

Assignment 1

Code:

Name of student: Omkar Sukhdeo Gangarde

Class: BE

Division: 1

Roll No : 407A041

Assignment Number: 1

Program on: To develop any distributed application through implementing client-server communication programs based on java RMI.

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); } } }
```

AddServer.java :

```
import java.net.*;

import java.rmi.*;
```

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

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; } }
```

AddServerIntf.java :

```
import java.rmi.*;  
public interface AddServerIntf extends Remote {  
    double add(double d1, double d2) throws RemoteException;  
}
```

Output:

```
omkar@Omkar: ~/Desktop/Ass1
omkar@Omkar:~$ cd Desktop/Ass1
omkar@Omkar:~/Desktop/Ass1$ javac *.java
omkar@Omkar:~/Desktop/Ass1$ rmic AddServerImpl
Warning: generation and use of skeletons and static stubs for JRMP
is deprecated. Skeletons are unnecessary, and static stubs have
been superseded by dynamically generated stubs. Users are
encouraged to migrate away from using rmic to generate skeletons and static
stubs. See the documentation for java.rmi.server.UnicastRemoteObject.
omkar@Omkar:~/Desktop/Ass1$ rmiregistry
```

```
omkar@Omkar: ~/Desktop/Ass1
omkar@Omkar:~$ cd Desktop/Ass1
omkar@Omkar:~/Desktop/Ass1$ java AddServer
```

```
omkar@Omkar: ~/Desktop/Ass1
omkar@Omkar:~/Desktop/Ass1$ java AddClient 10.0.2.15 8 9
The first number is: 8
The second number is: 9
The sum is: 17.0
omkar@Omkar:~/Desktop/Ass1$ java AddClient 10.0.2.15 7 9
The first number is: 7
The second number is: 9
The sum is: 16.0
omkar@Omkar:~/Desktop/Ass1$ java AddClient 10.0.2.15 9 9
The first number is: 9
The second number is: 9
The sum is: 18.0
omkar@Omkar:~/Desktop/Ass1$
```

Assignment 2

Code:

Name of student: Omkar Sukhdeo Gangarde

Class: BE

Division: 1

Roll No : 407A041

Assignment Number: 2

Program on: To develop any distributed application with CORBA program using JAVA IDL.

ReverseClient.java :

```
// Client
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 ReverseImpl=null;
try
{
// initialize the ORB object request broker
org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args,null);
org.omg.CORBA.Object objRef = orb.resolve_initial_references("NameService");
NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);
String name = "Reverse";
// narrow converts generic object into string type
```

```

        ReverseImpl = 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= ReverseImpl.reverse_string(str);
        System.out.println(tempStr);
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
}

```

ReverseImpl.java :

```

import ReverseModule.ReversePOA;
import java.lang.String;
class ReverseImpl extends ReversePOA
{
    ReverseImpl()
    {
        super();
        System.out.println("Reverse Object Created");
    }
    public String reverse_string(String name)
    {
        StringBuffer str=new StringBuffer(name);
        str.reverse();
        return ("Server Send "+str);
    }
}

```

```
}
```

ReverseModule.idl :

```
module ReverseModule
{
    interface Reverse
    {
        string reverse_string(in string 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 BOA/POA
            POA rootPOA = POAHelper.narrow(orb.resolve_initial_references("RootPOA"));
            rootPOA.the_POAManager().activate();
            // creating the object
```

```

ReverseImpl rvr = new ReverseImpl();
// get the object reference from the servant class
org.omg.CORBA.Object ref = rootPOA.servant_to_reference(rvr);
System.out.println("Step1");
Reverse h_ref = ReverseModule.ReverseHelper.narrow(ref);
System.out.println("Step2");
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 :

