

IPST-CNAM
Architecture Logicielles
NSY 205
Jeudi 18 Février 2016

Durée : **2 h 30**
Enseignant : LAFORGUE Jacques

1ère Session NSY 205

CORRECTION

1^{ère} PARTIE – SANS DOCUMENT (durée: 1h15)

1. QCM (35 points)

Mode d'emploi :

Ce sujet est un QCM dont les questions sont de 3 natures :

- **les questions à 2 propositions**: dans ce cas une seule des 2 propositions est bonne.
 - +1 pour la réponse bonne
 - -1 pour la réponse fausse
- **les questions à 3 propositions** dont 1 seule proposition est bonne
 - + 1 pour la réponse bonne
 - -1/2 pour chaque réponse fausse
- **les questions à 3 propositions** dont 1 seule proposition est fausse
 - + 1/2 pour chaque réponse bonne
 - -1 pour la réponse fausse

Il s'agit de faire une croix dans les cases de droite en face des propositions.

On peut remarquer que cocher toutes les propositions d'une question revient à ne rien cocher du tout (égal à 0).

Si vous devez raturer une croix, faites-le correctement afin qu'il n'y ait aucune ambiguïté.

N'oubliez pas d'inscrire en en-tête du QCM, votre nom et prénom.

Vous avez droit à **4 points** négatifs sans pénalité.

| | |
|------|---------|
| NOM: | PRENOM: |
|------|---------|

| | | |
|--|-----|------|
| L'Architecture Fonctionnelle est la description des exigences dites "non-fonctionnelles" identifiées pendant la conception de l'architecture d'un Système d'Information. | | Q 1. |
| 1 | OUI | |
| 2 | NON | X |

| | | |
|---|-----|------|
| Dans la démarche d'architecture d'un Système d'Information, l'Architecture Technique est le déploiement de la Configuration Architecturale dans une technologie donnée. | | Q 2. |
| 1 | OUI | X |
| 2 | NON | |

| | | |
|--|---|------|
| Dans la démarche d'architecture d'un Système d'Information, l'Architecture Dynamique : | | Q 3. |
| 1 | définit sur quoi, et où, s'exécutent les composants de son architecture. | |
| 2 | définit le comportement dynamique interne de chacun des composants de son architecture. | X |
| 3 | définit la coopération des composants entre eux au sein de son architecture. | X |

| | | |
|--|-----|------|
| Dans la démarche d'architecture, un connecteur entre deux composants définit toujours un lien distant de communication de machine à machine. | | Q 4. |
| 1 | OUI | |
| 2 | NON | X |

| | | |
|--|-----|------|
| Quand cela est possible, dans une démarche d'architecture, on essaye de privilégier un couplage faible entre les composants. | | Q 5. |
| 1 | OUI | X |
| 2 | NON | |

| | | |
|--|---|------|
| Une architecture 4-tiers est un style d'architecture qui est composée des tiers suivants : | | Q 6. |
| 1 | Client + Présentation + Composants métier + Base de Données | X |
| 2 | Client + Composants métier + Composant DAO + Base de Données | X |
| 3 | Client + Présentation + Base de Données + Gestion des erreurs | |

| | | |
|--|--|------|
| Une couche de persistance des données d'un Système d'Information est : | | Q 7. |
| 1 | une couche logicielle développée soi-même permettant de garder en base de données les attributs des objets (données) | X |
| 2 | une couche logicielle qui permet de garder en mémoire d'un composant dynamique les attributs des objets (données) | |

| | | |
|--|-----|------|
| Dans une architecture N-tiers la persistance des données de la couche métier peut être réalisée en utilisant l'approche ORM (Object Relationnel Mapping) | | Q 8. |
| 1 | OUI | X |
| 2 | NON | |

| | | |
|-----------------------------|--|------|
| L'approche ORM consiste à : | | Q 9. |
| 1 | réaliser la correspondance entre les objets de la couche métier et les tables d'une base de données relationnelle. | X |
| 2 | réaliser la correspondance entre les objets de la couche métier et les objets de la couche de présentation. | |

