I.U.T Clermont-Ferrand

Projet JavaFx
Team'Dex
Description

Team'Dex Documentation

Élèves :
Elouan Raffray
Gabin Salabert

Enseignant :
Marc Chevaldonné

Décembre 2017 / Janvier 2018

1 Contexte de l'application

Cette application est constituée de l'univers des Pokémons, et plus précisément de la 1ère génération créee en 1996 au Japon. Les Pokémons étant, comme leur nom l'indique, des monstres de poche qui ont été commercialisés sous la forme de cartes à jouer, de mangas ou encore de jeux vidéo (etc).

L'application permet donc d'avoir accès à la liste des 151 premiers Pokémons pour obtenir, dans un premier temps, toutes les caractéristiques spécifiques de chacun d'eux sous forme de tableau. À savoir : le(s) type(s), le nombre de points de vie, le niveau d'attaque, de défense, d'attaque spéciale, de défense spéciale et de vitesse.

Il est possible de trier la liste par numéros d'identification, par ordre alphabétique des noms ou encore par type. De plus, une barre de recherche est aussi disponible dans le but de trouver plus facilement les Pokémons grâce à l'immuabilité de la classe String (méthode "toLowerCase").

Par la suite, nous avons ajouté la possibilité de gérer des équipes de 6 Pokémons, ces dernières étant nécessaires afin de disputer des matchs dans les jeux vidéo. Il est donc possible d'en créer en leur donnant un nom, puis de le modifier ou encore de supprimer l'équipe sélectionnée.

Le système d'ajout de Pokémons tourne autour d'un mécanisme de drag and drop : il suffit de glisser et déposer les Pokémons désirés depuis la liste jusqu'à l'équipe sélectionnée au préalable. Une fois celle-ci pleine, il n'est plus possible de la compléter. Par ailleurs, la suppression d'un Pokémon au sein d'une équipe se fait par l'intermédiaire d'un bouton.

Nous avons pensé l'application de façon à ce qu'elle soit utilisée pour la stratégie Pokémon. En d'autres termes, elle doit permettre à chaque utilisateur de pouvoir enregistrer toutes ses différentes équipes (Ultra used, Offensiv bulk..), mais aussi de faire des tests en accèdant aux statistiques de nimporte quel Pokémon en un clic.

2 Précisions sur le Diagramme de classe

— **Package Métier :** Toutes classes souhaitant être affichée en tant que Pokémon doit implémenter cette interface. Ici, ce sont donc Pokémon et PokémonBin.

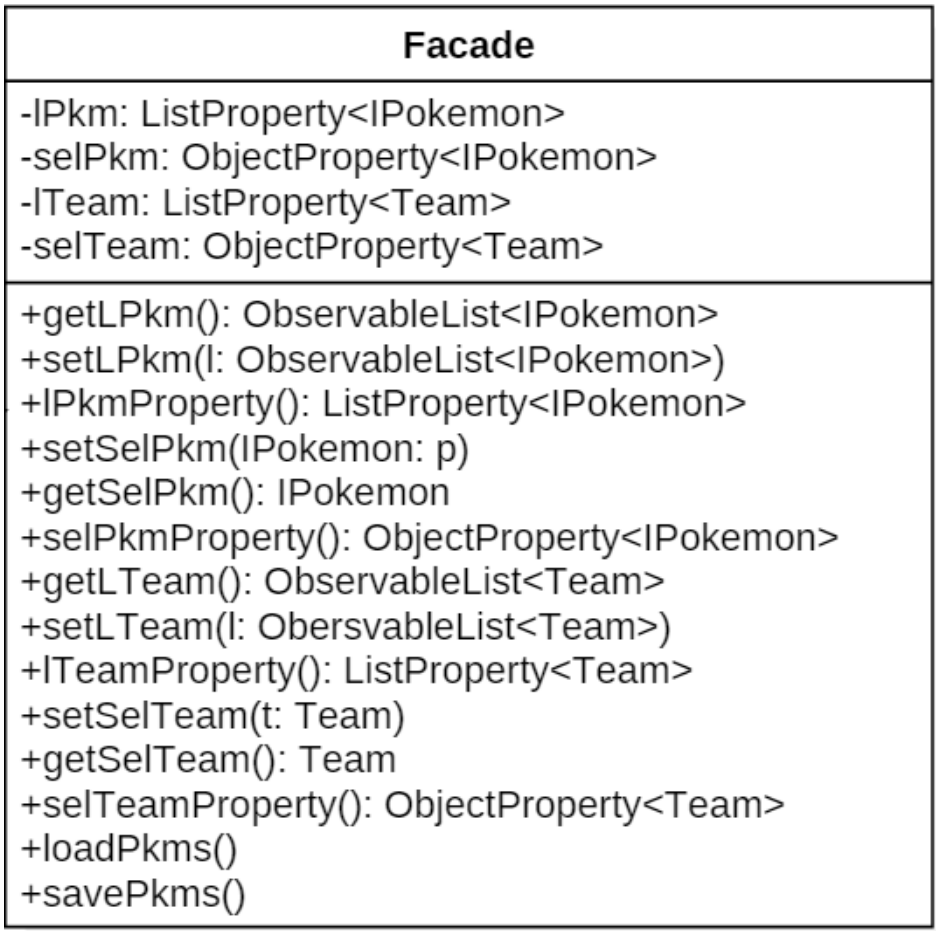
— **Package Comparator :** Les 3 comparateurs s'applique à des Pokémons afin de les trier selon 3 critères (id, nom, types) dans la liste où ils se situent.

