Exemple CORBA

Objectif :
Calculer un montant TTC à partir d'un montant HT et un taux de TVA en utilisant l'architecture CORBA.

Etape 1: Developper l'interface en IDL

```
module calcul
{
interface calcul_montants
{
double calcul_ttc(in double mt_ht, in double taux);
};
};
```

Etape 2 : Générer les fichiers java client et serveur à partir de l'interface IDL :

```
C:\Program Files\Java\jdk1.8.0_301\bin> idlj -td "C:\Users\Rabaa\eclipse-workspace\CorbaTest\src" -fall "C:\Users\Rabaa\eclipse-workspace\CorbaTest\src\spec.idl"
```

Cette commande génère les fichiers suivants :

- _calcul_montantsStub : Stub de l'objet calcul_montants (côté client).
- calcul_montants : Interface Java de l'interface IDL spec.idl
- calcul_montantsOperations : Interface des méthodes de l'objet Calcul
- calcul_montantsPOA : Squelette de l'objet calcul_montants (côté serveur)
- calcul_montantsHelper : Le Helper de l'objet calcul_montants
- calcul_montantsHolder : Le Holder de calcul_montants

Etape 3 : Créer le servant (la classe qui fait le calcul) :

```
package calcul;
import org.omg.CosNaming.*;
//inclure le package des exceptions pouvant etre generees
// par le service de nommage
import org.omg.CosNaming.NamingContextPackage.*;
// sert a manipuler les objets CORBA
import org.omg.CORBA.*;
// Classes necessaires pour referencer le POA
import org.omg.PortableServer.*;
import org.omg.PortableServer.POA;
// Proprietes pour initialiser l'ORB
import java.util.Properties;
public class Calcul_Servant extends calcul_montantsPOA {
	public Calcul_Servant() {
	}
	public double calcul_ttc(double mt_ht, double taux) {
		return mt_ht*(1+taux/100);
	}
}
```

Etape 4 : Créer le serveur :

```
package calcul;
// le serveur va utiliser le service de nommage
import org.omg.CosNaming.*;
//inclure le package des exceptions pouvant etre generees
// par le service de nommage
import org.omg.CosNaming.NamingContextPackage.*;
// sert a manipuler les objets CORBA
import org.omg.CORBA.*;
// Classes necessaires pour referencer le POA
import org.omg.PortableServer.*;
import org.omg.PortableServer.POA;
// Proprietes pour initialiser l'ORB
import java.util.Properties;
public class Serveur {
	public Serveur() {
	}
	public static void main(String args[]) {
		try {
			// creer et initialiser l'ORB qui integre
			// le service de noms
			ORB orb=ORB.init(args, null);
			// obtenir la reference de rootpoa &
			// activer le POAManager
			POA rootpoa =

	POAHelper.narrow(orb.resolve_initial_references("RootPOA"));
			rootpoa.the_POAManager().activate();
			// creer le servant
			Calcul_Servant calc= new Calcul_Servant();
			// obtenir la reference CORBA du servant
			org.omg.CORBA.Object ref = rootpoa.servant_to_reference(calc);
			calcul_montants href = calcul_montantsHelper.narrow(ref);
			// obtenir la reference du contexte de nommage
			org.omg.CORBA.Object objRef =
					orb.resolve_initial_references("NameService");
			// Utiliser NamingContextExt qui est Interoperable
			NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);
			// enregistrer le servant dans le service de nommage
			String name = "calcul_ttc";
			NameComponent path[] = ncRef.to_name( name );
			ncRef.rebind(path, href);
			System.out.println("Server is waiting");
			//attendre les invocations des clients
			orb.run();
		} catch(Exception e) {
			System.err.println("Erreur : "+e);
			e.printStackTrace(System.out);
		}
	}
}
```

