Assignment 2

Title : Remote Method Invocation

Program :

Interface - AddServerIntf.java

```
import java.rmi.*;

public interface AddServerIntf extends Remote {
   double add(double d1, double d2) throws RemoteException;
}
```

Class – AddServerImpl.java

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

public class AddServerImpl extends UnicastRemoteObject
       implements AddServerIntf {
   public AddServerImpl() throws RemoteException {
   }

   public double add(double d1, double d2) throws RemoteException {
      return d1 + d2;
   }
}
```

Class – AddServer.java

```
import java.rmi.*;

public class AddServer {
   public static void main(String args[]) {
      try {
         // create remote object
         AddServerImpl addServerImpl = new AddServerImpl();
         // bind the remote object
         Naming.rebind("AddServer", addServerImpl);
      } catch (Exception e) {
         System.out.println("Exception: " + e);
      }
   }
}
```

Class- AddClient.java

```
import java.rmi.*;

public class AddClient {
```

```
   public static void main(String args[]) {
      try {
         String addServerURL = "rmi://" + args[0] + "/AddServer";
         AddServerIntf addServerIntf = (AddServerIntf) Naming.lookup(addServerURL);
         System.out.println("The first number is: " + args[1]);
         double d1 = Double.valueOf(args[1]).doubleValue();
         System.out.println("The second number is: " + args[2]);
         double d2 = Double.valueOf(args[2]).doubleValue();
         System.out.println("The sum is: " + addServerIntf.add(d1, d2));
      } catch (Exception e) {
         System.out.println("Exception: " + e);
      }
   }
}
```

Output :

Kunal E:\BEIT\Sem 2\LP V\performed\Assignment1\src> javac AddServer.java

Kunal E:\BEIT\Sem 2\LP V\performed\Assignment1\src> javac AddClient.java

PS E:\BEIT\Sem 2\LP V\performed\Assignment1\src> rmiregistry

WARNING: A terminally deprecated method in java.lang.System has been called

WARNING: System::setSecurityManager has been called by sun.rmi.registry.RegistryImpl

WARNING: Please consider reporting this to the maintainers of sun.rmi.registry.RegistryImpl

WARNING: System::setSecurityManager will be removed in a future release.

Kunal E:\BEIT\Sem 2\LP V\performed\Assignment1\src> java AddServer

Server Started

Kunal E:\BEIT\Sem 2\LP V\performed\Assignment1\src>java AddClient

Addition : 46

Assignment 3

Title : Common Object Request Broker Architecture (CORBA)

Program :

ReverseClient.java

```
import ReverseModule.*;
import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContextPackage.*;
import org.omg.CORBA.*;
import java.io.*;

class ReverseClient {
   public static void main(String args[]) {
      Reverse Reverselmpl = null;
      try {
         // initialize the ORB
         org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args, null);
         org.omg.CORBA.Object objRef = rb.resolve_initial_references("NameService");
         NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);
         String name = "Reverse";
         Reverselmpl = ReverseHelper.narrow(ncRef.resolve_str(name));
         System.out.println("Enter String=");
         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
         String str = br.readLine();
         String tempStr = Reverselmpl.reverse_string(str);
         System.out.println(tempStr);
      } catch (Exception e) {
         e.printStackTrace();
      }
   }
}
```

ReverseImpl.java

```
import ReverseModule.ReversePOA;
import java.lang.String;

class Reverselmpl extends ReversePOA {
   Reverselmpl() {
      super();
      System.out.println("Reverse Object Created");
   }
```

```
    public String reverse_string(String name) {
        StringBuffer str = new StringBuffer(name);
        str.reverse();
        return ("Server Send " + str);
    }
}
```

ReverseServer.java

```
import ReverseModule.Reverse;
import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContextPackage.*;
import org.omg.CORBA.*;
import org.omg.PortableServer.*;

class ReverseServer {
    public static void main(String[] args) {
        try {
            // Initialize the ORB
            org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args, null);

            // Initialize the POA
            POA rootPOA = POAHelper.narrow(orb.resolve_initial_references("RootPOA"));
            rootPOA.the_POAManager().activate();

            // Creating the ReverseImpl object
            Reverselmpl rvr = new Reverselmpl();

