Praktikum 3
Distributed Object CORBA

A. Tujuan

1. Mampu memahami konsep CORBA
2. Mampu mengiplementasikan CORBA dalam pemrograman

B. Dasar Teori

Skema sistem komputasi tersebar secara umum diperlihatkan baik melalui teknologi DCOM, CORBA, maupun Java RMI, sesungguhnya merupakan middleware yang berada di atas sistem operasi dan berada di bawah aplikasi. Aplikasi -aplikasi (baik aplikasi disisi klien maupun aplikasi di sisiserver) saling berkomunikasi dengan middleware melalui antarmukanya.

CORBA (Common Object Request Broker Architecture) merupakan suatu spesifikasi yang dikembangkan oleh OMG (Object Management Group). Beberapa software yang mengimplementasi-kan CORBA misalnya ORBIX (oleh IONA Technologies), VisiBroker (oleh Insprise), dan JavaIDL (oleh JavaSoft). Keunikan dari CORBA adalah kemampuannya dalam menangani heterogenitas antara *client* dan *server* (dalam terminologi CORBA, obyek *server* dinamakan **implementasi obyek** *(object implementation)*. Keduanya dapat saja diimplementasikan dalam **hardware, sistem operasi, bahasa pemrograman, dan di lokasi yang berbeda,** tetapi tetap bisa saling berkomunikasi. Kuncinya ada pada sebuah lapisan software yang disebut dengan **ORB***(Object Request Broker)*.

CORBA pada dasarnya berbasis pada protokol yang dinamakan IIOP (Internet Inter-ORB Protocol)untuk berkomunikasi dengan objek-objek yang ada di komputer serveryang lain (remoting object) . Segala sesuatu yang ada di dalam arsitektur CORBA pada prinsip-nya bergantung pada ORB (Object Request Broker),di manaORB ini bertindak sebagai sesuatu yang menghubungkan (bertindak sebagai bus) masing-masing objek CORBA sehingga mereka dapat saling berinteraksi baiksecara lokal pada komputer yang sama (localhost) maupun berinteraksi secara jarak jauh (remote). Masing-masing objek CORBA yang bertindak sebagai servermemiliki antarmuka (interface) dan memiliki sejumlah method yang dapat dimanfaatkan oleh objek-objek CORBA yang lainnya. Untuk meminta layanan tertentu, klien CORBA bisa meminta rujukan objek tertentu ke komputer server. Selanjutnya klien dapat melakukan pemanggilan methodpada objek CORBA yang bertindak sebagai servermelalui rujukan objek sepertijika objek tersebut berada di ruang memori klien. Komponen-komponen ORB bertanggungjawab untuk menentukan implementasi objek yang mana yang diinginkan klien, bertanggungjawab untuk menyiapkan objek untuk menerima permintaan, berkomunikasi dengannya, dan bertanggungjawab untuk mengirimkan hasil eksekusi implementasi objek kembali ke arah klien. Sebuah objek CORBA berinteraksi dengan ORB melalui antarmuka ORB atau

melalui Object AdapterBOA (Basic Object Adapter) atau POA (Portable Object Adapter). DikarenakanCORBA merupakan spesifikasi, makapada dasarnya bisa digunakan di berbagai platform sistem operasi yang berbeda, mulai dari komputer-komputer besar (mainframe), komputer-komputer yang memiliki sistem operasi UNIX/Linux, hingga komputer-komputer dengan sistem operasi Windows.

