IPST-CNAM
Intranet et Designs patterns
NSY 102
Mercredi 4 Avril 2018

Durée : **2 h 45**
Enseignants : LAFORGUE Jacques

1ère Session NSY 102

1ère 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
  - –½ pour chaque réponse fausse
- **les questions à 3 propositions** dont 1 seule proposition est fausse
  - + ½ 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:

La "Démarche d'Architecture" est un guide méthodologique pour créer le dossier d'architecture d'un Système d'Information.		Q 1.
1	OUI	
2	NON	

A l'issu de la Configuration Architecturale, on obtient la description d'un "réseau" de description du Système d'Information dont les nœuds sont les Composants et les liens les Connecteurs.		Q 2.
1	OUI	
2	NON	

Les rôles fonctionnels d'un composant sont toujours des fonctionnalités décrites dans les exigences du besoin initial.		Q 3.
1	OUI	
2	NON	

Une application dite "distribuée" est une application logicielle dans lequel les données informatiques sont toutes centralisées dans un composant unique appelé Singleton		Q 4.
1	OUI	
2	NON	

En RMI de Java, la classe d'appartenance d'un objet distribué		Q 5.
1	hérite de UnicastRemoteObject et implémente une interface qui décrit les méthodes distantes et qui hérite de l'interface prédéfinie Remote.	
2	hérite de RemoteObject et implémente l'interface Remote	
3	n'hérite d'aucune classe particulière et doit être envoyé, par sérialisation; à l'annuaire pour y être enregistré.	

Un DP définit des principes de conception, et non des implémentations spécifiques de ces principes		Q 6.
1	OUI	
2	NON	

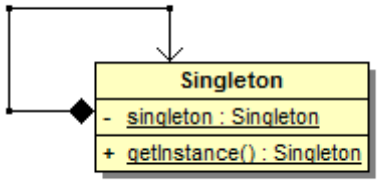
Un Design Pattern (DP) ou Patron de Conception est une norme de description des interfaces entre les composants d'une architecture logicielle orientée objet.		Q 7.
1	OUI	
2	NON	

Le rôle du Design Pattern **Singleton** est la création unique d'une instance d'une classe.		Q 8.
1	OUI	
2	NON	

Dans un système réparti, le DP Singleton est utilisé pour créer un objet distant unique sur le réseau.		Q 9.
1	OUI	
2	NON	

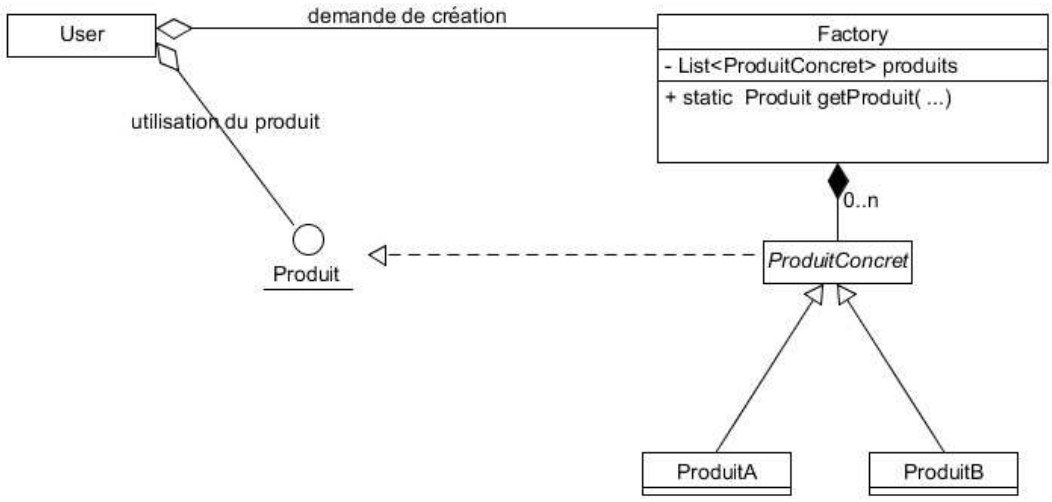
Le rôle du DP Factory qui est un Singleton est de rendre global à tout le programme, la création et l'utilisation de certains objets.		Q 10.
1	OUI	
2	NON	

