Analyse du projet ClubVélo pour le cours de Java

# Par Wilson Weets

Table des matières

[Par Wilson Weets 1](#_Toc499016189)

1. Analyse

## Diagramme des cas d’utilisations

Il existe trois types de personne interagissant avec le programme.

Le membre peut :

* Poster ses disponibilités
* Voir sa/ses catégorie(s)
* S’inscrire à une autre catégorie (et indiquer si le paiement a été effectué)
* Réserver sa place

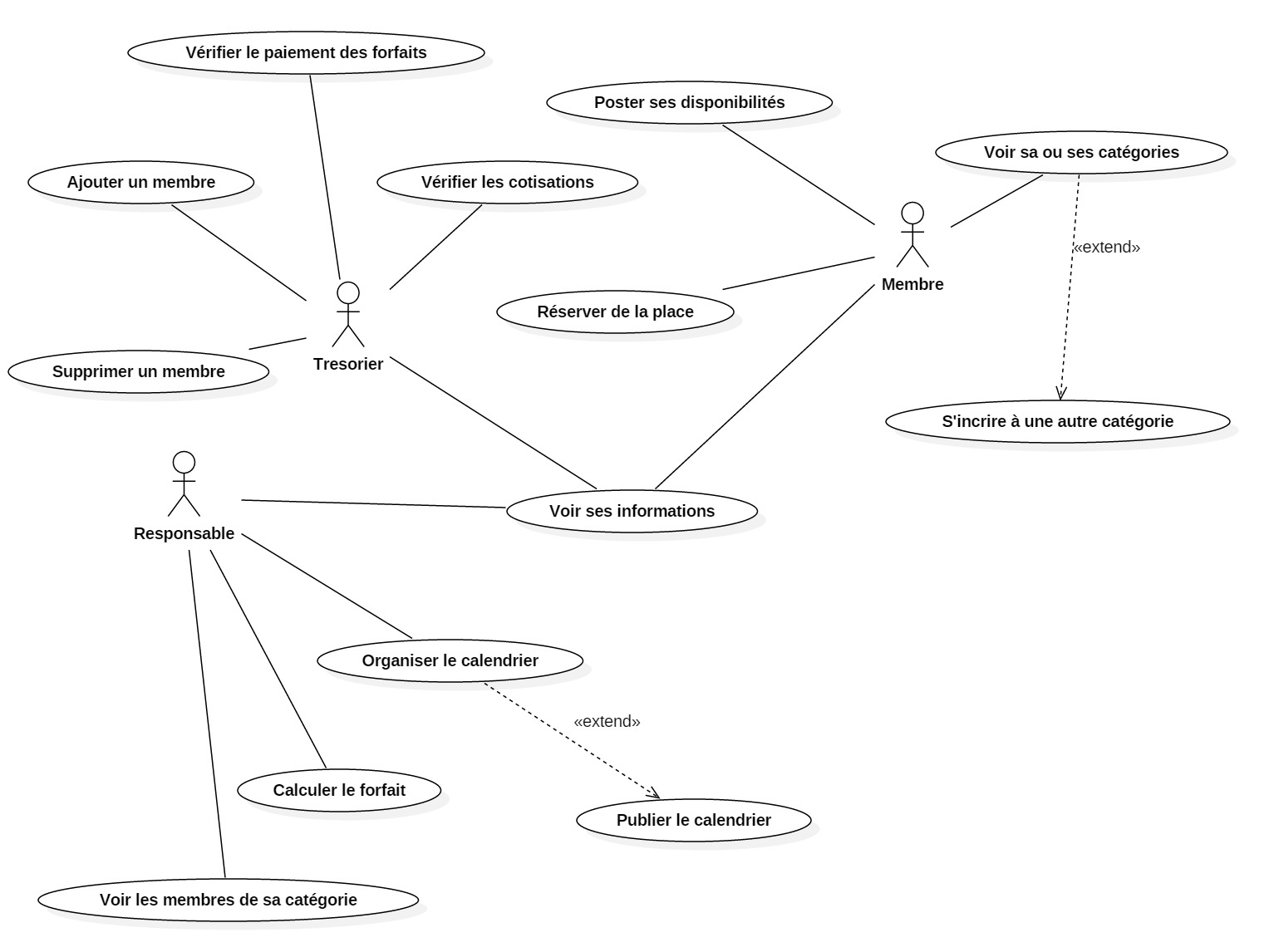
Le responsable peut :

* Organiser et publier le calendrier
* Calculer le forfait pour chaque balade de sa catégorie
* Voir les membres de sa catégorie

Le trésorier peut :

