Projet JAVA 2022

Gestion des sorties d'un club de cyclistes

Huygebaert Gabriel

Énoncé

Le client est un club de cyclistes qui aimerait s'informatiser.

Voici la façon dont ce club fonctionne :

Le club contient en son sein des amateurs de

- VTT
  - Des descendeurs
  - Des randonneurs
  - Des « trialistes »
- Cyclo (vélo sur route)

Chaque membre du club doit appartenir à au moins une des catégories.

Le club participe régulièrement à des balades. Dans ce cas, il faut se déplacer vers le lieu de départ d'une de ces balades. Afin d'organiser au mieux le covoiturage, une application sera mise au point afin d'optimiser l'offre et la demande : Chaque membre ayant la possibilité de transporter un (ou plusieurs) vélo(s) et / ou membre(s) du club « postera » la (ou les) place(s) disponible(s), tant pour les vélos que pour les membres.

Chaque membre « authentifié » aura donc la possibilité de poster ses disponibilités ou de réserver de la place pour lui-même et / ou son vélo.

Pour chaque catégorie, un « responsable » sera désigné afin d'organiser (et publier) le calendrier des sorties pour cette catégorie.

Lors de chaque balade, le départ du covoiturage sera organisé au départ de l'adresse du club. Afin de ne pas léser les personnes mettant à disposition leur véhicule, un « forfait » sera calculé lors de chaque déplacement par le « responsable » de catégorie.
Le suivi des remboursements vers les « chauffeurs » et le paiement des « passagers » sera assuré par le trésorier du club.

Pour chaque balade, un récapitulatif des disponibilités sera indiqué, afin de savoir s'il manque des chauffeurs ou si au contraire certains chauffeurs sont superflus.

Afin d'être en ordre d'inscription, chaque membre devra s'acquitter d'une cotisation annuelle de 20 EURO. S'il désire s'inscrire dans d'autres catégories, un supplément de 5 EURO par catégorie supplémentaire lui sera demandé.

Le trésorier du club est chargé de s'assurer que les membres ont bien payé leur(s) cotisation(s)

Il vous est demandé de modéliser cette application à l'aide des notations UML suivantes :

- Cas d'utilisation (use cases).
- Diagramme de classes.
- Diagrammes de séquences

Le programme implémentant la modélisation UML sera écrit en Java. Les données seront sauvegardées dans une base de données Oracle (localisée sur le serveur Oracle de l'école) ou Access (Tout doit être inclus dans le programme pour son exécution). Vous utiliserez le pattern DAO et WindowBuilder pour la partie graphique.

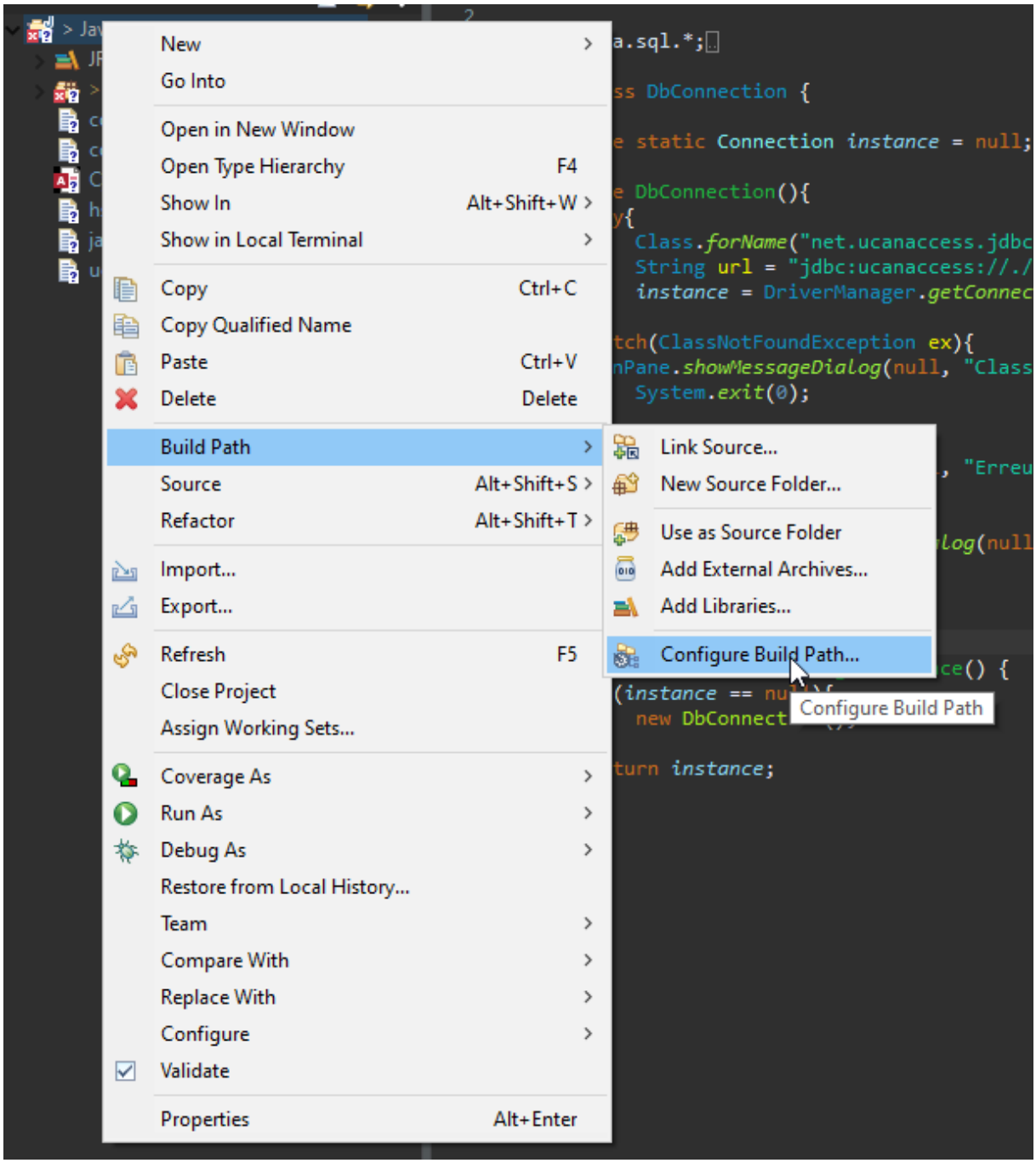
Le lundi 29 novembre 2021 entre 8h et 12h sur Moodle, vous devrez remettre dans un dossier compressé à votre nom :

- Le rapport, **en format pdf**, comprenant la modélisation UML (use cases, diagrammes de classes, diagrammes de séquence et explications), l'explication de l'implémentation du pgm Java et la manière d'accéder à l'application en indiquant des usernames si nécessaire. Ajoutez également le lien Github dans le rapport.
- Le code Java : donner le workspace de votre application. Tout doit être prévu dans le programme pour son exécution. Testez votre programme sur plusieurs Pcs différents.

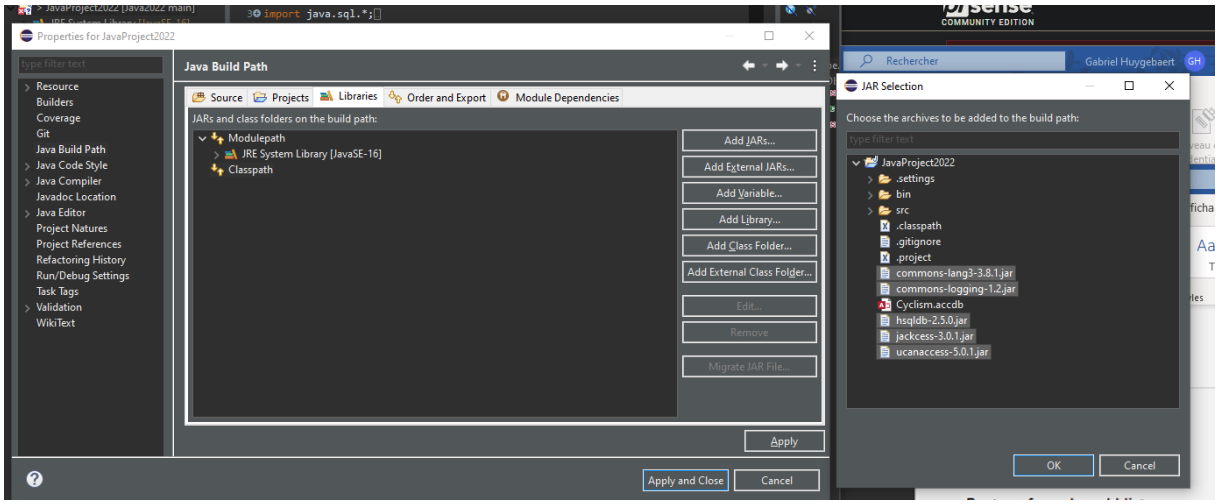
Table des matières

Façon d'utiliser le programme

Avoir accès à la base de données access



Librairies -> Class path -> Add JARs -> Sélectionner les 5 fichiers qui sont déjà dans la racine du projet