Komponen Utama penyusun CORBA

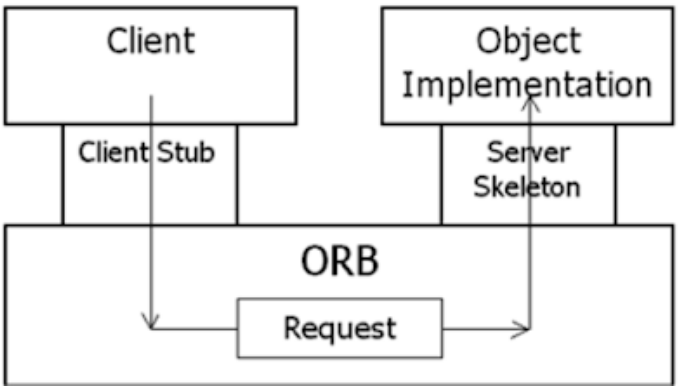
Object Request Broker (ORB)

ORB merupakan inti dari CORBA dan bertanggung jawab untuk menjalankan semua mekanisme yang dibutuhkan, yaitu:

- Menemukan implementasi objek untuk memenuhi suatu request
- Menyiapkan implementasi objek untuk menerima suatu request
- Melakukan komunkasi data untuk memenuhi suatu request

Sebuah permintaan (request) yang dikirimkan suatu client ke suatu object implementation akan melewati ORB. Dengan ORB, yang terdiri dari interface, suatu client dapat berkomunikasi dengan object implementation tanpa adanya batasan platform, teknologi jaringan, bahasa pemrograman, dan letak objek.

Dengan menggunakan ORB, objek client bisa meminta sebuah method pada sebuah object server yang bisa saja terdapat dalam satu mesin maupun jaringan yang berbeda. ORB menerima panggilan dan menemukan objek yang bisa mengimplementasikan permintaan, mengirim parameter, invoke method, dan mengembalikan hasil yang diperoleh.



Gambar : Proses Request

C. Praktikum

1. Instal JDK
2. tambah path yang berisi folder **idlj**
3. Buatlah direktori baru dilaptop masing-masing dengan nama **Corba**
4. Tulis file IDL didalam folder Corba dengan nama **EchoApp.idl**
   File IDL mendifinisikan interface yang akan digunakan oleh client dan server untuk komunikasi dan melewati objek

```
module EchoApp {
    interface Echo {
        string echoString();
    };
};
```

5. Generate code stub dan skeleton, dengan perintah berikut :
   `idlj -fall sum.idl`

   File berikut akan otomatis di generate :

   - _EchoStub.java
   - Echo.java
   - EchoHelper.java
   - EchoHolder.java
   - EchoOperations.java
   - EchoPOA.java

6. Tuliskan Program untuk server

   EchoServer.java

```
import EchoApp.EchoPOA;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.ArrayList;
import java.util.concurrent.atomic.AtomicInteger;

public class EchoServer extends EchoPOA {
    final AtomicInteger runningCount = new AtomicInteger(0);
    final Integer limit = 1;
    final String ErrorMessage = "limit exceeded";
```

```
    @Override
    public String echoString() {
        if (runningCount.incrementAndGet() <= limit) { //increment
number of clients and check

            doSomeThing();
            runningCount.decrementAndGet();
            return "Hello World!!!!!!!";
        } else {
            runningCount.decrementAndGet();
            System.out.println(ErrorMessage);
            return ErrorMessage;
        }
    }
    private void doSomeThing() {
        try {
            for (int i = 1; i < 5; i++) {
                Thread.sleep(i * 1000);
                System.out.println("sleep " + i);
            }
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

Server.java

kelas ini digunakan untuk komunikasi dengan Object Request Broker (ORB), mendaftarkan server pada ORB sehingga klien dapat menemukan server.

```
import EchoApp.Echo;
import EchoApp.EchoHelper;
import org.omg.CORBA.ORB;
import org.omg.CosNaming.NameComponent;
import org.omg.CosNaming.NamingContextExt;
import org.omg.CosNaming.NamingContextExtHelper;
import org.omg.PortableServer.POA;
import org.omg.PortableServer.POAHelper;

public class Server {
    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
            EchoServer server = new EchoServer();

            // get object reference from the servant
            org.omg.CORBA.Object ref =
rootpoa.servant_to_reference(server);
            Echo href = EchoHelper.narrow(ref);

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

            NameComponent path[] = ncRef.to_name( "ECHO-SERVER" );
            ncRef.rebind(path, href);

