**Practical No.3**

**Concept** : The Remote Method invocation is an API that provide a mechanism to create distributed application in java. The client invoke method via an interface. These methods are implement on the server side.

**A].Aim**: Implement a Server calculator containing ADD(), MUL(),SUB(),DIV() Using RMI

**CalciClient.java**

package com.mycompany.calciclient;

import java.net.MalformedURLException;

import java.rmi.Naming;

import java.rmi.NotBoundException;

import java.rmi.RemoteException;

import java.util.Scanner;

public class CalciClient {

public static void main(String[] args) throws NotBoundException, MalformedURLException, RemoteException {

Scanner sc=new Scanner(System.in);

try{

CalciInterface c= (CalciInterface)Naming.lookup("rmi://localhost:1099/CalciInterface");

System.out.println("Client is connected to server.");

System.out.println("Please enter your choice: \n1. add\n2. sub\n3. mul\n4. div\n");

int choice=sc.nextInt();

int x,y;

switch(choice){

case 1:

System.out.println("Enter x and y: ");

x=sc.nextInt();

y=sc.nextInt();

System.out.println(c.add(x,y));

break;

case 2:

System.out.println("Enter x and y: ");

x=sc.nextInt();

y=sc.nextInt();

System.out.println(c.sub(x,y));

break;

case 3:

System.out.println("Enter x and y: ");

x=sc.nextInt();

y=sc.nextInt();

System.out.println(c.mul(x,y));

break;

case 4

System.out.println("Enter x and y: ");

x=sc.nextInt();

y=sc.nextInt();

System.out.println(c.div(x,y));

break; }}

catch(Exception e){} }}

**Calciserver.java**

package com.mycompany.calciclient;

import java.rmi.NotBoundException;

import java.rmi.Remote;

import java.rmi.RemoteException;

import java.rmi.registry.Registry;

public class CalciServer {

public static void main(String[] args) throws RemoteException, NotBoundException {

Registry r=java.rmi.registry.LocateRegistry.createRegistry(1099);

r.rebind("CalciInterface", (Remote) new CalciRMI());

System.out.println("server is running");}}

**CalciRMI.java**

package com.mycompany.calciclient;

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

public class CalciRMI extends UnicastRemoteObject implements CalciInterface {

public CalciRMI() throws RemoteException{

int a,b;}

public int add (int a,int b)thro0000ws RemoteException{

return a+b;}

public int sub (int a,int b)throws RemoteException{

return a-b;}

public int mul (int a,int b)throws RemoteException{

return a\*b;}

public int div (int a,int b)throws RemoteException{ return a/b;}

public static void main(String[] args){

}}

**CalciInterface.java**

package com.mycompany.calciclient;

import java.rmi.Remote;

import java.rmi.RemoteException;

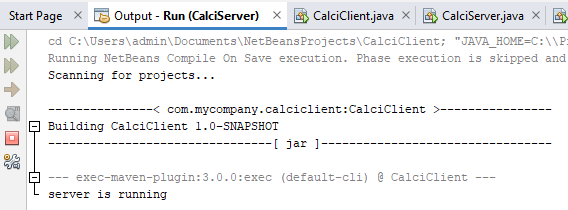
public interface CalciInterface extends Remote{public int add (int x,int y)throws RemoteException;

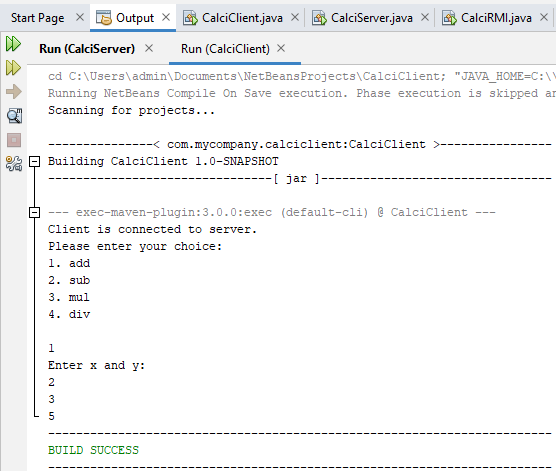
public int sub(int x,int y) throws RemoteException;

