IPST-CNAM
Intranet et Designs patterns
NSY 102
Vendredi 7 Avril 2017

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:

Réaliser l'architecture d'un Système d'Information (SI) permet de valider les choix technique pour la réalisation du SI.		Q 1.
1	OUI	
2	NON	

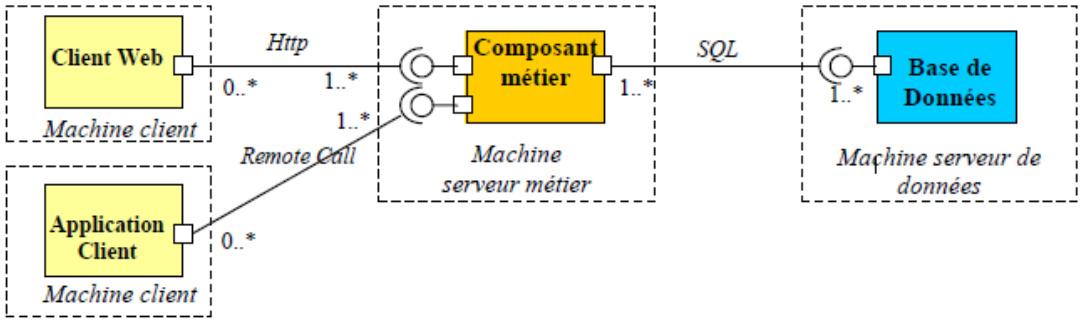
La **Configuration Architecturale** d'un Système d'Information		Q 2.
1	est la description de tous les composants du SI qui sont utilisés pour configurer l'exécution de ce dernier	
2	représente l'organisation des composants logiciels et des sous-composants qui constituent un Système d'Information	

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	

Dans une Configuration Architecturale, un lien entre deux composants correspond :		Q 4.
1	toujours une dépendance distante (machine à machine) entre les composants	
2	souvent à la transmission d'information entre les composants	

Dans la réalisation de l'architecture d'un futur SI, il n'est pas rare de réaliser une maquette ou un prototype qui représente au mieux l'ensemble du futur système permettant de s'assurer des choix réalisés tant sur la capacité que sur les performances.		Q 5.
1	OUI	
2	NON	

Q 6.



Ce schéma représente une architecture 4-Tiers

1	OUI	
2	NON	

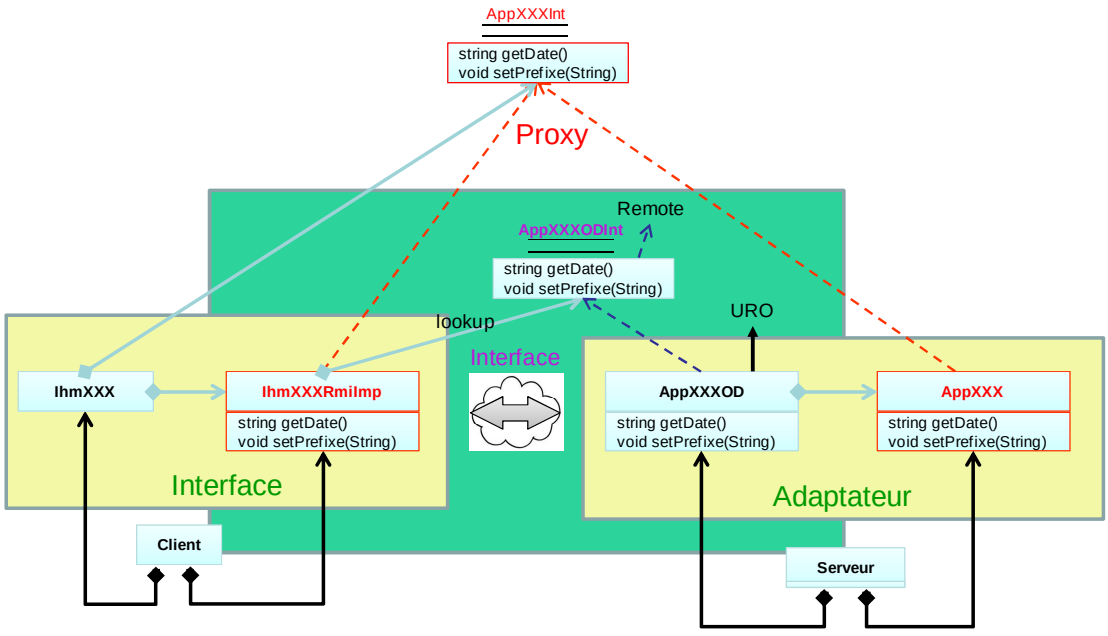
Une application dite "distribuée" est une application logicielle dans lequel les données informatiques sont		Q 7.
1	nécessairement centralisées dans un singleton (exemple un Factory)	
2	réparties sur le réseau et accessibles par tout logiciel qui utiliserait un ORB	

