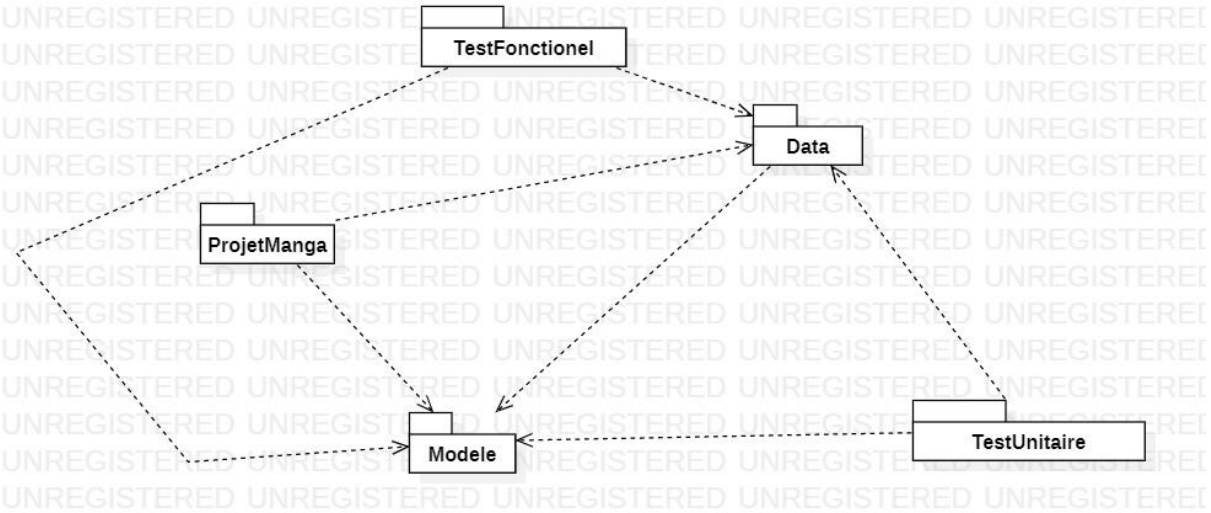
Louis Perret
Simon Dallet
3H-4H
IUT 2020/2021

