IPST-CNAM
Intranet et Designs patterns
NSY 102
Jeudi 16 Mai 2024

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:

Dans la démarche d'architecture d'un Système d'Information, l'**Architecture Technique** est l'implémentation de la **Configuration Architecturale** dans une ou plusieurs technologies données.		Q 1.
1	OUI	
2	NON	

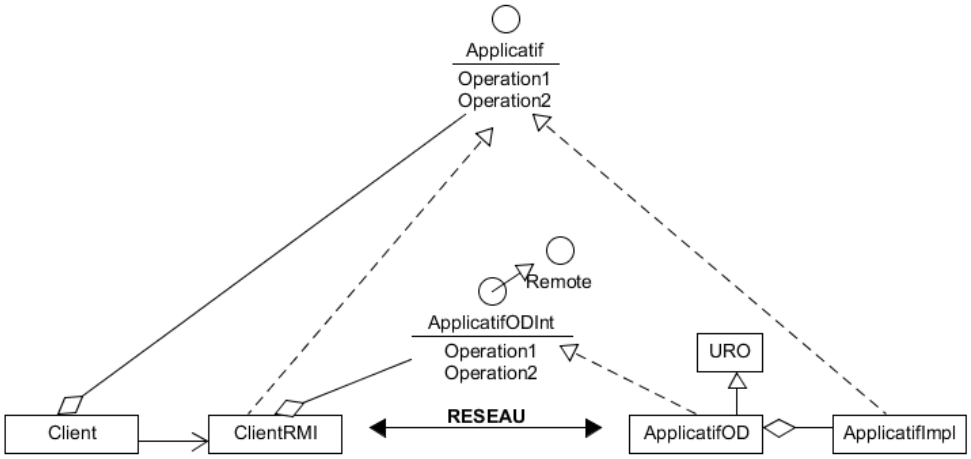
La **Configuration Architecturale** est la décomposition d'un Système d'Information en composants logiciels et sous-composants.		Q 2.
1	OUI	
2	NON	

Dans la description de l'architecture technique, un connecteur est un lien de dépendance entre deux composants qui peut être réalisé par le principe du design pattern de l'injection de dépendance.		Q 3.
1	OUI	
2	NON	

Soit un objet quelconque Obj (instance de la classe A qui n'hérite pas d'une autre classe). En Java RMI, il est très facile de transformer cet objet en un objet distant. Pour cela il suffit de :		Q 4.
1	faire que la classe A implémente l'interface Remote	
2	faire que la classe A implémente l'interface Serializable, puis écrire cet objet dans un annuaire RMI	
3	créer un proxy de A . Ce proxy hérite de UnicastRemoteObject et implémente l'interface de A qui hérite de Remote	

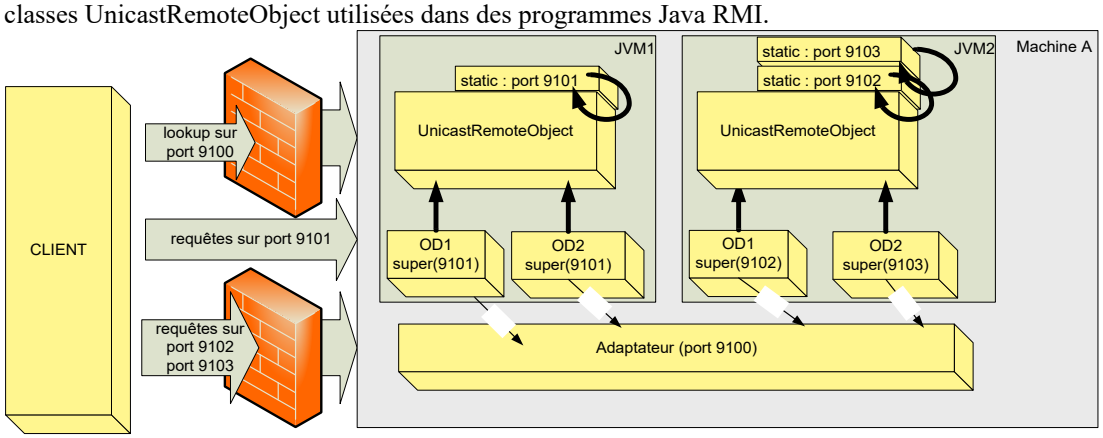
En RMI de Java, la classe d'appartenance d'un objet distant, hérite de RemoteObject et implémente l'interface Remote		Q 5.
1	OUI	
2	NON	

Ceci est le diagramme de classe d'un système composé d'un client et de son applicatif que l'on veut rendre distant.		Q 6.