```
omkar@Omkar: ~/Desktop/Ass2
omkar@Omkar:~$ cd Desktop/Ass2
omkar@Omkar:~/Desktop/Ass2$ idlj -fall ReverseModule.idl
omkar@Omkar:~/Desktop/Ass2$ javac *.java ReverseModule/*.java
ReverseModule/_ReverseStub.java:46: warning: IORCheckImpl is internal proprietary API and may be removed in a future release
    com.sun.corba.se.impl.orbutil.IORCheckImpl.check(str, "ReverseModule._ReverseStub");
                                   ^
Note: ReverseModule/ReversePOA.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
1 warning
omkar@Omkar:~/Desktop/Ass2$ orbd -ORBInitialPort 1050&
[1] 5309
omkar@Omkar:~/Desktop/Ass2$ java ReverseServer -ORBInitialPort 1050& -ORBInitialHost localhost&
[2] 5338
[3] 5339
omkar@Omkar:~/Desktop/Ass2$ -ORBInitialHost: command not found
Reverse Object Created
Step1
Step2
Step3
Step4
Reverse Server reading and waiting....
```

```
omkar@Omkar: ~/Desktop/Ass2
omkar@Omkar:~$ cd Desktop/Ass2
omkar@Omkar:~/Desktop/Ass2$ java ReverseClient -ORBInitialPort 1050 -ORBInitialHost localhost
Enter String=
hi this is corba application.
Server Send .noitacilppa abroc si siht ih
omkar@Omkar:~/Desktop/Ass2$
```


Assignment 3

Code :

Name of student: Omkar Sukhdeo Gangarde

Class: BE

Division: 1

Roll No : 407A041

Assignment Number: 3

Program on: To develop a distributed system, to find sum of N elements in an array using MPI or OpenMP.

ArrSum.java :

```
import mpi.MPI;
import java.util.Scanner;
import mpi.*;

public class ArrSum {
    public static void main(String[] args) throws Exception{
        MPI.Init(args);

        int rank = MPI.COMM_WORLD.Rank();
        int size = MPI.COMM_WORLD.Size();

        int unitSize = 5;
        int root = 0;
        int send_buffer[] = null;
        // 1 process is expected to handle 4 elements
        send_buffer = new int [unitSize * size];
        int receive_buffer[] = new int [unitSize];
        int new_receive_buffer[] = new int [size];
```

```

// Set data for distribution
if(rank == root) {
    int total_elements = unitsize * size;

    System.out.println("Enter " + total_elements + " elements");
    for(int i = 0; i < total_elements; i++) {
        System.out.println("Element " + i + "\t = " + i);
        send_buffer[i] = i;
    }
}

// Scatter data to processes
MPI.COMM_WORLD.Scatter(
    send_buffer,
    0,
    unitsize,
    MPI.INT,
    recieve_buffer,
    0,
    unitsize,
    MPI.INT,
    root
);

// Calculate sum at non root processes
// Store result in first index of array
for(int i = 1; i < unitsize; i++) {
    recieve_buffer[0] += recieve_buffer[i];
}

System.out.println(

```

```

        "Intermediate sum at process " + rank + " is " + recieve_buffer[0]
    );

    // Gather data from processes
    MPI.COMM_WORLD.Gather(
        recieve_buffer,
        0,
        1,
        MPI.INT,
        new_recieve_buffer,
        0,
        1,
        MPI.INT,
        root
    );

    // Aggregate output from all non root processes
    if(rank == root) {
        int total_sum = 0;
        for(int i = 0; i < size; i++) {
            total_sum += new_recieve_buffer[i];
        }
        System.out.println("Final sum : " + total_sum);
    }
    MPI.Finalize();
}
}

```

Output :

```
omkar@Omkar: ~/Desktop/Ass3
omkar@Omkar:~$ cd Desktop/Ass3
omkar@Omkar:~/Desktop/Ass3$ export MPJ_HOME=/home/omkar/Downloads/mpj-v0_44
omkar@Omkar:~/Desktop/Ass3$ export PATH=$MPJ_HOME/bin:$PATH
omkar@Omkar:~/Desktop/Ass3$ javac -cp .:$MPJ_HOME/lib/mpj.jar ArrSum.java
omkar@Omkar:~/Desktop/Ass3$ mpjrun.sh -np 4 ArrSum
MPJ Express (0.44) is started in the multicore configuration
Enter 20 elements
Element 0      = 0
Element 1      = 1
Element 2      = 2
Element 3      = 3
Element 4      = 4
Element 5      = 5
Element 6      = 6
Element 7      = 7
Element 8      = 8
Element 9      = 9
Element 10     = 10
Element 11     = 11
Element 12     = 12
Element 13     = 13
Element 14     = 14
Element 15     = 15
Element 16     = 16
Element 17     = 17
Element 18     = 18
Element 19     = 19
Intermediate sum at process 0 is 10
Intermediate sum at process 3 is 85
Intermediate sum at process 2 is 60
Intermediate sum at process 1 is 35
Final sum : 190
omkar@Omkar:~/Desktop/Ass3$
```

Assignment 4

Code :

Name of student: Omkar Sukhdeo Gangarde

Class: BE

Division: 1

Roll No : 407A041

Assignment Number: 4

Program on: To implement Berkeley algorithm for clock synchronization.

Berkeley.java :

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

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