| | | |
|---|-----|-------|
| Un DAO (Data Access Object) est une interface d'accès aux objets de la couche métier utilisée par les IHM du Système d'Information permettant ainsi de réaliser un couplage faible entre la couche de présentation et la couche métier. | | Q 10. |
| 1 | OUI | |
| 2 | NON | X |

| | | |
|--|---|-------|
| Dans un ORM (Object Relationnel Mapping), on veut réaliser le mapping d'une relation 1..* entre deux classes, par exemple Departement et Employe (un département contient plusieurs employés). On peut réaliser ce mapping : | | Q 11. |
| 1 | sans table de jointure, en mettant la clef primaire de la table EMPLOYE dans la table DEPARTEMENT. | |
| 2 | sans table de jointure, en mettant la clef primaire de la table DEPARTEMENT dans la table EMPLOYE. | X |
| 3 | avec une table de jointure, en mettant les clefs primaires des tables DEPARTEMENT et EMPLOYE dans la table de jointure. | X |

| | | |
|---|-----|-------|
| En JPA, l'annotation @OneToMany permet de traduire le mapping d'une relation 1..* | | Q 12. |
| 1 | OUI | X |
| 2 | NON | |

| | | |
|---|-----|-------|
| Le mapping ORM de l'héritage de classe peut se faire en créant autant de tables qu'il existe de classes réelles de l'arbre d'héritage | | Q 13. |
| 1 | OUI | X |
| 2 | NON | |

| | | |
|---|-----|-------|
| La technologie middleware RMI (Remote Method Invocation) est une technologie bien adaptée pour réaliser des architectures de type MOM (Middleware Orientés Message) | | Q 14. |
| 1 | OUI | |
| 2 | NON | X |

| | | |
|---|-----|-------|
| Dans une architecture à base de composant, un des principes de base est d'utiliser les propriétés d'un Framework pour prendre en charge l'exécution des composants de son Système d'Information | | Q 15. |
| 1 | OUI | X |
| 2 | NON | |

| | | |
|---|---|-------|
| Dans une architecture à base de composant, l'Injection de Dépendance est un principe fort qui : | | Q 16. |
| 1 | permet de relier statiquement l'invocation d'un composant à son implémentation concrète. | |
| 2 | permet de relier dynamiquement l'invocation d'un composant à son implémentation concrète. | X |
| 3 | permet de relier, à travers le réseau, les composants entre eux. | |

| | | |
|--|-----|-------|
| En J2EE, les EJB (Entreprise Java Beans) sont des composants qui s'exécutent dans le container "Web Layer" qui assure la représentation des données métier sur le composant Client | | Q 17. |
| 1 | OUI | |
| 2 | NON | X |

| | | |
|---|--|-------|
| Un client envoie 2 requêtes consécutives (REQ1, REQ2) à un EJB Stateless et à un EJB Stateful. La différence entre ces deux EJB est que : | | Q 18. |
| 1 | le EJB Stateless assure que si la requête REQ1 change l'état de l'EJB alors la requête REQ2 s'exécute avec l'état qui a été changé par la requête précédente REQ1. | |
| 2 | le EJB Stateful assure que si la requête REQ1 change l'état de l'EJB alors la requête REQ2 s'exécute avec l'état qui a été changé par la requête précédente REQ1. | X |

| | | |
|--|-----|-------|
| Pour s'abonner à une file de message (canal), un EJB Stateful peut créer un Message Driven Beans qui sera appelé quand un message sera déposé dans la file de message. | | Q 19. |
| 1 | OUI | X |
| 2 | NON | |

| | | |
|--|-----|-------|
| Dans une architecture J2EE, il faut soi-même faire l'instanciation des EJB Stateful car il faut initialiser les attributs métier de la classe EJB. | | Q 20. |
| 1 | OUI | |
| 2 | NON | X |

| | | |
|---|---|-------|
| Un MOM (Middleware Orienté Message) est : | | Q 21. |
| 1 | un composant de son Système d'Information permettant d'appeler une méthode distante sur un objet distribué sur le réseau. | |
| 2 | un middleware permettant un service d'échanges entre applications ou composants applicatifs. | X |

| | | |
|---|-----|-------|
| Dans un MOM, le mode "Queue" de communication par message assure que tous les consommateurs d'un canal d'évènement reçoivent bien le message. | | Q 22. |
| 1 | OUI | |
| 2 | NON | X |