L'IDL (Interface Definition Language) permet de créer les souches et les squelettes dans différents langages informatique assurant ainsi l'interopérabilité des services entre eux		Q 8.
1	OUI	
2	NON	

Il existe deux façons pour utiliser les méthodes distantes d'un objet distribué : 1/ demander le stub de connexion à un annuaire, 2/ demander le stub de connexion à celui qui a créé l'objet distribué (exemple un factory); puis d'utiliser ce stub pour appeler les méthodes distantes		Q 9.
1	OUI	
2	NON	

Soit 2 clients (A et B) qui appellent en même temps la méthode distante m1 de l'objet distribué OD1.		Q 10.
1	A et B peuvent appeler en même temps la méthode m1 de OD1	
2	A et B ne peuvent pas appeler en même temps la méthode m1 de OD1 et il faut donc que la méthode m1 soit "synchronized".	

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 distribué. Pour cela il suffit de :		Q 11.
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	

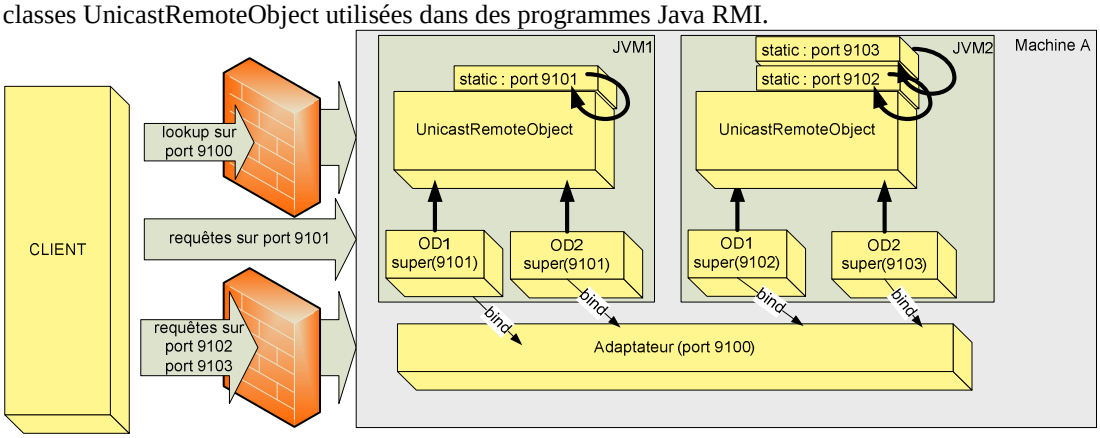


Q 12.

Ceci est un diagramme de classe d'un système composé d'un client IHM (classe IhmXXX) et de son applicatif (AppXXX) que l'on veut rendre distant.

IhmXXXRmiImp est un Proxy de AppXXX :

1	OUI	
2	NON	

Soit le schéma suivant qui représente un fonctionnement possible de plusieurs serveurs de socket des classes UnicastRemoteObject utilisées dans des programmes Java RMI.		Q 13.
		
1	On peut créer un nouvel OD dans la JVM1 qui s'exécute sur le port 9101	
2	On peut créer un nouvel OD dans la JVM1 qui s'exécute sur le port 9102	
3	On peut créer un nouvel OD dans la JVM2 qui s'exécute sur le port 9100	

En RMI, l'appel d'une méthode distante, entre un client et un objet distribué RMI se fait en utilisant un objet qui est un proxy de communication de l'objet distribué. Ce proxy implémente l'interface distante que l'objet distribué implémente.		Q 14.
1	OUI	
2	NON	

