Conception d'applications – L3 informatique – UBO

Sujet et évaluation

L'UE « conception d'applications » consiste à réaliser une application Java par groupe de 4 étudiants en utilisant des outils et méthodes pour développer collectivement du code et produire du code de qualité. La description des outils et leur utilisation se trouve dans le document "environnement de développement".

1. Cahier des charges

L'application à réaliser gère les événements d'une association et de ses membres.

Membre : un membre d'association est caractérisé par son nom, son prénom, son âge et son adresse. Il n'existe pas deux membres ayant le même nom et le même prénom. Dans l'association, un membre joue un rôle particulier, celui de président. Il y a toujours un et un seul président.

Événement : un événement est défini par un nom, une date et une heure, une durée, un lieu et un nombre maximum de personnes y participant. Deux événements ne peuvent pas avoir lieu en même temps dans le même lieu.

Inscription : un membre peut s'inscrire à un événement. La contrainte est que le membre ne doit pas être déjà inscrit à un événement qui se chevauche dans le temps avec le nouvel événement auquel il veut s'inscrire. On ne peut également s'inscrire à un événement que s'il n'a pas déjà eu lieu et que le nombre maximal de participants n'est pas atteint. Un membre peut ensuite se désinscrire d'un événement.

Suivi : on doit pouvoir visualiser l'ensemble des membres de l'association, l'ensemble des événements (tous ou ceux à venir), l'ensemble des inscriptions pour un événement donné (avec le nombre d'inscrits) et pour chaque membre, pouvoir lister les événements auxquels il est inscrit (tous ou ceux à venir). L'application doit permettre de créer et supprimer de nouveaux membres ou événements.

Mise en œuvre : Pour faciliter la maintenance de l'application, on a décidé de faire une application Java. L'ensemble des données est stocké dans des fichiers (Cf. UE Java 2).

2. Code fournis et plan de tests

Pour vous guider dans l'implémentation de l'application, une partie du code source et des tests est fournie dans le fichier src.zip :

- Package association : code de l'application à réaliser
    - Classe InformationPersonnelle : cette classe est complète et au format attendu. Elle contient la définition des informations personnelles d'un membre.
    - Classe Evenement : classe incomplète définissant un événement. Seuls les attributs et la signature de 2 méthodes sont donnés.
    - Interface InterGestionMembres : définit les méthodes de gestion des membres (ajout et suppression de membres, gestion du président).
    - Interface InterGestionEvenements : définit les méthodes de gestion des événements (création, suppression, gestion des membres).
    - Interface InterMembre : définit les méthodes associées à un membre (gestion des informations personnelles et liste des événements du membre)
    - Interface InterAssociation : définit les méthodes de sauvegarde des données et les méthodes de gestion de l'association.
- Package tests : tests unitaires et d'intégration des classes implémentées
    - TestInformationPersonnelle : tests unitaires de la classe InformationPersonnelle
- A la racine du répertoire src
    - Classe MainAssociation qui contient une méthode main() exécutable et servira d'essai pour créer un exécutable indépendant d'Eclipse.

Toutes les classes et interfaces fournies (à l'exception de Evenement et MainAssociation) sont documentées en respectant le format d'une Javadoc.

En addition, un fichier de configuration du formatage du code d'Eclipse vous est également fourni pour valider la qualité de votre code avec Checkstyle.

Le plan de test de la classe TestInformationPersonnelle est le suivant :

- info basique est une instance de InformationPersonnelle avec uniquement un nom et un prénom défini (Luke Skywalker)
- info complète est une instance de InformationPersonnelle avec toutes ses informations définies (nom, age de 20 et une adresse non null)

Test constructeur et getter :

Méthode de test	Actions	Attendu
testAdresseNonNull	Récupérer l'adresse des deux infos (basique et complète)	Aucune adresse n'est égale à null
testConstructeur	Instancier InformationPersonnelle avec un nom et un prénom corrects mais une adresse et un âge non valides.	Une instance est créée et les getter du nom, du prénom, de l'âge et de l'adresse renvoient les valeurs attendues.

Test setter :

Méthode de test	Actions	Attendu
testAge25Basique	Appelle le setter de l'âge de l'info basique avec la valeur 25	Le getter de l'âge renvoie 25
testAgeNegatifBasique	Appelle le setter de l'âge de l'info basique avec une valeur de -20	Le getter de l'âge renvoie null
testAgeNegatifComplet	Appelle le setter de l'âge de l'info complète avec une valeur de -20	L'âge n'a pas été modifié par l'appel du setter
testSetterAdresseNull	Appelle le setter de l'adresse de l'info complète avec la valeur null	Le getter de l'adresse ne renvoie pas une adresse null

3. Attendus de l'UE et évaluation

A la fin de l'UE, il est attendu une application fonctionnelle et exécutable en dehors d'Eclipse.

3.1 Organisation en groupes

Au début de l'UE, vous inscrirez votre groupe de 4 étudiants dans le tableur en ligne (connectez vous avec vos identifiants UBO avant d'ouvrir le fichier) : https://docs.google.com/spreadsheets/d/1HPekBc19QOcaWiF6UAhZ-AhS-nc78fKxjXKAmA8B7EA/edit#gid=0

Inscrivez 4 étudiants du même (sous)groupe de TP.

