Université de Lille Sciences et Technologies — Juin 2016
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 (5 points)

1.1 Quelle est la différence entre un *thread* et un processus ?
1.2 Quel est le point commun entre WDSL et Apache Thrift ?
1.3 Quelle est la différence principale entre Java RMI et Akka ?
1.4 Qu'est-ce que la notion de session dans une JSP ?
1.5 Outre le fait qu'il s'agit du langage Java, quel est le point commun entre JDBC et les EJB entités ?

## 2 REST (3 points)

On s'intéresse à une architecture répartie de type REST dont les ressources sont les commandes gérées par un site de commerce en ligne.

2.1 Que vont permettre de faire les quatre opérations *Create Read Update Delete* associées au principe REST dans cette architecture ? (1 point)

2.2 En présence de concurrence, par exemple avec un serveur *multi-thread*, quelle politique de synchronisation proposez-vous de mettre en œuvre pour les quatre opérations *Create Read Update Delete* ? Indiquer le rôle joué par chaque opération dans la politique de synchronisation. (1 point)

2.3 Au delà des aspects liés à l'implantation (HTTP, XML, HTML, JSON, etc.), quelle est la différence principale entre une architecture REST et une architecture répartie utilisant SOAP ? (1 point)

## 3 Gestion d'un parking (7 points)

On considère un parking dont on souhaite automatiser la gestion en l'administrant à distance avec des services définis à l'aide d'Apache Thrift. 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.

Le service `BarriereItf` de la barrière fournit deux méthodes :
- `entrer` : correspond à l'entrée d'un véhicule dans le parking. Retourne une structure `Ticket` si le véhicule peut rentrer (si place libre), sinon lève l'exception `ParkingComplet` (et dans ce cas le véhicule ne peut pas entrer). La structure `Ticket` comprend 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 une structure `Ticket`.

Le service `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 une structure `Ticket` et un montant de type réel double correspondant au montant fourni par l'automobiliste.
- `cestpaye` : à partir d'une structure `Ticket` fournie en paramètre, retourne vrai si le montant du ticket a été réglé, faux sinon.

Le service `ParkingItf` du parking fournit trois méthodes :
- `nbPlacesLibres` : retourne un entier indiquant le nombre de places libres dans le parking. Ne prend aucun paramètre.
- `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.

3.1 Définir ces trois services avec Apache Thrift. (2 points)

### Arrivée d'un véhicule

3.2 Lorsqu'un automobiliste se présente à la barrière, celle-ci interroge le parking avant de laisser entrer l'automobiliste. Pourquoi ? Quelle méthode invoque-t-il ? (0,5 point)

3.3 À l'issu de cela, il se peut que la barrière invoque le service `entreeVehicule` du parking. Ce service n'a pas de paramètres et a pourtant un rôle essentiel. Lequel ? (0,5 point)

### Sortie d'un véhicule

3.4 Décrire en français le scénario de la sortie d'un véhicule du parking quelles sont les méthodes appelés, par qui et pourquoi ? En particulier, on s'attachera à fournir un scénario de fonctionnement permettant à la barrière d'autoriser ou non la sortie du véhicule. (1 point)

### Ajout d'une barrière

3.5 On souhaite ajouter une deuxième barrière d'entrée/sortie au parking. Cela modifie-t-il les services définis à la question 1 ? Si oui comment, si non pourquoi. (0,5 point)

3.6 Après avoir rappelé les méthodes invoquées lors de l'arrivée d'un véhicule, expliquez en français quelle conséquence l'ajout d'une barrière a sur l'implémentation des services. (1 point)

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

### Evolution du parking

3.8 On souhaite maintenant connaître les heures d'entrée et de sortie des véhicules. Expliquer en français comment faire cela. (0,5 point)

## 4 Gestion de magasin avec Java EE et REST (5 points)

On considère une application Java EE qui permet de gérer des clients et des factures. Chaque client a un identifiant (entier) et un nom (chaîne de caractères). Chaque facture a un identifiant (entier), un montant (réel double) et fait référence au client devant régler cette facture. On souhaite pouvoir créer, supprimer et consulter des clients et des factures. On souhaite également pouvoir calculer le montant total des factures réglées par un client. L'interface d'accès à l'application doit pouvoir se faire avec REST. On ne peut pas supprimer la ou les factures d'un client existant.

4.1 Quelle(s) ressource(s) REST proposez-vous de définir pour cette application ? (1 point)

4.2 Quel(s) type(s) de *bean*(s) session et/ou entité proposez-vous de définir ? (1 point)

4.3 Écrire le code Java des interfaces et des classes correspondantes. (3 points)