CORBA Mars 2015 TP GMIN 204 M2

Nous allons travailler avec Java, sous Linux, et utiliser l'ORB fourni en standard avec le JDK. Le TP consiste d'abord à étudier l'exemple classique du `helloWorld`, puis à créer une petite application CORBA. Le compilateur IDL vers java s'appelle `idlj`. Le service de nommage est inclu dans le démon `orbd` (Object Request Broker Daemon) .

1 Hello world

Recopiez et désarchivez l'archive corba.tar disponible via http ://www.lirmm.fr/∼seriai/ Vous y trouverez un makefile, un fichier hello.idl contenant une interface, l'implémentation en Java de cette interface, un client et un serveur en java.

Pour utiliser cette application, il faut (voir *Makefile*) :

1. compiler l'idl,
2. compiler les sources java,
3. démarrer le service de nommage,
4. exécuter le serveur,
5. exécuter le client.

À la compilation de l'idl, plusieurs fichiers sont générés :

`HelloHelper.java` classe utilitaire, contient notamment une méthode `narrow`, servant à remplacer le cast java

`HelloHolder.java` classe gérant les paramètres out (non pris en charge par java)

`Hello.java` "cablage" CORBA

`HelloOperations.java` traduction java de l'idl

`HelloPOA.java` squelette (portable object adapter)

`_HelloStub.java` stub

Vous remarquerez les points suivants :

- le serveur communique au client la référence corba de l'objet hello en affichant son IOR sur la sortie standard.
- le serveur fait ses affichages sur la sortie d'erreur (par des `System.err.println`) de manière à ce que la seule chose qui soit affichée sur la sortie standard soit l'IOR de l'objet distribué.
- le client suppose prendre en paramètre de ligne de commande l'IOR de l'objet distribué auquel il doit s'adresser.
- pour que le client récupère facilement l'IOR transmise par le serveur, il est pratique d'utiliser la redirection unix de la sortie standard. C'est ce qui est proposé dans le *makefile* : lors du lancement du serveur, on redirige la sortie standard (via un >) vers un fichier `IOR.txt`, et lors du lancement du client, on passe en paramètre le contenu de ce fichier (via un `cat`).
- on lance le démon `orbd` sur un port donné.
- pour bien structurer ses sources java, on peut (comme celà est suggéré dans le *makefile*) séparer les sources génées (via `idlj`) de celles réellement écrites par vos soins.

Travail à réaliser

Question 1. Faire fonctionner l'application, comprendre son fonctionne-

```
1 void display(in string message)
```

b- Ajouter le code implémentant cette opération dans la classe `HelloImpl.java`.

c- Modifier le client `Client.java` afin d'utiliser cette opération.

d- Vérifiez que l'application fonctionne toujours !

2 Application bancaire

On désire réaliser une petite application bancaire avec CORBA. On manipulera :

- Des serveurs (banques) gérant des ensembles de comptes
- Des Comptes. Un compte = un solde et les opérations crédit/débit, chaque compte est associé à un client
- Des applications pour les utilisateurs
- Des applications pour les gestionnaires

Question 3. Comptes

a- Concevoir dans un fichier `.idl` le contrat IDL permettant de définir la notion de compte. On permettra le débit et le crédit de montants sur un compte. On définira un client comme une chaîne (pour simplifier l'exemple) et on veillera à ce que chaque compte soit associé à un client (unique, on ne gère pas ici les comptes joints). Dans un premier temps, on ne s'occupera pas des exceptions qui pourraient être levées quand on débite un solde d'un montant supérieur au crédit.

b- Compiler cet idl pour vous assurer qu'il est correct (syntaxiquement tout du moins), et pour obtenir la traduction en Java de votre interface.

c- Implémenter en java la notion de compte, en regardant dans les classes java générées lors de la compilation de l'idl (notamment `MONINTERFACEOperations.java`).

d- Implémenter un serveur de compte (un serveur par compte ...) qui crée un compte et diffuse son IOR (en utilisant le même mécanisme d'affichage de l'IOR que pour l'exemple `HelloWorld`).

e- Créer un client simple pour ce serveur de comptes.

f- Tester !

g- Mettre en place la gestion des exceptions (repartir de l'IDL, ne pas oublier de recompiler l'IDL ...).

Question 4. Banques

a- Concevoir (dans un fichier `.idl` qui importera la définition des comptes) le contrat IDL permettant de définir la notion de banque. On permettra la création de comptes (à partir d'un client), la destruction de comptes, et la recherche de compte par client. On n'autorisera qu'un compte par client

pour simplifier. On mettre en place des exceptions si l'on tente de créer un compte pour un client en ayant déjà un, ou si l'on recherche le compte d'un client n'en ayant pas.

b- Implémenter la notion de banque, et un serveur qui crée une banque (éventuellement avec des comptes) et diffuse son IOR (en utilisant le même mécanisme d'affichage de l'IOR que pour l'exemple `HelloWorld`).

c- Créer un client simple pour la banque.

d- Tester !

Question 5. Service de nommage

Jusqu'ici, le client et le serveur de banque se communiquent l'IOR de la banque via la sortie standard redirigée vers un fichier. On va maintenant passer par le service de nommage `orbd`.

a- Modifiez le serveur pour qu'il inscrive la référence de la banque auprès du service de nommage.

b- Modifiez le client pour qu'il obtienne la référence de la banque auprès du service de nommage.

c- Tester. Ne pas oublier de lancer l'`orbd`.

Question 6. On créera ensuite 2 types de clients : un pour les gestionnaires (pour la création et la modification des comptes), et un pour les clients de la banque (consultation).