Maîtrise Informatique
M5: Programmation Répartie et Architecture 3 1/3

DS – Session de Juin 2001

Denis Caromel

Professeur
Université de Nice - Sophia Antipolis, Département Informatique

Les documents et calculatrices sont autorisés.
Prendre le temps de bien lire et de bien comprendre les questions !
Les réponses ne sont pas longues, soyez *concis* et ... *lisible* !
Notez *clairement* sur votre feuille les numéros des questions (exemple: 3. b.).
Durée: 2H00

1 (5 points)

Questionnaire à choix multiples:
Pour chaque question, il y a 0, 1, ou plusieurs réponses correctes. Marquer sur votre feuille le numéro de la question, et le numéro des **réponse correctes**.

Par exemple:
Question 1:
A: 1, et 4
B: aucune
C: toutes
etc.

Question A:
Sur les RPC en C avec le système `RPCGEN`:

1. Un de ses problèmes est que tout le code doit être écrit à la main.
2. Il ne rentre pas dans la catégorie des RPC à objets.
3. Il ne permet pas de faire communiquer des machines ayant une représentation des types de base incompatibles.
4. C'est un système qui ne permet pas de garder une connexion entre un client et un serveur.
5. Pour chaque appel distant, il permet de libérer immédiatement la mémoire allouée dynamiquement.

Question B:

1. Avec RMI, il est impossible d'enregistrer un objet accessible à distance sur le `rmiregistry` d'une machine distante.
2. De même, il est impossible de créer à distance un objet accessible à distance.
3. Avec RPC en C, il est impossible de créer à distance un serveur.
4. Que cela soit pour RPC en C ou RMI en Java, les proxys servent à encoder les parties spécifiques à un serveur donné et aux procédures qu'ils implémentent.
5. Avec RPC en C, c'est le couple (Nom de serveur, Numéro de version) qui joue le rôle de la chaîne de caractère enregistrée dans le `rmiregistry`.

Question C:
Sur RMI en Java:

1. Une des différences importantes avec RPC est le nommage syntaxique du serveur accédé à distance (ici par une “notation pointée”).
2. Tous les types Java peuvent être passés en paramètre des appels distants, sauf les types dits “final”.
3. Il est également nécessaire de faire un traitement particulier pour pouvoir passer en paramètre des références vers d’autres objets distants.
4. Si des cycles existent dans les objets passés en paramètre des appels distants, ils sont automatiquement supprimés.
5. Il est possible de programmer un objet accessible à distance qui ne permet pas d’appeler toutes ses routines publiques à distance.

Question D:

1. Un avantage important de RMI est la possibilité de télécharger le code des classes par le même canal que les objets Java sérialisés.
2. La notion de “codebase” pour les objets qui transitent par RMI correspond à une sorte d’étiquette attachée à l’objet qui permettra de retrouver, si cela est nécessaire, l’endroit où est le .class correspondant.
3. Dès que l’on fait du polymorphisme entre clients et serveurs RMI, il devient nécessaire de prévoir le téléchargement de code.
4. Un polymorphisme paramétrique va induire un transit de .class de la machine du client vers la machine du serveur.
5. RMI permet de contrôler l’ordre d’exécution des requêtes sur un serveur.

Question E:

1. Le langage IDL de CORBA et l’implémentation de l’ORB apportent une intéropérabilité au niveau des langages (Java, C++, Cobol, etc.).
2. Par contre, il n’est pas suffisant, et il a fallu ajouter quelque chose, afin d’obtenir une intéropérabilité entre implémentations différentes de CORBA.
3. Un avantage de RMI: grâce à la portabilité de Java, on obtient automatiquement une intéropérabilité entre implémentations RMI.
4. CORBA permet de faire la recherche d’un serveur sur un type d’interface, alors que RMI ne le permet pas.
5. Si l’on utilise Jini avec RMI, il est possible de trouver un serveur sans savoir sur quelle machine il tourne, mais également de trouver celui-ci à partir de l’interface recherchée.

2 (5 points)

Une entreprise doit répartir entre plusieurs machines un système centralisé existant. Les fonctions C suivantes doivent pouvoir être appelées à distance:

```
int getSalaire (char *nom, char* prenom, short indice);
employes getByIndices (short indice);
```

où:
la première fonction retourne simplement le salaire (un entier) d’un employé donné (`nom` et `prenom` passé en paramètre), celui-ci étant à un `indice` également passé en paramètre,
la seconde retourne l’ensemble des noms, et prénoms des employés se trouvant à un indice donné.

