + p.sub(Integer.parseInt(strFirst),
                Integer.parseInt(strSecond)));
                System.out.println("Mul :" + p.mul(Integer.parseInt(strFirst),
                Integer.parseInt(strSecond)));
System.out.println("(a+b)^2  :" + p.calculateExpression(Double.parseDouble(strFirst),
                Double.parseDouble(strSecond)));
            }
            catch (Exception e)
            {
                System.out.println("Exception occurred : " + e.getMessage());
            }
        }
}
```

**CalculatorServer:**

```
import java.rmi.*;
import java.rmi.server.*;
public class CalculatorServer extends UnicastRemoteObject implements Calculator
{
        public CalculatorServer() throws RemoteException
        {
            System.out.println("Server is Instantiated");
        }
        public int sum(int first, int Second) throws RemoteException
        {
            return first + Second;
        }
```

```
        public int sub(int first, int Second) throws RemoteException
        {
                return first - Second;
        }
        public int mul(int first, int Second) throws RemoteException
        {
                return first * Second;
        }
        public double calculateExpression(double a, double b) throws RemoteException {

    double result = Math.pow(a, 2) + Math.pow(b, 2) + 2 * a * b;
    return result;
  }
        public static void main(String arg[])
        {
                try
                {
                        CalculatorServer p = new CalculatorServer();
                        Naming.rebind("Cal", p);
                }
                catch (Exception e)
                {
                        System.out.println("Exception occurred : " + e.getMessage());
                }
        }
}
```

Output:

C:\Windows\System32\cmd.exe - rmiregistry

```
Microsoft Windows [Version 10.0.19045.4412]
(c) Microsoft Corporation. All rights reserved.

C:\Users\User11\Desktop\pract2 - Copy (2)>rmiregistry
```

```
C:\Users\User11\Desktop\pract2 - Copy (2)>javac CalculatorServer.java

C:\Users\User11\Desktop\pract2 - Copy (2)>java CalculatorServer
Server is Instantiated
```

```
C:\Users\User11\Desktop\pract2 - Copy (2)>javac CalculatorClient.java

C:\Users\User11\Desktop\pract2 - Copy (2)>java CalculatorClient
Enter fisrt number
6
Enter second number
3
Sum :9
Sub :3
Mul :18
(a+b)^2  :81.0
```

**Practical No.6**

**Aim:** Write a java program to retrieve the list of books available in the library using concept of RMI using Mysql

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

interface Library extends Remote {
    public String[] getResult() throws RemoteException;

    public int getCount() throws RemoteException;
}


import java.rmi.*;
import java.net.*;
import java.rmi.server.*;
import java.sql.*;
import java.io.*;

class LibraryImpl extends UnicastRemoteObject implements Library,
Serializable {

    public static final String DRIVER = "com.mysql.jdbc.Driver";
    public static final String URL =
"jdbc:mysql://localhost:3306/library";
    public static final String USERNAME = "root";
    public static final String PASSWORD = "";

    Connection con = null;
    Statement stmt = null;
    ResultSet rs = null;
    ResultSetMetaData rsmd;
    int cnt = 0;

    public LibraryImpl() throws RemoteException {
        try {
            Class.forName(DRIVER);
            con = DriverManager.getConnection(URL, USERNAME, PASSWORD);
            stmt = con.createStatement();
        } catch (Exception ar) {
            System.out.println("Error At LibraryImpl() :");
            ar.printStackTrace();
        }

    }
```

