Lab Practice - V (Final Code)

1. Implement multi-threaded client/server Process communication using RMI.

MyClient.java

```
import java.util.*;
public class MyClient {
public static void main(String[] args) {
int a,b;
try {
MyInterface obj = (MyInterface)java.rmi.Naming.lookup("//localhost/MyRemoteClass");
Scanner sc= new Scanner(System.in); //System.in is a standard input stream.
System.out.print("Enter first number- ");
a= sc.nextInt();
System.out.print("Enter second number- ");
b= sc.nextInt();
System.out.println("The Addition is= "+obj.addition(a,b));
System.out.println("The Multiplication is= "+obj.mult(a,b));
} catch (Exception e) {
e.printStackTrace();
}
}
}
```

MyServer.java

```
public class MyServer {
public static void main(String[] args) {
try {
MyInterface obj = new MyRemoteClass();
java.rmi.registry.LocateRegistry.createRegistry(1099);
java.rmi.Naming.rebind("//localhost/MyRemoteClass", obj);
System.out.println("MyRemoteClass bound in registry");
}
catch (Exception e) {
System.err.println("MyRemoteClass exception:");
e.printStackTrace();
}
}
}
```

MyInterface.java

```
public interface MyInterface extends java.rmi.Remote
{
public int addition(int x,int y) throws java.rmi.RemoteException;
public int mult(int x,int y) throws java.rmi.RemoteException;
}
```

MyRemoteClass.java

```
public class MyRemoteClass extends java.rmi.server.UnicastRemoteObject implements
MyInterface
{
public MyRemoteClass() throws java.rmi.RemoteException
{
super();
}

public int addition(int x,int y)
{
return x+y;
}

public int mult(int x,int y)
{
return x*y;
}
}
```

2. Develop any distributed application using CORBA to demonstrate object brokering. (Calculator or String operations).

Addition.idl

```
module AdditionApp
{
  interface Addition
  {
    long add(in long a,in long b);
    oneway void shutdown();
    };
};
```

StartClient.java

```
/**
 *
 * @author imed
 */
import AdditionApp.*;

import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContextPackage.*;
import org.omg.CORBA.*;
import java.io.*;
import java.util.*;

public class StartClient {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
      try {
            ORB orb = ORB.init(args, null);
            org.omg.CORBA.Object objRef =   orb.resolve_initial_references("NameService");
            NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);
            Addition addobj = (Addition) AdditionHelper.narrow(ncRef.resolve_str("ABC"));

            Scanner c=new Scanner(System.in);
            System.out.println("Welcome to the addition system:");
                    for(;;){
                      System.out.println("Enter a:");
                      String aa = c.nextLine();
                      System.out.println("Enter b:");
                      String bb = c.nextLine();
                      int a=Integer.parseInt(aa);
                      int b=Integer.parseInt(bb);
                      int r=addobj.add(a,b);
                      System.out.println("The result for addition is : "+r);
                      System.out.println("--------------------------------------");
            }
       }
       catch (Exception e) {
          System.out.println("Hello Client exception: " + e);
          e.printStackTrace();
       }
```

```
        }

    }
```

StartServer.java

```
    import AdditionApp.*;
    import org.omg.CosNaming.*;
    import org.omg.CosNaming.NamingContextPackage.*;
    import org.omg.CORBA.*;
    import org.omg.PortableServer.*;
    import org.omg.PortableServer.POA;
    import java.util.Properties;

class AdditionImpl extends AdditionPOA {
  private ORB orb;

  public void setORB(ORB orb_val) {
    orb = orb_val;
  }

  // implement add() method
  public int add(int a, int b) {
    int r=a+b;
    return r;
  }

  // implement shutdown() method
  public void shutdown() {
    orb.shutdown(false);
  }
}



/*---------------------------------*/


    public class StartServer {

      public static void main(String args[]) {
        try{
          // create and initialize the ORB //// get reference to rootpoa &amp; activate the
POAManager
```

```
        ORB orb = ORB.init(args, null);
        POA rootpoa = POAHelper.narrow(orb.resolve_initial_references("RootPOA"));
        rootpoa.the_POAManager().activate();

        // create servant and register it with the ORB
        AdditionImpl addobj = new AdditionImpl();
        addobj.setORB(orb);

        // get object reference from the servant
        org.omg.CORBA.Object ref = rootpoa.servant_to_reference(addobj);
        Addition href = AdditionHelper.narrow(ref);

        org.omg.CORBA.Object objRef =  orb.resolve_initial_references("NameService");
        NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);

        NameComponent path[] = ncRef.to_name( "ABC" );
        ncRef.rebind(path, href);

        System.out.println("Addition Server ready and waiting ...");

        // wait for invocations from clients
        for (;;){
          orb.run();
        }
       }

        catch (Exception e) {
         System.err.println("ERROR: " + e);
         e.printStackTrace(System.out);
        }

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

      }
     }
```

Client Output:

```
sl1-14@sl114-Veriton-M200-H81:~$ cd CORBA_Addition-1/
sl1-14@sl114-Veriton-M200-H81:~/CORBA_Addition-1$ java StartClient -ORBInitialPort 1050
-ORBInitialHost localhost
Welcome to the addition system:
Enter a:
12
Enter b:
```

```
25
The result for addition is : 37
----------------------------------
Enter a:
12
Enter b:
35
The result for addition is : 47
----------------------------------
Enter a:
10
Enter b:
2
The result for addition is : 12
----------------------------------
Enter a:
```

Server Output:

```
sl1-14@sl114-Veriton-M200-H81:~$ cd CORBA_Addition-1/
sl1-14@sl114-Veriton-M200-H81:~/CORBA_Addition-1$ idlj -fall  Addition.idl
sl1-14@sl114-Veriton-M200-H81:~/CORBA_Addition-1$ javac *.java AdditionApp/*.java
Note: AdditionApp/AdditionPOA.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
sl1-14@sl114-Veriton-M200-H81:~/CORBA_Addition-1$ orbd -ORBInitialPort 1050&
[1] 4301
sl1-14@sl114-Veriton-M200-H81:~/CORBA_Addition-1$ java StartServer -ORBInitialPort 1050
-ORBInitialHost localhost&
[2] 4319
sl1-14@sl114-Veriton-M200-H81:~/CORBA_Addition-1$ Addition Server ready and waiting ...
```

3. Develop a distributed system, to find sum of N elements in an array by distributing N/n elements to n number of processors MPI or OpenMP. Demonstrate by displaying the intermediate sums calculated at different processors.

