Université Lille 1 Juin 2009
Master mention Informatique – M1
Module : Conception d'applications réparties (CAR)

Examen – 2nde session

Conception d'applications réparties (CAR)

Tous documents papier autorisés. Téléphones, calculatrices et ordinateurs interdits.
Le barème est donné à titre indicatif. 2 heures.
Ce sujet comporte 3 pages.

1 Questions de cours (4 points)

1.1 Quelle est la différence entre transmission par valeur et transmission par référence ? Vis-à-vis de RMI, quelle condition doit respecter la classe d'un objet transmis par valeur ? Même question pour un passage par référence ? (1 point)

1.2 Citer deux mécanismes d'encodage de données vus en cours ou en TD/TP pour la transmission de paramètres lors de l'invocation d'une méthode à distance. (1 point)

1.3 Le langage WSDL propose la notion de port. Quelle est la notion qui s'en approche le plus en CORBA IDL ? Justifier votre réponse. (1 point)

1.4 Comparer en 1 ou 2 phrases les modes d'invocation de méthode de CORBA et de Java RMI. (1 point)

2 Instanciation à la demande d'objets serveur et protocole client/serveur de détection d'activité (9 points)

Cet exercice s'intéresse à l'instanciation à la demande d'objets serveurs CORBA. L'idée est de ne pas laisser actifs en permanence tous les objets serveurs CORBA, mais de les activer en fonction des besoins des clients. Dans cet exercice, l'activation est réalisée par un serveur Web.

On dispose de trois catégories de programmes : programme utilisateur, serveur Web et objets serveurs CORBA. Le scénario de fonctionnement est le suivant :
- étape 1 : le programme utilisateur demande au serveur Web à l'aide d'une commande `GET` d'activer un objet serveur CORBA en lui fournissant le nom de l'objet qu'il souhaite invoquer.
- étape 2 : le serveur Web retourne au programme utilisateur la référence de l'objet serveur CORBA correspondant.
- étape 3 : le programme utilisateur invoque l'objet serveur CORBA.

2.1 Dans le scénario précédent, et pour les trois catégories de programme, dire qui est client (de qui), qui est serveur (pour qui) et qui est client/serveur (de qui et pour qui) ? (1 point)

Interface IDL CORBA

Les objets serveurs CORBA offrent les méthodes suivantes :
- `invoke` : permet d'exécuter une opération. Elle prend en paramètre le nom de l'opération à invoquer et un tableau de valeurs quelconques. Cette méthode retourne une valeur quelconque et lève une exception `InvocationException` qui est associée à un message (chaîne de caractères) que vous définirez.
- `getHost` : fournit l'adresse de la machine sur laquelle est instancié l'objet CORBA. Cette méthode ne prend pas de paramètre. Elle retourne une chaîne de caractères contenant l'adresse de la machine.
- `getPort` : retourne le numéro du port TCP sur lequel l'objet CORBA est capable de lire des messages. Cette méthode ne prend pas de paramètre. Elle retourne un entier correspondant au port TCP en question.

2.2 Ecrire l'interface `CalculItf` de ces objets serveurs CORBA. (1 point)

Programme utilisateur

On souhaite que les programmes utilisateurs puissent périodiquement scruter les objets CORBA qu'ils ont invoqué pour savoir s'ils sont toujours en fonctionnement. En même temps qu'ils appellent la méthode `invoke`, les programmes client testent l'activité des objets CORBA. Pour cela, ils utilisent un *thread* (classe `ClientPingThread`) pour envoyer un message TCP contenant les 4 octets P I N G, sur l'adresse fournie par la méthode `getHost` et le port fourni par `getPort`. Les objets serveurs CORBA utilisent un *thread* (classe `ServeurPingThread`) pour recevoir ce message et y répondre avec un message contenant les 4 octets P O N G. Si 10s après l'envoi du message P I N G le client n'a pas reçu de réponse ou si la réponse ne correspond pas à P O N G, il renvoie le message P I N G. Si au bout de 3 envois, il n'a toujours pas de réponse, il lève une exception pour signaler la panne de l'objet serveur CORBA.