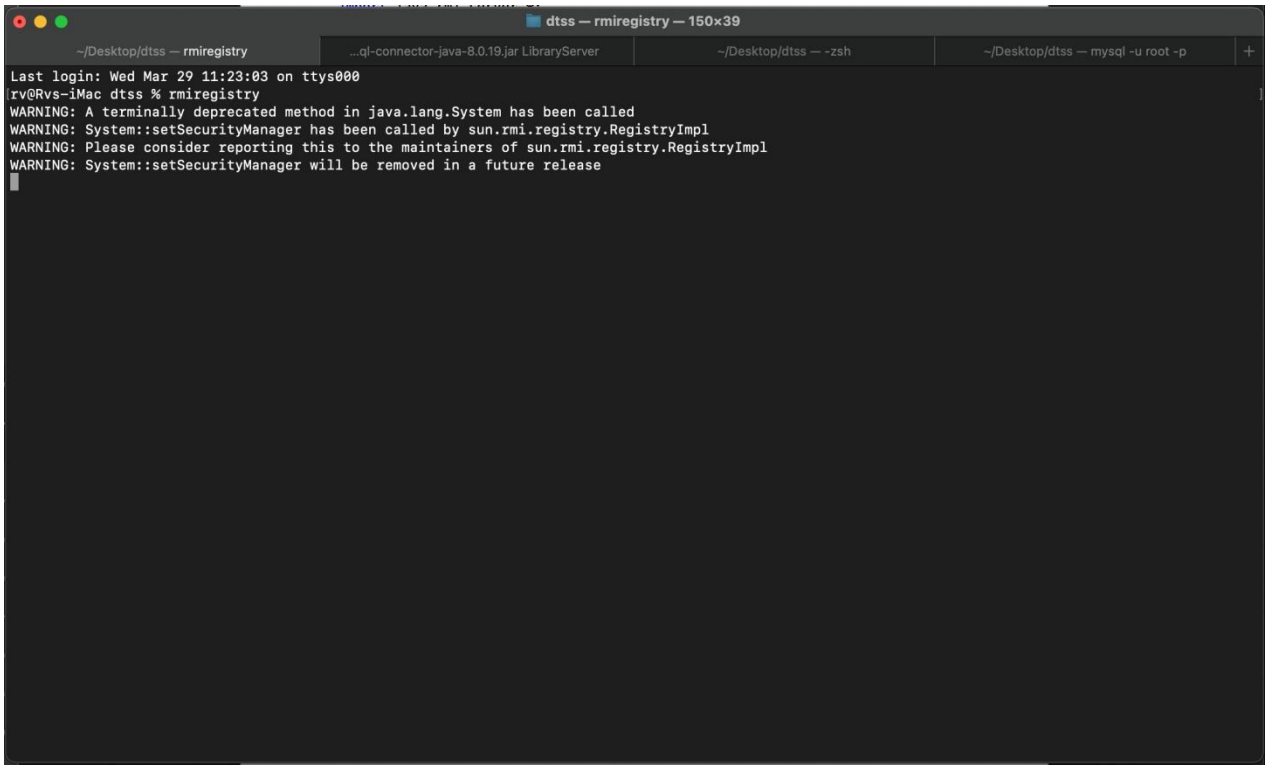
```java
    int getCounty() {
        int count = 0;
        try {
            ResultSet rs1 = stmt.executeQuery("Select * From Books");
            while (rs1.next()) {
                count++;
            }
        } catch (Exception e) {
            System.out.println("getCounty:");
            e.printStackTrace();
        }
        return count;
    }

    public String[] getResult() {
        int county = getCounty();
        String str[] = new String[county];
        try {
            rs = stmt.executeQuery("Select * From Books");
            rsmd = rs.getMetaData();
            int col = rsmd.getColumnCount();
            int j = 0;
            int count = 0;
            String temp = "";
            while (rs.next()) {
                cnt++;
                str[j] = "";
                for (int i = 1; i <= col; i++) {
                    temp = rs.getString(i);
                    str[j] = str[j] + temp + "\t";
                }
                j++;
            }
        } catch (Exception e) {
            System.out.println("Error At getResult:");
            e.printStackTrace();
        }
        return str;
    }

    public int getCount() {
        return cnt;
    }
}
```

