# Conception Orientée Objet – Examen – Modèle

**Durée : 3H - Documents autorisés sauf livres et appareils électroniques - Utilisation du crayon autorisée**

## Exercice 1. Diagramme d'activité : publication de travail de recherche (3,5 points)

Considérons la procédure de publication de travail de recherche d'un chercheur d'un laboratoire. Le chercheur fait sa recherche de son côté et a des résultats à publier. Il résume son travail avec les résultats dans un article. Il présente la version beta de l'article au sein du laboratoire pour avoir des conseils de ses collègues. Il aura ensuite la version prête et l'enverra à l'éditeur pour être revue par un reviewer désigné par l'éditeur. Le reviewer va soit accepter l'article avec des conseils d'amélioration envoyés au chercheur pour qu'il modifie son article afin d'avoir la version finale à publier ; soit refuser l'article en envoyant au chercheur les arguments. Si l'article est accepté, le chercheur demande au laboratoire de payer les frais de publication à l'éditeur qui va publier enfin la version finale de l'article. Avant de payer les frais de publication, le chercheur peut annuler à tout moment sa publication auprès de l'éditeur. Modéliser les activités du chercheur, du laboratoire, de l'éditeur et du reviewer dans cette procédure de publication, avec un diagramme d'activité UML.

## Exercice 2. Diagramme d'état-transition : drone pompier (4 points)

Considérons un drone volant qui peut intervenir dans les zones d'incendie. Une fois démarré à la base, le drone se met en état d'initialisation. Ensuite, il va voler depuis la base jusqu'à une destination programmée. En arrivant à la destination (la zone d'incendie), le drone commence à capturer, avec ses caméras, les images de l'environnement et puis analyse (traite) les images capturées, afin de repérer les endroits qui ont besoin des interventions. Après, le drone va projeter des matières à ces endroits pour essayer d'éteindre le feu. Le drone reste toujours autonome, après le départ depuis la base. Mais on peut aussi manipuler le drone à distance si on le souhaite. Après une opération manuelle, le drone peut revenir au mode autonome et continuer ses tâches. On peut aussi demander au drone d'annuler sa mission et de revenir à la base à tout moment. Quand le drone finit ses tâches, il revient tout seul à la base. Sur le chemin de retour, il réalise un petit résumé de sa mission et l'affiche sur son petit écran en arrivant à la base. Modéliser ce drone avec un diagramme d'état-transition UML.

## Exercice 3. Domain Diven Design (DDD) (3,5 points)

Une agence immobilière a besoin d'une application Java pour faciliter les ventes des biens immobiliers. Pour chaque bien, on doit connaître le nombre total de pièces, la surface en mètre carré et le prix unitaire par mètre carré. De plus, chaque bien se situe à une adresse postale composée du numéro de la rue, nom de la rue (avenue, boulevard…), nom de la ville (commune) et code postal. Comme il y a des immeubles en ville, certains biens peuvent être à la même adresse. Chaque bien a une référence unique générée par le système. Il y a deux types de biens immobiliers : appartement et maison. Pour un appartement, on s'intéresse en particulier au type de chauffage. Pour une maison, on devra connaître la surface en mètre carré du jardin de la maison (0 mètre carrée signifie qu'il n'y a pas de jardin). Toutes ces informations constituent un catalogue dans lequel on peut effectuer les recherches de biens immobiliers avec différents critères. On doit pouvoir bien évidemment ajouter, modifier ou supprimer les biens dans le catalogue. Pour chaque vente, on doit connaître le bien immobilier vendu et la date de la vente. Une fois que le bien est vendu, il change d'état et devient indisponible. Pour chaque vente, on voudrait aussi connaître les informations du client ayant acheté le bien. Pour cela, le nom, le prénom et le numéro de téléphone du client nous intéressent. Pour l'ensemble des ventes, on doit pouvoir effectuer des opérations d'ajoute, recherche, modification et suppression. Chaque transaction est gérée par un ou deux notaires désignés (choisis).

Langage ubiquitaire :

Bien immobilier : une maison ou un appartement qui se situe à une adresse postale
Adresse postale : une adresse connue dans l'annuaire, à laquelle peut se trouver une maison ou un immeuble contenant des appartements
Annuaire d'adresses : l'ensemble d'adresses postales existantes et fixes
Catalogue de biens : l'ensemble de biens immobiliers à vendre ou vendus
Vente de bien : une transaction entre l'acquéreur et le vendeur d'un bien immobilier
Notaire : personne qui s'occupe de la transaction de vente de bien.
Annuaire de notaires : l'ensemble de notaires existants et fixes

Modélisez ce problème avec un diagramme de classe UML en suivant l'approche DDD. Dans le diagramme, vous indiquerez les noms des classes*, ainsi que les liens entre les classes*. Vous légenderez / indiquerez aussi les agrégats avec la classe* racine, les types de classes* « Entité » et « Objet de valeur », ainsi que les classes* de type service : services fonctionnels, factory et repository. Vous n'avez **pas** besoin d'indiquer les attributs ni les méthodes dans les classes. (Certaines informations dans le sujet vous permettent uniquement de mieux comprendre comment le système fonctionne.)