            // Get the object reference from the servant class
            org.omg.CORBA.Object ref = rootPOA.servant_to_reference(rvr);
            System.out.println("Step1");
            Reverse href = ReverseModule.ReverseHelper.narrow(ref);
            System.out.println("Step2");
            // Resolve the initial references for the naming service
            org.omg.CORBA.Object objRef = orb.resolve_initial_references("NameService");
            System.out.println("Step3");
            NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);
            System.out.println("Step4");
            String name = "Reverse";
            NameComponent path[] = ncRef.to_name(name);
            ncRef.rebind(path, h_ref);

            System.out.println("Reverse Server reading and waiting....");
            orb.run();
        } catch (Exception e) {
```

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

Output :

```
kunal@ubuntu-22:~/Downloads/Ass 3$ idlj -fall ReverseModule.idl
kunal@ubuntu-22:~/Downloads/Ass 3$ javac*.java ReverseModule/*.java
kunal@ubuntu-22:~/Downloads/Ass 3$ ordb -ORBInitialPort 1050&
kunal@ubuntu-22:~/Downloads/Ass 3$ java ReverseServer -ORBInitialPort 1050& -
ORBInitialHost localhost&
Reversr Object Created
Step 1
Step 2
Step 3
Step 4
Reverse Server reading and waiting....
kunal@ubuntu-22:~/Downloads/Ass 3$ java ReverseServer -ORBInitialPort 1050& -
ORBInitialHost localhost&
Enter String=
Welcome to the home
Server send
```

Assignment 4

Title : Message Passing Interface (MPI)

Program :

ScatterGather.java

```
import mpi.MPI;

public class ScatterGather {
   public static void main(String args[]){
      //Initialize MPI execution environment
      MPI.Init(args);
      //Get the id of the process
      int rank = MPI.COMM_WORLD.Rank();
      //total number of processes is stored in size
      int size = MPI.COMM_WORLD.Size();
      int root=0;
      //array which will be filled with data by root process
      int sendbuf[]=null;
      sendbuf= new int[size];
         //creates data to be scattered
      if(rank==root){
         sendbuf[0] = 10;
         sendbuf[1] = 20;
         sendbuf[2] = 30;
         sendbuf[3] = 40;
         //print current process number
         System.out.print("Processor "+rank+" has data: ");
         for(int i = 0; i < size; i++){
            System.out.print(sendbuf[i]+" ");
            }
         System.out.println();
         }
      //collect data in recvbuf
      int recvbuf[] = new int[1];
      //following are the args of Scatter method
      //send, offset, chunk_count, chunk_data_type, recv, offset, chunk_count,
chunk_data_type,
      root_process_id
      MPI.COMM_WORLD.Scatter(sendbuf, 0, 1, MPI.INT, recvbuf, 0, 1, MPI.INT, root);
      System.out.println("Processor "+rank+" has data: "+recvbuf[0]);
      System.out.println("Processor "+rank+" is doubling the data");
      recvbuf[0]=recvbuf[0]*2;
      //following are the args of Gather method
      //Object sendbuf, int sendoffset, int sendcount, Datatype sendtype,
```

```
        //Object recvbuf, int recvoffset, int recvcount, Datatype recvtype,
        //int root)
        MPI.COMM_WORLD.Gather(recvbuf, 0, 1, MPI.INT, sendbuf, 0, 1, MPI.INT, root);
        //display the gathered result
        if(rank==root){
           System.out.println("Process 0 has data: ");
           for(int i=0;i<4;i++){
              System.out.print(sendbuf[i]+ " ");
              }
           }
        //Terminate MPI execution environment
        MPI.Finalize();
    }
}
```

Output :

```
kunal@ubuntu-22:~$ export MPJ_HOME=/home/samthube/Downloads/DS/Ass/mpj-v0_44
kunal@ubuntu-22:~$ cd /home/samthube/Downloads/DS/Ass
kunal@ubuntu-22:~/Downloads/DS/Ass$ ls
mpj-v0_44 Readme.odt ScatterGather.java
kunal@ubuntu-22:~/Downloads/DS/Ass$ javac -cp $MPJ_HOME/lib/mpj.jar
ScatterGather.java
kunal@ubuntu-22:~/Downloads/DS/Ass$ $MPJ_HOME/bin/mpjrun.sh -np 4 ScatterGather
MPJ Express (0.44) is started in the multicore configuration
Processor 0 has data: 10 20 30 40
Processor 1 has data: 20
Processor 1 is doubling the data
Processor 0 has data: 10
Processor 0 is doubling the data
Processor 3 has data: 40
Processor 3 is doubling the data
Processor 2 has data: 30
Processor 2 is doubling the data
Process 0 has data:
20 40 60 80
```

Assignment 5

Title : Clock Synchronization

Program :

Server.java

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

public class Server {
   public static void main(String[] args) {
      try {
         ServerSocket serverSocket = new ServerSocket(1234);
         System.out.println("Server started and listening on port 1234");
         while (true) {
            Socket clienSocket = serverSocket.accept();
            System.out.println("Client connected : 0 " +
clienSocket.getInetAddress().getHostAddress());
            Thread t = new Thread(new ClientHandler(clienSocket));
            t.start();

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

class ClientHandler implements Runnable {
   private Socket clientSocket;

   public ClientHandler(Socket socket) {
      this.clientSocket = socket;
   }

   @Override
   public void run() {
      try {
         PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);
         BufferedReader in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
         out.println(System.currentTimeMillis());
         in.close();
         out.close();
```

```
            clientSocket.close();

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

}
```