```
import java.util.*;
```

```
public class Berkeley {
```

```
    // Define the port number that will be used for communication
```

```
    private static final int PORT = 9876;
```

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

```
        // Create a server socket to listen for incoming messages
```

```
        ServerSocket serverSocket = new ServerSocket(PORT);
```

```
        // Create a list to store the time differences for each node
```

```
        List<Long> timeDiffs = new ArrayList<Long>();
```

```

// Create a new thread to handle the time requests from nodes
Thread timeServerThread = new Thread(new Runnable() {
    public void run() {
        while (true) {
            try {
                // Wait for a node to connect and request the current time
                Socket clientSocket = serverSocket.accept();

                ObjectInputStream in = new
                ObjectInputStream(clientSocket.getInputStream());

                // Read the current time from the node's request
                Date clientTime = (Date) in.readObject();

                // Send the current time to the node as a response
                ObjectOutputStream out = new
                ObjectOutputStream(clientSocket.getOutputStream());
                out.writeObject(new Date());

                // Calculate the time difference between the server and the node
                long timeDiff = (new Date().getTime() - clientTime.getTime()) / 2;
                timeDiffs.add(timeDiff);

                // Close the input/output streams and the socket
                in.close();
                out.close();
                clientSocket.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
}

```

```

});

timeServerThread.start();

// Create a new thread to periodically send time requests to the server
Thread timeClientThread = new Thread(new Runnable() {
    public void run() {
        while (true) {
            try {
                // Connect to the server and send a time request
                Socket socket = new Socket("localhost", PORT);

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

                out.writeObject(new Date());

                // Read the current time from the server's response
                ObjectInputStream in = new ObjectInputStream(socket.getInputStream());
                Date serverTime = (Date) in.readObject();

                // Calculate the time difference between the node and the server
                long timeDiff = (serverTime.getTime() - new Date().getTime()) / 2;
                timeDiffs.add(timeDiff);

                // Close the input/output streams and the socket
                in.close();
                out.close();
                socket.close();

                // Wait for a short period of time before sending the next time request
                Thread.sleep(1000);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
});

```

```

        }
    }
}

});

timeClientThread.start();

// Wait for a sufficient number of time differences to be recorded
Thread.sleep(10000);

// Compute the average time difference and adjust the node's clock
long sumTimeDiff = 0;
for (Long timeDiff : timeDiffs) {
    sumTimeDiff += timeDiff;
}

long avgTimeDiff = sumTimeDiff / timeDiffs.size();
System.out.println("Average time difference: " + avgTimeDiff);

// Adjust the node's clock by adding the average time difference
Calendar calendar = Calendar.getInstance();
calendar.setTime(new Date());
calendar.add(Calendar.MILLISECOND, (int) avgTimeDiff);
System.out.println("Adjusted time: " + calendar.getTime());
}
}

```


Output:

```
omkar@Omkar: ~/Desktop/Ass4  
omkar@Omkar:~$ cd Desktop/Ass4  
omkar@Omkar:~/Desktop/Ass4$ javac Berkeley.java  
omkar@Omkar:~/Desktop/Ass4$ java Berkeley  
Average time difference: 28  
Adjusted time: Sun Apr 21 14:56:34 IST 2024  
█
```

Assignment 5

Code:

Name of student: Omkar Sukhdeo Gangarde

Class: BE

Division: 1

Roll No : 407A041

Assignment Number: 5

Program on: To implement token ring based mutual exclusion algorithm.

Tring.java :