— **Cell factories :** Elles permettent de mettre en forme le contenu de nos ListView. Par exemple, ce sont donc les données des Pokémons et des teams qui en bénéficient dans notre application.

— **Package Controller :**

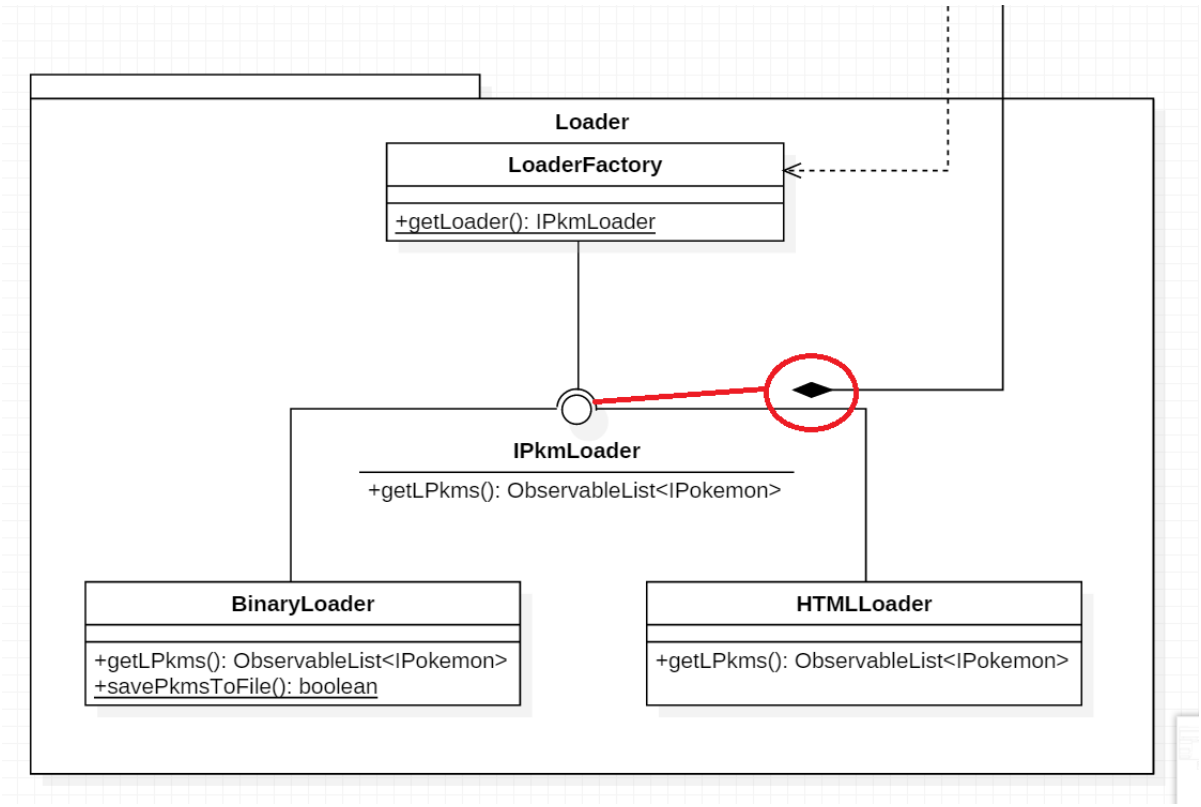
1. DexController : Gére les actions de la fenêtre principale.
2. TeamNameController : Gére la création/modification des équipes.
3. StatsController : Gére le tableau de statistiques du Pokémon sélectionné.

3 Patron "Façade"



Dans le cas présent, la façade est le point d'accès à l'environnement de notre application. Elle joue donc le rôle d'interface simple, de niveau supplémentaire dans la hiérarchie pour faciliter l'utilisation de notre sous système.