Projet

Sommaire

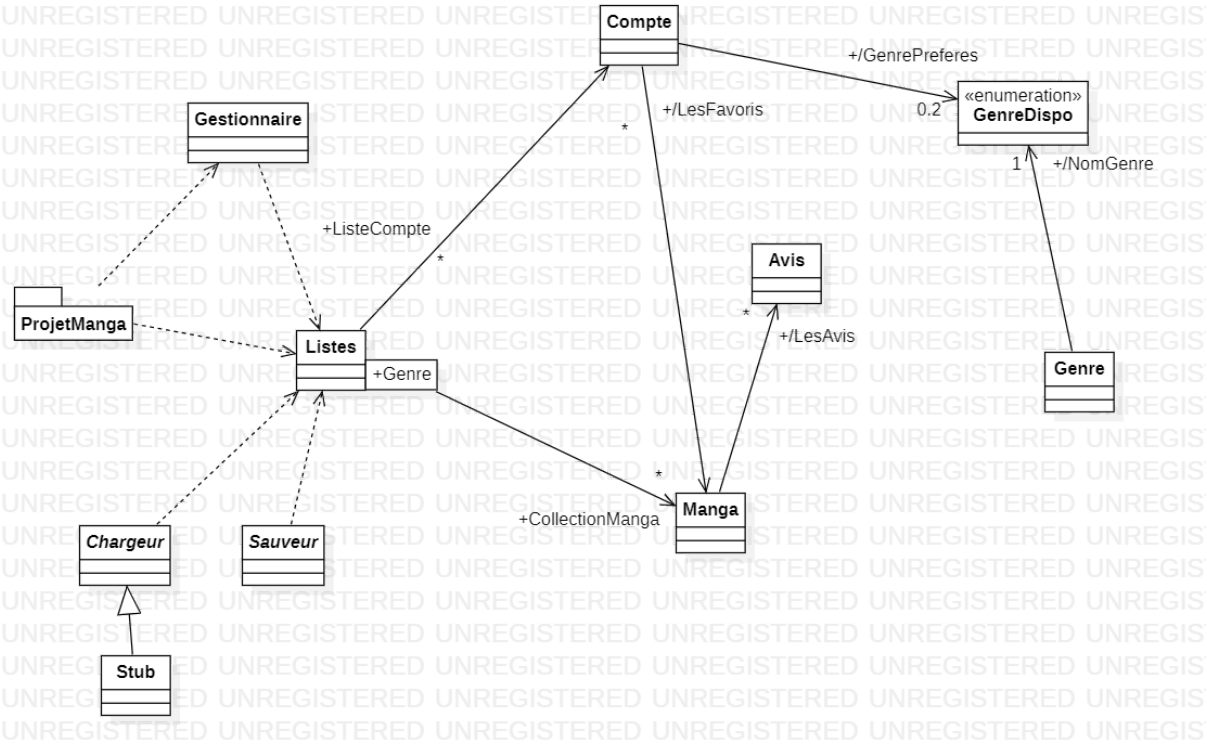
Diagramme de paquetage



Description du diagramme de paquetage

ProjetManga : Projet WPF qui contient la vue de notre application
Modèle : Bibliothèque de classes contenant la logique de notre application
Data : Bibliothèque de classes contenant nos classes Sauveur, Chargeur et Stub permettant le chargement et plus tard la sauvegarde de nos données.
TestFonctionnel et TestUnitaire nous permettent de tester nos fonctionnalités ainsi que certaines de nos méthodes

Diagramme de classe

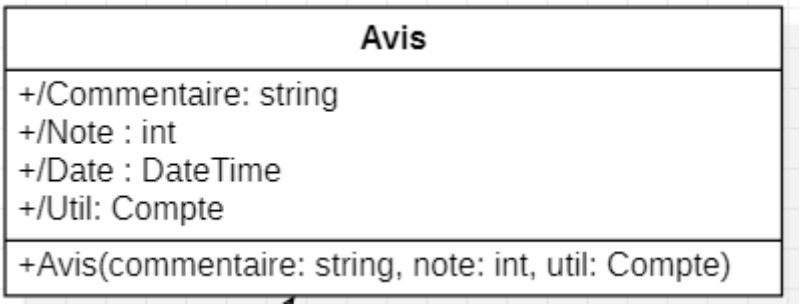


Pour un diagramme plus lisible, on a décidé de vous le montrer avec les classes vides, et on montrera les classes complètes lorsqu'on les décrira

ProjetManga représente notre vue (nos User-Control et nos fenêtres)

Description du diagramme de classe

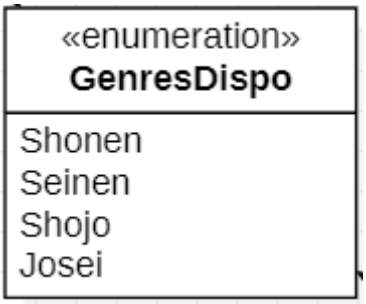
Classe Avis :



Cette classe représente les avis qu'on peut émettre sous un manga.
Elle comprend 4 propriétés : une note (entre 0 et 10) obligatoire, un commentaire (facultatif), une date de saisie correspondant au jour et à l'heure à laquelle il l'a posté, et un utilisateur permettant de faire figurer son nom sur l'avis. On prend directement une instance de Compte car cela permet de prendre en compte si le pseudo de l'utilisateur a été modifié sans avoir besoin de faire des vérifications.

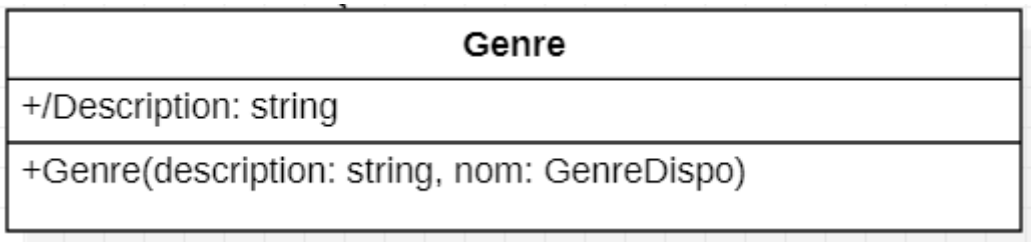
Cette classe ne comprend pas de méthodes, elle définit simplement la méthode ToString ainsi que son constructeur.