public int mul(int x,int y) throws RemoteException;

public int div(int x,int y) throws RemoteException; }

**OUTPUT**





**B].Aim:** Retrieve time and date function from server to client. This program should display server date and time.

**DatetimeClient.java**

package com.mycompany.datetimeclient;

import java.net.MalformedURLException;

import java.rmi.\*;

public class DateTimeClient {

public static void main(String args[]) {

try{

DateTimeInterface intf=(DateTimeInterface)

Naming.lookup("rmi://localhost:1099/DateServer");

System.out.println("The date on the server is: "+intf.getDate()); }

catch(MalformedURLException | NotBoundException | RemoteException e) {

} } }

**Datetimeserver.java**

package com.mycompany.datetimeclient;

import java.net.MalformedURLException;

import java.rmi.\*;

import java.rmi.registry.Registry;

public class DateTimeServer {

public static void main(String args[]) {

try{

Registry r= java.rmi.registry.LocateRegistry.

createRegistry(1099);

DateTimeRMI di=new DateTimeRMI();

Naming.rebind("DateServer", (Remote) di);

System.out.println("Datetime Server is ready"); }

catch(MalformedURLException | RemoteException e){} }}

**Datetimeinterface.java**

package com.mycompany.datetimeclient;

import java.rmi.\*;

public interface DateTimeInterface extends Remote{

String getDate() throws RemoteException;}

**DatetimeRMI.java**

package com.mycompany.datetimeclient;

import java.rmi.\*;

import java.rmi.server.\*;

import java.util.\*;

public class DateTimeRMI extends UnicastRemoteObject implements DateTimeInterface {

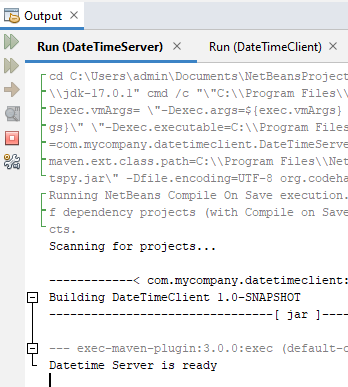
public DateTimeRMI() throws RemoteException {}

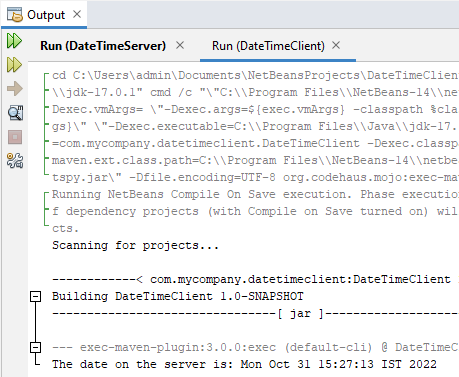
public String getDate() {

Date d=new Date();

return(d.toString()); }}

**output:**





**C]. Aim** Equation solver. The client should provide an equation to the server through an interface. The server will solve the expression given by the client. Such as (a-b)2 = a2 –2ab + b2 , (a+b)2 = a2 +2ab + b2, (a +b)3 = a3+3a2b+3ab2+b3  etc

**EqSolverClient .java**

package com.mycompany.eqsolverclient;

import java.rmi.\*;

import java.io.\*;

public class EqSolverClient {

public static void main(String[] args) {

try {

int num1=0, num2=0, res=0,choice;

EqSolverInterface object=(EqSolverInterface)Naming.lookup("hello");

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

System.out.println("Equations:-");

System.out.println("1. (a-b)2");

System.out.println("2. (a+b)2");

System.out.println("3. (a-b)3");

System.out.println("4. (a+b)3");

System.out.println("5. Exit");

while (true)