| | | |
|---|-----|-------|
| Dans un MOM, le modèle de communication Push-Pull correspond au mode "Queue". | | Q 23. |
| 1 | OUI | X |
| 2 | NON | |

| | | |
|--|---|-------|
| Une architecture logicielle basée sur un modèle MOM permet de réaliser | | Q 24. |
| 1 | un couplage faible entre les composants | X |
| 2 | un couplage fort entre les composants | |

| | | |
|---|-----|-------|
| Une architecture WEB Services permet de réaliser l'architecture des composants des Client WEB | | Q 25. |
| 1 | OUI | |
| 2 | NON | X |

| | | |
|--|---|-------|
| Le protocole SOAP des architectures WEB Services est un protocole basé sur : | | Q 26. |
| 1 | les standards d'Internet (HTTP) | X |
| 2 | les standards d'Intranet (RMI ou CORBA) | |

| | | |
|---|-----|-------|
| La complexité de mise en œuvre du protocole SOAP (déploiement d'une amorce sur le client) nécessite de faire une configuration particulière des Firewalls des serveurs. | | Q 27. |
| 1 | OUI | |
| 2 | NON | X |

| | | |
|-----------------------------------|--|-------|
| Dans les WEB Services, WSDL est : | | Q 28. |
| 1 | un composant particulier qui sert d'interface entre le client et le serveur. | |
| 2 | un langage de description de l'interface des services du serveur. | X |

| | | |
|---|---|-------|
| Dans les WEB Services, le WSDL définit un service comme un ensemble d'opération. Chacun de ces opérations est : | | Q 29. |
| 1 | le nom d'une méthode distante implémentée sur le serveur | |
| 2 | constitué (le plus souvent) de deux messages SOAP (un pour l'appel et un pour le résultat) échangés entre le client et le serveur | X |

| | | |
|---|-----|-------|
| Dans les WEB Service, le protocole SOAP est le seul protocole utilisable pour réaliser l'appel à un service d'un Web Service. | | Q 30. |
| 1 | OUI | |
| 2 | NON | X |

| | | |
|---|-----|-------|
| Dans une architecture Web Services, le client doit être écrit dans le même langage de programmation que celui utilisé pour écrire le serveur contenant l'implémentation des services. | | Q 31. |
| 1 | OUI | |
| 2 | NON | X |

| | | |
|---|-----|-------|
| Une architecture REST (Representational State Transfer) est une architecture dans laquelle le protocole utilisé pour la communication entre le client et le serveur est uniquement le protocole HTTP. | | Q 32. |
| 1 | OUI | X |
| 2 | NON | |

| | | |
|---|-----|-------|
| Dans une architecture REST, la ressource est nécessairement un composant physique du serveur (par exemple un EJB, un DAO, un objet RMI, ...). | | Q 33. |
| 1 | OUI | |
| 2 | NON | X |

| | | |
|---|-----|-------|
| Dans une architecture REST, pour le client, l'identification d'une ressource est une URI (Uniform Resource Identifier). | | Q 34. |
| 1 | OUI | X |
| 2 | NON | |

| | | |
|---|-----|-------|
| Une architecture REST est aussi une Architecture à Base de Composants dans laquelle les composants de la couche métier sont les RESSOURCES sur lesquels les clients demandent des propriétés. | | Q 35. |
| 1 | OUI | |
| 2 | NON | X |

Fin du QCM

Suite (Tournez la page)

2. Questions libres (15 points)

QUESTION NUMERO 1

Les 3 façons de réaliser le mapping de l'héritage de classe sont :

1/ Représentation par une table unique: toutes les classes d'une arborescence d'héritage sont alors représentées par une seule table. et un champ type permettant de déterminer le type réel de l'objet.

2/ Chaque classe (concrète ou abstraite) est représentée par une table spécifique contenant une clef primaire de jointure pour obtenir les caractéristiques de l'objet.

3/ Seules les classes concrètes ont un correspondant en table, et chacune des tables regroupe toutes les informations afférentes à un objet de la classe représentée. Chaque classe concrète est donc représentée par une table indépendante qui contient tous les attributs, y compris les attributs hérités.

