IPST-CNAM
Intranet et Designs patterns
NSY 102
Vendredi 6 Mai 2021

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:

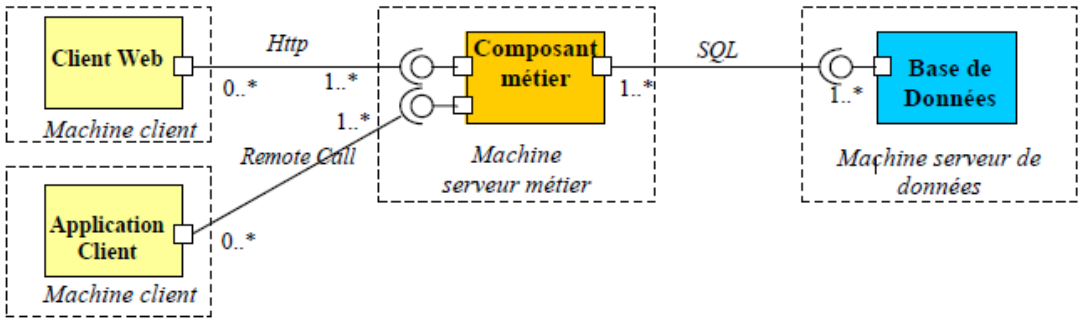
Réaliser l'architecture d'un Système d'Information (SI) permet de confirmer les choix techniques pour la réalisation du SI.		Q 1.
1	OUI	
2	NON	

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 à une dépendance distante (machine à machine) entre les composants	
2	souvent à la transmission d'information entre les composants	

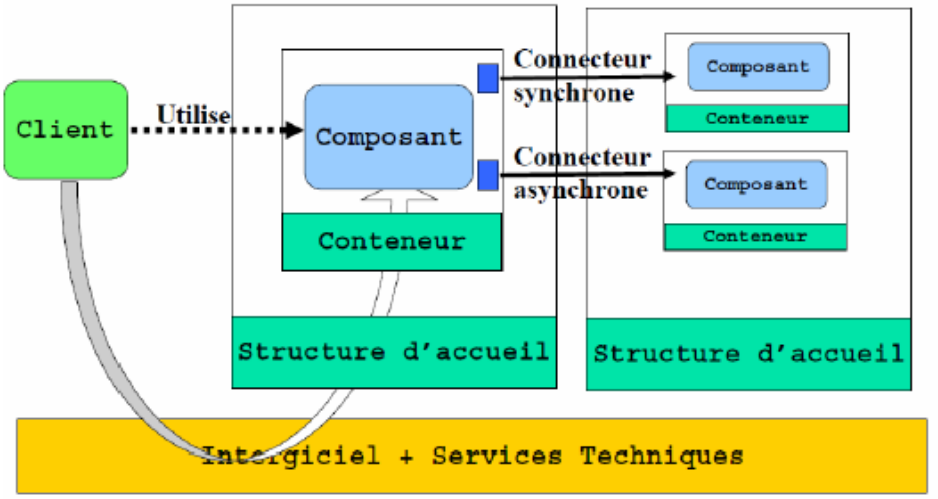
Dans la démarche d'architecture d'un Système d'Information, l'Architecture Technique est l'implémentation de la Configuration Architecturale dans une ou plusieurs technologies données.		Q 5.
1	OUI	
2	NON	



Ce schéma représente une architecture 4-Tiers		Q 6.
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	

Soit le schéma suivant de description d'une architecture type à base de composants :	Q 8.