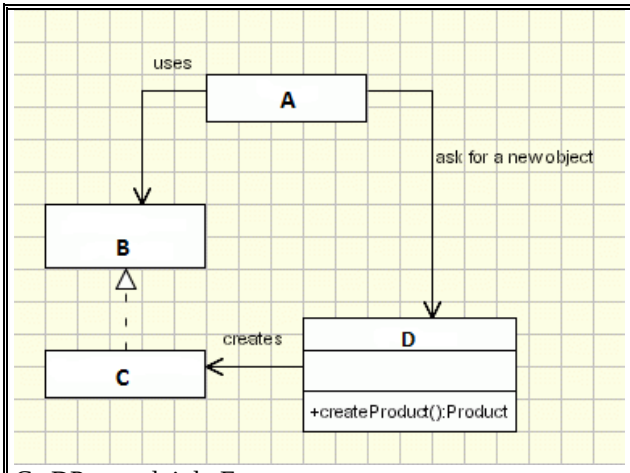
En RMI, l'amorce ou stub d'un Objet Distribué est un proxy de l'Objet Distribué qui est créé grâce à un Dynamic Proxy		Q 15.
1	OUI	
2	NON	

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 16.
1	OUI	
2	NON	

Dans un système réparti, le DP Singleton est utilisé pour créer un objet distribué unique sur le réseau		Q 17.
1	OUI	
2	NON	

Le rôle d'un factory dans une architecture distribuée est de créer à la demande de nouveaux objets distribués. Tous ces objets distribués doivent être enregistrés dans un adaptateur local à la machine sinon ils ne sont pas accessibles.		Q 18.
1	OUI	
2	NON	

Le DP Factory a pour fonction la création d'objet dont les classes héritent d'une même classe abstraite ou implémentent la même interface		Q 19.
1	OUI	
2	NON	



Q 20.

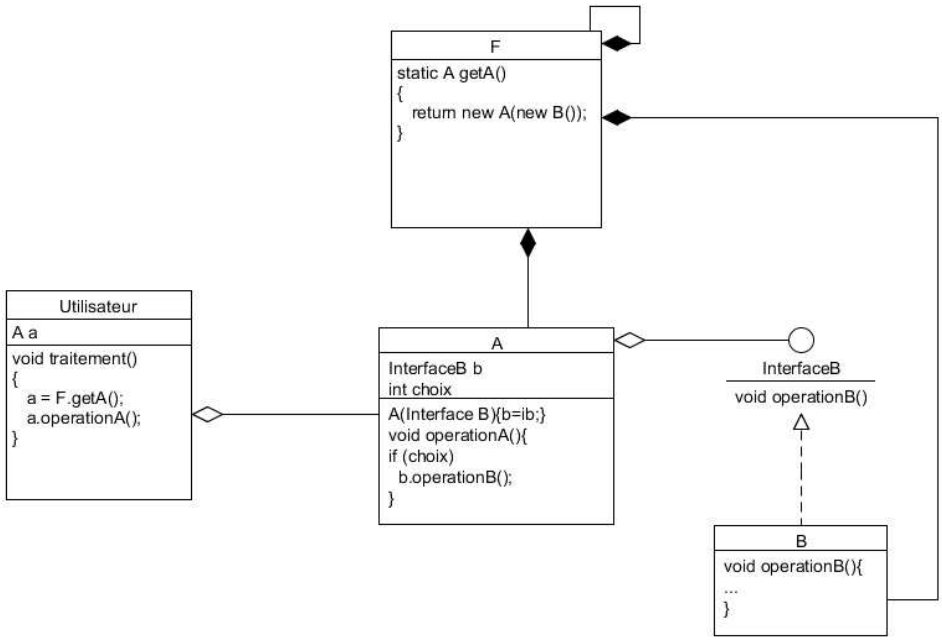
Ce DP est celui du Factory.
La signification des lettres A, B, C et D est :

1	A=Factory; B = Concrete Product; C=Product (Interface); D=Client	
2	A=Client; B=Factory; C=Product (interface); D=Concrete Product	
3	A = Client; B=Product (interface); C=Concrete Product; D = Factory	

Le rôle du DP "Délégation" est de déléguer à une autre classe de réaliser des traitements qu'une classe aurait dû implémenter.		Q 21.
1	OUI	
2	NON	

Le diagramme suivant :

Q 22.

