Assignment No 2

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

Objective:

basic implementation of a Java/CORBA application using static invocations.

Outcome:

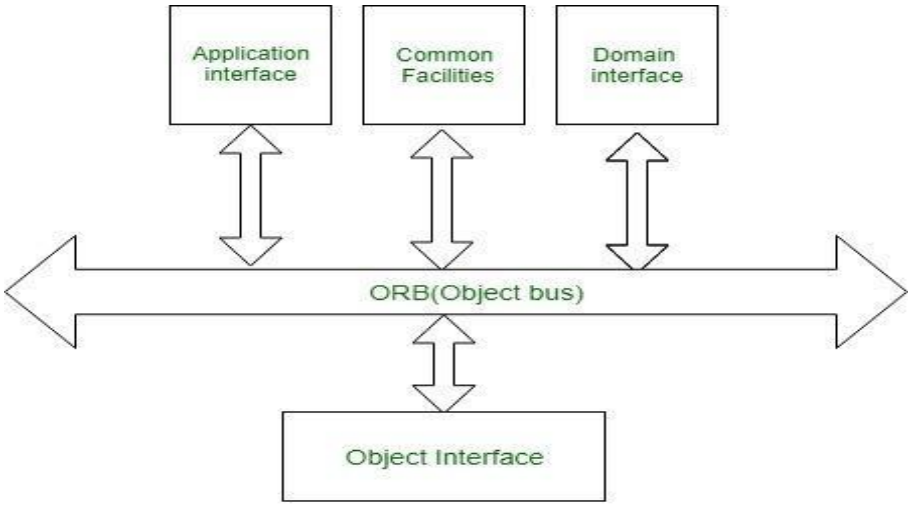
CORBA also provides a dynamic invocation capability where clients can discover the interfaces on-the-fly.

Explanation:

Common Object Request Broker Architecture (CORBA) could be a specification of a regular design for middleware. It is a client-server software development model

CORBA Reference Model:

The CORBA reference model known as Object Management design (OMA) is shown below figure. The OMA is itself a specification (actually, a group of connected specifications) that defines a broad vary of services for building distributed client-server applications



Object Management Architecture(OMA)

Divide and conquer. The remote objects can be independently designed.
Increase reusability. It is often possible to design the remote objects so that other systems can use them too.
Increase reuse. You may be able to reuse remote objects that others have created.
Design for flexibility. The broker objects can be updated as required, or you can redirect the proxy to communicate with a different remote object.
Design for portability. You can write clients for new platforms while still accessing brokers and remote objects on other platforms.
Design defensively. You can provide careful assertion checking in the remote objects.

The Calculator sample application is a basic implementation of a Java/CORBA application using static invocations. A static invocation is when a client is aware of what interfaces and methods are available at compile time. Note: CORBA also provides a dynamic invocation capability where clients can discover the interfaces on-the-fly.

The Java/COBRA development process for the Calculator example application is broken down in to ten manageable steps:

1. Download and install a Java ORB
2. Create IDL file
3. Compile IDL file
4. Create the client
5. Create the server
6. Create the interface implementation
7. Compile the client
8. Compile the server
9. Compile the interface implementation
10. Start the naming service (OS Agent)
11. Start the server
12. Start the client.

Program for Calculator Application

1. Calc.idl:

```
module WssCalculator

{
```

```
interface Calc

{

//Performs the Calculations:ADD/SUB/MUL/DIV

long calculate(in long operator,in long num1,in long num2);

//The Server EXITS when the Client prompts it to do so

oneway void shutdown();

};

};
```

2. CalcServer.java:

```
//Importing all the packages and classes

//Import the package which contains the Server Skeleton

import WssCalculator.*;

//Import the below two packages to use the Naming Service

import org.omg.CosNaming.*;

import org.omg.CosNaming.NamingContextPackage.*;

//Import this package to run the CORBA Application

import org.omg.CORBA.*;//

//Import the below to Classes for inheriting Portable Server

import org.omg.PortableServer.*;

import org.omg.PortableServer.POA;

//Initiate the ORB using the class Properties
```

```
import java.util.Properties;

//Perform the Input-Output functionalities

import java.io.*;

import java.util.*;



//Write the Servant class

//It inherits the general CORBA utilities generated by the Compiler

class Calcserverimpl extends CalcPOA

{

//orb variable is used to invoke the shutdown()

private ORB orb;

public void setORB(ORB orb_val)



{

orb = orb_val;

}



//Declaring and Implementing the required method

public int calculate(int a,int b,int c)

{
```

```
//ADDITION

if(a==43)

{

return (b+c);

}

//SUBTRACTION

else if(a==45)

{

return (b-c);

}

//MULTIPLICATION

else if(a==42)

{

return (b*c);

}

//DIVISION

else if(a==47)

{

return (b/c);

}
```

```
//DEFAULT

else

{

return 0;

}

}

//Closing the server

public void shutdown()

{

orb.shutdown(false);

}

}//end of the servant class



public class CalcServer

{

public static void main(String args[])

{

try

{

//Create and Initialize the ORB object

//init() allows to set the properties at run time
```