QUESTION NUMERO 2

1/ Contrôle orienté procédure : dans ce cas de coopération les composants A et B sont appelés par un autre composant appelé "chef d'orchestre".

Le principe général de ce cas de coopération : le composant "chef d'orchestre" implémente la logique métier de l'appel des composants métiers. Cette logique (algorithme) peut être paramétrable.

2/ Contrôle orienté évènements : dans ce cas de coopération, le composant A envoie un évènement à B.

Le principe général de ce cas de coopération : les composants exécutent leurs traitements en fonction des évènements reçus et envoyés entre les composants sans nécessairement que le producteur d'évènements ne connaissent l'existence des consommateurs : couplage très faible entre les composants.

3/ Contrôle orienté données : dans ce cas de coopération le composant A dépose une donnée dans un autre composant appelé "Repository. Le composant B vient chercher la donnée du "Repository".

Le principe général de ce cas de coopération : les données sont gérées dans un référentiel (Repository) qui gère la synchronisation entre les "écrivains" et "les lecteurs". Exemple : les bases de données.

QUESTION NUMERO 3

La technologie J2EE s'applique très bien à la réalisation des architectures N-tiers car J2EE gère des Containers : Client Container, Web Container, EJB Container, Entity Container. Chacun de ces containers permettent d'implémenter chaque couches d'un modèle N-Tiers : couche de présentation, couche métier, couche données. Il y a une bijection entre containers et couches.

Fin de la 1^{ère} partie sans document

2ème PARTIE – AVEC DOCUMENT (durée: 1h15)

3. PROBLEME (50 points)

Question 1 :