Classe GenresDispo :



Il s'agit d'une énumération des noms des genres des mangas de notre application. Le choix d'une énumération nous permet de contrôler le nom des instances de la classe Genre et donc leur instanciation. Cela nous permettra aussi de rajouter des noms dans le futur.

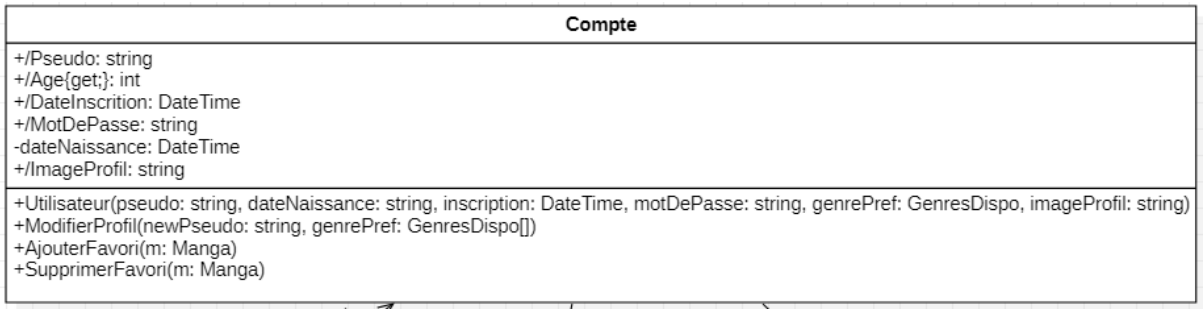
Classe Genre :



Cette classe décrit un genre. Elle comprend une propriété Description et une propriété Nom qui est récupérée dans l'énumération GenresDispo et qui correspond donc à un nom d'un genre.
Nous avons décidé de faire cette classe car dans notre vue on affiche la description du genre sélectionné par l'utilisateur et nous avions donc dû avoir besoin de stocker cette information, de plus la création de cette classe permet donc de faire coïncider la description avec le nom d'un genre de l'enum.

2 méthodes sont redéfinies : Equals et ToString. La méthode GetNom va permettre de récupérer le nom du genre comme il s'agit d'un attribut.

Classe Compte :