2.3 Un *thread* Java peut s'écrire de 2 façons. Donner en une. (0,5 point)

2.4 Le message P I N G permet de détecter si l'objet CORBA est en panne ou toujours en activité. Il existe une autre technique de détection de pannes. Donner son nom et expliquer brièvement son fonctionnement. (0,5 point)

2.5 Proposer une solution simple, n'utilisant pas de *thread* supplémentaire, pour savoir que l'attente de 10s du message P O N G est arrivée à échéance. (1 point)

2.6 Ecrire le code Java de la classe `ClientPingThread` qui teste l'activité du serveur. (2,5 points)

Objets serveur CORBA

2.7 Ecrire le code Java de la classe `ServeurPingThread` qui traite les messages P I N G du client. On utilisera le numéro de port TCP 4321. (2 points)

Serveur Web

2.8 Est-il nécessaire que les objets CORBA créés par le serveur Web soient enregistrés dans le l'annuaire ? Si oui, pourquoi ? Si non, comment les programmes utilisateurs peuvent invoquer des méthodes sur ces objets ? (1 point)

3 CORBA – Serveur de requêtes (7 points)

On considère un serveur CORBA stockant des données et capable d'exécuter des requêtes sur ces données. Les données peuvent être par exemple des noms et des numéros de téléphone, et les requêtes peuvent être des recherches de numéros de téléphone, mais cela n'a aucune importance pour cet exercice. Chaque requête peut fournir plusieurs résultats. Par exemple, plusieurs numéros de téléphone peuvent correspondre à un même nom.

Le serveur fonctionne de la manière suivante :
1. La méthode `open` permet d'ouvrir une connexion sur le serveur et de transmettre la requête que l'on souhaite exécuter. Elle prend en paramètre une chaîne de caractères qui contient le texte de la requête et retourne un entier qui est l'identifiant de connexion.
2. La méthode `next` permet d'itérer sur les résultats de la requête et récupère un résultat à la fois. On appelle `next` autant de fois qu'il y a de résultats à récupérer. La méthode `next` prend en paramètre l'identifiant de connexion. Le paramètre de retour de la méthode `next` est de type `union` IDL CORBA associé à un discriminant de type booléen :
   - si le booléen vaut vrai le type de l'union est une chaîne de caractères qui contient un des résultats de la requête,
   - si le booléen vaut faux, cela signifie qu'il n'y a plus de résultat à récupérer et le type de l'union est un entier contenant le nombre de résultats retournés au client.
3. La méthode `close` ferme la connexion. Cette méthode prend en paramètre l'identifiant de connexion et ne retourne rien.

3.1 Ecrire le code IDL CORBA de l'interface du serveur. (1,5 point)

3.2 Quel est l'utilité de l'identifiant de connexion ? (0,5 point)

3.3 Ecrire le code Java du programme client qui envoie au serveur (objet CORBA enregistré dans l'annuaire sous le nom « `corbaname::localhost:1704#sql` »), la requête « `SELECT tel FROM t WHERE nom='Bob'` » et qui affiche tous les résultats. (2 points)

3.4 Avec la méthode `next` précédente, on récupère les résultats un à un. Une autre façon de faire serait de retourner tous les résultats en une seule fois. Quels sont les avantages et/ou les inconvénients respectifs des deux façons de faire ? (1 point)

On souhaite mettre en place une solution mixant les deux précédentes, dans laquelle le serveur retourne en bloc un certain nombre de résultats (par exemple les 10 premiers, cette valeur devant pouvoir être spécifiée par le client), puis que l'on puisse itérer sur les résultats suivants.

3.5 Proposer le code IDL CORBA de l'interface serveur mettant en œuvre cette solution (on pourra omettre les méthodes `open` et `close`). (1 point)

On se place maintenant dans le cas où les requêtes transmises au serveur ne sont plus de simples consultations mais sont des traitements quelconques à effectuer sur les données gérées par le serveur. On envisage deux solutions :
1. comme précédemment on envoie le traitement au serveur qui l'exécute.
2. on récupère les données sur le client, on effectue le traitement sur le client et on renvoie au serveur les données modifiées.

3.6 Discuter des avantages et des inconvénients de ces deux solutions. (1 point)