



## Examen Final

### Architecture et Développement Logiciels

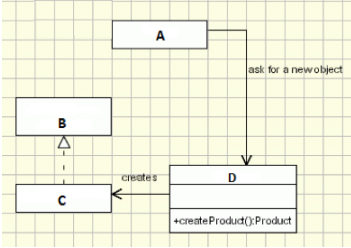
Nom	
Prénom	
Numéro Etudiant	

#### Remarques :

- Les documents ne sont pas autorisés ainsi que les appareils électroniques ( PC, Tablette, téléphone, etc).
- La première partie A est sous forme de QCM. **Attention : une réponse fausse annule une réponse juste.**
- La seconde partie B est sous forme de questions libres à répondre sur la double feuille.

#### **Partie A** : Cochez la (ou les) bonne(s) réponse(s).

Questions	Réponses
1. Un design pattern est :	<input type="checkbox"/> Un paradigme des langages de POO.
	<input type="checkbox"/> une définition des implémentations spécifiques à des principes de conceptions
	<input type="checkbox"/> une définition des principes de la POO.
	<input checked="" type="checkbox"/> aucune réponse juste.
2. Le design pattern Adapter :	<input type="checkbox"/> permet de parcourir des collections d'objet d'implémentation différentes.
	<input checked="" type="checkbox"/> est un patron de structure.
	<input checked="" type="checkbox"/> permet de s'adapter à des interfaces et des classes qui utilisent ces interfaces, sans les faire évoluer.
	<input type="checkbox"/> aucune bonne réponse.
3. En UML une interface est :	<input checked="" type="checkbox"/> une classe abstraite.
	<input type="checkbox"/> un composant graphique
	<input type="checkbox"/> une agrégation composite.
	<input type="checkbox"/> aucune bonne réponse.
4. Quelle pattern définit une interface pour la création d'un objet en déléguant à ses sous-classes le choix des classes à instanciée :	<input type="checkbox"/> Façade.
	<input checked="" type="checkbox"/> Factory.
	<input type="checkbox"/> Composite
	<input type="checkbox"/> Iterator.

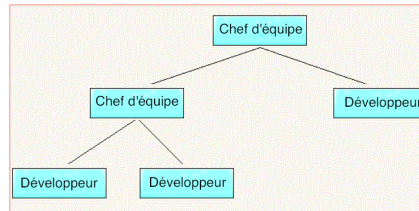
Questions	Réponses
<p>La figure suivante représente le pattern Factory :</p> 	
Questions	Réponses
<p>1. Les lettres A, B, C et D sont définies comme suite :</p>	<input type="checkbox"/> A=Client ; B=Factory ; C=Product (interface) ; D=Concrete Product .
	<input checked="" type="checkbox"/> A = Client ; B=Product (interface) ; C=Concrete Product ; D = Factory.
	<input type="checkbox"/> A=Factory ; B = Concrete Product ; C=Product (Interface) ; D=Client.
	<input type="checkbox"/> aucune bonne réponse.
<p>2. Les EJB (Entreprise Java Bean) :</p>	<input checked="" type="checkbox"/> permettent de construire des applications distribuées.
	<input checked="" type="checkbox"/> définissent un standard JavaBean pour faciliter la réutilisation et l'interopérabilité des composants middleware.
	<input checked="" type="checkbox"/> définissent l'un des modèles de composants principaux de J2EE
	<input type="checkbox"/> aucune bonne réponse.
<p>3. Un EJB entité :</p>	<input type="checkbox"/> est exécuté du côté client.
	<input checked="" type="checkbox"/> modélise une donnée persistante.
	<input type="checkbox"/> composé obligatoirement de deux interfaces local et distante.
	<input type="checkbox"/> aucune bonne réponse.
<p>4. Un EJB session Stateful :</p>	<input checked="" type="checkbox"/> maintient une conversation entre le client et le serveur.
	<input type="checkbox"/> est dédié à plusieurs clients.
	<input checked="" type="checkbox"/> représente la logique métier de l'application.
	<input type="checkbox"/> aucune bonne réponse.
<p>5. Dans les EJB qu'est ce qui est utilisé pour effectuer les opérations d'accès aux données :</p>	<input checked="" type="checkbox"/> l'entité manager.
	<input type="checkbox"/> Le conteneur.
	<input type="checkbox"/> l'entité bean.
	<input type="checkbox"/> aucune bonne réponse.

## Partie B : Questions libres.

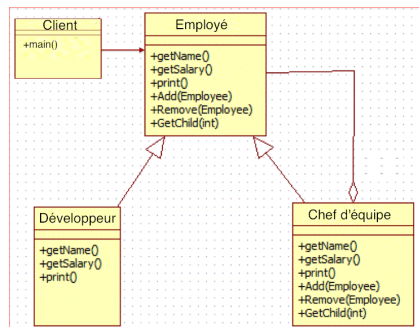
Questions	Réponses
-----------	----------

### Exercice 1 :

Dans une petite boîte d'informatique, il y a 5 employés : 2 chefs d'équipe et 3 développeurs. L'organisation de la boîte est illustrée dans la figure suivante :



On souhaite imprimer le nom et le salaire des employés de haut en bas. Voici le diagramme UML proposé :



### Questions :

- 1 Quelle est le design pattern utilisé ? Justifier.
- 2 Donner les avantages et inconvénients de ce design pattern.
- 3 Écrire en Java un exemple de code de Client qui crée la boîte d'informatique et affiche le nom et le salaire de ses employés de haut en bas.