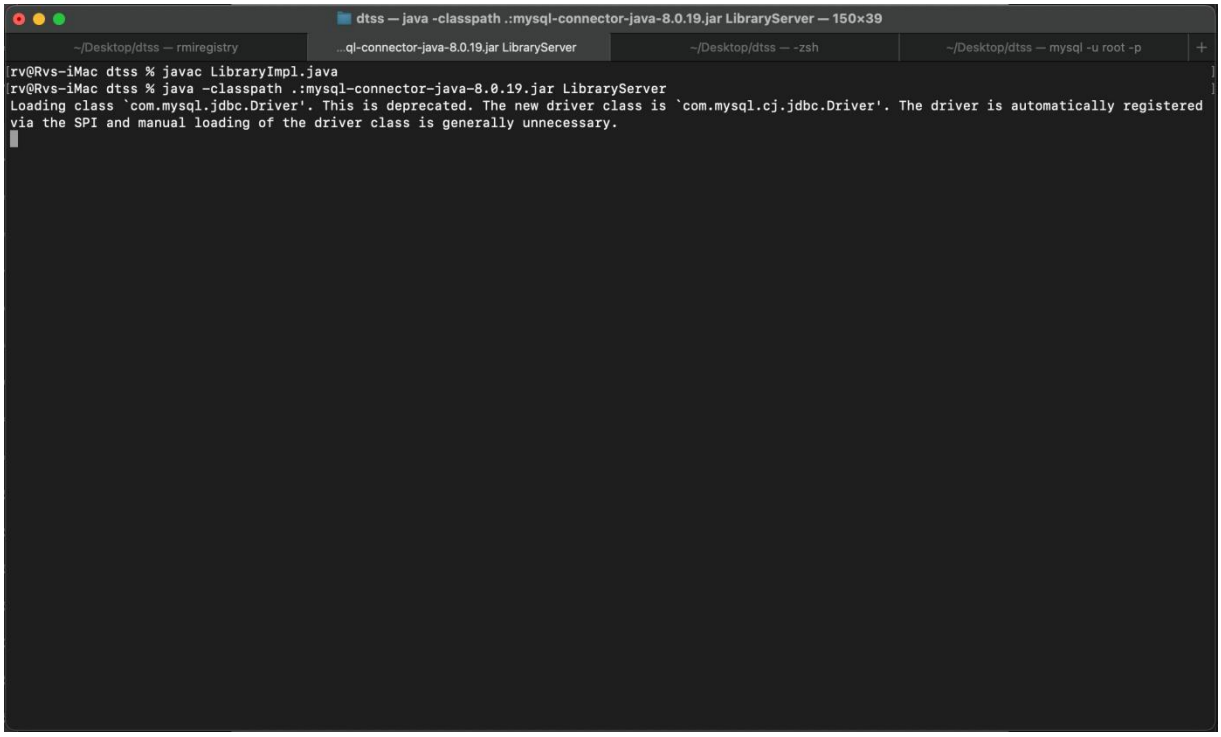
```java
import java.rmi.*;
import java.net.*;
import java.sql.*;
import java.io.*;

class LibraryClient implements Serializable {
    public static void main(String args[]) {

        try {

            ResultSet rs1;
            ResultSetMetaData rsmd;
            System.out.println("This is client screen");

            Library ob = (Library) Naming.lookup("Lib");
            String str[] = ob.getResult();
            int colcnt = ob.getCount();
            for (int i = 0; i < colcnt; i++) {

                System.out.println(str[i]);

            }

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

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

class LibraryServer implements Serializable {
    public static void main(String args[]) {
        try {
            LibraryImpl ob1 = new LibraryImpl();
            Naming.rebind("Lib", ob1);
        } catch (Exception ar) {
            ar.printStackTrace();
        }
    }
```

}

Output:


dtss — rmiregistry — 150×39
~/Desktop/dtss — rmiregistry
...ql-connector-java-8.0.19.jar LibraryServer
~/Desktop/dtss — -zsh
~/Desktop/dtss — mysql -u root -p
Last login: Wed Mar 29 11:23:03 on ttys000
rv@Rvs-iMac dtss % rmiregistry
WARNING: A terminally deprecated method in java.lang.System has been called
WARNING: System::setSecurityManager has been called by sun.rmi.registry.RegistryImpl
WARNING: Please consider reporting this to the maintainers of sun.rmi.registry.RegistryImpl
WARNING: System::setSecurityManager will be removed in a future release


dtss — java -classpath .:mysql-connector-java-8.0.19.jar LibraryServer — 150×39
~/Desktop/dtss — rmiregistry
...ql-connector-java-8.0.19.jar LibraryServer
~/Desktop/dtss — -zsh
~/Desktop/dtss — mysql -u root -p
rv@Rvs-iMac dtss % javac LibraryImpl.java
rv@Rvs-iMac dtss % java -classpath .:mysql-connector-java-8.0.19.jar LibraryServer
Loading class `com.mysql.jdbc.Driver'. This is deprecated. The new driver class is `com.mysql.cj.jdbc.Driver'. The driver is automatically registered via the SPI and manual loading of the driver class is generally unnecessary.

dtss — mysql -u root -p — 150×39

~/Desktop/dtss — rmiregistry | ...ql-connector-java-8.0.19.jar LibraryServer | ~/Desktop/dtss — -zsh | ~/Desktop/dtss — mysql -u root -p

```
mysql> select * from books
    -> ;
+-----------------------------------------------------+----+-------+
| name                                                | id | price |
+-----------------------------------------------------+----+-------+
| In Search of Lost Time by Marcel Proust             |  1 |   500 |
| Ulysses by James Joyce                              |  2 |   600 |
| Don Quixote by Miguel de Cervantes                  |  3 |   700 |
| One Hundred Years of Solitude by Gabriel Garcia Marquez |  4 |   800 |
| War and Peace by Leo Tolstoy                        |  5 |   900 |
+-----------------------------------------------------+----+-------+
5 rows in set (0.00 sec)

mysql>
```

