Rapport de développement

Ange Leyrit | Océane Pezennec | Hippolyte Jean | Adil Zaim (Dragons)

08 décembre 2022

Université de Bretagne Occidentale

FACULTÉ
DES SCIENCES
& TECHNIQUES

Le Contexte

Dans le cadre de l'Unité d'Enseignement Conceptions d'Application, nous avons été amener à développer une application executable en Java permettant à une association de gérer ses membres, ses événements et l'inscription de ses membres à un événements.

Un cahier des charges, des users stories et des test JUNit ont été fournies/demander pour encadrer le développement et donner une direction vers le rendu final de cette application Java.

Les nouveautés dans ce projet furent nombreuses et enrichissantes, que ce soit l'utilisation d'un gitlab pour organiser le code, de checkstyle pour le formatage ou bien des Test Junit pour tester le comportement des méthodes implémenter.

Etat Global du développement:

Les classes et méthodes utilisées par l'application sont complètes et fonctionnelles, à propos de L'IHM, les boutons sont tous utilisables et les zone d'affichage de texte affiche les messages attendues, il est également possible de sauvegarder/charger les données du gestionnaire d'association.

Changements apportés depuis évaluation:

-Ange Javadoc Classe Association et classe Controlleur, description du test (dans ce fichier) testSaveAndLoad

-Adil Javadoc méthode "ajoutermembre" de la classe "GestionMembre" Bien fait plus le test associé "testajouterMembre" dans" testGestionMembres"+plans test complet pour cette méthode.

-Hippolyte Modification de la javadoc de la méthode "creerEvenement" de "GestionEvenements" et suppression d'un bout de code commenté. Ajout du test unitaire "testInscriptionParticipants" à "TestGestionEvenements" et de son plan de test. Ajout de la javadoc pour les méthodes "equals", "hashcode", et "toString" de la classe Evenement.

User stories pour test Interface:

Scénario 1 : création et modification d’un membre (validé par test professeur)

- L’utilisateur charge des données préalablement enregistrées et affiche la liste des membres. -> OK
- 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. -> OK
- L’application l’informe que le membre a été correctement créé (on l'appellera par la suite Membre1). -> OK
- 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 différente, puis clique sur Valider. -> OK
- L’application l’informe que le membre existant a bien été modifié. -> OK
- L’utilisateur essaye d’ajouter un membre mais ne remplit pas le nom ou/et le prénom et clique sur Valider, un message d’erreur indiquant de correctement remplir les champs s’affiche -> OK

Scénario 2 : création d’un événement(validé par test professeur mais problème affichage message confirmation/erreurs)

- 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. -> OK
- L’application l’informe que l’événement a été correctement créé (on l’appellera par la suite Evt1). -> OK
- 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. -> OK
- L’application l’informe que l’événement a été correctement créé (on l’appellera par la suite Evt2). -> OK
- 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. -> OK
- L’application l’informe que l’événement n’a pas été créé pour cause de conflit. -> OK
- L’utilisateur affiche tous les événements à venir de l'association et voit bien les deux événements Evt1 et Evt2. -> OK

Scénario 3 : inscription d’un membre à un événement

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

Scénario 4 : désinscription d’un participant à un événement

- L’utilisateur affiche la liste des événements à venir de l’association, ainsi que la liste de ses membres.
- Il sélectionne Evt1 et Membre1 dans leur liste respective, puis clic sur “désinscrire un membre” pour valider.
- L’application indique que le membre a bien été désinscrit de l’événement.
- Afficher la liste des événements auxquels participent Membre1 ne contient plus Event1, et inversement.
- L’utilisateur affiche les listes des événements et membre de l’association. Il sélectionne à nouveau Membre 1 et Evt1 pour le désinscrire. Membre1 n’est plus participant, et l’application affiche donc un message d’erreur. ->

Scénario 5 : suppression d’un membre de l’association

- L’utilisateur affiche la liste des membres à de l’association en cliquant sur “affiche tous les membres”.
- Il sélectionne le membre a supprimer dans leur liste respective, puis clic sur “supprimer”.
- L’application lui dit que le membre a bien été supprimer.
  puis il clique sur “affiche tous les membres” pour visualiser la suppression du membre

Classe GestionEvenement:

Plan des test :

- gestionnaireVide est une instance de base de GestionEvenement, ne comportant aucun événement.
- gestionnaireClassique est une instance de GEstionEvenement avec quelques événement prédéfinis.
- evt1, evt2, et evt3 sont des événements normaux et valides servant aux tests et à l'initialisation de gestionnaireClassique.