1	Client est un proxy de ApplicatifImpl	
2	ApplicatifOD est un Adaptateur de ApplicatifImpl à ApplicatifODInt	
3	ClientRmi est un proxy de ApplicatifImpl	

Soit le schéma suivant qui représente un fonctionnement possible de plusieurs serveurs de socket des classes UnicastRemoteObject utilisées dans des programmes Java RMI.		Q 7.



1	On peut créer un nouvel OD dans la JVM1 qui s'exécute sur le port 9102	
2	Sur la machine A, on peut créer une nouvelle JVM3 dans laquelle, on crée un nouvel OD qui s'exécute sur le port 9103	
3	Dans la JVM2, on peut créée un nouvel objet distribué RMI sur le port 9102	

En Java, la méthode **public static Remote toStub(Remote obj)** permet :		Q 8.
1	d'enregistrer l'objet distant 'obj' dans l'annuaire RMI	
2	de retourner un proxy de communication qui implémente l'interface de l'objet distant obj	

Soit un objet, instance de la classe A. Pour transformer cet objet en un objet distant, il suffit que :		Q 9.
1	A hérite de la classe UnicastRemoteObject et implémente une interface publique (I) qui hérite de Remote	
2	A soit un proxy de la classe UnicastRemoteObject	
3	A soit une agrégation d'une classe B qui hérite de la classe UnicastRemoteObject et implémente une interface publique (I) qui hérite de Remote	

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 10.
1	OUI	
2	NON	

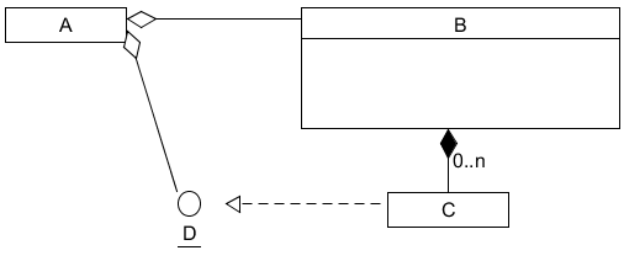
Dans un système réparti, le DP Singleton est utilisé pour créer un objet distribué unique sur le réseau		Q 11.
1	OUI	
2	NON	

Il existe deux façons de créer un singleton de classe Singleton : - soit d'instancier le singleton dans la définition de la classe - soit d'instancier le singleton dans l'appel de la méthode *Singleton getInstance()*		Q 12.
1	OUI	
2	NON	

Il est toujours possible de créer des setteurs sur un singleton permettant de modifier des attributs du singleton.		Q 13.
1	OUI	
2	NON	

Le DP Factory a pour fonction la création d'objet dont les classes héritent d'une même classe abstraite et/ou implémentent la même interface		Q 14.
1	OUI	
2	NON	

Le rôle du DP Factory est de créer des objets qui sont vus par le reste du programme comme des singletons :		Q 15.
1	OUI	
2	NON	

Ce DP est celui du Factory.  La signification des lettres A, B, C et D est :		Q 16.
1	A=Factory; B = Concrete Product; C=Product (Interface); D=Client	
2	A=Client; B=Factory; C=Concrete Product; D=Product (interface);	
3	A = Client; B=Product (interface); C=Concrete Product; D = Factory	

Le DP Factory mémorise toujours les produits créés dans une collection.		Q 17.
1	OUI	
2	NON	

Le DP "Délégation" est utiliser dans le DP "injection de dépendance"		Q 18.
1	OUI	
2	NON	