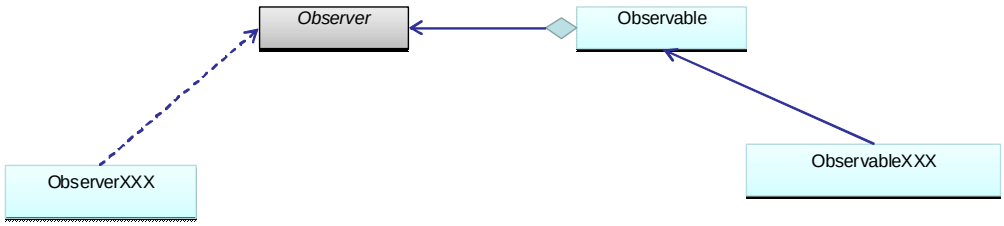


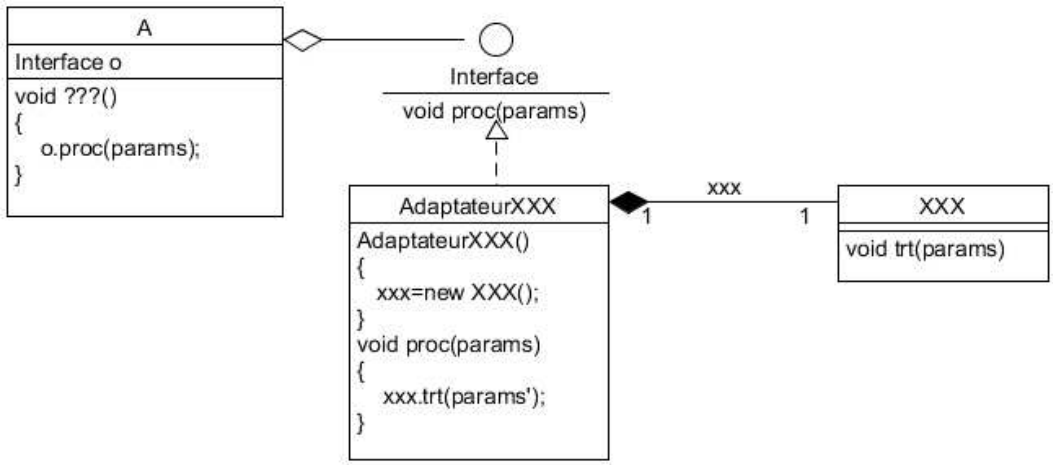
représente une injection de dépendance par l'utilisation d'un setteur

1	OUI	
2	NON	

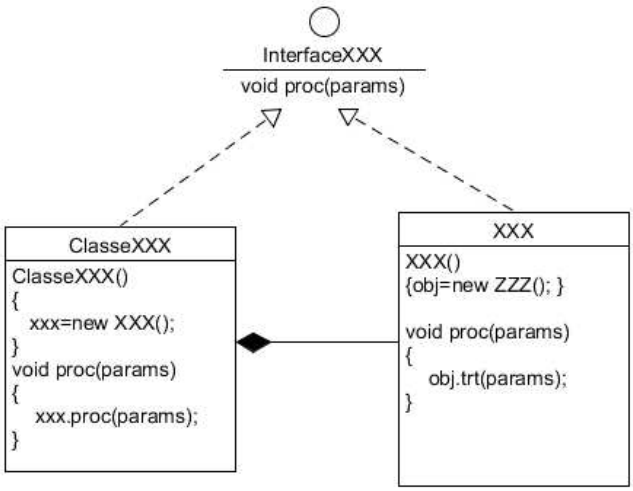
Dans le DP Observer, la communication entre l'Observable (producteur d'évènement) et l'Observer (consommateur d'évènement) est :		Q 23.
1	synchrone ou asynchrone (choix de conception)	
2	toujours asynchrone	
3	toujours synchrone	

Le DP Observeur/Observable, peut être utilisé pour réaliser un connecteur Producteur/Consommateur		Q 24.
1	OUI	
2	NON	

Soit le Design Pattern Observateur :		Q 25.
		
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	

Soit le diagramme de classe suivant :		Q 26.
		