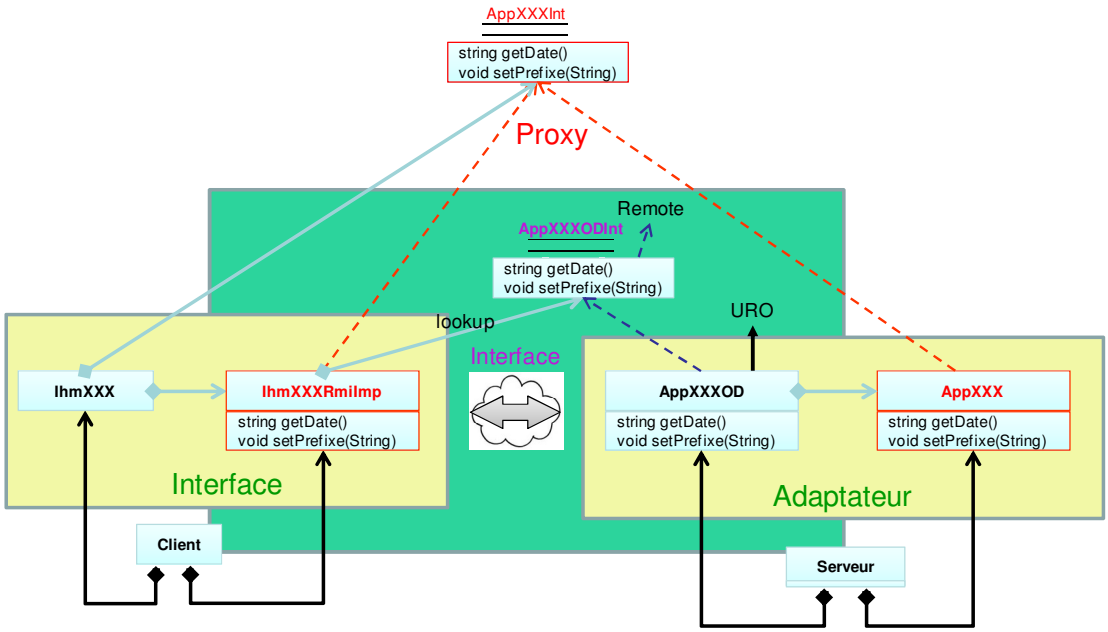
Ce schéma montre qu'un Client du SI basé sur ce type d'architecture utilise directement les composants du SI.

1	OUI	
2	NON	

Dans une architecture à base de composant, le conteneur assure la localisation et la résolution des dépendances entre les composants.	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. A et B peuvent appeler en même temps la méthode m1 de OD1	Q 10.
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 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	

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 13.
1 OUI	
2 NON	

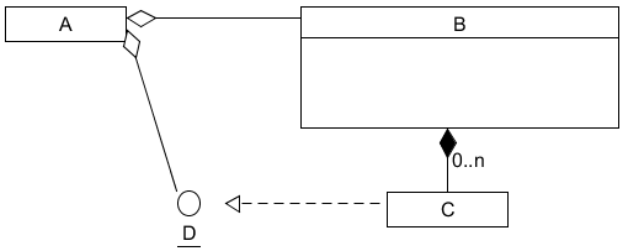
A l'opposé de la communication synchrone, la communication asynchrone est un type de communication notamment basé sur le modèle du pull, comme par exemple un thread d'un client qui tire régulièrement les événements d'un serveur.	Q 14.
1 OUI	
2 NON	

Un Design Pattern (DP) ou Patron de Conception est un template de code représentant l'implémentation d'un concept informatique qui répond à une problématique récurrente dans la réalisation d'un Système d'information.	Q 15.
1 OUI	
2 NON	