dtss — -zsh — 150×39

~/Desktop/dtss — rmiregistry | ...ql-connector-java-8.0.19.jar LibraryServer | ~/Desktop/dtss — -zsh | ~/Desktop/dtss — mysql -u root -p

```
rv@Rvs-iMac dtss % java LibraryClient
This is client screen
rv@Rvs-iMac dtss % java LibraryClient
This is client screen
In Search of Lost Time by Marcel Proust 1       500
Ulysses by James Joyce  2       600
Don Quixote by Miguel de Cervantes      3       700
One Hundred Years of Solitude by Gabriel Garcia Marquez 4       800
War and Peace by Leo Tolstoy    5       900
rv@Rvs-iMac dtss %
```

**Practical No.7**

**Aim:** Write a java program to retrieve electricity bill information using concept of RMIusing Mysql.

```
public interface Electricity extends Remote{
    public String[] getBills() throws RemoteException;
}


import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.Statement;

public class ElectricityImpl extends UnicastRemoteObject implements
Electricity {

    public static final String DRIVER = "com.mysql.jdbc.Driver";
    public static final String URL =
"jdbc:mysql://localhost:3306/Electicity";
    public static final String USERNAME = "root";
    public static final String PASSWORD = "";

    Connection con = null;
    Statement stmt = null;
    ResultSetMetaData rsmd;
    ResultSet rs = null;
    int cnt = 0;

    public ElectricityImpl() throws RemoteException{
        try {
            Class.forName(DRIVER);
            con = DriverManager.getConnection(URL, USERNAME, PASSWORD);
            stmt = con.createStatement();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private int getCount(){
        int counter = 0;
        try {
            ResultSet resultSet = stmt.executeQuery("SELECT * FROM Bill");
            while(resultSet.next()) counter++;
```

```
        } catch (Exception e) {
            e.printStackTrace();
        }
        return counter;
    }

    @Override
    public String[] getBills() throws RemoteException {
        int county = getCount();
        String str[] = new String[county];
        try {
            rs = stmt.executeQuery("SELECT * FROM Bill");
            rsmd = rs.getMetaData();
            int col = rsmd.getColumnCount();
            int j = 0;
            int count = 0;
            String temp = "";
            while (rs.next()) {
                cnt++;
                str[j] = "";
                for (int i = 1; i <= col; i++) {
                    temp = rs.getString(i);
                    str[j] = str[j] + temp + "\t";
                }
                j++;
            }
        } catch (Exception e) {
            System.out.println("Error At getResult:");
            e.printStackTrace();
        }
        return str;
    }

}


import java.io.Serializable;
import java.rmi.Naming;

public class ElectricityServer implements Serializable{
    public static void main(String[] args) {
        try {
            ElectricityImpl electricityImpl = new ElectricityImpl();
            Naming.rebind("electricity", electricityImpl);
        } catch (Exception e) {
            e.printStackTrace();
```

```
        }
    }

}

import java.io.Serializable;
import java.rmi.Naming;

public class ElectricityClient implements Serializable{
    public static void main(String[] args) {
        try {
            System.out.println("This is client screen");
            Electricity obj = (Electricity) Naming.lookup("electricity");
            String str[] = obj.getBills();
            for (String bill : str) {
                System.out.println(bill);
            }
        } catch (Exception e) {
            e.addSuppressed(e);
        }
    }
}
```

Output:

Start rmiregistry

```
dtss — rmiregistry — 122×35
~/Desktop/dtss — rmiregistry   ...Desktop/dtss — mysql -u root -p   ...va-8.0.19.jar ElectricityServer   ~/Desktop/dtss — -zsh

rv@Rvs-iMac ~ % cd /Users/rv/Desktop/dtss
rv@Rvs-iMac dtss % rmiregistry
WARNING: A terminally deprecated method in java.lang.System has been called
WARNING: System::setSecurityManager has been called by sun.rmi.registry.RegistryImpl
WARNING: Please consider reporting this to the maintainers of sun.rmi.registry.RegistryImpl
WARNING: System::setSecurityManager will be removed in a future release
```