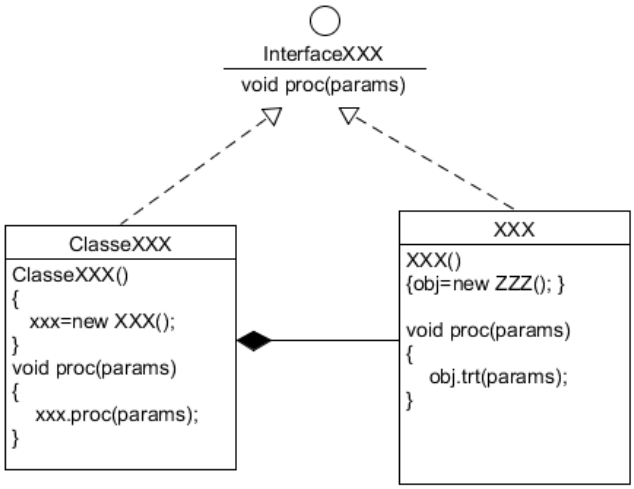
Si la classe A est un décorateur de la classe B alors les classes A et B héritent toutes deux d'une même classe abstraite.		Q 19.
1	OUI	
2	NON	

Dans le DP Observateur, la communication entre l'Observer (consommateur d'évènement) et l'Observable (producteur d'évènement) est nécessairement asynchrone car la communication se fait toujours par l'envoi d'un objet sans valeur de retour.		Q 20.
1	OUI	
2	NON	

Dans le DP Observateur, la communication entre l'objet observé (producteur d'évènement) et l'observateur (consommateur d'évènement) est :		Q 21.
1	synchrone ou asynchrone (choix de conception)	
2	toujours asynchrone	
3	toujours synchrone	

Le DP Observateur , peut être utilisé, dans une architecture MOM, pour réaliser		Q 22.
1	le connecteur entre le Provider et les Consommateurs	
2	le connecteur entre le Producteur et le Provider	

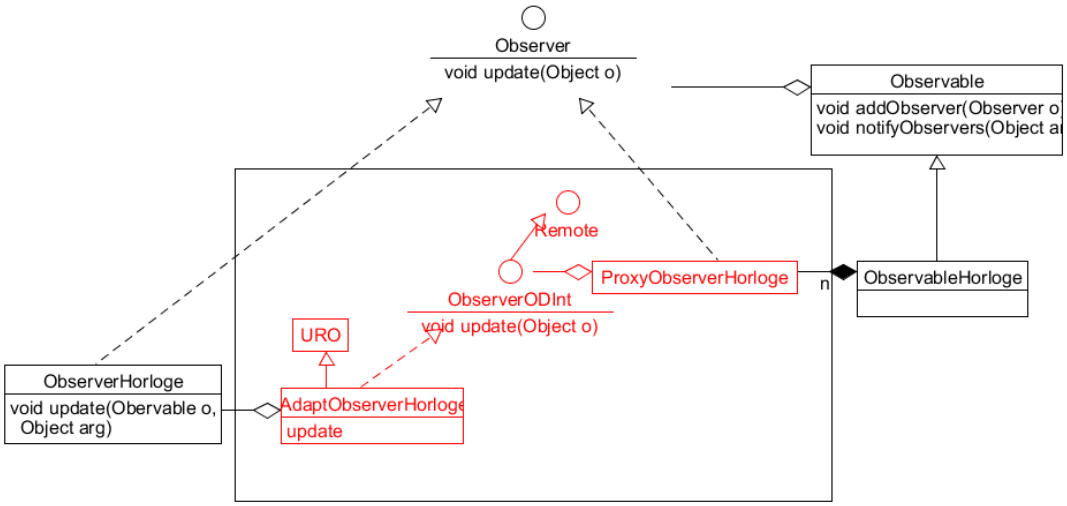
Dans le DP Adaptateur, l'adaptateur et l'adapté implémente la même interface		Q 23.
1	OUI	
2	NON	

Soit le diagramme de classe suivant :		Q 24.
		
1	Ce diagramme de classe représente le DP Adaptateur	
2	Ce diagramme de classe représente le DP Proxy	

Dans la communication synchrone, en mode push, via un "canal d'évènement", entre un producteur et des consommateurs, le producteur utilise un proxy de consommateur (et non les consommateurs directement), afin de leurs pousser un évènement.		Q 25.
1	OUI	
2	NON	

Un canal d'évènement est constitué de deux DP : un DP Factory permettant de créer des évènements et un DP Iterator permettant de parcourir ces évènements.		Q 26.
1	OUI	
2	NON	

