**Assignment no :1**

**Search.java**

import java.rmi.\*;

public interface Search extends Remote

{

public String query(String search)throws

RemoteException;

}

**SearchQuery.java**

import java.rmi.\*;

import java.rmi.server.\*;

public class SearchQuery extends RemoteObject

implements Search

{

public String query(String search)

throws RemoteException

{

String result;

if(search.equals("Reflection in Java"))

result = "Found";

else

result = "Not Found";

return result;

}

}

**Server.java**

import java.net.\*;

import java.io.\*;

public class Server

{

private Socket socket = null;

private ServerSocket server = null;

private DataInputStream in = null;

public Server(int port)

{

try

{

server = new ServerSocket(port);

System.out.println("Server Started");

System.out.println("Waiting for a client....");

 socket = server.accept();

 System.out.println("Client accepted");

 in = new DataInputStream(new BufferedInputStream(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[])

{

Server server = new Server(5000);

}

}

**Client.java**

import java.io.\*;

import java.net.\*;

public class Client

{

private Socket socket = null;

private BufferedReader d = null;

private InputStream input = null;

private DataOutputStream out = null;

public Client(String address , int port)

{

try

{

socket = new Socket(address,port);

System.out.println("Connected");

System.out.println("Done with 1st program Of DS");

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

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

}

catch(UnknownHostException u)

{

System.out.println(u);

return;

}

catch(IOException i)

{

System.out.println(i);

return;

}

String line = " " ;

while(!line.equals("Over"))

{

try

{

line = d.readLine();

out.writeUTF(line);

}

catch(IOException i)

{

System.out.println(i);

}}

try

{

input.close();

out.close();

socket.close();

}

catch(IOException i)

{

System.out.println(i);

}}

public static void main(String args[])

{

Client client  = new Client("127.0.0.1",5000);

}

}

**Command:**

**1]**

**javac Search.java**

**javac SearchQuery.java**

**rmic SearchQuery**

**rmiregistry &**

**javac Server.java**

**java Server**

**2]**

**javac Client.java**

**java Client**

**Assignment no : 2**

**Calc.idl**

module CalcApp

{

interface Calc

{

exception DivisionByZero {};

float sum(in float a, in float b);

float div(in float a, in float b) raises (DivisionByZero);

float mul(in float a, in float b);

float sub(in float a, in float b);

};

};

**CalcServer.java**

import CalcApp.\*;

import CalcApp.CalcPackage.DivisionByZero;

import org.omg.CosNaming.\*;

import org.omg.CosNaming.NamingContextPackage.\*;

import org.omg.CORBA.\*;

import org.omg.PortableServer.\*;

import java.util.Properties;

class CalcImpl extends CalcPOA {

@Override

public float sum(float a, float b) {

return a + b;

}

@Override

public float div(float a, float b) throws DivisionByZero {

if (b == 0) {

throw new CalcApp.CalcPackage.DivisionByZero();

} else {

return a / b;

}

}

@Override

public float mul(float a, float b) {

return a \* b;

}

@Override

public float sub(float a, float b) {

return a - b;

}

private ORB orb;

public void setORB(ORB orb\_val) {

orb = orb\_val;

}

}

public class CalcServer {

public static void main(String args[]) {

try {

// create and initialize the ORB

ORB orb = ORB.init(args, null);

// get reference to rootpoa & activate the POAManager

POA rootpoa = POAHelper.narrow(orb.resolve\_initial\_references("RootPOA"));

rootpoa.the\_POAManager().activate();

// create servant and register it with the ORB

CalcImpl helloImpl = new CalcImpl();

helloImpl.setORB(orb);

// get object reference from the servant

org.omg.CORBA.Object ref = rootpoa.servant\_to\_reference(helloImpl);

Calc href = CalcHelper.narrow(ref);

// get the root naming context

// NameService invokes the name service

org.omg.CORBA.Object objRef = orb.resolve\_initial\_references("NameService");

// Use NamingContextExt which is part of the Interoperable

// Naming Service (INS) specification.

NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);

// bind the Object Reference in Naming

String name = "Calc";

NameComponent path[] = ncRef.to\_name(name);

ncRef.rebind(path, href);

System.out.println("Ready..");

// wait for invocations from clients

orb.run();

} catch (Exception e) {

System.err.println("ERROR: " + e);

e.printStackTrace(System.out);

}

System.out.println("Exiting ...");

}

}

**CalcClient.java**

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import CalcApp.\*;

import CalcApp.CalcPackage.DivisionByZero;

import org.omg.CosNaming.\*;

import org.omg.CosNaming.NamingContextPackage.\*;

import org.omg.CORBA.\*;

import static java.lang.System.out;

public class CalcClient {

static Calc calcImpl;

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

public static void main(String args[]) {

try {

// create and initialize the ORB

ORB orb = ORB.init(args, null);

// get the root naming context

org.omg.CORBA.Object objRef = orb.resolve\_initial\_references("NameService");

// Use NamingContextExt instead of NamingContext. This is

// part of the Interoperable naming Service.

NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);

// resolve the Object Reference in Naming

String name = "Calc";

calcImpl = CalcHelper.narrow(ncRef.resolve\_str(name));

System.out.println(calcImpl);

while (true) {

out.println("1. Sum");

out.println("2. Sub");

out.println("3. Mul");

out.println("4. Div");

out.println("5. exit");

out.println("--");

out.println("choice: ");

try {

String opt = br.readLine();

if (opt.equals("5")) {

break;

} else if (opt.equals("1")) {

out.println("a+b= " + calcImpl.sum(getFloat("a"), getFloat("b")));

} else if (opt.equals("2")) {

out.println("a-b= " + calcImpl.sub(getFloat("a"), getFloat("b")));

} else if (opt.equals("3")) {

out.println("a\*b= " + calcImpl.mul(getFloat("a"), getFloat("b")));

} else if (opt.equals("4")) {

try {

out.println("a/b= " + calcImpl.div(getFloat("a"), getFloat("b")));

} catch (DivisionByZero de) {

out.println("Division by zero!!!");

}

}

} catch (Exception e) {

out.println("===");

out.println("Error with numbers");

out.println("===");

}

out.println("");

}

//calcImpl.shutdown();

} catch (Exception e) {

System.out.println("ERROR : " + e);

e.printStackTrace(System.out);

}

}

static float getFloat(String number) throws Exception {

out.print(number + ": ");

return Float.parseFloat(br.readLine());

}

}

**Command:**

1]

idlj –fall Calc.idl

javac \*.java CalcApp/\*.java

orbd -ORBInitialPort 1050&

2]

javac CalcServer.java

java CalcServer -ORBInitialPort 1050

3]

javac CalcClient.java

java CalcClient -ORBInitialPort 1050

**Assignment No : 3**

**Hello.c**

#include <stdio.h>

#include "mpi.h"

int main(int argc, char\* argv[])

{

int rank, size, len;

MPI\_Init(&argc, &argv);

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank);

MPI\_Comm\_size(MPI\_COMM\_WORLD, &size);

printf("Hello, world, I am %d of %d\n",rank, size);

MPI\_Finalize();

return 0;

}

**World.c**

#include <stdio.h>

#include "mpi.h"

int main(int argc, char\* argv[])

{

int rank, size, len;

int num=10;

MPI\_Init(&argc, &argv);

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank);

MPI\_Comm\_size(MPI\_COMM\_WORLD, &size);

if(rank == 0)

{

printf("Sending message containing: %d from rank %d\n", num,rank);

MPI\_Send(&num, 1, MPI\_INT, 1, 1, MPI\_COMM\_WORLD);

}

else

{

printf(" at rank %d\n",rank);

MPI\_Recv(&num, 1, MPI\_INT, 0, 1, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);

printf("Received message containing: %d at rank %d\n", num,rank);

}MPI\_Finalize();

return 0;

}

**World1.c**

#include <stdio.h>

#include "mpi.h"

int main(int argc, char\* argv[])

{

int rank, size;

int num[20]; //N=20, n=4

MPI\_Init(&argc, &argv);

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank);

MPI\_Comm\_size(MPI\_COMM\_WORLD, &size);

for(int i=0;i<20;i++)

num[i]=i+1;

if(rank == 0){

int s[4];

printf("Distribution at rank %d \n", rank);

for(int i=1;i<4;i++)

MPI\_Send(&num[i\*5], 5, MPI\_INT, i, 1, MPI\_COMM\_WORLD); //N/n i.e. 20/4=5

int sum=0, local\_sum=0;

for(int i=0;i<5;i++)

{

local\_sum=local\_sum+num[i];

}

for(int i=1;i<4;i++)

{

MPI\_Recv(&s[i], 1, MPI\_INT, i, 1, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);

}

printf("local sum at rank %d is %d\n", rank,local\_sum);

sum=local\_sum;

for(int i=1;i<4;i++)

sum=sum+s[i];

printf("final sum = %d\n\n",sum);

} else {

int k[5];