MySql Screen

```
dtss — mysql -u root -p — 122×35
~/Desktop/dtss — rmiregistry   ...Desktop/dtss — mysql -u root -p   ...va-8.0.19.jar ElectricityServer   ~/Desktop/dtss — -zsh

rv@Rvs-iMac dtss % mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 8.0.32 Homebrew

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use Electicity
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from Bill
    -> ;
+----+------------+---------------+--------+
| id | account_no | customer_name | amount |
+----+------------+---------------+--------+
|  1 |         39 | Yohannes      |    500 |
|  2 |         82 | Kwadwo        |   1200 |
|  3 |         54 | Isolda        |   1500 |
|  4 |         60 | Lycurgus      |   1900 |
|  5 |         51 | Lèja          |   2000 |
+----+------------+---------------+--------+
5 rows in set (0.00 sec)

mysql>
```

ElectricityServer

```
dtss — java -classpath .:mysql-connector-java-8.0.19.jar ElectricityServer — 122×35
~/Desktop/dtss — rmiregistry | ...Desktop/dtss — mysql -u root -p | ...va-8.0.19.jar ElectricityServer | ~/Desktop/dtss — -zsh
rv@Rvs-iMac dtss % java -classpath .:mysql-connector-java-8.0.19.jar ElectricityServer
Loading class `com.mysql.jdbc.Driver'. This is deprecated. The new driver class is `com.mysql.cj.jdbc.Driver'. The driver
is automatically registered via the SPI and manual loading of the driver class is generally unnecessary.
```

ElectricityClient

```
dtss — -zsh — 122×35
~/Desktop/dtss — rmiregistry | ...Desktop/dtss — mysql -u root -p | ...va-8.0.19.jar ElectricityServer | ~/Desktop/dtss — -zsh
rv@Rvs-iMac dtss % java ElectricityClient
This is client screen
1       39      Yohannes        500
2       82      Kwadwo  1200
3       54      Isolda  1500
4       60      Lycurgus        1900
5       51      Lėja    2000
rv@Rvs-iMac dtss %
```

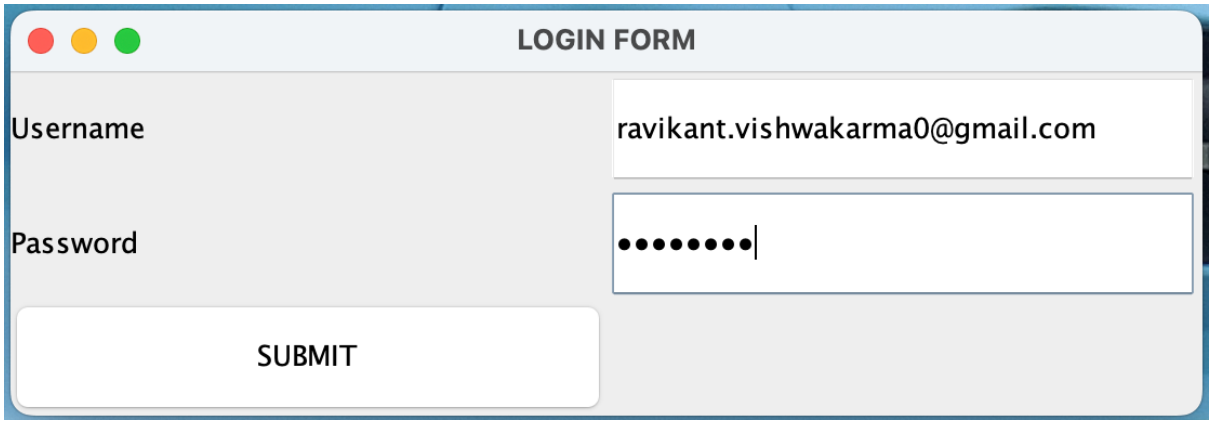
**Practical No.8**

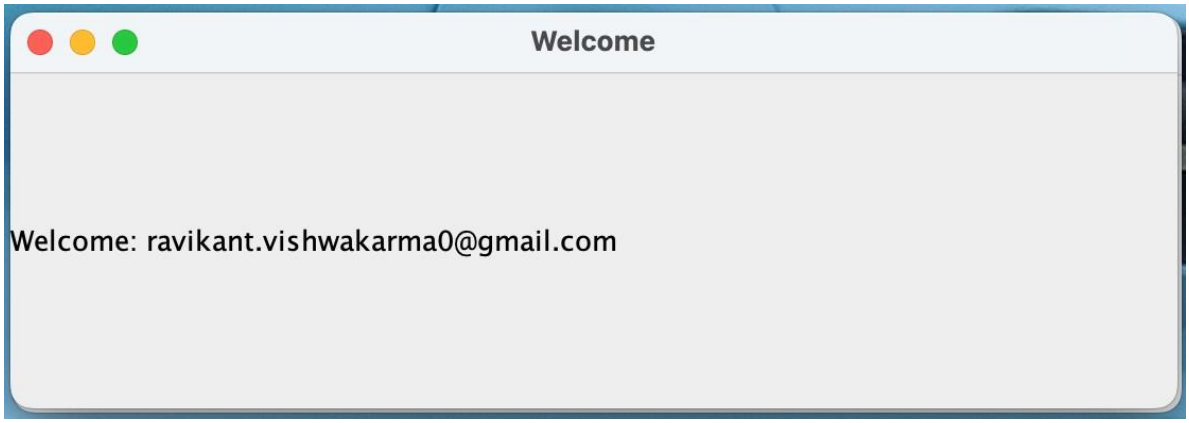
**Aim:** Write a Java Program to check their Authentication of User

