TP 3 – CORBA

Kenneth VANHOEY [1]
https ://dpt-info.u-strasbg.fr/~kvanhoey

1 Carré

1. Rendez-vous dans le répertoire *1-Carre/*. Celui-ci contient l'interface IDL d'un service de calcul du carré d'un nombre.
2. Réaliser la projection de l'interface IDL :

   ```
   idlj -fall -oldImplBase Icarre.idl
   ```

   Notez tous les fichiers **.java* qui sont génères automatiquement ; puis lancez la compilation complète par **./compile**.
   Important : ouvrez les fichiers *Icarre.java* et *IcarreOperations.java*. C'est ce dernier fichier **Operations.java* qui vous donne l'interface Java que vous devrez utiliser pour votre implémentation. À chaque fois que vous écrirez ou utiliserez une interface IDL, il faudra regarder ce fichier.
3. Lancez l'application en commençant par le serveur :

   ```
   java Serveur
   ```

   qui normalement affiche son *IOR* (Interoperable Object Reference).
   Attention : la chaine « *IOR :* » fait partie de l'IOR du serveur.
   Puis lancer le client :

   ```
   java Client <IOR_du_serveur> <nombre>
   ```

   Question (à répondre dans reponse.txt) : Comment le Client arrive-t-il à savoir sur quelle machine se trouve le Serveur ?
4. N'oubliez pas de tuer le serveur !

2 Paramètres et Holders

1. Recopier *1-Carre/* dans un nouveau répertoire que vous appelerez *2-Carre_Holder* et référez-vous au cours/TD pour cet exercice (paramètres de type *in*, *out* ou *inout* et les classes *Holder*).
2. Questions (à répondre dans reponse.txt) :
   – Comment génère-t'on une classe Holder ?
   – Pour quels types de paramètres a-t'on besoin d'une classe Holder ?
3. Modifier la fonction « *long carre(in long source)* » afin de faire passer le résultat de la fonction en second paramètre. La nouvelle signature de la fonction devient donc « *void carre(in long source, out long resultat)* ».
   Pour cela vous interviendrez sur les codes de : *Icarre.idl*, puis *IcarreImpl.java*, puis *Client.java*.

1. Sujet créé à partir de documents de Guillaume LATU

3 POA

Précédemment, nous avons utilisé le paramètre **-oldImplBase** à la compilation avec **idlj** et avons ainsi généré une interface CORBA compatible avec des ORB d'avant la JDK 1.4. Depuis cette version, cela se passe différement, grâce aux POA (Portable Object Adaptor).

1. Recopiez *1-Carre/* dans un nouveau répertoire que vous appelerez *3-Carre_POA* et référez-vous au cours/TD pour cet exercice.
2. On va utiliser le POA dans cette nouvelle version. Par rapport à l'exercice 1, un itermédiaire (le POA) se place entre le squelette de l'objet distant et l'ORB au niveau du serveur. Ce POA normalisé permet de faire interopérer le serveur avec différents ORBs (pas uniquement celui de java SUN).

   Vous modifierez le fichier *compile* en enlevant l'option **-oldImplbase** lorsque vous appelez **idlj**.

   Vous ferez attention que la projection IDL vers java change alors (le squelette généré n'est plus *_IcarreImplBase* mais *IcarrePOA*). Les fichiers à modifier sont alors : *IcarreImpl.java* et *Serveur.java* (veillez à importer *PortableServer*).

4 MatricesNN, le retour

Rendez-vous dans le répertoire *4-MatricesNN/*. Le fichier *OpMatrice.idl* définit l'interface IDL d'un service proposant la multiplication de deux matrices. Projetez l'interface IDL en Java. Regarder en quel type Java est projeté le type IDL « *long* ».

Proposer une implémentation CORBA de ce service équivalente à ce que l'on a fait précedemment avec RMI dans le TP2 (vous avez à créer les fichiers *OpMatriceImpl.java*, *Client.java* et *Serveur.java*). Cette fois-ci on pourra multiplier des matrices rectangulaires quelconques : « A » de dimension $n \times k$ et « B » de dimension $k \times m$. Vous utiliserez obligatoirement le POA.

Votre Client demandera au service distant de multiplier les deux matrices A et B telles que :

```
int[][] A = { {1, 0, 0, 0}, {0, 3, 0, 0}, {0, 1, 3, 0}, {0, 4, 0, 7} } ;
int[][] B = { {1, 4}, {1, 8}, {2, 3}, {1, 0} } ;
```

Vous devez bien sur trouver : $res = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 1 & 3 & 0 \\ 0 & 4 & 2 & 7 \end{bmatrix} \begin{bmatrix} 1 & 4 \\ 1 & 8 \\ 2 & 3 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 4 \\ 3 & 24 \\ 7 & 17 \\ 11 & 32 \end{bmatrix}$

Remarque pour éviter de faire un copier/coller de l'IOR à partir du shell, on peut écrire l'IOR dans un fichier avec le Serveur puis le lire depuis le Client. À cet effet, une constante « *iorfile* » contenant le nom du fichier où l'on va stocker l'IOR est définie dans le fichier IDL.

Dans le serveur pour écrire l'IOR dans un fichier, utilisez le code :

```
import java.io.* ;
...
FileOutputStream file = new FileOutputStream(iorfile.value) ;
PrintWriter out = new PrintWriter(file) ;
out.println(ior) ; out.flush() ;
file.close() ;
```

Dans le client pour lire l'IOR dans un fichier, utilisez le code :

```
import java.io.*;
...
FileReader file = new FileReader(iorfile.value) ;
BufferedReader in = new BufferedReader(file) ;
ior = in.readLine() ;
file.close() ;
```

5 Annuaire

Rendez-vous dans le répertoire *5-Annuaire/*. Le fichier *Annuaire.idl* définit l'interface IDL d'un service d'annuaire. Proposer une implémentation CORBA de ce service équivalente à celle vue avec RMI. Pour cela, vous utiliserez le POA. Pour démarrer, vous pouvez utiliser la correction de l'exercice RMI dans le repertoire *TelephoneExerciceRMI/*.