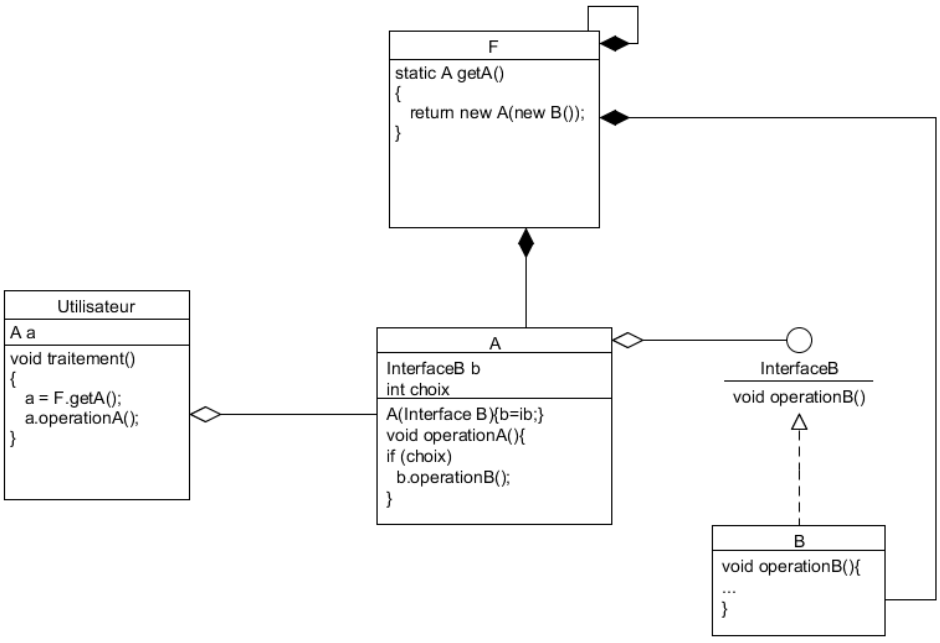
Dans un système réparti, le DP Singleton est utilisé pour créer un objet distribué unique sur le réseau	Q 16.
1 OUI	
2 NON	

Le rôle d'un factory dans une architecture distribuée peut être celui de créer, à la demande, de nouveaux objets distribués définis sur une interface distante donnée.	Q 17.
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 18.
1	OUI	
2	NON	

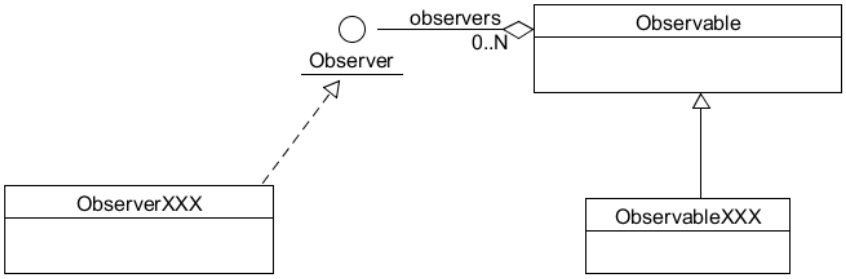
Ce DP est celui du Factory. 		Q 19.
La signification des lettres A, B, C et D est :		
1	A=Factory; B = Concrete Product; C=Product (Interface); D=Client	
2	A=Client; B=Factory; C=Concrete Product; D=Product (interface);	
3	A = Client; B=Product (interface); C=Concrete Product; D = Factory	

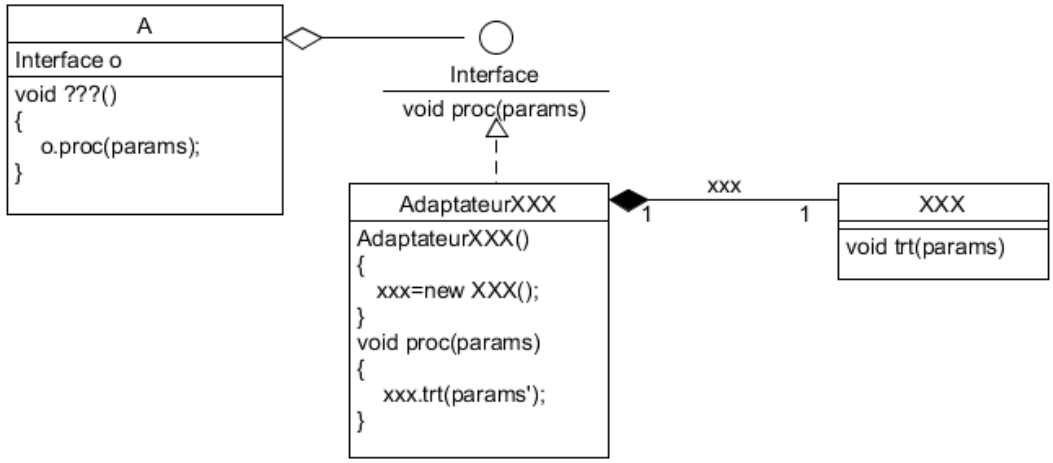
Le rôle du DP "Délégation" est de déléguer à une autre classe de réaliser des traitements qu'une classe aurait dû implémenter.		Q 20.
1	OUI	
2	NON	