```
import java.util.Scanner;

class Tring {

    public static void main(String args[]) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the number of nodes: ");
        int n = sc.nextInt();

        // Decides the number of nodes forming the ring
        int token = 0;

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

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

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

```

try {
    while (true) {
        System.out.print("Enter sender: ");
        int s = sc.nextInt();
        System.out.print("Enter receiver: ");
        int r = sc.nextInt();
        System.out.print("Enter Data: ");
        String d = sc.next();

        System.out.print("Token passing:");
        //current token not equal to sender, increment i by 1 and j by j+1%n

        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: " + d);

        // start forwarding from node after sender until it becomes equal to receiver and
        increment by i+1%n
        for (int i = (s + 1) % n; i != r; i = (i + 1) % n) {
            System.out.println("Data " + d + " forwarded by " + i);
        }
        System.out.println("Receiver " + r + " received data: " + d);
        token = s;
    }
} catch (Exception e) {
    System.out.println("Error occurred: " + e.getMessage());
}
}
}

```

Output :

```
omkar@Omkar: ~/Desktop/Ass5
omkar@Omkar:~$ cd Desktop/Ass5
omkar@Omkar:~/Desktop/Ass5$ javac TokenRing.java
omkar@Omkar:~/Desktop/Ass5$ java TokenRing
Enter the number of nodes: 10
0 1 2 3 4 5 6 7 8 9 0
Enter sender: 3
Enter receiver: 7
Enter Data: 80
Token passing: 0-> 1-> 2-> 3
Sender 3 sending data: 80
Data 80 forwarded by 4
Data 80 forwarded by 5
Data 80 forwarded by 6
Receiver 7 received data: 80
Enter sender: 1
Enter receiver: 4
Enter Data: Hello
Token passing: 3-> 4-> 5-> 6-> 7-> 8-> 9-> 0-> 1
Sender 1 sending data: Hello
Data Hello forwarded by 2
Data Hello forwarded by 3
Receiver 4 received data: Hello
Enter sender:
```

Assignment 6

Code :

Name of student: Omkar Sukhdeo Gangarde

Class: BE

Division: 1

Roll No : 407A041

Assignment Number: 6

Program on: To implement Bully and Ring algorithm for leader election.

BullyAlgorithm.java:

```
import java.util.Scanner;
```

```
public class BullyAlgorithm {
```

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

```
    static int coordinator = 5;
```

```
    public static void up(int up) {
```

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

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

```
        } else {
```

```
            state[up - 1] = true;
```

```
            System.out.println("Process " + up + " is up.");
```

```
            // If the newly up process has a higher ID than the current coordinator,
```

```
            // initiate an election
```

```

    if (up > coordinator) {
        System.out.println("Process " + up + " initiates an election.");
        election(up);
    }
}
}

```

```

public static void down(int down) {
    if (!state[down - 1]) {
        System.out.println("Process " + down + " is already down.");
    } else {
        state[down - 1] = false;
        System.out.println("Process " + down + " is down.");
        if (down == coordinator) {
            System.out.println("Coordinator (Process " + down + ") is down.");
            election(down);
        }
    }
}
}

```

```

public static void mess(int mess) {
    if (!state[mess - 1]) {
        System.out.println("Process " + mess + " is down.");
    } else {
        if (mess == coordinator) {
            System.out.println("Coordinator (Process " + coordinator + ") received the message: OK");
        } else {
            System.out.println("Process " + mess + " sends a message.");
        }
    }
}
}

