Preuves des compétences

Conception et Programmation Orientées Objets

Documents

Je sais concevoir un diagramme de classes qui représente mon application [sur 9 points]
Se trouve dans le fichier *Documentation/Diagrammes UML/UML.mdj*, les classes mères sont maintenant au-dessus.

Je sais réaliser un diagramme de paquetages qui illustre bien l'isolation entre les parties de mon application [sur 2 points]
Il y a un début.

Je sais décrire mes deux diagrammes en mettant en valeur et en justifiant les éléments essentiels [sur 9 points]
Se trouve dans le fichier *Documentation/Rapports/C#_Tournaments.pdf*, la description du diagramme de classes est encore à compléter.

Programmation

Je maîtrise les bases de la programmation C# (classes, structures, instances...) [sur 2 points]
Beaucoup d'exemples dans toute la solution.

Je sais utiliser l'abstraction à bon escient (héritage, interfaces, polymorphisme) [sur 3 points]
Competitor et *Team* qui héritent de *Participant* qui est abstraite, utilisation d'une fabrique abstraite et autres exemples. Le polymorphisme est assez bien démontré dans *App.xaml.cs* où nous utilisons les types les plus hauts pour récupérer les sources des événements. **Note à mettre à jour.**

Je sais gérer des collections simples (tableaux, listes...) [sur 2 points]
Nous avons une liste dans *Team*.

Je sais gérer des collections avancées (dictionnaires) [sur 2 points]
Nous en avons une d'instanciée dans *Manager* mais qui est exploitée dans *TournamentCreation.xaml.cs* où il nous est nécessaire de filtrer la liste de participants à intégrer dans un tournoi sur sélection du sport dans la ComboBox.

Je sais contrôler l'encapsulation au sein de mon application [sur 5 points]
Nous avons essayé de garder un maximum d'éléments en *private* ou *internal* (notamment pour que la vue qui passe par la façade n'ait que les classes mères abstraites, les constructeurs de *TeamMatch* et *IndividualMatch* qui héritent de la classe abstraite *Match* sont en *internal*, seule *AbstractFactory* est visible à partir de la façade et c'est elle seule qui se charge d'appeler la bonne fabrique), tout cela pour essayer de garder l'implémentation protégée au maximum. **Note à mettre à jour.**

Je sais tester mon application [sur 4 points]

Nous avons testé tous les cas listés dans le diagramme de cas d'utilisation, à l'exception de l'impression de l'arbre qui ne peut se faire que pour la partie graphique.

Je sais utiliser LINQ [sur 1 point]

Nous avons utilisé LINQ aussi bien pour filtrer des listes dans *Tournament.cs* (région *Filters*, ligne 85) que pour caster des listes dans *TournamentCreation.xaml.cs* (ligne 79).

Je sais gérer les évènements [sur 1 point]

Nous en avons un, les classes sont définies dans l'assembly Facade pour qu'il puisse être à la fois testé par l'application console et utilisé par la vue. Il s'agit d'un événement qui ajoute une petite surprise à la saisie du nom de famille "BOUHOURS" pour un compétiteur.

Interface Homme-Machine

Documents

Je sais décrire le contexte de mon application pour qu'il soit compréhensible par tout le monde [sur 4 points]

Se trouve dans le fichier *Documentation/Rapports/IHM_Tournaments.pdf*.

Je sais dessiner des sketchs pour concevoir les fenêtres de mon application [sur 4 points]

Se trouve dans le fichier *Documentation/Rapports/IHM#_Tournaments.pdf*.

Je sais enchaîner mes sketchs au sein d'un story-board [sur 4 points]

Se trouve dans le fichier *Documentation/Rapports/IHM#_Tournaments.pdf*.

Je sais concevoir un diagramme de cas d'utilisation qui représente les fonctionnalités de mon application [sur 5 points]

Se trouve dans le fichier *Documentation/Diagrammes UML/UML.mdj*. Il est également expliqué dans */Documentation/Rapports/IHM_Tournaments.pdf*.

Je sais concevoir une application ergonomique [sur 2 points]

Nous avons essayé de faire au plus simple pour l'utilisateur en essayant de lui laisser le plus de choix possible en limitant les risques d'erreurs. Par exemple on lui laisse le choix du fichier de chargement ou de sauvegarde mais la création d'un participant lors de la création d'un tournoi est contrôlée (notamment au niveau du choix du sport). L'application reste intuitive et agréable à utiliser avec un style de fenêtre propre à nous, c'est-à-dire, des formes dans l'ensemble arrondies, que ce soit pour les fenêtres, ou pour les icônes de sports. Les informations essentielles sont visibles sur la page principale et l'utilisateur est guidé le plus possible dans la création des tournois et participants.

Je sais concevoir une application avec une prise en compte de l'accessibilité [sur 1 point]

Nous avons fait en sorte d'ajouter les actions valider à l'appui sur la touche *Entrée* à la création d'un tournoi, d'un participant ou encore l'ajout d'un commentaire, et la fermeture des fenêtres à l'appui sur la touche *Échap*.

Programmation

Je sais choisir mes layouts à bon escient [sur 1 points]

Validé à la première évaluation blanche, nous n'avons pas changé grand chose.

Je sais choisir mes composants à bon escient [sur 1 point]

Validé à la première évaluation blanche, nous n'avons pas changé grand chose.

Je sais créer mon propre composant [sur 2 points]

Il y en a quelques-uns dans le dossier *UserControls* du projet WPF, pour la barre d'état, les temps-forts d'un match, ou encore la liste des participants à un match pour une équipe.

Je sais personnaliser mon application en utilisant des ressources et des styles [sur 2 points]

Il y en a un paquet, que ce soit dans *App.xaml* ou dans les fenêtres directement. Ils se situent entre les balises *Resources*.

