

<b>MK</b>	<b>:</b>	<b>DSAD (PASTI)</b>
<b>Week</b>	<b>:</b>	<b>09</b>
<b>Session</b>	<b>:</b>	<b>02</b>
<b>Topic</b>	<b>:</b>	<b>Developing simple CORBA applications</b>
<b>Due</b>	<b>:</b>	<b>-</b>

Gunakanlah *editor* yang membantu anda belajar, semacam **Netbeans** dan **Eclipse**. Penggunaan *basic editor* tidak akan mengakselerasi anda sama sekali dan berpotensi 'menyesatkan'.

## Introduction

**CORBA** merupakan standar yang diterbitkan **OMG** untuk menjamin interoperabilitas sistem yang heterogen. Pada dasarnya, konsep **CORBA** sangat mirip dengan **RMI**, perbedaannya **RMI** *platform-specific*, yakni hanya dikembangkan untuk sistem berbasis **Java** sementara **CORBA** lebih general.

**Java** sendiri, sejak **JDK 1.2** sudah memberikan implementasi dari standar **CORBA**. Implementasi ini relatif lebih stabil daripada implementasi lainnya sekalipun pada **JDK 1.6** implementasi tersebut tidak di-*update*.

Untuk mengembangkan aplikasi berbasis **CORBA**, anda perlu melakukan langkah-langkah sebagai berikut:

1. Mendesain *interface (IDL)* dari yang pada akhirnya menyediakan fungsionalitas bagi pihak lain.
2. Men-*generate code* yang sesuai dengan bahasa pemrograman yang akan dipergunakan, dalam hal ini **Java**.
3. Mengimplementasi *interface* dan *server-side application*.
4. Mengimplementasi *client-side application*.
5. Menjalankan aplikasi.

Pada praktikum kali ini anda akan mengembangkan dua aplikasi berbeda, yakni **HelloApp** dan **MathApp**. Aplikasi pertama ditujukan untuk memperkenalkan kepada anda dasar-dasar pengembangan aplikasi berbasis CORBA, sementara aplikasi kedua ditujukan untuk mempertajam sense anda terkait pengembangan berbasis CORBA.

## Preparation

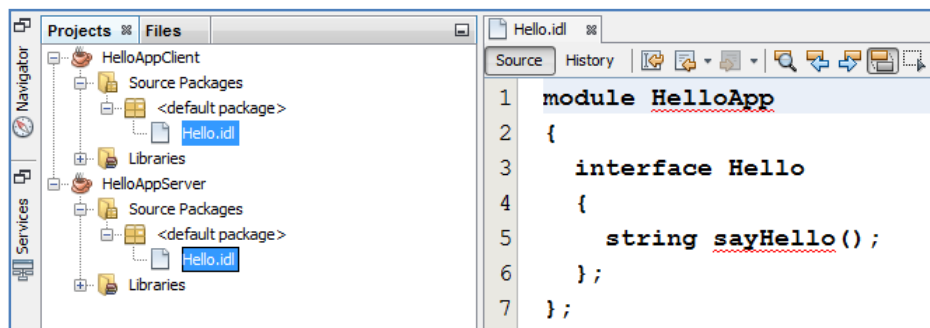
1. Dengan menggunakan NetBeans, buatlah dua buah Java Application baru dengan nama **HelloAppServer** dan **HelloAppClient**.

## Designing interface (IDL)

2. Pada kedua **Java Application**, buatlah sebuah *Empty File* dengan nama `Hello.idl`, tuliskan kode berikut:

```
module HelloApp
{
    interface Hello
    {
        string sayHello();
    };
};
```

3. Sehingga struktur direktori tampak seperti berikut:



## Generating language-specific code

4. Melalui **CLI** (*command line interface*), navigasi ke direktori di mana *file idl* berada (**HelloAppServer**), sebagai contoh, **NetBeans project** berada pada pada:

```
D:\courses\2014-2\PASTI\03. Materi Praktikum\w09s02p\HelloAppServer
```

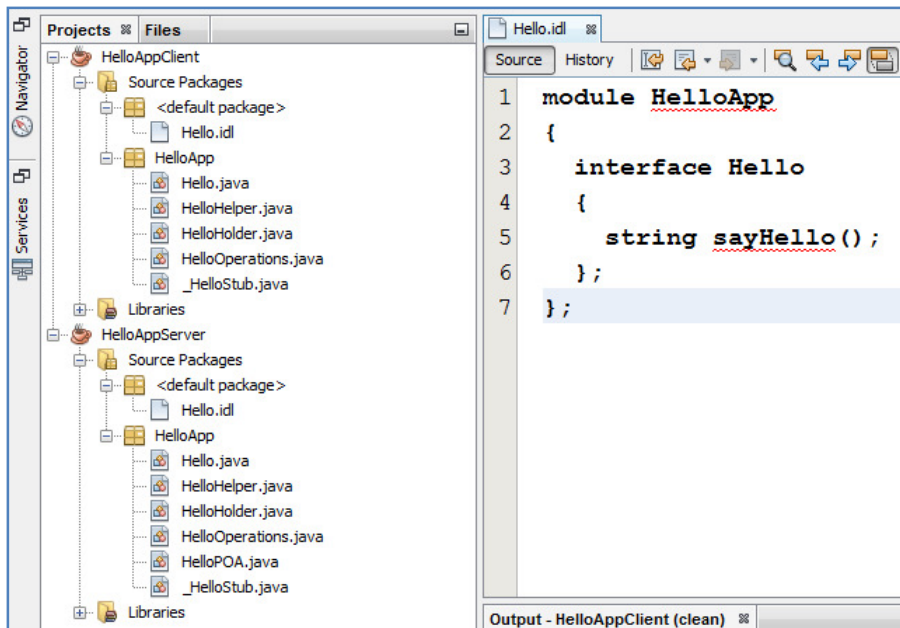
dengan demikian, `Hello.idl` berada pada subdirektori `src`.