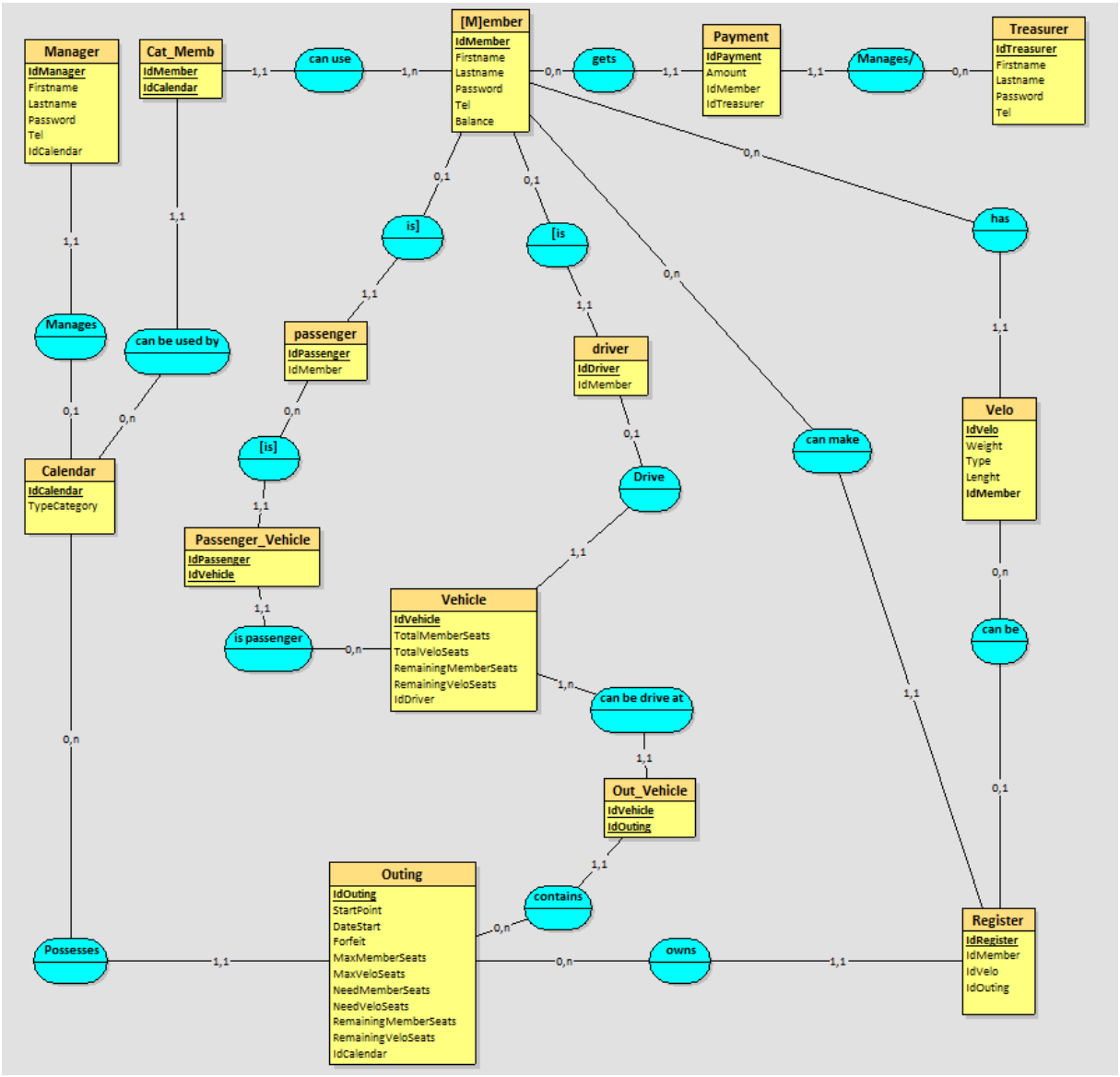


Lancer le programme via ‘Init’.

Lien Git

https://github.com/Usagi08/Java2022

Base de données



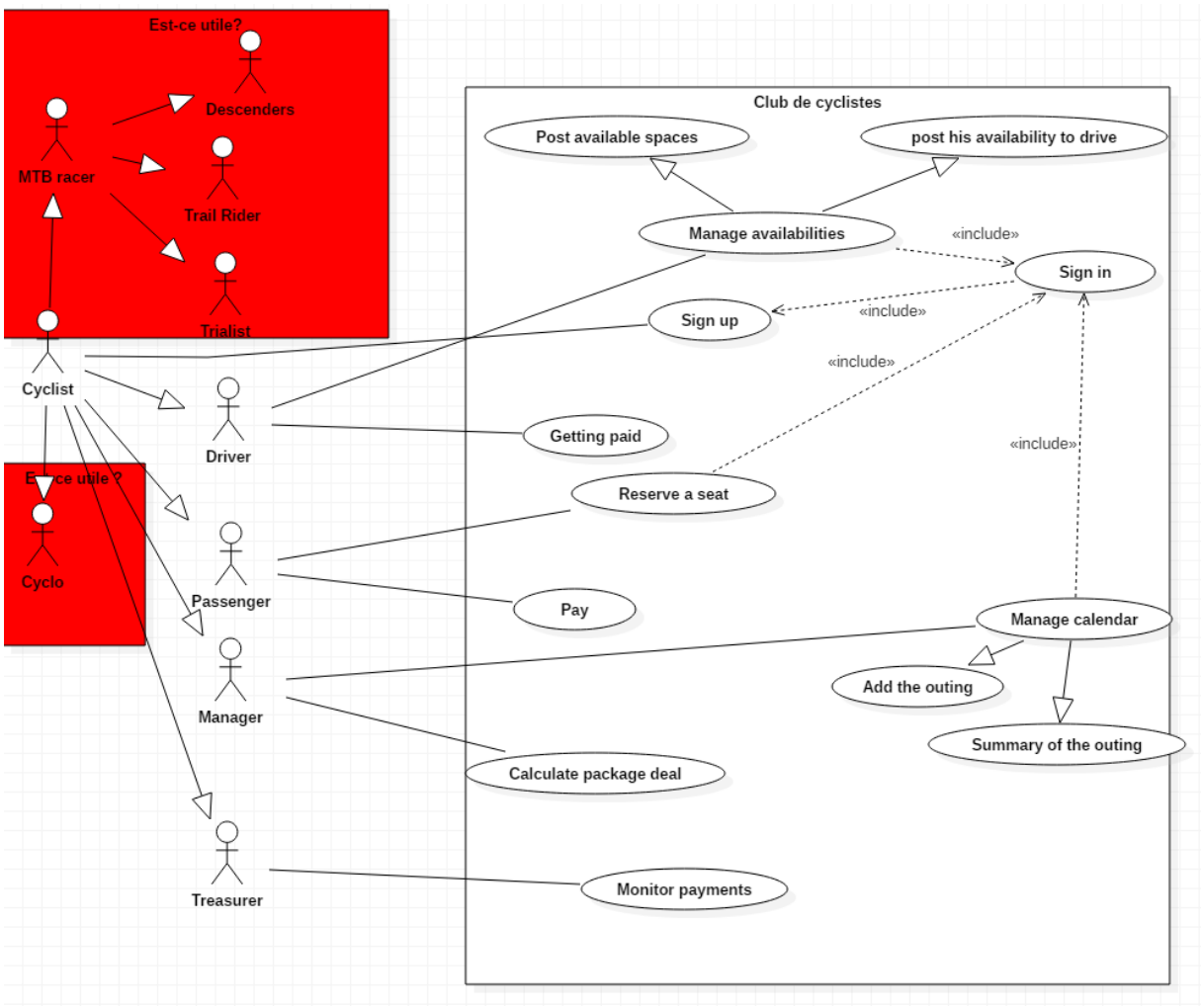
Commentaire

Une table catégorie et Calendrier étaient en lien 1-1 => Fusion des tables. C'est pourquoi le type de la catégorie apparaît dans la table Calendrier.
Si Calendrier(catégorie) – Manager avaient conservé le lien 1-1, une fusion aurait été également envisagée.

Analyse

Use case diagram

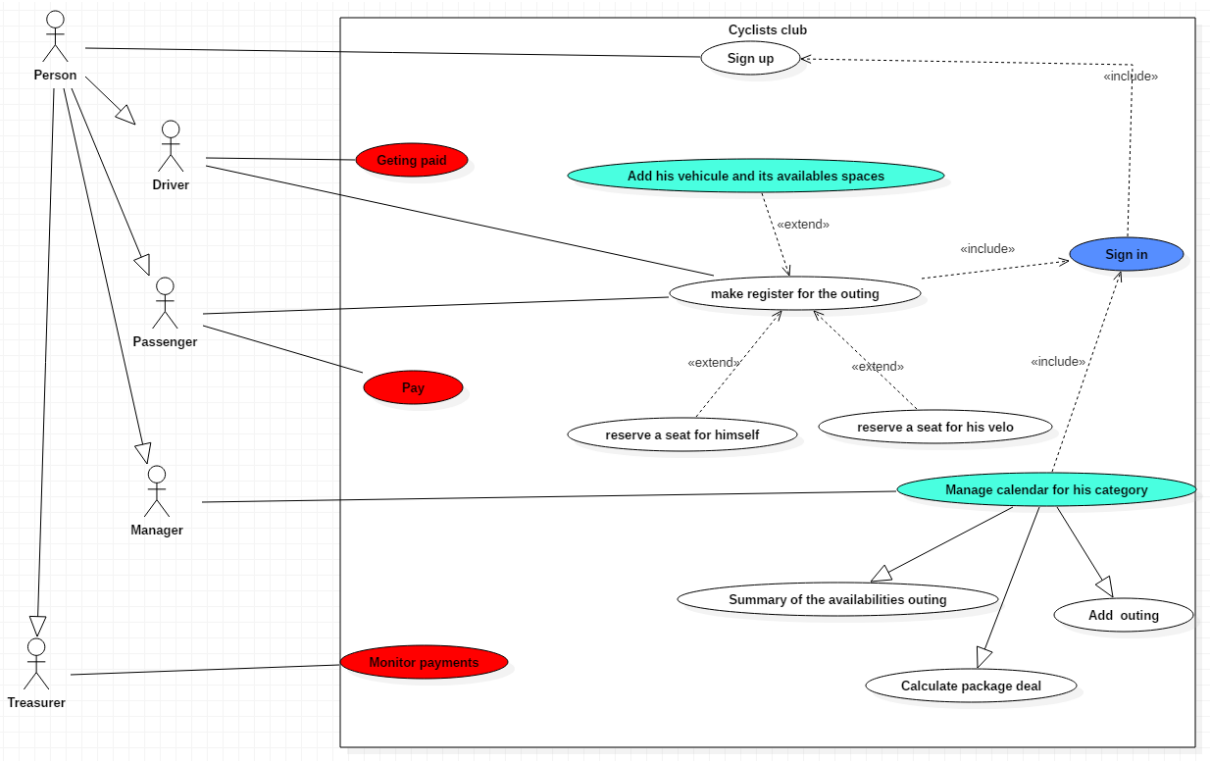
Jet 1



Commentaire

Les acteurs dont je doutais l'utilité ne devaient pas exister sur ce diagramme.