Le diagramme suivant : 		Q 21.
représente une injection de dépendance par l'utilisation d'un setteur		
1	OUI	
2	NON	

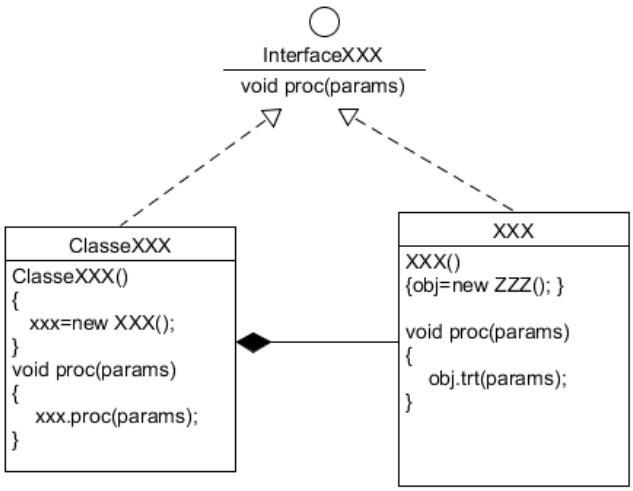
Dans le DP Observateur, la communication entre l'objet observé (producteur d'évènement) et l'observateur (consommateur d'évènement) est :		Q 22.
1	synchrone ou asynchrone (choix de conception)	
2	toujours asynchrone	
3	toujours synchrone	

Le DP Observateur, peut être utilisé pour réaliser un connecteur Producteur/Consommateur		Q 23.
1	OUI	
2	NON	

Soit le Design Pattern Observateur décrit (volontairement simplifié) de la manière suivante :		Q 24.
		
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 25.
		
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 26.
1	OUI	
2	NON	

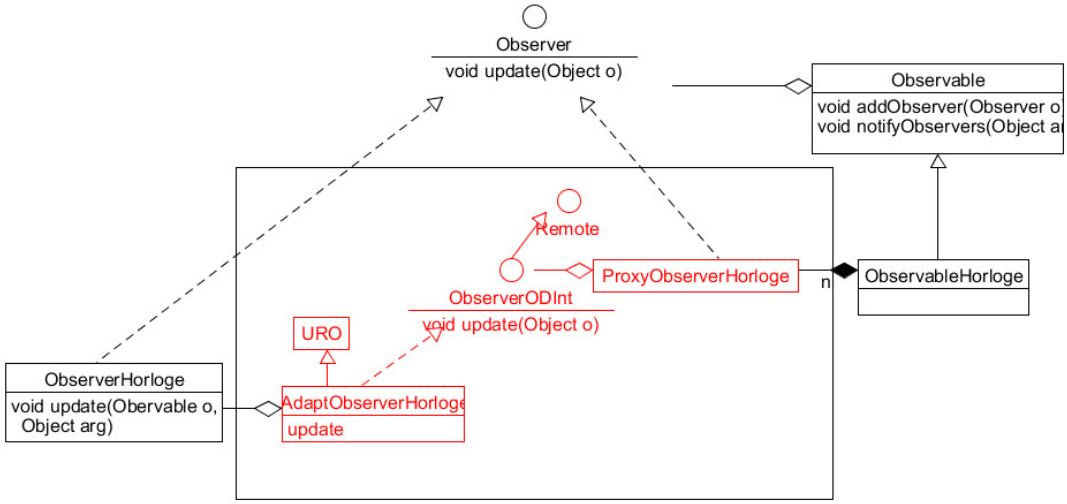
Soit le diagramme de classe suivant :		Q 27.
		