Un DP Singleton définit une classe particulière dont le rôle est de créer l'objet unique lors de sa première utilisation.		Q 11.
1	OUI	
2	NON	

Le schéma suivant :  est celui du DesignPattern **Singleton**		Q 12.
1	OUI	
2	NON	

Dans une application distribuée, un **factory** est un objet distant, enregistré dans un annuaire, et dont le rôle est de créer des objets distants qui sont utilisés par le client à travers un proxy client (appelé "stub")		Q 13.
1	OUI	
2	NON	

Le rôle de la classe abstraite, dans un Factory, est de servir de proxy entre les classes concrètes d'implémentation des produits du factory.		Q 14.
1	OUI	
2	NON	

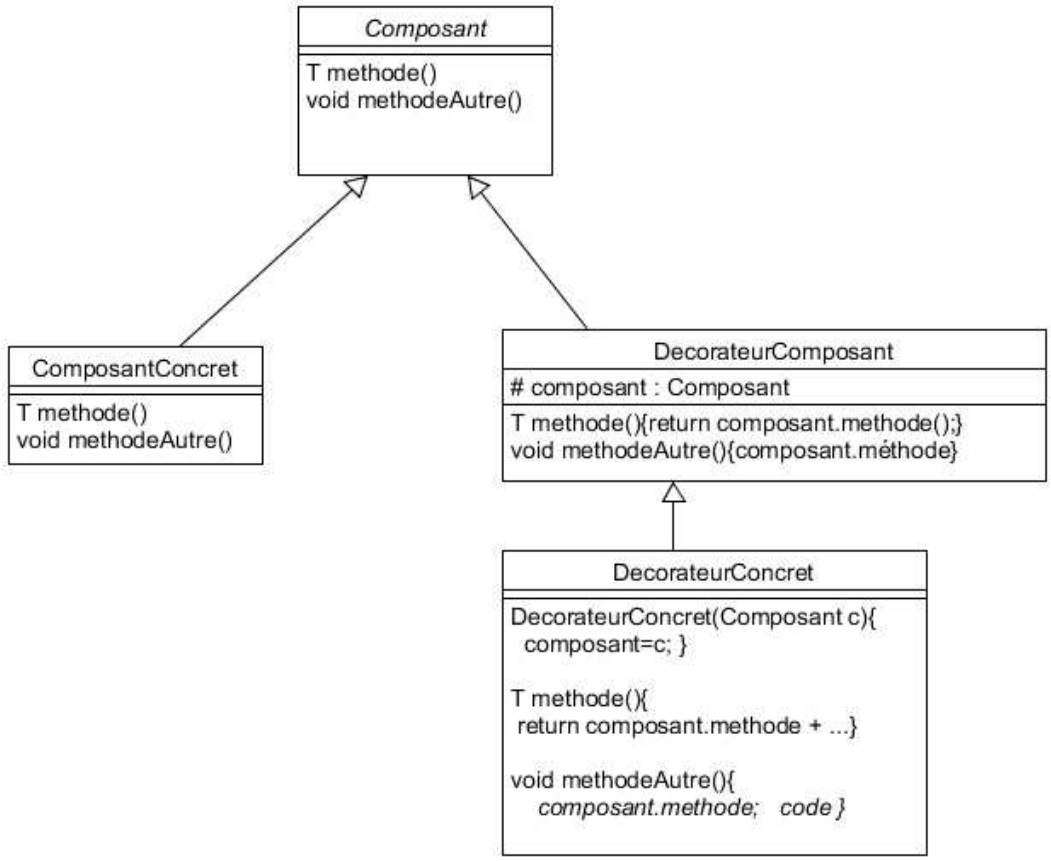
 Ce DP est celui du Factory. Le rôle de l'interface **Produit** est d'abstraire, à la classe User, la classe ProduitA ou ProduitB utilisé pour créer l'objet		Q 15.
1	OUI	
1	NON	

