

Deccan Education Society's
Navinchandra Mehta Institute of Technology
and Development

CERTIFICATE

This is to certify that Ms. **Sayali Suresh Patil** of M.C.A. Semester III with Roll No. **C22096** has completed **All** practicals of MCAL32 **Distributed Systems and Cloud Computing Lab** under my supervision in this college during the year 2022-2023.

CO	R1 (Attendance)	R2 (Performance during lab session)	R3 (Innovation in problem solving technique)	R4 (Mock Viva)	R5 (Variation in implementation of learnt topics on projects)
CO1					
CO2					
CO3					
CO4					

Practical-in-charge

Head of Department
MCA Department
(NMITD)

INDEX

Sr. No	Contents	Date	Sign
1.	INTRODUCTION TO JAVA SOCKET PROGRAMMING		
a)	Using TELNET Client to Communicate with NIST Time	05/09/2023	
b)	Using Socket API to Implement Java Client to Communicate with NIST Time of the Day Service	05/09/2023	
c)	Implementing A Java Program to Convert Between Domain Name and IP Address	05/09/2023	
2.	IMPLEMENTING CLIENT-SERVER COMMUNICATION		
a)	Implementing A Simple Echo Chat Server Using Java Server Socket API	06/09/2023	
b)	Making the Echo Chat Server Multi-Threaded to Support Multiple Simultaneous Chatting Sessions	06/09/2023	
3.	INTRODUCTION TO DATAGRAM SOCKET API		
a)	Using Datagram Socket API to Implement Number Addition Service	13/09/2023	
4.	IMPLEMENTING TOKEN-RING PROTOCOL FOR DISTRIBUTED MUTUAL-EXCLUSION		
a)	Using Datagram Socket API to Implement Token Ring Protocol to Enforce Distributed Mutual Exclusion	21/09/2023	
5.	Implement a simple client server chat application using Socket API	27/09/2023	
6.	To develop a program for multi-client chat server.	04/10/2023	
7.	To implement a Server calculator using RPC concept. (Make use of datagram)	27/10/23	
8.	To implement a Date Time Server using RPC concept.	27/10/23	
9.	Using MySQL create Library database. Create table Book (Book_id, Book_name, Book_author) and retrieve the Book information from Library database using Remote Object Communication concept.	21/11/23	
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 concept.	28/11/23	
11.	To develop applications using Google App Engine by using Eclipse IDE.	13/12/23	

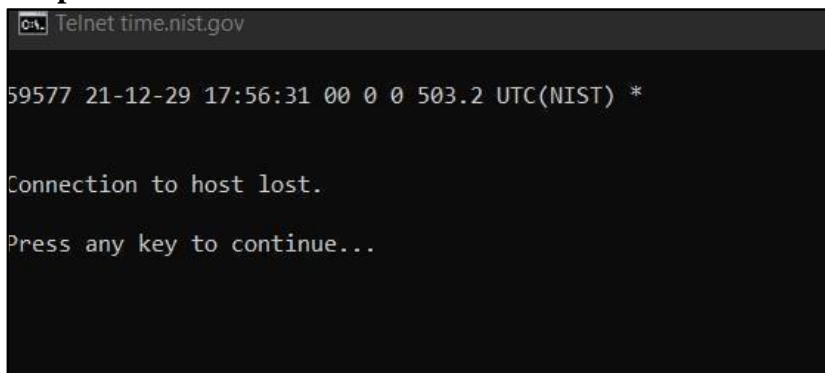
1. INTRODUCTION TO JAVA SOCKET PROGRAMMING

a) Using TELNET Client to Communicate with NIST Time

Code:

```
import java.io.IOException;
import java.net.Socket;
import java.util.Scanner;
public class hellosocketprogramming_q1 {
    public static void main(String args[]) throws IOException {
        Socket s = new Socket ("time.nist.gov", 13);
        Scanner ins = new Scanner (s.getInputStream());
        while (ins.hasNextLine()) {
            System.out.println (ins.nextLine());
        }
    }
}
```

Output:



```
Telnet time.nist.gov

59577 21-12-29 17:56:31 00 0 0 503.2 UTC(NIST) *

Connection to host lost.

Press any key to continue...
```

b) Using Socket API to Implement Java Client to Communicate with NIST Time of the Day Service Code:

```
import java.io.IOException;
import java.net.Socket;
import java.util.Scanner;
public class hellosocketprogramming_q1 {
    public static void main(String args[]) throws IOException {
        Socket s = new Socket ("time.nist.gov", 13);
        Scanner ins = new Scanner (s.getInputStream());
        while (ins.hasNextLine()) {
            System.out.println (ins.nextLine());
        }
    }
}
```

Output:

```
-----<< com.mycompany:socketproj1 >-----
[ Building socketproj1 1.0-SNAPSHOT
-----[ jar ]-----

--- exec-maven-plugin:3.0.0:exec (default-cli) @ socketproj1 ---
59577 21-12-29 17:56:00 00 0 0 784.8 UTC(NIST) *

BUILD SUCCESS

Total time: 2.130 s
Finished at: 2021-12-29T23:25:59+05:30
-----
```

c) Implementing a Java Program to Convert Between Domain Name and IP Address.

Code:

```
import java.net.InetAddress;
import java.io.IOException;
import java.net.UnknownHostException;
import java.util.Scanner;
public class socketprogram_q3 {
    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) throws UnknownHostException {
        if (args.length > 0){
            String domainname = args[0];
            InetAddress[] inet = InetAddress.getAllByName(domainname);
            for(InetAddress addr : inet){
                System.out.println(addr);
            }
        }else{
            System.out.println(InetAddress.getLocalHost());
        }
        // TODO code application logic here
    }
}
```

Output:

```
-----< com.mycompany:socketproj1 >-----
[ Building socketproj1 1.0-SNAPSHOT
-----[ jar ]-----

--- exec-maven-plugin:3.0.0:exec (default-cli) @ socketproj1 ---
DESKTOP-MUK2EVO/192.168.49.1

BUILD SUCCESS

Total time: 1.252 s
Finished at: 2021-12-29T23:32:34+05:30
-----
```