MPI\_Recv(k, 5, MPI\_INT, 0, 1, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);

int local\_sum=0;

for(int i=0;i<5;i++)

{

local\_sum=local\_sum+k[i];

}

printf("local sum at rank %d is %d\n", rank, local\_sum);

MPI\_Send(&local\_sum, 1, MPI\_INT, 0, 1, MPI\_COMM\_WORLD);

}

MPI\_Finalize();

return 0;

}

**Command:**

mpicc hello.c

mpirun –np 4 ./a.out

mpicc world1.c

mpirun –np 0 ./a.out

mpicc world.c

mpirun –np 0 ./a.out

**Assignment No : 4**

**BerkeleyClockSync.java**

import java.util.ArrayList;

public class BerkeleyClockSync {

public static void main(String[] args) {

int[] systemClocks = { 10, 12, 13, 11, 14 };

int masterClock = 0;

// Print the initial system clocks

System.out.print("System clocks: ");

for (int clock : systemClocks) {

System.out.print(clock + " ");

}

System.out.println();

// Calculate the average system clock

int sum = 0;

for (int clock : systemClocks) {

sum += clock;

}

int averageClock = sum / systemClocks.length;

ArrayList<Integer> timeDifferences = new ArrayList<>();

for (int clock : systemClocks) {

timeDifferences.add(averageClock - clock);

}

int timeAdjustment = 0;

for (int clock : systemClocks) {

timeDifferences.add(averageClock - clock);

}

// Calculate the time adjustment for the master clock

timeAdjustment = 0;

for (int difference : timeDifferences) {

timeAdjustment += difference;

}

timeAdjustment /= timeDifferences.size();

// Update the master clock

masterClock = averageClock - timeAdjustment;

System.out.print("Updated system clocks: ");

for (int clock : systemClocks) {

System.out.print((clock - timeAdjustment) + " ");

}

System.out.println();

System.out.println("Master clock: " + masterClock);

}

}

**Command:**

javac BerkeleyClockSync.java

java BerkeleyClockSync

**Assignment No : 5**

**Tokenring.java**

import java.io.\*;

import java.util.\*;

class tokenring {

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

Scanner scan = new Scanner(System.in);

System.out.println("Enter the num of nodes:");

int n = scan.nextInt();

int m = n - 1;

// Decides the number of nodes forming the ring

int token = 0;

int ch = 0, flag = 0;

for (int i = 0; i < n; i++) {

System.out.print(" " + i);

}

System.out.println(" " + 0);

do{

System.out.println("Enter sender:");

int s = scan.nextInt();

System.out.println("Enter receiver:");

int r = scan.nextInt();

System.out.println("Enter Data:");

int a;

a = scan.nextInt();

System.out.print("Token passing:");

for (int i = token, j = token; (i % n) != s; i++, j = (j + 1) % n) {

System.out.print(" " + j + "->");

}

System.out.println(" " + s);

System.out.println("Sender " + s + " sending data: " + a);

for (int i = s + 1; i != r; i = (i + 1) % n) {

System.out.println("data " + a + " forwarded by " + i);

}

System.out.println("Receiver " + r + " received data: " + a +"\n");

token = s;

do{

try {

if( flag == 1)

System.out.print("Invalid Input!!...");

System.out.print("Do you want to send again?? enter 1 for Yes and 0 for No : ");

ch = scan.nextInt();

if( ch != 1 && ch != 0 )

flag = 1;

else

flag = 0;

} catch (InputMismatchException e){

System.out.println("Invalid Input");

}

}while( ch != 1 && ch != 0 );

}while( ch == 1 );

}

}

**Command:**

javac tokenring.java

java tokenring

**Assignment No : 6**

**Bully.java**

import java.io.InputStream;

import java.io.PrintStream;

import java.util.Scanner;

public class Bully

{

static boolean[] state = new boolean[5];

int coordinator;

public static void up(int up)//4

{

if (state[up - 1])// 0 1 2 3 4

{

System.out.println("process" + up + "is already up");

}

else

{

int i;

Bully.state[up - 1] = true;

System.out.println("process " + up + "held election");

for (i = up; i < 5; ++i)

{

System.out.println("election message sent from process" + up + "to process" + (i + 1));

}

for (i = up + 1; i <= 5; ++i)

{

if (!state[i - 1]) continue;

System.out.println("alive message send from process" + i + "to process" + up);

break;

}

}

}

public static void down(int down)

{

if (!state[down - 1])

{

System.out.println("process " + down + "is already dowm.");

}

else

{

Bully.state[down - 1] = false;

}

}

public static void mess(int mess)

