Génie Logiciel : Conception (INF 5153) – *Examen Intra Automne 2019*

- Les seuls documents autorisés sont *(i)* la *cheatsheet* UML de la page web du cours et *(ii)* une feuille de note manuscrite (format *Letter, recto-verso, non photocopiée*) personnelle.
- Vos réponses sont à donner directement dans le cahier d'examen fourni. Vous pouvez utiliser toutes les pages du cahier (les pages blanches préférablement pour les diagrammes UML)
- La « propreté » de vos diagrammes (et de votre copie) fait partie des critères d'appréciations.
- Si le sujet vous parait ambiguë, expliquez dans votre copie le choix pris pour lever l'ambiguïté. **Le surveillant ne répondra à aucune question**.
- Indiquez très clairement dans votre copie à quelle question vous répondez pour faciliter la correction.

Partie A : Des petites questions de cours (/50, ≈ 90 minutes)

Chacune des questions de cette partie est sur 7,5 points. Vous pouvez répondre a autant de questions que vous le souhaitez, mais la somme totale des points obtenus à cette partie de l'examen ne pourra pas dépasser 50 points. Choisissez vos questions et répartissez votre temps en conséquence.

Question A.1 : Relations de Composition, de Réalisation et de Généralisation */ 7,5*

Définissez ces trois types de relations : comment les représenter en UML, quel est leur sémantique dans la conception, et comment les implémenter en Java. Quand choisiriez-vous l'une plutôt que l'autre ?

Question A.2 : Relation d'héritage et relation de sous-typage */ 7,5*

On considère le diagramme de Venn décrit dans la figure 1, qui représente les relations de sous-typage (par inclusion d'ensembles) existant entre six types A, B, C, D, E et F. Proposez un diagramme de classes respectant ces relations. Comment implémenteriez-vous cette hiérarchie de type en Java (qui serait une classe, et qui serait une interface) ?

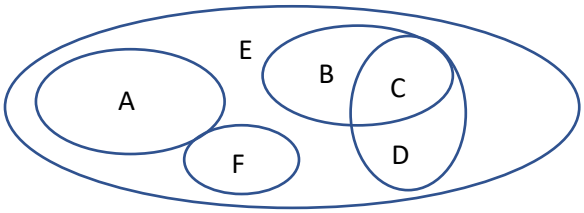

***Figure 1**. Diagramme de Venn représentant les relations de sous-typage pour les types [A-F], par inclusion d'ensemble.*

Question A.3 : Patron « Créateur » */ 7,5*

La famille de patrons GRASP définit le patron créateur comme un guide permettant d'affecter la responsabilité de créer des instances de classes au « bon » endroit. Donnez une définition de ce patron GRASP.

Question A.4 : Principe Ouvert-Fermé */ 7,5*

L'UQAM souhaite se doter d'un système de contrôle d'accès au bâtiment reposant sur des cartes magnétiques. Les portes sont équipées de lecteurs de cartes, et certaines zones sont réservées aux employés. Les cartes des visiteurs ouvrent les portes non restreintes de 08h à 20h. Les cartes des employés et des stagiaires donnent accès aux zones restreintes (24h/24 pour les employés, et uniquement entre 09h et 17h pour les stagiaires).

On considère le diagramme de classe de la figure 2. Expliquez en quoi il ne respecte pas le principe ouvert-fermé. Proposez une conception équivalente mais respectant ce principe.

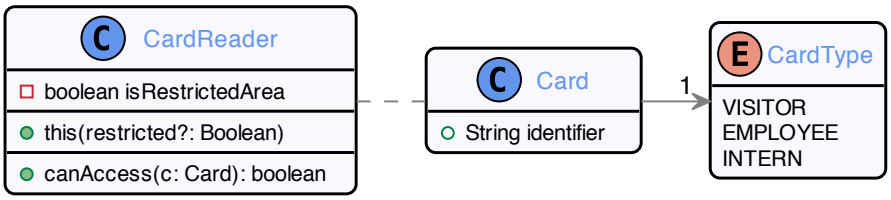

***Figure 2**. Diagramme de classe ne respectant pas le principe ouvert-fermé*

Question A.5 : Extraction d'un diagramme de classe / 7,5

On considère le diagramme de séquence de la figure 3. Proposez un diagramme de classes permettant d'exprimer les enchainements de messages décrit dans la figure 1 (en exploitant du polymorphisme).