{ System.out.println("Choose the equation: ");

choice=Integer.parseInt(br.readLine());

if(choice<=4)

{

System.out.println("Enter the values of a and b");

num1=Integer.parseInt(br.readLine());

num2=Integer.parseInt(br.readLine());

}

switch(choice)

{ case 1:{

res=object.solveEq1(num1,num2);

System.out.println("Result is: "+res);

break;}

case 2:

{res=object.solveEq2(num1,num2);

System.out.println("Result is: "+res);

break;}

case 3:

{res=object.solveEq3(num1,num2);

System.out.println("Result is: "+res); break;} case 4:

{res=object.solveEq4(num1,num2);

System.out.println("Result is: "+res);

break;} case 5:

{ System.exit(0);

break;}

default:

{ System.out.println("Invalid option"); break; }

} } }

catch(Exception e) {} }}

**EqSolverServer.java**

package com.mycompany.eqsolverclient;

import java.rmi.\*;

import java.rmi.registry.Registry;

public class EqSolverServer {

public static void main(String[] args) throws RemoteException, NotBoundException {

try{

Registry r= java.rmi.registry.LocateRegistry.createRegistry(1099);

EquationSolverRMI obj=new EquationSolverRMI();

r.rebind("hello", obj);

System.out.println("Equation Solver Server is ready"); }

catch(Exception e) {} } }

**EqSolverInterface .java**

package com.mycompany.eqsolverclient;

import java.rmi.\*;

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface EqSolverInterface extends Remote{

public int solveEq1(int a, int b) throws RemoteException;

public int solveEq2(int a, int b) throws RemoteException;

public int solveEq3(int a, int b) throws RemoteException;

public int solveEq4(int a, int b) throws RemoteException; }

**EquationSolverRMI.java**

package com.mycompany.eqsolverclient;

import java.rmi.\*;

import java.rmi.server.\*;

import java.util.\*;

public class EquationSolverRMI extends UnicastRemoteObject implements EqSolverInterface {

public EquationSolverRMI() throws RemoteException {}

public int solveEq1(int a,int b) throws RemoteException {

int ans=(a\*a)-(2\*a\*b)+(b\*b);

return ans; }

public int solveEq2(int a,int b) throws RemoteException {

int ans=(a\*a)+(2\*a\*b)+(b\*b);

return ans; }

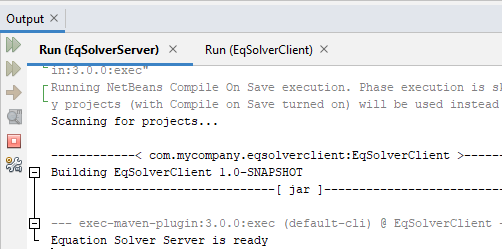
public int solveEq3(int a,int b) throws RemoteException {

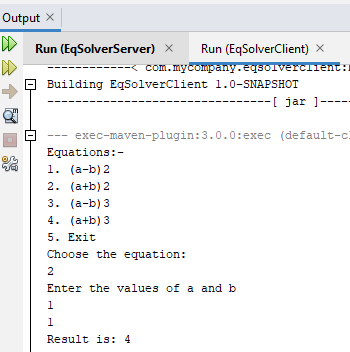
int ans=(a\*a\*a)-(3\*a\*a\*b)+(3\*a\*b\*b)-(b\*b\*b); return ans; }

public int solveEq4(int a,int b) throws RemoteException { int ans=(a\*a\*a)+(3\*a\*a\*b)+(3\*a\*b\*b)+(b\*b\*b);

return ans; } }

output





**PRACTICAL NO.5**

**Mutual Exclusion**

**Concept**: Token ring algorithm solves the mutual exclusion existing in the process communication.

**Aim:** Implementation of mutual exclusion using Token Ring Technique.

**TokenRing.java**

import java.net.\*;

import java.io.\*;

public class TokenRing {

public static DatagramSocket ds;

public static DatagramPacket dp;

