

# Smt.Chandibai Himathmal Mansukhani College

## Contents

USCSP301 – USCS303: Operating System (OS) Practical – 04 .....	2
Practical – 04: Process Communication.....	2
Practical Date: 7 <sup>th</sup> August 2021 .....	2
Practical Aim: Producer-Consumer Problem, RMI .....	2
➤ Process Communication.....	2
➤ Producer-Consumer Solution using Shared Memory.....	3
A) QUESTION – 01 : - .....	3
B) SOURCE CODE .....	3
File Name: P4_PC_SM_Buffer_YashParab.java .....	3
File Name: P4_PC_SM_BufferImpl_YashParab.java .....	4
File Name: P4_PC_SM_Yash Parab.java .....	6
C) OUTPUT: - .....	7
➤ USING MESSAGE PASSING .....	7
➤ Producer-Consumer Solution using Shared Memory.....	8
A) QUESTION - 02: - .....	8
B) SOURCE CODE .....	8
File Name: P4_PC_MP_Channel_YashParabjava.....	8
FileName:P4_PC_MP_MessageQueue_Yash Parab.java .....	9
File Name: P4_PC_MP_YashParab.java .....	10
C) OUTPUT: - .....	11
Remote Procudure Calls.....	12
➤ Remote Method Invocation (RMI)Calculator. ....	12
A) QUESTION-03: - .....	13
C) SOURCE CODE.....	15
File Name: P4_RMI_CalcServerIntf_YashParab java .....	15
File Name: P4_RMI_CalcServerImpl_YashParab.java.....	15
File Name: P4_RMI_CalcServer_YashParab.java .....	17
File Name: P4_RMI_CalcClient_YashParab.java .....	18
D) OUTPUT: .....	20

**USCSP301 – USCS303: Operating System (OS) Practical  
– 04**

**Practical – 04: Process Communication**

**Practical Date: 7<sup>th</sup> August 2021**

**Practical Aim: Producer-Consumer Problem, RMI**

➤ **Process Communication**

- Process often need to communicate with each other.
- This is complicated in distributed systems by the fact that the communicating processes may be on different workstations.
- Inter-process communication provides a means for processes to cooperate and compete.
- **Message passing and remote procedure calls** are the most common methods of inter-process communication in distributed systems.
- A less frequently used but no less valuable method is distributed shared memory.

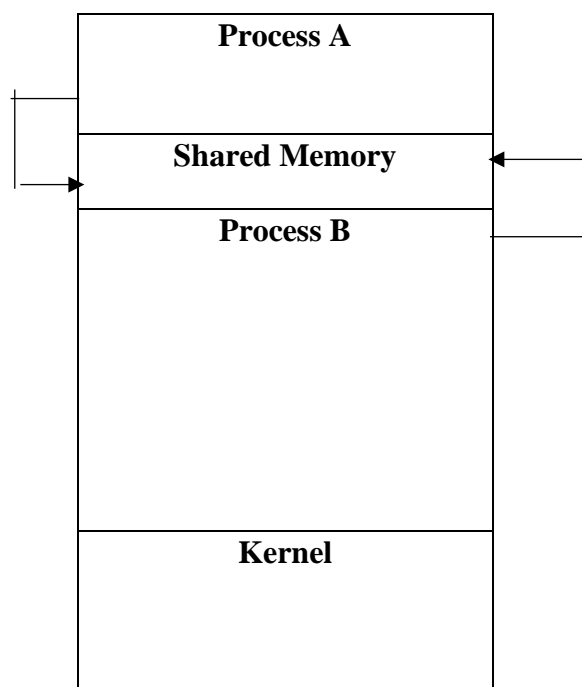
○ **Producer-Consumer Problem**

- In a **producer/consumer relationship**, the **producer** portion of an application generates data and stores it in a shared object, and the **consumer** portion of an application reads data from the shared object.
- One example of a common producer/consumer relationship is print spooling. A word processor spools data to a buffer (typically a file) and that data is subsequently consumed by the printer as it prints the document. Similarly, an application that copies data onto compact discs places data in a fixed-size buffer that is emptied as the CD-RW drive burns the data onto the compact disc.

○ **USING SHARED MEMORY**

## **Smt.Chandibai Himathmal Mansukhani College**

- Shared memory is memory that may be simultaneously accessed by multiple processes with an intent to provide communication among them or avoid redundant copies.
- Shared memory is an efficient means of passing data between processes.