Le responsable du système informatique décide d’utiliser un RPC en C à l’aide du système RPCGEN, et vous demande de réaliser les tâches suivantes (bien respecter l’ordre et numéroter les réponses):

a. Écrire le fichier d’interface nécessaire à la première procédure.

b. Écrire ce qu’il faut ajouter au fichier d’interface pour la seconde procédure.

c. En supposant que la machine serveur s’appelle `serveur.unice.fr`, écrire la ou les instructions qui établissent la connexion avec le serveur.

d. Écrire du coté du client l'appel à la fonction `getSalaire`.

e. Écrire le code de la procédure qui implémente `getByInfices` du coté du serveur (c'est à dire avec les conventions vues en cours: la procédure qui se trouve dans le fichier `nomDuProgramme_srp.c`).

Note:
Pour la dernière question, on supposera que l'on dispose de :
`getByIndicesImp (...)`
qui correspond à l'implémentation sur le serveur de la procédure
employes getByIndices (short indice);
A vous de définir son profil.

3 (7 points)

On souhaite réaliser en Java un logiciel 3 tiers pour la réservation de DVD. Les composants du système seront une interface client du type AWT ou Swing, et les deux autres tiers.

Les DVD sont caractérisés par juste un *titre*, une *année*, et une *référence*. Les utilisateurs du système pourront simplement faire des recherches en donnant une chaîne de caractères; ils pourront ainsi voir les caractéristiques de tous les DVD qui sont disponibles à cet instant pour la location (ni réservés, ni résultats d'une recherche en cours par un autre utilisateur), et qui ont cette chaîne dans leur titre. Ils disposent ensuite de 5 minutes pour réserver l'un d'entre eux. S'il n'en a pas réservé un au bout de ce délai, ou s'il fait une autre requête, tous ces DVD redeviennent accessibles à la réservation pour d'autres utilisateurs. Une réservation étant faite, le client doit aller chercher physiquement son DVD dans le magasin.

Il est imposé d'utiliser l'équivalent d'un **session bean statefull** par utilisateur, ainsi qu'un pattern (motif de conception) de type "*factory*" pour établir la session.

Au niveau métier, on va considérer uniquement les 2 opérations suivantes:

```
<liste-de-DVD> rechercherDVD (string chaine);
<integer-numero-de-confirmation> reserverDVD (int reference)
```

Attention:
- Pour cette question, il ne s'agit surtout pas d'écrire des pages et des pages de code, mais simplement les éléments qui vous semblent pertinents, liés aux aspects répartition.
- La taille globale des réponses attendues est d'environ une page et demie, ou 2 pages pour les 5 questions.
- Ne pas rentrer dans le détail des interactions avec le 3ième tiers, ni des aspects interface graphique du 1er tiers.
- Bien structurer vos réponses.

a. (1 point)
Faire un dessin du système à l'exécution, en prenant soin de présenter les différents tiers et leur rôles, les objets accessibles à distance, les éléments de l'infrastructure nécessaire à l'établissement des connexions, etc.

b. (1 point)
Donner maintenant les interfaces nécessaires à la programmation du système au niveau des 2 premiers tiers.

c. (2 points)
Donner le code du premier tiers pour l'établissement de la connexion avec le second tiers, et l'envoi des deux requêtes mentionnées ci-dessus.

d. (2 points)
Donner les parties pertinentes des classes et du code du tiers du milieu.

e. (1 point)
En pratique, une requête de recherche `rechercherDVD` correspond à une pré-réservation.
C'est le tiers du milieu qui, s'il n'a pas reçu de requête de réservation d'un DVD après une requête de recherche, va au bout de 5 minutes annuler la pré-réservation auprès de la base de donnée, et informer le 1er tiers afin que celui-ci visualise cette information (la perte de la pré-réservation).
Expliquer comment programmer cette fonctionnalité, en donner les grandes lignes.

4 (4 points)

Question de synthèse : en une demi-page, donner les avantages et inconvénients respectifs de la programmation répartie avec Java et Corba.