Université des sciences et technologies de Lille (USTL) Janvier 2007
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 (2 points)

1.1 Quel est le rôle de la « clé » dans un IOR CORBA ?
1.2 Citer deux différences entre le service de nommage de CORBA et celui de RMI.

2 RMI et Web (14 points)

Cet exercice s'intéresse à l'instanciation à la demande d'objets serveurs RMI. L'idée est de ne pas laisser actifs en permanence tous les objets serveurs RMI, mais de les activer en fonction des besoins des clients. Dans cet exercice, l'activation est réalisée par un serveur Web. L'activation peut être une vraie création ou une récupération d'objet existant dans un *pool*. Les objets serveurs RMI exécutent des calculs mathématiques quelconques. Chaque type de calcul est identifié par un nom sous forme d'une chaîne de caractères (ex. : intégration, dérivation, équation différentielle, etc.). Les utilisateurs demandent donc au serveur Web d'activer des objets serveurs RMI d'un certain type en fonction de leurs besoins.

On dispose de trois catégories de programmes : programme utilisateur, serveur Web et objets serveurs RMI. 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 RMI en lui fournissant un nom (correspondant au type de calcul dont il a besoin).
- étape 2 : le serveur Web retourne au programme utilisateur la référence de l'objet serveur RMI.
- étape 3 : le programme utilisateur invoque l'objet serveur RMI.

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 RMI

Les objets serveurs RMI offrent les méthodes suivantes :
- `submit` : permet d'effectuer un calcul. Cette méthode prend en paramètre une chaîne de caractères qui contient les arguments du calcul et retourne un objet Java contenant le résultat du calcul. Cet objet Java est transmis par valeur.

- `getHost` : fournit l'adresse de la machine sur laquelle est instancié l'objet RMI. Cette méthode ne prend pas de paramètres. Elle retourne une chaîne de caractères contenant l'adresse de la machine.
- `getPort` : retourne un port TCP sur lequel l'objet RMI est capable de lire des messages. Cette méthode ne prend pas de paramètres. Elle retourne un entier correspondant au port TCP en question.

2.2 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 ? (0,5 point)

2.3 Ecrire l'interface `CalculItf` de ces objets serveurs RMI. (1 point)

Programme utilisateur

On souhaite que les programmes clients puissent périodiquement scruter les objets RMI auxquels ils ont soumis des calculs pour savoir s'ils sont toujours en fonctionnement. En même temps qu'ils appellent la méthode `submit`, les programmes client testent l'activité de l'objet RMI. 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 RMI utilisent un *thread* (classe `ServeurPingThread`) pour recevoir ce message et y répondre avec un message contenant les 3 octets A C K. 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 A C K, 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 RMI.

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

2.5 Le message P I N G permet de détecter si l'objet RMI 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)

2.6 Proposer une solution simple, n'utilisant pas de *thread* supplémentaire, pour savoir que l'attente de 10s du message A C K est arrivée à échéance. (0,5 point)

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

2.8 Ecrire le code Java du programme client qui invoque la méthode `submit("12.5")` sur l'objet RMI `obj` et qui en même temps, teste l'activité du serveur. (0,5 point)

Serveur RMI

2.9 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 2003. (2,5 points)

2.10 Ecrire le code Java de la méthode `submit`. On laissera une zone de commentaire pour indiquer l'emplacement du calcul et la valeur retournée par la méthode. (0,5 point)

Serveur Web

2.11 Est-il nécessaire que les objets serveurs RMI créés par le serveur Web soient enregistrés dans le service de noms de RMI (`rmiregistry`) ? Si oui, pourquoi ? Si non, comment les programmes utilisateurs peuvent invoquer des méthodes sur ces objets ? (1 point)

2.12 Si l'on veut que le serveur Web puisse retourner la même référence d'objets RMI à deux clients s'exécutant simultanément, de quoi faut-il s'assurer ? (1 point)

Le serveur Web reçoit des requêtes de la forme :
`GET /index?calcul=`*`nom`*` HTTP/1.1`
où *nom* est le nom du calcul mathématique demandé.

2.13 Décrire en français l'algorithme de fonctionnement du serveur Web. (2 points)

3 CORBA (4 points)

3.1 Indiquer quelles sont les erreurs contenues dans l'interface IDL suivante. Expliquer brièvement en quoi consiste l'erreur et proposer une correction. (2 points)

```
public interface Carte {
    const long echelle;
    typedef enum {nord,sud,est,ouest} Cardinaux;
    void orientation( Cardinaux direction );
    interface Michelin {
       string numero();
       long echelle() { return echelle; }
    };
};
```

Soit un objet CORBA appelé Calculateur qui fournit une méthode `multiplier` pour effectuer des multiplications de matrice. Les matrices sont représentées par des tableaux à 2 dimensions de taille fixe. Chaque dimension comprend 100 éléments de type réel double. La méthode `multiplier` prend en entrée 2 matrices et retourne une matrice résultat.

3.2 Définir l'interface IDL `CalculateurItf` de cet objet Calculateur. (0,5 point)

On considère une liste de `n` objets Calculateur géré par un objet CORBA appelé Frontal. Le frontal reçoit les requêtes des clients et les renvoie à un des objets Calculateur dans l'ordre de la liste. La 1ère requête est envoyée au 1er objet, la 2ème au 2ème objet, etc. Lorsque la fin de la liste est atteinte, on recommence avec le 1er objet.

3.3 Définir l'interface IDL `FrontalItf` de l'objet Frontal. (0,5 point)

3.4 Écrire la classe `FrontalImpl` qui implémente l'interface `FrontalItf` par délégation (mécanisme dit *tie* en CORBA). (1 point)