Perret Louis
Dallet Simon
3H-4H
IUT 2020/2021

Preuves

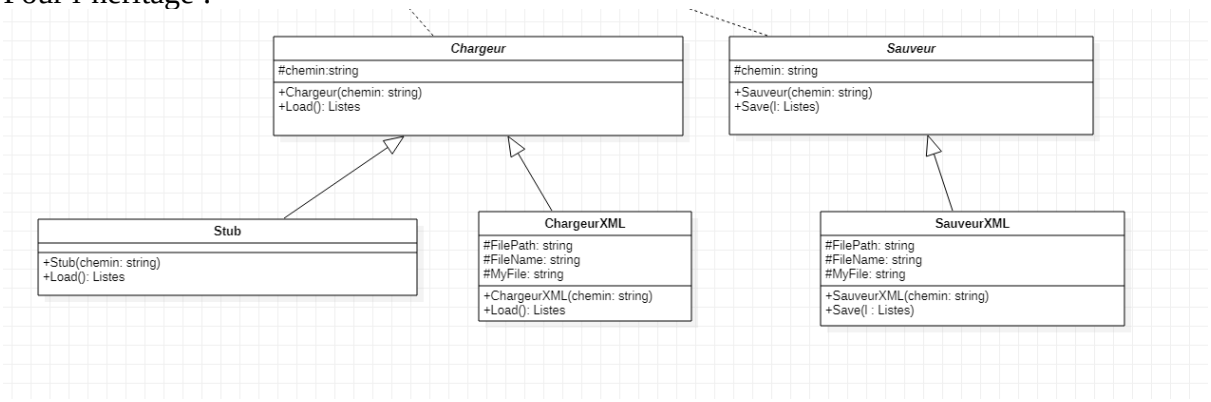
Objets 2 : Conception et Programmation Orientées Objets (C#, .NET)

Je maîtrise les bases de la programmation c# (classes, structures, instances…)

- Nous avons codé majoritairement des classes et une énumération GenreDispo. Nous avons aussi utilisé des string, des int et des DateTime.

Je sais utiliser l'abstraction à bon escient (héritage, interfaces, polymorphismes)

- Pour l'héritage :



- Et voici un bout de code dans App.xml.cs sur notre polymorphisme qui nous permettra facilement de changer de persistance.

```
1 référence
internal Chargeur Chargeur { get; set; } = new ChargeurXML("../../../XML");
2 références
internal Sauveur Sauveur { get; set; } = new SauveurXML("../../../XML");
4 références
```

Je sais gérer les collection simples(tableaux, listes….)

- Nous avons un tableau de notre énumération GenreDispo et une liste Manga contenu dans notre classe Compte, une liste Avis contenu dans notre classe Manga. Nous avons également une liste Compte et une liste de Genre dans notre classe Listes. Toutes ces collections sont gérées via des méthodes dans la classe qui les contiennent.

Je sais gérer des collections avancées (Dictionnaires)

- Attribut et propriété contenus dans notre classe Listes

```
3 références
private IDictionary<Genre, SortedSet<Manga>> collectionManga { get; set; }

11 références
public ReadOnlyDictionary<Genre,SortedSet<Manga>> CollectionManga { get; private set; } //Dictionnaire de tous les mangas
```

Je sais gérer l'encapsulation au sein de mon application

- La grande majorité de nos propriétés possèdent un private set. De plus nous avons utilisé des ReadOnlyCollection notamment pour notre collection de manga (voir screen précédent).
- Exemple d'une de nos méthodes qui fait les vérifications (méthode ModifierProfil au sein de notre classe Compte). Nous avons des méthodes similaires dans notre classe Manga.

```
1 reference
public void ModifierProfil(string newPseudo, GenreDispo[] genrePref, string imageName)
{
    if (String.IsNullOrEmpty(newPseudo))
    {
        throw new ArgumentException("Pseudo vide");
    }
    if (imageName == null)
    {
        imageName = "/Image;Component/Image/question.png";
    }
    Pseudo = newPseudo;
    GenresPreferes = genrePref;
    ImageProfil = imageName;
}
```

Je sais tester mon application

- Nous avons réalisé des tests fonctionnels et quelques tests unitaires.

