100059 _ 25.25

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

Juin 2010

Examen – 2nde session

Conception d'applications réparties (CAR)

Tous documents papier autorisés.
Téléphones, calculatrices, ordinateurs et équipements électroniques interdits.
Le barème est donné à titre indicatif. 3 heures.
Ce sujet comporte 4 pages.

1 Question de cours (4 points)

1.1 Rappeler la définition des trois modes de passage de paramètre de CORBA.

1.2 Sachant qu'en Java les objets sont transmis par référence, quel est parmi les trois modes de passage de paramètre celui qui s'en rapproche le plus ?

1.3 Pourquoi un appel de méthode semi-synchrone permet d'introduire de la concurrence de traitement entre un client et un serveur ?

1.4 Pourquoi dans la politique de synchronisation lecteurs/écrivain vue en cours, l'appel à la méthode Java `notify()` n'est pas suffisant à la fin d'une demande de lecture et pourquoi faut-il un appel à la méthode `notifyAll()` ?

2 Ping-pong (10 points)

Soient deux objets A et B. L'objet A envoie un message `PING` à l'objet B qui lui répond par un message `PONG`. La première partie de l'exercice met en œuvre cet échange avec des *sockets* Java et la seconde avec CORBA.

Sockets Java

Les messages échangés via les *sockets* ont la structure suivante :
- 1 octet pour identifier le message. L'octet vaut 0 pour `PING` et 1 pour `PONG`.
- 1 octet pour le nombre de paramètres associés au message.
- Les octets suivants contiennent les paramètres associés au message.
  Pour chaque paramètre, on trouve :
  - 1 octet pour son type. L'octet vaut 0 si le paramètre est un entier et 1 si le paramètre est une chaîne de caractères.
  - La valeur du paramètre.
    - Les entiers sont codés sur 2 octets. L'octet de poids fort est placé en tête.

- Les chaînes de caractères sont précédées d'un octet qui fournit la taille de la chaîne. Chaque caractère occupe 1 octet.

Le message `PING` est associé à cinq paramètres. Chaque paramètre est un entier. Les quatre premiers correspondent à l'adresse IP de la machine qui envoie le message `PING` (par exemple 10.3.250.88). Le cinquième paramètre correspond au numéro de port TCP sur lequel la machine attend la réception d'un message `PONG`. Le message `PONG` est associé à un paramètre de type chaîne de caractères.

On suppose que la machine A connaît l'adresse IP et le port TCP sur laquelle la machine B reçoit les messages `PONG`. Soit l'échange suivant :
- la machine A envoie PING 10.3.250.88 1024 à la machine B,
- la machine B reçoit ce message, en extrait l'adresse IP et le port TCP et utilise ces informations pour envoyer à la machine A un message `PONG` avec la chaîne de caractères « ACK ».
- la machine A reçoit le message `PONG`.

2.1 Donner la séquence d'octets transmis par la machine A à la machine B. (1 point)
2.2 Donner la séquence d'octets transmis par la machine B à la machine A. (1 point)
2.3 Donner le code Java du programme qui s'exécute sur la machine A. (1,5 point)
2.4 Donner le code Java du programme qui s'exécute sur la machine B. (1,5 point)

2.5 Plutôt que de transmettre l'adresse IP avec le message `PING`, on aurait pu utiliser une fonctionnalité de l'API *socket* de Java pour connaitre l'adresse de l'émetteur. Laquelle ? (0,5 point)

2.6 De la même façon, peut-on se dispenser de l'envoi du numéro de port TCP et profiter des fonctionnalités de l'API *socket* de Java ? Justifier votre réponse. (0,5 point)

CORBA

Soient deux objets CORBA A et B :
- l'objet B implémente l'interface `BItf` qui fournit une méthode `ping` prenant en paramètre d'entrée une référence d'objet CORBA de type `AItf`. La méthode `ping` ne retourne rien (type de retour `void`). La méthode `ping` appelle la méthode `pong` sur la référence passée en paramètre.
- l'objet A implémente l'interface `AItf` qui fournit une méthode `pong` (aucun paramètre, type de retour `void`). La méthode `pong` est vide.