```

```
}
```

```
public static void election(int initiator) {  
    System.out.println("Election initiated by Process " + initiator);  
    for (int i = initiator + 1; i <= 5; i++) {  
        if (state[i - 1]) {  
            System.out.println("Election message sent from Process " + initiator + " to Process  
" + i);  
        }  
    }  
    // If no higher priority process responds, declare the initiator as the new coordinator  
    coordinator = initiator;  
    System.out.println("Process " + coordinator + " becomes the new coordinator.");  
    System.out.println("Coordinator message sent from Process " + coordinator + " to all.");  
}
```

```
public static void main(String[] args) {  
    int choice;  
    Scanner sc = new Scanner(System.in);  
    for (int i = 0; i < 5; ++i) {  
        state[i] = true;  
    }  
    System.out.println("5 active processes 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.");  
    } while (true);  
}
```

```
choice = sc.nextInt();
switch (choice) {
    case 1: {
        System.out.println("Bring process up:");
        int up = sc.nextInt();
        if (up > 5) {
            System.out.println("Invalid process number.");
            break;
        }
        up(up);
        break;
    }
    case 2: {
        System.out.println("Bring down any process:");
        int down = sc.nextInt();
        if (down > 5) {
            System.out.println("Invalid process number.");
            break;
        }
        down(down);
        break;
    }
    case 3: {
        System.out.println("Which process will send message:");
        int mess = sc.nextInt();
        if (mess > 5) {
            System.out.println("Invalid process number.");
            break;
        }
        mess(mess);
    }
}
```



```
        break;
    }
}
} while (choice != 4);
}
}
```

Output :

```
omkar@Omkar: ~$ cd Desktop/Ass6
omkar@Omkar:~/Desktop/Ass6$ javac BullyAlgorithm.java RingAlgorithm.java
omkar@Omkar:~/Desktop/Ass6$ java BullyAlgorithm
5 active processes 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:
2
Process 2 is down.
.....
1. Up a process.
2. Down a process.
3. Send a message.
4. Exit.
1
Bring process up:
3
Process 3 is already up.
.....
1. Up a process.
2. Down a process.
3. Send a message.
4. Exit.
3
Which process will send message:
1
Process 1 sends a message.
.....
1. Up a process.
```

RingAlgorithm.java:

```
import java.util.Scanner;
```

```
public class RingAlgorithm {
```

```
    public static void main(String[] args) {
```

```
        int i, j;
```

```
        Scanner scanner = new Scanner(System.in);
```

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

```
        int numberOfProcesses = scanner.nextInt();
```

```
        Process[] processes = new Process[numberOfProcesses];
```

```
        // Object initialization
```

```
        for (i = 0; i < processes.length; i++)
```

```
            processes[i] = new Process();
```

```
        // Getting input from users
```

```
        for (i = 0; i < numberOfProcesses; i++) {
```

```
            processes[i].index = i;
```

```
            System.out.println("Enter the ID of process " + (i + 1) + ": ");
```

```
            processes[i].id = scanner.nextInt();
```

```
            processes[i].state = "active";
```

```
            processes[i].hasSentMessage = false;
```

```
        }
```

```
        // Sorting the processes based on their IDs
```

```

for (i = 0; i < numberOfProcesses - 1; i++) {
    for (j = 0; j < numberOfProcesses - i - 1; j++) {
        if (processes[j].id > processes[j + 1].id) {
            Process temp = processes[j];
            processes[j] = processes[j + 1];
            processes[j + 1] = temp;
        }
    }
}

```

```

System.out.println("Processes in sorted order:");
for (i = 0; i < numberOfProcesses; i++) {
    System.out.println "[" + processes[i].index + " ] " + processes[i].id);
}

```

```

// Initiating coordinator selection
int initiatorIndex;
while (true) {
    System.out.println("\n1. Initiate Election\n2. Quit");
    int choice = scanner.nextInt();

    switch (choice) {
        case 1:
            System.out.println("Enter the index of the process initiating the election: ");
            initiatorIndex = scanner.nextInt();
            initiateElection(processes, initiatorIndex, numberOfProcesses);
            break;
        case 2:
            System.out.println("Program terminated.");
            return;
    }
}