* Les classes peuvent être classes concrètes, classes abstraites ou interfaces.

**Exercice 4. Principes d'objet et design patterns (5,5 points)**

**4.1** Expliquez **comment** le pattern « *Proxy* » rassure la Loi de Déméter (*LoD*). **(1point)**

**4.2** Comparez les deux modes du pattern « *Composite* » : mode « sécurisé » et mode « transparent » (expliquez brièvement la différence entre eux ainsi que les avantages et inconvénients des deux solutions). **(1 point)**

**4.3** On souhaite modéliser un mini système pour gérer la réservation des vols d'une compagnie aérienne. Un vol concerne un aéroport de départ et un aéroport d'arrivée. Un vol est aussi associé à une date/heure de départ et une date/heure d'arrivée. Chaque réservation de vol peut inclure au maximum 3 voyageurs dont 2 passagers adultes, et éventuellement un bébé de moins de 2 ans. Un bébé doit être obligatoirement associé à un voyageur adulte. Il y a deux types de réservations : en ligne ou en agence de la compagnie aérienne. On doit pouvoir annuler une réservation. Lors du traitement de l'annulation, on a besoin de calculer le montant de remboursement. Ce montant sera calculé différemment en fonction du type de réservation (on ne considère pas la date d'annulation dans le calcul de remboursement, mais l'annulation peut se faire uniquement au plus tard 24h avant l'heure du vol.) Les différentes fonctionnalités du système sont regroupées dans différentes interfaces accessibles pour différents utilisateurs : clients et ceux qui travaillent dans les agences. Modéliser ce programme par un diagramme de classe UML, vous y indiquerez **uniquement** les classes avec leurs noms, ainsi que les relations entre les classes, **PAS BESOIN de** mentionner les attributs, ni les méthodes, ni les constructeurs. Indiquez les **design patterns** utilisés par votre modélisation, directement dans votre diagramme. (**3,5 points**)

**!!! Attention : Vous choisirez UN SEUL exercice à faire parmi 5a et 5b**

**Exercice 5a. Diagramme de cas d'utilisation : gestion d'une bibliothèque self-service (3,5 points)**

Considérons un système de gestion de bibliothèque self-service. Les lecteurs enregistrent eux-mêmes les livres à emprunter. Pour cela, ils doivent s'authentifier avec leur carte d'abonnement et scanner le code-barres du livre. Pour le retour des livres, les lecteurs peuvent, soit le faire eux-mêmes sur la machine, soit à l'aide d'un bibliothécaire. Quotidiennement, les bibliothécaires ont besoin de vérifier, à l'aide du système, la durée des emprunts de tous les lecteurs car l'emprunt d'un exemplaire d'un livre est limité à trois semaines. Si l'exemplaire n'est pas restitué dans ce délai, le bibliothécaire mettra une amende au lecteur. Si l'exemplaire n'est toujours pas rendu au bout d'un an, une procédure judiciaire sera déclenchée par le bibliothécaire. De plus, les bibliothécaires ont deux autres tâches à assurer : **1)** Gestion des livres qui concerne essentiellement l'ajoute et la suppression des livres **2)** Recherche des livres en fonction des demandes des lecteurs qui donnent les informations clés, par exemple, titres, genre, auteur, etc. Modélisez ce système de gestion de bibliothèque par un diagramme de cas d'utilisation UML.

**Exercice 5b. Diagramme de séquence : gestion des formations des employés (3,5 points)**

Considérons le travail de la gestion des formations des employés dans une entreprise. On dispose d'un système de gestion de formation (SFG) permettant de faciliter ce travail. Les principales personnes concernées sont : 1) employé (demandeur de formation) 2) responsable de formation de l'entreprise 3) organisateur de formation (fournisseur de formation). Tout d'abord, l'employé consulte le catalogue du système pour connaître les formations disponibles. Ensuite, pour effectuer une demande, il doit choisir une formation. S'il le souhaite, il peut éventuellement préciser une session particulière de la formation choisie. Une fois qu'il valide sa demande, le système SFG va automatiquement enregistrer cette demande et envoyer un email de notification au responsable de formation de l'entreprise. Ce dernier va traiter cette demande immédiatement. Il a deux options possibles : soit il accepte la demande en notifiant le système SFG. Ce dernier envoie ensuite un email de notification à l'employé demandeur et envoie en même temps un bon de commande à l'organisateur de formation ; soit il refuse la demande en saisissant un motif sur le système SFG qui va ensuite transmettre ce motif de refus à l'employé demandeur. Modélisez les interactions entre ces différentes personnes et le SFG par un diagramme de séquence UML.