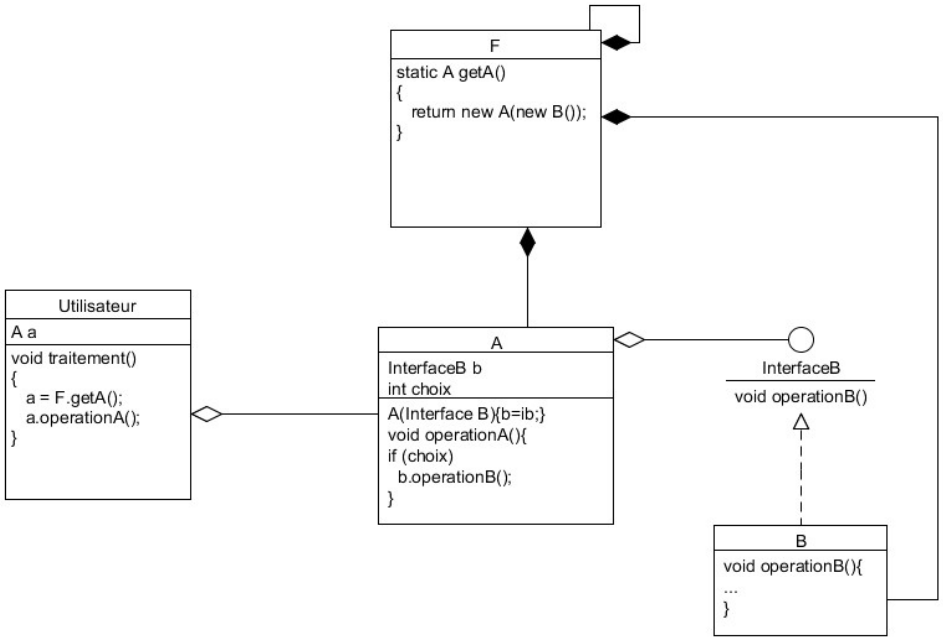
Soit le schéma suivant : Q 27.



Les classes et interfaces en rouge encadrées représentent :

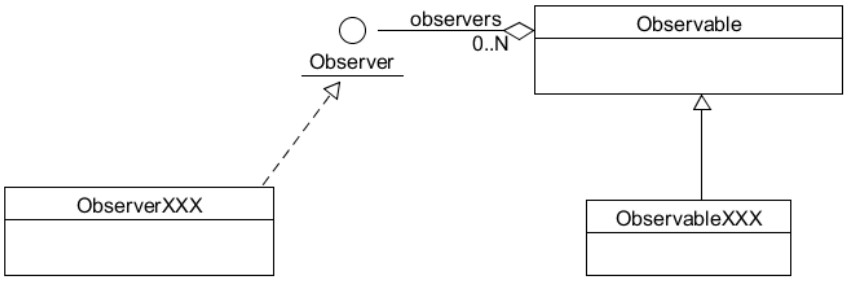
1	un proxy client de communication de ObserverHorloge vers ObservableHorloge	
2	un pont de communication permettant à un Observable (ObservableHorloge) de notifier les évènements à un Observer (ObserverHorloge) se trouvant dans une autre JVM.	

Le diagramme suivant : Q 28.



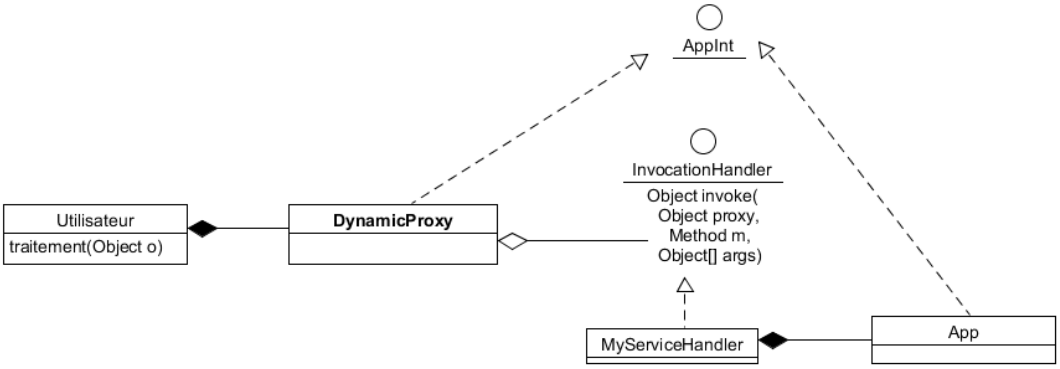
représente

1	une injection de dépendance par l'utilisation d'un setteur	
2	une injection de dépendance par l'utilisation d'un constructeur	
3	une injection de dépendance par l'utilisation d'un proxy	

