RICM 4ème année

12/1/2012

Examen de Systèmes Répartis
Durée : 2h, Documents autorisés à l'exception des livres. Le barème est indicatif.

Partie A – Applications Web

Question 1. Dans un répertoire contenant les ressources d'une application web (html,images, jsp, servlet, etc.), les jsp doivent se trouver :

(a) Dans le répertoire "WEB-INF"
(b) A la racine du répertoire
(c) Dans le répertoire "tags"
(d) Dans le répertoire "lib"

Question 2. Dans une architecture MVC (Modèle Vue Contrôler), quel est le rôle :

- Des JSP
- Des servlets

Les servlets représentent la partie Contrôle et les JSP la partie Vue.

Question 3. Quelle est la différence entre les deux méthodes suivantes :

- request.getRequestDispatcher("...").forward(request, response)
- request.getRequestDispatcher("...").include(request, response)

Forward permet à un servlet de transférer complètement le traitement d'un requête à une autre ressource (servlet, HTML file, ..) sur le serveur local. Include permet à un servlet d'inclure le traitement effectué par une autre ressource dans le traitement courant d'une requête.

Question 4. Les servlets implémentent quelles interface(s):

(a)java.servlet.Servlet
(b)java.entreprise.Servlet
(c)javax.servlet.Servlet
(d)java.servlet.HttpServlet
(e) javax.servlet.http.HttpServlet

Les servlets implémentent l'interface javax.servlet.Servlet.

Partie B – Approche à services (OSGi)

Question 1. Pourquoi faut-il que le module d'un bundle soit résolu pour que l'activateur soit créé et démarré? (5 lignes max)

Le module d'un bundle doit être résolu parce que toute classe importée par un bundle est susceptible d'être utilisée au moment du démarrage d'un bundle. Si le module n'est pas résolu, c'est que certaines de ces classes ne sont pas disponibles et cela risque d'aboutir à un ClassNotFound exception lors du démarrage du bundle.

Question 2. Sous quelles conditions deux références de classes Java représentent la même classe à l'exécution? (2 lignes max)

Deux références de classes Java représentent la même classe à l'exécution lorsqu'elles référencent le même objet class.

Partie C – Systèmes répartis à objets

On s'intéresse à la réalisation d'un service de location de véhicules comprenant une agence par grande ville. Le protocole de réservation est le suivant :

- Un client consulte la liste des véhicules disponibles pour la période souhaitée à l'agence de son choix.
- Après consultation, il peut réserver un véhicule. Le système de réservation renvoie un identifiant unique associé à la réservation.
- Plus tard, le client se reconnecte et libère la voiture.

On réalise l'application en Java en utilisant RMI comme support de la distribution. Les fonctions fournies par le système de réservation d'une agence sont :

- *ListeVehicules rechercherVéhicule (String client, Periode periode, TypeVehicule typeVehicule)*
  Renvoie la liste des véhicules disponibles à la période demandée.

- *Reservation reserverVehicule (String client, Periode periode, Vehicule vehicule)*
  Enregistre le triplet *<client, vehicule, periode>* pour une réservation et renvoie un identifiant unique pour cette réservation. Renvoie un code d'erreur si le véhicule n'est pas disponible à la période demandée.

- *libererVehicule (String client, Reservation reservation)*
  Libère une réservation. Si l'identifiant n'existe pas, un code d'erreur est retourné.

On suppose que les interfaces de ces fonctions sont spécifiées dans l'interface Java *AgenceItf*. On suppose également que l'application est composée des types (classe ou interface) suivants: *Periode, TypeVehicule, Vehicule, ListeVehicules, Reservation, Agence*. Les classes *Periode*, *Vehicule* et *TypeVehicule* sont composées d'un ensemble de champs de type String. La classe *Agence* implémente *AgenceItf* et définit une méthode main permettant de lancer le système de réservation pour une agence donnée.