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

//Obtain the initial Naming Context

//Obtain an initial object reference to the name server

//orb retrieves the reference to the Root POA

//Activate the POA Manager

//activate() causes the POAs to process the client requests

POA rootpoa=POAHelper.narrow(orb.resolve_initial_references("RootPOA"));

rootpoa.the_POAManager().activate();

//The server instantiates the servant objects

//The servant performs the operations defined in the idlj interface

Calcserverimpl simpl=new Calcserverimpl();

simpl.setORB(orb);

//Get the object reference associated with the servant

//narrow() is used to cast CORBA obj ref to its proper type

org.omg.CORBA.Object ref = rootpoa.servant_to_reference(simpl);

Calc href=CalcHelper.narrow(ref);

//Obtain the initial Naming Context

//Obtain an object reference to the Name Server

org.omg.CORBA.Object objRef=orb.resolve_initial_references("NameService");
```

```
//Narrow the objref to its proper type

NamingContextExt ncRef=NamingContextExtHelper.narrow(objRef);

//Register the Servant with the Name Server

String name = "Calc";

//NameComponent array contains the path to Calc

NameComponent path[]=ncRef.to_name(name);

//Pass the path and the servant object to the Naming Service

//Bind the servant object to Calc

ncRef.rebind(path,href);

System.out.println("The SERVER is READY");

System.out.println("The SERVER is WAITING to receive the CLIENT requests");

//run() is called by the main thread

//run() enables the ORB to perform work using the main thread

//the server waits until an invocation comes from the ORB

orb.run();

}

catch (Exception e)

{

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

e.printStackTrace(System.out);

}
```

```
//This statement is executed when the Client wishes to discontinue

System.out.println("The Server Exits");

}//end of main()

}//end of CalcServer()
```

3. CalcClient.java:

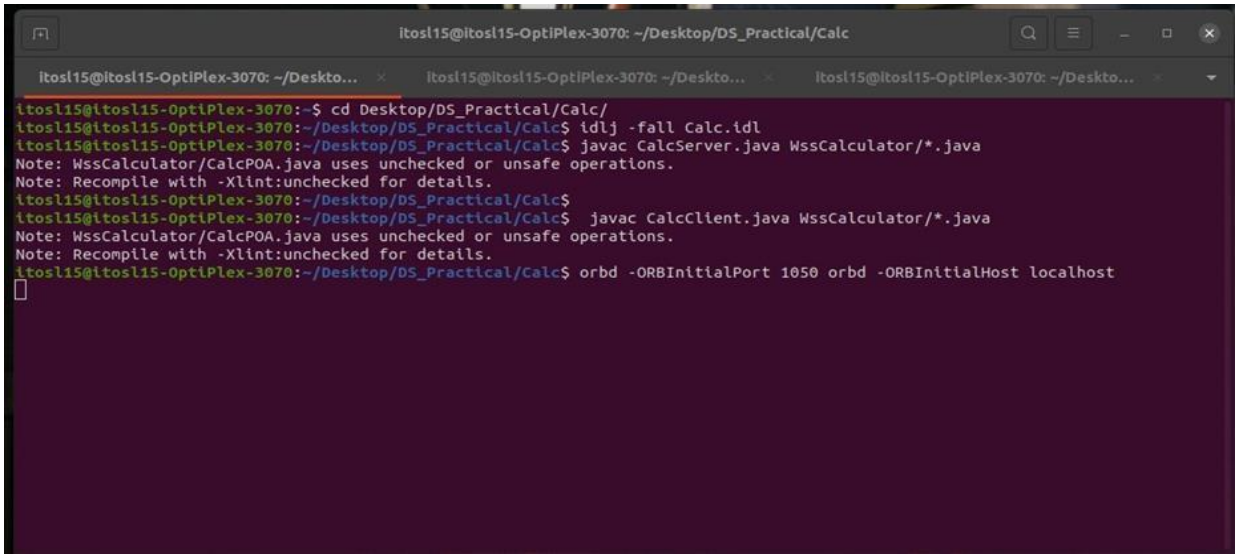
```
//Import all the important packages

