**DISTRIBUTED COMPONENTS LAB**

**SEMESTER LAB EXAM**

**Aim:**

To implement socket programming for multiple clients and a server with java and to create an RMI Application for seeing Log details.

**Question 1**

**Initialize a communication between multiple clients and a server using Java socket**

**Programming**

**Algorithm:**

**Step 1:** Start

**Step 2:** In Server side, socket is created and a thread is created for each client which is connected to the server with the same port number.

**Step 3:** In Client side, socket is created and it is connected with the same port number as that of server.

**Step 4:** Multiple clients are created and all are connected with the same port number.

**Step 5:** The server responds to the client from where the message was originated.

**Step 6:** Stop

**CODE:**

**SERVER.JAVA**

import java.net.\*;

import java.io.\*;

import java.util.\*;

public class Server

{

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

  {

    try

    {

      Scanner sc = new Scanner(System.in);

      System.out.println("Enter port number: ");

      int PORT = sc.nextInt();

      ServerSocket server=new ServerSocket(PORT);

      int clientCount=0;

      System.out.println("Server Started at port number " + PORT);

      while(true)

      {

        clientCount++;

        Socket serverClient=server.accept();

        System.out.println("Client No:" + clientCount + " started!");

        ServerClientThread sct = new ServerClientThread(serverClient,clientCount);

        sct.start();

      }

    }

    catch(Exception e)

    {

      System.out.println(e);

    }

  }

}

**SERVERCLIENTTHREAD.JAVA**

import java.net.\*;

import java.io.\*;

import java.util.\*;

class ServerClientThread extends Thread

{

    Socket serverClient;

    int clientNo;

    int factorial;

    ServerClientThread(Socket inSocket,int counter)

    {

      serverClient = inSocket;

      clientNo=counter;

    }

    public void run()

    {

      try

      {

        DataInputStream inStream = new DataInputStream(serverClient.getInputStream());

        DataOutputStream outStream = new DataOutputStream(serverClient.getOutputStream());

        String clientMessage="", serverMessage="";

        while(!clientMessage.equals("bye")) {

          clientMessage=inStream.readUTF();

          System.out.println("From Client: " +clientNo+ ": Message is :"+ clientMessage);

          System.out.println("Enter a text to respond");

          Scanner sc = new Scanner(System.in);

          String message = sc.next();

          serverMessage="From Server to Client: " + clientNo + " " + message;

          outStream.writeUTF(serverMessage);

          outStream.flush();

        }

        inStream.close();

        outStream.close();

        serverClient.close();

      }

      catch(Exception ex)

      {

        System.out.println(ex);

      }

      finally

      {

        System.out.println("Client -" + clientNo + " exit!! ");

      }

    }

  }

**CLIENT.JAVA**

import java.net.\*;

import java.io.\*;

import java.util.\*;

public class Client {

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

  {

    try

    {

      Scanner sc = new Scanner(System.in);

      System.out.println("Enter port number: ");

      int PORT = sc.nextInt();

      Socket socket=new Socket("127.0.0.1",PORT);

      DataInputStream inStream=new DataInputStream(socket.getInputStream());

      DataOutputStream outStream=new DataOutputStream(socket.getOutputStream());

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

      String clientMessage="",serverMessage="";

      while(!clientMessage.equals("bye"))

      {

        System.out.println("Enter a message :");

        clientMessage=br.readLine();

        outStream.writeUTF(clientMessage);

        outStream.flush();

        serverMessage=inStream.readUTF();

        System.out.println(serverMessage);

      }

      outStream.close();

      outStream.close();

      socket.close();

    }

    catch(Exception e)

