TD 3 Common Object Request Broker Architecture (CORBA)

Supposant qu'on veut créer un serveur CORBA qui crée un objet qui offre le service distant suivant à un client :

- ❑ Envoyer au client un simple message de "hello world.

- ✓ Nous allons travailler avec le service CORBA gratuit offre par Java, il s'appelle ORBD.
- ✓ Nous allons faire communiquer un client et un serveur écrits tous les deux avec Java (malgré qu'on peut en principe faire communiquer deux programmes écrits en des langages différents).

1. Interface en langage IDL

Pour commencer, il faut écrire un fichier d'interface en langage IDL (Interface Definition Language) qui décrit les prototypes des méthodes de l'objet distant (on décrit le nom de la méthode et les types des paramètres et du résultat).

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

2. Complication du fichier IDL

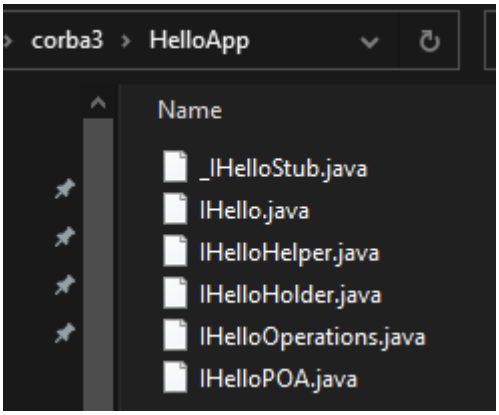
Le compilateur **`idlj`** génère des liaisons Java™ depuis un fichier IDL. nous tapons sur l'invite de commande :

idlj -fall Hello.idl

2.1 Fichiers générés :

Le dossier **HelloApp** sera créé automatiquement selon du nom du module de IDL.

Ce dossier contient les fichiers suivants :



3. Fichiers à créer

Les trois fichiers que nous devons créer sont :

- **HelloImpl.java** (classe de l'objet distant qui hérite HelloPOA en implémentant HelloOperations)
- **HelloServeur.java** (classe du serveur qui crée un objet HelloImpl et l'inscrit au bus CORBA)
- **HelloClient.java** (classe du client)

Les trois fichiers sont placés dans le dossier **HelloApp**

3.1 Classe de l'objet distant (coté serveur): HelloImpl.java

```
package HelloApp;
public class HelloImpl extends
HelloApp.IHelloPOA{
    public String sayHello(){
    return "\n Hello World\n";
    }
```

3.2 Classe du Client: HelloClient.java

```
package HelloApp;
import org.omg.CORBA.*;
import org.omg.CosNaming.*;

public class HelloClient{
public static void main(String[]args)throws Exception{

// Initialisation de l'ORB
ORB orb=ORB.init(args,null);

// Récupérer la référence du service de nommage
org.omg.CORBA.Object nsRef = orb.resolve_initial_references("NameService");
NamingContextExt nce = NamingContextExtHelper.narrow(nsRef);

// Générer un objet Stub par une recherche dans le service de nommage
String serviceName = "HelloServices";
IHello hRef = IHelloHelper.narrow(nce.resolve_str(serviceName));

// Appeler la méthode distante
System.out.println("Reponse du serveur : " + hRef.sayHello());

}
}
```

3.3 Classe du Serveur: HelloServeur.java

```
package HelloApp;
import org.omg.CORBA.*;
import org.omg.PortableServer.*;
import org.omg.CosNaming.*;

public class HelloServeur{
public static void main(String[]args)throws Exception{

// Initialisation de l'ORB
ORB orb=ORB.init(args,null);

//Récupérer la référence du RootPOA et activer le POAManager
POA rootpoa = POAHelper.narrow(orb.resolve_initial_references("RootPOA"));
rootpoa.the_POAManager().activate();

//création de l'objet distant
HelloImpl helloImpl= new HelloImpl();

//Conversion (narrow) de l'objet en sa interface
org.omg.CORBA.Object ref = rootpoa.servant_to_reference(helloImpl);
IHello href = IHelloHelper.narrow(ref);

// récupération du service de nommage
org.omg.CORBA.Object refNServ =orb.resolve_initial_references("NameService");
NamingContextExt nce =NamingContextExtHelper.narrow(refNServ);

// Inscrire l'interface de l'objet dans le service de nommage "HelloServices"
String serviceName = "HelloServices";
NameComponent nc[] = nce.to_name(serviceName);
nce.rebind(nc, href);

// Démarrer le service et attendre les requêtes des clients
System.out.println("On traite les requetes des clients ...");
orb.run();

}
}
```

4. Compilation

javac -cp . HelloApp/HelloServeur.java

javac -cp . HelloApp/HelloClient.java

5. Lancement

- Lancement du service ORBD :