### **➤ Producer-Consumer Solution using Shared Memory**

#### **A) QUESTION – 01 : -**

**Write a java program for producer consumer problem using shared memory.**

#### **B) SOURCE CODE**

**File Name: P4\_PC\_SM\_Buffer\_YashParab.java**

//Name: Yash Parab

## Smt.Chandibai Himathmal Mansukhani College

// Batch: B1  
// PRN: 2020016400922513  
// Date: 7th August,2021  
// Prac-04: Process Communication

```
public interface P4_PC_SM_Buffer_YashParab
{
    //Producers call this method
    public void insert(String item);

    //Consumers call this method
    public String remove();
} //interface ends
```

**File Name: P4\_PC\_SM\_BufferImpl\_YashParab.java**

//Name: Yash Parab  
// Batch: B1  
// PRN: 2020016400922513  
// Date: 7th August,2021  
// Prac-04: Process Communication

```
public class P4_PC_SM_BufferImpl_YashParab implements
P4_PC_SM_Buffer_YashParab
{
    private static final int BUFFER_SIZE = 5;
    private String[] elements;
    private int in,out,count;

    public P4_PC_SM_BufferImpl_YashParab() //constructor intializing the variable to
initial value

    {
```

## Smt.Chandibai Himathmal Mansukhani College

```
count = 0;

in = 0;

out = 0;

elements= new String[BUFFER_SIZE];


} // constructor ends

// Producers call this method

public void insert(String item)

{

    while(count == BUFFER_SIZE)

        ; // do nothing as there is no free space

    // add an item to the buffer

    elements[in] = item;

    in = (in + 1) % BUFFER_SIZE;

    ++count;

    System.out.println("Item Produced:" + item + " at position " + in + " having
total items " + count);

} // insert() ends

// Consumers call this method

public String remove()

{

    String item;

    while(count == 0)

        ; // do nothing as there is nothing to consume

    // remove an item from the buffer

    item = elements[out];

    out = (out + 1)% BUFFER_SIZE;

    --count;
```

## Smt.Chandibai Himathmal Mansukhani College

```
        System.out.println("Item Consumed: " + item + " from position " + out + "
remaining total items " + count);
```

```
        return item;

    } // remove() ends

} // class ends
```

**File Name: P4\_PC\_SM\_Yash Parab.java**

//Name: Yash Parab

// Batch: B1

// PRN: 2020016400922513

// Date: 7th August,2021

// Prac-04: Process Communication

```
public class P4_PC_SM_YashParab
{
```

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

        P4_PC_SM_BufferImpl_YashParab bufobj = new
P4_PC_SM_BufferImpl_YashParab();
```

```
        System.out.println("\n=====PRODUCER producing the
ITEMS=====\\n");
```

```
        bufobj.insert("Name: Yash Parab");
        bufobj.insert("CHMCS: Batch - B1");
        bufobj.insert("PRN: 2021202212345432");
        bufobj.insert("USCSP301 - USCS303: OS Practical - P4");
```

```
        System.out.println("\n=====CONSUMER consuming the
ITEMS=====\\n");
```

## Smt.Chandibai Himathmal Mansukhani College

```
String element = bufobj.remove();

System.out.println(element);

System.out.println(bufobj.remove());

System.out.println(bufobj.remove());

System.out.println(bufobj.remove());

} // main ends

} // class ends
```

### C) OUTPUT: -

```
C:\Users\parab>cd C:\USCSP301_USCSP303_OS_B1\Prac_04_YashParab_06_08_2021\Q1_PC_SM_YashParab

C:\USCSP301_USCSP303_OS_B1\Prac_04_YashParab_06_08_2021\Q1_PC_SM_YashParab>C:\Program Files\Java\jdk-16.0.1\bin"
"C:\Program Files\Java\jdk-16.0.1\bin" is not recognized as an internal or external command,
operable program or batch file.