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

class CreateLoginForm extends JFrame implements ActionListener {
    JButton b1;
    JPanel newPanel;
    JLabel userLabel, passLabel;
    final JTextField textField1, textField2;

    CreateLoginForm()  {
        userLabel = new JLabel();
        userLabel.setText("Username"); // set label value for textField 1
        textField1 = new JTextField(15); // set length of the text field 1

        passLabel = new JLabel();
        passLabel.setText("Password"); // set label value for textField 2
        textField2 = new JPasswordField(15); // set length for the
password field

        b1 = new JButton("SUBMIT"); // set label to button
        newPanel = new JPanel(new GridLayout(3, 1));
        newPanel.add(userLabel); // add username label to panel
        newPanel.add(textField1); // add text field 1 to panel
        newPanel.add(passLabel); // add password label to panel
        newPanel.add(textField2); // add text field 2 to panel
        newPanel.add(b1); // add button to panel

        add(newPanel, BorderLayout.CENTER);
        b1.addActionListener(this); // add action listener to button
        setTitle("LOGIN FORM"); // set title to the login form
    }

    public void actionPerformed(ActionEvent ae) {
        String userValue = textField1.getText();
        String passValue = textField2.getText();

        // check whether the credentials are authentic or not
        if (userValue.equals("admin@gmail.com") &&
passValue.equals("password")) {
            // if authentic, navigate user to a new page

            // create instance of the NewPage
            NewPage page = new NewPage();
```

```
            // create a welcome label and set it to the new page
            JLabel wel_label = new JLabel("Welcome: " + userValue);
            page.getContentPane().add(wel_label);

            // make page visible to the user
            page.setVisible(true);
        } else {
            // show error message
            JOptionPane.showMessageDialog(null, "Please enter valid
username and password");
        }
    }
}

class NewPage extends JFrame {
    NewPage() {
        setSize(300, 100);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setTitle("Welcome");
    }
}

class LoginFormDemo {
    public static void main(String arg[]) {
        try {
            // create instance of the CreateLoginForm
            CreateLoginForm form = new CreateLoginForm();
            form.setSize(300, 100); // set size of the frame
            form.setVisible(true);
        } catch (Exception e) {
            // handle exception
            JOptionPane.showMessageDialog(null, e.getMessage());
        }
    }
}
```

Output:


LOGIN FORM
Username
ravikant.vishwakarma0@gmail.com
Password
SUBMIT


Welcome
Welcome: ravikant.vishwakarma0@gmail.com

**Practical No.9**

**Aim:** Implementation of mutual exclusion using the Token ring algorithm

TokenServer.java

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

class TokenServer {
    public static DatagramSocket ds;
    public static DatagramPacket dp;

    public static void main(String[] args) {
        try {
            ds = new DatagramSocket(1000);
        } catch (Exception e) {
            e.printStackTrace();
        }
        while (true) {
            byte buff[] = newbyte[1024];
            ds.receive(dp = new DatagramPacket(buff, buff.length));
            tring str = new String(dp.getData(), 0, dp.getLength());
            System.out.println("Messagefrom" + str);
        }
    }

}
```

TokenClient1.java

```
import java.net.*;
import java.io.*;
class TokenClient1{
    public static DatagramSocket ds;
    public static DatagramPacket dp;
    public static BufferedReader br;
    static int cp = 100;

    public static void main(String[] args) {
        boolean hasToken;
        try {
            ds=new DatagramSocket(100);
        } catch (Exception e) {
            e.printStackTrace();
        }
        hasToken = true;
```