Méthode de test	Actions	Attendu
testSetterListeEvenement	Insère une nouvelle liste d'événements à la place de la liste des événements courante d'un gestionnaire	La liste stockée dans l'instance de GestionEvenements est bien celle insérée, et non une fusion des deux listes ou l'ancienne
testCreationEvenement	Créée de nouveau événement à partir de la liste des événements de l'instance de GestionObjet via la méthode creerEvenement	Seul deux Evenement sur 3 sont créés, le troisième se chevauchant sur le premier vaut null
testListeEvenementNonNull	Vérifie que la liste des événement ne peut être nulle	La liste des événement de l'instance est une liste vide
testGetterEvenementAvenir	Appelle la méthode ensembleEvenementAvenir()	Vérifie que la liste obtenue ne contient pas les événements passés de la liste des événements d'un gestionnaire d'événements
testSuppressionEvenement	Appelle testSuppressionEvenement()	Vérifie que la liste des événement ne contient plus les événement supprimé, et qu'il n'y a pas de problème lors de la suppression d'un événement qui n'y figure pas
testInscriptionParticipants	Essaie d'inscrire un participant mbr à un	mbr n'est inscrit qu'aux événement event1 et event3

	événement event1, puis à un événement event2 ayant lieu sur le même créneaux horaire que event1, puis à event3 ayant lieu juste après event1	
testGestionParticipants	Test les méthodes inscriptionEvenement() et annulerEvenement()	Les membres ne peuvent pas être inscrit à un événement passé, et s'ils sont inscrit leur attribut listant les événement auxquels ils sont inscrit est bien modifié, et ils peuvent se désinscrire

Etat de développement:

Sensée être complète, mais il subsiste un problème dans testGestionParticipants, où l'inscription d'un membre à un événement ne semble pas incrémenter la taille de l'ensemble des événements auxquels ce dernier est inscrit.

Classe Evenement:

Plan des test :

- eventClassique est une instance de base de Evenement, ne comportant aucun participant.

Méthode de test	Actions	Attendu
testChangementDate	Modifie la date d'un Evenement (eventClasssique)	La modification a bien eu lieu
testChangementDuree	Modifie la durée d'un Evenement (eventClasssique)	La modification a bien eu lieu
testLieuNonNull	Récupère le lieu d'un Evenement (eventClassique)	Le lieu est différent de null.
testSetterLieuNull	Essaie de remplacer le lieu d'un Evenement	Le lieu est différent de null.

	(eventClassique) par null	
testNomNonNull	Récupère le nom d'un Evenement (eventClassique)	Le lieu est différent de null.
testSetterNomNull	Essaie de remplacer le nom d'un Evenement (eventClassique) par null	Le nom est différent de null.
testConstructeur	Vérifie que l'instanciation d'un Evenement se fait correctement, même avec des paramètres non valide	L'instance créée est composée de champs respectant une valeur attendue.
testChevauchement	Instancie 4 Evenements dont la date et le lieu diffèrent, et vérifie s'ils ont lieu en même temps et au même endroit, ou juste en même temps, ou à des moment différents	
testManipulationParticipants	Vérifie les méthodes d'ajout et retrait d'une instance d'objet implémentant InterMembre à la liste des participant	

Etat de développement:

Classe normalement complète.

Classe Association:

Plan des test :

- assoBefore est une instance d'Association qui sera initialisée et ensuite sauvegardée dans un fichier.
- assoAfter est une instance d'Association dont les attributs récupèrent les données du chargement du fichier précédemment créé.

Méthode de test	Actions	Attendu
testSaveAndLoad	Un premier gestionnaire d'Association va être créé (AssoBefore) dans lequel des données (membres et événements via leurs gestionnaires respectifs) vont être ajouter. Ensuite une sauvegarde dans le fichier save_asso.ser va être effectuer (via la méthode sauvegarderDonnées). ET pour finir ce même fichier sera chargé dans l'instance assoAfter(via la méthode chargerDonnées).	Les test va vérifier l'égalité (en utilisant les méthode equals de toute les classes) entre les deux instances, ce ne sont donc pas les objet eux même qui sont comparer mais leur contenu. Vrai est retourner si l'égalité est respecter, faux sinon.

Etat de développement:

La classe est complète, elle possède un constructeur faisant appel aux méthodes de l'interface InterGestionAssociation afin d'initialiser une instance possédant un attribut eventAssociation de classe GestionEvenement et un attribut memberAssociation de la classe GestionMembres.

Elle possède également une méthode permettant de sauvegarder une des ces instance dans un fichiers et inversement une méthode permettant de charger une instance sauvegarder dans un fichier

Les getters setters et autres méthodes classiques sont implémentés.