Pour le schéma : 25 points

Pour les commentaires : 10 points

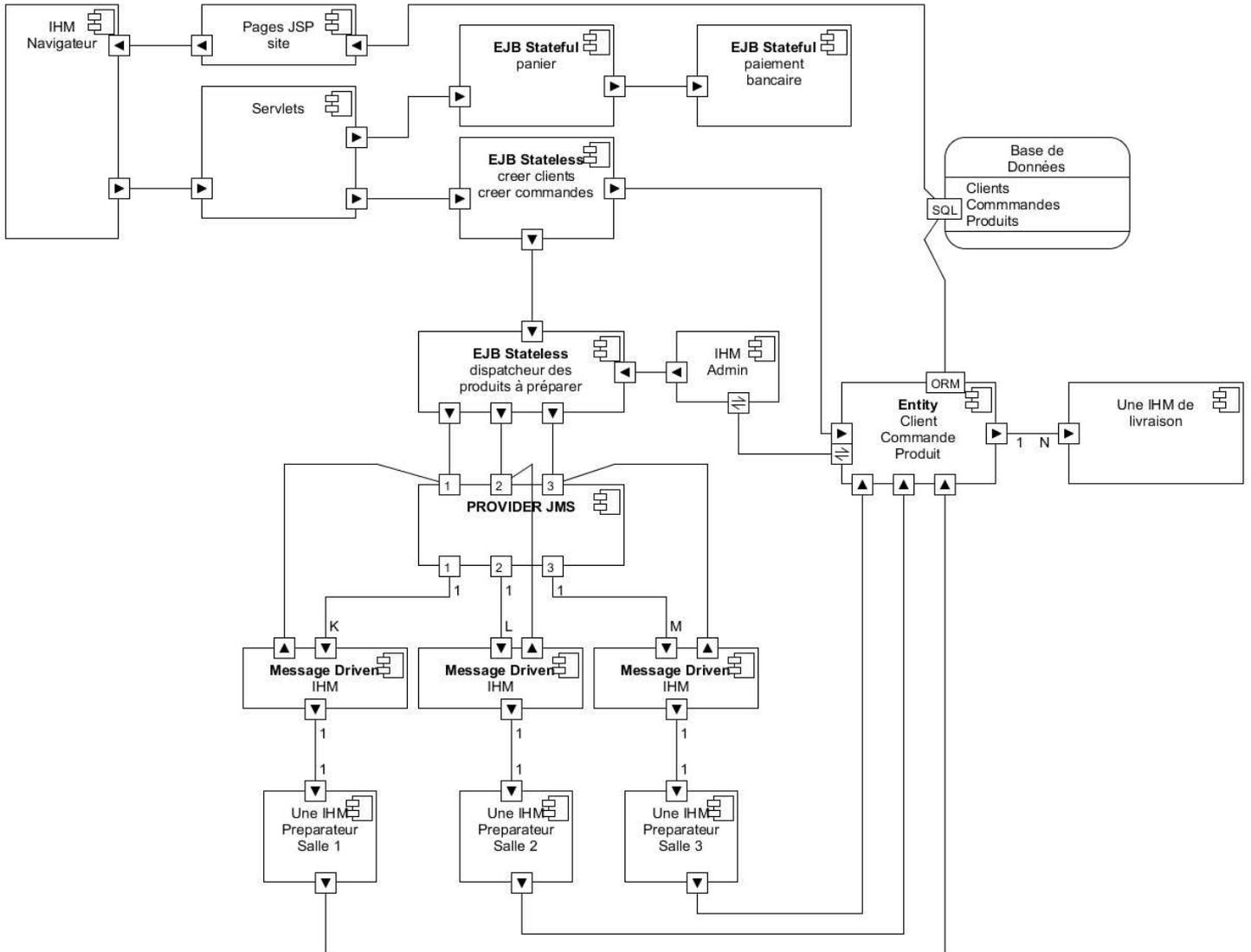


Schéma d'architeture Technique du SI

L'inscription d'un client par internet se fait via un Servlet qui utilise un EJB Stateless "créer Clients" qui, via l'Entity Produit, enregistre le client en base de données.

La consultation du catalogue des produits pouvant être commandés par internet est réalisée par une page JSP qui réalise une requête SQL sur la base de données contenant les produits.

La gestion des commandes réalisées par Internet se fait dans un panier contenant les produits de la commande d'un utilisateur dans une EJB Stateful "panier".

Lors de la validation du panier trois traitements sont réalisés :

- un EJB Stateful "paiement bancaire" gère le paiement en ligne par carte bancaire
- un EJB Stateless "créer commande" crée la commande en base de données via l'Entity Commande et Produit.
- si les salles de préparation sont ouvertes alors ce même EJB Stateless "créer commande" demande à un EJB Stateless "dispatcheur des produits à préparer" de dispatcher la commande vers les différentes salles de préparation.

Cet EJB Stateless "dispatcheur des produits à préparer" poussent autant de messages que de produits de la commande. En fonction du type de produit, le message est poussé dans un des canaux (1 canal par salle de préparation) du PROVIDER JMS.

Chacune des IHM de préparation s'abonnent à un des 3 canaux d'évènement en fonction de sa salle d'appartenance.

Ainsi les IHM sont prévenues par évènement des produits à préparer par le PROVIDER JMS. Un préparateur verrouille le produit qui envoie un message dans le PROVIDER JMS afin que toutes les IHM de la salle reçoivent le produit verrouillé.

Toutes les IHMs mettent à jour l'état des produits dans la base de données via les Entity Produit et Commande.

L'IHM Admin de l'administrateur fait les traitements suivants :

- interroge la base de données via les Entity Produit et Commande pour déterminer les produits des commandes qui ont été faites pendant la nuit
- utilise l'EJB Stateless "dispatcheur des produits à préparer" pour envoyer les produits à préparer aux salles de préparation.

L'IHM de livraison tire régulièrement les commandes prêtes de la base de données via l'Entity Produit et Commande.

Toutes les Entity de cette architecture utilisent un ORM pour accéder aux données de la base de données.

Question 2 : (15 Points)

```
@Entity
public class Client
{
    @Id
    @GeneratedValue
    private Integer idClient;

    @Column
    private String nomPrenom;
}

@Entity
public class Commande
{
    @Id
    @GeneratedValue
    private Integer idCommande;

    @Column
    private Integer idClient;

    @OneToMany(cascade=CascadeType.ALL,
               fetch=FetchType.EAGER, orphanRemoval=true)
    @JoinColumn(name="idCommande")
    private Set<Produit> produits;
}

@Entity
public class Produit
{
    @Id
    @GeneratedValue
    private Integer idProduit;

    @Column
    private String reference;
}
```

Fin du sujet