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

try {

ds=new DatagramSocket(1000); }

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

**TokenRingClient1.java**

import java.net.\*;

import java.io.\*;

public class TokenRingClient1 {

public static DatagramSocket ds;

public static DatagramPacket dp;

public static BufferedReader br;

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

boolean hasToken;

try {

ds=new DatagramSocket(100); }

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

hasToken=true;

while(true) {

if(hasToken==true) {

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("Enter the data: ");

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

String str="Client-1===> "+br.readLine();

byte buff[]=new byte[1024];

buff=str.getBytes();

ds.send(new DatagramPacket (buff, buff.length,InetAddress.getLocalHost(), 1000));

System.out.println("Now sending..."); }

else if(ans.equalsIgnoreCase("no")) {

System.out.println("I am busy.");

//Sending message to client 2

String msg="Token";

byte buff1[]=new byte[1024];

buff1=msg.getBytes();

ds.send(new DatagramPacket(buff1, buff1.length, InetAddress.getLocalHost(),200));

hasToken=false;

//receiving message from Client 2

byte buff2[]=new byte[1024];

ds.receive(dp=new DatagramPacket(buff2, buff2.length));

String clientmsg=new String(dp.getData(),0,dp.getLength());

System.out.println("The data is: "+clientmsg);

if(clientmsg.equals("Token"))

hasToken=true;

System.out.println("I am leaving busy state.");

} }

else {

System.out.println("Enter in receive mode.");

byte buff[]=new byte[1024];

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; } } } } }

**TokenRingClient2.java**

import java.net.\*;

import java.io.\*;

public class TokenRingClient2 {

public static DatagramSocket ds;

public static DatagramPacket dp;

public static BufferedReader br;

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

boolean hasToken;

try {

ds=new DatagramSocket(200); }

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

hasToken=false;

while(true) {

if(hasToken==true) {

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("Enter the data: ");

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

String str="Client-2===> "+br.readLine();

byte buff1[]=new byte[1024];

buff1=str.getBytes();

ds.send(new DatagramPacket(buff1, buff1.length, InetAddress.getLocalHost(),1000));

System.out.println("Data sent."); }

else {

//Sending message to client 1

String clientmsg="Token";

byte buff2[]=new byte[1024];

buff2=clientmsg.getBytes();

ds.send(new DatagramPacket(buff2, buff2.length, InetAddress.getLocalHost(),100));

hasToken=false; } }

else {

try {

byte buff[]=new byte[1024];

System.out.println("Enter in receive mode.");

ds.receive(dp=new DatagramPacket(buff, buff.length));

String clientmsg1=new String(dp.getData(),0,dp.getLength());

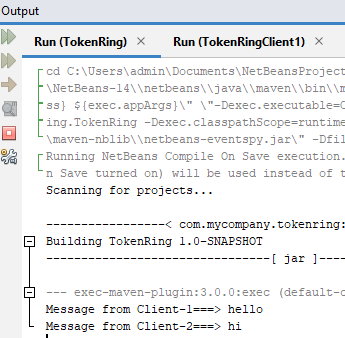
System.out.println("The data is: "+clientmsg1);

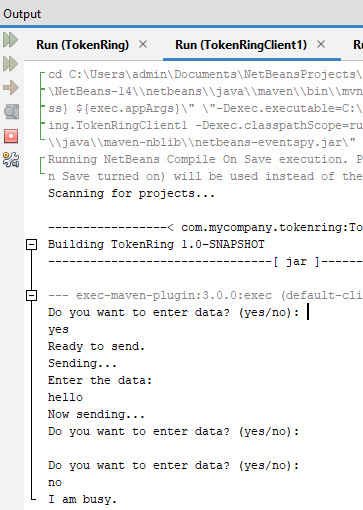
if(clientmsg1.equals("Token"))