### Réponses :

- 1 Le design pattern utilisé est composite. Il permet de créer des modèles hiérarchiques d'objet toute en masquant au client la complexité des objets qu'il manipule.
- 2 Revenir au cours.
- 3 Le code du client est le suivant :

```
public class Client {  
  
    public static void main(String[] args) {  
        Employe emp1=new Developper("Sarrah", 10000);  
        Employe emp2=new Developper("Ali", 15000);  
        Employe manager1=new ChefProjet("Meriem",25000);  
        manager1.add(emp1);  
        manager1.add(emp2);  
        Employe emp3=new Developper("Yacine", 20000);  
        Employe manager2=new ChefProjet("Mohammed", 50000);  
        manager2.add(emp3);  
        manager2.add(manager1);  
        manager2.print();  
    }  
}
```

Questions	Réponses
<p><b>Exercice 2 :</b></p> <p>Soit une application appartenant à un fournisseur de matériel informatique. L'application est composée d'une Entité stockée dans une BD. Vous y accéder à travers une Session Façade qui sera implémentée comme un composant EJB Session.</p> <p><b>Questions :</b></p> <ol style="list-style-type: none"> <li>1 Écrire la classe du bean entité, nommé Produit, elle contient : le libellé du produit et sa quantité en stock.</li> <li>2 Écrire la classe correspondant à un <b>EJB</b> session qui consiste à gérer le produit. Quel type d' <b>EJB</b> session pensez-vous utiliser ? Pourquoi ?</li> <li>3 Donner les différentes étapes nécessaires pour que cette application soit opérationnelle sur un serveur d'application ?</li> <li>4 Proposer l'implémentation d'un client Java simple.</li> <li>5 Avant d'accéder à un produit, le client devra s'authentifier. Proposer une solution élégante en utilisant la programmation orientée aspect.</li> </ol> <p><b>Réponses :</b></p>	
<pre> @Entity // Annotation indiquant qu'il s'agisse d'une Entité public class Produit implements Serializable {     private static final long serialVersionUID = 1L;     @Id     // Annotation indiquant que id est la clé de Produit     private String id;     private String libelle;     private int quantiteEnStock;      // Constructeurs     public Produit() {         super();     }      public Produit(String id) {         this.id = id;     }      public Produit(String id, String libelle, int quantiteEnStock) {         this.id = id;         this.libelle = libelle;         this.quantiteEnStock = quantiteEnStock;     }      // Méthodes Setters et getters de la classe     public String getLibelle() {         return libelle;     }      public void setLibelle(String libelle) {         this.libelle = libelle;     }      public int getQuantiteEnStock() {         return quantiteEnStock;     }      public void setQuantiteEnStock(int quantiteEnStock) {         this.quantiteEnStock = quantiteEnStock;     }      public String getId() {         return id;     } } </pre>	<pre> @Stateless public class GestionProduitBean implements GestionProduit {     @PersistenceContext     //(name = "sample")     private EntityManager em; // L'Entity Manager      public void ajouter(Produit produit) {         em.persist(produit);     }      public Produit rechercherProduit(String id) {         return em.find(Produit.class, id);     }      public List&lt;Produit&gt; listerTousLesProduits() {         return em.createQuery(             "SELECT p FROM Produit p ORDER BY             p.quantiteEnStock")             .getResultList();     } } </pre> <pre> public class MonClientGestionProduits {     public static void main(String[] args) {         try {             int i=0;             Context context = new InitialContext();             GestionProduit stock = (GestionProduit)             context.lookup("GestionProduitBean/remote");              // Ne pas faire l'ajout plusieurs fois, commenter             // après la première exécution.             stock.ajouter(new Produit("001", "ecran plat", 100));             stock.ajouter(new Produit("002", "imprimantes laser",             5680));             stock.ajouter(new Produit("003", "souris optique", 23));             stock.ajouter(new Produit("004", "pc portable", 115));             stock.ajouter(new Produit("005", "unité centrale", 48));              List&lt;Produit&gt; produits =             stock.listerTousLesProduits();             for (Iterator&lt;Produit&gt; iter = produits.iterator();             iter.hasNext();) {                 Produit eachProduit = (Produit) iter.next();                 System.out.println(eachProduit); // Invocation             }         } catch (javax.naming.NamingException ne) {             System.out.println("lookup error");             ne.printStackTrace();         }     } } </pre>
<p>3 Pour que cette application soit opérationnelle sur un serveur d'application, il faut :</p> <ul style="list-style-type: none"> <li>– Associer un serveur, exemple Jboss, au projet et cela à sa création et lui spécifier le chemin vers ce serveur.</li> <li>– Déployer le projet fini sur le serveur.</li> </ul>	

Bon courage et bonne continuation.