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

    }
}
```

7. Tuliskan Program Client

Client.java

```
import EchoApp.Echo;
import EchoApp.EchoHelper;
import org.omg.CORBA.ORB;
import org.omg.CORBA.ORBPackage.InvalidName;
```

```
import org.omg.CosNaming.NamingContextExt;
import org.omg.CosNaming.NamingContextExtHelper;
import org.omg.CosNaming.NamingContextPackage.CannotProceed;
import org.omg.CosNaming.NamingContextPackage.NotFound;

public class Client {
    public static void main(String args[]) {
        try {
            // create and initialize the ORB
            ORB orb = ORB.init(args, null);
            org.omg.CORBA.Object objRef =
orb.resolve_initial_references("NameService");
            NamingContextExt ncRef =
NamingContextExtHelper.narrow(objRef);
            Echo href =
EchoHelper.narrow(ncRef.resolve_str("ECHO-SERVER"));

            String hello = href.echoString();
            System.out.println(hello);
        } catch (InvalidName invalidName) {
            invalidName.printStackTrace();
        } catch (CannotProceed cannotProceed) {
            cannotProceed.printStackTrace();
        } catch
(org.omg.CosNaming.NamingContextPackage.InvalidName invalidName)
{
            invalidName.printStackTrace();
        } catch (NotFound notFound) {
            notFound.printStackTrace();
        }
    }
}
```

8. Compile stub dan skeleton dari direktori yang berisi file IDL
   `javac EchoApp/*.java` dan
   `javac -Xlint *.java EchoApp/*.java`

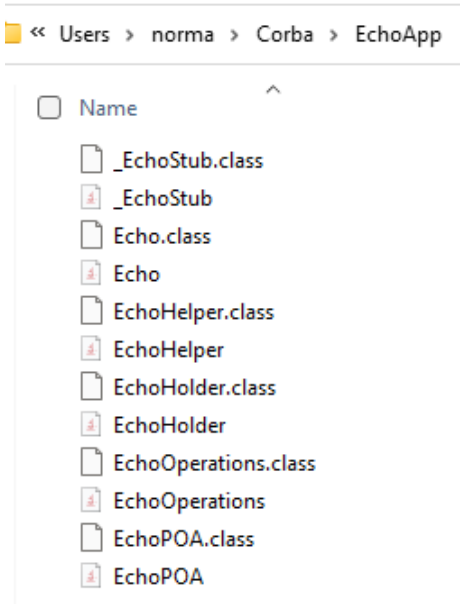
```
C:\Users\norma>cd Corba

C:\Users\norma\Corba>idlj -fall EchoApp.idl

C:\Users\norma\Corba>javac *.java EchoApp/*.java
Note: EchoApp\EchoPOA.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

C:\Users\norma\Corba>javac -Xlint *.java EchoApp/*.java
EchoApp\EchoPOA.java:17: warning: [rawtypes] found raw type: Hashtable
  private static java.util.Hashtable _methods = new java.util.Hashtable ();
                              ^
  missing type arguments for generic class Hashtable<K,V>
```

folder yang ter generate :



9. Generate sebuah file jar dari stub dan skeleton yang dikompilasi
   `jar cvf echoapp.jar EchoApp\*.class`

```
C:\Users\norma\Corba>jar cvf echoapp.jar EchoApp\*.class
added manifest
adding: EchoApp/Echo.class(in = 195) (out= 154)(deflated 21%)
adding: EchoApp/EchoHelper.class(in = 2419) (out= 1112)(deflated 54%)
adding: EchoApp/EchoHolder.class(in = 855) (out= 437)(deflated 48%)
adding: EchoApp/EchoOperations.class(in = 161) (out= 127)(deflated 21%)
adding: EchoApp/EchoPOA.class(in = 2093) (out= 1043)(deflated 50%)
adding: EchoApp/_EchoStub.class(in = 2490) (out= 1240)(deflated 50%)
```

10. Compile server dan client
    `javac -classpath .;echoapp.jar Server.java EchoServer.java Client.java`