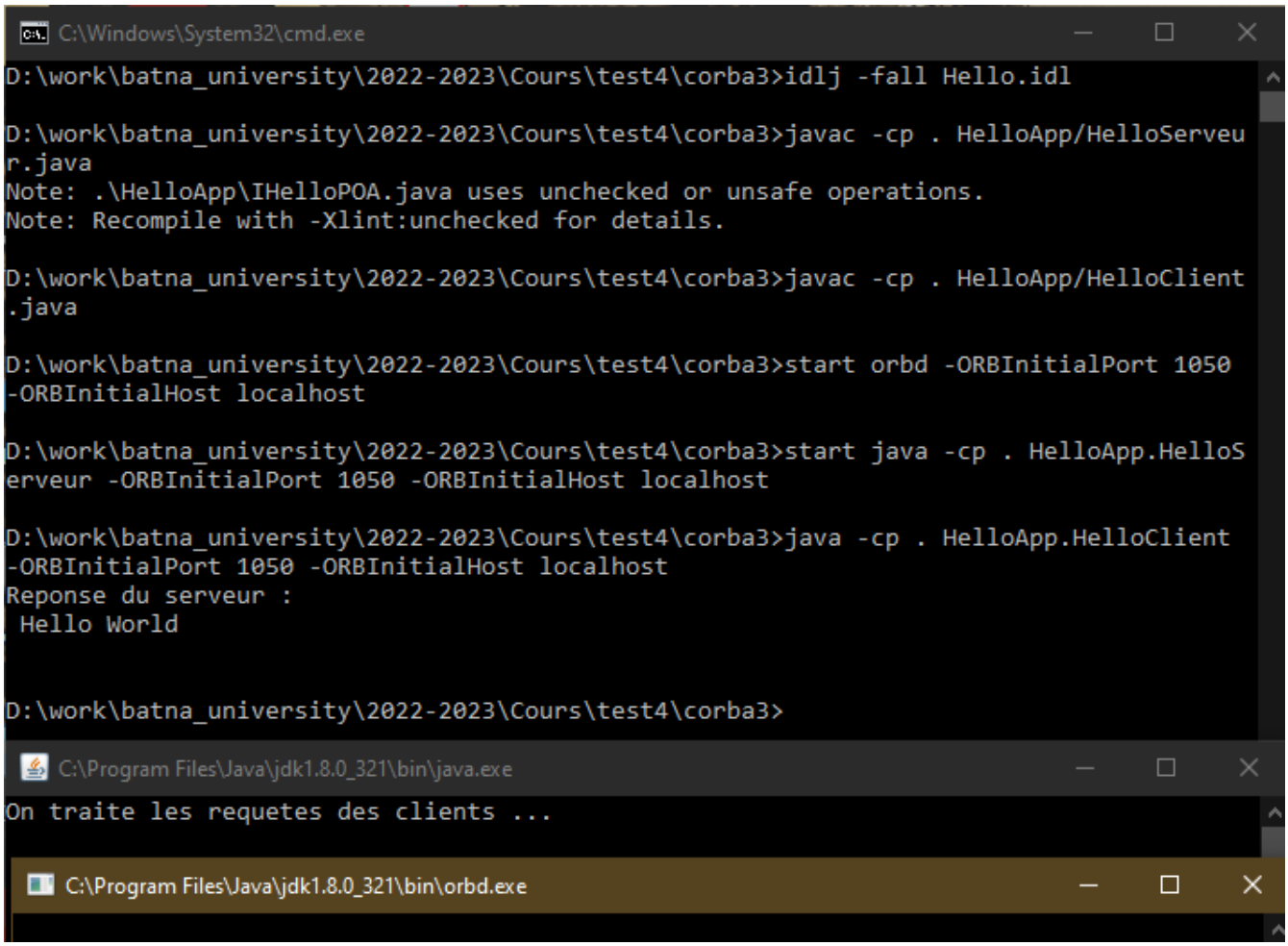
start orbd -ORBInitialPort 1050 -ORBInitialHost localhost

- Lancement du serveur :

start java -cp . HelloApp.HelloServeur -ORBInitialPort 1050 -ORBInitialHost localhost

- Lancement du client :

java -cp . HelloApp.HelloClient -ORBInitialPort 1050 -ORBInitialHost localhost



6. TP 3 à domicile

Créez une application CORBA qui permettre à un client d'invoquer une méthode d'un objet distant et récupérer le résultat de calcul de la factorielle d'un nombre fourni par le client.

Question de Bonus :

Refaire le TP en écrivant le client en langage C++ et en liant au serveur CORBA OMNIORB au lieu d'ORBD.

Vous pouvez utiliser les commandes IDL Compiler pour C++ telles que définies dans :

https://cis.temple.edu/~giorgio/cis307/software/mico/doc/doc/node47.html

Remarque**: Les TPs à faire à domicile et à rendre à bentahar.info@aol.com avant le **04/12/2022.