2. IMPLEMENTING CLIENT-SERVER COMMUNICATION

a) Implementing a Simple Echo Chat Server Using Java Server Socket API

Code:

```
import java.io.*;
import java.net.*;
import java.util.*;
public class EchoServer {
    public static void main(String args[]) throws IOException {
        ServerSocket ss = new ServerSocket (8189);
        System.out.println("I am about to listen on 8189");
        Socket conn = ss.accept();
        InputStream inS = conn.getInputStream();
        OutputStream outS = conn.getOutputStream();
        Scanner in = new Scanner(inS);
        PrintWriter out = new PrintWriter(outS, true);
        out.println("Hello!.. I am the chat server. Let's Chat"+" Say BYE to
disconnect");
        boolean bye = false;
        while(!bye && in.hasNextLine()){
            String cMsg = in.nextLine();
            out.println("Echo: " +cMsg);
            if (cMsg.trim().equals("BYE")) { bye= true;}
        }
        conn.close();
        ss.close();
    }
}
```

Output:

```
-----< com.mycompany:socketproj1 >-----
Building socketproj1 1.0-SNAPSHOT
-----[ jar ]-----

--- exec-maven-plugin:3.0.0:exec (default-cli) @ socketproj1 ---
I am about to listen on 8189
```

```
C:\> Telnet localhost
Hello!.. I am the chat server. Let's Chat Say BYE to disconnect
hello
Echo: hello
how are you?
Echo: how are you?
BYE
Echo: BYE

Connection to host lost.
Press any key to continue...
```

b) Making the Echo Chat Server Multi-Threaded to Support Multiple Simultaneous Chatting Sessions
Code:

```
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
public class MultiUserEchoChatServer {
    public static void main(String[] args) throws IOException{
        ServerSocket server = new ServerSocket (8189);
        System.out.println("I am about to listen on 8189");
        int userCnt =0;
        while (true){
            Socket userSocket = server.accept();
            userCnt++;
            EchoChatHandler echoChatter = new EchoChatHandler(userSocket,
userCnt);
            Thread userChatThread = new Thread(echoChatter);
            System.out.println("Spawing a new chatting thread for user " +
userCnt);
            userChatThread.start();
        }
    }
}
```

Code (Handler Class):

```
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.io.PrintWriter;
import java.net.Socket;
import java.util.Scanner;
class EchoChatHandler implements Runnable {
    private final Socket userSocket;
    private int userId;
    public EchoChatHandler(Socket userSocket, int userCnt){
        this.userSocket = userSocket;
        this.userId = userId;
    }
    @Override
    public void run(){
        try{
            InputStream inS = userSocket.getInputStream();
            OutputStream outS = userSocket.getOutputStream();
            Scanner in = new Scanner(inS);
            PrintWriter out = new PrintWriter(outS, true);
            out.println("Hello user!" + userId + "I am the chat server. Let's
```

```
Chat"+" Say BYE to disconnect");  
boolean bye = false;  
while(!bye && in.hasNextLine()){
```



```
        String cMsg = in.nextLine();
        out.println("Echo: " +cMsg);
        if (cMsg.trim().equals("BYE")) { bye= true;}
    }
    userSocket.close();
}
catch(IOException ioEX){ ioEX.printStackTrace();}
}
```

Output:

<pre>Telnet localhost Hello user!OI am the chat server. Let's Chat Say BYE to disconnect heelo cli Echo: heelo cli client 1 Echo: client 1</pre>	<pre>Telnet localhost Hello user!OI am the chat server. Let's Chat Say BYE to disconnect client 2 Echo: client 2</pre>
--	--

```
-----< com.mycompany:socketproj1 >-----
Building socketproj1 1.0-SNAPSHOT
-----[ jar ]-----

--- exec-maven-plugin:3.0.0:exec (default-cli) @ socketproj1 ---
I am about to listen on 8189
Spawing a new chatting thread for user 1
Spawing a new chatting thread for user 2
```

3. INTRODUCTION TO DATAGRAM SOCKET API

a) Using Datagram Socket API to Implement Number AdditionService

Code (Client):

```
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.io.PrintWriter;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.ServerSocket;
import java.net.Socket;
import java.net.SocketException;
import java.net.UnknownHostException;
import java.util.Scanner;
public class AdditionClient {
    public static void main(String[] args){
        try{
            InetAddress addServiceIP = InetAddress.getLocalHost();
            int addServicePort = 8189;
            DatagramSocket clientSocket = new DatagramSocket();
            Scanner sc = new Scanner(System.in);
            System.out.println("Please enter the list of number: ");
            String numberList = sc.nextLine();
            DatagramPacket outDP = new DatagramPacket(numberList.getBytes
            (),numberList.length(),addServiceIP, addServicePort);
            clientSocket.send(outDP);
            System.out.println("Adding the number " + numberList + "
            together");
            byte[] buffer = new byte[256];
            DatagramPacket inDP = new DatagramPacket(buffer, buffer.length);
            clientSocket.receive(inDP);
            String servResp = new String(inDP.getData(),0,inDP.getLength());
            System.out.println(servResp);
            String stop = "STOP";
            outDP = new DatagramPacket(stop.getBytes(),stop.length
            (),addServiceIP, addServicePort);
            clientSocket.send(outDP);
            clientSocket.close();
        }
        catch(IOException ioEX){
            System.out.println(ioEX);
        }
    }
}
```

```
}  
}
```