```

```

        default:

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

        }

    }

}

public static void initiateElection(Process[] processes, int initiatorIndex, int
numberOfProcesses) {

    System.out.println("Election initiated by Process " + processes[initiatorIndex].id);
    processes[initiatorIndex].hasSentMessage = true;

    int tempIndex = initiatorIndex;
    int nextIndex = (initiatorIndex + 1) % numberOfProcesses;
    while (nextIndex != initiatorIndex) {

        if (processes[nextIndex].state.equals("active") &&
!processes[nextIndex].hasSentMessage) {

            System.out.println("Process " + processes[initiatorIndex].id + " sends message to
Process " +
processes[nextIndex].id);

            processes[nextIndex].hasSentMessage = true;
            tempIndex = nextIndex;
        }

        nextIndex = (nextIndex + 1) % numberOfProcesses;
    }

    System.out.println("Process " + processes[tempIndex].id + " sends message to Process "
+
processes[initiatorIndex].id);

    System.out.println("Process " + processes[initiatorIndex].id + " becomes the
coordinator.");

    processes[initiatorIndex].state = "inactive";

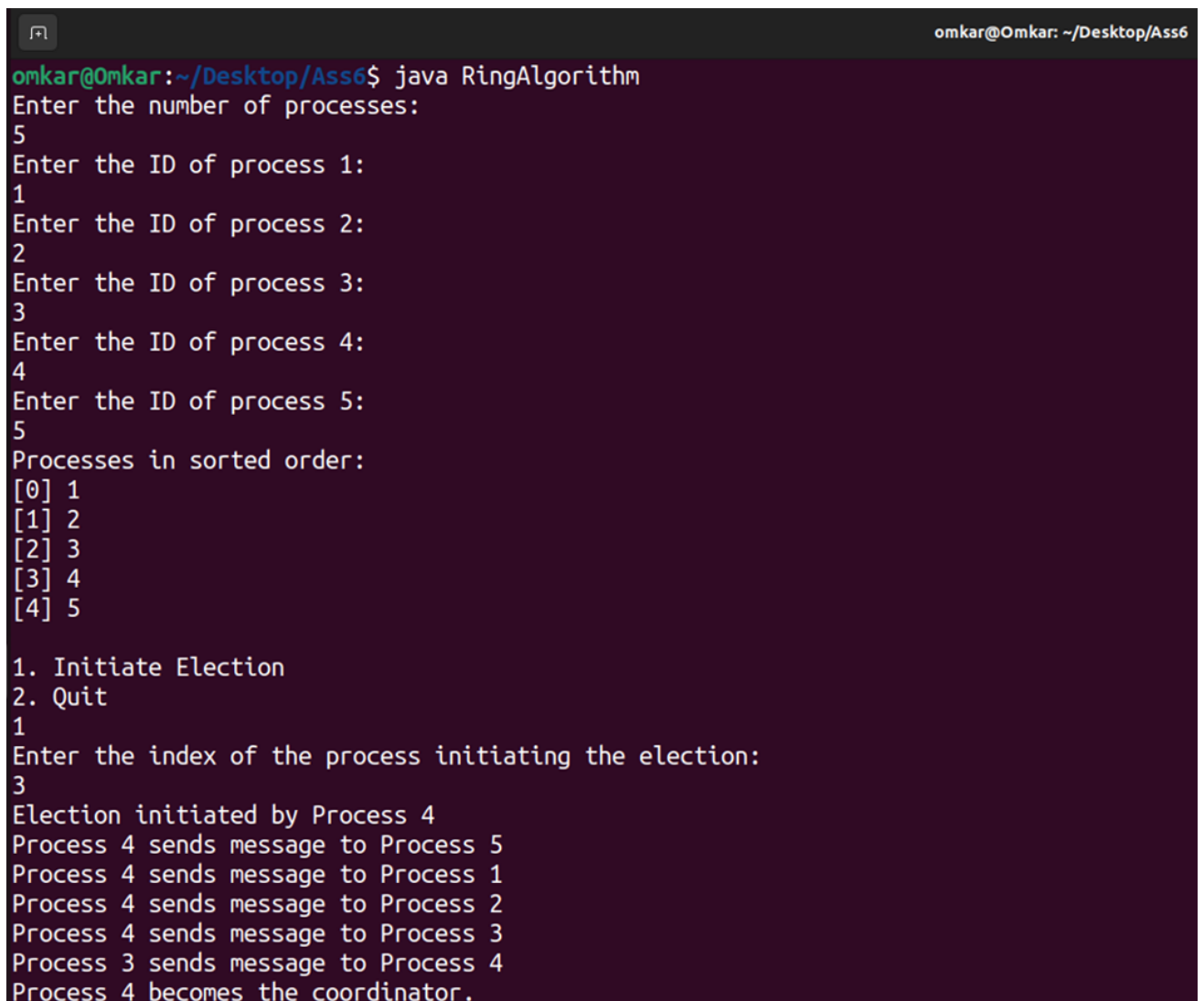
}

