**Preuves**

***Je maîtrise les bases de la Programmation C# (classes, structures, instances…) :***

Nous avons créé de nombreuses classes dans plusieurs projet différents, par exemple *InformationsJeu* que l’on utilise énormément dans notre programme :

Une image contenant texte

Description générée automatiquement

On instancie un *InformationsJeu* dans la classe *Jeu*(ligne 26):

Une image contenant texte

Description générée automatiquement

Nous avons aussi créé des Enum, par exemple *Plateformes*:Une image contenant texte

Description générée automatiquement

***Je sais utiliser l’abstraction à bon escient (héritage, interfaces, polymorphisme) :***

Pour la persistance, de l’application, nous devions sérialiser la liste de jeux et le dictionnaire de franchise contenues dans *StockApp*. Pour cela, pour avons créé l’interface *IPersistanceStockApp*. Ainsi toutes les classes l’implémentant pourrais être utilisé comme « stockage » de données.

Une image contenant texte, intérieur, capture d’écran

Description générée automatiquement

Pour faire les tests, nous avons donc utiliser la classe *Stub*, qui implémente *IPersistanceStockApp*. La méthode *ChargeDonnees()* possédant des données utilisées seulement pour les tests :

Une image contenant texte

Description générée automatiquement

Mais pour le déploiement, nous avions besoin d’une persistance dans un fichier, c’est pourquoi nous avons fait la classe *PersistanceFichierTexte*, qui comme son nom l’indique, utilise un fichier texte pour charger et sauvegarder les données :

Une image contenant texte

Description générée automatiquement

*(ici on ne voit que la méthode ChargeDonnees(), mais il y en d’autres, cf Data/PersistanceFichierTexte)*

Ainsi, dans app, on peut changer facilement de « mode » de persistance (on pourrait en rajouter d’autre dans le futur : xml, bases de données…) tout en ne changeant pas le type statique de la propriété *Pers* d’*App.* Il suffit d’instancier la classe de persistance que l’on souhaite :

Une image contenant texte, moniteur, écran, noir

Description générée automatiquement

***Je sais gérer des collections simples ( tableaux, listes…) :***

Dans la classe *Jeu*, on a utilisé des collections très simples pour stocker les *Theories, les Musiques et les Visuels*. Nous avons préféré utiliser des *Set* car on ne veut pas de doublons et on n’a pas besoins qu’elles soient triées :

Une image contenant texte

Description générée automatiquement

***Je sais gérer des collections avancées(dictionnaires) :***

Pour stocker les *Franchise*, nous avons utilisé un dictionnaire. Chaque clé est donc une *Franchise* et chaque valeur associée est une *List<Jeu>*, qui sont donc les jeux d’une franchise.



***Je sais contrôler l’encapsulation au sein de mon application :***

Pour encapsuler nos données, nous avons utilisé des Propriété, qui sont donc publiques, d’où on peut récupérer les valeurs stockées dans les champs privés associé.

Exemple : Propriété publique *TousLesJeux* => Champ privé *tousLesJeux*

Une image contenant texte

Description générée automatiquement

***Je sais tester mon application :***

Nous avons un projet entier consacré aux tests de notre application, *ConsoleTest.* Cependant nous n’avons fait que des tests fonctionnels :

Une image contenant texte

Description générée automatiquement

*(ici il n’y en a que 3, pour les autres : cf ConsoleTest/Program)*

***Je sais utiliser LINQ :***

Nous avons, dans notre *Manager*, la liste de tous les jeux disponibles dans l’application. Cette liste est reliée à une propriété calculée qui permet d’effectuer la recherche de jeux dans la liste, ainsi que le tri. Pour faire cela on utilise LINQ, ce qui facilite grandement les choses :

Une image contenant texte

Description générée automatiquement

***Je sais gérer les évènements :***

Nous avons utilisé l’évènement *PropertyChanged* pour notifier lorsqu’une propriété avait changé, surtout dans le *Manager :*

Une image contenant texte

Description générée automatiquement

***Je sais choisir mes layouts à bon escient :***

Nous avons dû choisir nos layouts précisément pour que notre application ressemble à ce que nous avions prévu. C’est pourquoi, dans notre fenêtre *MainWindow* nous avons une *Grid.* En effet, nous voulions deux colonnes pour afficher le master (les franchises) à gauche de la fenêtre et le détail à droite (les jeux) :

Une image contenant texte

Description générée automatiquement