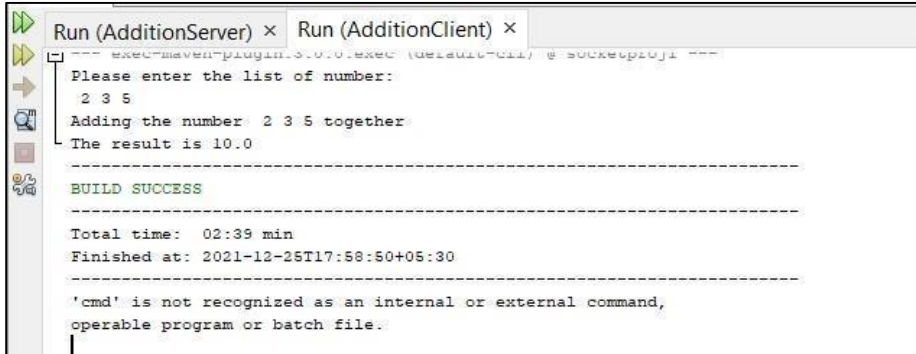
Code (Server):

```
import java.io.IOException;  
import java.net.DatagramPacket;  
import java.net.DatagramSocket;  
import java.net.InetAddress;  
import java.net.SocketException;  
import java.util.StringTokenizer;  
public class AdditionServer {  
    private int port = 8189;  
    public static void main(String[] args){  
        Datagramadditionservice addService = new Datagramadditionservice();  
        addService.start();  
    }  
    public void start(){  
        try{  
            DatagramSocket serverConn = new DatagramSocket(port);  
            byte[] buffer = new byte[256];  
            DatagramPacket inDP = new DatagramPacket (buffer, buffer.length);  
            String clientReq, serverResp;  
            do{  
                serverConn.receive(inDP);  
                InetAddress clientAddress = inDP.getAddress();  
                int clientPort = inDP.getPort();  
                clientReq = new String(inDP.getData(), 0, inDP.getLength());  
                if(clientReq != null && !clientReq.trim().equals("STOP")){  
                    double sumResult = 0;  
                    StringTokenizer st = new StringTokenizer(clientReq);  
                    try{  
                        while(st.hasMoreTokens()){  
                            Double d = new Double(st.nextToken());  
                            sumResult += d.doubleValue();  
                        }  
                        serverResp = "The result is " + sumResult;  
                    }  
                    catch(NumberFormatException nEx){  
                        serverResp = "Sorry, your list contains an invalid number";  
                    }  
                    DatagramPacket outDP = new  
DatagramPacket(serverResp.getBytes(),serverResp.length(),  
clientAddress, clientPort);  
serverConn.send(outDP);  
                }  
            }while(true);  
        }  
        catch(SocketException ex){  
            System.out.println("SocketException: " + ex);  
        }  
    }  
}
```

} }

```
        while(clientReq.trim().equals("STOP"));  
        serverConn.close();  
    } catch(IOException ioex){ System.out.println(ioex);}  
    }  
}
```

Output:



```
Run (AdditionServer) × Run (AdditionClient) ×  
--- exec-maven-plugin:3.0.0:exec (default-cli) @ socketproj1 ---  
Please enter the list of number:  
2 3 5  
Adding the number 2 3 5 together  
The result is 10.0  
-----  
BUILD SUCCESS  
-----  
Total time: 02:39 min  
Finished at: 2021-12-25T17:58:50+05:30  
-----  
'cmd' is not recognized as an internal or external command,  
operable program or batch file.
```

4. IMPLEMENTING TOKEN-RING PROTOCOL FOR DISTRIBUTED MUTUAL-EXCLUSION

a) Using Datagram Socket API to Implement Token Ring Protocol to Enforce Distributed Mutual Exclusion

Code (Server):

```
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.SocketException;
public class TokenChatServer {
    private static DatagramSocket ds;
    private static DatagramPacket dp;
    public static void main(String[] args) throws IOException{
        try{
            ds = new DatagramSocket(1000);
        } catch(SocketException ex){ ex.printStackTrace();}
        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);
        }
    }
}
```

Code (Client1):

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.SocketException;
import java.util.logging.Level;
import java.util.logging.Logger;
public class TokenClient1 {
    private static DatagramSocket ds;
    private static DatagramPacket dp;
    private static BufferedReader br;
    private static int cp=100;
    public static void main(String[] args) throws IOException{
        boolean hasToken = true;
        try{
            ds= new DatagramSocket(cp);
        }
        catch(SocketException ex){
            Logger.getLogger(TokenClient1.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}
```



```

        throw ex;
    }
    while(true){
        if(hasToken){
            System.out.println("Do you want to say Something"+"(i.e.,Send Data)to
Server?:"+"Type Y for Yes/N for No");
            br=new BufferedReader(new InputStreamReader(System.in));
            String userResp= br.readLine();
            if(userResp.equalsIgnoreCase("Y")){
                System.out.println("Enter what you want to send:");
                String userData="Client 1===>> "+br.readLine();
                System.out.println("Getting ready to send data ...");
                byte buff[]=userData.getBytes();
                System.out.println("Sending...");
                ds.send(new DatagramPacket(buff,buff.length,
InetAddress.getLocalHost(),1000));
                System.out.println("Data Sent.");
            }else{
                System.out.println("Since I am in "+"busy state ... passing token to
client 2.");

                String tokenMsg = "Token";
                byte[] bf1 = new byte[1024];
                bf1 = tokenMsg.getBytes();
                ds.send(new
DatagramPacket(bf1,bf1.length,InetAddress.getLocalHost(),200));
                hasToken= false;
            }
        }else{
            System.out.println("Entering in the receiving mode ...");
            byte bf[]=new byte[1024];
            ds.receive(dp = new DatagramPacket(bf,bf.length));
            String msgClient3= new String(dp.getData(),0,dp.getLength());
            System.out.println("The data received from left neighbor :"+"client 3
is"+msgClient3);

            if(msgClient3.equalsIgnoreCase("Token")){
                hasToken = true;
            }
        }
    }
}

```

Code (Client2):

```

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

```



```
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.SocketException;
import java.util.logging.Level;
import java.util.logging.Logger;
public class TokenClient2 {
    private static DatagramSocket ds;
    private static DatagramPacket dp;
    private static BufferedReader br;
    private static int cp = 200;
    public static void main(String[] args) throws IOException{
        boolean hasToken = false;
        try{
            ds = new DatagramSocket(cp);
        }
        catch(SocketException ex){
            Logger.getLogger(TokenClient2.class.getName()).log(Level.SEVERE, null,ex);
            throw ex;
        }
        while(true){
            if(hasToken){
                System.out.println("Do you want to say something, " + "(i.e. Send Data) To
Server?: " + "Type Y for Yes/N for No");
                br = new BufferedReader(new InputStreamReader(System.in));
                String userResp = br.readLine();
                if(userResp.equalsIgnoreCase("Y")){
                    System.out.println("Enter what you want to send: ");
                    String userData = "Client 2 ==> "+br.readLine();
                    System.out.println("Getting ready to send data ...");
                    byte buff[] = userData.getBytes();
                    System.out.println("Something...");
                    ds.send(new DatagramPacket (buff,buff.length,
InetAddress.getLocalHost(),1000));
                    System.out.println("Data Sent.");
                }else{
                    System.out.println("Since I am in "+ "busy state ... passing token to
client 3.");
                    String tokenMsg = "Token";
                    byte[] bf1 = new byte[1024];
                    bf1 = tokenMsg.getBytes();
                    ds.send(new
DatagramPacket(bf1,bf1.length,InetAddress.getLocalHost(),300));
                    hasToken= false;
                }
            }
        }
    }
}
```