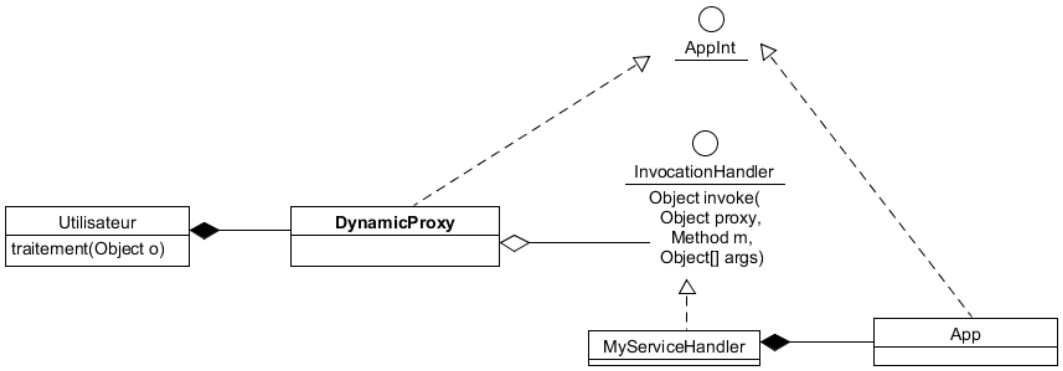
Soit le Design Pattern Observateur décrit (volontairement simplifié) de la manière suivante :		Q 29.
		
1	La classe ObservableXXX notifie les évènements à une instance de Observable	
2	La classe ObserverXXX implémente la méthode update de l'interface Observer qui est appelée par Observable	
3	La classe Observable pousse (modèle du "push") les évènements à ObserverXXX	

Laquelle des descriptions suivantes est un principe de communication synchrone :		Q 30.
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, à un même canal d'évènement du bus du MOM, en mode Topic, reçoivent tous les évènements publiés dans le fournisseur de service MOM (Provider)		Q 31.
1	OUI	
2	NON	

		Q 32.
	 Ce schéma est celui du DP Dynamic proxy. Le rôle de la classe MyServiceHandler est d'implémenter toutes les méthodes de l'interface AppInt.	
1	OUI	
2	NON	

Q 33.

