TD JMS – *Java Message Service*

Rappels de cours

1) Qu'est-ce qu'un middleware orienté message (MOM) ?
   Quelles différences faites-vous entre un MOM et JMS ?
2) Parmi les applications suivantes, indiquez si l'utilisation d'un middleware orienté message est adapté, en justifiant vos réponses :
   a. Visiophonie
   b. Consultation d'informations météo à distance
   c. Messagerie d'entreprise (courrier électronique, messages vocaux)

Exercice 1

Nous souhaitons simuler trois unités de production dans une ligne automatisée d'usine. La première unité produit des pièces qui sont consommées par la seconde. Cette dernière produit ensuite une nouvelle pièce qui est consommée par la troisième.

1.1 Représentez les interactions entre entités réparties à l'aide d'un diagramme de séquences UML.
1.2 Représentez sur un schéma l'architecture JMS : c'est-à-dire, les différentes entités logicielles et les communications en utilisant des sujets de discussion (*topics*) et/ou des files de messages (*queues*).
1.3 Quel type de message utiliseriez-vous pour le corps des messages échangés ?

Exercice 2

Le but de cet exercice est de simuler un service web qui diffuse les cours de la bourse d'un certain nombre de titres à un ensemble de courtiers en bourse (un courtier pouvant bien sûr s'abonner à plusieurs titres). Un titre est caractérisé par un nom et une valeur du cours de bourse actuel.

2.1 Représentez les interactions entre entités réparties à l'aide d'un diagramme de séquences UML.
2.2 Représentez sur un schéma l'architecture JMS : c'est-à-dire, les différentes entités logicielles et les communications en utilisant des sujets de discussion (*topics*) et/ou des files de messages (*queues*).
2.3 Quel type de message utiliseriez-vous pour le corps des messages échangés ?

Exercice 3

Une application de type `eCommerce` en JMS : cette application simule une application complète de `eCommerce` avec les participants suivants :

- `ServeurWeb` : un serveur web sur lequel le client choisit des objets à acheter ;
- `ServeurClientèle` : centralise le traitement des commandes ;
- `ServeurStocks` : vérifie la disponibilité des objets commandés ;
- `ServeurFacturation` : centralise le traitement des paiements bancaires ;
- `ServeurBanque` : vérifie les références bancaires des clients et gère les ordres de paiement ;
- `ServeurLivraison` : s'occupe de la livraison des commandes.

Remarques :

- Pour qu'une commande soit livrée par le `ServeurLivraison`, il faut que le `ServeurStocks` et le `ServeurFacturation` l'aient validée ;
- Le `ServeurClientèle` s'occupe de cette validation ;
- Le `ServeurFacturation` contacte le `ServeurBanque` pour s'assurer que les références bancaires sont correctes et pour demander un ordre de paiement ;

3.1 Représentez les interactions entre entités réparties à l'aide d'un diagramme de séquences UML.
3.2 Représentez sur un schéma l'architecture JMS : c'est-à-dire, les les différentes entités logicielles et les communications en utilisant des sujets de discussion (*topics*) et/ou des files de messages (*queues*).
3.3 Faites un scénario en indiquant quels messages transitent par quels files/sujets.

Exercice 4

On se propose de réaliser une application répartie avec JMS permettant la consultation à distance du solde et de l'historique d'un compte bancaire.

4.1 Diagramme de séquence (idem exercice précédent)
4.2 Architecture JMS (idem exercice précédent)
4.3 Décrivez les messages qui seront échangés entre les applications : nom du message, type de données du corps et contenu du message.