École Supérieure d'Ingénieurs Léonard-de-Vinci

Problème Programmation Orientée Objet

Daniel JEAN & Marwan MENAA

Chargé de TD : Safwan CHENDEB

TD K

Page

Chapitre 1

Organisation & UML

1.1 Construction de l'UML

Au cours de ce projet nous avons compris **l'importance** d'un diagramme UML clair et précis. Pour construire cette "pièce maitresse" nous avons choisi d'utiliser le logiciel *StarUML*. Aussi, c'est un diagramme que nous avons actualisé à plusieurs reprises, **la construction du projet n'est pas unique**. Pourtant, nous avons dû faire des choix décisifs lors des fondations de ce projet.

1.1.1 Club & Statistiques

La classe *Club* est dite "principale". En effet, celle-ci gère les interactions entre la plupart des classes. Nous y faisons référence constamment dans notre WPF par exemple, afin de réaliser les différentes opérations demandées. Nous avons choisi de liée cette classe à une interface *IClub*. Une base permettant de définir précisément quelles méthodes doivent être implémentées dans un club afin de réaliser toutes les opérations nécessaires.

La classe *Statistiques* permet de faire le point sur la situation du club , sur les différents membres. Cette classe est donc directement associée à la classe *Club*.

1.1.2 Membres & Équipes

Les membres sont des personnes. Aussi, nous avons décidé de créer une classe *Personne* abstraite afin de définir les attributs communs à toutes les personnes. Ainsi, la classe *Membre* sert de base pour la création des membres du club, des salariés (*Administratif*) ainsi que des entraineurs (*Entraineur*).

Nous avons fait le choix de définir comme membre toutes les personnes interagissant dans le club. En particulier les salariés et les membres administratifs sont dispensés de cotisation à l'inverse des membres compétions (*MembreCompete*) et des membres loisirs (*MembreLoisir*)

1.1.3 Événements

La classe *Evenement* permet de créer des cours (*Cours*), des stages (*Stage*), des compétitions (*Competition*). Cette classe regroupe alors tous les événements du club (*Club).*

1.2 Nos choix d'implémentations

- Toute personne du club est un membre car nous considérons qu'ils peuvent tous participer à des cours.
- Seul les membres s'inscrivant en loisir ou en compétition possèdent un numéro de licence attribué lors de l'inscription leur permettant d'être assurés et d'établir les statistiques du club.
- Un entraineur ou plusieurs entraineurs peuvent faire partie de différentes équipes peu importe le niveau requis et ainsi participer à tout type de compétitions.
- Le règlement de la cotisation pour les membres se fait physiquement auprès du secrétaire. Ainsi, celui-ci pourra à l'aide du logiciel s'assurer que tout le monde a payé sa cotisation en fonction du code postal du membre.
- Nous avons fait le choix pour les membres salariés et non salariés ainsi que les membres administratifs de réitérer 3 fois l'instanciation : "iban" et "salaire/rémunaration" car cette opération n'est répétée que 3 fois.
- La classe *Evenement* n'est pas abstraite car nous considérons qu'en plus des stages et des compétitions, les membres administratifs peuvent créer différents types d'évènements.
- La classe *Resultat* est abstraite mais ne contient pas d'attributs, ni de méthodes car les classes *ResultatJoueur* et *ResultatEquipe* contiennent peu d'attributs communs.