5. Eksekusi perintah berikut untuk menghasilkan **Java code** yang diperlukan untuk mengembangkan *server-side application*.

```
idlj -fall Hello.idl
```

6. Dengan menggunakan prosedur yang sama, navigasi ke **HelloAppClient** dan eksekusi perintah berikut untuk menghasilkan **Java code** yang diperlukan untuk mengembangkan *client-side application*.

```
idlj -fclient Hello.idl
```



## Implementing interface

7. Pada **HelloAppServer** buatlah sebuah kelas dengan nama **HelloImpl**, tuliskan kode berikut:

```

import HelloApp.*;
import org.omg.CORBA.*;

public class HelloImpl extends HelloPOA {
    private ORB orb;
    public void setORB(ORB orb_val) {
        orb = orb_val;
    }

    // implement sayHello() method
    public String sayHello() {
        return "\nHello world !\n";
    }
}

```

## Developing server-side application

8. Buatlah sebuah kelas baru dengan nama **HelloServer**, lalu tuliskan kode berikut:

```

import HelloApp.*;
import org.omg.CosNaming.*;
import org.omg.CORBA.*;
import org.omg.PortableServer.*;
import org.omg.PortableServer.POA;

```

```

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 ...");
    }
}

```

## Developing client-side application

9. Pada *project HelloAppClient*, buatlah sebuah kelas baru dengan nama `HelloClient`, dan tuliskan kode berikut:

```

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.
        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: " + helloImpl);
        System.out.println(helloImpl.sayHello());
    } catch (Exception e) {
        System.out.println("ERROR : " + e);
        e.printStackTrace(System.out);
    }
}
}

```

## Run the application

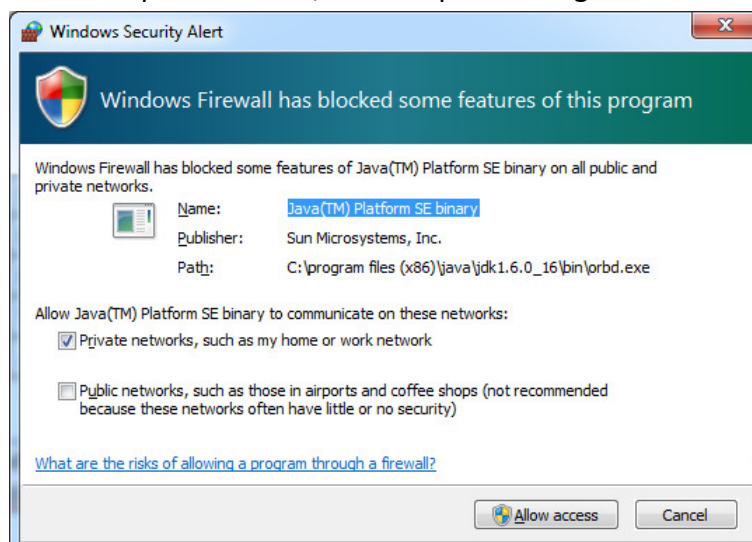
10. Hal pertama yang harus dilakukan adalah mengaktifasi **ORBD (ORB Daemon)**. Untuk melakukan hal ini, buka sebuah **CLI** lalu eksekusi perintah dengan format berikut:

```
start orbd -ORBInitialPort <PORT NUMBER> -ORBInitialHost <HOST ADDRESS>
```

pada praktikum ini, gunakan perintah:

```
start orbd -ORBInitialPort 1050 -ORBInitialHost localhost
```

Secara *default*, sistem operasi **Windows** memblokir **ORBD** untuk dieksekusi karena membuka *port* tertentu, beri izin pada dialog berikut:



11. Tahap berikutnya adalah mem-*build* **NetBeans projects**, klik kanan pada **HelloAppServer** lalu pilih **Build**, lakukan hal yang sama dengan **HelloAppClient**.

12. Buka **CLI** baru kemudian navigasi ke direktori di mana *project* **HelloAppServer** berada, masuk ke subdirektori **build\classes**. Kemudian jalankan **HelloServer** dengan perintah:

```
start java HelloServer -ORBInitialPort 1050 -ORBInitialHost localhost
```

atau anda dapat mengeksekusi *jar file* yang dihasilkan.

13. Dengan mengikuti prosedur yang sama dengan poin di atas, jalankan *clent* (**HelloAppClient**) dengan perintah berikut:

```
java HelloClient -ORBInitialPort 1050 -ORBInitialHost localhost
```

## Observation & Experimentation

Lakukanlah observasi dari *project* yang anda buat dan silakan melakukan *cross-referencing* dengan dokumen yang diberikan *pada self-study* yang lalu.

Lakukan perubahan pada perintah eksekusi dengan menuliskan *IP address* rekan anda untuk dapat mengakses **CORBA object** secara *remote*.

## Challenge

Anda diminta untuk mengimplementasi IDL **MathApp** berikut sehingga memberikan fungsionalitas penambahan (**add**) dan pengurangan (**subtract**):

```
module MathApp {  
    interface Math {  
        short add(in short a, in short b);  
        short subtract(in short a, in short b);  
    };  
};
```

-EOF-