Examen Construction d'Applications Réparties

Maîtrise Informatique

Université des Sciences et Technologies de Lille

2003-2004

Session Septembre 2004- 3 heures

Tous documents autorisés

1. Gestion d'un Parking (10 points)

On considère un parking dont on souhaite automatiser la gestion en l'administrant à distance avec un *middleware* CORBA. Le parking peut accueillir au maximum 50 véhicules. Une barrière gère les entrées et les sorties de véhicules : elle délivre un ticket à l'entrée d'un véhicule et elle contrôle le ticket fourni par l'automobiliste à la sortie. Un automate de paiement permet sur présentation du ticket, de régler le montant correspondant au temps de stationnement. L'automobiliste règle le montant du stationnement à l'automate avant de franchir la barrière pour sortir. La barrière, l'automate et le parking sont représentés chacun par un objet CORBA.

L'interface `BarriereItf` de la barrière fournit deux méthodes :

- `entrer` : correspond à l'entrée d'un véhicule dans le parking. Retourne vrai si le véhicule peut rentrer (si place libre), faux sinon (et dans ce cas le véhicule ne peut pas entrer). Cette méthode fournit en paramètre de sortie une structure `Ticket` comprenant un numéro de ticket de type entier et une heure d'entrée sous la forme d'une chaîne de caractères.
- `sortir` : correspond à la sortie d'un véhicule. Retourne vrai si l'automobiliste a réglé le montant du stationnement, faux sinon (et dans ce cas le véhicule ne peut pas sortir). Cette méthode prend en paramètre d'entrée une structure `Ticket`.

L'interface `AutomateItf` de l'automate fournit deux méthodes :

- `payer` : correspond au paiement en fonction de la durée de stationnement. Retourne vrai si le montant fourni par l'automobiliste est suffisant, faux sinon. Cette méthode prend en paramètre d'entrée une structure `Ticket` et un montant de type réel double. Cette méthode fournit également en sortie une valeur de type réel double représentant la monnaie à rendre à l'utilisateur.
- `cestregle` : à partir d'une structure `Ticket` fournie en entrée, retourne vrai si le montant du ticket a été réglé, faux sinon.

L'interface `ParkingItf` du parking fournit trois méthodes :

- `nbPlacesLibres` : retourne un entier indiquant le nombre de places libres dans le parking.
- `entreeVehicule` : signale l'entrée d'un véhicule sur le parking. Ne prend aucun paramètre, ne retourne rien.
- `sortieVehicule` : signale la sortie d'un véhicule du parking. Ne prend aucun paramètre, ne retourne rien.

1 Définir la structure `Ticket` et les interfaces IDL correspondant à ces trois objets CORBA dans un module `ParkingPkg`. (2 points)

Arrivée d'un véhicule

2 Lorsqu'un automobiliste se présente à la `Barrière`, l'objet CORBA barrière interroge l'objet `Parking` avant de laisser entrer l'automobiliste. Pourquoi ? Quelle méthode invoque-t-il ? (0,5 point)

3 A l'issue de cela, il se peut que l'objet `Barrière` invoque la méthode `entreeVehicule` de l'objet `Parking`. Cette méthode n'a pas de paramètres et a pourtant un rôle essentiel. Lequel ? (0,5 point)

Sortie d'un véhicule

4 Décrire le scénario de la sortie d'un véhicule du parking : quel(s) objet(s) appelle(nt) quelle(s) méthode(s) de quel(s) autre(s) objet(s) et pourquoi ? En particulier, on s'attachera à fournir un scénario de fonctionnement permettant à l'objet `Barrière` d'autoriser ou non la sortie du véhicule. (1 point)

Implantation du parking

5 Ecrire le code de la classe `ParkingImpl` implantant l'interface `ParkingItf`. (1 point)

Ajout d'une barrière

6 On souhaite ajouter une deuxième barrière d'entrée/sortie au parking. Cela modifie-t-il les interfaces définies à la question 1 ? Si oui comment, si non pourquoi. (1 point)

7 La classe `ParkingImpl` de la question 5 est modifiée suite à l'ajout d'une barrière. Après avoir rappelé les méthodes invoquées lors de l'arrivée d'un véhicule, expliquez en quoi consiste cette modification (on ne vous demande pas de la programmer). (1 point)

8 Même question pour la sortie d'un véhicule. (1 point)

Evolution du parking

9 On souhaite maintenant connaître les heures d'entrée et de sortie des véhicules. Sans donner le code, expliquez comment vous pourriez faire cela. (1 point)

2. Gestion d'une bibliothèque (10 points)

Le système d'information de la bibliothèque municipale de votre ville a été automatisé, il y a quelques années. Les bibliothécaires disposent d'un logiciel accessible depuis des terminaux de type ordinateur de bureau. Ce logiciel leur permet de gérer à la fois les livres, les usagers et les emprunts. En quelques lignes, voici les opérations possibles sur ce logiciel :

Le bibliothécaire peut :

- ajouter un nouveau livre,
- supprimer un livre,
- consulter la liste des livres selon différents critères (auteur, titre, éditeur, emprunteur),
- consulter les données associées à un livre à partir de son titre ou de son ISBN (identifiant d'un livre),
- ajouter un nouvel usager,
- supprimer un usager,
- consulter la liste des usagers selon différents critères (nom, catégorie (scolaire, étudiant, senior, travailleur, chômeur), adresse),
- consulter les données associées à un usager à partir de son nom.