C:\USCSP301_USCSP303_OS_B1\Prac_04_YashParab_06_08_2021\Q1_PC_SM_YashParab>set path="C:\Program Files\Java\jdk-16.0.1\bin"

C:\USCSP301_USCSP303_OS_B1\Prac_04_YashParab_06_08_2021\Q1_PC_SM_YashParab>javac P4_PC_SM_YashParab.java

C:\USCSP301_USCSP303_OS_B1\Prac_04_YashParab_06_08_2021\Q1_PC_SM_YashParab>java P4_PC_SM_YashParab

=====PRODUCER producing the ITEMS=====

Item Produced:Name: Yash Parab at position 1 having total items 1
Item Produced:CHMCS: Batch - B1 at position 2 having total items 2
Item Produced:PRN: 2021202212345432 at position 3 having total items 3
Item Produced:USCSP301 - USCSP303: OS Practical - P4 at position 4 having total items 4

=====CONSUMER consuming the ITEMS=====

Item Consumed: Name: Yash Parab from position 1 remaining total items 3
Name: Yash Parab
Item Consumed: CHMCS: Batch - B1 from position 2 remaining total items 2
CHMCS: Batch - B1
Item Consumed: PRN: 2021202212345432 from position 3 remaining total items 1
PRN: 2021202212345432
Item Consumed: USCSP301 - USCSP303: OS Practical - P4 from position 4 remaining total items 0
USCSP301 - USCSP303: OS Practical - P4
```

### ➤ USING MESSAGE PASSING

- Message passing is the basis of most inter-process communication in distributed systems.

## **Smt.Chandibai Himathmal Mansukhani College**

- It is at the lowest level of abstraction and requires the application programmer to be able to identify the destination process, the message, the source process and the data types expected from these processes.
- Communication in the message passing paradigm, in its simplest form, is performed using the send() and receive() primitives. The syntax is generally of the form:

**send (receiver,  
message) receive  
(sender, message)**

- The send() primitive requires the name of the destination process and the message data as parameters. The addition of the name of the sender as a parameter for the send() primitive would enable the receiver to acknowledge the message. The receive() primitive requires the name of the anticipated sender and should provide a storage buffer for the message.

### ➤ **Producer-Consumer Solution using Shared Memory**

#### **A) QUESTION - 02: -**

**Write a java program for producer consumer problem using message passing.**

#### **B) SOURCE CODE**

**File Name: P4\_PC\_MP\_Channel\_YashParabjava**

//Name: Yash Parab

// Batch: B1



## **Smt.Chandibai Himathmal Mansukhani College**

// PRN: 2020016400922513

// Date: 7th August,2021

// Prac-04: Process Communication

```
public interface P4_PC_MP_Channel_YashParab<E>
```

```
{
```

```
// Send a message to the channel
```

```
public void send(E item);
```

```
// Receive a message from the channel
```

```
public E receive();
```

```
}//interface ends
```

**FileName:P4\_PC\_MP\_MessageQueue\_Yash Parab.java**

//Name: Yash Parab

// Batch: B1

// PRN: 2020016400922513

// Date: 7th August,2021

// Prac-04: Process Communication

```
import java.util.Vector;
```

```
public class P4_PC_MP_MessageQueue_YashParab<E> implements
```

```
P4_PC_MP_Channel_YashParab<E>
```

```
{
```

```
    private Vector<E> queue;
```

```
    public P4_PC_MP_MessageQueue_YashParab(){
```

```
        queue = new Vector<E>();
```

```
    }
```

## **Smt.Chandibai Himathmal Mansukhani College**

```
// This implements a nonblocking send
public void send(E item){
    queue.addElement(item);
} //send() ends

//This implements a nonblocking receive
public E receive(){
    if(queue.size() == 0)
        return null;
    else
        return queue.remove(0);
} // receive() ends
} // class ends
```

**File Name: P4\_PC\_MP\_YashParab.java**

```
//Name: Yash Parab
// Batch: B1
// PRN: 2020016400922513
// Date: 7th August,2021
// Prac-04: Process Communication
```

```
import java.util.Date;
public class P4_PC_MP_YashParab
{
    public static void main(String args[])
    {
        //Producer and Consumer process
        P4_PC_MP_Channel_YashParab<Date> mailBox = new
P4_PC_MP_MessageQueue_YashParab<Date>();
        int i = 0;
```

## Smt.Chandibai Himathmal Mansukhani College

```
do
{
    Date message = new Date();