Je sais utiliser les DataTemplate (locaux et globaux) [sur 2 points]

Il y a des DataTemplate locaux dans *Home.xaml*, pour la ListView des tournois tout en haut (lignes 83 à 106), ou encore celle des sports tout à gauche (lignes 65 à 76). Nous avons également un DataTemplate global pour la ListView des participants qui se trouve dans *App.xaml* (lignes 25 à 36) car elle est présente dans dexu fenêtres (la création de tournoi ainsi que la création de participants).

Je sais intercepter les évènements de la vue [sur 2 points]

Il y a beaucoup d'exemples dans *App.xaml.cs*.

Je sais notifier la vue depuis des évènements métier [sur 2 points]

Match.cs implémente *INotifyPropertyChanged* car le détail de Match s'actualise même quand on passe les matches au tour suivant et qu'on reste sur la fenêtre de détail.

Je sais gérer le DataBinding sur mon master [sur 2 points]

Nous avons un master qui est la liste des sports à gauche, qui affiche une liste de tournois sur la barre du haut une fois le sport sélectionné.

Je sais gérer le DataBinding sur mon détail [sur 2 points]

La liste de tournois se met à jour à l'ajout d'un tournoi, et va à son tour afficher le tableau de tournoi correspondant au tournoi sélectionné.

Je sais gérer le DataBinding et les Dependency Property sur mes UserControl [sur 2 points]

Nous avons 2 exemples de binding avec les UserControls *ParticipantsListUserControl.xaml* où on bind l'image et le nom d'un participant et *HighlightUserControl.xaml* où il y a un binding sur la date du post et son contenu. Nous avons une DependencyProperty dans *StateBarUserControl.xaml*, qui ajoute un titre à côté de l'icône de l'application dans la barre d'état.

Je sais développer un Master/Detail [sur 2 points]

Oui, nous avons un master à gauche qui représente la liste des sports, qui va à la sélection afficher la bonne liste de tournois associée au sport sélectionné, et la sélection du tournoi affiche les détails du tournoi.

Projet Tuteuré

Documents

Je sais mettre en avant dans mon diagramme de classes la persistance de mon application [sur 4 points]

Se trouve dans le fichier */Documentation/Rapports/PTut_Tournaments.pdf*. Nous avons essayé d'expliquer au mieux pour expliquer le rôle de chaque classe mise en valeur...

Je sais mettre en avant dans mon diagramme de classes ma partie personnelle [sur 4 points]

Se trouve dans le fichier */Documentation/Rapports/PTut_Tournaments.pdf*. Nous avons essayé d'expliquer au mieux pour expliquer le rôle de chaque classe mise en valeur...

Je sais mettre en avant dans mon diagramme de paquetages la persistance de mon application [sur 2 points]

Se trouve dans le fichier */Documentation/Rapports/PTut_Tournaments.pdf*. Ici, ce sont les packages impliqués qui sont mis en valeur, avec une description qui explique quel package a quel rôle.

Je sais réaliser une vidéo de 1 à 3 minutes qui montre la démo de mon application [sur 10 points]

Il y a un raccourci vers une vidéo YouTube dans *Documentation/Video*.

Programmation

Je sais coder la persistance au sein de mon application [sur 4 points]

L'application utilise au premier lancement, les données du stub qui implémente l'interface *IDataManager*. À la fermeture de l'application, l'utilisateur peut choisir de sauvegarder ou non ces données. Cette sauvegarde (et au lancement suivant le chargement), sont gérés par le *XMLDataManager* qui implémente aussi *IDataManager*. La sauvegarde et le chargement des données sont gérés par le *Manager* qui fait appel aux méthodes du XMLDataManager. Les classes sérialisées sont décorées de DataContract, les propriétés à sérialiser sont bien décorées de DataMember, et les classes filles sont spécifiées avec KnownType.

Je sais coder une fonctionnalité qui m'est personnelle [sur 4 points]

L'arbre qui se génère automatiquement avec le binding en code behind avec l'algorithme pensé dans *Documentation/Rapports/PTut_Tournaments.pdf*. On le voit dans *TournamentView.cs* et *Tournament.cs*.

Je sais documenter mon code [sur 3 points]

J'ai commenté à peu près toutes les méthodes en laissant volontairement celles qui sont redéfinies ou qui sont des implémentations d'interface.

Je sais utiliser SVN. [sur 5 points]

La preuve se trouve sur la forge.

Je sais développer une application qui compile [sur 1 point]

Sauf conflit avec les .csproj, il n'y a aucun warning et aucune erreur. Il suffit de revert ses changements avant d'update.

Je sais développer une application fonctionnelle [sur 1 point]

L'application est fonctionnelle, l'ajout d'un tournoi se fait, sauf si un des champs n'est pas rempli, l'utilisateur en est notifié avec une MessageBox. Donc il faut tout renseigner, même une image (une par défaut qui se trouve dans l'assembly *Data*, dans le dossier Images).

Je sais mettre à disposition un outil pour déployer mon application [sur 2 points]

L'installeur est dans le dossier *Installeur*, où se trouvent notamment un dossier intitulé *Application Files* qui contient les dll déployées et l'installeur nommé *setup.exe*.

Je sais ajouter un Easter-Egg à mon application [points bonus]

Il s'agit de notre événement personnel, qui à la saisie du nom "BOUHOURS" définit lui-même le prénom pour un compétiteur. Cela ne marche donc pas pour les équipes. Il faut soit avoir sélectionné *TennisSimple* dans la ComboBox de la fenêtre de création de tournoi, soit créer un nouveau membre pour une équipe pour que cela fonctionne.