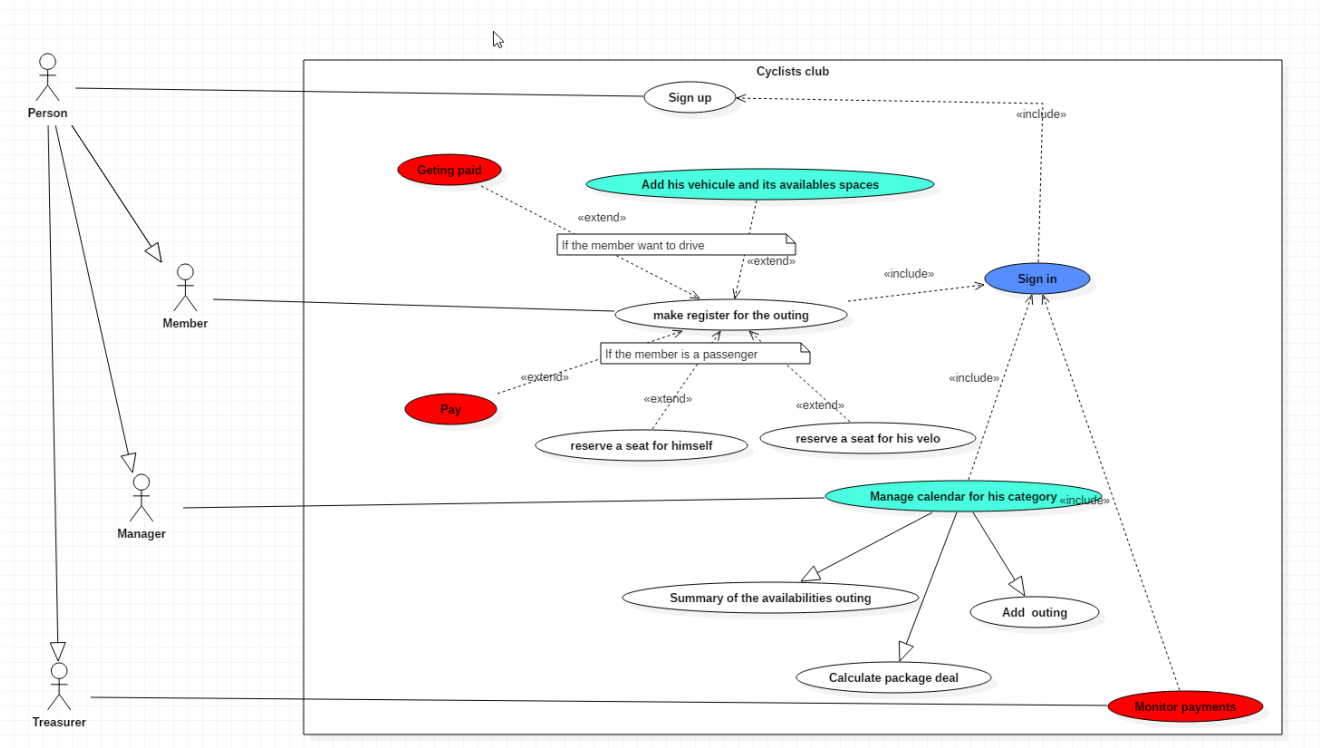
Jet 2



Commentaire

Bien que le conducteur et le passager soient deux acteurs dans le use case, cela ne veut pas dire qu'ils seront caractérisés par deux classes distinctes dans le diagramme de classe. Ces derniers sont des membres et ce sera leur relation avec le véhicule qui les différenciera.

Jet 3



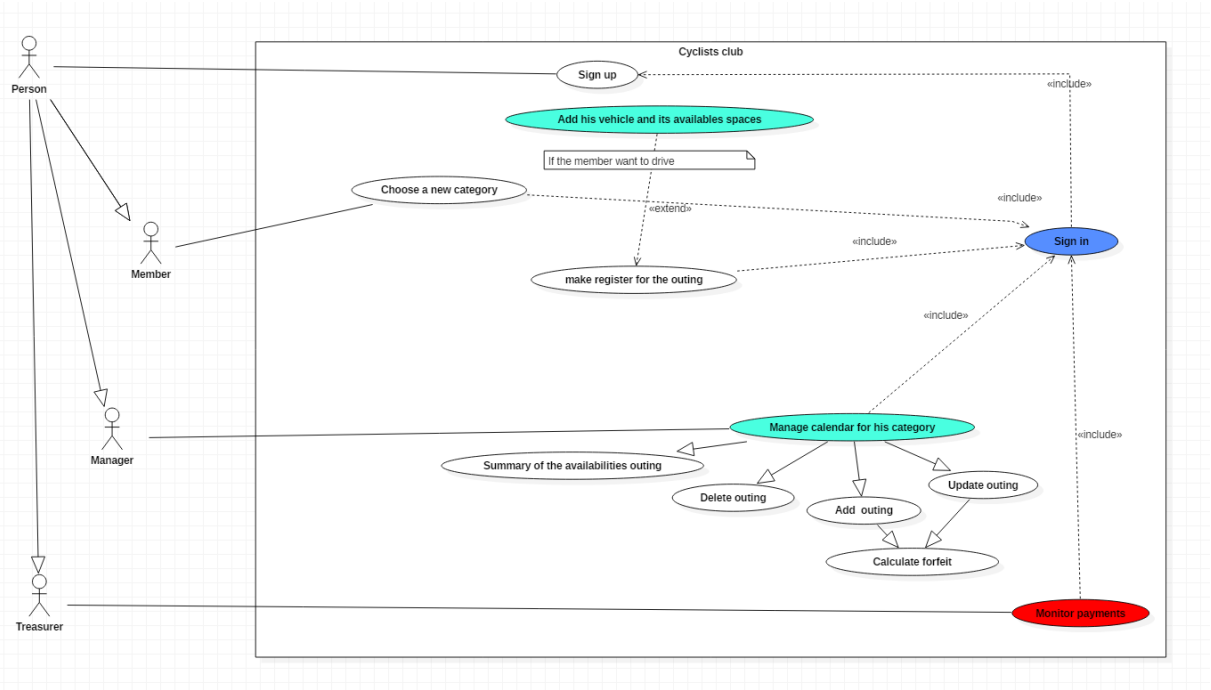
Commentaire

C'était en réalité un non-sens de séparer le passager et le conducteur. Ces deux sont des membres et il était possible de gérer leur relation envers le véhicule grâce à des cas d'utilisations optionnels.

De plus, pour payer et être payé, il faudrait avoir participé à la sortie en tant que conducteur OU passager, donc avoir été connecté.

De même pour le trésorier qui pouvait avoir accès aux paiements effectués ou non sans être connecté.

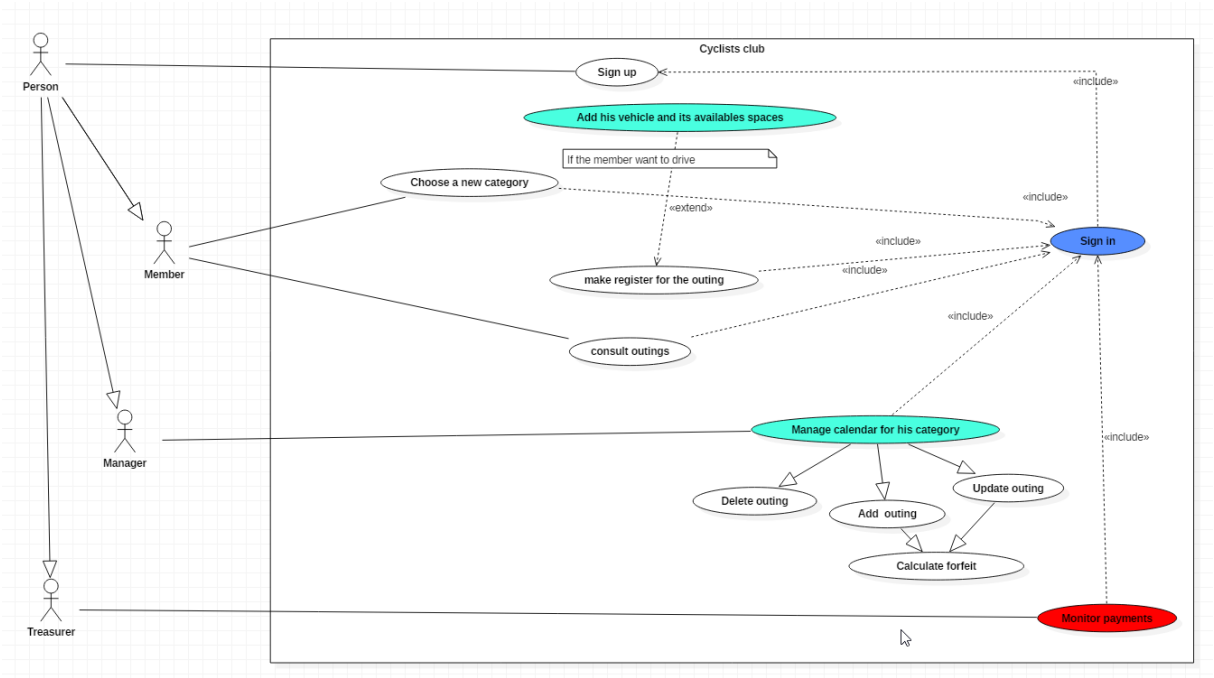
Jet 4



Commentaire

Après réflexion, le programme ne gérera pas les paiements des utilisateurs. Il s'agit simplement d'une application de gestion des sorties. De plus, j'avais oublié de permettre aux membres de choisir une nouvelle catégorie. (Lien 1-plusieurs dans le diagramme de class)
Le forfait est calculé lorsque le manager modifie ou ajoute une sortie.
Grâce à la class registration, le choix d'ajouter ou non son vélo, être ou non un passager peut faire au moment de l'inscription à la sortie.

Jet 5

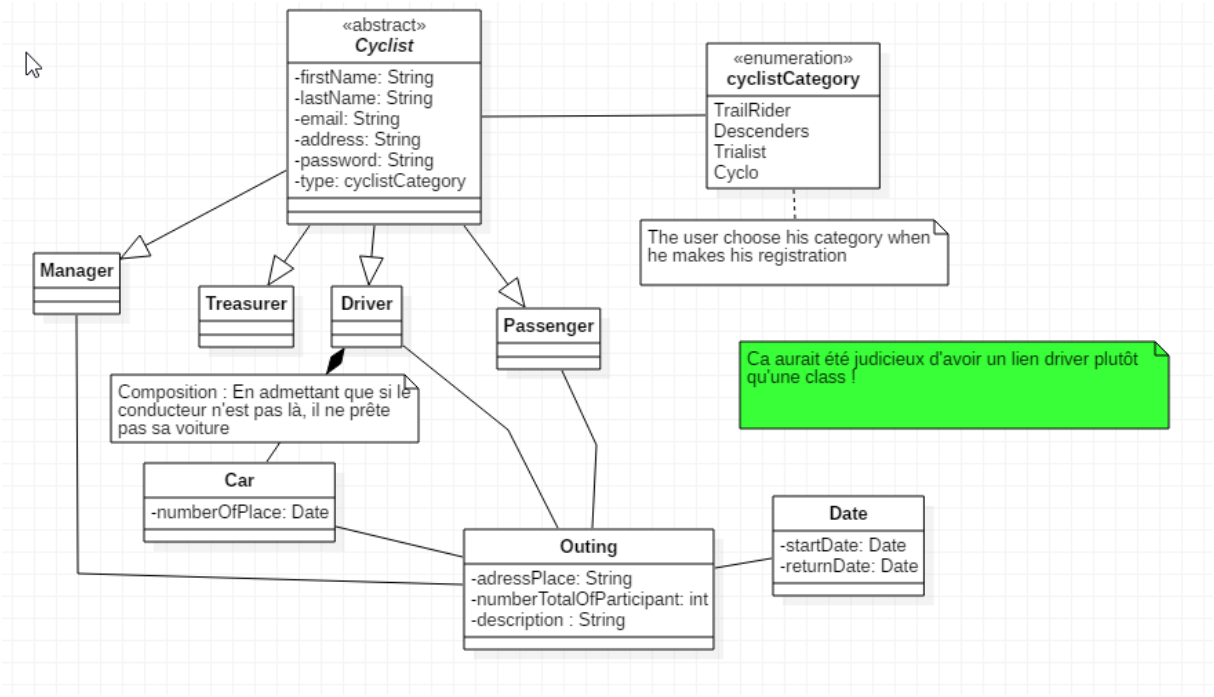


Commentaire

Oubli : consulter les sorties pour les membres.
Suppression du cas « résumer les places ».