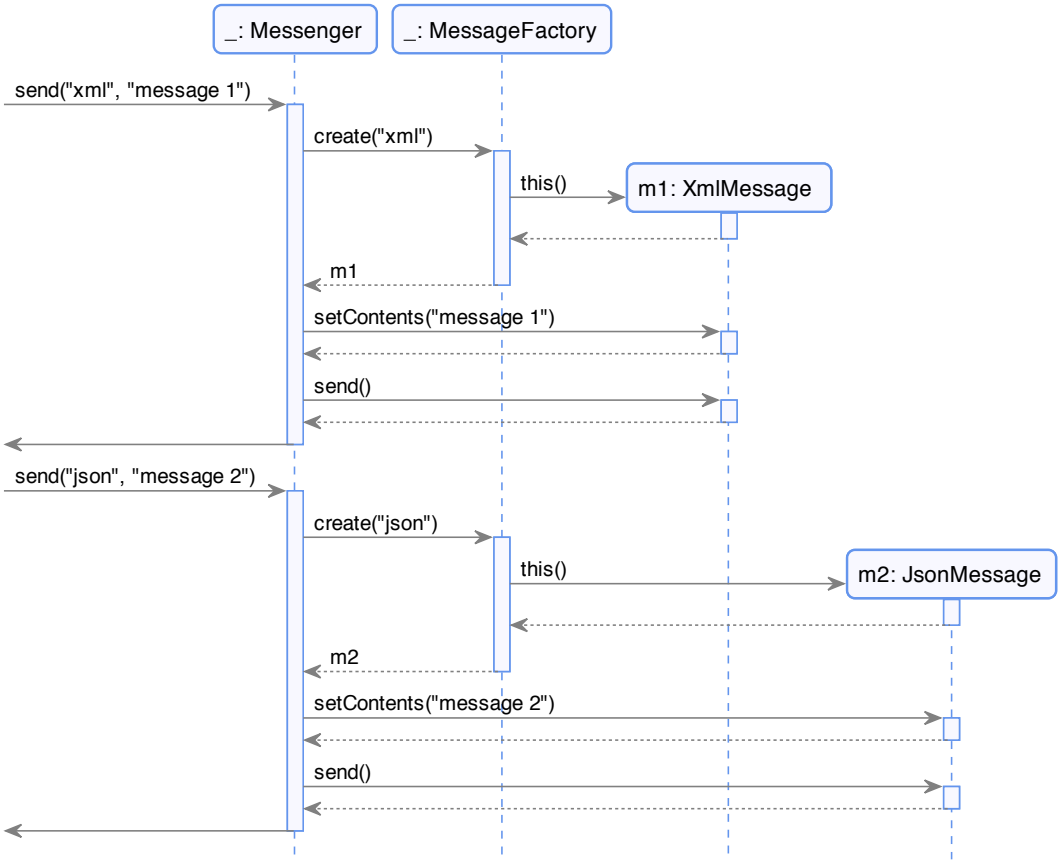

***Figure 3**. Diagramme de Séquence*

Question A.6 : Égalité et Équivalence / 7,5

On considère qu'un étudiant de l'UQAM est défini par son nom, son code permanent, et la liste des cours (leur sigle) qu'il a choisi à la session courante. Définissez et justifiez une relation d'équivalence ainsi qu'une relation d'égalité pour cette conception. Comment différenciez-vous ces deux relations ? Par quel moyen les implémenteriez-vous en Java ?

Question A.7 : Alternatives de conception pour le polymorphisme

On considère un calculateur d'empreinte carbone pour les trajets individuels (qui calcule la quantité de CO_2 en grammes rejetée pour une distance donnée en kilomètres), qui se comporte différemment selon que le trajet à effectuer est réalisé en train, en voiture ou en avion.

Le polymorphisme en Java est utilisable sous deux formes : surcharge de méthode (*overloading*), ou aiguillage dynamique (*overriding*). Sur l'exemple précédent, proposez une conception (diagramme de classe) utilisant la surcharge de méthode, et une autre utilisant l'aiguillage dynamique.

Question A.8 : Faible couplage et forte cohésion */ 7,5*

Une bonne conception objet vise à définir une architecture où le couplage est minimal et la cohésion maximale. Définissez chacune de ces notions, et expliquez dans vos mots pourquoi cet objectif est intéressant dans le cadre d'un développement logiciel.

Question A.9 : Rôle des diagrammes UML */ 7,5*

Le langage UML définit trois familles de diagrammes : statique, fonctionnel, et dynamique. Classez les diagrammes vus dans le cours selon ces trois familles, et expliquez leur rôle dans le développement d'une application orientée objet.

Question A.10 : Loi de Demeter */ 7,5*

Donnez une définition de la Loi de Demeter en conception objet. Expliquez en quoi il est nécessaire de respecter ce principe de conception lors du développement d'une application objet.

Partie B : Simulation Informatique (/ 50, ≈ 90 minutes)

Vos diagrammes UML doivent être cohérents entre eux. Par exemple les diagrammes de séquences doivent utiliser des méthodes définies dans le diagramme de classe, ou les diagrammes d'objet doivent instancier des classes existantes. Lisez toute la partie avant de répondre, typiquement la question B.5 peut vous aider à prendre du recul pour bien concevoir le problème décrit.