Ce diagramme de classe représente celui d'un DP Adaptateur		
1	OUI	
2	NON	

Un Adaptateur est un DP constitué d'une classe A qui implémente une interface I à la place d'une autre classe B qui ne peut pas implémenter cette interface		Q 27.
1	OUI	
2	NON	

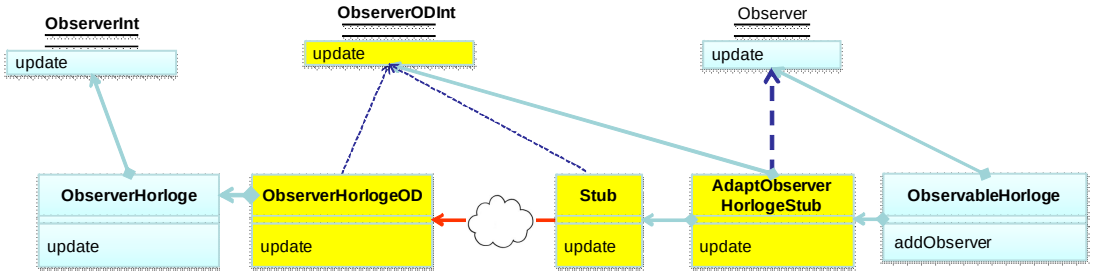
Soit le diagramme de classe suivant :		Q 28.
		
1	Ce diagramme de classe représente le DP Adaptateur	
2	Ce diagramme de classe représente le DP Proxy	

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 29.
1	OUI	
2	NON	

Le DP Observateur est implémentée en Java via la classe Observable et l'interface Observer. Cette implémentation utilise par conception le modèle de communication synchrone suivant :		Q 30.
1	modèle du pull	
2	modèle du push	
3	modèle du push-and-pull	

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	

Le modèle de communication "Push asynchrone" est un DP dans lequel la classe qui implémente l'interface Observer crée un thread qui réalise l'appel à la notification de l'Observable.		Q 32.
1	OUI	
2	NON	

Soit le schéma suivant :		Q 33.
		
Les classes et interfaces en gris représentent :		
1	un proxy client de communication entre ObservableHorloge et ObserverHorloge	
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 principe d'un MOM (Model Orienté Message) est d'utiliser un composant logiciel qui sert d'intermédiaire entre les producteurs et les consommateurs. Ce composant logiciel utilise :		Q 34.
1	un DP Factory de canaux d'évènement pour créer les canaux d'évènement	
2	Un DP Observer/Observable pour notifier les Consommateurs des évènements produits par les Producteurs	
3	Un DP ModelVueControler pour produire les évènements des Producteurs	

Dans le cadre de la communication entre un composant Java et un composant C++ sur un bus CORBA		Q 35.
1	on doit créer un socket de communication dans chacun des composants pour les faire communiquer	
2	on peut exécuter les deux composants sur la même machine	
3	on définit un IDL qui réalise une projection Java et une projection C++ des composants logiciels utilisés pour faire communiquer les deux composants	

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

Faire la description du Design Pattern du **Dynamic Proxy** sous la forme d'un diagramme de classe UML.
Quel est le rôle de ce Design Pattern ?

QUESTION NUMERO 2

Citez 3 exemples de l'utilisation du Design Pattern Proxy en précisant, pour chacun, le rôle fonctionnelle du proxy.

QUESTION NUMERO 3

Dans un MOM (Message Oriented Middleware), il existe deux modes de communication entre les producteurs et les consommateurs. Expliquez ce que sont ces deux modes de communication.

Fin de la 1ère partie sans document

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

3. PROBLEME (50 points)

On se propose de faire la conception d'un Système d'Information (SI) : **une plate-forme médicale**.

Cette plate-forme est composée de :

- un serveur [COMPOSANT LOGICIEL 1] qui :
  - centralise les dossiers de suivi de tous les patients des médecins (en mémoire)
  - centralise les agendas de chacun des médecins (en mémoire)
  - permet de créer une conversation en ligne avec 1 médecin ou la secrétaire,
  - permet de joindre une conversation existante;
- chaque médecin a son ihm [COMPOSANT LOGICIEL 2] qui lui permet de :
  - créer, consulter et mettre à jour son agenda personnel (créneaux disponibles);
  - créer, consulter et mettre à jour le dossier d'un patient
  - créer ou joindre une conversation
- la secrétaire a son ihm [COMPOSANT LOGICIEL 3] qui lui permet
  - d'enregistrer les rendez-vous dans l'agenda d'un médecin.
  - créer ou joindre une conversation

L'ihm d'un médecin contient un élément qui affiche en temps réel la liste des rendez-vous de la journée. Dès que son agenda est mis à jour par la secrétaire, cet élément se rafraichit instantanément.

1/ Faites le schéma d'architecture logicielle de votre solution (composants logiciels et sous-composants, liens fonctionnels entre les composants logiciels et les sous-composants).
Commentez votre schéma (fonctionnement, rôle, fonctions).

Un composant logiciel 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