Class diagram

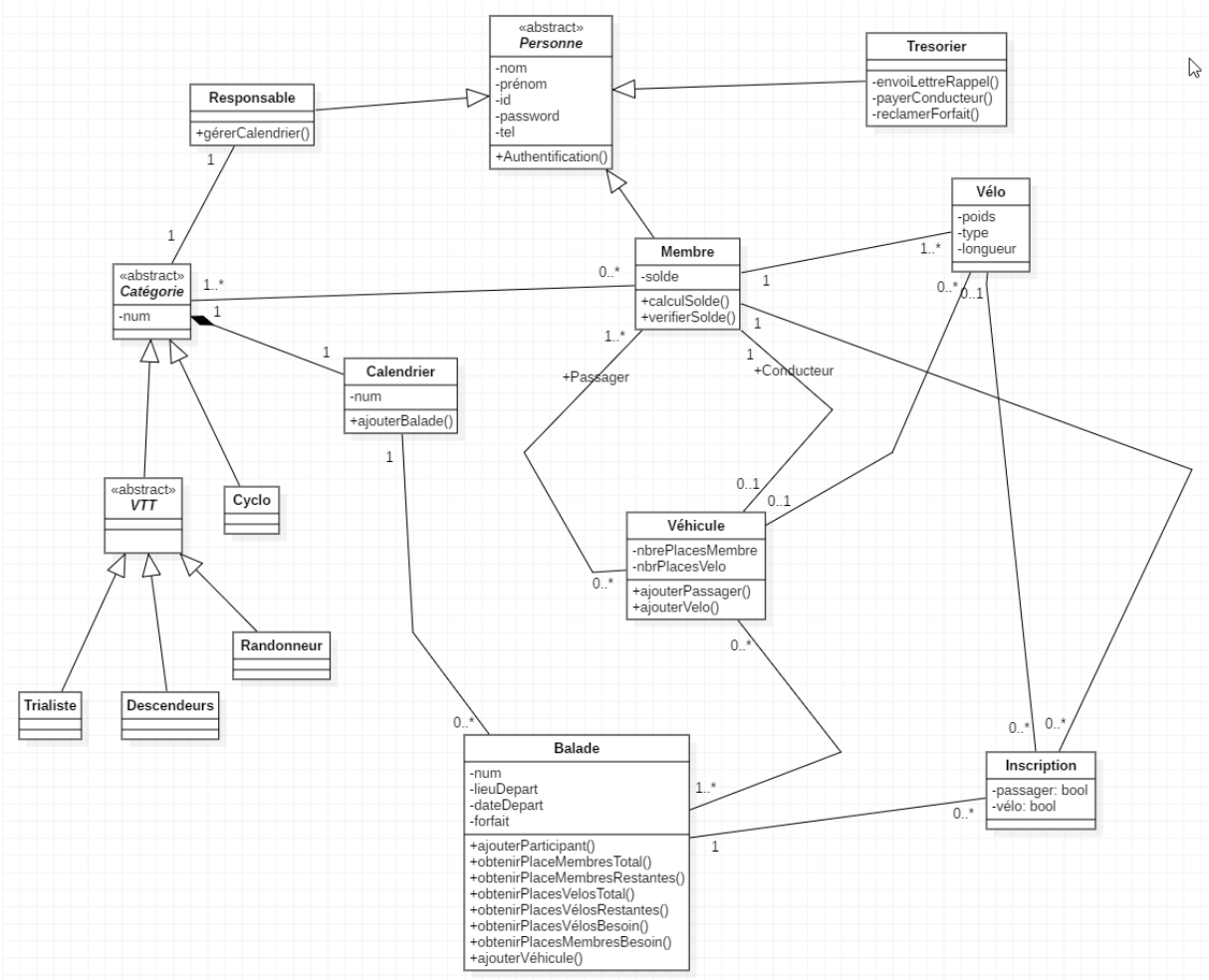
Jet 1



Commentaire

Mauvaise perception de ce qui était demandé. Il aurait été plus judicieux d'avoir des liens nommés plutôt que de créer deux class (Passager/conducteur). J'ignorais que c'était possible de faire cela.

Jet 2 (base proposée par l'enseignant)



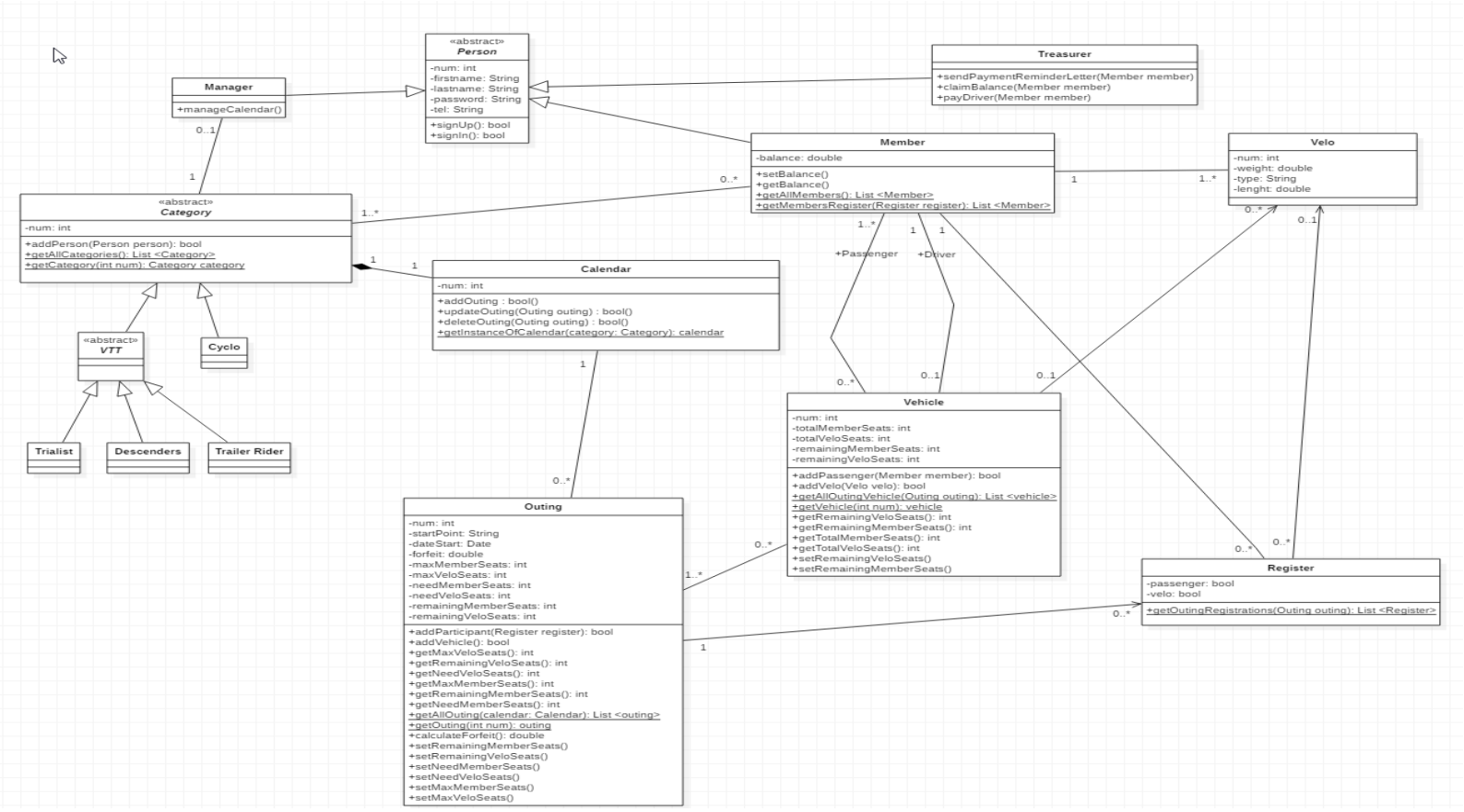
Commentaire

Pas de sortie en tandem. Un vélo = une personne. Le conducteur n'a qu'un seul ou aucun véhicule.
Le calendrier n'existe pas si une catégorie n'existe pas (prévoir que le calendrier peut être supprimé si la catégorie est supprimée).
Le forfait pourrait être un total pour tout le monde à diviser et payer ou un forfait individuel.
Le participant n'est pas obligé de prendre une place dans le véhicule.

- ⇨ Quand il s'inscrit en tant que participant, il doit alors préciser s'il a besoin ou non d'une place pour le vélo ou lui-même.

Dans l'héritage, s'il n'y a pas de méthodes ou d'attributs en particulier dans les class filles, il est possible de remplacer par un énumérateur. (Cf. les catégories)

Jet 3



Commentaire

Une inscription doit savoir quel membre et/ou quel vélo inscrire à la sortie (via addParticipant qui prend un register). L'inverse n'est, en revanche, pas nécessaire. Pas besoin de savoir à quelle inscription un vélo se trouve.
Pas la peine de savoir dans quel véhicule se trouve untel vélo, je sais quels vélos se trouvent dans unetelle voiture.

Modification de la multiplicité entre Manager et Category.
Ce que j'aurais fait pour respecter le lien 1-1 : il aurait fallu implémenter dans le constructeur du manager l'instanciation de la catégorie (La catégorie aurait été responsable de son calendrier et le manager de sa catégorie. Instanciation en cascade : Manager -> Catégorie -> Calendrier).

En pratique : nos catégories existent déjà sous forme de class qu'il faudra instancier nous-même (PAS l'utilisateur. Aucun user ne doit pouvoir ajouter/modifier/supprimer une catégorie).
Ainsi, chaque responsable dans la réalité aurait reçu son mot de passe personnel puisque les comptes auraient déjà été créés à l'avance. Rien n'empêchait la modification de ce manager par la suite, par le développeur.

J'ai vu la chose différemment :

Argument 1 : Je n'ai aucune idée de « QUI » sera le manager d'une catégorie.

Par conséquent, j'ai décidé de laisser la liberté aux futurs managers de créer leur propre compte et de choisir la catégorie et restreindre la création d'un seul compte par catégorie. C'est pourquoi une catégorie pourrait se retrouver à un moment donné, sans manager. Sans manager, il n'est pas possible de créer de sortie et donc aucun intérêt pour les membres d'être inscrit.
Cela n'a pas plus d'intérêt d'avoir un manager qui ne poste pas de sortie.

Argument 2 : l'échange de mot de passe.

Ecrire un mot de passe sur un bout de papier pose un problème en matière de sécurité. Les utilisateurs le laissent parfois traîner. Au départ, il n'est pas envisagé pour un utilisateur de changer son mot de passe dans le programme (à voir pour amélioration).