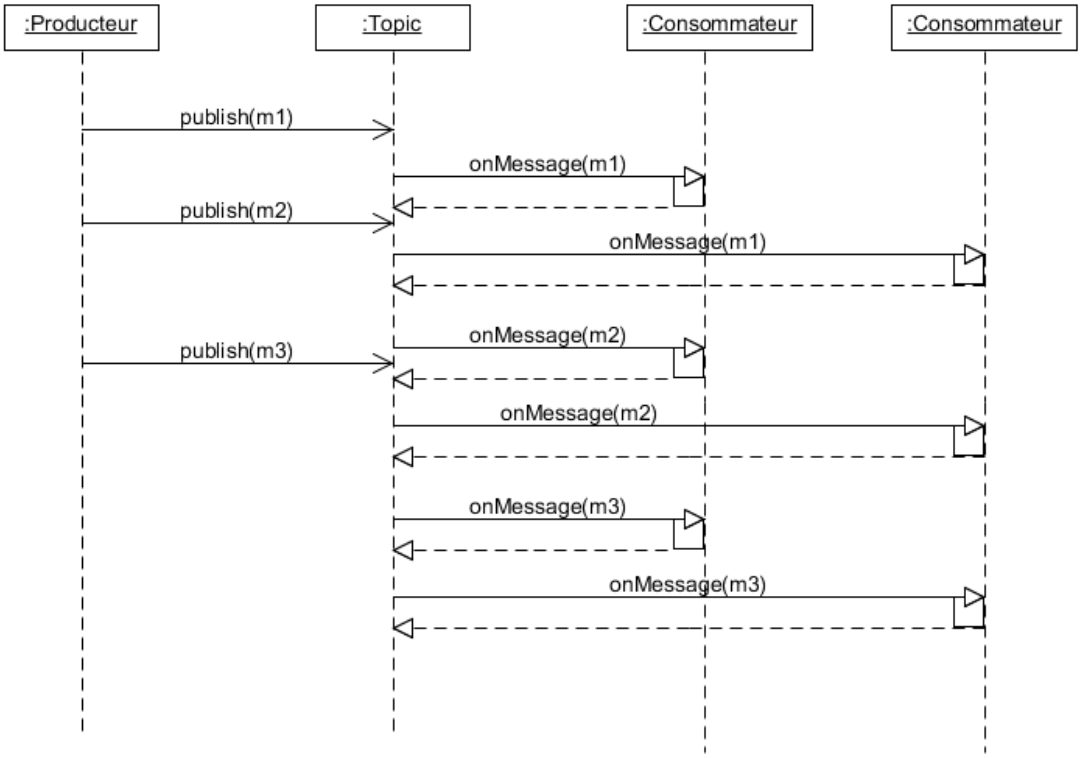


L'API RMI de Java utilise le DP DynamicProxy pour :

1	créer par l'Injection de Dépendance, la dépendance entre la classe MyServiceHandler et App.	
2	créer le proxy client dans la méthode loookup, permettant la communication avec l'objet.	

Q 34.

En JMS (Java Messaging System), il existe (notamment) deux modes de communication : Queue et Topic.



Ce diagramme de transition correspond au mode :

1	Queue	
2	Topic	

Q 35.

En JMS (Java Messaging System), les producteurs et les consommateurs sont tous des clients d'un composant logiciel appelé JMS Provider

1	OUI	
2	NON	

Fin du QCM

Suite (Tournez la page)

2. Questions libres (15 points)

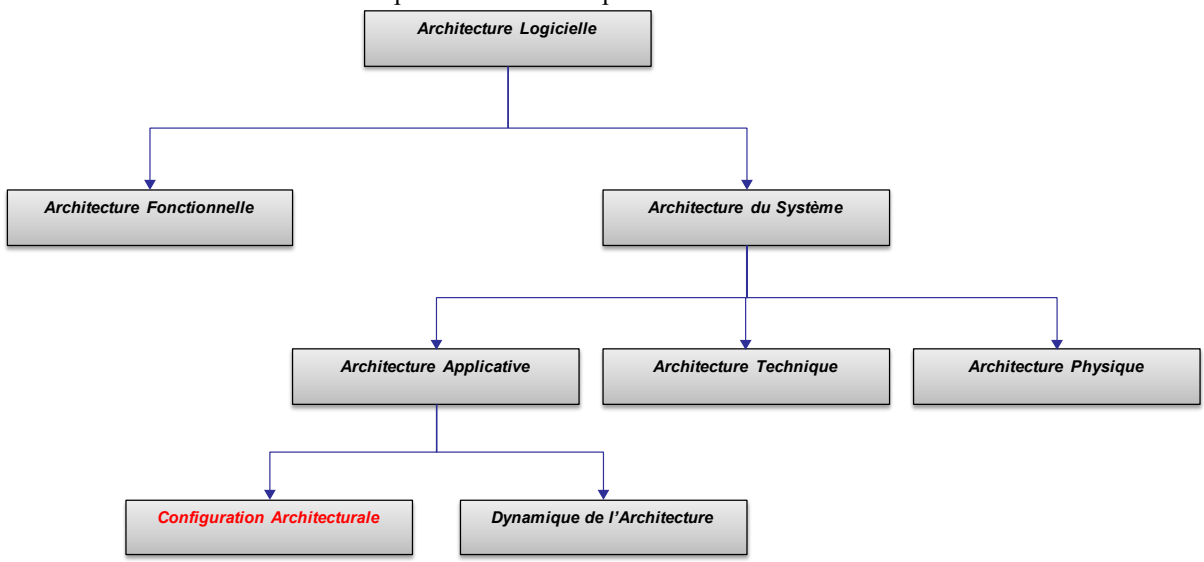
Chaque question est notée sur 5 points.
*Vous répondez à ces questions sur une **copie vierge double** en mettant bien le numéro de la question, sans oublier votre nom et prénom.*
Vous mettez le QCM dans la copie vierge double.