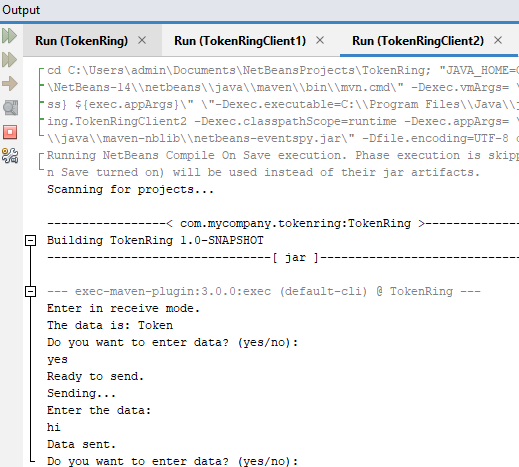
hasToken=true; }

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

**Output:**







**PRACTICAL NO.4**

**Remote Object Communication**

**Concept:** Pass remote objects from the server to the client. The client will receive the stub object (through remote interfaces) and saves it in an object variable with the same type as the remote interface. Then the client can access the actual object on the server through the variable.

**A]. Aim:** Using MySQL create College database. Create table Book  and retrieve the Book information from Library database using Remote Object Communication concept.

**DBClient.java**

package com.mycompany.dbclient;

import java.rmi.\*;

import java.io.\*;

public class DBClient

{

public static void main(String[] args)

{

String db="", sql="", ch="", ch1="", res="";

try

{

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

while(true)

{

System.out.println("Retrieve College Information.");

db="college";

System.out.println("Select an option");

System.out.println("a) Retrieve Student Information.");

System.out.println("b) Retreive Books Information.");

System.out.println("Enter your choice: ");

ch1=br.readLine();

if(ch1.equals("a"))

{

sql="select \* from student";

}

else if(ch1.equals("b"))

{

sql="select \* from book";

}

else

{

System.out.println("Please select a valid option.");

System.exit(0);

}

DBIntf id=(DBIntf)Naming.lookup("rmi://localhost:1099/DBConn");

res=id.getData(sql,db);

System.out.println(res);

}

}

catch (Exception e)

{

e.printStackTrace();

}

}

}

**DBServer.java**

package com.mycompany.dbclient;

import java.rmi.\*;

import java.rmi.registry.Registry;

public class DBServer {

public static void main(String[] args) {

try {

Registry r= java.rmi.registry.LocateRegistry.

createRegistry(1099);

DBCollege di=new DBCollege();

Naming.rebind("DBConn",(Remote) di);

System.out.println("Server Registered."); }

catch(Exception e) {

e.printStackTrace();} } }

**DBIntf.java**

package com.mycompany.dbclient;

import java.rmi.\*;

public interface DBIntf extends Remote

{

public String getData(String s, String db) throws RemoteException;

}

**DBCollege.java**

package com.mycompany.dbclient;

import java.rmi.\*;

import java.rmi.server.\*;

import java.sql.\*;

public class DBCollege extends UnicastRemoteObject implements DBIntf{

String str="", str1="";

public DBCollege() throws RemoteException {}

public String getData(String sql, String dsn) throws RemoteException {

String URL="jdbc:mysql://localhost/"+dsn; //dsn=data source name

try {

Class.forName("com.mysql.jdbc.Driver");

Connection con=DriverManager.

getConnection(URL,"root","");

System.out.println("Database Connected Successfully.");

Statement s=con.createStatement();

ResultSet rs=s.executeQuery(sql);