```
        }else{
            System.out.println("Entering in the receiving mode ...");
            byte bf[]=new byte[1024];
            ds.receive(dp = new DatagramPacket(bf,bf.length));
            String msgClient3= new String(dp.getData(),0,dp.getLength());
            System.out.println("The data received from left neighbor :"+"client 1
is"+msgClient3);
            if(msgClient3.equalsIgnoreCase("Token")){
                hasToken = true;
            }
        }
    }
}
```

Code (Client3):

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.SocketException;
import java.util.logging.Level;
import java.util.logging.Logger;
public class TokenClient3 {
    private static DatagramSocket ds;
    private static DatagramPacket dp;
    private static BufferedReader br;
    private static int cp = 300;
    public static void main(String[] args) throws IOException{
        boolean hasToken = false;
        try{
            ds = new DatagramSocket(cp);
        }
        catch(SocketException ex){
            Logger.getLogger(TokenClient3.class.getName()).log(Level.SEVERE, null,ex);
            throw ex;
        }
        while(true){
            if(hasToken){
                System.out.println("Do you want to say something, " + "(i.e. Send Data) To
Server?: " + "Type Y for Yes/N for No");
                br = new BufferedReader(new InputStreamReader(System.in));
                String userResp = br.readLine();
```



```
        if(userResp.equalsIgnoreCase("Y")){
            System.out.println("Enter what you want to send: ");
            String userData = "Client 3 ==> "+br.readLine();
            System.out.println("Getting ready to send data ...");
            byte buff[] = userData.getBytes();
            System.out.println("Something...");
            ds.send(new DatagramPacket (buff,buff.length,
InetAddress.getLocalHost(),1000));
            System.out.println("Data Sent.");
        }else{
            System.out.println("Since I am in "+ "busy state ... passing token to
client 2.");

            String tokenMsg = "Token";
            byte[] bf1 = new byte[1024];
            bf1 = tokenMsg.getBytes();
            ds.send(new
DatagramPacket(bf1,bf1.length,InetAddress.getLocalHost(),100));
            hasToken= false;
        }
    }else{
        System.out.println("Entering in the receiving mode ...");
        byte bf[]=new byte[1024];
        ds.receive(dp = new DatagramPacket(bf,bf.length));
        String msgClient3= new String(dp.getData(),0,dp.getLength());
        System.out.println("The data received from left neighbor :"+"client 2
is"+msgClient3);

        if(msgClient3.equalsIgnoreCase("Token"))
        {
            hasToken = true;
        }
    }
}
}
```

Output:

```
Scanning for projects...
```

```
-----< com.mycompany:socketproj1 >-----  
Building socketproj1 1.0-SNAPSHOT  
-----[ jar ]-----  
  
--- exec-maven-plugin:3.0.0:exec (default-cli) @ socketproj1 ---  
Message From: Client 1==>> Hello  
Message From: Client 2 ==>> Hi
```

```
Do you want to say Something(i.e.,Send Data)to Server?:Type Y for Yes/N for No  
Y  
Enter what you want to send:  
Hello  
Getting ready to send data ...  
Sending...  
Data Sent.  
Do you want to say Something(i.e.,Send Data)to Server?:Type Y for Yes/N for No  
N  
Since I am in busy state ... passing token to client 2.  
Entering in the receiving mode ...  
The data received from left neighbor :client 3 isToken  
Do you want to say Something(i.e.,Send Data)to Server?:Type Y for Yes/N for No
```

```
--- exec-maven-plugin:3.0.0:exec (default-cli) @ socketproj1 ---  
Entering in the receiving mode ...  
The data received from left neighbor :client 1 isToken  
Do you want to say something, (i.e. Send Data) To Server?: Type Y for Yes/N for No  
Y  
Enter what you want to send:  
Hi  
Getting ready to send data ...  
Something...  
Data Sent.  
Do you want to say something, (i.e. Send Data) To Server?: Type Y for Yes/N for No  
N  
Since I am in busy state ... passing token to client 3.  
Entering in the receiving mode ...
```

```
Running NetBeans Compile On Save execution. Phase execution is skipped and output directories d  
Scanning for projects...  
  
-----< com.mycompany:socketproj1 >-----  
Building socketproj1 1.0-SNAPSHOT  
-----[ jar ]-----  
  
--- exec-maven-plugin:3.0.0:exec (default-cli) @ socketproj1 ---  
Entering in the receiving mode ...  
The data received from left neighbor :client 2 isToken  
Do you want to say something, (i.e. Send Data) To Server?: Type Y for Yes/N for No  
Heya  
Since I am in busy state ... passing token to client 2.  
Entering in the receiving mode ...
```

5. Implement a simple client server chat application using Socket API

Code (Client):

```
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import static java.lang.System.out;
import java.net.Socket;
import java.net.UnknownHostException;

public class Client_Q1 {
    private Socket socket=null;
    private DataInputStream input =null;
    private DataOutputStream output =null;
    public Client_Q1(String address, int port) throws IOException{
        try{
            socket=new Socket(address, port);
            System.out.println("Connected");
            input=new DataInputStream(System.in);
            output=new DataOutputStream(socket.getOutputStream());
        }
        catch(UnknownHostException u){
            System.out.println(u);
        }
        catch(IOException i){
            System.out.println(i);
        }
        String line="";
        while(!line.equals("Over")){
            try{
                line=input.readLine();
                output.writeUTF(line);
            }
            catch(IOException i){
                System.out.println(i);
            }
        }
        try{
            input.close();
            output.close();
            socket.close();
        }
        catch(IOException i){
            System.out.println(i);
        }
    }
}
```



```
        public static void main(String args[]) throws IOException{
            Client_Q1 client=new Client_Q1("127.0.0.1",5000);
        }
    }
```