1.3 UML

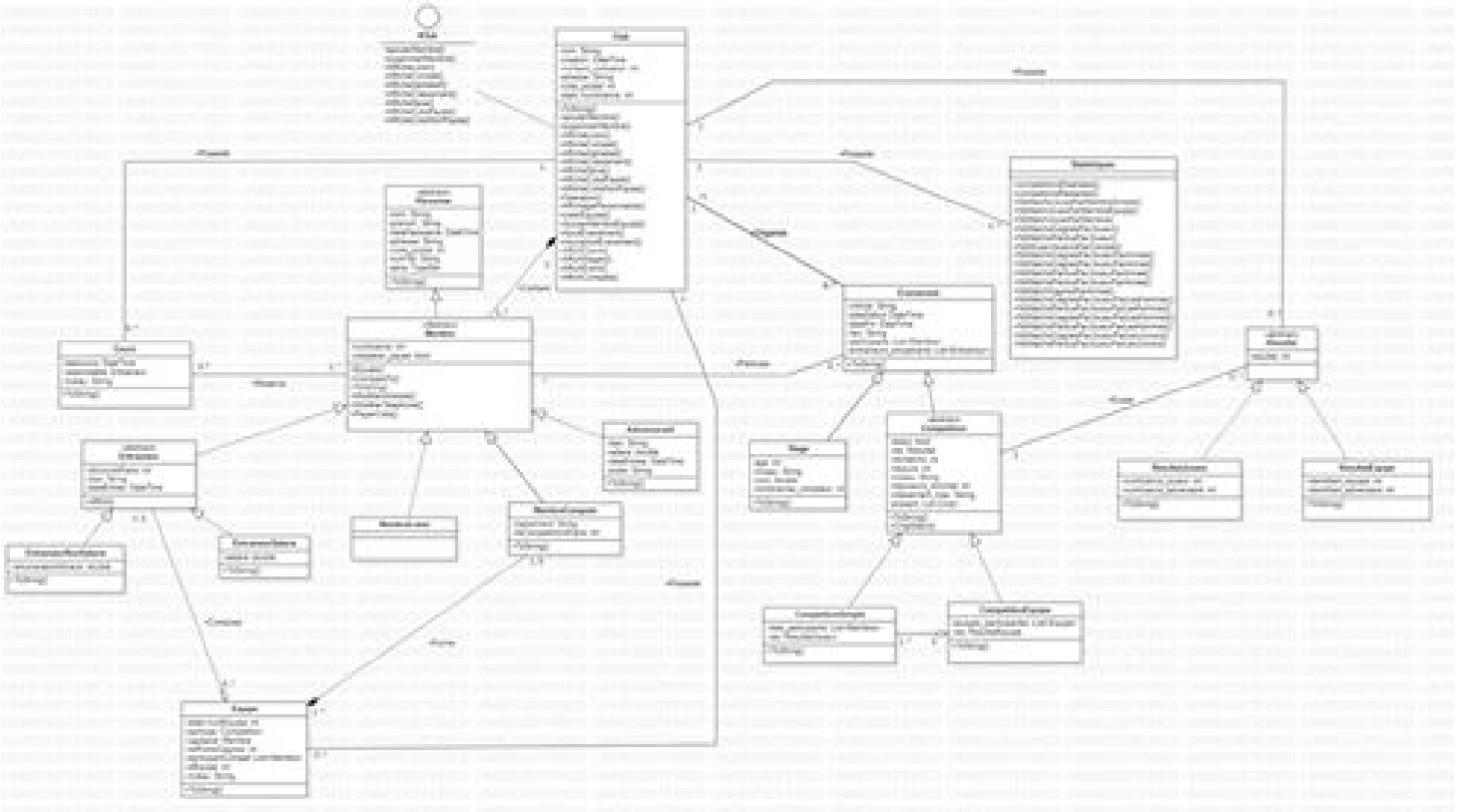

Figure 1.1 – UML

Chapitre 2

Structures C#

2.1 Utilisation des héritages

— La classe *Membre* hérite de la classe *Personne*
— Les classes *MembreCompete*, *MembreLoisir*, *Entraineur* et *Administratif*, héritent de la classe mère *Membre*
— Les classes *EntraineurNonSalarie* et *EntraineurSalarie* héritent de *Membre*
— Les classes *Stage* et *Competition* héritent de la classe *Evenement*

2.2 Classes Abstraites

Nous avons défini les classe suivantes comme abstraites :
— La classe *Personne*
— La classe *Membre*
— La classe *Entraineur*
— La classe *Competition*
— La classe *Resultat*

2.3 Interface

2.3.1 Création & Utilisation

La création d'une interface *IClub* associée à *Club* nous a paru essentielle pour la création d'un nouveau club de tennis. Ainsi, dans la dite interface, nous avons intégré toutes les méthodes nécessaires à la création d'un nouveau club de la même façon que *Club*).

2.4 Délégation

Dans *Club*, la délégation *Operation* associée à la méthode *AffichagePersonnalise(Operation o)* simplifie l'appel des méthodes d'affichages demandées dans le cahier des charges. Aussi, les fonctions anonymes nous ont été d'une aide précieuse pour une recherche plus rapide, efficace.

2.5 Collections Génériques

A travers les différentes classes nous avons utilisés bon nombre de collections génériques comme des listes d'entraineurs, de membres, d'événements, de résultats, et différents types de listes, capital pour la création de ce projet.

Conclusion & Perspectives

— Conclusion :

La réalisation de ce projet a mis à l'épreuve nos compétences en C#. Ainsi, nous avons réutilisé l'ensemble de nos connaissances apprises ces derniers mois.De plus, travailler à distance n'était pas chose aisée. L'utilisation de GitHub à alors grandement facilité : notre travail d'équipe à distance, la répartition des tâches et de fait la gestion du projet. Aussi, nous réalisons à quel point les fondations du code avec l'UML ont été des moments **très** importants et déterminants pour la suite du projet. Dans l'ensemble, nous sommes satisfaits de notre projet, la réalisation du WPF a été laborieuse pour autant nous y sommes parvenus malgré nos faibles connaissances de l'outil. Nous nous réjouissons de cela et avons hâte d'en découvrir davantage sur le sujet.

— Perspectives :

Nous aurions souhaité approfondir ce projet. Cependant, le temps imparti ne nous l'a pas permis. Aussi, nous avons réfléchi aux différentes perspectives d'améliorations possibles de notre programme :

- Ajouter un certificat médical pour chaque membre d'une durée validité de 3 ans
- Simuler le déroulé d'une compétition
- Ajouter une classe Sponsor
- Gérer des Arbitres
- Faire un planning pour la gestion du bar par les membres bénévoles
- Gérer le covoiturage entre les membres pour les compétitions extérieurs
- Étoffer nos méthodes de classes
- Utiliser du "DataBinding" pour un code plus efficace
- Réaliser un WPF "Responsive"
- Lier une base de données au programme afin de stocker les modifications apportées