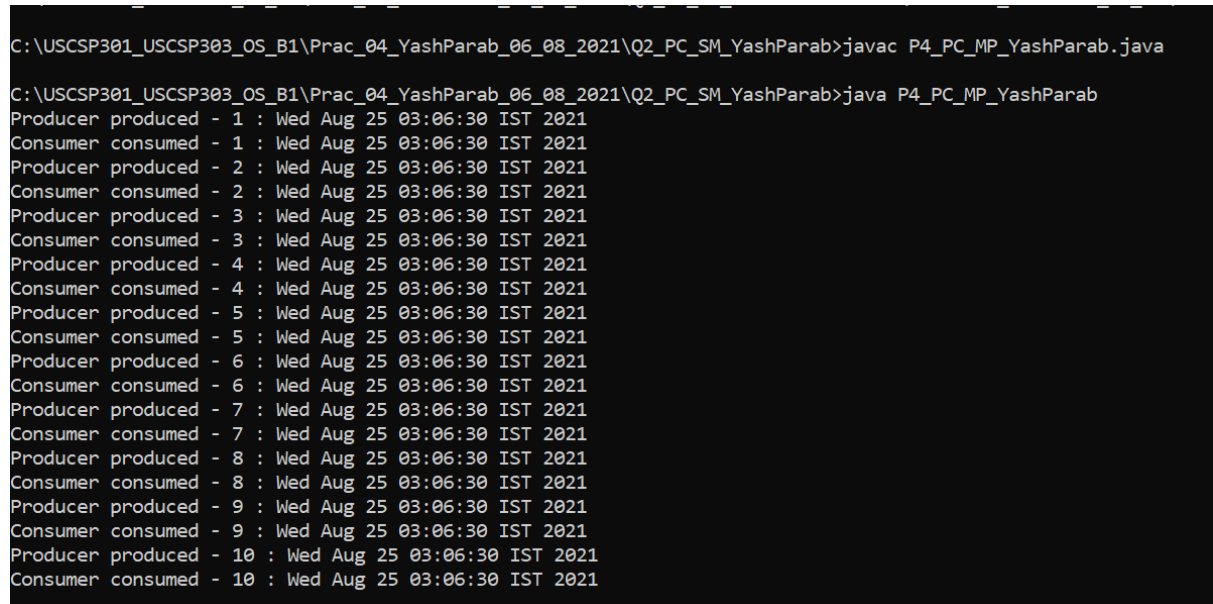
    System.out.println("Producer produced - " + (i+1) + " : " + message);

    mailBox.send(message);

    Date rightNow = mailBox.receive();

    if(rightNow != null)
    {
        System.out.println("Consumer consumed - " + (i+1) + " : " +
rightNow);
    }
    i++;
}while (i<10);
} // main ends
} // class ends
```

### C) OUTPUT: -



