Université de Lille 1 Juin 2015
Master mention Informatique – M1
Module : Conception d'applications réparties (CAR)

Examen – 2nde session

Construction des applications réparties (CAR)

Tous documents papier autorisés. Téléphones, calculatrices 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 Comment sont définies les interfaces de service en SOAP ? Même question pour les EJB sessions ?
1.2 Quelle est la différence entre un EJB session et un EJB entité ?
1.3 Du point de vue du protocole de communication, quel est le point commun entre REST et SOAP ?

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 1ère partie de l'exercice met en œuvre cet échange avec des *sockets* Java, la 2ème avec Java RMI et la 3ème avec REST.

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 5 paramètres. Chaque paramètre est un entier. Les 4 premiers correspondent à l'adresse IP de la machine qui envoie le message `PING` (par exemple 163.172.139.14). Le 5ème 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 163.172.139.14 2048 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 « Ok ».
- 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. À titre d'information, le code ASCII du caractère O est 79 et celui du caractère k est 107. (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)

Java RMI

Les machines A et B communiquent maintenant via Java RMI. La machine A implémente l'interface `AItf` et la machine B implémente l'interface `BItf`.

2.5 Précédemment le message `PING` était associé à 5 paramètres (adresse IP + numéro de port TCP). Dans le contexte de Java RMI, par quoi suggérez-vous de remplacer ces paramètres ? (1 point)
2.6 Donner le code Java des interfaces `AItf` et `BItf`. (1 point)

On suppose que l'objet RMI s'exécutant sur la machine B est enregistré dans l'annuaire `rmiregistry` sous le nom « machineB ». On suppose que la machine A possède une méthode `main` qui démarre l'échange en invoquant la méthode `PING`.

2.7 Donner le code Java des classes `AImpl` et `BImpl` implémentant respectivement les interfaces `AItf` et `BItf`. La méthode `PONG` affiche à l'écran le message qu'elle reçoit. (2 points)

REST

2.8 On suppose maintenant que les communications utilisent REST. Que proposez-vous pour le code des objets A et B ? (1 point)

3 RMI – Agenda réparti (7 points)

Cet exercice porte sur la mise en œuvre d'un système d'agendas répartis, multi-utilisateurs et *multi-thread*. Soient des objets `Agenda` et des objets `Utilisateur` représentant respectivement des agendas et des utilisateurs de ces agendas. Les objets `Agenda` stockent des rendez-vous, représentés par des objets `RDV`, qui comprennent un identifiant unique (entier), une date (`java.util.Date`), une heure de début (entier) et une heure de fin (entier). Un agenda fournit quatre services permettant respectivement de créer, modifier, lire et détruire un rendez-vous. Les `Agenda` sont des objets accessibles à distance en Java RMI. Les `RDV` sont des objets transmis par copie.

3.1 Proposer une interface Java RMI `AgendaItf` pour les objets `Agenda` et proposer une classe pour les objets `RDV`. Expliquer en français vos choix de conception en rapport avec l'écriture de cette interface et de cette classe. (1 point)

3.2 Dans le cas où ces quatre services sont accessibles via REST, quelles ressources proposez-vous de considérer ? Quelles actions proposez-vous pour ces ressources ? (1 point)

On souhaite pouvoir lister les rendez-vous stockés dans un `Agenda`. Pour cela, le service `list` retourne un itérateur. L'itérateur permet de récupérer à distance et un à un les rendez-vous stockés dans un `Agenda`.

Chaque itérateur fournit les services `hasNext` (aucun paramètre, retourne un booléen qui vaut vrai si il y a un rendez-vous suivant, faux sinon) et `next` (aucun paramètre, retourne le rendez-vous suivant, ou `null` si il n'y en a pas).

3.3 Proposer une ou plusieurs interfaces Java RMI permettant de mettre en œuvre cette fonctionnalité. (1 point)

On souhaite pouvoir inviter des utilisateurs à un rendez-vous. Chaque utilisateur doit pouvoir à tout moment notifier son acceptation ou son refus de participer au rendez-vous. Il doit pouvoir également changer d'avis. Enfin, si un rendez-vous est modifié ou détruit, les utilisateurs l'ayant accepté doivent être prévenus.

3.4 Quel design pattern proposez-vous d'utiliser pour mettre en œuvre cette fonctionnalité ? (1 point)

3.5 Proposer une ou plusieurs interfaces Java RMI permettant de mettre en œuvre ces fonctionnalités. (1 point)

3.6 Du point de vue de la gestion des accès concurrents, quelle politique proposez-vous de mettre en œuvre pour les objets `Agenda`. (0,5 point)

3.7 Écrire la classe Java RMI implémentant un `Agenda`. Le corps des méthodes peut être écrit en pseudo-code. (1,5 point)