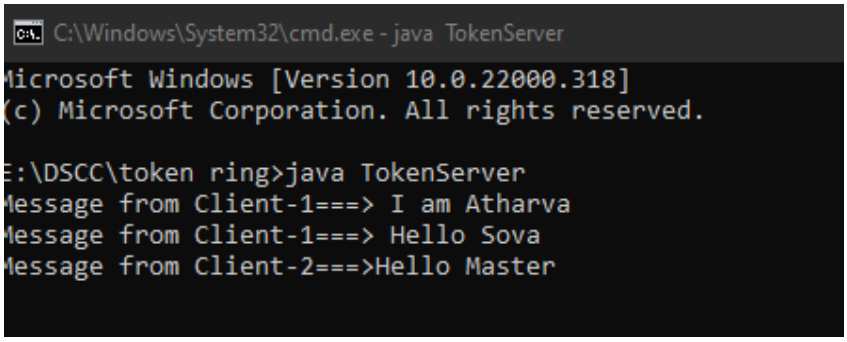
```
        while(true){
            if(hasToken){
                System.out.println("Do you want to enter
data...(yes/no):");
                br=new BufferedReader(new InputStreamReader(System.in));
                String ans=br.readLine();
                if(ans.equalsIgnoreCase("yes")){
                    System.out.println("ready to send");
                    System.out.println("sending");
                    System.out.println("Enterthedata");
                    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("nowsending");
                }else if(ans.equalsIgnoreCase("no")){
                    System.out.println("I am busy state");
                    byte bf1[]=new byte[1024];
                    bf1=msg.getBytes();
                    ds.send(new
DatagramPacket(bf1,bf1.length,InetAddress.getLocalHost(),200));
                    hasToken=false;
                    ds.receive(dp=new DatagramPacket(bf2,bf2.length));
                    String clientmsg=new
String(dp.getData(),0,dp.getLength());
                    System.out.println("Thedatais"+clientmsg);
                    if(clientmsg.equals("Token")) hasToken = true;
                    System.out.println("I am leaving busy state");
                }else{
                    System.out.println("Entering in receive mode.");
                    byte bf[]=new byte[1024];
                    ds.receive(dp=new DatagramPacket(bf,bf.length));
                    String clientmsg1=new
String(dp.getData(),0,dp.getLength());
                    System.out.println("The data is "+clientmsg1);
                    if(clientmsg1.equals("Token")){
                        hasToken = true;
                    }
                }
            }
        }
    }
}
```