QUESTION NUMERO 1

Le diagramme de **Communication** UML permet de mettre en évidence des aspects importants dans la conception d'un Système d'Information. Quels sont ces aspects ? (En citer au moins 5)

QUESTION NUMERO 2

La démarche d'architecture se décompose en différentes parties :



Quel est le rôle de :
- la Configuration Architecturale
- la Dynamique de l'Architecture
- l' Architecture Technique
- l'Architecture Physique
- l'Architecture Fonctionnelle

(Pas plus de une à deux phrases pour chacun)

QUESTION NUMERO 3

Quel est le rôle du Design Pattern de l'Injection de Dépendance ?
Décrire son fonctionnement (vous pouvez vous aider d'un diagramme mais cela n'est pas obligatoire).

Fin de la 1ère partie sans document

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

3. PROBLEME [50 points]

On se propose de réaliser la conception d'un Système d'Information qui permet à un organisme bancaire de valider des ordres de transactions boursières réalisés par leurs clients.

Nous ne traitons pas ici la façon dont les clients créent leurs ordres de transactions boursières. Les ordres de transaction boursières proviennent de l'extérieur et arrivent sur un serveur de la banque [COMPOSANT 1].
Le serveur gère, en mémoire, le stocke de ces ordres sous forme d'objets. Il crée en mémoire autant d'objet qu'il existe d'ordre. Il existe plusieurs types d'ordre.

Les postes d'IHM [COMPOSANT 2] sont sur le réseau intranet de la banque et communiquent en RMI avec le serveur.
Le rôle de ces postes est de visualiser des ordres en fonction de certains critères (intervalle de date, nombre, type, montant, donneur d'ordre, ...) puis de valider manuellement ces ordres.
La validation consiste à accepter ou refuser l'ordre. Dans le cas d'un refus, l'opérateur renseigne le motif du refus.
Toutes ces informations (acceptation, refus, motif) sont renseignées sur le serveur dans l'objet de l'ordre correspondant.
C'est l'opérateur qui demande explicitement de s'occuper de certains ordres en fonction de critères.
La liste de ces ordres est alors affichée et l'opérateur verrouille sur le serveur autant d'ordre qu'il souhaite (s'ils ne sont pas déjà verrouillés par un autre opérateur). Il a alors tout le temps pour analyser les ordres verrouillés par lui.
Pour faire cette analyse, l'opérateur peut consulter toutes les informations d'un client qui sont gérées par chaque Agence [COMPOSANT 4] bancaire répartis sur le territoire français.

Les ordres sont déverrouillés par l'opérateur (explicitement ou automatiquement lors de l'acceptation ou du refus).

Chaque liste des ordres de chaque IHM est rafraîchie si un des ordres affichés est modifié par ailleurs.

Pour des raisons de sécurité, il existe un deuxième serveur, appelé "serveur redondant" [COMPOSANT 3] qui est une duplication du serveur principal. A chaque modification d'un ordre de transaction boursière par le serveur principal, automatiquement, la même modification est réalisée sur le serveur secondaire.
Cela permet en cas de panne du serveur principal, de basculer, à chaud, sur le serveur secondaire, les IHM se reconnectant alors automatiquement sur le serveur secondaire.

1/ **[15 points]**
Faites le diagramme de communication de ce Système d'Information. Il y a donc 4 composants à décrire qui communiquent entre eux de manière distant.
Commentez votre schéma (rôles des composants et sous-composants, comportement dynamique général, échanges des informations, localisation des données).
Nous rappelons que ce schéma doit permettre de connaître vos choix d'organisation des composants, le sens des communications, et les designs patterns envisagés.

2/ **[35 points]**
Faites le(s) diagramme(s)e de classe UML des [COMPOSANT 1] et [COMPOSANT 2] et [COMPOSANT 3] en mettant en évidence les Designs Patterns utilisés.
Commentez chacun de(s) diagramme(s).

Précisions :
Un composant applicatif [COMPOSANT X] correspond à une JVM ou process. Cela signifie que les COMPOSANTS X communiquent sur le réseau à travers des interfaces distantes.
Ainsi, 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 (utilisation de la classe URO = UnicastRemoteObject et de l'interface Remote).

Fin du sujet