Je sais utiliser linq

- Méthode RecupererGenre dans notre classe Listes

```
/// <summary>
/// Permet de recuperer le genre à partir de son nom de l'enum
/// </summary>
/// <param name="nomGenre">nom du genre dans l'enum</param>
/// <returns>g de la classe Genre</returns>
11 références
public Genre RecupererGenre(GenreDispo nomGenre)
{
    var genres = CollectionManga.Keys;
    var g = from genre in genres
            where genre.NomGenre.Equals(nomGenre)
            select genre;
    return g.FirstOrDefault();
}
```

Je sais gérer les évènements

- Utilisation des évènements NotifyPropertyChanged dans nos classes Listes, Manga et Compte.

IHM : Interface Homme-Machine (XAML, WPF)

Je sais choisir mes layouts à bon escient

- Un exemple avec notre MainWindow, on a utilisé une Grid pour la diviser en 4 parties distinctes. Puis des Stack Panel pour empiler les informations verticalement, et des WrapPanel pour empiler les informations horizontalement. (voir toutes nos fenêtres)

Je sais choisir mes composants à bon escient

- Exemple d'utilisation d'une ListBox sur notre MainWindow qui nous permet de sélectionner la collection de manga en fonction du genre cliqué dans la ListBox

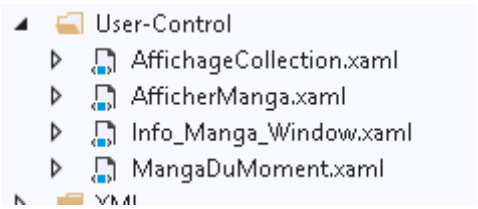
```
<ListBox x:Name="genreClick" Height="209" ItemsSource="{Binding ListeGenre}" HorizontalContentAlignment="Cen
    <ListBox.ItemTemplate>
        <DataTemplate>
            <WrapPanel>
                <TextBlock Text="{Binding NomGenre}" Style="{StaticResource ListBoxGenre}"></TextBlock>
            </WrapPanel>
        </DataTemplate>
    </ListBox.ItemTemplate>


</ListBox>
```

- Utilisation de boutons pour naviguer, valider les choix des utilisateurs lorsqu'ils utilisent nos fonctionnalités.
- Utilisation d'une CheckBox pour activer/désactiver le mode nuit.
- Utilisation d'un ContentControl afin de naviguer au sein de notre fenêtre principale en changeant nos User-Control

Je sais créer mon propre composants

- Création de 4 User-Control pour naviguer au sein de notre fenêtre principale.



Je sais personnaliser mon application en utilisant des ressources et des styles

- Style global à notre application

```
<Application.Resources>

    <Style TargetType="Button" x:Key="ButtonValider">
        <Setter Property="Content" Value="Valider"/>
        <Setter Property="Background" Value="#FF7FBFF7"/>
        <Setter Property="BorderBrush" Value="#FF728BCF"/>
    </Style>

    <Style TargetType="TextBlock" x:Key="ListBoxGenre">
        <Setter Property="HorizontalAlignment" Value="Center"></Setter>
        <Setter Property="Margin" Value="40,10,40,10"></Setter>
        <Setter Property="Padding" Value="45,10,45,10"></Setter>
        <Setter Property="Background" Value="#FF7FBFF7"/>

    </Style>

    <Style TargetType="TextBlock" x:Key="TextMargin">
        <Setter Property="Margin" Value="10,10,10,0"/>
        <Setter Property="FontFamily" Value="Century"/>
    </Style>


</Application.Resources>
```