Question 1. Définir l'interface *AgenceItf*. Argumenter sur le choix du passage par valeur ou par référence pour les différents arguments. Différentes options sont possibles et l'on vous demande de décrire les avantages et inconvénients de la solution que vous choisissez. `

```
public interface AgenceItf extends Remote {
 public ListeVehicules rechercherVehicule(String client, Periode p, TypeVehicule tv) throws RemoteException;
 public Reservation reserverVehicule(String client, Periode p, Vehicule v) throws RemoteException;
 public void LibererVehicule(String client, Reservation r) throws RemoteException;
}
```

Les données de type primitif sont automatiquement transférées par valeur.
Les données de type Periode sont des données immuables et de petite taille, il n'y a donc aucun avantage à les transférer par référence – elles dont donc transférées par valeur.
Les données de type Reservation correspondent à des identifiants, donc des données immuables et de petite taille,– elles dont donc transférées par valeur.

Les données de type Vehicule peuvent être transmises par valeur ou bien oar référence. Dans le premier cas, le client peut consulter les informations associées à une voiture localement alors que dabs le deuxième cas, toute consultation engendre un appel distant. Cependant, un avantage du passage par référence est qu'il permet au client de modifier les données associées à une voiture (pour la réserver par exemple) et d'avoir une vision à jour de ces données lorsqu'elles sont modifiées par d'autres clients.

Concernant le résultat ListeVehicules retourné par rechercherVehicule:
Ce résultat est passé par valeur car il représente le résultat d'une fonction de recherche qui n'a pas lieu d'être gardé au niveau du serveur (on ne se préoccupe pas de gérer un cache de résultat coté serveur).

Question 2. Décrire les implications de vos choix sur le passage des arguments sur les différents types de l'application (préciser pour chaque type si c'est une classe ou une interface, et si c'est une classe préciser quelles contraintes s'appliquent sur cette classe).

En termes de contraintes de programmation

Si passage par valeur, la classe doit implémenter l'interface Serializable.

Si passage par valeur, la classe doit implémenter une interface qui étand l'interface Remote, et elle doit étendre la classe UnicastRemoteObject. Toutes les méthodes doivent émettre l'exception RemoteException.

Question 3. Proposer une mise en œuvre de la classe Agence en laissant vide le corps des méthodes.

```
public class Agence implements AgenceItf extends UnicastRemoteObject{

        public ListeVehicules rechercherVehicule(String client, Periode p, TypeVehicule tv) throws RemoteException{
            ...
          }

         public Reservation reserverVehicule(String client, Periode p, Vehicule v) throws RemoteException{
            ...
          }

         public void LibererVehicule(String client, Reservation r) throws RemoteException {
            ...
          }
  ...
}
```

Question 4. Proposer une mise en œuvre de la méthode main de la classe Agence. On supposera que chaque agence s'enregistre auprès du service de nommage de RMI via un nom composé de Agence-<nom-ville>, où <nom-ville> est passé en argument du lancement d'une agence.

```
public class Agence implements AgenceItf extends UnicastRemoteObject{
  …
   public void main(..) {
     LocateRegistry.createRegistry() ;
     Agence agence new Agence(args[0]);
     Naming.rebind(args[0], agence);
   }
}
```

Question 5. Lors de l'enregistrement d'une agence auprès du service de nommage, comment est transmis un objet Agence ? par valeur ? par référence ? au final, qu'est-ce qui est réellement transmis au service de nommage ?

L'objet Agence est transmis par référence. Ce qui est transmis est donc un stub sur l'objet distant Agence.

Question 6. Lorsque la dernière instruction de la méthode main de la classe Agence a terminé d'être exécutée, est-ce que la machine virtuelle Java s'arrête ?

Non la JVM ne s'arrête pas bien que la dernière instruction de la méthode main() ait été exécutée.

Question 7. La classe Reservation contient un identifiant unique permettant d'identifier une réservation de manière globale dans le système. Comment gérer l'allocation de ces identifiants au niveau des différentes agences qui sont mises en œuvre par des objets répartis ?. Proposer une solution. On vous laisse libre de déterminer la structure d'un identifiant de réservation.

