Rapport de projet POO

Diagramme UML :

Nous sommes partis d’un club ayant pour attribut un nom, une ville, et une liste de membres. Ce club peut organiser un évènement de type compétition ou évents, il y a donc deux classes qui héritent d’évènements.

Il existe deux types de compétitions : les compétitions individuelles composées uniquement de matchs simples et les compétitions par équipe composées de matchs simples et doubles.

Les membres et le personnel héritent de la classe identité. Les membres peuvent être soit des joueurs loisirs, soit des joueurs compétition. Dans le personnel, il existe un attribut booléen entraineur\_est\_joueur qui est true si le joueur est un entraineur. A ce moment-là, un joueur avec les attributs de l’entraineur est créé. La classe membre hérite d’Incomparable afin que l’on puisse Comparer les membres par nom, prénom, naissance (du plus jeune au plus vieux) et ceux qui ont payé de ceux qui n’ont pas payé.

Dans la fenêtre Liste\_membre on utilise donc des délégations pour trier et afficher les joueurs.

Interface :

**Module membre :**

**Dans ce module il y a 5 Boutons pour effectuer des actions différentes :**

* **Ajouter un membre**

Sur cette page, on va pouvoir ajouter un membre à la base de données du club en remplissant tous les champs dans la fenêtre et en choisissant le type de cours (compétition / loisirs).

* **Supprimer un membre**

Sur cette page, il est possible de supprimer une personne de la base de données du club seulement en remplissant son nom, son prénom et son type de cours.

* **Modifier un membre**

Sur cette page, on peut modifier certaines informations sur un joueur juste en remplissant les cases où l’on veut modifier les informations.

* **Liste membre**

Sur cette page, on peut choisir d’afficher les joueurs Compétition ou les joueurs Loisirs en cliquant sur le bouton approprié. De plus, on peut afficher la liste de tous les membres dans ce même module et les trier si on clique sur le bouton nom.

On peut trier par ordre alphabétique des Noms ou des Prénoms ou alors du plus jeune au plus âgé ou encore par Sexe en appuyant sur le bouton respectif. Cela va être valable pour les 2 types de joueurs et ils seront triés indépendamment.

* **Liste Personnel**

Pareil que membre mais pour le personnel.

**Module Statistiques :**

Dans ce module, on va avoir à disposition les statistiques globales du club par année en appuyant sur l’un des 2 boutons.

Si on appuie sur Résultats par joueur, on va avoir accès aux résultats individuels de chaque joueur compétition du club et donc voir le nombre de matchs joueurs, ceux gagnés et ceux perdus.

Si on appuie sur Résultats du club par année on va voir le nombre de matchs joués et le nombre de ceux gagnés et ceux perdus.

**Module Compétitions :**

Dans ce module, il y a 2 boutons afin de créer une compétition individuelle ou par équipe. Pour cela il faut saisir les joueurs avec leur nom de famille en les séparant par des « / », il faut également rentrer toutes les informations demandées tel que le lieu, le nom, le nombre de joueurs minimum pour créer la compétition, etc…

Lorsque la compétition est créée, les matchs sont créés aléatoirement et de la même manière on définit qui gagne chaque match.

L’autre bouton va nous permettre d’afficher toutes les compétitions qui ont été créées et d’avoir les informations qui y sont associées.

**Module Autres :**

Dans ce module il y a 4 boutons qui sont :

* « Trouver un partenaire » qui permet de trouver un partenaire qui a le même âge si un partenaire a le même âge.
* « Commentaire » qui permet de mettre un commentaire que le club pourra lire dans un fichier texte.
* « Environnement – Tri sélectif » qui est une page sur le recyclage et sur ce que fais le club en la matière
* « Règlement intérieur » où l’on peut voir le règlement intérieur du club.

**Exigence des concepts à utiliser dans le code :**

* Nous avons utilisés l’interface Incomparable afin de classer par ordre alphabétique (CompareTo), de plus nous avons utiliser les delegate avec le ForEach() et le Sort().
* 2 Types de collection générique ont été utilisés : Les List<> et les SortedList<> (voir résultat club).
* Il y a plusieurs Abstract class (Evènement & Identité)
* Nous avons utilisé les héritages tout au long du code notamment Competiton\_simple qui hérite de Compétition qui hérite d’évènement
* Utilisation de System.IO pour la base de données (fichier .Txt).

**Problèmes rencontrés :**

* Nous ne savions pas comment ajouter des lignes dans un fichier texte déjà existant mais la solution était de mettre StreamReader(fichier,true)
* La gestion de l’entraineur qui fait parti de membre et de personnel (voir explications dans le diagramme uml)

**Bonus :**

* Il y a une musique pour rendre l’utilisation plus agréable, de plus l’arrière-plan est un dégradé, il y a aussi un certain nombre d’images afin de rendre l’interface plus sympathique.
* Lors de ce projet, nous avons utilisé GitHub ce qui est un plus afin de travailler ensemble.
* L’utilisation est fluide et intuitive grâce aux nombreux boutons dont « précèdent » qui permet de naviguer de page en page.