TP 5 – Callbacks en CORBA

Remarque : le répertoire *0-HelloCallBack/* contient un exemple duquel vous pourrez vous inspirer. Analysez-le et tentez de déterminer, avant l'exécution, où (sur le serveur ou le client) et dans quel ordre se feront les affichages sur la sortie standard.

1 Premier callback

(a) Rendez-vous dans le répertoire *1-ExoCallBack/* et examinez les fichiers **.java* ;

(b) Compilez le code en exécutant le script **./compile** et exécutez le serveur et le client (script **./run**) afin de comprendre le déroulement de l'exécution.

(c) Expliquez (dans *reponse.txt*) le fonctionnement de l'interaction qui se produit : à quoi servent les interfaces déclarées dans le fichier IDL ? Qu'est-ce qui est fait par les fichiers *ServeurCalcul.java* et *AfficheurClient.java* ?

(d) Modifiez le serveur pour qu'il mette à disposition le service suivant :
- le client s'inscrit auprès du serveur avec la méthode *inscription(double val, Afficheur peer)*, ce qui a également pour effet d'incrémenter le compteur de *val* unités ;
- ensuite, le serveur fait afficher toutes les secondes la valeur de l'accumulateur au niveau du client.

(e) Développez un client qui utilise ce service.

2 Annuaire téléphonique

(a) Se rendre dans le repertoire *2-TelephoneExo/* ;

(b) Examiner les fichiers **.java* ;

(c) Compiler le code en exécutant le script **./compile** ;

(d) Exécuter le serveur et les clients ;

(e) Transformer ces codes afin d'avoir uniquement des appels distants non bloquants. Pour les deux premières méthodes du fichier *Annuaire.idl*, cela impliquera nécessairement de passer par un mécanisme de CallBack. Vous pourrez utiliser un objet de rappel du type suivant :

```
interface Afficheur {
  oneway void afficheRes(in string res);
  oneway void arreteORB();
};
```

3 Chat

3.1 Client

Le répertoire *3-Chat/* contient un fichier *chat.idl* et un squelette de client *ClientChatImpl.java* (qui utilise *ThreadRun.java*). Un serveur tourne sur Turing et met à disposition sur le port 1050 un objet de type *ServeurChat* (dont l'interface figure dans le fichier IDL) et nommé « ServeurChat » sur le serveur de noms. Vous aurez à compléter le squelette du client afin que celui-ci puisse se logger sur le serveur de chat, converser (en broadcast ou en message privé) avec les autres utilisateurs loggés, et se délogger à la fin. Le rôle du serveur est de maintenir une liste des personnes loggées afin de pouvoir transmettre

les messages reçus à tout le monde (méthode *broadcast*) ou à une seule personne (méthode *chat*). Pour pouvoir transmettre des messages à n'importe quel moment aux clients connectés, le serveur nécessite de connaître, pour chaque client, une classe faisant office de callback. Voici ce que vous devez faire :

(a) Examiner les fichiers fournis et déterminer la classe qui fera office de callback et comment celui-ci est transmis au serveur.

(b) Compléter les methodes *receiveNewChatter*, *receiveExitChatter* et *receiveChat* dans le fichier client.

(c) Compléter la méthode principale (main) suivi de la méthode loop qui gèrera le chat. Vous aurez à vous logger sur le serveur en déclarant votre nom, tester le bon fonctionnement en affichant le résultat de la méthode *ping* du serveur et écrire une boucle permettant de lire sur votre entrée standard un message afin de le transmettre au serveur.

(d) Compiler à l'aide du script **./compile** fourni.

(e) L'exercice sera totalement réussi lorsqu'en une seule connexion, vous aurez réussi à vous logger, à identifier le pseudo de l'enseignant (demandez-lui s'il est bien loggé), à lui envoyer un message indiquant que vous avez réussi à vous logger, puis à annoncer en broadcast que vous avez terminé.

3.2 Serveur

Écrivez un serveur qui propose les fonctionnalités nécessaires au bon fonctionnement d'un chat tel que défini ci-dessus.