**EXPERIMENT - 11**

**Aim**: Demonstration of CORBA

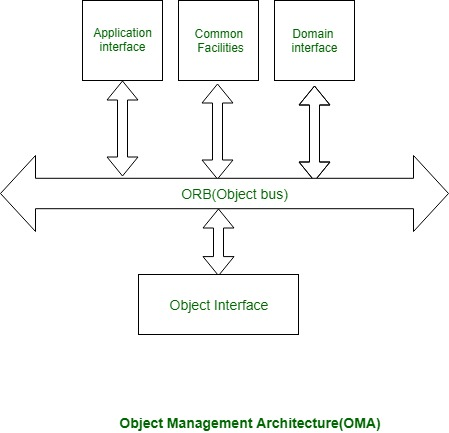
**Theory**:

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

Using a CORBA implementation, a shopper will transparently invoke a way on a server object, which may air a similar machine or across a network. The middleware takes the decision associated to blame for finding an object which will implement the request, passing it the parameters, invoking its methodology, and returning the invocation results. The shopper doesn’t need to remember wherever the item is found, its programming language, software package, or the other aspects that don’t seem to be a part of the associated object’s interface.

**CORBA Reference Model:**

The CORBA reference model, known as Object Management design (OMA) is shown below. The OMA is a specification (actually, a group of connected specifications) that defines various services for building distributed client-server applications. several services one may expect to search out in a very middleware product like CORBA (e.g., naming, dealings, and asynchronous event management services) are literally fixed as services within the OMA



Different parts communicate victimization ORB. ORB is additionally referred to as the item bus. An associate example of the application interface is a distributed document facility. In a very domain interface, it will have domain-dependent services, for instance, producing domain.

Object interface has some domain freelance services:

1. **Naming Service:**

Naming service is additionally known as a white page service. victimization naming service server name will be searched, and it’s location, or address will be pointed out.

1. **Trading Service**:

Commercialism is also known as a yellow page service. victimization commercialism service, a selected service will be searched. This often corresponds to looking out for a service like an automobile store in a very yellow page directory.

The CORBA Application is composed of three programs:

1. **idl program**– which contains the declaration of methods to be called by the client and defined by the server program. The return type of method or parameters should not be integer as CORBA does not support integer data type; instead short or double can be used.
2. **Server Program** – which contains definition of the methods which are declared in idl file and called by the client program.
3. **Client program** – which contains Method calling defined at the server.

**Code**:

| // Hello.idl  module HelloApp{  interface Hello {  // sayHello() is declared which can be replaced with own method declaration  string sayHello();oneway  void shutdown();  };}; |
| --- |

| // HelloServer.java  import HelloApp.\*;  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 HelloImpl extends HelloPOA {  private ORB orb;  public void setORB(ORB orb\_val) {Ff  orb = orb\_val;  }  // implement sayHello() method this definition can be replaced with own method  public String sayHello() {  return "\nHello world !!\n";  }  // implement shutdown() method  public void shutdown() {  orb.shutdown(true);  }  }  public class HelloServer {  public static void main(String args[]) {  try {  // create and initialize the ORB  ORB orb = ORB.init(args, null);  // get reference to rootpoa & activate the POAManager  POA rootpoa = POAHelper.narrow(orb.resolve\_initial\_references("RootPOA"));  rootpoa.the\_POAManager().activate();  // create servant and register it with the ORB  HelloImpl helloImpl = new HelloImpl();  helloImpl.setORB(orb);  // get object reference from the servant  org.omg.CORBA.Object ref = rootpoa.servant\_to\_reference(helloImpl);  Hello href = HelloHelper.narrow(ref);  // get the root naming context  org.omg.CORBA.Object objRef = orb.resolve\_initial\_references("NameService");  // Use NamingContextExt which is part of the Interoperable  // Naming Service (INS) specification.  NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);  // bind the Object Reference in Naming  String name = "Hello";  NameComponent path[] = ncRef.to\_name(name);  ncRef.rebind(path, href);  System.out.println("HelloServer ready and waiting ...");  // wait for invocations from clients  orb.run();  } catch (Exception e) {  System.err.println("ERROR: " + e);  e.printStackTrace(System.out);  }  System.out.println("HelloServer Exiting ...");  }  } |
| --- |

| // HelloClient.java  import HelloApp.\*;  import org.omg.CosNaming.\*;  import org.omg.CosNaming.NamingContextPackage.\*;  import org.omg.CORBA.\*;  public class HelloClient {  static Hello helloImpl;  public static void main(String args[]) {  try {  // create and initialize the ORB  ORB orb = ORB.init(args, null);  // get the root naming context  org.omg.CORBA.Object objRef = orb.resolve\_initial\_references("NameService");  // Use NamingContextExt instead of NamingContext. This is  // part of the Interoperable naming Service.  NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);  // resolve the Object Reference in Naming  String name = "Hello";  helloImpl = HelloHelper.narrow(ncRef.resolve\_str(name));  System.out.println("Obtained a handle on server object: " + helloImpl);  System.out.println(helloImpl.sayHello());  helloImpl.shutdown();  } catch (Exception e) {  System.out.println("ERROR : " + e);  e.printStackTrace(System.out);  }  }  } |
| --- |

| start orbd -ORBInitialPort 1050 -ORBInitialHost localhost |
| --- |

| java HelloServer -ORBInitialPort 1050 -ORBInitialHost localhost |
| --- |

**Output**:

| PS C:\Users\JARVIS\OneDrive - Shri Vile Parle Kelavani Mandal\Desktop\DJSCE\SEM 7\Distributed Computing\Practical X4 -- CORBA> idlj -fall Hello.idl  PS C:\Users\JARVIS\OneDrive - Shri Vile Parle Kelavani Mandal\Desktop\DJSCE\SEM 7\Distributed Computing\Practical X4 -- CORBA> cd HelloApp  PS C:\Users\JARVIS\OneDrive - Shri Vile Parle Kelavani Mandal\Desktop\DJSCE\SEM 7\Distributed Computing\Practical X4 -- CORBA> javac \*.java |
| --- |

| PS C:\Users\JARVIS\OneDrive - Shri Vile Parle Kelavani Mandal\Desktop\DJSCE\SEM 7\Distributed Computing\Practical X4 -- CORBA> java HelloServer -ORBInitialPort 1050 -ORBInitialHost localhost  HelloServer ready and waiting … |
| --- |

| PS C:\Users\JARVIS\OneDrive - Shri Vile Parle Kelavani Mandal\Desktop\DJSCE\SEM 7\Distributed Computing\Practical X4 -- CORBA>java HelloServer -ORBInitialPort 1050 -ORBInitialHost localhost  Hello World !!! |
| --- |