1	Ce diagramme de classe représente le DP Adaptateur	
2	Ce diagramme de classe représente le DP Proxy	

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 28.
1	OUI	
2	NON	

Le DP Observateur est décrit par la classe Observable et l'interface Observer. Ce DP utilise par conception le modèle de communication synchrone suivant :		Q 29.
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 30.
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 31.
1	OUI	
2	NON	

Soit le schéma suivant :		Q 32.
		
Les classes et interfaces encadrées 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 33.
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	

Laquelle des descriptions suivantes est un principe de communication synchrone :		Q 34.
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	

Dans la communication asynchrone via un "canal d'évènement" entre un producteur et un consommateur :		Q 35.
1	le producteur utilise un proxy de producteur (et non le producteur directement), afin de lui pousser un évènement	
2	le producteur utilise un proxy de consommateur (et non les consommateurs directement), afin de lui pousser un évènement	

Fin du QCM

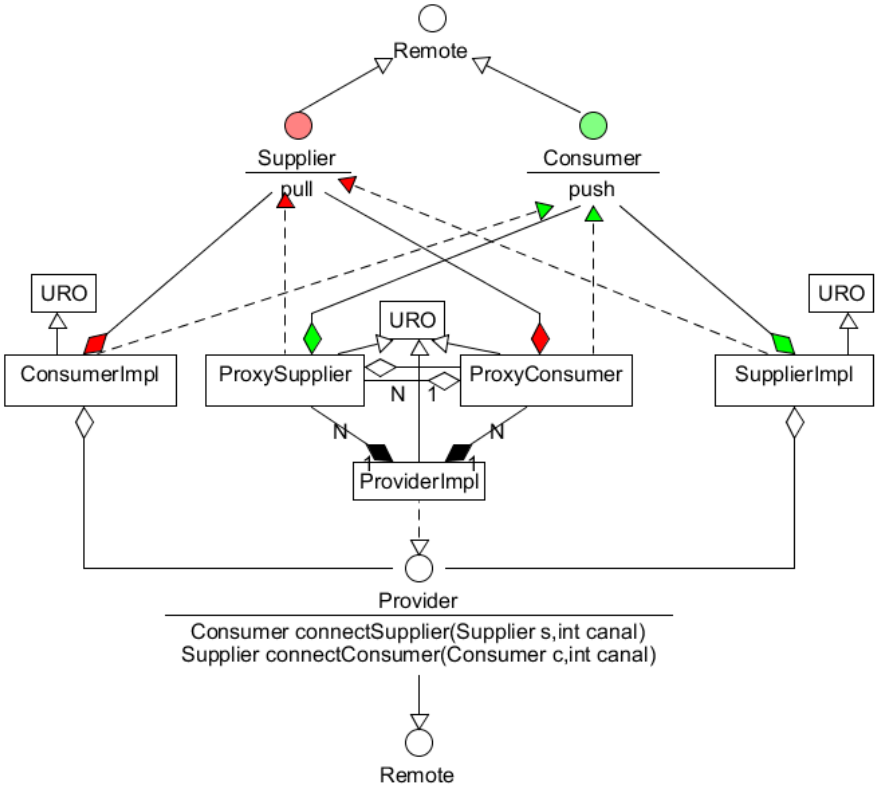
Suite (Tournez la page)