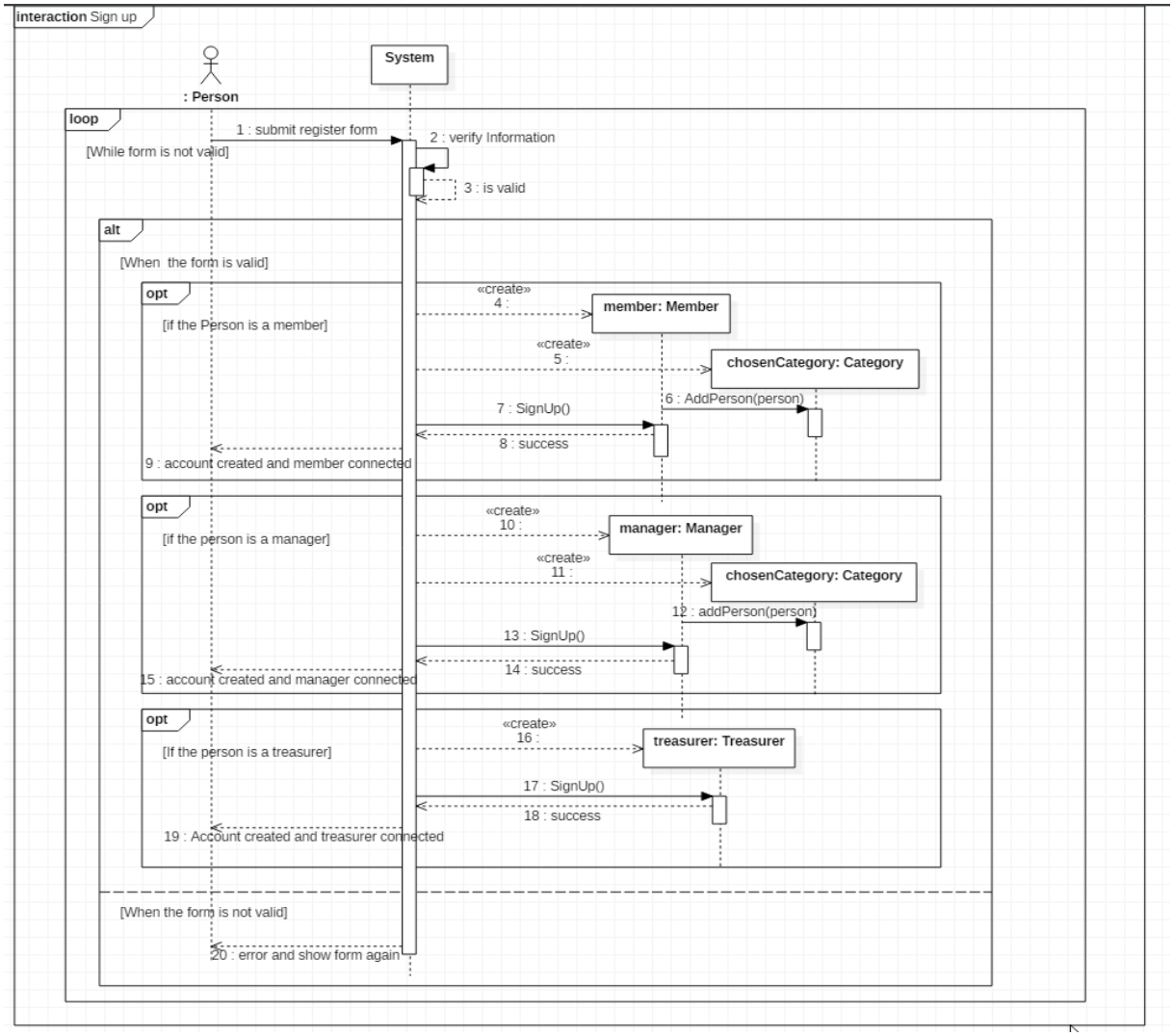
(tout comme rien ne m'indique qu'un utilisateur n'écrirait pas son mot de passe choisi sur un papier… Mais cela réduit les « risques ».)

Si le choix avait été l'envoi par e-mail, cela fait quelques démarches pour un détail.

Dans un cadre réaliste, cette adaptation aurait bien entendu nécessité discussion avec l'analyste et/ou le client.

Sequence diagram

Sign up



Commentaire

À la création des comptes utilisateurs « member » et « manager », le choix d'une catégorie est imposé. En effet, dans le diagramme de classe, les liens qui unissent les classes signifient que la personne doit obligatoirement avoir choisi sa catégorie à sa création. En revanche, ça n'est pas le cas du trésorier qui n'a aucun lien avec la catégorie. Un trésorier est une personne, rien de plus.

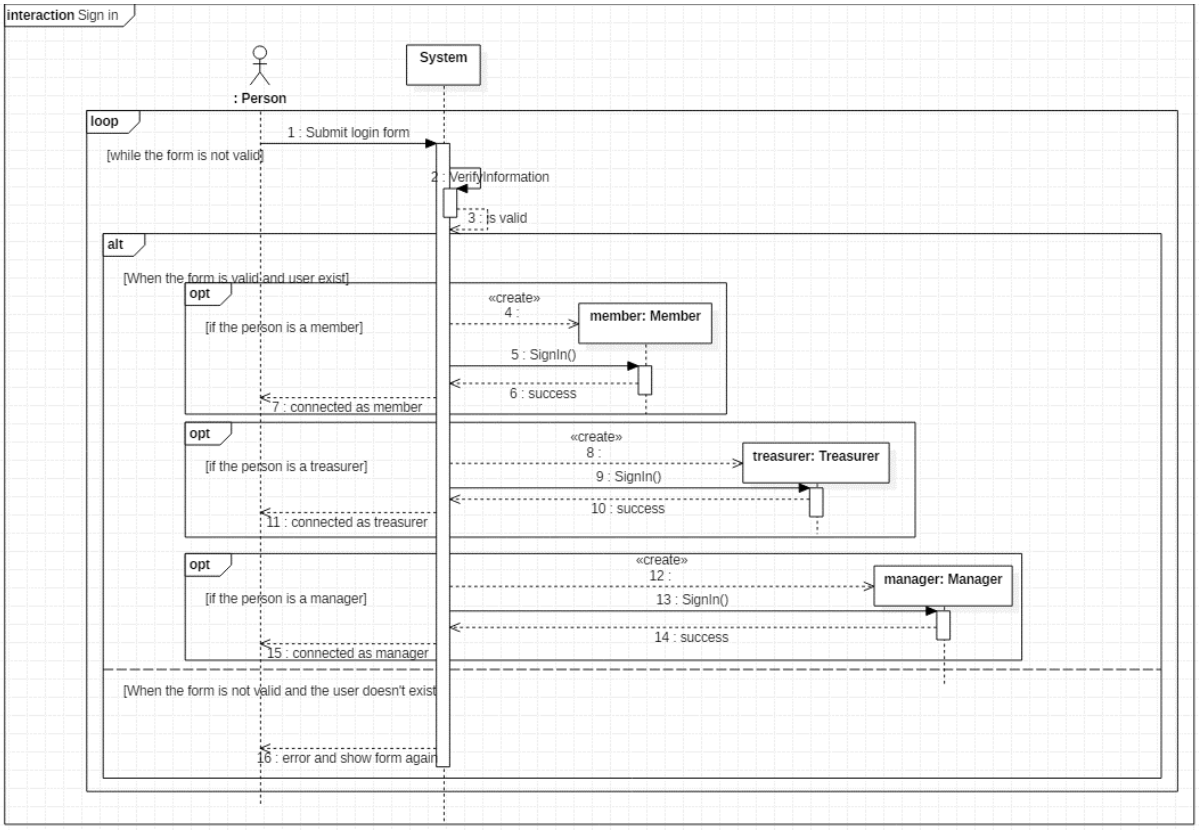
La catégorie est créée par le système : il n'est pas possible d'ajouter, de supprimer ou de la modifier. Ce n'est pas un acteur qui la créée. Le Manager gère le calendrier des sorties. Pas les catégories ! Donc, pour tout ajout de catégorie, le chef du club de cycliste devrait contacter le développeur pour mettre à jour le programme. À priori, la hiérarchie d'un club ne se modifie pas souvent.

En partant du principe qu'un manager choisisse sa catégorie à l'inscription, les liens 1-1 sont dérangeants entre les deux class. C'est pourquoi, j'ai accepté l'idée qu'une catégorie puisse ne pas avoir de manager avant la création de ce compte et ai changé la multiplicité dans le diagramme de class.

Erratum

A l'inscription du member, l'ajout d'un vélo doit lui être proposé car le lien est 1…*. Il devrait également être possible pour un membre de posséder plusieurs vélos et les inscrire.

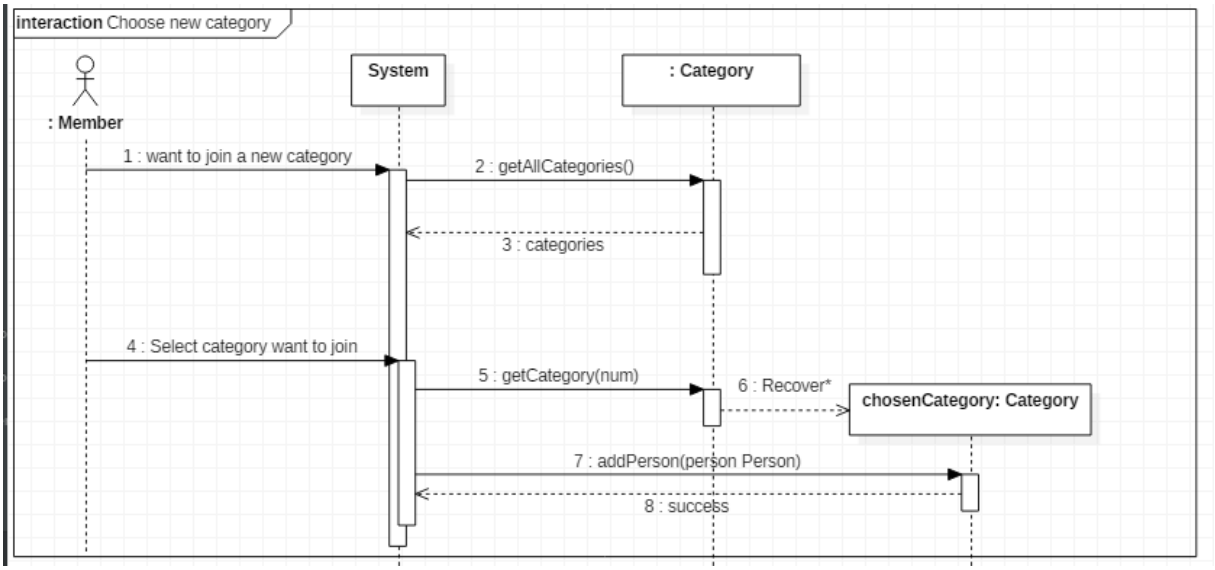
Sign in



Commentaire

Les utilisateurs se connectent via le même formulaire.