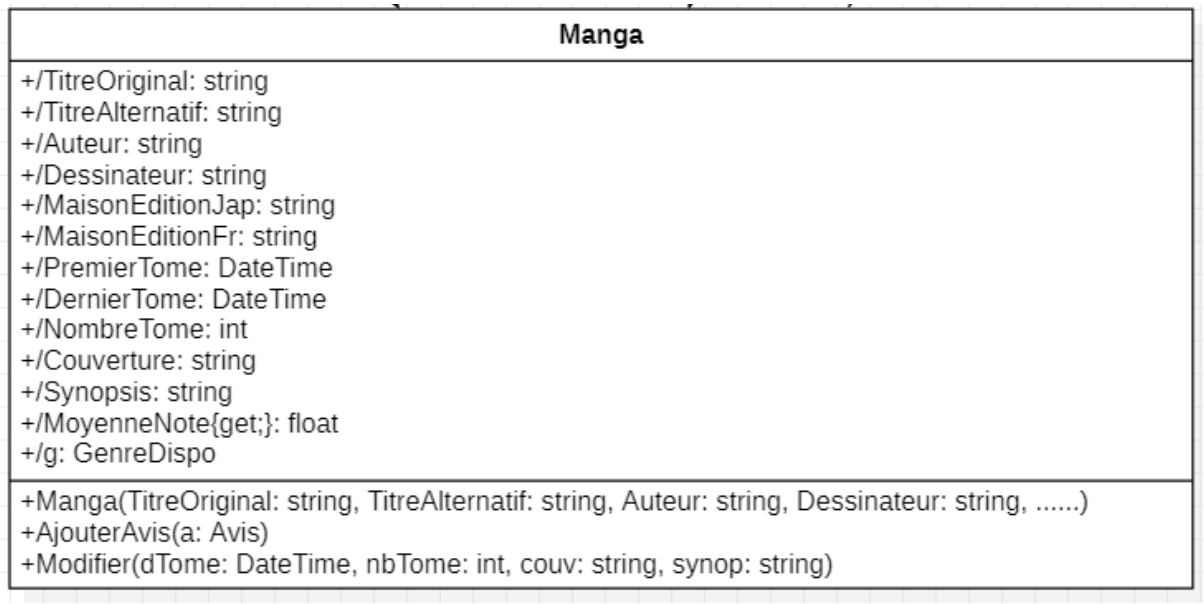


Cette classe définit les utilisateurs de notre application. Il s'agit de leur compte.
Les propriétés Pseudo, Age, DateInscription, MotDePasse constituent les caractéristiques d'un compte. A noter que Age est une propriété calculée grâce à un attribut privé dateNaissance et que DateInscription sera défini automatiquement à la création d'un compte.
Le compte possède une collection de Manga nommée Favoris qui permettra à un Compte d'indiquer (ou non) les mangas qu'il adore et un tableau de GenreDispo qu'il préfère (de 0 à 2).

La méthode ModifierProfil permet de modifier les genres favoris et/ou le pseudo.
Les méthodes AjouterFavori et SupprimerFavori permettent l'ajout ou la suppression d'un manga dans sa collection de Favoris.

Cette classe possède également Equals & GetHashCode

Classe Manga



Pour des raisons de lisibilité nous avons raccourci les paramètres du constructeur mais ses paramètres correspondent aux propriétés de Manga à l'exception de MoyenneNote qui est une propriété calculée et de LesAvis qui est une collection d'Avis.

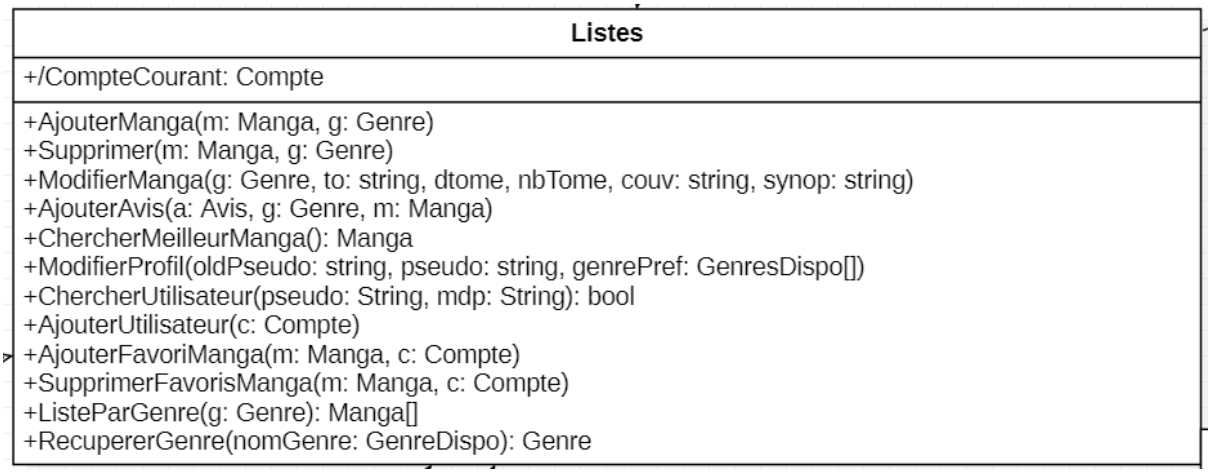
Cette classe définit le type d'éléments de la collection de notre master-detail. Elle possède de nombreuses propriétés permettant de la définir :

A noter :

- ❏ DernierTome est une date de parution qui représente la date de parution du dernier tome et qu'elle peut donc être null
- ❏ Couverture est un string représentant le chemin de l'image,
- ❏ LesAvis (voir diagramme de classes) est une collection d'Avis représentant tous les avis qui sont écris sur ce manga
- ❏ MoyenneNote est une propriétée calculée à partir des notes des avis émis qui sont stockés dans sa propriété LesAvis

Elle possède une méthode AjouterAvis lui permettant de recevoir un Avis (voir classe Avis) qui s'ajoute à sa collection LesAvis.
La méthode Modifier permet de modifier ou rajouter des éléments qui peuvent changer (date du dernier tome, nombre de tome) ou des informations comme l'image de la couverture et le synopsis.