3.2 Livrable 1 : Vendredi 25 Novembre

Pour le vendredi 25 Novembre à 19h, vous fournirez une archive contenant :

- Le code source du projet dans le package association : ajout des classes implémentant les interfaces, finalisation de la classe Evenement. Le code source devra être commenté en respectant le format Javadoc et être validé par le style Checkstyle.
- Les plans de tests des interfaces et de la classe Evenement (via un fichier PDF) avec leur implémentation en JUnit dans le package tests. Vous pouvez également commencer à prévoir des tests d'intégration entre les classes.
- Le HTML de la Javadoc générée pour votre code et vos tests.

Si vous n'avez pas tout terminé, envoyez votre code en l'état (le plus compréhensible possible). Le but de ce premier livrable est de vous faire un retour sur la qualité de votre code et de vos tests. Ce livrable ne sera pas noté directement (sauf un malus de points si vous ne rendez rien).

Organisation suggérée

Pour travailler en groupe sur le même projet, vous utiliserez Git. Un étudiant du groupe (le chef de projet) crée un projet Git et ajoute les 3 autres en tant que propriétaires du projet (voir la documentation technique à ce sujet).

Le problème principal que l'on rencontre avec Git est le conflit de code : quelqu'un a modifié une partie d'un fichier qui a également été modifiée par quelqu'un d'autre. Pour limiter au plus les conflits, vous pouvez chacun travailler sur des classes distinctes : il y a plusieurs classes à implémenter par rapport aux 4 interfaces et la classe Evenement à compléter, vous avez de quoi travailler chacun sur des classes séparées. Vous pouvez vous répartir la responsabilité de la gestion des classes : seul le responsable de la classe fait le *push* sur sa classe.

Communication

Au-delà des séances de TP, vous pourrez communiquer avec vos enseignants via un serveur Discord : https://discord.gg/qsJmKB2A

Il est recommandé également que vous créiez un salon Discord pour votre groupe de travail. La communication est un point important dans la bonne réalisation d'un projet.

3.3 Livrable 2 : dernier créneau de TP (mercredi 7 ou jeudi 8 Décembre)

L'évaluation finale de votre projet aura lieu lors de la dernière séance de TP pour chaque groupe. Vous fournirez à ce moment-là l'exécutable de votre projet ainsi que toutes les sources et tests. Vous définirez des tests d'intégration entre les classes avec les plans de tests.

Interface utilisateur

Dans cette deuxième version de votre code, vous rajouterez une interface utilisateur pour interagir avec les classes déjà implémentées et gérer les données (membres, événements, …).

Au choix, selon l'avancement de votre projet, vous intégrerez :

- Une interface graphique que l'on vous fournira via du code écrit en Java FX
- Une interface textuelle où l'utilisateur tape les informations et les commandes dans la console

Un bonus sera donné si vous optez pour la version graphique.

Une évaluation sur machine aura lieu : l'enseignant lancera votre application et l'utilisera pour en vérifier le bon fonctionnement.

Voici les scénarios (ou “user stories”) qui devront être validés (vous pouvez/devez bien entendu en prévoir d’autres) :

Scénario 1 : création et modification d’un membre

- L’utilisateur charge des données préalablement enregistrées et affiche la liste des membres.
- L’utilisateur clique sur le bouton pour créer un nouveau membre, il remplit tous les champs en mettant des champs valides puis clique sur Valider.
- L’application l’informe que le membre a été correctement créé (on l'appellera par la suite Membre1).
- L’utilisateur clique sur le bouton pour créer un nouveau membre, il remplit tous les champs avec les mêmes valeurs de nom et de prénom que Membre1 mais un âge et une adresse différents, puis clique sur Valider.
- L’application l’informe que le membre existant a bien été modifié.

Scénario 2 : création d’un événement

- L’utilisateur clique sur le bouton pour créer un nouvel événement , il remplit tous les champs correctement avec 10 participants maximum et clique sur Valider.
- L’application l’informe que l’événement a été correctement créé (on l’appellera par la suite Evt1).
- L’utilisateur clique sur le bouton pour créer un nouvel événement, il remplit tous les champs en précisant les mêmes heures mais un lieu différent que pour l’événement précédent puis clique sur Valider.
- L’application l’informe que l’événement a été correctement créé (on l’appellera par la suite Evt2).
- L’utilisateur clique sur le bouton pour créer un nouvel événement, il remplit tous les champs en précisant la même heure (légèrement décalée) et le même lieu que pour l’événement précédent puis clique sur Valider.
- L’application l’informe que l’événement n’a pas été créé pour cause de conflit.
- L’utilisateur affiche tous les événements à venir de l'association et voit bien les deux événements Evt1 et Evt2.

Scénario 3 : inscription d’un membre à un événement

- L’utilisateur affiche la liste des événements à venir pour l’association.
- Il sélectionne Evt1 et Membre1 puis clique sur “inscrire membre à événement”.
- L’application lui dit que le membre a bien été inscrit à l’événement.
- L’utilisateur affiche la liste de tous les événements de l’association. Il sélectionne un événement passé et constate qu’il ne peut pas ajouter de membre.
- L’utilisateur affiche la liste des événements de l’association, il sélectionne Evt2 et inscrit Membre1.
- L’application l’informe que l’inscription ne peut pas se faire à cause d’un conflit temporel.