ResultSetMetaData rsmd=rs.getMetaData();

str1="";

str="";

for(int i=1;i<=rsmd.getColumnCount();i++) {

str1=str1+rsmd.getColumnName(i)+"\t"; }

System.out.println();

while(rs.next()) {

for(int i=1;i<=rsmd.getColumnCount();i++) {

str=str+rs.getString(i)+"\t"; }

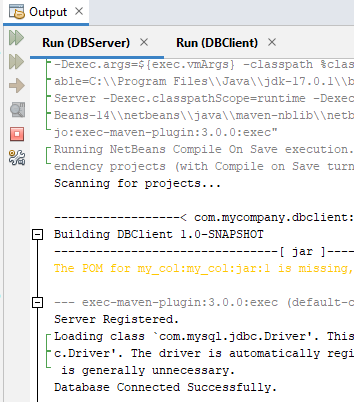
str=str+"\n"; } }

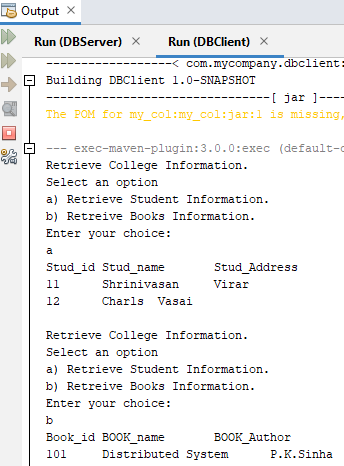
catch(Exception e) {

e.printStackTrace(); }

return(str1+"\n"+str); }}

**OUTPUT:**





**B]. Aim:** Using MySQL create Employee database. Create table employee and retrieve the employee information from the Employee database using Remote Object Communication concept.

**EmployeeClient.java**

package com.mycompany.employeeclient;

import java.rmi.\*;

import java.io.\*;

public class EmployeeClient {

public static void main(String[] args) {

String db="", sql="", ch="", ch1="", res="";

try {

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

while(true) {

System.out.println("1. Press E to retrieve Employee information");

System.out.println("2. Press x to exit the system.");

ch=br.readLine();

if(ch.equals("E")||ch.equals("e")) {

db="emp";

sql="select \* from employee"; }

else if(ch.equals("x")) {

System.exit(0); }

else {

System.out.println("Please select a valid option."); }

EmployeeIntf id=(EmployeeIntf)Naming.

lookup("rmi://localhost:1099/EmpDBConn");

res=id.getInfo(sql,db);

System.out.println(res); } }

catch (Exception e) {

e.printStackTrace(); } }}

**EmployeeServer.java**

package com.mycompany.employeeclient;

import java.rmi.\*;

import java.rmi.registry.Registry;

public class EmployeeServer {

public static void main(String[] args) {

try {

Registry r= java.rmi.registry.LocateRegistry.

createRegistry(1099);

EmployeeDBRMI di=new EmployeeDBRMI();

Naming.rebind("EmpDBConn",(Remote) di);

System.out.println("Server Registered."); }

catch(Exception e) { } }}

**EmployeeIntf.java**

package com.mycompany.employeeclient;

import java.rmi.\*;

public interface EmployeeIntf extends Remote

{

public String getInfo(String s, String db) throws RemoteException; }

**EmployeeDBRMI.java**

package com.mycompany.employeeclient;

import java.rmi.\*;

import java.rmi.server.\*;

import java.sql.\*;

public class EmployeeDBRMI extends UnicastRemoteObject implements EmployeeIntf{

String str="", str1="";

public EmployeeDBRMI() throws RemoteException {}

public String getInfo(String sql, String dsn) throws RemoteException {

String URL="jdbc:mysql://localhost/"+dsn;

try {

Class.forName("com.mysql.jdbc.Driver"); Connection con=DriverManager.getConnection(URL,"root","");

System.out.println("Database Connected Successfully.");

Statement s=con.createStatement();

ResultSet rs=s.executeQuery(sql);

ResultSetMetaData rsmd=rs.getMetaData();

str1="";

str="";

for(int i=1;i<=rsmd.getColumnCount();i++) {

str1=str1+rsmd.getColumnName(i)+"\t"; }

System.out.println();

while(rs.next()) {

for(int i=1;i<=rsmd.getColumnCount();i++) {

str=str+rs.getString(i)+"\t"; }

str=str+"\n"; } }

catch(Exception e) {

e.printStackTrace();}

return(str1+"\n"+str); }}

**OUTPUT:**