Classe Listes

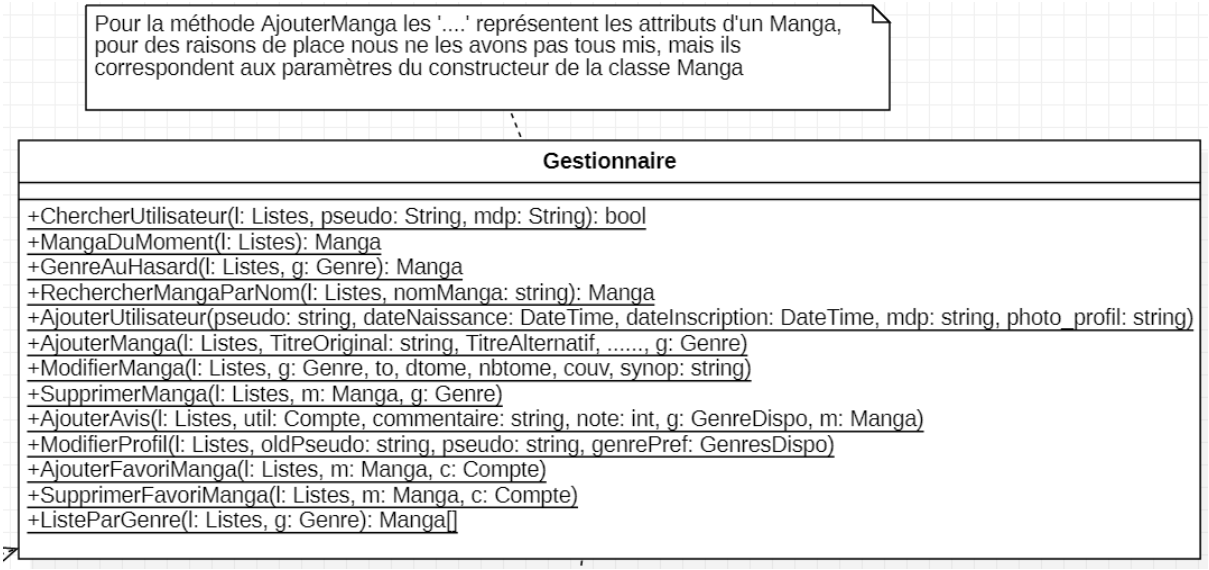


Cette classe nous permet de stocker tous nos éléments sous forme de collection qui seront affichées dans la vues et sérialiser.
Elle possède deux propriétés, une collection de Compte : ListeCompte et une collection qui est un Dictionary CollectionManga avec comme clé un Genre et comme valeur correspondant à ce Genre, une collection de Manga puisque plusieurs mangas peuvent posséder le même genre.

Pour ses méthodes :

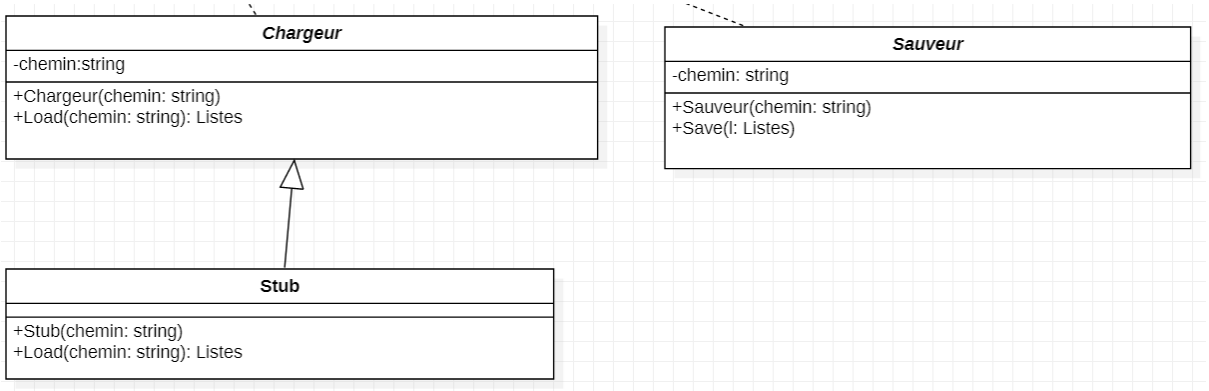
- ❏ AjouterManga, SupprimerManga et ModifierManga sont des méthodes clés pour gérer la collection de manga réalisable que si l'utilisateur est un administrateur. Elles reçoivent toutes les trois un Manga et un Genre en paramètre pour permettre de le retrouver dans le Dictionary.
- ❏ La méthode AjouterAvis appelle la méthode de Manga (du même nom) pour ajouter un avis à un Manga.
- ❏ La méthode ChercherMeilleurManga va renvoyer le Manga ayant la meilleure note à travers tous les mangas de tous les genres.
- ❏ La méthode ModifierProfil va appeler la méthode ModifierProfil de Compte
- ❏ La méthode ChercherUtilisateur va permettre d'actualiser la propriété CompteCourant en faisant pointer sa référence sur le compte rechercher.
- ❏ La méthode AjouterUtilisateur va permettre d'ajouter un Compte dans notre collection de Compte.
- ❏ Les méthodes AjouterFavoriManga et SupprimerFavoriManga vont appeler les méthodes du même nom que le compte passer en paramètre.
- ❏ La méthode ListeParGenre va permettre de renvoyer une collection de manga en fonction du Genre passé en paramètre.
- ❏ La méthode RecupererGenre renvoie un Genre à partir de son nom.