//Import the package which contains the Client Stub

import WssCalculator.*;

//Import the below two packages to use the Naming Service

import org.omg.CosNaming.*;

import org.omg.CosNaming.NamingContextPackage.*;

//Import this package to run the CORBA Applicaion

import org.omg.CORBA.*;

//Import to perform Input-Output functionalities

import java.io.*;

import java.util.*;

public class CalcClient

{

static Calc cimpl;

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

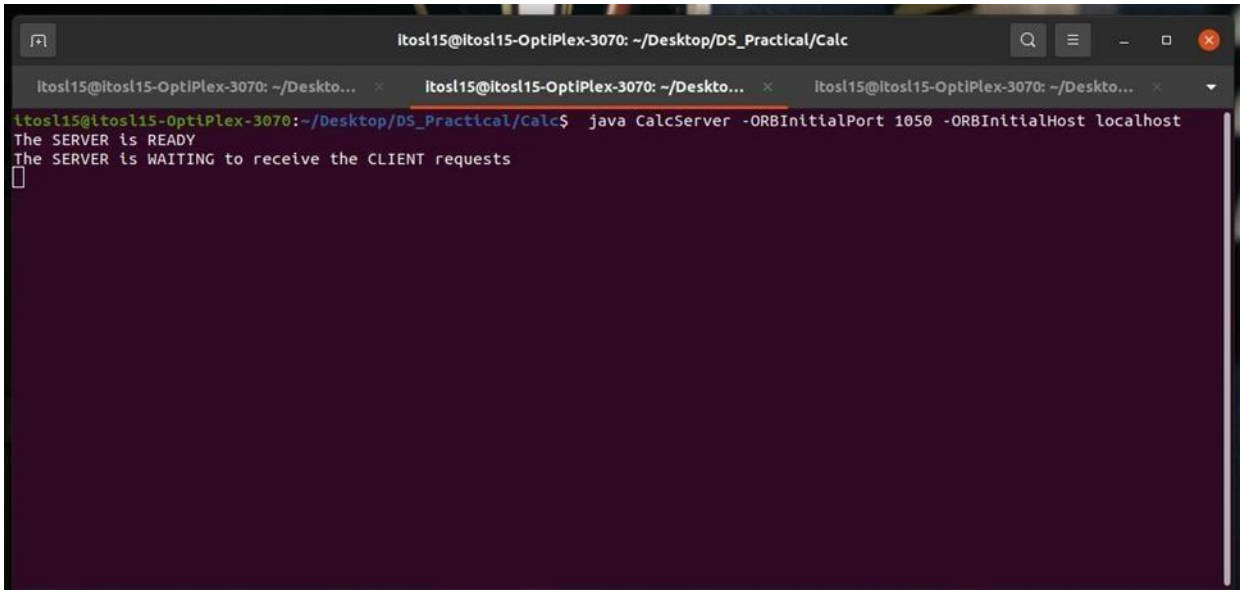
```
{

try

{

//Declaring and initializing the variables

int dec=1;

int  i=0;

int  j=0;

int k=0;

int result=0;

int x=1;

char c='x';

char d='y';

char f='z';

String abc="vas";

//Create and Initialize the ORB object

//init() allows to set properties at run time

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

//ORB helps the Client to locate the actual services which it needs

//COS Naming Service helps the client to do so

//Obtain the initial Naming Context

//Obtain an object reference to the name server
```