```
C:\USCSP301_USCSP303_OS_B1\Prac_04_YashParab_06_08_2021\Q2_PC_SM_YashParab>javac P4_PC_MP_YashParab.java
C:\USCSP301_USCSP303_OS_B1\Prac_04_YashParab_06_08_2021\Q2_PC_SM_YashParab>java P4_PC_MP_YashParab
Producer produced - 1 : Wed Aug 25 03:06:30 IST 2021
Consumer consumed - 1 : Wed Aug 25 03:06:30 IST 2021
Producer produced - 2 : Wed Aug 25 03:06:30 IST 2021
Consumer consumed - 2 : Wed Aug 25 03:06:30 IST 2021
Producer produced - 3 : Wed Aug 25 03:06:30 IST 2021
Consumer consumed - 3 : Wed Aug 25 03:06:30 IST 2021
Producer produced - 4 : Wed Aug 25 03:06:30 IST 2021
Consumer consumed - 4 : Wed Aug 25 03:06:30 IST 2021
Producer produced - 5 : Wed Aug 25 03:06:30 IST 2021
Consumer consumed - 5 : Wed Aug 25 03:06:30 IST 2021
Producer produced - 6 : Wed Aug 25 03:06:30 IST 2021
Consumer consumed - 6 : Wed Aug 25 03:06:30 IST 2021
Producer produced - 7 : Wed Aug 25 03:06:30 IST 2021
Consumer consumed - 7 : Wed Aug 25 03:06:30 IST 2021
Producer produced - 8 : Wed Aug 25 03:06:30 IST 2021
Consumer consumed - 8 : Wed Aug 25 03:06:30 IST 2021
Producer produced - 9 : Wed Aug 25 03:06:30 IST 2021
Consumer consumed - 9 : Wed Aug 25 03:06:30 IST 2021
Producer produced - 10 : Wed Aug 25 03:06:30 IST 2021
Consumer consumed - 10 : Wed Aug 25 03:06:30 IST 2021
```

# Smt.Chandibai Himathmal Mansukhani College

## Remote Procedure Calls.

- Message passing leaves the programmers with the burden of the explicit control of the movement of data. Remote procedure calls (RPC) relieves this burden by increasing the level of abstraction and providing semantics similar to local procedure call.
- Syntax:
- The syntax of a remote procedure call is generally of the form:  
**call procedure\_name(value\_arguments;result\_arguments)**
- The client process blocks at the **call()** until the reply is received
- The remote procedure is the server processes which has already begun executing on a remote machine.
- It blocks at the **receive()** until it receives a message and parameters from the sender.
- The server then sends a **reply()** when it has finished its task.
- The syntax is as follows:  
**receive procedure\_name(in value\_parameters; out result\_parameters)**  
**reply(caller, result\_parameters)**
- In the simplest case, the execution of **call()** generates a client stub which marshals the arguments into a message and sends the message to the server machine. On the server machine the server is blocked awaiting the message. On receipt of the message the server stub is generated and extracts the parameters from the message and passes the parameters and control to the procedure. The result are returned to client with the same procedure in reverse.

## ➤ Remote Method Invocation (RMI) Calculator.

## Smt.Chandibai Himathmal Mansukhani College

### A) QUESTION-03: -

Write a java RMI program for adding , subtracting , multiply and dividing two numbers.

Steps:-

- ✓ **Step 1: Creating the Remote Interface**
  - This file defines the remote interface that is provided by the server. It contains four methods that accepts two **integer** arguments and returns their sum, difference, product and quotient. All remote interfaces must extend the **Remote** interface, which is part of java.rmi. **Remote** defines no members. Its purpose is simply to indicate that an interface uses remote methods. All remote methods can throw a **RemoteException**.
- ✓ **Step 2: Implementing the Remote Interface**
  - This file implements the remote interface. The implementation of all the four methods is straight forward. All remote methods must extend **UnicastRemoteObject**, which provides functionality that is needed to make objects available from remote machines.
- ✓ **Step 3: Creating the server**
  - This file contains the main program for the server machine. Its primary function is to update the RMI registry on that machine. This is done by using the **rebind()** method of the **Naming** class (found in **java.rmi**), that method associates a name with an object reference. The first argument to the **rebind()** method is a string that names the server. Its second argument is a reference to an instance of **CalcServerImpl**.
- ✓ **Step 4: Creating the Client**
  - This file implements the client side of this distributed application. It accepts three command-line arguments. The first is the IP address or name of the server machine. The second and third arguments are the two numbers that are to be operated.
  - The application begins by forming a string that follows the URL syntax. This URL uses the rmi protocol. The string includes the IP address or name of the server and the string "CSB1". The program then invokes the **lookup()** method of the **Naming** class. This method accepts one argument, the rmi URL, and returns a reference to an object of type **CalcServerInf**.  
All remote method invocations can then be directed to this object.
- ✓ **Step 5: Manually generate a stub, if required**

## Smt.Chandibai Himathmal Mansukhani College

- Prior to Java 5, stubs needed to be built manually by using `rmic`. This step is not required for modern versions of Java. However, if we work in a legacy environment, then we can use the `rmic` compiler, as shown here, to build a stub.

### **`rmic CalcServerImpl`**

#### ✓ **Step 6: Install Files on the Client and Server Machines**

- Copy **`P4_RMI_CalcClient_YashParab.class`**, **`P4_RMI_CalcServerImpl_YashParab_Stub.class`** (if needed), and **`P4_RMI_CalcServerIntf_YashParab.class`** to a directory on the client machine.
- Copy **`CalcServerIntf.class`**, **`P4_RMI_CalcServerImpl_YashParab.class`**, **`P4_RMI_CalcServerImpl_YashParab_Stub.class`** (if needed), and **`P4_RMI_CalcServer_YashParab.class`** to a directory on the server machine.

#### ✓ **Step 7: Start the RMI Registry on the Server Machine**

- The JDK provides a program called `rmiregistry`, which executes on the server machine. It maps names to object references. Start the RMI Registry from the command line, as shown here: **`start rmiregistry`**
- When this command returns, a new window gets created. Leave this window open until we are done experimenting with the RMI example.

#### ✓ **Step 8: Start the server**

- The server code is started from the command line, as shown here:

**`java P4_RMI_CalcServer_YashParab`**

#### ✓ **Step 9: Start the client**

- The client code is started from the command line, as shown here:

**`java P4_RMI_CalcClient_YashParab 127.0.0.1 15 5`**

### ‡ **For Execution:**

#### ○ **Open 2 Command Terminals:**

##### ✦ **Terminal-01:**

- ✦ Compile all the .java files

- ✦ Command:

**`javac`**

**`*.java`** ✦ Start  
the RMI  
registry:

- ✦ Command:

**`start rmiregistry`**

##### ✦ **Terminal-02:**

- ✦ **Start the Server:**

# Smt.Chandibai Himathmal Mansukhani College

✦ Command:

Java P4\_RMI\_CalcServer\_  
YashParab

✦ Run the Client:

✦ Command:

java P4\_RMI\_CalcClient\_  
YashParab 127.0.0.1 15 5

## C) SOURCE CODE

**File Name: P4\_RMI\_CalcServerIntf\_ YashParab java**

//Name: Yash Parab

// Batch: B1

// PRN: 2020016400922513

// Date: 7th August,2021

// Prac-04: Process Communication

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