Classe Gestionnaire



Cette classe va permettre de gérer toute notre application en faisant la liaison entre la vue et notre modèle. On a fait toutes ses méthodes statiques car la classe ne possède pas d'attributs ni de propriétés et donc il est inutile de l'instancier.
La plupart de ses méthodes sont des appels des méthodes de Listes, notre classe qui est à sérialiser.
La méthode MangaDuMoment va se servir de la méthode MeilleurManga pour retourner le manga à afficher à la Une.

La méthode GenreAuHasard va retourner, à partir du genre calculé au hasard, la collection manga qui lui est attribuée dans notre dictionnaire.
La méthode ChercherMangaParNom va permettre de retourner un manga en indiquant son nom en paramètre.



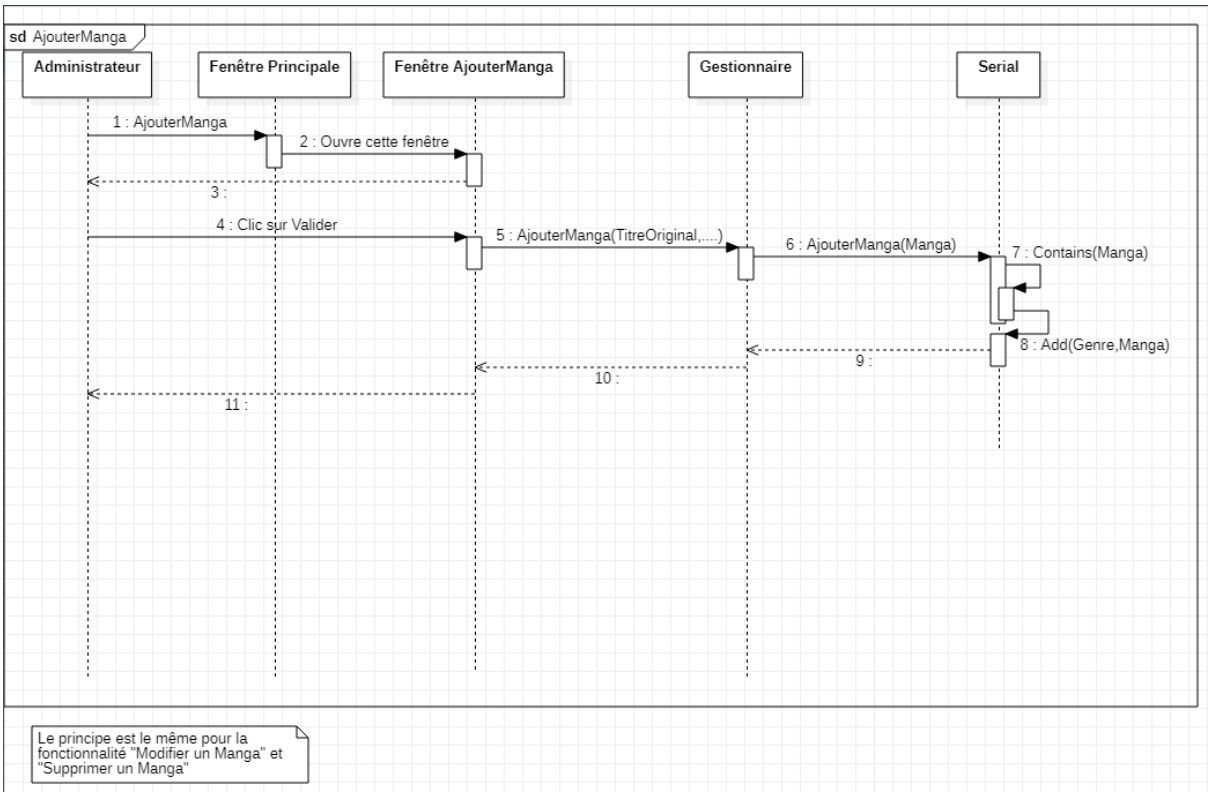
Ceci est la partie de notre diagramme de classes concernant le chargement et la sauvegarde.
Nos classes Chargeur et Sauveur sont des classes abstraites et contiennent chacune un constructeur et une méthode qui leur est propre. Cette stratégie de conception nous permet donc avec l'héritage de spécialiser quel type de chargement ou de sauvegarde nous voulons. Si on veut passer par une base de données, par exemple, il nous suffira de généraliser ces deux classes avec leur méthode respective qui nous permettra de charger et sauvegarder avec une base de données.