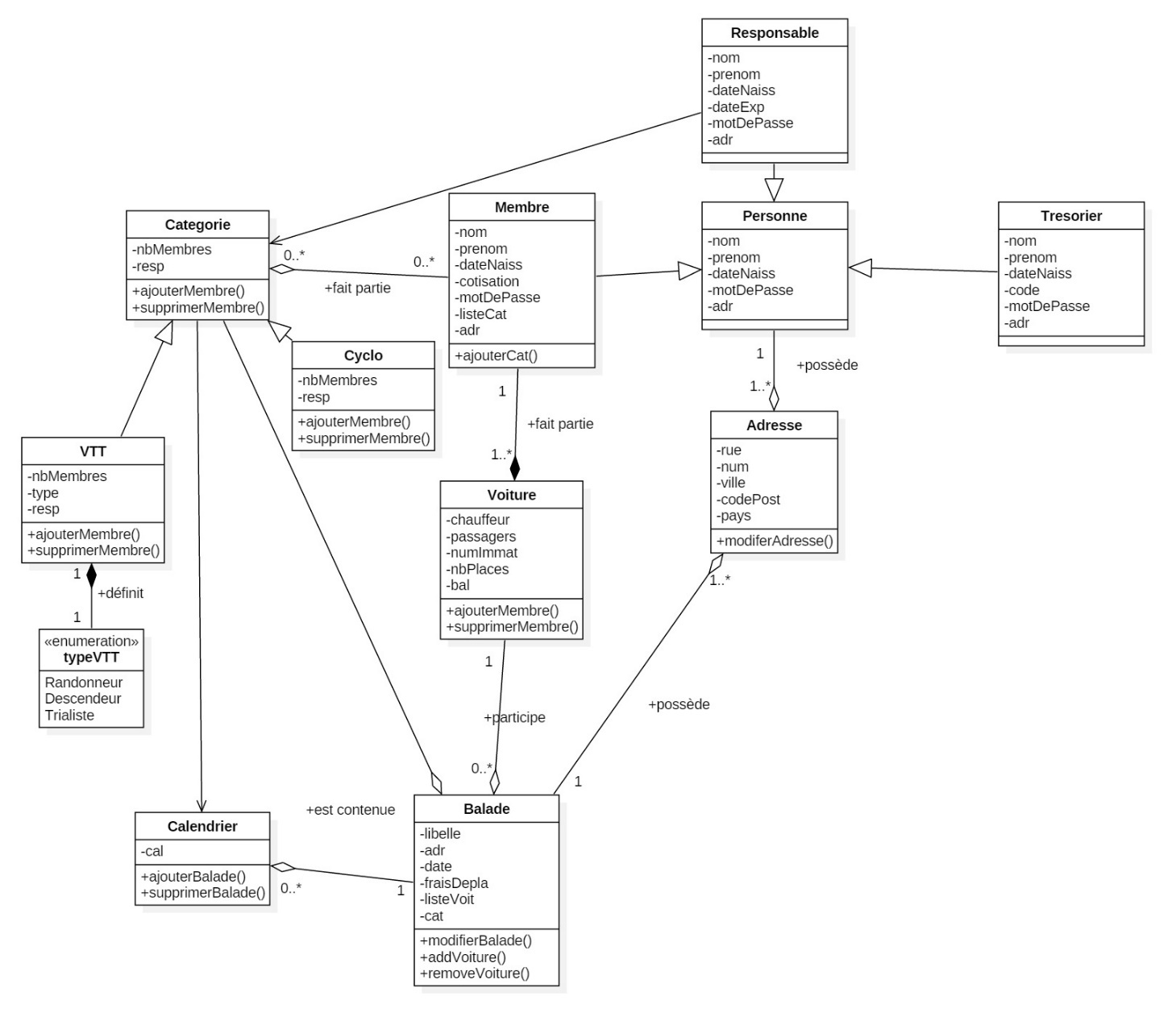
* Ajouter ou supprimer des membres
* Vérifier les paiements (forfaits et cotisations)

Tout le monde peut voir ses informations personnelles.



## Diagramme de classes

Le diagramme de classe est similaire au diagramme proposé en classe. J’ai juste pris la liberté d’utiliser une classe d’adresse pour les différentes adresses des personnes et balades. Cette classe a donc des références dans les types Personne et Balade.



## Diagrammes de séquence

Je ne pourrai pas vous rendre de diagrammes de séquence car mes classes métier n’interagissent pas vraiment entre elles. En effet, elle se modifient chacune comme le veut le principe d’encapsulation mais les appels ne viennent que des classes principales qui vont utiliser leurs méthodes et propriétés.

1. Implémentation

J’implémente la base données grâce à une classe DBConnexion. J’utilise le paterne DAO accompagné d’un paterne Factory. Chaque classe DAO est appelée et instanciée grâce à la Factory. Les appels se font dans le code principal.

Ces pages de codes principales ont été créées grâce à l’outil WindowBuilder d’Eclipse. Cet outil génère du code Swing. Cependant, j’ai commenté mon code pour que vous puissiez ne pas prêter attention au code d’affichage qui est inutile pour vous.

Les commentaires sont structurés comme suit :

//SECTION : <Nom de section>

//Elément important à décrire (par exemple : Boutons)

// FIN DE SECTION : <Nom de section>

La page de départ, qui par conséquent contient la seule méthode « main », se nomme ConnexionWindow.java.

J’ai dû créer des classes supplémentaires pour définir les modèles d’affichage des différents tableaux. Ces classes ne servent qu’à l’affichage.

Le Workspace se structure comme ceci :

* be.wilson.ClubVeloConnection : Contient la classe de connexion (DBConnection)
* be.wilson.ClubVeloFactory : Contient les classes du paterne Factory (AbstractDAOFactory, DAOFactory)
* be.wilson.ClubVeloDAO : Contient les classes DAO (DAO, AdresseDAO, BaladeDAO, CategorieDAO, MembreDAO, ResponsableDAO, TresorierDAO, VoitureDAO)
* be.wilson.ClubVeloPOJO : Contient les classes POJO/Métier (Adresse, Balade, Calendrier, Categorie, VTT, Cyclo, Membre, Personne, Responsable, Tresorier, TypeVTT, Voiture)
* be.wilson.ClubVelo.UI : Contient les classes d’affichages (ConnexionWindow, MenuWindow, RespWindow, TresWindow, TresCodeWindow, BaladeTableModel, VoitureTableModel, CalendrierTableModel, CotisationTableModel, CategorieTableModel)