SimpleMPIProgram.java

```
import mpi.*;

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

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

        if (size < 2) {
            System.err.println("This program requires at least two processes to run.");
            MPI.Finalize();
            System.exit(1);
        }

        if (rank == 0) {
            // Process 0 sends a message to process 1
            String message = "Hello from process 0!";
            MPI.COMM_WORLD.Send(message.toCharArray(), 0, message.length(), MPI.CHAR, 1,
0);
            System.out.println("Process 0 sent: " + message);
        } else if (rank == 1) {
            // Process 1 receives the message from process 0
            char[] receivedMessage = new char[20]; // Assuming max message length is 20
            Status status = MPI.COMM_WORLD.Recv(receivedMessage, 0, 20, MPI.CHAR, 0, 0);
            System.out.println("Process 1 received: " + new String(receivedMessage) + " from
process " + status.source);
        }

        MPI.Finalize();
    }
}
```

OUTPUT:

```
mpijavac SimpleMPIProgram.java
mpirun -np 2 java SimpleMPIProgram
```

4. Implement Berkeley algorithm for clock synchronization.

Berkeley.py

```
import time
import random

# Function to calculate the clock offset
def calculate_offset(remotes):
    local_time = time.time()
    offsets = [remote_time - local_time for remote_time in remotes]
    average_offset = sum(offsets) / len(offsets)
    return average_offset

# Function to synchronize clocks using the Berkeley algorithm
def synchronize_clocks():
    num_peers = int(input("Enter the number of peers: "))
    local_time = time.time()

    # Simulate remote clocks with random offsets
    remote_times = [local_time + random.uniform(-1, 1) for _ in range(num_peers)]

    print("Local time:", local_time)
    print("Remote times:", remote_times)

    # Calculate the clock offset
    offset = calculate_offset(remote_times)

    # Adjust local clock
    adjusted_time = local_time + offset

    print("Adjusted local time:", adjusted_time)

# Execute the clock synchronization
synchronize_clocks()
```

OUTPUT:

```
Enter the number of peers:  4

Local time: 1711953848.3981674
Remote times: [1711953849.3690608, 1711953848.8295243, 1711953848.8412962,
1711953847.4755003]
Adjusted local time: 1711953848.6288455
```

5. Implement token ring based mutual exclusion algorithm.
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 );
    }
}
```

OUTPUT:

```
javac tokenring.java
java tokenring
```

6. Implement Bully and Ring algorithm for leader election.

LeaderElection.java

```
import java.util.ArrayList;
import java.util.List;

// Class representing a node in the distributed system
class Node {
    private int id;
    private boolean isCoordinator;

    public Node(int id) {
        this.id = id;
    }

    public int getId() {
        return id;
    }

    public boolean isCoordinator() {
        return isCoordinator;
    }

    public void setCoordinator(boolean coordinator) {
        isCoordinator = coordinator;
    }

    // Method to initiate election
    public void initiateElection(List<Node> nodes) {
        for (Node node : nodes) {
            if (node.getId() > this.id) {
                System.out.println("Node " + this.id + " sends election message to Node " +
node.getId());
                node.startElection(nodes);
            }
        }
        this.setCoordinator(true);
        System.out.println("Node " + this.id + " becomes the coordinator.");
    }

    // Method to start election
    public void startElection(List<Node> nodes) {
        for (Node node : nodes) {
            if (node.getId() > this.id) {
                System.out.println("Node " + this.id + " sends election message to Node " +
node.getId());
                node.startElection(nodes);
            }
        }
        this.setCoordinator(true);
        System.out.println("Node " + this.id + " becomes the coordinator.");
    }
}

public class LeaderElection {
    public static void main(String[] args) {
        // Create nodes
        List<Node> nodes = new ArrayList<>();
        for (int i = 1; i <= 5; i++) {
            nodes.add(new Node(i));
        }

        // Simulate Bully Algorithm
        System.out.println("Bully Algorithm:");
        // Node with highest ID starts the election
        nodes.get(nodes.size() - 1).initiateElection(nodes);

        // Simulate Ring Algorithm
        System.out.println("\nRing Algorithm:");
        // Node with lowest ID starts the election
        nodes.get(0).startElection(nodes);
    }
}
```

OUTPUT:

```
javac LeaderElection.java
java LeaderElection
```

7. Create a simple web service and write any distributed application to consume the web service.

Client Code

client.py

```
import requests

data = {
    'a': 10,
    'b': 20
}

response = requests.post('http://localhost:5000/add', json=data)
result = response.json()
print(result['result'])
```

Server Code

server.py

```
from flask import Flask, jsonify, request

app = Flask(__name__)

@app.route('/add', methods=['POST'])
def add():
    data = request.get_json()
    a = data['a']
    b = data['b']
    result = a + b
    answer = jsonify({'result': result})
    return answer

if __name__ == '__main__':
    app.run()
```