UPMC
SORBONNE UNIVERSITÉS

SRCS : Systèmes Répartis Client/Serveur

Examen du 24 Mai 2011

durée 2h - **Tous les documents papiers sont autorisés**
Téléphones portables, baladeurs et autres appareils électroniques doivent être éteints
Attention l'énoncé est imprimé recto-verso sur 2 feuilles
Le barème est donné à titre indicatif.

Julien Sopena

Exercice 1 : Questions de cours

Question 1 – 1,5 point
Quels sont les 3 types de synchronisme disponibles en CORBA ? Donnez, en une phrase, la méthode de mise-en-œuvre pour chacun d'eux.

Question 2 – 1,5 point
Parmi les multiples points de vue sur les composants, donnez (en une phrase) trois des définitions vues en cours.

Question 3 – 1 point
Pourquoi *JMS* est largement utilisé dans les architectures à composant *EJB* ?

Question 4 – 1 point
Comment peut-on avec JMS envoyer régulièrement des informations peu prioritaires sans surcharger le destinataire ?

Question 5 – 1 point
On considère une base de donnée universitaire où chaque étudiant référence une liste de modules. Comment éviter la suppression des modules lorsque l'on supprime un `Entity Bean` correspondant à un étudiant et ayant un attribut listant l'ensemble des modules auxquels il est inscrit.

Exercice 2 : Objet distant RMI

On souhaite définir un objet distant *RMI* proposant la méthode distante de signature suivante :

```
B truc(A a)
```

Question 1 – 1 point
Écrivez l'interface `IDistant` correspondante en supposant données les classes ou interfaces A et B.

Question 2 – 1,5 point

On souhaite que le paramètre de la méthode soit transmis par valeur du client au serveur. Quelle(s) condition(s) doit (doivent) être satisfaite(s) par A ?

Question 3 – 1,5 point

On souhaite que le résultat de la méthode corresponde à un objet distant. Quelle(s) condition(s) doit (doivent) être satisfaite(s) par B ?

Exercice 3 : Programmation Publish/Subscribe en *CORBA*

Dans cet exercice, on reprend l'idée d'un service de Publish/Subscribe (Publication/Abonnement) qui permet à des "subscribers" de s'enregistrer auprès de "publishers" dont ils souhaitent recevoir les publications. Pour simplifier, on ne s'intéresse ici qu'à l'implémentation des "subscribers".

L'architecture repose, comme pour le partiel, sur un arbre de "subscribers" et leur fonctionnement reste le même :

1. Connexion et enregistrement auprès d'un publisher ou d'un autre subscriber. Le nœud récupère alors le nombre de messages publiés avant sa connexion.
2. Attente :
   - de connexion d'autre "subscriber" auxquels il retourne le nombre de messages déjà publiés et pour lesquels il jouera ensuite le rôle de "publisher"
   - de nouvelles diffusions émises par le "publisher" (ou par le "subscriber" sur lequel il s'est lui-même connecté).

Question 1 – 1 point

Cette nouvelle implémentation *Corba* des subsribers s'appuie sur deux services : `publish` et `subscribe`. En vous appuyant sur un schéma, montrez pour chacun de ces services qui a le rôle de client et qui a le rôle de serveur.

Question 2 – 1 point

Définir l'interface *IDL* d'un objet subscriber ; l'interface aura pour nom `SubNode` et sera intégrée à un module `PubSub`. Pour des raisons d'efficacité, les services devront être asynchrones lorsque cela est possible.

Question 3 – 1 point

Quel problème de synchronisation va poser l'implémentation de ces deux services.

Question 4 – 2 points

Donnez une implémentation *Java* complète d'une classe `SubNodeImpl` qui implémente votre interface. Vous veillerez, entre autre, à résoudre le problème de synchronisation.

Question 5 – 2 points

Donnez l'implémentation d'un programme *MonSub* à 2 paramètres : un identifiant (un entier) et le nom d'un fichier compatible avec la méthode `restoreIOR()` du cours. Ce programme instanciera votre classe en la connectant à l'objet "subscriber" dont l'IOR est contenu dans le fichier, puis génèrera à l'aide de la fonction `saveIOR()` un fichier permettant à un autre "subscriber" de ce connecter à votre instance.

Question 6 – 2 points

Quel mécanisme *CORBA* permet d'éviter l'utilisation des fichiers de connexion ? Modifier votre implémentation de la classe ~~`SubNodeImpl`~~ MonSub pour bénéficier de ce service.

Question 7 – 1 point

Pour en faciliter le monitorage, on souhaite faire hériter la classe `SubNodeImpl` d'une classe `LoggedNode`. Quel mécanisme *CORBA* permet cela ? Modifier votre implémentation de la classe `SubNodeImpl` pour bénéficier de ce service.