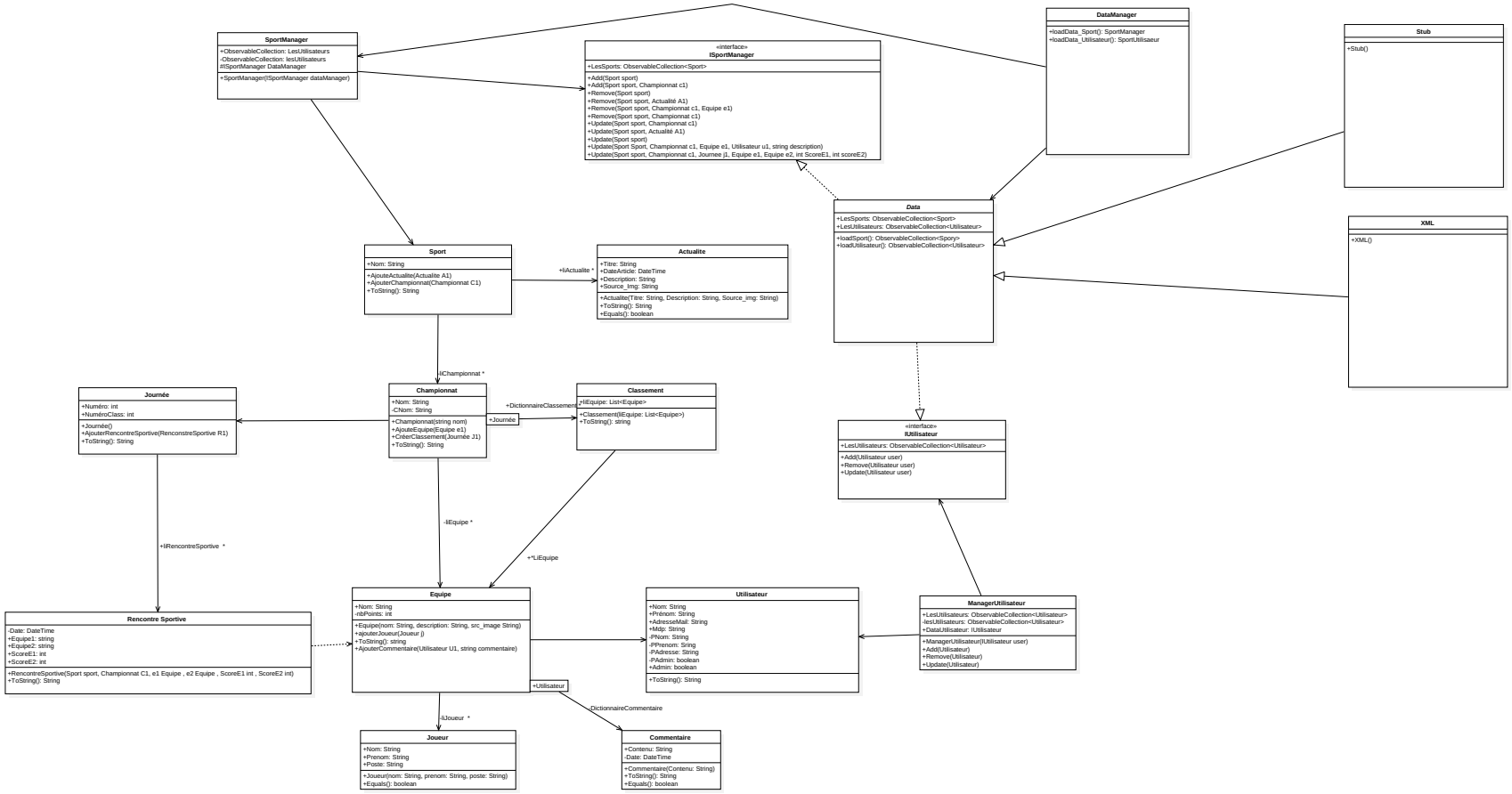
Model::Main


SportManager
+ObservableCollection: LesUtilisateurs
-ObservableCollection: lesUtilisateurs
#ISportManager DataManager
+SportManager(ISportManager dataManager)
«interface»
ISportManager
+LesSports: ObservableCollection<Sport>
+Add(Sport sport)
+Add(Sport sport, Championnat c1)
+Remove(Sport sport)
+Remove(Sport sport, Actualité A1)
+Remove(Sport sport, Championnat c1, Equipe e1)
+Remove(Sport sport, Championnat c1)
+Update(Sport sport, Championnat c1)
+Update(Sport sport, Actualité A1)
+Update(Sport sport)
+Update(Sport Sport, Championnat c1, Equipe e1, Utilisateur u1, string description)
+Update(Sport sport, Championnat c1, Journee j1, Equipe e1, Equipe e2, int ScoreE1, int scoreE2)
DataManager
+loadData_Sport(): SportManager
+loadData_Utilisateur(): SportUtilisaeur
Stub
+Stub()
Data
+LesSports: ObservableCollection<Sport>
+LesUtilisateurs: ObservableCollection<Utilisateur>
+loadSport(): ObservableCollection<Spory>
+loadUtilisateur(): ObservableCollection<Utilisateur>
XML
+XML()
Sport
+Nom: String
+AjouteActualite(Actualite A1)
+AjouterChampionnat(Championnat C1)
+ToString(): String
+lIActualite *
Actualite
+Titre: String
+DateArticle: DateTime
+Description: String
+Source_Img: String
+Actualite(Titre: String, Description: String, Source_img: String)
+ToString(): String
+Equals(): boolean
+lIChampionnat *
Journée
+Numéro: int
+NuméroClass: int
+Journée()
+AjouterRencontreSportive(RenconstreSportive R1)
+ToString(): String
Championnat
+Nom: String
-CNom: String
+Championnat(string nom)
+AjouteEquipe(Equipe e1)
+CréerClassement(Journée J1)
+ToString(): String
+Journée
+DictionnaireClassement
Classement
-lIEquipe: List<Equipe>
