1 Objectifs d'apprentissage

- Être capable de représenter par un modèle UML une solution au problème exposé dans un cahier des charges.
- Être capable de réaliser des diagrammes UML (séquences, classes) pour décrire une solution.
- Être capable de modéliser des aspects structurels et temporels d'un système.
- Réfléchir à la réalisation des cas d'utilisation en inventant, découvrant, choisissant les informations que les objets doivent échanger.

2 Travail à réaliser

On progresse dans le cycle en V :

Analyse des besoins Recette

Analyse

Conception Test intégration

Conception détaillée Tests unitaires

Réalisation

Après l'analyse des besoins (BE1) et la préparation de la recette (BE2), voici venu le temps de la *conception* du produit souhaité. On quitte la perspective « boîte noire » pour s'interroger sur l'organisation interne de cette boîte (« boîte blanche »). L'objectif de cette troisième étape est d'aboutir à une description de votre solution sous la forme d'un **diagramme de classes** accompagné d'un ou plusieurs **diagrammes de séquence** précisant les interactions temporelles (envois de message) entre ces classes. Pour concevoir ce diagramme de classes, vous aurez à imaginer et mettre en scène (cf diagrammes de séquence) des classes qui vous paraîtront utiles et nécessaires pour offrir les services attendus, tels que décrits dans le cahier des charges.

Vous adopterez encore une fois une démarche incrémentale, précisée ci-dessous.

Dans un premier temps, vous allez démarrer uniquement avec les fonctionnalités relatives à la *gestion des membres* : votre premier objectif est donc de ne considérer que les méthodes `nbMembers()`, `addMember()` et (une première version de) `toString()` de `SocialNetwork`.

- Élaborez un *diagramme de classes* minimal permettant au `SocialNetwork` de gérer des membres.
- Élaborez conjointement les *diagrammes de séquence* correspondant à l'appel de la méthode `addMember()` spécifiée par l'interface `ISocialNetwork`**.** BibTel exige un diagramme de séquence par type d'anomalie (exception), en plus bien sûr d'un diagramme illustrant le fonctionnement nominal du service.

Ces diagrammes sont à déposer sur Moodle pour le fin de la séance.

Vous pouvez ensuite passer à la deuxième itération de la conception : enrichissez votre diagramme de classes tout en élaborant des diagrammes de séquence *nominaux* pour permettre la *gestion des films* (cf méthodes `nbFilms()`, `addItemFilm()` et `reviewItemFilm()`). **Cette deuxième itération fera l'objet d'une livraison avant la prochaine séance (nouveau diagramme de classes, plus diagrammes de séquence des cas nominaux pour `addItemlFilm()` et `reviewItemFilm()`).** Si vous le souhaitez, le diagramme de séquence de `addItem()` (et seulement lui) pourra être réalisé au crayon pour gagner du temps.

Lors de la prochaine séance de travail, vous terminerez la troisième itération de conception du lot1 en prenant aussi en compte la gestion des livres (cf méthodes `nbBooks()`, `addItemBook()` et `reviewItemBook()`).

3 Annexes

3.1 Quelques bonnes pratiques pour construire les diagrammes

- Toujours privilégier une démarche *incrémentale* : répéter le processus en accroissant à chaque itération le périmètre traité permet d'aboutir progressivement à une solution globale, et ce de façon beaucoup plus avisée qu'en tentant d'aboutir directement. Cela correspond au trois itérations qui vous sont indiquées précédemment.
- De même, à chaque itération, commencez par un *diagramme de classes*, avec des classes *sans attribut ni méthode*. Explicitez ensuite les *associations* entre ces classes. Vous pouvez alors ajouter des *attributs* (sans chercher l'exhaustivité). L'identification des *méthodes* de chaque classe pourra s'appuyer sur l'élaboration des *diagrammes de séquences*. Ce diagramme donne en effet un point de vue plus propice aux questionnements et aux choix quant aux responsabilités respectives des différentes classes, et guide ainsi la spécification des méthodes associées à ces classes. Tout ce processus sera réitéré autant de fois que nécessaire.
- Sur le diagramme de classes, une extrémité d'association ne doit être déclarée *navigable* qu'à raison : toujours questionner l'intérêt de cette navigabilité par rapport au coût de gestion que cela induit (car en cas de navigabilité, il faudra gérer l'attribut correspondant). Par ailleurs, sur le diagramme de classes, l'attribut engendré par une extrémité navigable d'une association (le rôle) ne doit être affiché qu'à cet endroit et non dans la classe où il sera finalement implémenté lors du codage.
- Dans la même optique d'encapsulation, un attribut n'aura des accesseurs (méthodes `get` et `set`) que pour raisons dûment justifiées.

3.2 UML

Vous pourrez éditer vos diagrammes UML avec votre outil de prédilection, notamment parmi ceux que vous avez testés en INF111. Si cela a un sens pour l'outil que vous utilisez, les diagrammes que vous devez élaborer doivent faire partie du package `opinion`.

En plus des ressources déjà mises à disposition dans le cadre de l'UE INF111, vous pourrez également trouver des réponses à vos questions sur UML dans les références suivantes :

[1] http://uml.free.fr/index-cours.html « Cours UML » consulté le 10 avril 2019.

[2] http://laurent-piechocki.developpez.com/uml/tutoriel/lp/cours/ « UML, le langage de modélisation objet unifié » par Laurent Piechocki. Date de publication : 22/10/07, date de mise à jour : 14/09/09, consulté le 10 avril 2019.