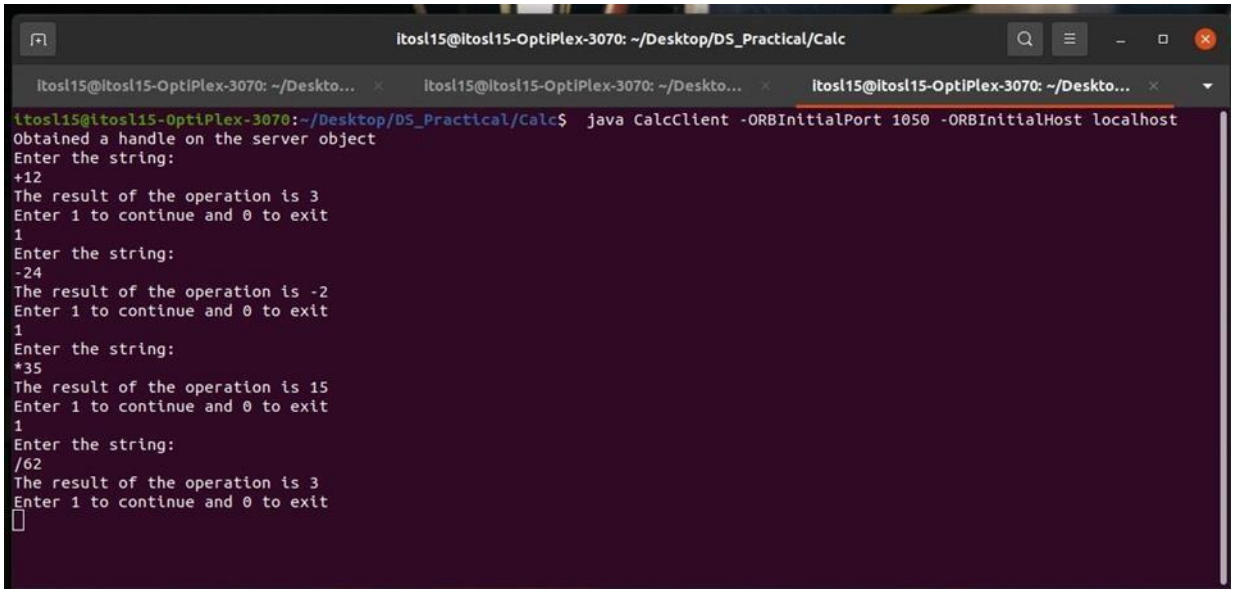
```
org.omg.CORBA.Object objRef=orb.resolve_initial_references("NameService");

//Narrow the objref to its proper type

NamingContextExt ncRef=NamingContextExtHelper.narrow(objRef);

//Identify a String to refer the Naming Service to Calc object

String name="Calc";

//Get a reference to the CalcServer and Narrow it to Calc object

cimpl=CalcHelper.narrow(ncRef.resolve_str(name));

System.out.println("Obtained a handle on the server object");

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

while(x==1)

{

System.out.println("Enter the string:");

abc=br.readLine();

//Separate the input string into separate characters

c=abc.charAt(0);

d=abc.charAt(1);

f=abc.charAt(2);

//Get the ASCII value of the Operator

i=(int)c;

//Get the Integer values of the other two characters

j=Character.getNumericValue(d);
```

```
k=Character.getNumericValue(f);

result=cimpl.calculate(i,j,k);

System.out.println("The result of the operation is "+result);

System.out.println("Enter 1 to continue and 0 to exit ");

x=Integer.parseInt(br.readLine());

}

//If the Client wants to discontinue

cimpl.shutdown();

}

catch(Exception e)

{

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

e.printStackTrace(System.out);

}

}//end of main()

}//end of class
```

Output:


itosl15@itosl15-OptiPlex-3070: ~/Desktop/DS_Practical/Calc
itosl15@itosl15-OptiPlex-3070: ~/Deskto...
itosl15@itosl15-OptiPlex-3070:~$ cd Desktop/DS_Practical/Calc/
itosl15@itosl15-OptiPlex-3070:~/Desktop/DS_Practical/Calc$ idlj -fall Calc.idl
itosl15@itosl15-OptiPlex-3070:~/Desktop/DS_Practical/Calc$ javac CalcServer.java WssCalculator/*.java
Note: WssCalculator/CalcPOA.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
itosl15@itosl15-OptiPlex-3070:~/Desktop/DS_Practical/Calc$
itosl15@itosl15-OptiPlex-3070:~/Desktop/DS_Practical/Calc$ javac CalcClient.java WssCalculator/*.java
Note: WssCalculator/CalcPOA.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
itosl15@itosl15-OptiPlex-3070:~/Desktop/DS_Practical/Calc$ orbd -ORBInitialPort 1050 orbd -ORBInitialHost localhost


itosl15@itosl15-OptiPlex-3070: ~/Desktop/DS_Practical/Calc
itosl15@itosl15-OptiPlex-3070: ~/Deskto...
itosl15@itosl15-OptiPlex-3070:~/Desktop/DS_Practical/Calc$ java CalcServer -ORBInitialPort 1050 -ORBInitialHost localhost
The SERVER is READY
The SERVER is WAITING to receive the CLIENT requests



Result:

Successfully Created CORBA system for Timestamp application.

Date:	
Marks obtained:	
Sign of course coordinator:	
Name of course Coordinator :	