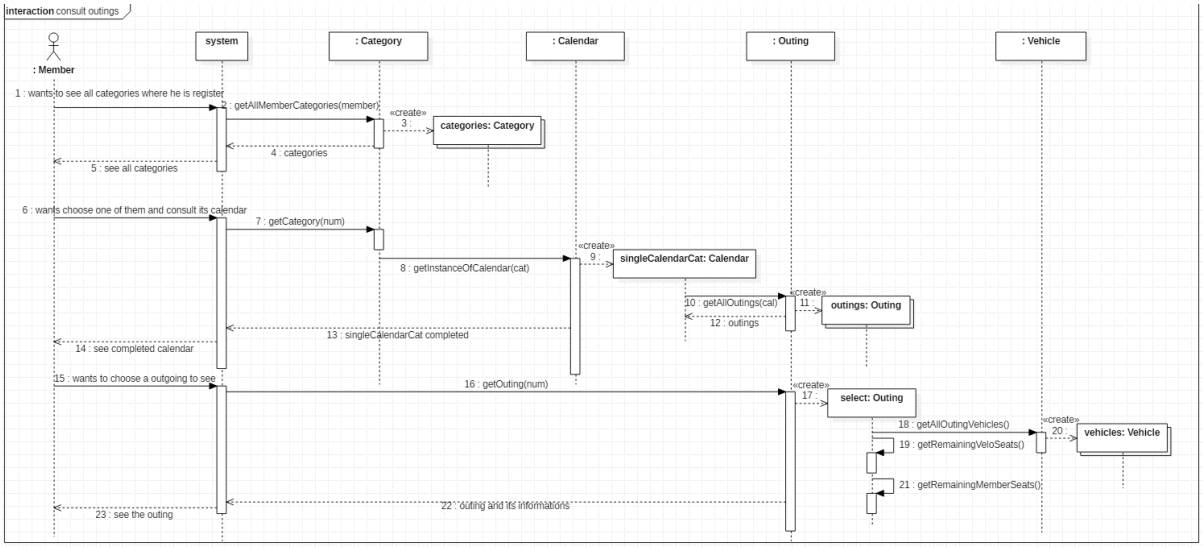
Choose new category



Commentaire

La catégorie choisie est une instanciation unique (puisque classe.)

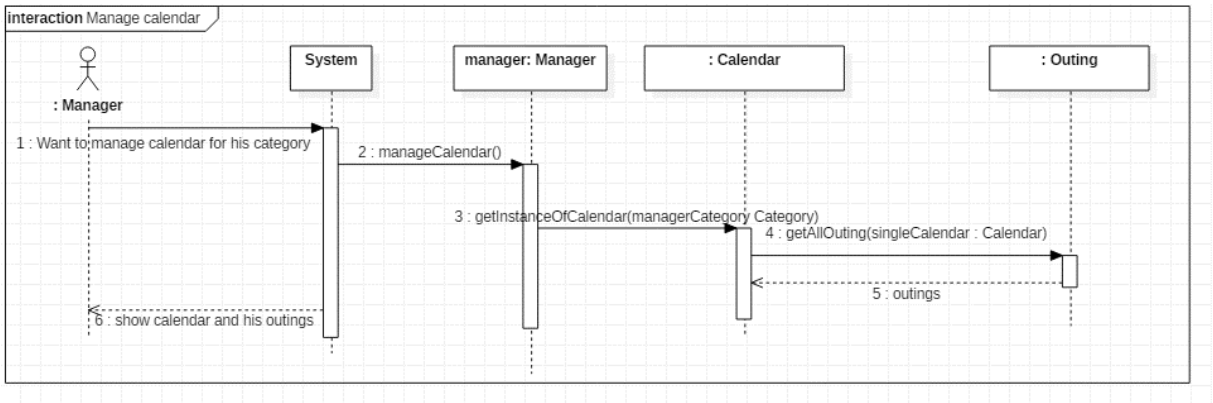
Consult outings



Commentaire

11/11 : Avec ce diagramme de séquence, il n'est pas possible de consulter qui participe pour les autres participants. Ils savent juste s'il y a suffisamment de places, s'il y a un/des véhicule(s) pour la sortie.

Manage calendar



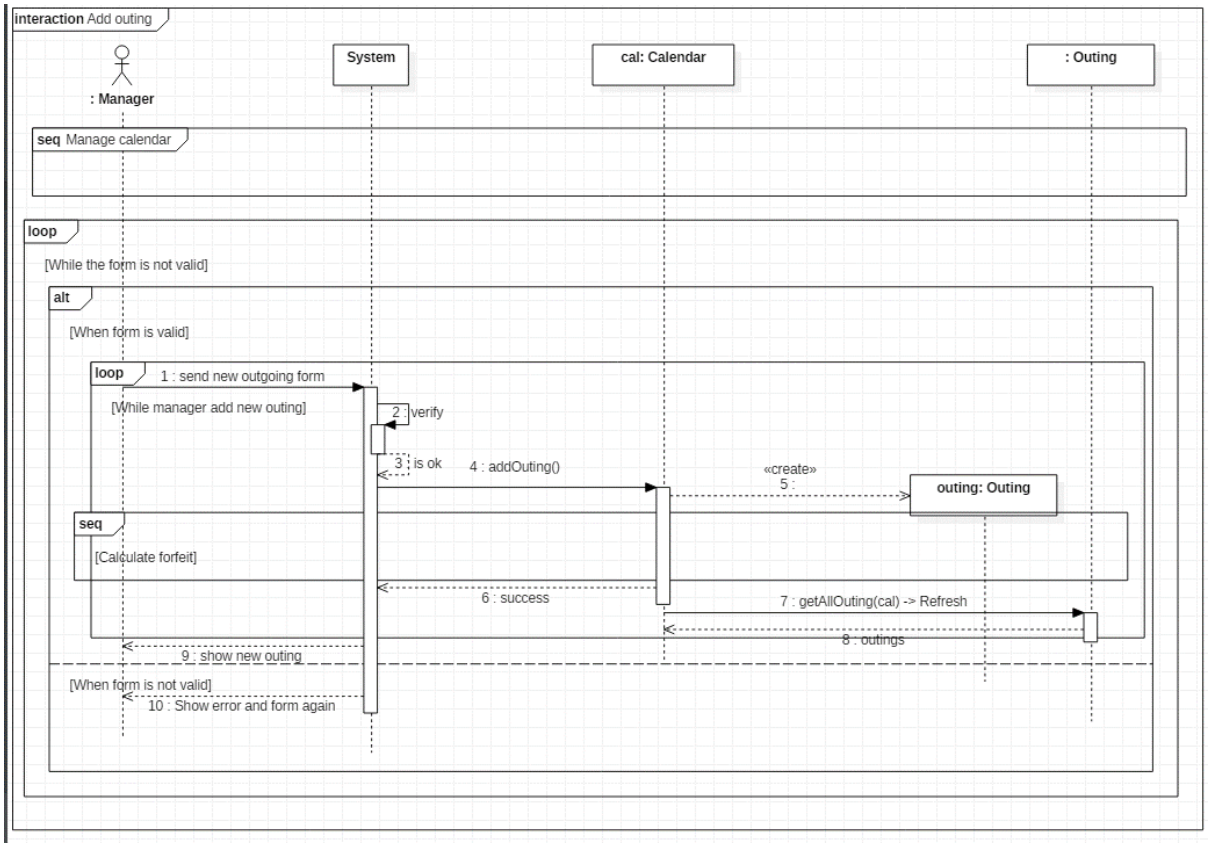
Commentaire

Le diagramme de class ainsi étant élaboré propose les catégories déjà présentes. Ainsi, il n'est pas possible d'ajouter une catégorie, de la modifier ou de la supprimer pour le manager.

➔ Pour rendre l'ajout et suppression possible pour le manager, il aurait fallu adapter l'analyse. Actuellement, le manager aurait dû être capable de supprimer l'unique catégorie dans laquelle il se trouve et il doit obligatoirement être dans une. Ça n'est pas possible. Il aurait fallu permettre aux users de faire partie d'aucune catégorie à un moment donné.

Ce diagramme de séquence permet simplement au Manager de voir les sorties pour un calendrier de sa catégorie. Les actions possibles seront détaillées dans les diagrammes de séquence suivants.

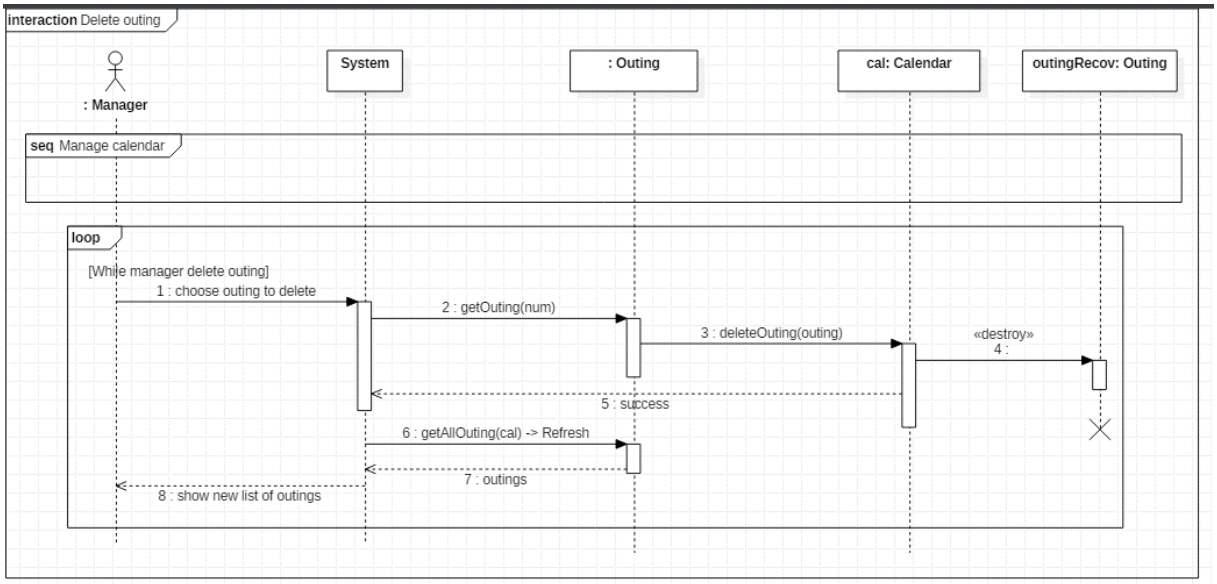
Add outing



Commentaire

La class calendrier est responsable de la création de sa sortie : c'est pourquoi ce n'est pas le système qui créé l'objet sortie pour le passer en paramètre de addOuting().

