Date:12.02.2022

**Third Year B. Tech., Sem VI 2021-22**

**Cloud Computing**

**Assignment submission**

**PRN No: 2019BTECS00064**

**Full name: Kunal Santosh Kadam**

**Batch: T2**

**Assignment: 2**

**Title of assignment: Implementation of CORBA (Common Object Request Broker Architecture)**

**Implementation of CORBA**

**CORBA**

The Common Object Request Broker Architecture (CORBA) is a standard developed by the Object Management Group (OMG) to provide interoperability among distributed objects.

**CORBA- Middleware**

CORBA is the world’s leading middleware solution enabling the exchange of information, independent of hardware platforms, programming languages and operating systems. CORBA uses an object-oriented model although the systems that use the CORBA do not have to be object-oriented.

**Types of Models**

1. Inheritance Model

IDL interface is implemented using an implementation class that also extends the compiler-generated skeleton.

1. Delegation Model

The Delegation model is also known as the Tie model, or the Tie Delegation model. It inherits from either the POA or ImplBase compiler-generated skeleton, so the models will be described as POA/Tie or ImplBase/Tie models in this document.

**Implementation of Server**

* The server consists of two classes, the servant and the server.
* The servant, AdditionImpl, is the implementation of the Addition IDL interface; each Addition instance is implemented by a AdditionImpl instance.
* The servant is a subclass of AdditionPOA, which is generated by the idlj compiler from the example IDL.
* The servant contains one method for each IDL operation, in this example, the add() and shutdown() methods.
* Servant methods are just like ordinary Java methods; the extra code to deal with the ORB, with marshalling arguments and results, and so on, is provided by the skeleton.

**Implementation of CORBA in Java**

**Creating Interface**

//save file add Addition.idl

module AdditionApp

{

interface Addition

{

long add(in long a,in long b);

oneway void shutdown();

};

};

**Creating Server Side**

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 ...");

}

}

**Creating Client Side**

import AdditionApp.\*;

import org.omg.CosNaming.\*;

import org.omg.CosNaming.NamingContextPackage.\*;

import org.omg.CORBA.\*;

import java.io.\*;

import java.util.\*;

public class StartClient

{

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);

if(a==0)

break;

System.out.println("The result for addition is : "+r);

System.out.println("-----------------------------------");

}

}

catch (Exception e)

{

System.out.println("Hello Client exception: " + e);

e.printStackTrace();