```
C:\Users\norma\Corba>javac -classpath ./echoapp.jar Server.java EchoServer.java Client
.java

C:\Users\norma\Corba>
```

11. Run Aplikasi dengan tahapan berikut :
    - Start server ORB

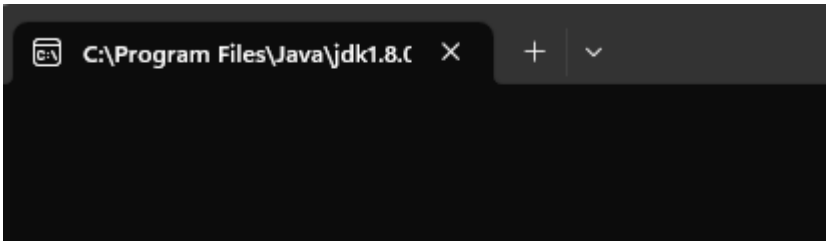
      `start orbd -ORBInitialPort 1050`

      `orbd -ORBInitialPort 1050 -ORBInitialHost localhost`

```
C:\Users\norma\Corba>start orbd -ORBInitialPort 1050

C:\Users\norma\Corba>
```

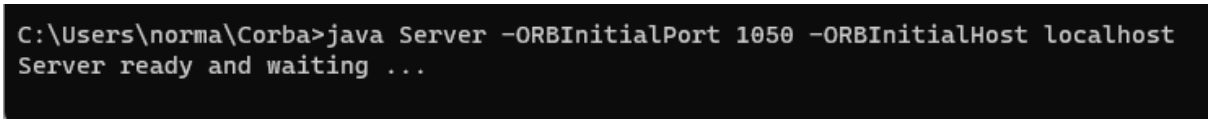
port 1050 dipilih untuk naming service, port dapat diganti port lain jika port 1050 telah dipakai, jika port tidak dituliskan pada perintah maka port standar yang diguankan adalah 900. jika berhasil muncul jendela berikut :




```
Command Prompt
C:\Users\norma\Corba>orbd -ORBInitialPort 1050 -ORBInitialHost localhost
Feb 27, 2023 10:06:50 AM com.sun.corba.se.impl.transport.SocketOrChannelAcceptorImpl i
nitialize
SEVERE: "IOP00410216: (COMM_FAILURE) Unable to create listener thread on the specified
 port: 1049"
org.omg.CORBA.COMM_FAILURE:   vmcid: SUN  minor code: 216  completed: No
        at com.sun.corba.se.impl.logging.ORBUtilSystemException.createListenerFailed(O
RBUtilSystemException.java:2632)
        at com.sun.corba.se.impl.logging.ORBUtilSystemException.createListenerFailed(O
RBUtilSystemException.java:2651)
        at com.sun.corba.se.impl.transport.SocketOrChannelAcceptorImpl.initialize(Sock
etOrChannelAcceptorImpl.java:164)
        at com.sun.corba.se.impl.transport.CorbaTransportManagerImpl.getAcceptors(Corb
aTransportManagerImpl.java:218)
        at com.sun.corba.se.impl.transport.CorbaTransportManagerImpl.addToIORTemplate(
CorbaTransportManagerImpl.java:236)
        at com.sun.corba.se.spi.oa.ObjectAdapterBase.initializeTemplate(ObjectAdapterB
ase.java:122)
        at com.sun.corba.se.impl.oa.toa.TOAImpl.<init>(TOAImpl.java:96)
        at com.sun.corba.se.impl.oa.toa.TOAFactory.getTOA(TOAFactory.java:90)
        at com.sun.corba.se.impl.orb.ORBImpl.connect(ORBImpl.java:1633)
        at com.sun.corba.se.impl.activation.RepositoryImpl.<init>(RepositoryImpl.java:
94)
```

- Start aplikasi server

```
java Server -ORBInitialPort 1050 -ORBInitialHost localhost
```



- Start aplikasi client

  ```
  java Client -ORBInitialPort 1050 -ORBInitialHost localhost
  ```

  buka CMD baru dan ketikkan perintah berikut :