Etape 5 : Créer le client :

```
package calcul;
import org.omg.CosNaming.*; // inclure le service de nommage
import org.omg.CORBA.*; // manipuler des objets CORBA
import org.omg.CosNaming.NamingContextPackage.*;
public class Client {
	public Client() {
	}
	public static void main (String args[]) {
		try {
			double mt_ht;
			double taux;
			double mt_ttc;
			mt_ht = Double.valueOf(args[0]);
			taux = Double.valueOf(args[1]);
			// creer et initialiser l'ORB
			ORB orb = ORB.init(args, null);
			// obtenir une reference au service de nommage
			org.omg.CORBA.Object objRef =
				orb.resolve_initial_references("NameService");
			// Utiliser NamingContextExt au lieu de NamingContext.
			//car interoperable
			NamingContextExt ncRef =
				NamingContextExtHelper.narrow(objRef);
			// demander la reference de l'objet au service de noms
			String nom = "calcul_ttc";
			calcul_montants calcul_ttc = calcul_montantsHelper.narrow
				(ncRef.resolve_str(nom));
			// faire appel a l'objet serveur et imprimer les resultats
			mt_ttc = calcul_ttc.calcul_ttc(mt_ht,taux);
			System.out.println("le montant ttc "+ mt_ttc);
		}
		catch(Exception e) {
			System.out.println("Erreur : "+e);
			e.printStackTrace(System.out);
		}
	} // fin du main
  }
```

Etape 6 : Demarrer l'ORB sur la machine serveur

```
start orbd -ORBInitialPort 1500
```

Etape 7 : Lancer le serveur :

```
java calcul.Serveur -ORBInitialPort 1500
```

Etape 8 : Executer le client :

```
java calcul.Client 100 20 –ORBInitialHost localhost –ORBInitialPort 1500
```

Autres exercices CORBA

EXO 1 :

Le premier exercice consiste en un exemple simple d'application client/serveur CORBA. L'objet hébergé par le serveur est constitué de deux méthodes (`incrementer` et `decrementer`) définies dans l'interface IDL `calcul` (fichier `server.idl`). Dans cet exercice, vous devez compiler et exécuter l'application. Pour ce faire, il faut :

- Créez un projet **tpcorba** sous eclipse et ajoutez-y le fichier **server.idl**
- Depuis le répertoire du projet, lancez la commande `idlj` (avec les bons arguments) pour générer les souches et squelettes. Le code Java généré par le compilateur IDL est placé dans le répertoire `tpcorba\src\exo1`. Les fichiers `.class` sont placés dans le répertoire `tpcorba\bin\exo1`.
- Récupérez tous les différents fichiers de cet exercice qui vous sont fournis. **Vous les stockerez tous dans le répertoire `exo1`. Pour une compilation correcte de ce programme, il est important de respecter les noms des répertoires**.

- Lancer le serveur.
- Enfin, lancer le client en indiquant le nombre sur lequel le serveur doit effectuer le calcul. Attention : le serveur écrit sur disque la référence de l'objet qu'il gère (fichier `calcul.ref`), il est donc nécessaire de lancer le client et le serveur **dans le même répertoire**.
- Faites un refresh de votre projet, vous verrez le fichier `calcul.ref` qui vous permet de **sauvegarder la référence d'objet**.

On vous demande de regarder quelles sont les étapes que le serveur et le client réalisent respectivement, pour initialiser l'objet CORBA, et pour invoquer les méthodes de l'objet. Vous regarderez plus précisément les points suivants :

1. Quelles sont les interactions entre le serveur et l'OA ?

2. Quelle est la relation entre la classe d'implémentation (classe `calculImpl.java`) et le squelette (le squelette est défini par la classe `calculPOA.java` dans le répertoire `tpcorba\src\exo1`) ?
3. Dans la souche (classe `_calculStub.java`), où se trouvent la construction et l'émission de la requête vers le serveur ?
4. Chercher dans le squelette (classe `calculPOA.java`) où se situent les appels à l'implémentation de l'objet (classe `calculImpl.java`).

EXO 2 :

Reprendre l'exemple de l'EXO1 en utilisant cette fois-ci non pas la méthode de sauvegarde des références d'objet dans un fichier IOR mais la méthode d'enregistrement dans le naming service (local).

Vous pouvez vous baser sur le code du TP « CORBA_exemple » pour modifier les fichiers **Serveur.java** et **Client.java** (Utiliser `NamingContextExt` à la place de `NamingContext`)