

## Présentation de l'architecture

### I/Description de la façade de l'application :

Les classes « SportManager » et « UtilisateurManager » sont les classes qui regroupent respectivement les sports et les utilisateurs présents dans notre application. En effet, la classe « SportManager » possède une ObservableCollection nommée « LesSports » qui regroupe tous les sports alors que la classe « UtilisateurManager » possède une ObservableCollection nommée « LesUtilisateurs » regroupant tous les utilisateurs.

La classe « SportManager » possède une propriété de type « ISport ». Cette interface regroupe toutes les méthodes permettant l'ajout, la suppression et la mise à jour de sport dans notre application. Elle va donc implémenter la classe abstraite « Data » qui va posséder toutes les méthodes de l'interface « ISport ». Toutes ces méthodes sont abstraites puisque nous les définissons dans les classes « Stub » et « XML ». Nous ajoutons également la méthode « loadSport » dans la classe « Data » en abstraite de façon à la redéfinir dans les classes filles « Stub » et « XML » pour le chargement des Sports. Ainsi, le Stub permet le chargement « en dur » des données concernant les sports alors que la classe XML lit le fichier « lesSports.xml » pour charger tous les sports dans l'application.

De la même façon que la classe « SportManager », la classe « UtilisateurManager » possède une propriété de type « IUtilisateur ». Cette interface regroupe toutes les méthodes permettant l'ajout, la suppression et la mise à jour des utilisateurs dans notre application. Elle aussi va donc implémenter la classe abstraite « Data » qui va posséder en plus des méthodes de l'interface « ISport », les méthodes de l'interface « IUtilisateur ». Toutes ces méthodes sont également abstraites puisque nous les définissons dans les classes « Stub » et « XML ». Nous ajoutons également les méthodes « loadUtilisateur » dans la classe « Data » en abstraite pour les redéfinir dans les classes filles « Stub » et « XML ». Ainsi, le Stub permet le chargement « en dur » des données concernant les utilisateurs alors que la classe XML lit le fichier « lesSports.xml » pour charger tous les utilisateurs dans l'application.

Ainsi, la classe « DataManager », permet le chargement des données dans les classes « SportManager » et « SportUtilisateur » grâce aux deux méthodes « loadData\_Sport » et « loadData\_Utilisateur ». Il est ainsi possible de charger les données provenant du Stub ou du fichier XML en modifiant deux lignes de codes.

### II/Description de l'aspect « interne » de l'application.

Nous avons donc vu précédemment que les classes « SportManager » et « UtilisateurManager » possèdent les listes de « sports » et d'« utilisateurs » de l'application.

La classe « Sport » possède une liste d'« actualité » concernant ce « sport » (avec titre, description, date et image). La classe « Sport » possède également une liste de « championnat » (par exemple, le Rugby possède le Top 14 et la ProD2 en championnat).

La classe « Championnat » possède un « Classement » correspondant à la dernière journée du championnat. Elle possède aussi une liste d'éléments de la classe « Journée Sportive » et de la classe « Equipe ».

La classe « Classement » possède elle aussi une liste triée des équipes en fonction du nombre de points.

La classe « Journée » possède une liste de rencontres sportives. Chaque rencontre sportive s'effectue entre deux équipes, pour un sport donné, un championnat donné et avec un score pour chaque équipe.

La classe équipe possède enfin une liste de joueurs ayant un nom, un prénom et un poste. La classe équipe possède un dictionnaire de « Commentaires » avec un « utilisateur » comme clé et un « Commentaire » en valeur.