Dans un DP Factory, le Client doit connaître les différents types de produits créés par le factory		Q 16.
1	OUI	
2	NON	

Le DP Builder est un DP utilisé dans celui du Factory afin de construire le produit par assemblage d'autres classes.		Q 17.
1	OUI	
2	NON	

Le DP "Délégation" est un Design Pattern qui appartient à la famille des Proxys.		Q 18.
1	OUI	
2	NON	

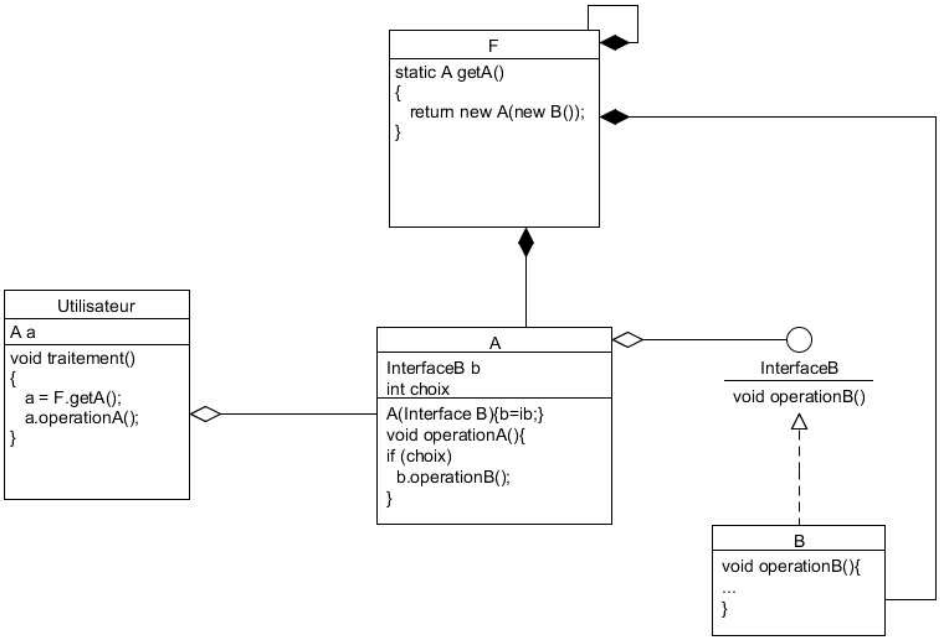
Q 19.



Ce schéma est celui du DP Décorateur.
Ce schéma est correct.

1	OUI	
2	NON	

Le diagramme suivant :		Q 20.



représente une injection de dépendance par l'utilisation d'un proxy		
1	OUI	
2	NON	

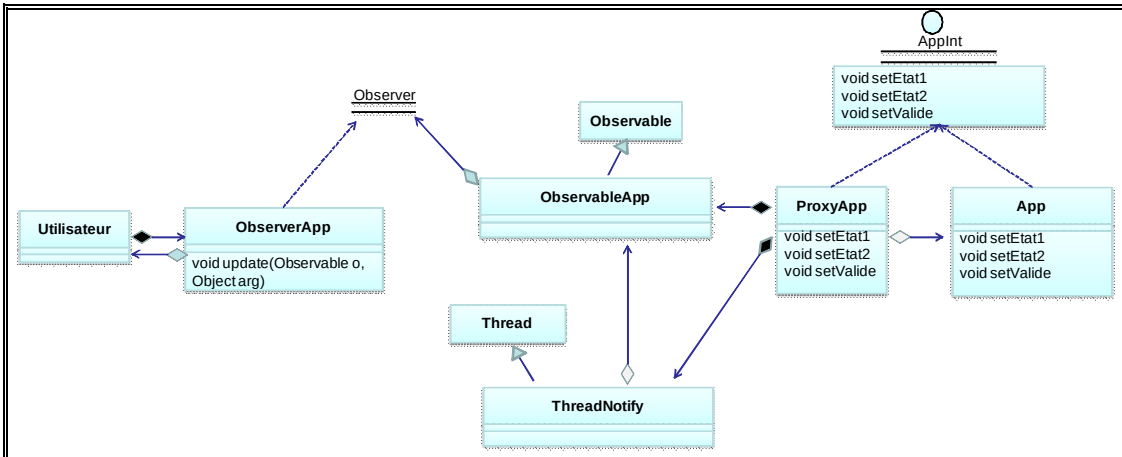
Le rôle du Design Pattern **Observateur** est de créer, dynamiquement des objets dont la classe d'appartenance implémente l'interface **Observer**		Q 21.
1	OUI	
2	NON	