Code (Server):

```
import com.sun.corba.se.spi.activation.Server;
import java.io.BufferedReader;
import java.io.DataInputStream;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
public class Server_Q1 {
    private Socket socket = null;
    private ServerSocket server = null;
    private DataInputStream in = null;
    public Server_Q1(int port) throws IOException{
        try{
            server = new ServerSocket(port);
            System.out.println("Server started");
            System.out.println("waiting for client");
            socket = server.accept();
            System.out.println("Client Accepted");
            in = new DataInputStream(new BufferedReader(socket.getInputStream()));
            String line = "";
            while(!line.equals("Over")){
                try{
                    line = in.readUTF();
                    System.out.println(line);
                }
                catch(IOException i){
                    System.out.println(i);
                }
            }
            System.out.println("Closing connection");
            socket.close();
            in.close();
        }
        catch(IOException i){
            System.out.println(i);
        }
    }
    public static void main(String args[]) throws IOException{
```

```
        Server_Q1 server = new Server_Q1(5000);  
    }  
}
```

Output:

```
-----< com.mycompany:socketproj2 >-----  
Building socketproj2 1.0-SNAPSHOT  
-----[ jar ]-----  
  
--- exec-maven-plugin:3.0.0:exec (default-cli) @ socketproj2 ---  
Server started  
waiting for client  
Client Accepted  
Hello  
Hii  
Hi  
Over  
Closing connection  
  
BUILD SUCCESS  
  
Total time: 01:02 min  
Finished at: 2021-12-30T12:40:49+05:30  
-----
```

```
-----< com.mycompany:socketproj2 >-----  
Building socketproj2 1.0-SNAPSHOT  
-----[ jar ]-----  
  
--- exec-maven-plugin:3.0.0:exec (default-cli) @ socketproj2 ---  
Connected  
Hello  
Hi  
Over  
  
BUILD SUCCESS  
  
Total time: 56.986 s  
Finished at: 2021-12-30T12:40:49+05:30  
-----
```

6. To develop a program for multi-client chat server.

Code (Client):

```
import java.io.*;
import java.net.*;
import java.util.Scanner;
public class MultiChatClient
{
    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 {
                        dos.writeUTF(msg);
                        // write on the output stream
                    }
                    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();
                    }
                }
            }
        });
    }
}
```



```
        }  
        catch (IOException e) {  
            e.printStackTrace();  
        }  
    }  
});  
}  
sendMessage.start();  
readMessage.start();  
}  
}
```

Code (Server):

```
import java.io.*;  
import java.util.*;  
import java.net.*;  
public class MultiChatServer  
{  
    // 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
            }
            catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
}
```



```

        for (ClientHandler mc : MultiChatServer.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();
}
}
}

```

Output:

```

Building socketproj2 1.0-SNAPSHOT
-----[ jar ]-----

--- exec-maven-plugin:3.0.0:exec (default-cli) @ socketproj2 ---
New client request received : Socket[addr=/127.0.0.1,port=49479,localport=1234]
Creating a new handler for this client...
Adding this client to active client list
New client request received : Socket[addr=/127.0.0.1,port=49481,localport=1234]
Creating a new handler for this client...
Adding this client to active client list
hello#client 1
Hey#client 0

```

```

-----< com.mycompany:socketproj2 >-----
Building socketproj2 1.0-SNAPSHOT
-----[ jar ]-----

--- exec-maven-plugin:3.0.0:exec (default-cli) @ socketproj2 ---
hello#client 1
client 1 : Hey

```

```
-----< com.mycompany:socketproj2 >-----  
] Building socketproj2 1.0-SNAPSHOT  
-----[ jar ]-----  
] --- exec-maven-plugin:3.0.0:exec (default-cli) @ socketproj2 ---  
  client 0 : hello  
- Hey#client 0
```

7. To implement a Server calculator using RPC concept. (Make use of datagram)

Code (Client):

```
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.util.Scanner;
public class Client {
    public static void main(String[] args)throws IOException {
        Scanner sc = new Scanner(System.in);
        DatagramSocket ds= new DatagramSocket();
        InetAddress ip=InetAddress.getLocalHost();
        byte buf[]=null;
        while(true){
            System.out.println("Enter the equaion in the format:");
            System.out.println("Operand1 And operand2");
            String inp=sc.nextLine();
            buf=new byte[65535];
            buf=inp.getBytes();
            DatagramPacket DpSend = new DatagramPacket(buf,buf.length, ip, 1234);
            ds.send(DpSend);
            if(inp.equals("Exit"))
                break;
            buf=new byte[65535];
            DatagramPacket DpReceive = new DatagramPacket(buf, buf.length);
            ds.receive(DpReceive);
            System.out.println("Answer="+ new String(buf,0, buf.length));
        }
    }
}
```

Code (Server):

```
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.util.StringTokenizer;
public class Server {
    public static void main(String args[]) throws IOException{
        DatagramSocket ds=new DatagramSocket(1234);
        byte buf[] = null ;
        DatagramPacket DpSend=null;
        DatagramPacket DpReceive=null;
        while(true){
            buf=new byte[65535];
            DpReceive=new DatagramPacket(buf,buf.length);
```



```

ds.receive(DpReceive);
String inp=new String(buf, 0, buf.length);
inp=inp.trim();
System.out.println("Equaion Received:-"+inp);
if(inp.equals("Exit")){
    System.out.println("client Exiting");
    break;
}
int result;
StringTokenizer st=new StringTokenizer(inp);
int op1=Integer.parseInt(st.nextToken());
String operation=st.nextToken();
int op2=Integer.parseInt(st.nextToken());
if(operation.equals("+"))
    result=op1+op2;
else if(operation.equals("-"))
    result=op1-op2;
else if(operation.equals("*"))
    result=op1*op2;
else
    result=op1/op2;
System.out.println("Sending the result...");
String res=Integer.toString(result);
buf=res.getBytes();
int port=DpReceive.getPort();
DpSend =new DatagramPacket(buf, buf.length, InetAddress.getLocalHost(),port);
ds.send(DpSend);
}
}
}

```

Output:

```

Building socketproj2 1.0-SNAPSHOT
-----[ jar ]-----

--- exec-maven-plugin:3.0.0:exec (default-cli) @ socketproj2 ---
Enter the equaion in the format:
'Operand1 And operand2'
1 + 2
Answer=3...line is too long, please switch to wrapped mode to see whole line...
Enter the equaion in the format:
'Operand1 And operand2'
2 * 2
Answer=4...line is too long, please switch to wrapped mode to see whole line...
Enter the equaion in the format:
'Operand1 And operand2'
8 / 4
Answer=2...line is too long, please switch to wrapped mode to see whole line...
Enter the equaion in the format:
'Operand1 And operand2'