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

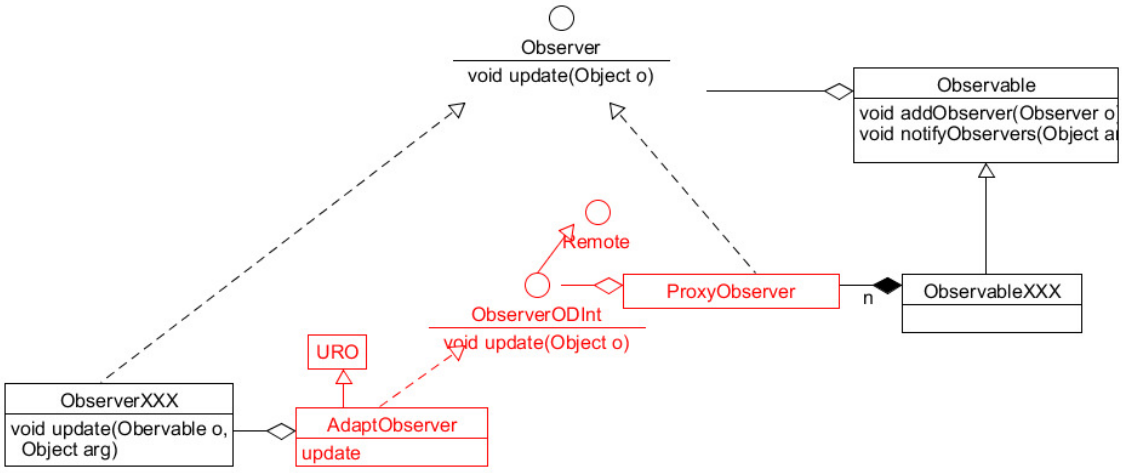
Soit le diagramme complet du DP d'une communication MOM suivant :



Expliquer le rôle de chacune des classes : ConsumerImpl, ProxySupplier, ProxyConsumer et SupplierImpl.

QUESTION NUMERO 2

Soit le Design Pattern suivant :



A quoi sert ce DP ?
Expliquez son fonctionnement.

QUESTION NUMERO 3

Expliquez le rôle du diagramme de communication dans la réalisation de l'architecture d'un Système d'Information.

Fin de la 1ère partie sans document

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

3. PROBLEME [50 points]

Nous envisageons de réaliser un Système d'Information (SI) qui déplace à distance des robots (non autonomes) sur une piste d'aviation. Le rôle de ses robots est de détecter, grâce à une caméra embarquée, un objet intrus qui se trouverait sur la piste. Chaque robot connait sa position au cm près (UHF-RFID) et fait une acquisition à 1 image/s.

Ce SI est composé d'un serveur de gestion des robots [COMPOSANT 1 - SERVEUR]. Chaque robot est représenté par un Objet Distant créé sur ce serveur [COMPOSANT 2 – ROBOT OD]. Chaque robot physique [COMPOSANT 3 – ROBOT] envoie ses images à son Objet Distant associé.
Un traitement du serveur calcule les parcours de déplacement de chaque robot en fonction de la zone à couvrir (choix dans les configurations des pistes d'aviation). Les ordres de déplacement calculés sont effectués par chaque robot durant tout le temps de la mission de contrôle de la piste d'aviation.

Le serveur de gestion des robots contient un traitement d'analyse d'image permettant d'envoyer une alerte à des opérateurs qu'il y a peut-être un objet intrus détecté.

Chaque opérateur est posté derrière une IHM [COMPOSANT 4 - IHM]. Toutes les IHM sont notifiées de toutes les alertes. Un opérateur peut alors sélectionner un groupe d'alertes d'un robot (les alertes sont alors bloquées pour les autres opérateurs). Il peut alors visualiser les images de ces alertes afin de prévenir si nécessaire le personnel technique qui ira voir sur zone pour ramasser l'objet intrus. Les images sont stockées sur le serveur de gestion des robots.
Une alerte est caractérisée par l'id du robot et l'id de l'image. Une image est caractérisée par son heure (hhmmss) et sa position).

Le traitement d'analyse d'image consiste à appliquer différents algorithmes de traitement successifs qui doivent être possibles de faire évoluer facilement.

Un fichier de configuration contient la description de la zone à couvrir, le nombre de robot, le nombre d'IHM des opérateurs.

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 et les sous-composants si besoin.
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 connaitre vos choix d'organisation des composants, le sens des communications et les différents points d'entrée de communication dans les composants distants.

2/ **[35 points]**
Faites le(s) diagramme(s)e de classe UML des [COMPOSANT 1], [COMPOSANT 2] et [COMPOSANT 4] 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