```
public interface P4_RMI_CalcServerIntf_ YashParab extends Remote
```

```
{
```

```
    int add(int a, int b)throws RemoteException;
```

```
    int subtract(int a, int b)throws RemoteException;
```

```
    int multiply(int a, int b)throws RemoteException;
```

```
    int divide(int a, int b)throws RemoteException;
```

```
}//interface ends
```

**File Name: P4\_RMI\_CalcServerImpl\_ YashParab.java**

//Name: Yash Parab

// Batch: B1

// PRN: 2020016400922513

## **Smt.Chandibai Himathmal Mansukhani College**

// Date: 7th August,2021

// Prac-04: Process Communication

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

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

```
public class P4_RMI_CalcServerImpl_YashParab extends UnicastRemoteObject  
implements
```

```
P4_RMI_CalcServerIntf_YashParab
```

```
{
```

```
public P4_RMI_CalcServerImpl_YashParab()throws RemoteException{
```

```
}
```

```
public int add(int a, int b)throws RemoteException
```

```
{
```

```
return a + b;
```

```
}
```

```
public int subtract(int a, int b)throws RemoteException
```

```
{
```

```
return a - b;
```



## Smt.Chandibai Himathmal Mansukhani College

}

```
public int multiply(int a, int b)throws RemoteException
```

```
{
```

```
    return a * b;
```

```
}
```

```
public int divide(int a, int b)throws RemoteException
```

```
{
```

```
    return a / b;
```

```
}
```

```
}//class ends
```

**File Name: P4\_RMI\_CalcServer\_ YashParab.java**

```
//Name: Yash Parab
```

```
// Batch: B1
```

```
// PRN: 2020016400922513
```

```
// Date: 7th August,2021
```

```
// Prac-04: Process Communication
```

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

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

## **Smt.Chandibai Himathmal Mansukhani College**

```
public class P4_RMI_CalcServer_YashParab

{

    public static void main(String args[])

    {

        try

        {

            P4_RMI_CalcServerImpl_YashParab csi = new
P4_RMI_CalcServerImpl_YashParab();

            Naming.rebind("CSB1", csi);

        } //try ends

        catch(Exception e) {

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

        } //catch ends

    } //main ends

} //class ends
```

**File Name: P4\_RMI\_CalcClient\_YashParab.java**

## **Smt.Chandibai Himathmal Mansukhani College**

//Name: Yash Parab

// Batch: B1

// PRN: 2020016400922513

// Date: 7th August,2021

// Prac-04: Process Communication

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

```
public class P4_RMI_CalcClient_YashParab
```

```
{
```

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

```
{
```

```
try
```

```
{
```

```
String CSURL = "rmi://" + args[0] + "/CSB1";
```

```
P4_RMI_CalcServerIntf_YashParab CSIntf = (P4_RMI_CalcServerIntf_YashParab)
```