L'outil informatique est souvent utilisé pour faire des simulations. On s'intéresse dans cette partie à une famille de simulation classique, les jeux de simulations. Ces systèmes utilisent une grille comme support, et appliquent des règles pour décider des actions faites sur la grille.

Les deux simulations les plus connues sont : Le « Jeu de la Vie » de Conway (1970), et la « Fourmi » de Langton (1986). Le principe d'exécution de ces simulations repose sur des tours d'exécution. Chaque tour, on applique aux cases l'ensemble des règles disponibles, et une fois que toutes les règles sont appliquées, on passe au tour suivant. Dans cet examen, on s'intéressera uniquement au Jeu de la Vie, mais **votre conception doit être extensible** (*au sens du principe ouvert-fermé*) **à d'autres simulations**.

Le Jeu de la Vie repose sur un damier de taille connue, dont les cases peuvent être « *vivantes* » ou « *mortes* ». La simulation est lancée pour un nombre de tours connus à l'avance, sur un damier initialisé au hasard. Chaque tour, les règles de simulation sont appliquées sur le damier, et on passe ensuite au tour suivant. Les règles de simulation du jeu de la vie sont les suivantes :

R1. Une case morte devient « vivante » si elle est entourée d'exactement 3 cases vivantes voisines (naissance) ;
R2. Une case « vivante » qui a 2 ou 3 cases vivantes voisine reste vivante (maintien) ;
R3. Une case « vivante » meurt si elle a moins de 2 cases vivantes voisines (sous-population) ;
R4. Une case « vivante » qui a plus de 3 case vivantes voisine meurt (surpopulation).

Un exemple de 3 tours sur un damier de 4x4 est présenté ci-dessous. Les cellules vivantes sont représentées en noir, les cellules mortes sont en blanc.

1. Au temps T0, le damier a été initialisé en rendant vivantes les cases {5,6,7,8,10,11}.
2. Au temps T1, les règles suivantes ont été appliquées pour obtenir le nouveau damier :
    a. Les cases {2,3, 9, 12} deviennent vivantes par la règle R1 (naissance) ;
    b. Les cases {5,8} restent vivantes par la règle R2 (maintien) ;
    c. Les cases {6,7,10,11} meurent par la règle R4 (surpopulation).

3. Au temps T2, les règles suivantes ont été appliquées pour obtenir le nouveau damier :
    a. Les cases {2,3,5,8} restent vivantes par la règle R2 (maintien) ;
    b. Les cases {9, 12} meurent par la règle R3 (sous-population).
4. Au temps T3, les règles suivantes ont été appliquées pour obtenir le nouveau damier :
    a. Les cases {6,7} deviennent vivantes par la règle R1 (naissance) ;
    b. Les cases {2,3} restent vivantes par la règle R2 (maintien) ;
    c. Les cases {5,8} meurent par la règle R3 (sous-population).

Il est intéressant de noter qu'à partir de maintenant, la structure du jeu de la vie est dans une configuration dite « stable », et que seule la règle du maintien (R2) peut s'appliquer, puisque chaque case « vivante » possède exactement trois voisines et que les conditions nécessaires à des naissances (R1) ne sont plus réunies.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

T0 (initial)

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

T1

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

T2

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

T3 (stable)

***Tableau 1**. Illustration de trois tours de simulation du Jeu de La Vie*

Question B.1 : Conception orientée objet / 15

Proposez une conception permettant de représenter la simulation du jeu de la vie sous la forme d'une application orientée objet. Vous représenterez cette conception sous la forme d'un diagramme de classe UML.

Question B.2 : Initialisation du damier / 5

Le commanditaire n'a pas spécifié ce qu'il entendait par « un damier initialisé au hasard ». Proposez un diagramme de séquence explicitant votre mise en œuvre de ce point de fonctionnalité.

Question B.3 : Déroulement de la simulation / 10

En utilisant le type de diagramme de votre choix, décrivez en un seul diagramme comment se déroule une simulation.

Question B.4 : Justification des choix de conceptions / 15

Justifiez votre proposition de conception au regard des principes et patrons vu en cours durant la session (par exemle., les principes SOLID, les patrons GRASP)

Question B.5 : Extension à la Fourmi de Langton / 5

La fourmi de Langton est une simulation qui repose sur un damier dont les cases sont « *blanche* » ou « *noire* » (initialisé au hasard), avec une fourmi posée sur une case. Les règles de simulation sont les suivantes : si la fourmi est sur une case noire, elle tourne de 90º à gauche. Si elle est sur une case blanche, elle tourne de 90º à droite. Puis elle inverse la couleur de la case courante, et avance d'une case. La simulation termine quand la fourmi sort du damier.

Expliquez en quelques lignes comment il sera possible de réutiliser votre conception pour permettre de définir la *Fourmi de Langton* à partir des abstractions identifiées dans le *Jeu de la Vie*.