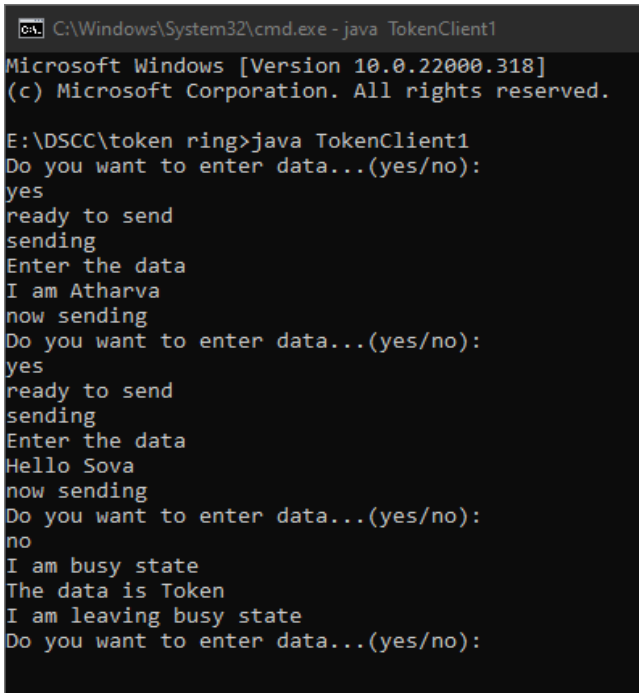
TokenClient2.java

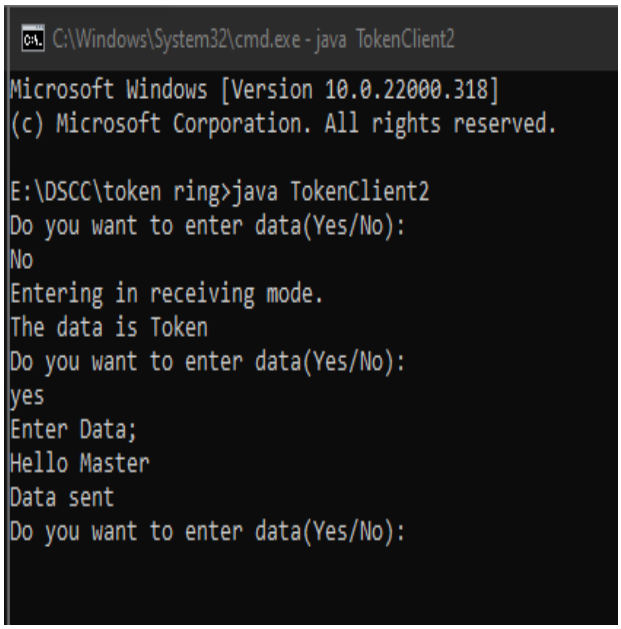
```
import java.net.*;
import java.io.*;
class TokenClient2 {
    static DatagramSocket ds;
    static DatagramPacket dp;
    static BufferedReader br;

    public static void main(String[] args) throws Exception {
        try {
            ds = new DatagramSocket(200);
        } catch (Exception e) {
            e.printStackTrace();
        }
        boolean hasToken = true;
        while (true) {
            // System.out.println("Enteringif");
            if (hasToken == true) {
                System.out.println("Do you want to enter data(Yes/No):");
                br = new BufferedReader(new InputStreamReader(System.in));
                String str = br.readLine();
                if (str.equalsIgnoreCase("yes")) {
                    System.out.println("EnterData;");
                    br = new BufferedReader(new
InputStreamReader(System.in));
                    String msg = "Client-2===>" + br.readLine();
                    byte bf1[] = newbyte[1024];
                    bf1 = msg.getBytes();
                    ds.send(new DatagramPacket(bf1, bf1.length,
InetAddress.getLocalHost(), 1000));
                    System.out.println("Datasent");
                } else {
                    // sendtoclient1.
                    String clientmsg = "Token";
                    byte bf2[] = new byte[1024];
                    bf2 = clientmsg.getBytes();
                    ds.send(new DatagramPacket(bf2, bf2.length,
InetAddress.getLocalHost(), 100));
                    hasToken = false;
                }
            } else {
                try {
                    byte buff[] = new byte[1024];
                    System.out.println("Entering in receiving mode.");
```

```
                    ds.receive(dp = new DatagramPacket(buff,
buff.length));
                    String clientmsg1 = new String(dp.getData(), 0,
dp.getLength());
                    System.out.println("The data is " + clientmsg1);
                    if (clientmsg1.equals("Token"))
                        hasToken = true;
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        }
    }
}
```

Output :



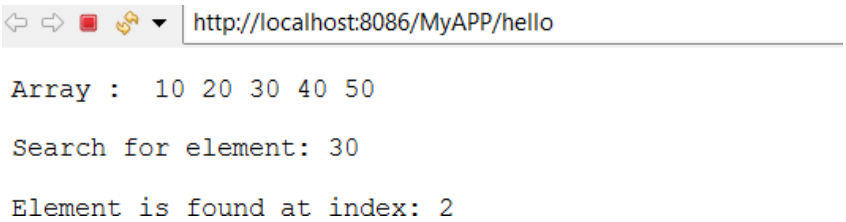
**Practical No.10**

**Aim:** To develop application using Google App Engine by using Eclipse IDEFind the position of a target value within a sorted integer array. (Binary Search)

```
import java.io.IOException;
import java.util.Arrays;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet(name="HelloAppEngine",urlPatterns={"/hello"})
public class HelloAppEngine extends HttpServlet {
    @Override
    public void doGet(HttpServletRequest request, HttpServletResponse
response) throws IOException {
        response.setContentType("text/plain");
        response.setCharacterEncoding("UTF-8");
        i ntarr[] = { 10, 20, 30, 40, 50 };
        int key = 30;
        response.getWriter().print("Array : ");
        for (int i = 0; i < 5; i++) {
            response.getWriter().print("" + arr[i]);
        }
        response.getWriter().println("\n\nSearch for element: " + key +
"\n");
        int last = arr.length - 1;
        int first = 0;
        int mid = (first + last) / 2;
        while (first <= last) {
            if (arr[mid] < key) {
                first = mid + 1;
            } else if (arr[mid] == key) {
                response.getWriter().print("Element is found at index:" +
mid + "\n");
                break;
            } else {
                last = mid - 1;
            }
            mid = (first + last) / 2;
        }
        if (first > last) {
            response.getWriter().print("Element is not found!");
        }
    }
```

```
}
```

Output :


http://localhost:8086/MyAPP/hello
Array :  10 20 30 40 50
Search for element: 30
Element is found at index: 2