Client.java

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

public class Client {
    public static void main(String[] args) {
        try {
            Socket socket = new Socket("localhost", 1234);
            BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
            PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
            float serverTime = Float.parseFloat(in.readLine());
            float clientTime = System.currentTimeMillis();
            float timeDifference = serverTime - clientTime;

            System.out.println("Server TIme : " + serverTime);
            System.out.println("Client time : " + clientTime);
            System.out.println("Time Difference : " + timeDifference);

            in.close();
            out.close();
            socket.close();

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

Output :

```
Kunal E:\BEIT\Sem 2\LP V\performed\Assignment6> cd src
Kunal E:\BEIT\Sem 2\LP V\performed\Assignment6\src> javac Server.java
Kunal E:\BEIT\Sem 2\LP V\performed\Assignment6\src> javac Client.java
```

```
Kunal E:\BEIT\Sem 2\LP V\performed\Assignment6\src> java Server
Server started and listening on port 1234
Client connected : 0 127.0.0.1
Client connected : 0 127.0.0.1

Kunal E:\BEIT\Sem 2\LP V\performed\Assignment6\src> java Client
Server TIme : 1.7132069E12
Client time : 1.7132069E12
Time Difference : 0.0
Kunal E:\BEIT\Sem 2\LP V\performed\Assignment6\src> java Client
Server TIme : 1.7132069E12
Client time : 1.7132069E12
Time Difference : 0.0
```

Assignment 6

Title : Mutual Exclusion

Program :

Tokering.java

```
package org.met.ds;

import java.util.*;

class tokenring {
   public static void main(String args[]) throws Throwable {
      Scanner scan = new Scanner(System.in);
      System.out.println("Enter no of Nodes:");
      int n = scan.nextInt();
      int m = n - 1;
      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);
  }
}
```

Output :

```
Enter no of Nodes:
6
0123450
Enter sender:
2
Enter receiver:
5
Enter Data:
1
Token Passing0->1->2
Sender2Sending Data:1
Data1Forwarded By:3
Data1Forwarded By:4
Receiver5Received Data:1
Do you want to send again?? Enter 1 for yes and 0 for No:0
```

Assignment 7

Title : Election Algorithms

Program :

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) {
      if (state[up - 1]) {
         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);
    }
}
```

Output :

```
5 active process are:
Process up = p1 p2 p3 p4 p5
Process 5 is coordinator
.........
1 up a process.
2.down a process
3 send a message
4.Exit
2
bring down any process.
1
.........
1 up a process.
2.down a process
3 send a message
4.Exit
1
bring proces up
3
process3is already up
.........
1 up a process.
```

```
2.down a process
3 send a message
4.Exit
3
which process will send message
2
0K
.........
1 up a process.
2.down a process
3 send a message
4.Exit
1
bring proces up
4
process4is already up
.........
1 up a process.
2.down a process
3 send a message
4.Exit
4
```

Program :

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

}
```

Output :

```
Enter the number of process :
3
Enter the id of process :
1
Enter the id of process :
2
Enter the id of process :
3
[0] 1 [1] 2 [2] 3
process 3select as co-ordinator
1.election 2.quit
1
 Enter the Process number who initialsied election :
2
Process 3 send message to 1
Process 1 send message to 2
Process 2 send message to 3
process 3select as co-ordinator
```

1.election 2.quit
1
Enter the Process number who initialsied election :
1
Process 2 send message to 1
Process 1 send message to 2
process 2select as co-ordinator
1.election 2.quit
2
Program terminated ..