Classe Sauveur
Cette classe abstraite va permettre de sérialiser notre instance de Listes.
Elle possède une méthode qui reçoit un objet Listes en paramètre pour le serialiser.

Classe Chargeur
Cette classe va permettre de charger nos données à partir d'un objet Listes

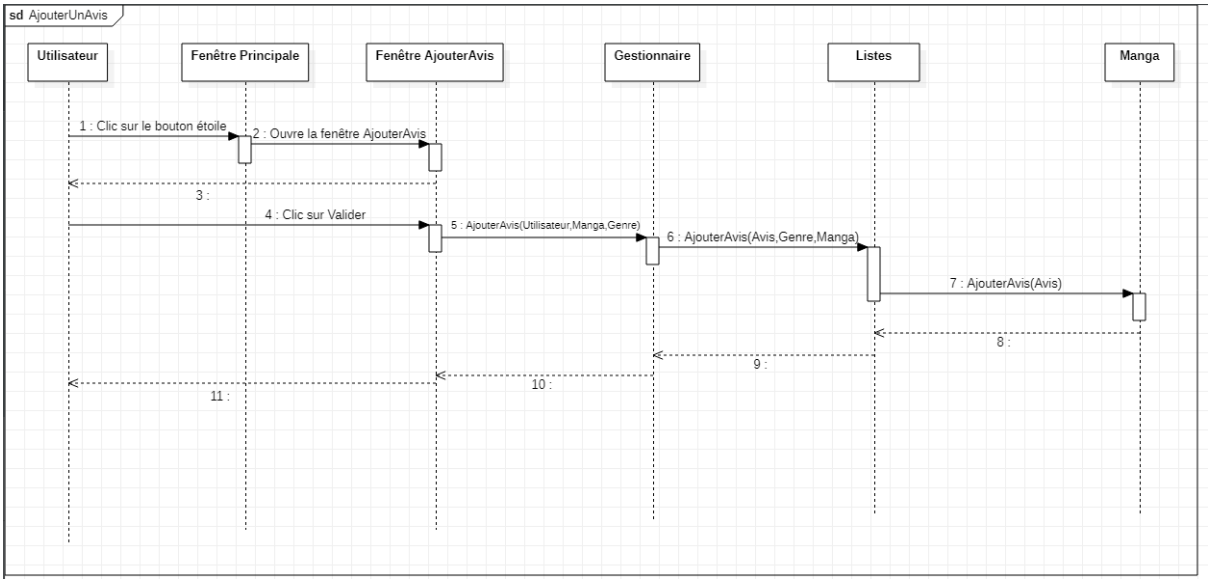
Classe Stub
Cette classe contient nos données rentrées en dur, elle est notamment utile pour nos tests.