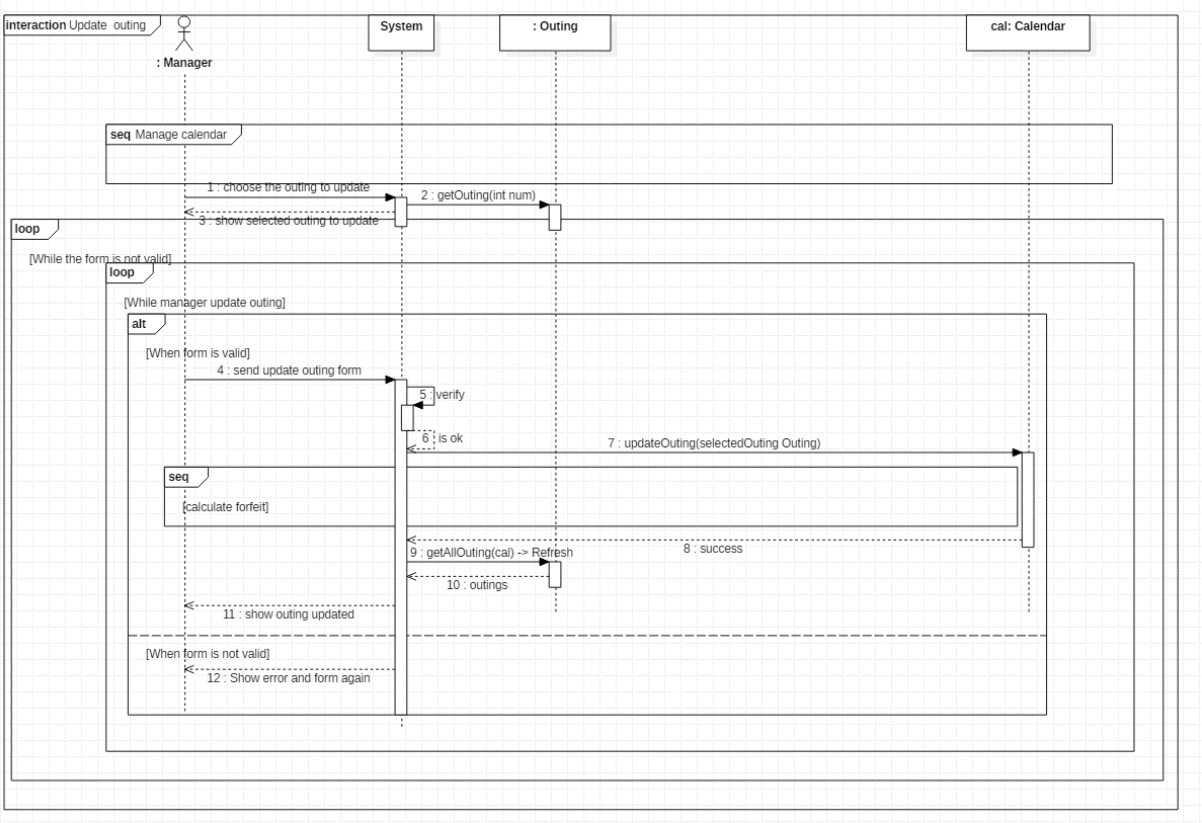
Delete outing



Commentaire

Bien que ce soit le calendrier qui supprime sa sortie à la demande du manager (via le system), la suppression de l'OBJET même dépend de sa propre classe.

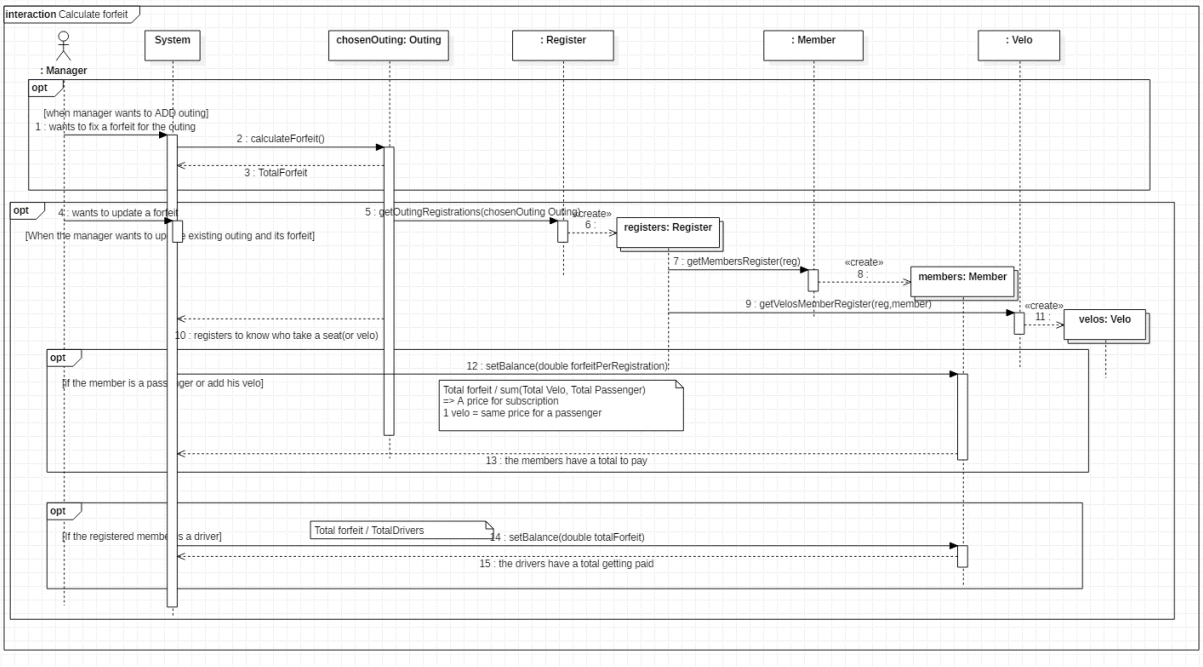
Update outing



Commentaire

Même si le manager ne change pas le prix du forfait, ce dernier est recalculé et le solde de la personne qui s'était inscrite est mis à jour.

Calculate forfeit



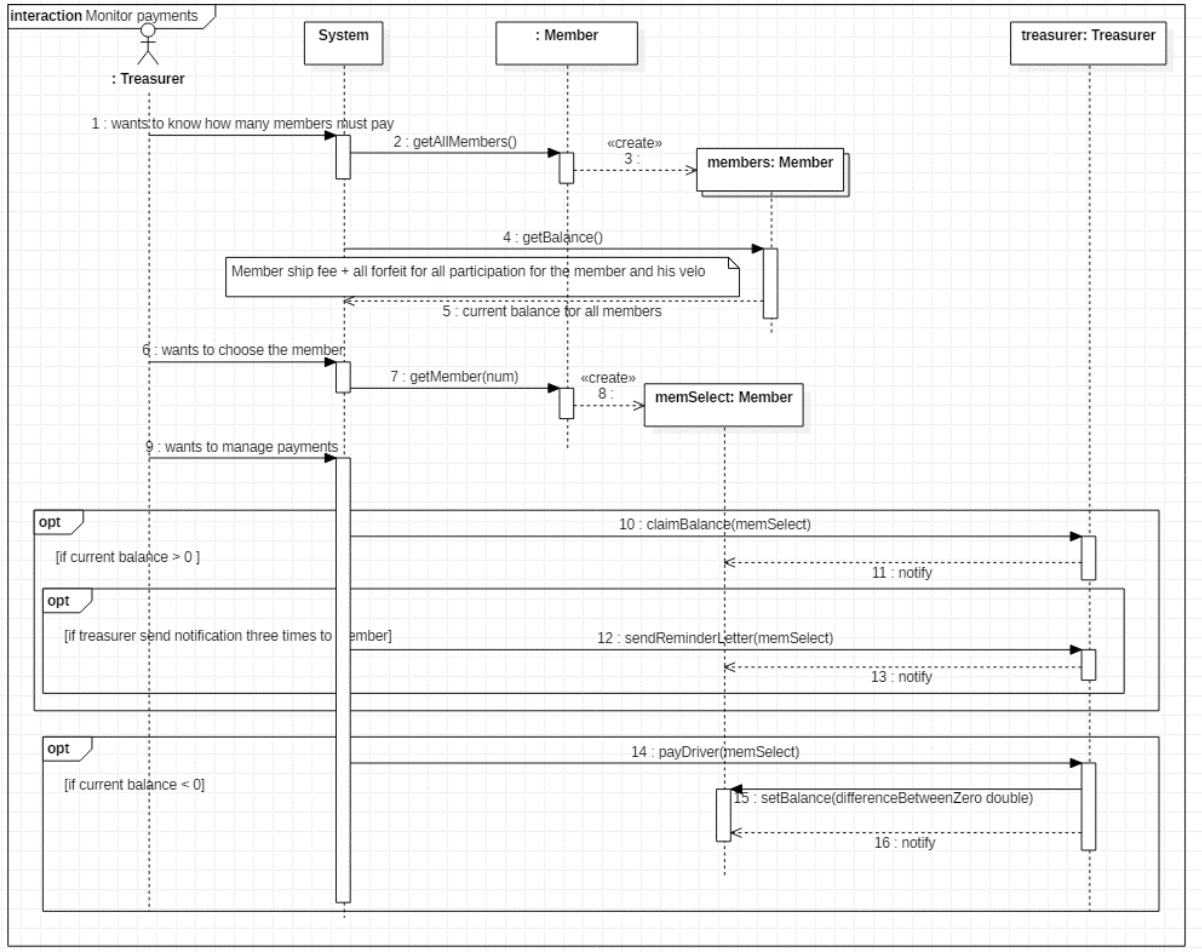
Commentaire

À la création de la sortie, il n'y a pas encore d'inscrits : il n'est pas nécessaire de communiquer avec les passengers et les drivers à ce stade.

Il sera alors possible de tenir à jour le prix du forfait par inscription pour la personne qui s'était déjà inscrite dans le cas d'une mise à jour du forfait de la sortie.
Étant donné qu'un passager peut être conducteur à un moment ou l'autre, j'ai pensé que modifier le solde total dû au club via des soustractions et des additions était une bonne idée.

Exemple : membre 1 a un solde de -20 -> Il a été conducteur et le club lui doit 20€. S'il participe à une sortie à 20 € ensuite en étant passager -> son solde serait -20 + 20 = 0. Ainsi, il ne doit rien à personne et personne ne lui doit. Cela éviterait les transactions monétaires inutile et mettrait en place un système de « pré-paiement » et d'arrangements.