Le DP Observateur contient		Q 22.
1	une classe qui hérite de la classe Observer	
2	une classe qui hérite de la classe Observable	
3	une classe Factory qui crée les évènements à notifier	

Toute classe qui implémente l'interface Observer peut devenir un observateur (Observer) de n'importe quel observé (Observable)		Q 23.
1	OUI	
2	NON	

Le Design Pattern Observateur est utilisé dans :		Q 24.
1	le Design Pattern Factory	
2	le Design Pattern MVC (Model-Vue-Controleur)	

Le DP Observeur/Observable, peut être utilisé, dans une architecture MOM, pour réaliser		Q 25.
1	le connecteur entre le Provider et les Consommateurs	
2	le connecteur entre le Producteur et le Provider	



Q 26.

Ce schéma de Design Pattern est :

1	Le DP Observer/Observable de type pull	
2	Le DP Observer/Observable de type push synchrone	
3	Le DP Observer/Observable de type push asynchrone	

Le ClientProxy est un DP Proxy utilisé par un client, dans lequel les méthodes réelles ont été remplacées par un appel distant à ces méthodes.		Q 27.
1	OUI	
2	NON	

On peut utiliser le Design Pattern Proxy pour rendre distant :		Q 28.
1	une classe quelconque	
2	une classe qui implémente l'interface du Proxy	

En RMI, le "stub" est un Proxy sur l'interface qui hérite de Remote		Q 29.
1	OUI	
2	NON	

A l'opposé de la communication synchrone, la communication asynchrone est un type de communication notamment basé sur le modèle du pull, comme par exemple un thread d'un client qui tire régulièrement les événements d'un serveur.		Q 30.
1	OUI	
2	NON	

Dans la communication synchrone via un "canal d'évènement" entre un producteur et un consommateur, le producteur utilise un proxy de consommateur (et non les consommateurs directement), afin de lui pousser un évènement.		Q 31.
1	OUI	
2	NON	

Laquelle des descriptions suivantes est un principe de communication synchrone :		Q 32.
1	le producteur dépose à son rythme ses évènements dans une file. Le ou les consommateurs peuvent alors récupérer ces évènements	
2	le producteur pousse ("push") chaque évènement vers chacun des consommateurs via une méthode distante qui retourne un état de consommation	

Le principe général d'un MOM (Model Orienté Message) est que tous les composants connectés au bus du MOM reçoivent tous les évènements publiés dans le fournisseur de service MOM (Provider) quels que soit les canaux d'évènement utilisés.		Q 33.
1	OUI	
2	NON	

Dans une architecture MOM, le mode "Queue" assure que tous les consommateurs connectés au canal d'évènement de la queue d'évènement, reçoivent l'évènement publié par le Producteur.		Q 34.
1	OUI	
2	NON	

Le Design:Pattern **DynamicProxy** est composé d'une classe qui implémente l'interface du proxy. Cette classe est :		Q 35.
1	une classe que l'on crée et qui implémente aussi l'interface InvocationHandler	
2	une classe créée dynamiquement par le framework ou le langage utilisé (ex Java)	