+Classement(IEquipe: List<Equipe>)
+ToString(): string
«interface»
IUtilisateur
+LesUtilisateurs: ObservableCollection<Utilisateur>
+Add(Utilisateur user)
+Remove(Utilisateur user)
+Update(Utilisateur user)
+lIRencontreSportive *
-lEquipe *
+*LiEquipe
Rencontre Sportive
-Date: DateTime
+Equipe1: string
+Equipe2: string
+ScoreE1: int
+ScoreE2: int
+RencontreSportive(Sport sport, Championnat C1, e1 Equipe , e2 Equipe , ScoreE1 int , ScoreE2 int)
+ToString(): String
Equipe
+Nom: String
-ntbPoints: int
+Equipe(nom: String, description: String, src_image String)
+ajouterJoueur(Joueur j)
+ToString(): string
+AjouterCommentaire(Utilisateur U1, string commentaire)
+Utilisateur
Utilisateur
+Nom: String
+Prénom: String
+AdresseMail: String
+Mdp: String
-PNom: String
-PPrenom: Sring
-PAdresse: String
-PAdmin: boolean
+Admin: boolean
+ToString(): String
ManagerUtilisateur
+LesUtilisateurs: ObservableCollection<Utilisateur>
-lesUtilisateurs: ObservableCollection<Utilisateur>
+DataUtilisateur: IUtilisateur
+ManagerUtilisateur(IUtilisateur user)
+Add(Utilisateur)
+Remove(Utilisateur)
+Update(Utilisateur)
+lJoueur *
-DictionnaireCommentaire
Joueur
+Nom: String
+Prenom: String
+Poste: String
+Joueur(nom: String, prenom: String, poste: String)
+Equals(): boolean
Commentaire
+Contenu: String
-Date: DateTime
+Commentaire(Contenu: String)
+ToString(): String
+Equals(): boolean

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 nommé « LesSports » qui regroupe tous les sports alors que la classe « UtilisateurManager » possèdent une ObservableCollection nommé « 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 méhodes « 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).