Une possibilité est d'associer un identifiant unique à chaque agence. Un numéro de réservation peut alors être composé de l'identifiant de l'agence suivi d'un identifiant unique local à l'agence. Ces identifiant peuvent être des entiers.

Question 8. On souhaite fournir à un client la fonction suivante permettant de récupérer la liste des véhicules libres dans une ville donnée pour une période donnée sans connaître l'agence associée à cette ville. Définir le corps de cette fonction.

ListeVehicule rechercherVehicule (String client, String ville, Periode periode, TypeVehicule typeVehicule)

```
ListeVehicules rechercherVehicule(string client, string ville, Periode p, TypeVehicule tv){
      AgenceItf ag = (AgenceItf) Naming.lookup(ville);
      return ag.rechercherVehicule(p, tv);
}
```

Question 9. Donner la distribution des classes et interfaces associées à cette application, en termes de ce qui doit se trouver du coté client et ce qui doit se trouver au niveau d'une agence.

JVM Agence:
classes Agence,
interfaces Vehicule, Periode, TypeVehicule,
interfaces AgenceItf, ListeVehicules.

JVM client
interface AgenceItf
interface ListeVehicules

Question 10. On considère la fonction suivante qui permet de l'historique des locations pour un client donné. Le type *Historique* correspond à une classe de l'application. Lorsqu'un client invoque cette fonction, peut-il recevoir un résultat dont la classe est inconnue de lui ? A quelles conditions ? Que se passe t-il à ce moment là ?

Historique historiqueClient(String client)

Oui le client peut recevoir un résultat de classe inconnue mais cette classe doit être une sous-classe de la classe Historique. Cette sous-classe sera chargée dynamiquement via le class loader, via l'option codebase si le programme client a été lancé avec cette option.

Partie D – Systèmes répartis à messages

En TP, vous avez réalisé un mécanisme de transfert de fichiers au dessus de TCP/IP. On s'intéresse à la mise en œuvre d'un tel mécanisme au dessus d'UDP. On considère les algorithmes suivants.

```
public void send(File file, string servHostName, int servPort) throws Exception {
        InetAddress servIp = InetAddress.getByName(servHostName);
        DatagramSocket socket = new DatagramSocket();
        FileInputStream stream = new FileInputStream(file);
        byte[] content = new byte[(int) file.length()];
        stream.read(content);
        DatagramPacket packet = new DatagramPacket(contenu, content.length, servIp, servPort);
        socket.send(packet);
        socket.close();
}
```

```
public void recv(File dir, int port) throws Exception {
        DatagramSocket ds = new DatagramSocket(port);
        File f = new File(dir.getAbsolutePath()+ File.separator +i);
        FileOutputStream  fis = new FileOutputStream(f);
        byte[] b = new byte[4096];
        DatagramPacket  dp = new DatagramPacket(b,4096);
        ds.receive(dp);
        fis.write(dp.getData(),0,dp.getLength());
        ds.close();
}
```

Question 1. Dans ces algorithmes, décrire ce qui ne marche pas (au niveau conceptuel).

Problème 1 : l'envoi et la réception d'un fichier de taille supérieure à 4096 octets ne sont pas supportés.
Problème 2 : il n'y a pas d'acknowledge, donc il n'y a pas de gestion des cas de panne engendrant la non réception du fichier ou la réception partielle de celui-ci.

Question 2. Pour chaque problème identifié, proposer un principe de solution.

Problème 1 :

Envoyer les données par paquets successifs de 4096 octets maximum.

Envoyer préalablement un premier message qui indique la taille totale des données qui vont être envoyées et la longueur des paquets.

Problème 2 :

Numéroter les paquets pour gérer la réception dans le bon ordre.

Chaque paquet contient un header qui décrit le numéro du paquet.

Coté réception, temporiser pour les paquets arrivant en avance, dans un buffer d'attente.

Régulièrement, envoyer un acknowledge à l'émetteur précisant la liste des paquets attendus mais toujours non reçus.