    {

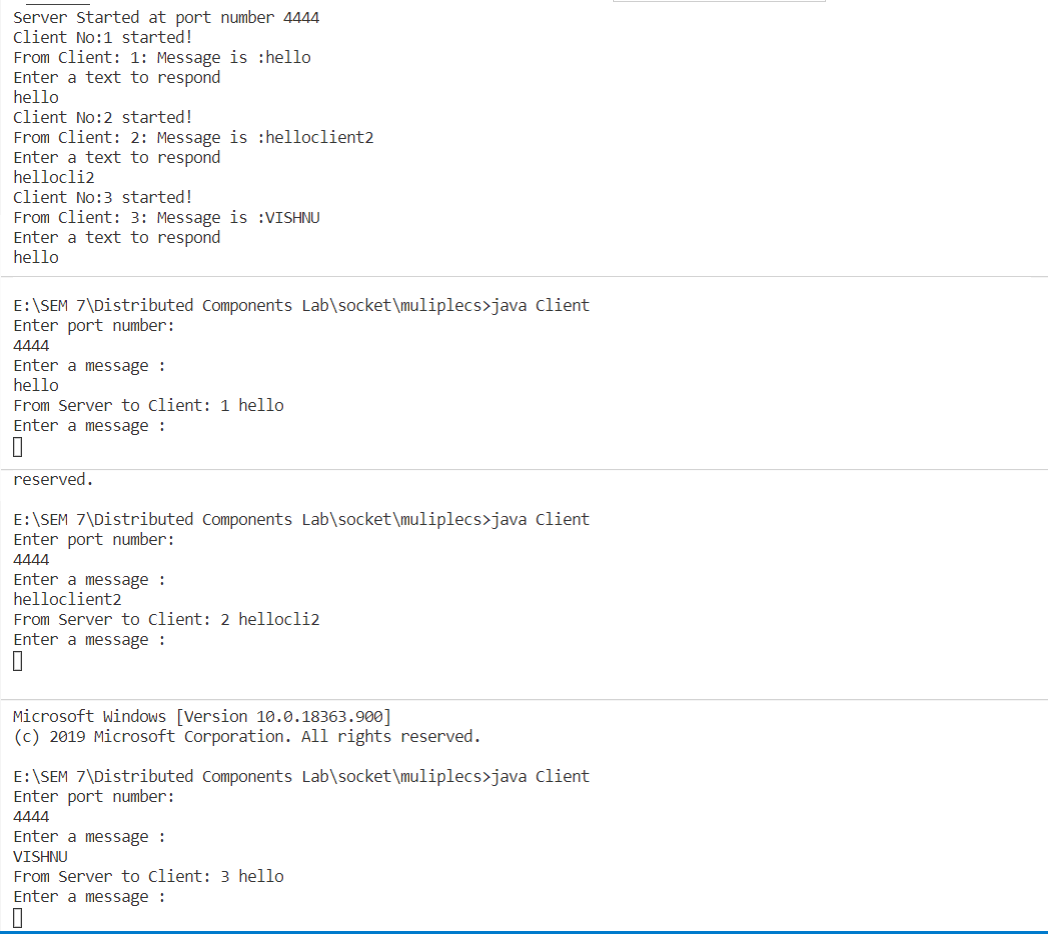
      System.out.println(e);

    }

  }

}

**Output:**



**QUESTION 2:**

**Develop a distributed application using RMI which implements the Log in operation.**

** Client passes the name to the server.**

** The remote server checks whether the name is already in a DATABASE or not. If found**

**then send login time and date to the client along with the “welcome” message else send**

**an error message.**

** All the valid users can search and view the details of other users by giving the name**

**ALGORITHM:**

**Step 1:** Start

**Step 2:** RMI\_Interface contains the functions that are to be implemented for the RMI Application.

**Step 3:** RMI\_Impl contains the definitions of the functions for the RMI Application.

**Step 4:** RMI\_Server creates a RMI, with RMI\_INSTANCE and starts listening.

**Step 5:** RMI\_Client creates an instance for the RMI\_INSTANCE server and calls the methods of RMI\_Server using that instance.

**Step 6:** Stop

**CODE:**

**RMI\_INTERFACE**

public interface RMI\_Interface extends java.rmi.Remote

{

    public String view(String s) throws java.rmi.RemoteException;

}

**RMI\_IMPL**

import java.rmi.\*;

import java.time.\*;

import java.util.\*;

import java.rmi.server.UnicastRemoteObject;

public class RMI\_Impl extends UnicastRemoteObject implements RMI\_Interface

{

    private static final long serialVersionUID = 1L;

    ArrayList<String> database = new ArrayList<String>();

    public RMI\_Impl() throws RemoteException

    {

        database.add("vishnu");

        database.add("vardhan");

        database.add("rag");

        database.add("raghul");

    }

    public String view(String name) throws RemoteException

    {

        if(database.contains(name))

        {

            LocalDate date = LocalDate.now();

            LocalTime time = LocalTime.now();

            String result = date.toString() + ";" + time.toString() + ";" + "Welcome!";

            return result;

        }

        else

        {

            return "Name not found!";

        }

    }

}

**RMI\_SERVER**

import java.rmi.\*;

public class RMI\_Server {

    public static void main(String args[])

    {

        try

        {

            RMI\_Impl implclass=new RMI\_Impl();

            Naming.rebind("RMI\_INSTANCE",implclass);

        }

        catch (Exception e)

        {

            System.out.println("Exception occured:" + e);

        }

    }

}

**RMI\_CLIENT**

import java.rmi.\*;

import java.util.\*;

import java.awt.\*;

import javax.swing.\*;

import java.awt.event.\*;

import javax.swing.JFrame;

import javax.swing.border.Border;

public class RMI\_Client

{

    public static void main(String[] argv)

    {

        if (argv.length !=1)

        {

            System.out.println("Usage: java RMI\_Client &lt;IP addr of host running RMI Server");

            System.exit(0);

        }

        String serverName = argv[0];

        try

        {

            RMI\_Interface myobj = (RMI\_Interface)Naming.lookup("rmi://"+serverName+"/RMI\_INSTANCE");