```

```
-----< com.mycompany:socketproj2 >-----  
Building socketproj2 1.0-SNAPSHOT  
-----[ jar ]-----  
  
--- exec-maven-plugin:3.0.0:exec (default-cli) @ socketproj2 ---  
Equaion Received:-1 + 2  
Sending the result...  
Equaion Received:-2 * 2  
Sending the result...  
Equaion Received:-8 / 4  
Sending the result...
```

8. To implement a Date Time Server using RPC concept.

Code (Client):

```
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.util.*;
public class UDP_Client {
    public static void main(String[] args) throws IOException {
        Scanner sc = new Scanner(System.in);
        DatagramSocket ds = new DatagramSocket();
        InetAddress ip= InetAddress.getLocalHost();
        byte buf[] = null;
        while(true){
            System.out.println("What do you want to know? Date / Time");
            String inp = sc.nextLine();
            buf = new byte[65535];
            buf = inp.getBytes();
            DatagramPacket DpSend = new DatagramPacket(buf, buf.length, ip, 1234);
            ds.send(DpSend);
            if(inp.equals("BYE"))
                break;
            buf = new byte[65535];
            DatagramPacket DpReceive = new DatagramPacket(buf,0, buf.length);
            ds.receive(DpReceive);
            System.out.println(inp+ ":" +new String(buf,0, buf.length));
        }
    }
}
```

Code (Server):

```
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.util.*;
import java.text.*;
public class UDP_Server {
    public static void main(String[] args) throws IOException {
        DatagramSocket ds = new DatagramSocket(1234);
        System.out.println("the chat server is Listenting on the port 1234. ");
        DateFormat forDate = new SimpleDateFormat("yyyy/mm/dd");
        DateFormat forTime = new SimpleDateFormat("hh:mm:ss");
        byte buf[]= null;
        DatagramPacket DpSend = null;
```

```
DatagramPacket DpReceive = null;
while(true){
    String toReturn = "";
    buf = new byte[65535];
    DpReceive = new DatagramPacket(buf, buf.length);
    ds.receive(DpReceive);
    String inp = new String (buf,0 ,buf.length);
    inp = inp.trim();
    if(inp.equals("BYE")){
        System.out.println("Client is saying Bye... exiting");
        break;
    }
    Date date = new Date();
    if(inp.equals("Date"))
        toReturn = forDate.format(date);
    else if(inp.equals("Time"))
        toReturn = forTime.format(date);
    System.out.println("Sending result ...");
    buf = toReturn.getBytes();
    int port = DpReceive.getPort();
    DpSend = new DatagramPacket(buf, buf.length, InetAddress.getLocalHost(),port);
    ds.send(DpSend);
}
}
```

Output:

```
-----< com.mycompany:socketproj2 >-----
Building socketproj2 1.0-SNAPSHOT
-----[ jar ]-----

--- exec-maven-plugin:3.0.0:exec (default-cli) @ socketproj2 ---
the chat server is Listenting on the port 1234.....
Sending result ...
Sending result ...
Client is saying Bye...exiting
-----
BUILD SUCCESS
-----

Total time: 24.878 s
Finished at: 2021-12-30T13:02:41+05:30
```



```
-----< com.mycompany:socketproj2 >-----  
Building socketproj2 1.0-SNAPSHOT  
-----[ jar ]-----  
  
--- exec-maven-plugin:3.0.0:exec (default-cli) @ socketproj2 ---  
What do you want to know? Date / Time  
Date  
Date:2021/02/30...line is too long, please switch to wrapped mode to see whole line...  
What do you want to know? Date / Time  
Time  
Time:01:02:35...line is too long, please switch to wrapped mode to see whole line...  
What do you want to know? Date / Time  
BYE  
-----  
BUILD SUCCESS  
-----  
Total time: 21.253 s  
Finished at: 2021-12-30T13:02:41+05:30  
-----
```

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

Code (Client):

```
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.util.*;
public class Client {
    private Client() {}
    public static void main(String[] args) throws Exception {
        try {
            // Getting the registry
            Registry registry = LocateRegistry.getRegistry(null);
            // Looking up the registry for the remote object
            Hello stub = (Hello) registry.lookup("Hello");
            // Calling the remote method using the obtained object
            @SuppressWarnings("unchecked")
            List<Student> list = (List) stub.getStudents();
            for (Student s: list) {
                // System.out.println("bc "+s.getBranch());
                System.out.println("ID: " + s.getId());
                System.out.println("name: " + s.getName());
                System.out.println("branch: " + s.getBranch());
                System.out.println("percent: " + s.getPercent());
                System.out.println("email: " + s.getEmail());
            }
            // System.out.println(list);
        } catch (Exception e) {
            System.err.println("Client exception: " + e.toString());
            e.printStackTrace();
        }
    }
}
```

Code (Server):

```
import java.rmi.registry.Registry;
import java.rmi.registry.LocateRegistry;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
public class Server extends ImplExample {
    public Server() {}
}
```

```

public static void main(String args[]) {
    try {
        // Instantiating the implementation class
        ImplExample obj = new ImplExample();
        // Exporting the object of implementation class (here we are exporting the remote
object to the stub)
        Hello stub = (Hello) UnicastRemoteObject.exportObject(obj, 0);
        // Binding the remote object (stub) in the registry
        Registry registry = LocateRegistry.getRegistry();
        registry.bind("Hello", stub);
        System.err.println("Server ready");
    } catch (Exception e) {
        System.err.println("Server exception: " + e.toString());
        e.printStackTrace();
    }
}
}

```

Code (Hello):

```

import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.*;
// Creating Remote interface for our application
public interface Hello extends Remote {
    public List<Student> getStudents() throws Exception; }