2.7 Écrire le code CORBA IDL des interfaces `AItf` et `BItf`. (1 point)

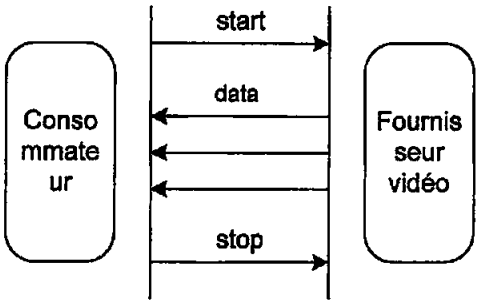
2.8 Écrire des classes `AImpl` et `BImpl` qui implémentent respectivement, les interfaces `AItf` et `BItf`. Écrire le code de la méthode `main` de la classe `Client` qui instancie les objets A et B et appelle la méthode `ping` sur l'objet B. (1 point)

2.9 On souhaite mesurer le temps mis par les invocations de méthodes pour faire l'aller-retour (appel de `ping` suivi de `pong`). À titre indicatif, la méthode Java `System.currentTimeMillis()` retourne un entier long qui correspond à l'heure courante. Quelle méthode faut-il modifier pour réaliser cette mesure ? Donner le code de cette modification. (0,5 point)

2.10 On considère maintenant que les méthodes `ping` et `pong` sont `oneway`. Peut-on toujours mesurer le temps pour faire l'aller-retour (appel de `ping` suivi de `pong`) de la même façon que précédemment ? Justifier votre réponse. En cas de changement, proposer une modification du code. (1,5 point)

3 CORBA – Serveur de vidéo à la demande (6 points)

On considère les objets CORBA « Consommateur » et « Fournisseur vidéo » (voir schéma ci-dessous). Le second est un objet qui diffuse des vidéos qui sont consommées (visualisées) par le premier.



La diffusion se fait via une interface `DiffusionItf` fournissant la méthode suivante :

- `data` : transmet une partie de la vidéo ; cette méthode `oneway` prend en entrée un tableau de taille fixe d'octets contenant les données diffusées.

Le consommateur peut lancer, suspendre et arrêter la diffusion via l'interface `TeleCommandeItf` qui fournit les méthodes suivantes :

- `start` : déclenche la visualisation d'une vidéo et donc l'envoi de méthodes `data` au consommateur ; cette méthode prend en entrée un nom (chaîne de caractères) de la vidéo à visualiser, fournit en sortie un entier indiquant la durée (en seconde) de la vidéo et retourne un identifiant (entier) de séance ;
- `pause` : suspend la diffusion ; cette méthode prend en entrée un identifiant (entier) de séance ;
- `goto` : avance ou retour rapide vers une certaine position exprimée en nombre de secondes par rapport au début de la vidéo ; cette méthode prend en entrée un identifiant (entier) de séance et un entier indiquant la position à atteindre ;
- `stop` : arrête la diffusion ; cette méthode prend en entrée un identifiant (entier) de séance.

3.1 Étant donnés les objets CORBA « Consommateur » et « Fournisseur vidéo », indiquez celui qui implémente l'interface `DiffusionItf` et celui qui implémente l'interface `TeleCommandeItf`. (1 point)

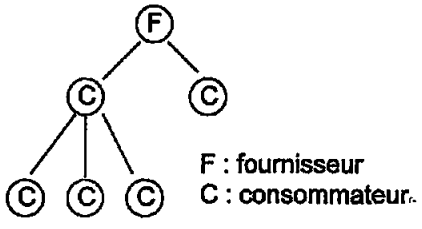
3.2 Donner le code IDL des interfaces `DiffusionItf` et `TeleCommandeItf`. (2 points)

Dans le fonctionnement précédent, on suppose que l'objet « Fournisseur vidéo » connaît l'objet « Consommateur ». Cette information est donc codée en dur dans l'objet « Fournisseur vidéo ».

3.3 Proposer une modification de l'IDL précédent pour faire en sorte que cette information puisse être fournie lors de l'invocation de la méthode `start`. Expliquer en quoi consiste cette modification. (1 point)

3.4 Proposer une modification de l'IDL précédent pour que le « Consommateur » puisse visualiser plusieurs vidéos en même temps. Expliquer en quoi consiste cette modification. (1 point)

Jusqu'à présent un seul « Consommateur » pouvait visualiser la vidéo. On considère maintenant que la vidéo peut être diffusée à plusieurs « Consommateurs » organisés sous la forme d'un arbre (voir schéma ci-dessous). Les « Consommateurs » sont soit des feuilles de l'arbre, soit des nœuds intermédiaires. Dans ce dernier cas, les « Consommateurs » affichent la vidéo et la rediffusent à leurs nœuds fils.



3.5 Donner le code Java de la classe qui implémente le comportement d'un consommateur. Justifier vos choix de codage. (1 point)

Pour cela, on suppose :

- qu'il existe une méthode `display(byte[])` qu'il suffit d'appeler pour afficher les données provenant d'une invocation de méthode `data`,
- que l'on ne prend pas en compte la gestion de l'interface `TeleCommandeItf`.