            Font myFont = new Font("Times New Roman",Font.BOLD,20);

            Border border = BorderFactory.createLineBorder(new Color(255, 77, 77), 2);

            JFrame jf\_s = new JFrame();

            jf\_s.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

            jf\_s.setExtendedState(JFrame.MAXIMIZED\_BOTH);

            Label center = new Label("LOG FILE MANAGEMENT");

            Label l1 = new Label("User name");

            JTextField jt1 = new JTextField("");

            JButton search\_button = new JButton("SEARCH");

            center.setFont(myFont);

            l1.setFont(myFont);

            jt1.setFont(myFont);

            search\_button.setFont(myFont);

            center.setBounds(700,120,350,40);

            l1.setBounds(650,200,180,40);

            jt1.setBounds(850,200,220,40);

            search\_button.setBounds(700,280,220,40);

            search\_button.addActionListener(new ActionListener()

            {

                public void actionPerformed(ActionEvent e)

                {

                    try

                    {

                        String result = myobj.view(jt1.getText());

                        String unpack[] = result.split(";");

                        if(unpack.length==1)

                        {

                            JOptionPane.showMessageDialog(null,"User not found!","Error",JOptionPane.ERROR\_MESSAGE);

                        }

                        Label l1 = new Label("Login date: " + unpack[0]);

                        Label l2 = new Label("Login time: " + unpack[1]);

                        Label l3 = new Label("Message: " + unpack[2]);

                        l1.setFont(myFont);

                        l2.setFont(myFont);

                        l3.setFont(myFont);

                        l1.setBounds(650,360,220,40);

                        l2.setBounds(650,400,220,40);

                        l3.setBounds(650,440,220,40);

                        jf\_s.add(l1);

                        jf\_s.add(l2);

                        jf\_s.add(l3);

                    }

                    catch(Exception d2)

                    {

                        System.out.println(d2);

                    }

                }

            });

            jf\_s.add(center);

            jf\_s.add(l1);

            jf\_s.add(jt1);

            jf\_s.add(search\_button);

            jf\_s.setLayout(null);

            jf\_s.setVisible(true);

        }

        catch(Exception e)

        {

            System.out.println("Exception Occured " + e);

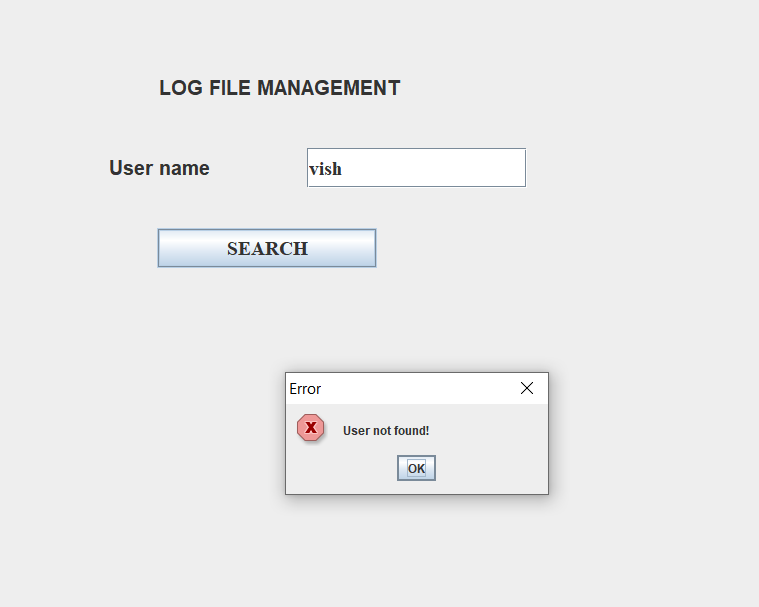
            System.exit(0);

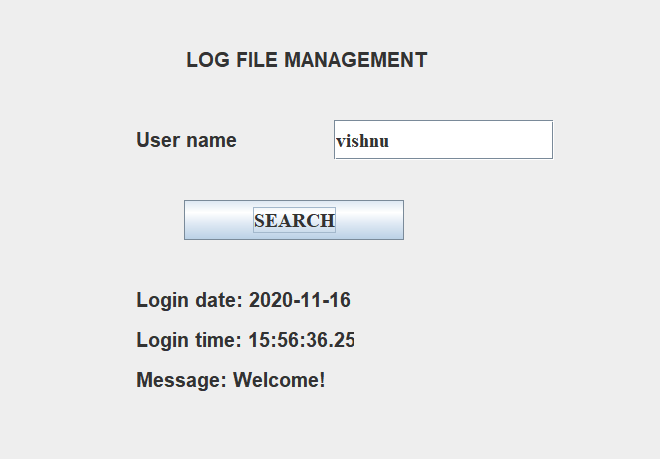
        }

    }

}

**OUTPUT:**

****

****

**RESULT:**

The given applications are created using Java Socket programming and Java RMI successfully.