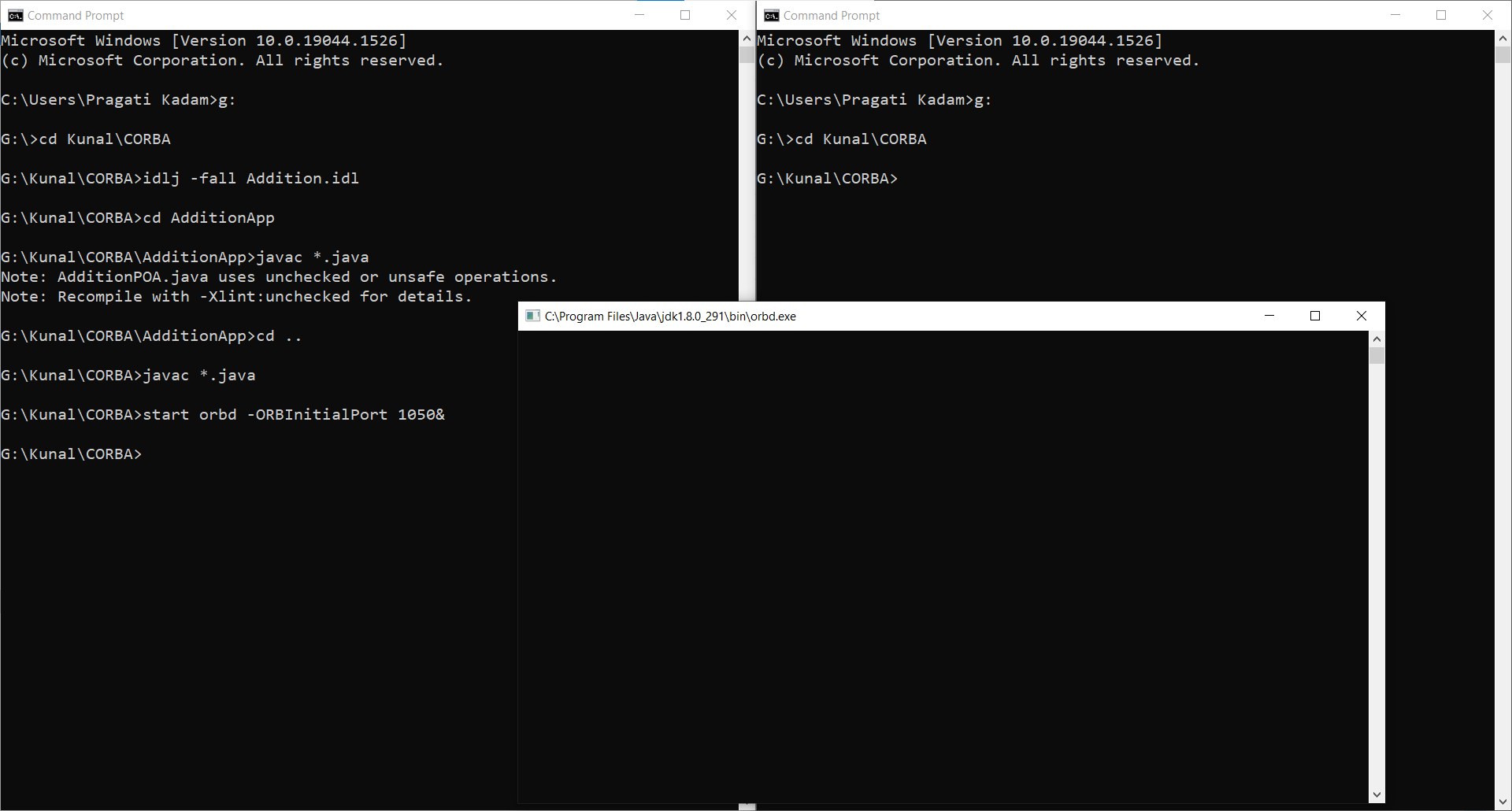
}

}

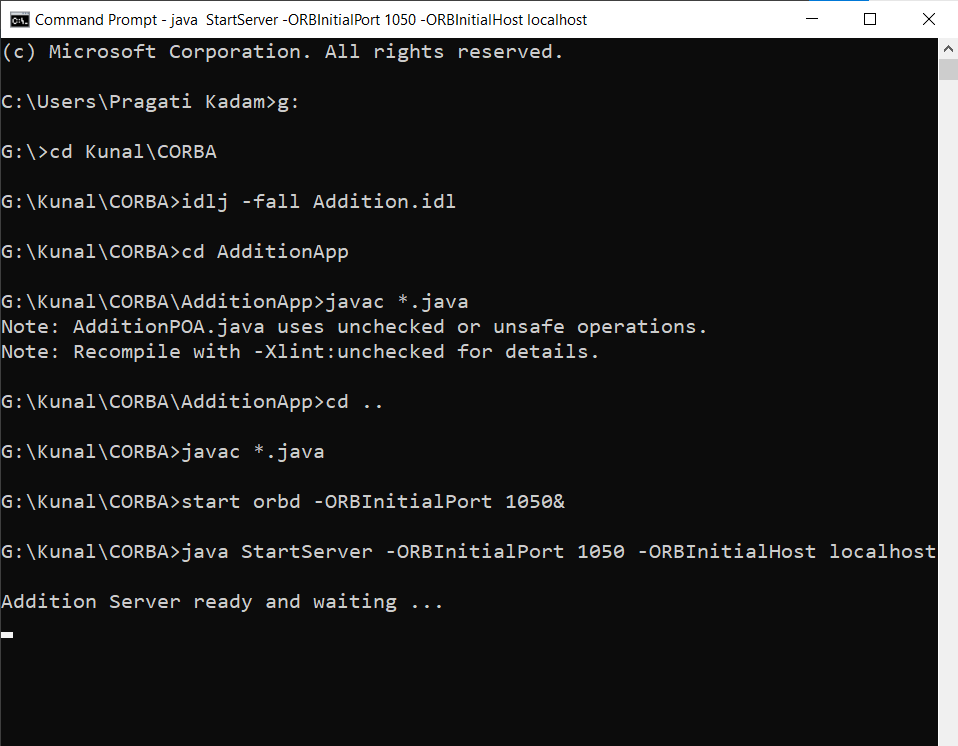
}

**Output:-**

Starting orbd and executing all .java files



Running StartServer.java file



Running StartClient.java file