```

Code (ImplExample):

```

import java.sql.*;
import java.util.*;
// Implementing the remote interface
public class ImplExample implements Hello {
    // Implementing the interface method
    `public List<Student> getStudents() throws Exception {
        List<Student> list = new ArrayList<Student>();
        // JDBC driver name and database URL
        String JDBC_DRIVER = "com.mysql.jdbc.Driver";
        String DB_URL = "jdbc:mysql://localhost:3306/details";
        // Database credentials
        String USER = "root";
        String PASS = "admin";
        Connection conn = null; Statement stmt = null;
        //Register JDBC driver
        Class.forName("com.mysql.jdbc.Driver");
        //Open a connection
    }
}

```



```
System.out.println("Connecting to a selected database...");
conn = DriverManager.getConnection(DB_URL, USER, PASS);
System.out.println("Connected database successfully...");
//Execute a query
System.out.println("Creating statement...");
stmt = conn.createStatement();
String sql = "SELECT * FROM student_data";
ResultSet rs = stmt.executeQuery(sql);
//Extract data from result set
while(rs.next()) {
    // Retrieve by column name
    int id = rs.getInt("id");
    String name = rs.getString("name");
    String branch = rs.getString("branch");
    int percent = rs.getInt("percentage");
    String email = rs.getString("email");
    // Setting the values
    Student student = new Student();
    student.setID(id);
    student.setName(name);
    student.setBranch(branch);
    student.setPercent(percent);
    student.setEmail(email);
    list.add(student);
}
rs.close();
return list;
}
```

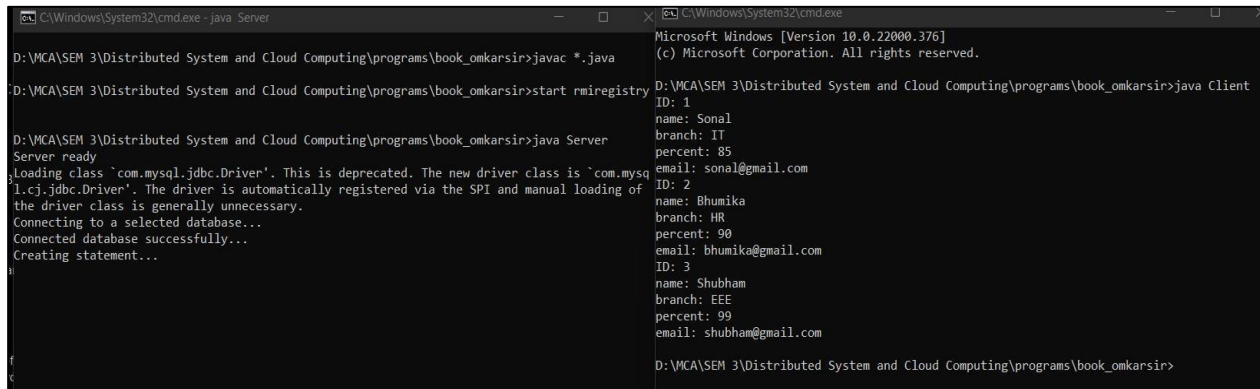
Code (Student):

```
public class Student implements java.io.Serializable {
    private int id, percent;
    private String name, branch, email;
    public int getId() {
        return id;
    }
    public String getName() {
        return name;
    }
    public String getBranch() {
        return branch;
    }
}
```



```
public int getPercent() {  
    return percent;  
}  
public String getEmail() {  
    return email;  
}  
public void setID(int id) {  
    this.id = id;  
}  
public void setName(String name) {  
    this.name = name;  
}  
public void setBranch(String branch) {  
    this.branch = branch;  
}  
public void setPercent(int percent) {  
    this.percent = percent;  
}  
public void setEmail(String email) {  
    this.email = email;  
}  
}
```

Output:



```
C:\Windows\System32\cmd.exe - java Server  
D:\MCA\SEM 3\Distributed System and Cloud Computing\programs\book_omkarsir>javac *.java  
D:\MCA\SEM 3\Distributed System and Cloud Computing\programs\book_omkarsir>start rmiregistry  
D:\MCA\SEM 3\Distributed System and Cloud Computing\programs\book_omkarsir>java Server  
Server ready  
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.  
Connecting to a selected database...  
Connected database successfully...  
Creating statement...  
D:\MCA\SEM 3\Distributed System and Cloud Computing\programs\book_omkarsir>  
C:\Windows\System32\cmd.exe  
Microsoft Windows [Version 10.0.22000.376]  
(c) Microsoft Corporation. All rights reserved.  
D:\MCA\SEM 3\Distributed System and Cloud Computing\programs\book_omkarsir>java Client  
ID: 1  
name: Sonal  
branch: IT  
percent: 85  
email: sonal@gmail.com  
ID: 2  
name: Bhumika  
branch: HR  
percent: 90  
email: bhumika@gmail.com  
ID: 3  
name: Shubham  
branch: EEE  
percent: 99  
email: shubham@gmail.com  
D:\MCA\SEM 3\Distributed System and Cloud Computing\programs\book_omkarsir>
```

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

Code (Client):

```
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.util.*;
public class Client {
    private Client() {}
    public static void main(String[] args) throws Exception {
        try {
            // Getting the registry
            Registry registry = LocateRegistry.getRegistry(null);
            // Looking up the registry for the remote object
            Hello stub = (Hello) registry.lookup("Hello");
            // Calling the remote method using the obtained object
            @SuppressWarnings("unchecked")
            List<Student> list = (List) stub.getStudents();
            for (Student s: list) {
                // System.out.println("bc "+s.getBranch());
                System.out.println("ID: " + s.getId());
                System.out.println("name: " + s.getName());
                System.out.println("branch: " + s.getBranch());
                //System.out.println("percent: " + s.getPercent());
                //System.out.println("email: " + s.getEmail());
            }
            // System.out.println(list);
        } catch (Exception e) {
            System.err.println("Client exception: " + e.toString());
            e.printStackTrace();
        }
    }
}
```

Code (Server):

```
import java.rmi.registry.Registry;
import java.rmi.registry.LocateRegistry;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
public class Server extends ImplExample {
```



```
public Server() {}
public static void main(String args[]) {
    try {
        // Instantiating the implementation class
        ImplExample obj = new ImplExample();
        // Exporting the object of implementation class (here we are exporting the remote
object to the stub)
        Hello stub = (Hello) UnicastRemoteObject.exportObject(obj, 0);
        // Binding the remote object (stub) in the registry
        Registry registry = LocateRegistry.getRegistry();
        registry.bind("Hello", stub);
        System.err.println("Server ready");
    } catch (Exception e) {
        System.err.println("Server exception: " + e.toString());
        e.printStackTrace();
    }
}
}
```

