Université des sciences et technologies de Lille (USTL) Janvier 2009
Master mention Informatique – M1
Module : Conception d'applications réparties (CAR)

Examen

Conception d'applications réparties (CAR)

Tous documents papier autorisés. Téléphones et ordinateurs interdits.
Le barème est donné à titre indicatif. 3 heures.
Ce sujet comporte 3 pages.

1 Questions de cours (3 points)

1.1 Dans un POA CORBA, on trouve deux catégories d'objets : les objets CORBA proprement dit et les objets dit servant. Expliquer la différence qui est faite entre ces deux catégories d'objets. Donner un exemple de cas où un objet CORBA est associé à plusieurs objets servants. Donner un exemple de cas où un objet servant est associé à plusieurs objets CORBA. (2 points)

1.2 Lorsque l'on écrit des *threads* en Java, quelle est la différence entre la méthode `start` et la méthode `run` ? Que se passe-t-il si l'on appelle `run` sans appeler `start` ? (1 point)

2 RMI et serveur FTP (3 points)

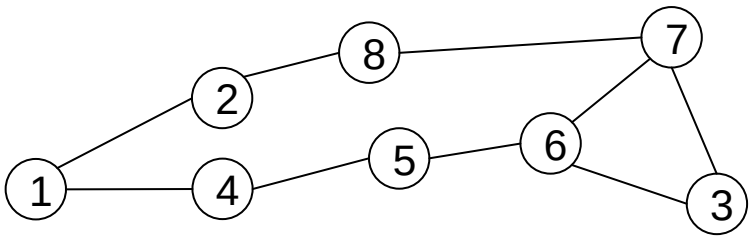
On souhaite pouvoir accéder à un serveur FTP via un objet RMI. On met en place les trois programmes Java suivants :
- `Utilisateur.java` : le programme qui utilise l'objet RMI pour accéder au serveur FTP.
- `RMIImpl.java` : la classe de l'objet RMI.
- `FTP.java` : le serveur FTP.

2.1 Parmi ces trois programmes, dire qui est client (de qui) ? serveur (pour qui) ? (1 point)
2.2 Proposer une interface pour l'objet RMI. Cette interface doit comporter une méthode par opération possible sur un serveur FTP. Les paramètres de chaque méthode doivent être auto-suffisants et ne pas présupposer d'effet de bord (i.e. pas d'accès direct au système de fichier). Vous définirez clairement et brièvement le comportement de chaque méthode (1 ou 2 phrases maximum par méthode) ainsi que la définition des paramètres et éventuellement des exceptions (hormis `RemoteException` qui est supposée connue de tous). (2 points)

3 Diffusion dans un graphe d'objets CORBA (7 points)

On considère un ensemble d'objets CORBA organisés selon une topologie en graphe. Chaque objet a un numéro unique fixé lors de la création du graphe. Chaque objet connaît uniquement ses voisins immédiats

grâce à une liste de références d'objets CORBA. Ainsi, dans l'exemple ci-dessous, 1 connaît 2 et 4, 2 connaît 1 et 8, etc.



On souhaite mettre en place un parcours du graphe de façon à trouver l'objet de numéro le plus élevé. Pour cela, on propose de mettre en place une invocation de méthode synchrone CORBA qui, à partir d'un objet dit initiateur (par exemple l'objet 2), est transmis de proche en proche sur l'ensemble du graphe.

3.1 Comment sait-on que tous les objets du graphe ont été visités ? (0,5 point)

3.2 Comment faire en sorte, le plus simplement possible, qu'à la fin du parcours, le numéro de site le plus élevé soit connu de l'initiateur ? (0,5 point)

Soit `SiteItf` l'interface IDL implantée par chaque objet du graphe. Soit `Id` une structure IDL comprenant la référence de l'objet CORBA et son numéro (entier) associé. On suppose, dans un premier temps, que chaque objet transmet l'invocation séquentiellement à ses voisins. On suppose que la méthode `_this()` fait partie des librairies CORBA et peut être appelée pour récupérer la référence de l'objet CORBA courant.

3.3 Définir l'interface IDL `SiteItf` et la classe Java `SiteImpl` implantant cette interface ? (3 points)

3.4 En fonction d'un arbre donné, y a-t-il un seul parcours possible du graphe ? Si oui, expliquer pourquoi. Si non, indiquer de quoi dépendent les différents parcours possibles. (0,5 point)

3.5 Donner une évaluation du nombre d'invocations nécessaire pour le parcours d'un graphe. (1 point)

On considère maintenant que l'invocation est transmise concurremment sur l'ensemble des voisins.

3.6 Est-ce que cela modifie l'interface `SiteItf` ? Si oui, expliquer en quoi consiste la modification. Si non, dire pourquoi. (0,5 point)

3.7 Même question pour la classe `SiteImpl`. (1 point)

4 Méthode de rappel en RMI (4 points)

On considère deux objets RMI A et B. B fournit une méthode `mult` permettant de faire le produit de deux matrices X et Y de taille identique. Chaque matrice est un tableau bidimensionnel de réels double. On considère que la méthode `mult` effectue une itération sur le nombre de lignes de la matrice X (on ne demande pas de fournir plus de détails sur le code de la méthode `mult`). On souhaite qu'à chaque pas de l'itération, l'objet B fournisse à l'objet A l'indice de l'itération en invoquant une méthode `rappel` que vous définirez. A affichera l'indice reçu. Par ailleurs, A fournit une méthode `main` qui invoque la méthode `mult` en transmettant deux matrices X et Y que l'on suppose connues.

4.1 Proposer le code Java de l'interface `AItf` de l'objet RMI A ainsi que de la classe `AImpl` implantant cette interface. Même question pour l'interface `BItf` de l'objet RMI B et de sa classe d'implantation `BImpl`. (2 points)

4.2 Que se passerait-il si l'objet A n'était pas *multi-threadé* ? (1 point)

4.3 On souhaite que l'invocation de la méthode `rappel` soit asynchrone. Java RMI propose-t-il cette notion ? Si oui, comment ? Si non, proposez une solution de remplacement ? (1 point)

5 RMI et Multicast-IP (3 points)

5.1 Quel est le protocole de transport utilisé de façon standard par Java RMI pour les invocations de méthode ? (0,5 point)

On souhaite remplacer ce protocole par Multicast IP afin de faire en sorte que chaque invocation de méthode puisse atteindre, non pas un seul objet serveur RMI, mais plusieurs objets serveurs RMI. Pour cela, on propose de remplacer la classe `UnicastRemoteObject` par une nouvelle classe appelée `MulticastRemoteObject`.

5.2 Quel est le rôle de cette nouvelle classe `MulticastRemoteObject` ? Décrire en français son comportement. (1 point)

5.3 Quel est le rôle de l'outil `rmic` ? Quel est l'impact de classe `MulticastRemoteObject` sur `rmic` ? (1 point)

5.4 Quel est impact de cette classe `MulticastRemoteObject` sur l'annuaire `rmiregistry` ? (0,5 point)