```

```
}
```

```
class Process {  
    public int index;  
    public int id;  
    // Index of the process  
    // ID of the process  
    public boolean hasSentMessage; // Flag to indicate whether the process has sent a message  
    public String state;  
}
```

Output:



```
omkar@Omkar: ~/Desktop/Ass6  
omkar@Omkar:~/Desktop/Ass6$ java RingAlgorithm  
Enter the number of processes:  
5  
Enter the ID of process 1:  
1  
Enter the ID of process 2:  
2  
Enter the ID of process 3:  
3  
Enter the ID of process 4:  
4  
Enter the ID of process 5:  
5  
Processes in sorted order:  
[0] 1  
[1] 2  
[2] 3  
[3] 4  
[4] 5  
  
1. Initiate Election  
2. Quit  
1  
Enter the index of the process initiating the election:  
3  
Election initiated by Process 4  
Process 4 sends message to Process 5  
Process 4 sends message to Process 1  
Process 4 sends message to Process 2  
Process 4 sends message to Process 3  
Process 3 sends message to Process 4  
Process 4 becomes the coordinator.
```

Assignment 7

Code:

Name of student: Omkar Sukhdeo Gangarde

Class: BE

Division: 1

Roll No : 407A041

Assignment Number: 7

Program on: To create a simple web service and write any distributed application to consume the web service.

Calculator.java :

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this
 * license
 * Click nbfs://nbhost/SystemFileSystem/Templates/WebServices/WebService.java to edit
 * this template
 */
package com.unique;
import javax.jws.WebService;
import javax.jws.WebMethod;
import javax.jws.WebParam;
@WebService(serviceName = "Calculator")
public class Calculator {
    @WebMethod(operationName = "getNumber")
    public int getNumber(@WebParam(name = "num1") int num1, @WebParam(name =
    "num2") int num2) {
        int sum=num1+num2;
        return sum;
    }
}
```

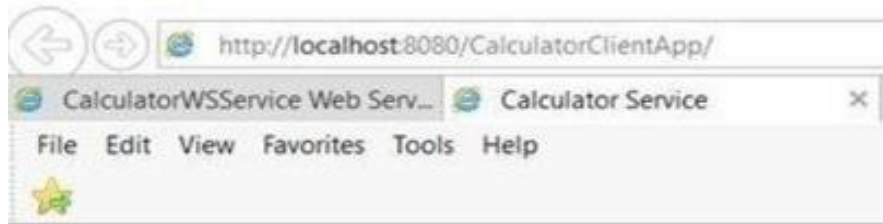
Calculator.java (Client side):

```
/*  
  
* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this  
license  
  
* Click nbfs://nbhost/SystemFileSystem/Templates/WebServices/WebService.java to edit  
this template  
  
*/  
  
package com.unique;  
  
import javax.jws.WebService;  
  
import javax.jws.WebMethod;  
  
import javax.jws.WebParam;  
  
@WebService(serviceName = "Calculator")  
  
public class Calculator {  
  
    @WebMethod(operationName = "getNumber")  
  
    public int getNumber(@WebParam(name = "num1") int num1, @WebParam(name =  
"num2") int num2) {  
  
        int sum=num1+num2;  
  
        return sum;  
  
    }  
  
}
```


Output:

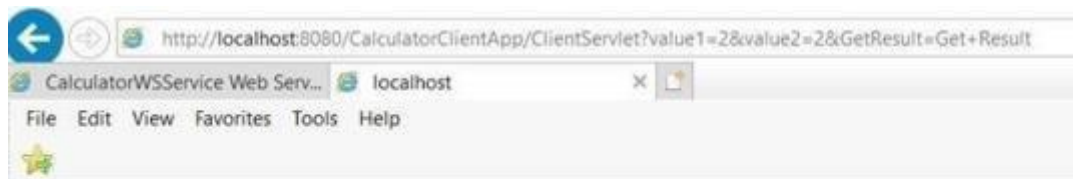
Compiling and executing the solution:

Right Click on the Project and Choose Run



Calculator Service

2 + 2 =



Servlet ClientServlet at /CalculatorClientApp

Result: 2 + 2 = 4