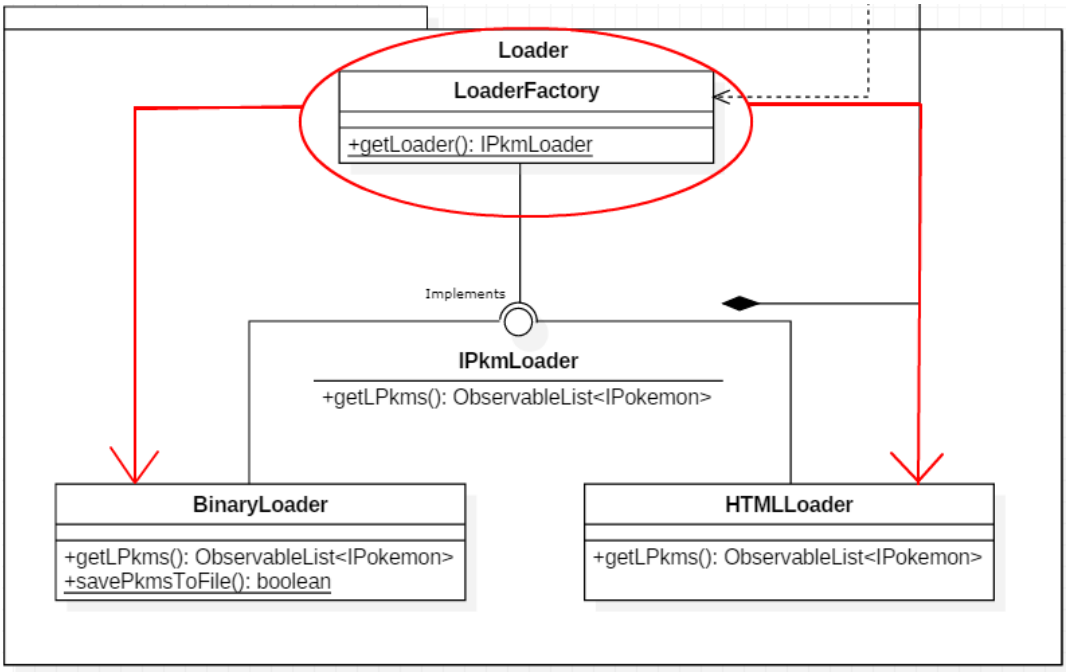
Le constructeur de la façade initialise les listes observables : liste de Pokémons et liste de Teams. Par conséquent, elle est composée d'une fonction permettant d'obtenir le loader souhaité et de charger la liste de Pokémons à partir de ce dernier. Elle donne aussi accès au pokémon et à l'équipe selectionnée dans la vue. Ces 4 objets sont des points névralgiques de l'application, leurs accès doivent être regroupés pour une utilisation plus aisée du code, et aussi pour faciliter une future extension. En effet, si l'on change le code derriere la façade, le binding des listview et autres composants de la vue sera toujours fonctionnel.



En effet, elle est directement liée par une composition au contenu du package "Loader" qui, lui, permettra par la suite de choisir la provenances des informations : fichier binaire ou web (HTML).

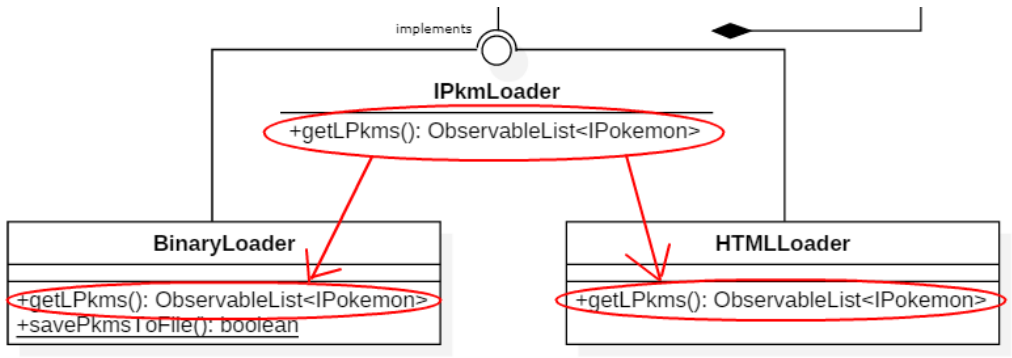
Comme vu ci-avant, notre façade possède des Pokémons par le biais de la liste récupérée précédemment. Elle a donc aussi accès aux méthodes contenues dans le contrat de l'interface "IPokémon".

4 Patron "Factory"



Nous avons aussi implémenté le patron factory (simple), déléguant ainsi l'instanciation du loader à la classe LoaderFactory. Ici, étant donnée qu'on ne peut pas prévoir la classe de l'objet qu'on aura à créer, la fabrique nous permet de résoudre rapidement et simplement ce problème. LoaderFactory possède la fonction "getLoader" qui fait le choix du type de loader en fonction de la présence d'un fichier binaire ou non.

5 Patron "Strategy"



Enfin, nous avions besoin de spécialiser le comportement des loaders. Le patron "Strategy" s'imposait donc. En effet, la méthode getLPkmn permettant de charger la liste de Pokémon ne se fait pas de la même manière selon le loader. Cette dernière est soit chargée depuis le Web (comprenant une constante (URL)), soit chargée à partir d"un fichier binaire. C'est pourquoi il nous fallait diverses variantes de cet algorithme. Les deux loader implémentent donc l'interface IPkmLoader contenant cette méthode.

6 Patron "Use Case Diagram"