Code (Hello):

```
import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.*;
// Creating Remote interface for our application
public interface Hello extends Remote {
    public List<Student> getStudents() throws Exception; }

```

Code (Student):

```
public class Student implements java.io.Serializable {
    private int id, percent;
    private String name, branch, email;
    public int getId() {
        return id;
    }
    public String getName() {
        return name;
    }
    public String getBranch() {
        return branch;
    }
    public int getPercent() {
        return percent;
    }
}
```



```
public String getEmail() {  
    return email;  
}  
public void setID(int id) {  
    this.id = id;  
}  
public void setName(String name) {  
    this.name = name;  
}  
public void setBranch(String branch) {  
    this.branch = branch;  
}  
public void setPercent(int percent) {  
    this.percent = percent;  
}  
public void setEmail(String email) {  
    this.email = email;  
}  
}
```

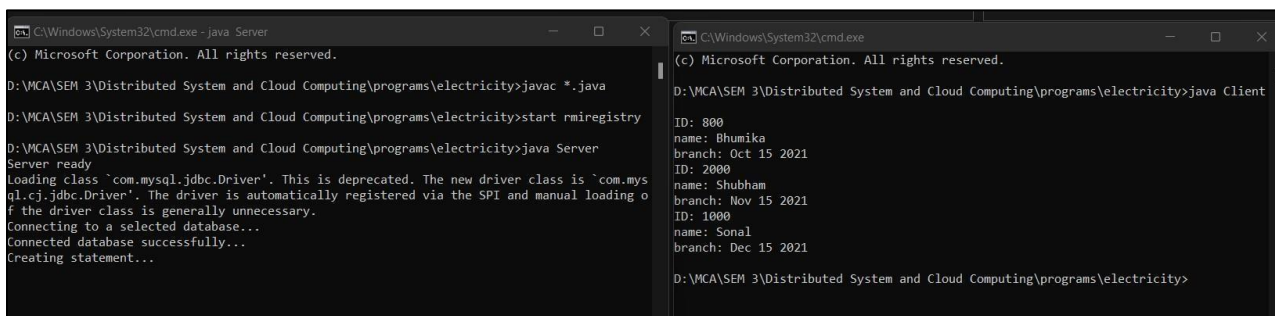
Code (ImplExample):

```
import java.sql.*;  
import java.util.*;  
// Implementing the remote interface  
public class ImplExample implements Hello {  
    // Implementing the interface method  
    public List<Student> getStudents() throws Exception {  
        List<Student> list = new ArrayList<Student>();  
        // JDBC driver name and database URL  
        String JDBC_DRIVER = "com.mysql.jdbc.Driver";  
        String DB_URL = "jdbc:mysql://localhost:3306/details1";  
        // Database credentials String USER = "root";  
        String PASS = "admin";  
        Connection conn = null;  
        Statement stmt = null;  
        //Register JDBC driver  
        Class.forName("com.mysql.jdbc.Driver");  
        //Open a connection  
        System.out.println("Connecting to a selected database...");  
        conn = DriverManager.getConnection(DB_URL, USER, PASS);  
        System.out.println("Connected database successfully...");  
        //Execute a query System.out.println("Creating statement...");
```



```
stmt = conn.createStatement();
String sql = "SELECT * FROM bill";
ResultSet rs = stmt.executeQuery(sql);
//Extract data from result set
while(rs.next()) {
    // Retrieve by column name
    int id = rs.getInt("bill_amount");
    String name = rs.getString("consumer_name");
    String branch = rs.getString("bill_due_date");
    //int percent = rs.getInt("percentage");
    //String email = rs.getString("email");
    // Setting the values
    Student student = new Student();
    student.setID(id);
    student.setName(name);
    student.setBranch(branch);
    //student.setPercent(percent);
    //student.setEmail(email);
    list.add(student);
}
rs.close();
return list;
}
```

Output:



```
C:\Windows\System32\cmd.exe - java Server
(c) Microsoft Corporation. All rights reserved.

D:\MCA\SEM 3\Distributed System and Cloud Computing\programs\electricity>javac *.java

D:\MCA\SEM 3\Distributed System and Cloud Computing\programs\electricity>start rmiregistry

D:\MCA\SEM 3\Distributed System and Cloud Computing\programs\electricity>java Server
Server ready
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.
Connecting to a selected database...
Connected database successfully...
Creating statement...

C:\Windows\System32\cmd.exe
(c) Microsoft Corporation. All rights reserved.

D:\MCA\SEM 3\Distributed System and Cloud Computing\programs\electricity>java Client
ID: 800
name: Bhumiika
branch: Oct 15 2021
ID: 2000
name: Shubham
branch: Nov 15 2021
ID: 1000
name: Sonal
branch: Dec 15 2021

D:\MCA\SEM 3\Distributed System and Cloud Computing\programs\electricity>
```

11. To develop applications using Google App Engine by using Eclipse IDE.

Steps:

1. Open any browser of your choice, go to cloud.google.com, login to your google account.
2. At the Top Right hand side click on Console, Click on Current Project -> Click on New Project.
3. Under Project Name -> Enter name of application -> Click on Create.
4. When the project is created, At top Right There is a icon of “Activate Cloud Shell” ->Click on it.
5. When the terminal is opened, Enter command “mkdir symca”.
6. Click on Open Editor -> Click on New File - > Enter file name “demo.java” -> Then select our directory “symca” -> Click on OK.
7. When the file is created -> Enter the following code :

```
public class demo{  
    public static void main(String[] args) {  
        System.out.println("Atharva Kale C22059 NMITD");  
    }  
}
```
8. Save the file “demo.java” -> Then click on Open Terminal
9. Enter command “cd symca” and press Enter
10. Enter command “javac demo.java” and press enter
11. Enter command “java demo” and press enter

Output:

```
atharvakale9@cloudshell:~/symca (javaproject-408608)$ javac demo.java  
atharvakale9@cloudshell:~/symca (javaproject-408608)$ java demo  
Atharva Kale C22059 NMITD  
atharvakale9@cloudshell:~/symca (javaproject-408608)$
```