Diagrammes de séquences



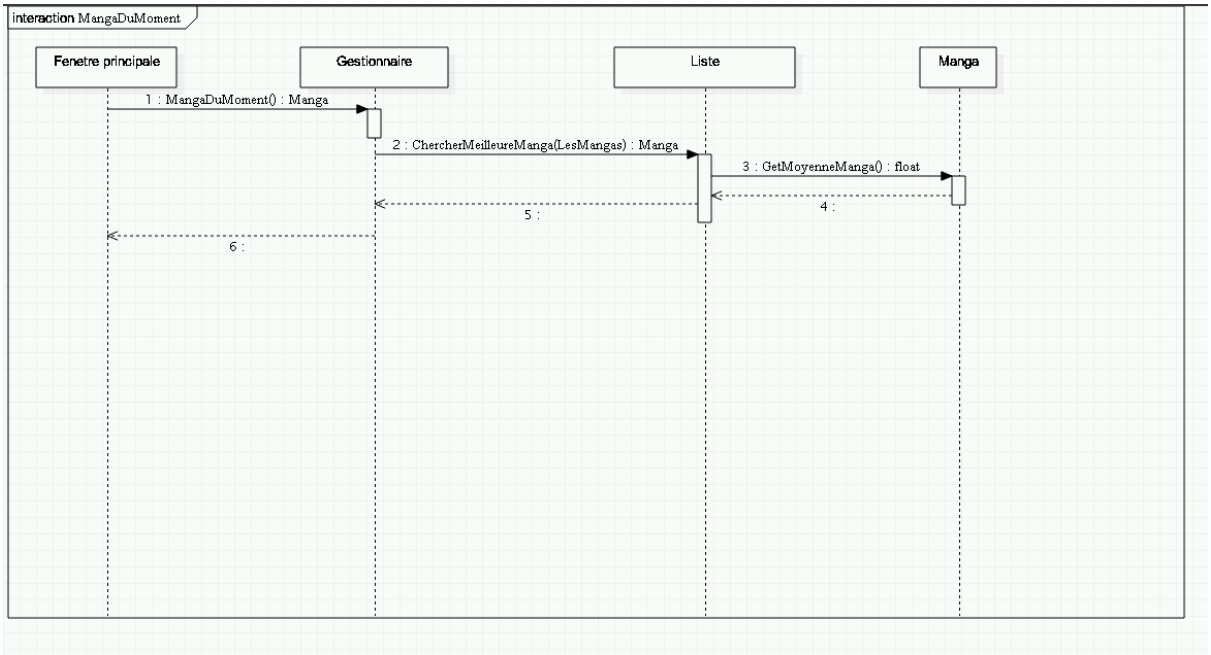
Description de ce diagramme de séquence

Ce diagramme de séquence représente le fonctionnement de notre fonctionnalité “ajouter manga”. Tout d’abord seul un administrateur est en capacité d’ajouter un manga à notre collection (respectivement de le modifier et de le supprimer).
Notre fenêtre principale fait apparaître une fenêtre dans laquelle il peut saisir les informations à rentrer (dans le cas de supprimer, ça fait apparaître une fenêtre de confirmation). Ensuite notre classe Gestionnaire se charge d’effectuer les opérations/appels nécessaires à la réalisation de la fonctionnalité.
Remarque : Pour des raisons de place, les informations passées en paramètres sont conséquentes nous avons donc préféré ne pas toutes les afficher mais elles correspondent aux paramètres du constructeur de la classe Manga. (Dans le cas de la fonction supprimer, le manga est passé directement en paramètre).



Description de ce diagramme de séquence

Ce diagramme de séquence représente le fonctionnement de notre fonctionnalité “ajouter un avis”. L’utilisateur rentre la note (et un commentaire mais ce dernier est facultatif) sur une fenêtre lancée par notre fenêtre principale. Ensuite le gestionnaire se charge d’instancier un avis puis de l’ajouter dans le manga concerné.



Description de ce diagramme de séquence

Ce diagramme de séquence représente le fonctionnement de notre fonctionnalité “manga du moment”. Cette fonctionnalité est lancée dès l’ouverture de notre fenêtre principale et permet d’afficher le manga possédant la meilleure note, tous genres confondus.