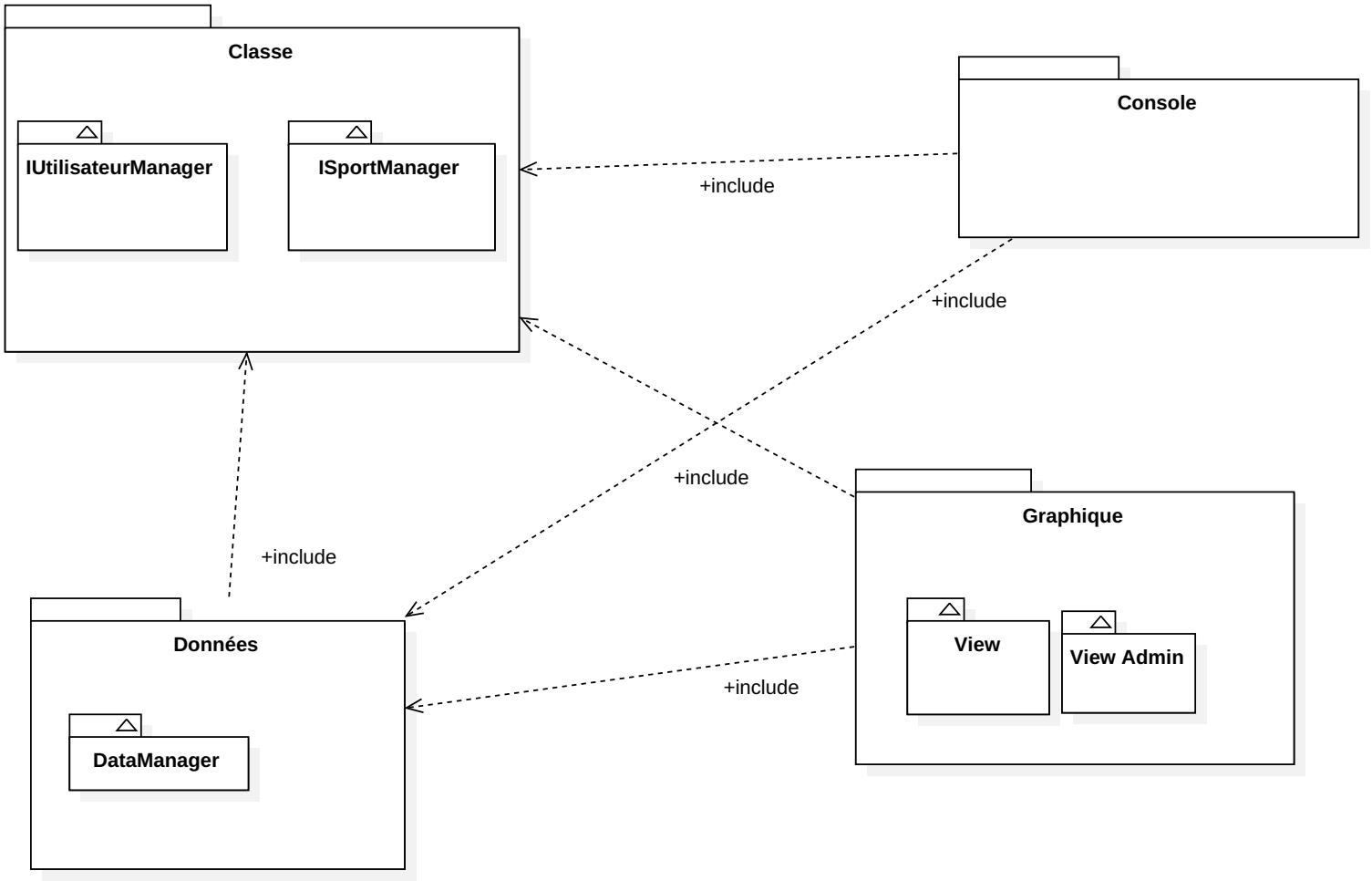
La classe « Championnat » possède un « Classement » correspondant à la dernière journée du championnat. Elle possède aussi une liste d'élément de la classe « JournéeSportive » et de la classe « Equipe ».

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

La classe « Journée » possède une liste de rencontre Sportive. 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 joueur ayant un nom, un prénom et un poste. La classe équipe possède un dictionnaire de « Commentaire » avec un « utilisateur » comme clé et un « Commentaire » en valeur.

Model::Main


Classe
IUtilisateurManager
ISportManager
Console
+include
+include
+include
+include
Graphique
View
View Admin
+include
Données
DataManager