Le bibliothécaire gère les emprunts pour un usager. Ce dernier peut :

- emprunter 1 livre (au maximum 3) pour une durée de 15 jours,
- ramener un livre avant ou à l'expiration de ce délai de 15 jours. Cela lui permet d'emprunter un nouveau livre,
- demander une prolongation de 15 jours de son prêt (renouvelable deux fois, c'est-à-dire au maximum un mois),
- avoir une pénalité, si le livre n'a pas été ramené à temps. Cette pénalité consiste en une interdiction d'emprunt pendant la durée correspondant au retard.

Un usager peut :

- consulter le catalogue des livres selon différents critères (no ISBN, auteur, titre, éditeur) et permet de connaître, pour chaque livre, son emplacement dans la bibliothèque et sa disponibilité.

1 Identifier la structure de données nécessaires permettant de gérer les livres, les usagers et les emprunts. Commenter votre solution (2 points).

2 A partir de cette structure de données, identifiez les classes Java nécessaires et donnez les interfaces de ces classes Java (1 point)

3 Le bibliothécaire souhaite que l'accès au logiciel, en particulier pour la partie consultation, soit maintenant possible au travers d'un navigateur Web sans pour cela retoucher au logiciel initial. L'informaticien propose une première solution client/serveur à base de Web Services. Rappelez brièvement en quoi consiste cette technique et décrivez les avantages et les inconvénients (1 point).

4 Proposez une autre solution. Quels sont ses avantages et inconvénients par rapport à la solution proposée par l'informaticien ? (1 point)

Nous restons sur la solution à base de Web Services. Les deux messages suivants circulent entre le client et le serveur :

1er message du client vers le serveur :

```
1  POST /Biblio HTTP/1.1
```

```
2  Host: www.biblioserver.org
3  Content-Type: text/xml; charset="utf-8"
4  Content-Length: 999
5  SOAPAction: http://some.host/SOAPServer/convertisseur
6
7  <SOAP-ENV:Envelope
8   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
9   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
10  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
11   <SOAP-ENV:Body>
12    <m:requestBook
13     xmlns:m="urn:xmethods-Bibliotheque"
14     SOAP-ENV:encodingStyle=
15         "http://schemas.xmlsoap.org/soap/encoding/">
16     <identifiant xsi:type=xsd:string>
17       7890765R55
18     </identifiant>
19    </m:requestBook >
20   </SOAP-ENV:Body>
21   </SOAP-ENV:Envelope>
```

2ème message du serveur vers le client :

```
22   HTTP/1.1 200 OK
23   Content-Type: text/xml; charset="utf-8"
24   Content-Length: 999
25
26   <SOAP-ENV:Envelope
27    xmlns:SOAP-ENV=
28     "http://schemas.xmlsoap.org/soap/envelope/"
29    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
30    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
31    <SOAP-ENV:Body>
32     <m:requestBook
33      xmlns:m="urn:xmethods-Bibliotheque"
34      SOAP-ENV:encodingStyle=
35     "http://schemas.xmlsoap.org/soap/encoding/">
36      <return xmlns:ns2="urn:bibliotheque"
37           xsi:type="ns2:product">
38       <name xsi:type="xsd:string">Victor Hugo </name>
39       <titre xsi:type="xsd:string">Les Miserables</titre>
40       <emplacement xsi:type="xsd:string"> B5 </emplacement>
41       <dispo xsi:type="xsd:string"> oui </dispo>
42      </return>
43     </m:requestBook >
44    </SOAP-ENV:Body>
45   </SOAP-ENV:Envelope>
```

5 Donnez la signification des lignes 1 à 5 dans le premier message et 21 à 24 dans le second message (0,5 point)

6 Que définissent les lignes 7, 8, 9, 10 ? (0,5 point)

7 Expliquez la différence entre les lignes 8, 9, 10 et la ligne 13 ? (0,5 point)

8 A quoi sert l'attribut "`xsi:type`" dans les lignes 37, 38, 39, 40, 41 ? (0,5 point)

9 Pourquoi préfixe-t-on `string` par `xsd:` dans la ligne 38 ? (0,5 point)

10 Le langage de description de service (WSDL) sert à définir les échanges. La première partie de cette définition concerne le type des messages échangés. Définissez le type du message de retour (balise `return`) sous forme d'un XML schéma (1 point)

11 La bibliothèque souhaite maintenant proposer un accès à distance de façon à étendre son service de prêts à des usagers éloignés. Quelle est la première étape à mettre en œuvre pour faire connaître ce service ? (1 point)

12 Ce service étant maintenant accessible via Internet, il faut en sécuriser l'accès. Quelle solution proposez vous ? (0,5 point)