Fin du QCM

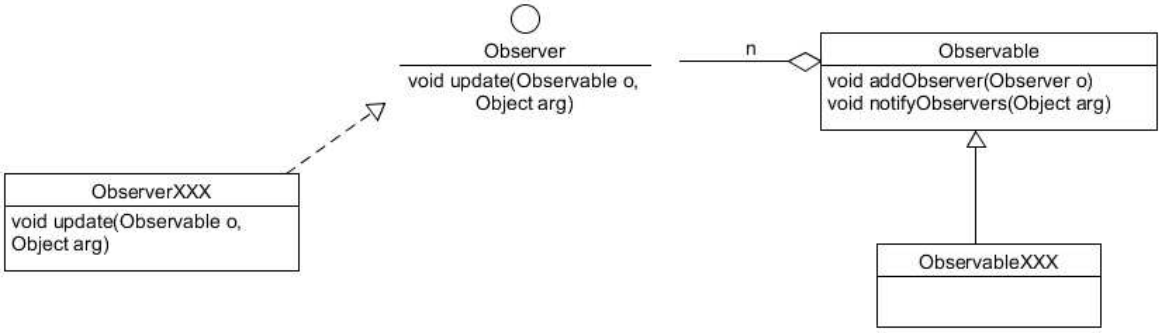
Suite (Tournez la page)

2. Questions libres (15 points)

Chaque question est notée sur 5 points.
*Vous répondez à ces questions sur une **copie vierge double** en mettant bien le numéro de la question, sans oublier votre nom et prénom.*
Vous mettez le QCM dans la copie vierge double.

QUESTION NUMERO 1

Soit le Design Pattern Observateur, local à une JVM, suivant :



Complétez ce schéma afin que les Observer et l'Observable soient distants (vous ne modifiez pas les classes et interfaces existantes).
Commentez.

QUESTION NUMERO 2

Expliquez le principe du DynamicProxy.

QUESTION NUMERO 3

Soit un Factory implémenté dans un serveur qui crée des produits à la demande d'un client. Ces produits restent locaux au serveur et sont utilisés de manière distante par le client qui les a créés.
Faites le diagramme de classe de ce Factory.

Fin de la 1ère partie sans document

2ème PARTIE – AVEC DOCUMENT (durée: 1h30)

3. PROBLEME (50 points)

Nous voulons réaliser un Système d'Information (SI) permettant de gérer des chambres dans un hôtel high-tech.
Le poste d'IHM présent à l'accueil permet :

- d'informer qu'un service a été demandé par une chambre :
  - activation de certaines chaines payantes (réalisé par le SI)
  - commande du petit déjeuner (un robot-chariot automatique réalise le service de la cuisine à la chambre dont la coiffe du chariot est verrouillée par un code)
  - service de réveil automatique par téléphone (réalisé par le SI)
  - service de nettoyage (réalisé par un humain) :
- de gérer les réservations des chambres de l'hôtel.

Chaque chambre contient une tablette (Android/Java via Wifi en RMI) permettant de demander un service.
Tous les services sont payants (sauf le réveil) et sont facturés (mise à jour de la note de la chambre).
Le SI notifie les tablettes répartis dans l'hôtel pour le suivi de la demande d'un service, et pour mettre à jour l'écran d'accueil de toutes les tablettes avec des messages simples d'offres publicitaires.

Le SI est composé de 5 composants :

- le poste d'IHM se trouvant à l'accueil [COMPOSANT 1]
- le serveur qui centralise toutes les données du SI (services, chambres) et envoie les commandes aux robots [COMPOSANT 2]
- les tablettes de service réparties dans chacune des chambres [COMPOSANT 3]
- les robots [COMPOSANT 4]
- un poste d'IHM se trouvant en cuisine [COMPOSANT 5] pour préparer les petits déjeuners, charger le robot et le démarrer.

Les robots peuvent être de technologies diverses. Il doit être possible d'intégrer de nouveaux modèles de robot dans le système, le plus facilement possible.

1/ Faites le schéma d'architecture logicielle de ce Système d'Information.
Commentez votre schéma (fonctionnement, rôle, fonctions).

Un composant logiciel [COMPOSANT X] correspond à une JVM.

2/ Faire le diagramme de classe UML des composants : [COMPOSANT 1], [COMPOSANT 2], et [COMPOSANT 3] en mettant en évidence certains des Designs Patterns vus en cours.

Pour une description précise de vos diagrammes de classe, on fait le choix que toutes les communications distantes entre les composants sont réalisées en RMI.

Fin du sujet