Classe GestionMembre:

Plan des test :

- Inf1,inf2,inf3,inf4,inf5,inf6 sont des instance de la class InformationPersonnelle
- inf_mis_a_jour3, est une instance informationpersonnelle avec même nom , prenom que l'instance inf3.
- m1,m2,m3,m4,m5,m6, sont des instance de la class Member
- m_like3 est une instance de Membre avec le même Information que l'instance m3
- m_mis_a_jour3 une instance membre avec même nom et prenom que m3 mais avec different age et adresse.
- la class membre a besoin des instance de la classe InformationPersonnelle ses membres créé par la suit , la Class GestionMembre permit de faire la gestion de ses membre dans une liste

Méthode de test	Actions	Attendu
testajouterMembre	ajoute de 5+1 membres dans l'association le 1 ajouter est une dans un temps la même instance déjà existant et dans un deuxième temps est une instance différente avec même informations.	le nombre de membre dans la liste de l'association et 5
testajouterMembre	Met à jour l'adresse et l'âge d'un membre déjà existant parmi les 5 avec une instance différente de celle existant déjà.	Le membre n'est pas ajouté mais l'adresse et l'âge de ce membre est mis à jour + le nombre de membres reste à 5.
testsupprimerMembre	état initial->4 membre dans l'association →suppression du membre m5 dans la liste de l'association	Le nombre des membres restant dans l'association est 3.
testdesignerPresident	désigner le membre m4 comme président de l'association	m4 est président de l'association

Méthode de test	Actions	Attendu
testajouterMembre	ajoute de 5+1 membres dans l'association le 1 ajouter est une dans un temps la même instance déjà existant et dans un deuxième temps est une instance différente avec même informations.	le nombre de membre dans la liste de l'association et 5
testpresidentNull	pas désigner le président de l'association	le président est null
testAjoutPresidentNonMembre	état initial-> 5 membres dans l'association avec 1 président. designer m6 président dont ce berner n'est pas membre dans l'association.	m1 rest toujour le président de l'association
testensembleMembres	ajouter de 6 membres dans l'association avec un ajout dont la valeur est null	le taille de la liste des membres retourner est de 6

Etat de développement:

La classe GestionMembres est finie et implémente les méthodes de l'interface GestionMembres.
Dans son constructeur, le class instancie une liste de membres.

La méthode ajouterMembre permet l'ajout du membre dans l'association ou mettre à jour son adresse et son âge si il existe déjà .

La méthode supprimerMembre permet de supprimer un membre dans l'association.

La méthode designerPresident permet de désigner le président de l'association.

La méthode ensembleMembres est un getter permet de renvoyer la liste des membres de l'association.

La méthode president est un getter permet de renvoyer le president de l'association.

Il y a une méthode HashCode, toString et equals.

Classe Membre:

Plan des test :

- infoComplete est une instance de Membre qui contient les informations du membre.
- memb est une instance de Membre qui contient la liste des évènements pour un membre.
- evtsAvenir est une instance de Membre qui contient la liste des évènements à venir pour un membre.

Constructeur et getter

Méthode de test	Actions	Attendu
testConstructeur	Instancier Membre avec des informations personnelles valides.	Une instance est créée et les Getter des informations personnelles renvoient les valeurs attendues. La liste d'événements est vide.
testGetInformationPersonnelle	Appelle le getter des informations du membre.	Le getter des informations renvoie null.
testensembleEvénements	Appelle le getter de la liste des évènements du membre. on ajoute un évènement,on vérifie que l'évènement est présent. Ensuite, on retire l'évènement et on vérifie que l'évènement n'est plus présent.	Le getter de la liste des évènements renvoie null.

setter

Méthode de test	Actions	Attendu
testDefinirInformationPersonnelle	Appelle le setter des informations du membre.	Le setter des informations renvoie null.

Etat de développement:

La classe Membre est finie(normalement) et implémente les méthodes de l'interface InterMembre.
Dans son constructeur, la classe fait appel à la classe InformationsPersonnelles.

Une surcharge du constructeur fait appel dans ses paramètres la classe InformationsPersonnelles et Evenement.

Les constructeurs contiennent l'attribut info qui récupère les informations personnelles du membre, l'attribut evt qui récupère l'ensemble des évènements auquel le membre est inscrit, l'attribut evtAvenir qui récupère la liste des évènements qui ne sont pas encore passés .

La méthode DefinirInformationPersonnelle est un getter qui permet de définir les informations d'un membre.

La méthode testGetInformationPersonnelle est un setter qui permet de récupérer les informations d'un membre.

Il y a une méthode HashCode, toString et equals.

Classe InformationPersonelle:

Plan des test :

- info basique est une instance de InformationPersonnelle avec uniquement un nom et un prénom définis (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	Une instance est créée et les getter du nom, du prénom, de

	non valides.	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

Etat de développement:

Classe complétée, aucune modification à apporter.

Outils utiliser pour le developpement:

- Eclipse IDE : https://www.eclipse.org/ide/
- JavaFX : https://docs.oracle.com/javase/8/javase-clienttechnologies.htm
- Junit : https://junit.org/junit5/
- CheckStyle : https://checkstyle.org/
- Gitlab : https://about.gitlab.com/
- Git : https://git-scm.com/
- Fork : https://git-fork.com/