Monitor payments

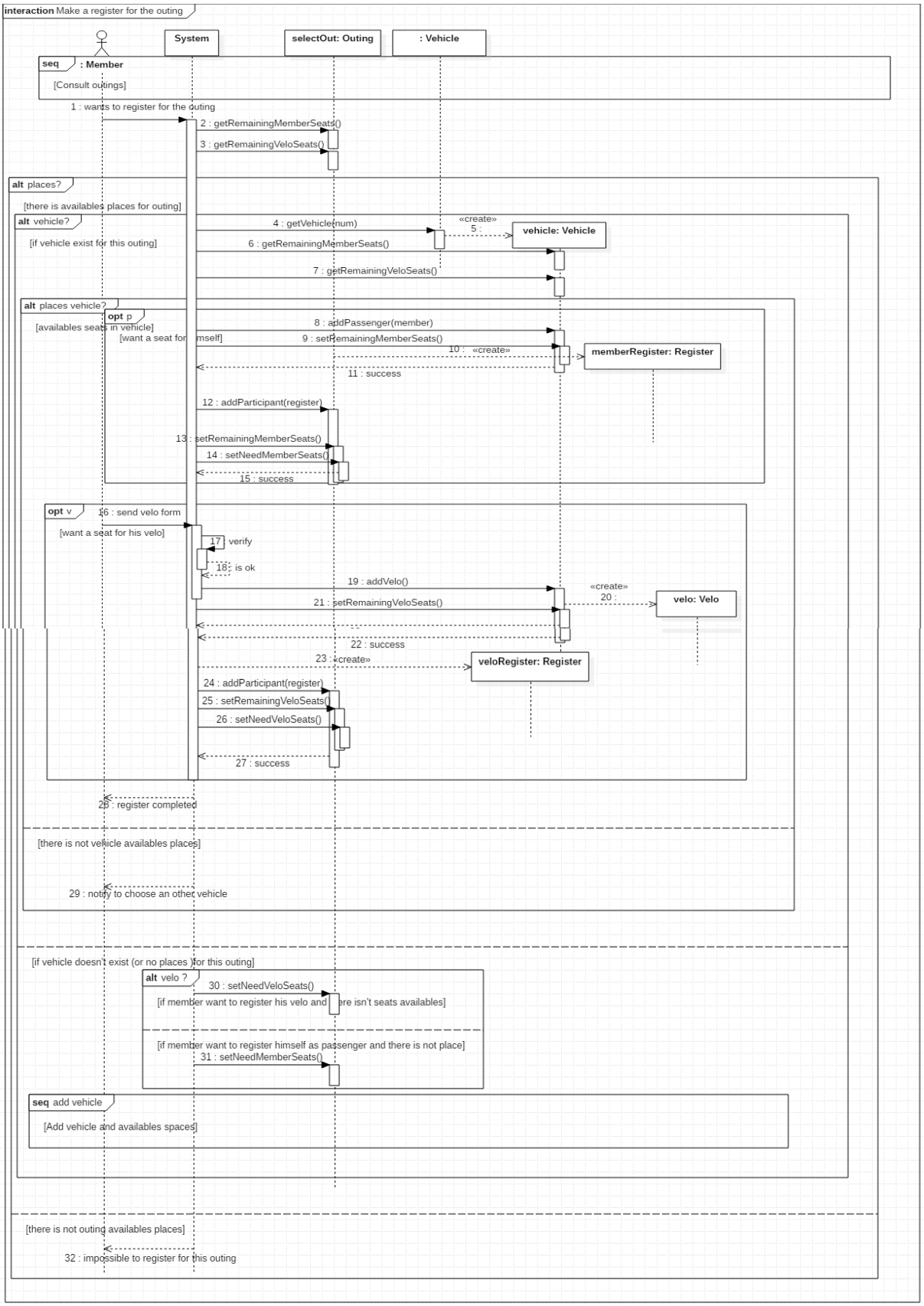


Commentaire

10/11

À ce stade-ci, le trésorier verra le montant total dû « au club » et celui que le club doit aux conducteurs. Non pas chaque montant pour chaque sortie et pour chaque membre.

Make a register for the outing



Commentaire

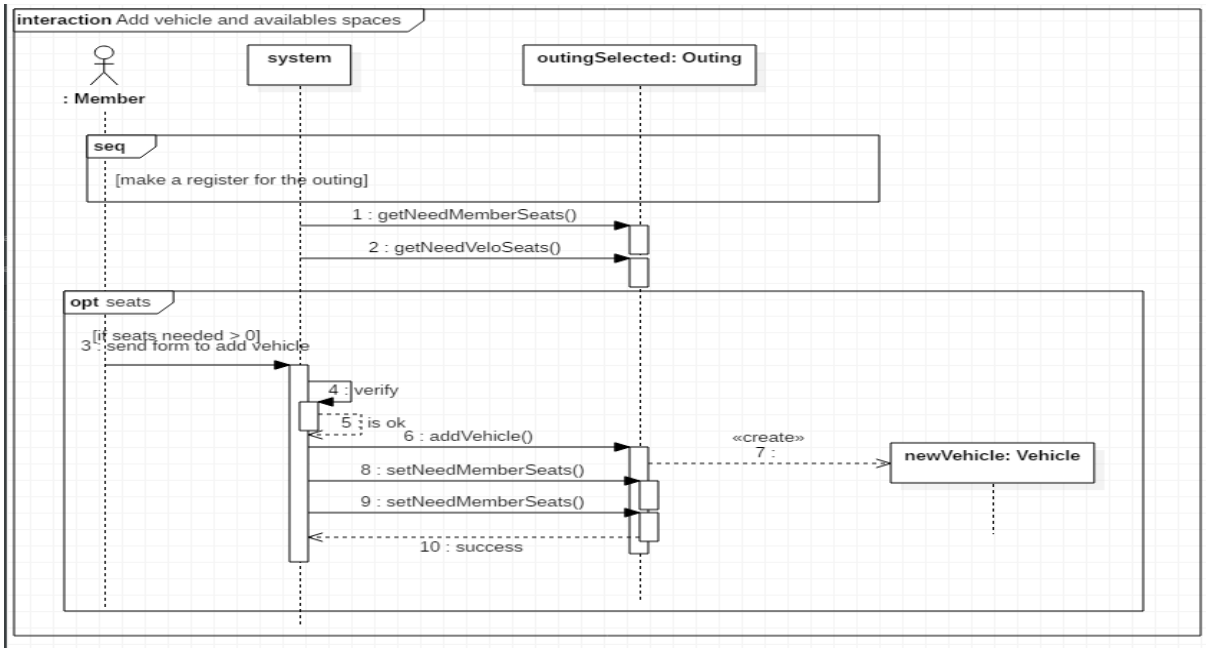
D'abord vérifier que la sortie n'est pas complète.
Si sortie pas complète : choisir voiture, vérifier qu'il y a suffisamment de place dans celle-ci.
Si sortie complète : pas possible de s'inscrire.
Si voiture assez de place : ajouter un passager ou un vélo.
Si voiture pleine : proposer de choisir un autre véhicule. Retour à « est-ce qu'un véhicule existe ? » ajouter une place nécessaire (vélo ou passager), retirer une place(vélo ou passager) de la sortie

La méthode addParticipant prend une inscription en paramètre pour savoir si c'est un membre ou un vélo qui est ajouté.

Si la personne s'inscrit à la sortie mais ne souhaite pas prendre de place, la participation sera créée mais les sièges nécessaires/pris ne seront pas mis à jour.

Lorsqu'un passager ou vélo est ajouté à la voiture, la sortie ajoute une inscription à celle-ci. De cette manière, il sera possible pour une personne de s'inscrire seule et d'ajouter son vélo par après. Ou les deux en même temps.

Add vehicle and available spaces



Commentaire

S'il manque des sièges pour un vélo ou pour un passager, n'importe quel participant aura le droit d'ajouter son véhicule s'il le souhaite. La personne qui souhaitait prendre un siège pourra le choisir une fois le véhicule ajouté. Ou alors, dès l'inscription d'une personne, il sera possible d'ajouter un véhicule si c'est nécessaire. Chaque fois, la personne devra ajouter son véhicule (donc, il ne sera pas enregistré pour cette personne !)
Amélioration possible : notifier les membres inscrits qu'un véhicule a été ajouté et leur proposer de le choisir.

Conclusion

J'ai mis beaucoup de temps à peaufiner « ce que je veux faire, comment le faire et ce que le client souhaite » et les diagrammes de séquence dont je ne suis même pas sûr. Je divague de trop. C'est une remarque qui m'a déjà été formulée. J'en suis conscient et je fournis des efforts pour me focaliser sur ce qui est nécessaire pour ne pas m'attarder sur des détails.

« Make a register » m'a donné beaucoup de fil à retordre, avec la classe « Register » et le fait qu'une personne puisse s'inscrire à la sortie, être passager ou non et inscrire ou non son vélo. Non pas à cause de l'énoncé mais des fautes de réflexion de ma part.

Aussi, j'ai mal géré mon temps entre les autres cours, projets à travailler et ma vie privée. Je préfère travailler seul pour n'emporter personne avec moi dans mes erreurs. Ce trait de caractère nécessitera adaptation pour l'avenir professionnel.

Dans tous les cas, ce projet m'a permis d'une nouvelle fois cibler mes failles et une remise en question au sujet de mes compétences. Il me faut m'améliorer, faire mieux et rendre un projet décent pour août.

Afin de faire perdre de temps à personne, je souhaite ***ne PAS présenter l'oral***. Ce torchon n'est pas encore prêt. Si je vous ai remis le projet, c'est dans l'espoir d'avoir un retour lors de la vision de copie.

J'aimerais pouvoir en discuter avec vous. Toutefois, si vous n'avez pas de temps à m'accorder en réel, un feed back écrit me va aussi !

Questionnement

Toujours en vue de m'améliorer, j'aurais quelques questions qui me viennent en tête pour la vision des copies.

Ne pas oublier de fermer la connexion à la bdd.

J'ai tenté de justifier mes choix dans les commentaires mais est-ce que ceux-ci sont bien raisonnés ?

- Méthodes

Register -> getOutingRegistrations => Inutile ? Car référence
Outing -> getAllOuting -> Inutile? Référence.
Vehicle -> getAllOutingVehicle(Outing outing) -> Inutile? Référence.
Vehicle -> getVehicle -> Inutile ? Je peux simplement passer par l'abstract factory dans le programme appelant ??
Category -> DeleteCategory : Comment supprimer la référence d'un objet courant ?

- POJO ?

Une classe est POJO quand elle :

- Implémente Serializable ;
- contient des getteurs/setteurs pour ses attributs ;
- possède au moins un constructeur par défaut.

Qu'en est-il lorsque ces classes proviennent d'un héritage et sont en plus de cela un singleton ?
=> Les catégories. J'ai mis un constructeur par défaut dans la catégorie alors que celle-ci n'est pas instanciable. Mais est-ce une bonne pratique ? Est-ce que cela fait de ces classes (les dérivées)des POJOs ?
Je ne peux pas mettre un constructeur par défaut dans les classes singletons.

- Github

Était-ce +- ce que vous attendiez pour l'utilisation de github ?

- Window builder

Pourquoi est-ce que mes radios box apparaissent au survol de la souris ?

- DAO (feedback du problème que je rencontrais)

Un problème avait été rencontré dans la DAO des catégories. Je souhaitais tout simplement set le manager d'une catégorie récupérée mais une exception se lançait :

(sur la ligne if(result.first()){…

Pistes proposées : ouvrir une connexion dans une connexion à la bdd = pas bonne idée et préférer utiliser les jointures.
Aucune de ces solutions n'a résolu le problème, malheureusement. Pour une raison que j'ignore toujours aujourd'hui, j'ai pu faire fonctionner la requête en passant par un while(result.next()){…

Pourquoi je ne peux pas récupérer un manager lorsque ce dernier a créé son compte ?

```
java.util.regex.PatternSyntaxException: Stack overflow during pattern compilation near index 144
#(0[1-9]|[1-9]|1[012])/(0[1-9]|[1-9]|[12][0-9]|3[01])/(\d\d\d\d)\s*(([0-9]|0[0-9]|1[0-9]|2[0-4]):([0-9]|[0-5][0-9]):([0-9]|[0-5][0-9]))\s*
```

Ce genre d'Exception apparaît. Je n'utilise pas de regex… ?

De même, mon programme ne se lance pas si j'ai un membre qui a sign up.