{

if (state[mess - 1])

{

if (state[4])

{

System.out.println("0K");

}

else if (!state[4])

{

int i;

System.out.println("process" + mess + "election");

for (i = mess; i < 5; ++i)

{

System.out.println("election send from process" + mess + "to process " + (i + 1));

}

for (i = 5; i >= mess; --i)

{

if (!state[i - 1]) continue;

System.out.println("Coordinator message send from process" + i + "to all");

break;

}

}

}

else

{

System.out.println("Prccess" + mess + "is down");

}

}

public static void main(String[] args)

{

int choice;

Scanner sc = new Scanner(System.in);

for (int i = 0; i < 5; ++i)

{

Bully.state[i] = true;

}

System.out.println("5 active process are:");

System.out.println("Process up = p1 p2 p3 p4 p5");

System.out.println("Process 5 is coordinator");

do

{

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

System.out.println("1 up a process.");

System.out.println("2.down a process");

System.out.println("3 send a message");

System.out.println("4.Exit");

choice = sc.nextInt();

switch (choice)

{

case 1:

{

System.out.println("bring proces up");

int up = sc.nextInt();

if (up == 5)

{

System.out.println("process 5 is co-ordinator");

Bully.state[4] = true;

break;

}

Bully.up(up);

break;

}

case 2:

{

System.out.println("bring down any process.");

int down = sc.nextInt();

Bully.down(down);

break;

}

case 3:

{

System.out.println("which process will send message");

int mess = sc.nextInt();

Bully.mess(mess);

}

}

} while (choice != 4);

}

}

**Ring.java**

import java.util.Scanner;

public class Ring

{

public static void main(String[] args)

{

// TODO Auto-generated method stub

int temp, i, j;

char str[] = new char[10];

Rr proc[] = new Rr[10];

// object initialisation

for (i = 0; i < proc.length; i++)

proc[i] = new Rr();

// scanner used for getting input from console

Scanner in = new Scanner(System.in);

System.out.println("Enter the number of process : ");

int num = in.nextInt();

// getting input from users

for (i = 0; i < num; i++)

{

proc[i].index = i;

System.out.println("Enter the id of process : ");

proc[i].id = in.nextInt();

proc[i].state = "active";

proc[i].f = 0;

}

// sorting the processes from on the basis of id

for (i = 0; i < num - 1; i++)

{

for (j = 0; j < num - 1; j++)

{

if (proc[j].id > proc[j + 1].id)

{

temp = proc[j].id;

proc[j].id = proc[j + 1].id;

proc[j + 1].id = temp;

}

}

}

for (i = 0; i < num; i++)

{

System.out.print(" [" + i + "]" + " " + proc[i].id);

}

int init;

int ch;

int temp1;

int temp2;

int ch1;

int arr[] = new int[10];

proc[num - 1].state = "inactive";

System.out.println("\n process " + proc[num - 1].id + "select as co-ordinator");

while (true)

{

System.out.println("\n 1.election 2.quit ");

ch = in.nextInt();

for (i = 0; i < num; i++)

{

proc[i].f = 0;

}

switch (ch)

{

case 1:

System.out.println("\n Enter the Process number who initialsied election : ");

init = in.nextInt();

temp2 = init;

temp1 = init + 1;

i = 0;

while (temp2 != temp1)

{

if ("active".equals(proc[temp1].state) && proc[temp1].f == 0)

{

System.out.println("\nProcess " + proc[init].id + "send message to " + proc[temp1].id);

proc[temp1].f = 1;

init = temp1;

arr[i] = proc[temp1].id;

i++;

}

if (temp1 == num)

{

temp1 = 0;

}

else

{

temp1++;

}

}

System.out.println("\nProcess " + proc[init].id + " send message to " +

proc[temp1].id);

arr[i] = proc[temp1].id;

i++;

int max = -1;

// finding maximum for co-ordinator selection

for (j = 0; j < i; j++)

{

if (max < arr[j])

{

max = arr[j];

}

}

// co-ordinator is found then printing on console

System.out.println("\n process " + max + "select as co-ordinator");

for (i = 0; i < num; i++)

{

if (proc[i].id == max)

{

proc[i].state = "inactive";

}

}

break;

case 2:

System.out.println("Program terminated ...");

return ;

default:

System.out.println("\n invalid response \n");

break;

}

}

}

}

class Rr

{

public int index; // to store the index of process

public int id; // to store id/name of process

public int f;

String state;

// indiactes whether active or inactive state of node

}

**Command:**

javac Bully.java

java Bully