```
Naming.lookup(CSURL);
```

```
System.out.println("The first number is: " + args[1]);
```

```
int x = Integer.parseInt(args[1]);
```

## Smt.Chandibai Himathmal Mansukhani College

```
System.out.println("The second number is: " + args[2]);
```

```
int y = Integer.parseInt(args[2]);
```

```
System.out.println("=====Arithmetic Operations=====");
```

```
System.out.println("Addition: " + x + " + " + y + " = " + CSIntf.add(x,y));
```

```
System.out.println("Subtraction: " + x + " - " + y + " = " + CSIntf.subtract(x,y));
```

```
System.out.println("Addition: " + x + " * " + y + " = " + CSIntf.multiply(x,y));
```

```
System.out.println("Addition: " + x + " / " + y + " = " + CSIntf.divide(x,y));
```

```
}//try ends
```

```
catch(Exception e){
```

```
System.out.println("Exception: " + e);
```

```
}//catch ends
```

```
}//main ends
```

```
}//class ends
```

### D) OUTPUT:

# Smt.Chandibai Himathmal Mansukhani College

```
C:\USCSP301_USCSP303_OS_B1\Prac_03_YashParab_27_07_2021>cd C:\USCSP301_USCSP303_OS_B1\Prac_04_YashParab_06_08_2021\Q3_PC_RMI_YashParab
C:\USCSP301_USCSP303_OS_B1\Prac_04_YashParab_06_08_2021\Q3_PC_RMI_YashParab>dir *.*
Volume in drive C is Windows
Volume Serial Number is 2C7E-3D5F

Directory of C:\USCSP301_USCSP303_OS_B1\Prac_04_YashParab_06_08_2021\Q3_PC_RMI_YashParab

21-08-2021  00:50    <DIR>          .
21-08-2021  00:50    <DIR>          ..
22-08-2021  23:07          1,834 P4_RMI_CalcClient_YashParab.class
21-08-2021  00:46          1,020 P4_RMI_CalcClient_YashParab.java
22-08-2021  23:07           604 P4_RMI_CalcServerImpl_YashParab.class
21-08-2021  00:35           616 P4_RMI_CalcServerImpl_YashParab.java
22-08-2021  23:07           326 P4_RMI_CalcServerIntf_YashParab.class
21-08-2021  00:28           324 P4_RMI_CalcServerIntf_YashParab.java
22-08-2021  23:07          1,136 P4_RMI_CalcServer_YashParab.class
21-08-2021  00:40           408 P4_RMI_CalcServer_YashParab.java
                8 File(s)              6,268 bytes
                2 Dir(s)  140,542,812,160 bytes free

C:\USCSP301_USCSP303_OS_B1\Prac_04_YashParab_06_08_2021\Q3_PC_RMI_YashParab>javac *.java

C:\USCSP301_USCSP303_OS_B1\Prac_04_YashParab_06_08_2021\Q3_PC_RMI_YashParab>start rmi registry
The system cannot find the file rmi.

C:\USCSP301_USCSP303_OS_B1\Prac_04_YashParab_06_08_2021\Q3_PC_RMI_YashParab>start rmi registry

C:\USCSP301_USCSP303_OS_B1\Prac_04_YashParab_06_08_2021\Q3_PC_RMI_YashParab>java P4_RMI_CalcServer_YashParab
```

```
C:\Users\parab>cd C:\USCSP301_USCSP303_OS_B1\Prac_04_YashParab_06_08_2021\Q3_PC_RMI_YashParab

C:\USCSP301_USCSP303_OS_B1\Prac_04_YashParab_06_08_2021\Q3_PC_RMI_YashParab>set path="C:\Program Files\Java\jdk-16.0.1\bin"

C:\USCSP301_USCSP303_OS_B1\Prac_04_YashParab_06_08_2021\Q3_PC_RMI_YashParab>java P4_RMI_CalcClient_YashParab 127.0.0.1 15 16
The first number is: 15
The second number is: 16
=====Arithmetic Operations=====
Addition: 15 + 16 = 31
Subtraction: 15 - 16 = -1
Addition: 15 * 16 = 240
Addition: 15 / 16 = 0

C:\USCSP301_USCSP303_OS_B1\Prac_04_YashParab_06_08_2021\Q3_PC_RMI_YashParab>
```