- Style local dans notre fenêtre MainWindow.xaml

```
<Window.Resources>
    <Style TargetType="ListBoxItem" x:Key="ButtonGenre">
        <Setter Property="Background" Value="#FFC0E0FD"></Setter>
        <Setter Property="BorderBrush" Value="Black"></Setter>
        <Setter Property="Width" Value="100"></Setter>
        <Setter Property="Height" Value="30"></Setter>
        <Setter Property="Margin" Value="0,10,0,0"></Setter>
        <Setter Property="HorizontalContentAlignment" Value="Center"></Setter>
    </Style>
    <Style TargetType="TextBlock" x:Key="ProfilText">
        <Setter Property="Margin" Value="5"/>
    </Style>
</Window.Resources>
```

Je sais utiliser des DataTemplate (locaux, globaux)

- Voir screen précédemment sur notre ListBox pour DataTemplate local.
- Utilisation de DataTemplate local sur nos toutes nos ListBox, ComboBox, ListView et ItemsControl.

Je sais intercepter les éléments de la vue

- Voici quelque méthode qui sont appelées lors d'évènement de notre vue dans notre fenêtre MainWindow.xaml.

```
1 référence
private void Settings_button_Click(object sender, RoutedEventArgs e)
{
    var menuWindow = new Menu();
    menuWindow.M = this;
    menuWindow.ShowDialog();
}

1 référence
private void Selection_Changed_Genre(object sender, SelectionChangedEventArgs e)
{
    L.GenreCourant = genreClick.SelectedItem as Genre;
    L.ListeParGenre(L.GenreCourant);
    Navigator.NavigationTo(Navigation.UC_AFFICHAGE_COLLECTION);
}

1 référence
private void Home_Button_Click(object sender, RoutedEventArgs e)
{
    Navigator.NavigationTo(Navigation.UC_AFFICHAGE_MANGA_DU_MOMENT);
}
```

- Toutes nos fenêtres utilisent des évènements pour faire appel à nos fonctionnalités

Je sais notifier la vue depuis des évènements métiers

- Utilisation des NotifyPropertyChanged dans nos classes Listes, Compte et Manga lorsqu'une propriété change après l'appel d'une de nos fonctionnalités.
- Utilisation d'ObservableCollection dans Listes, Compte et Manga afin de notifier du changement d'éléments dans nos collections.

Je sais gérer le DataBinding sur notre Master

- Notre Master est la ListBox des Genres situé à gauche de notre fenêtre MainWindow.xaml : voir Screen page 3 sur notre ListBox pour le binding.
- Une fois un genre sélectionné, la liste des manga de ce genre apparaissent, c'est notre deuxième partie de notre Master : Binding qui permet d'afficher cette liste de manga

```
</StackPanel>
<ListBox x:Name="listeManga" DockPanel.Dock="Bottom" ItemsSource="{Binding ListeMangaCourant}" SelectionChanged="AffichageDuMangaSelectionne"  >
    <ListBox.ItemsPanel>
        <ItemsPanelTemplate>
            <WrapPanel></WrapPanel>
        </ItemsPanelTemplate>
    </ListBox.ItemsPanel>
    <ListBox.ItemTemplate>
        <DataTemplate>
            <User_Control:AfficherManga />
        </DataTemplate>
    </ListBox.ItemTemplate>
</ListBox>
```

Je sais gérer le DataBinding sur mon Detail

- Voir notre User-Control 'Info_Manga_Window' qui affiche toutes les informations d'un manga.

Je sais gérer le DataBinding et les Dependency Property sur nos User-Control

- Comme dit précédemment on bind sur nos User-Control pour afficher une partie de notre master et notre detail
- Pas eu besoin d'utiliser des Dependency Property

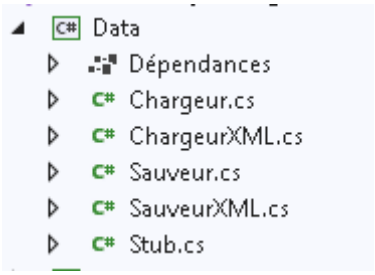
Je sais développer un Master/Detail

- Notre projet est fonctionnel donc oui
-

<u>Projet Tuteuré S2</u>

Je sais coder la persistance au sein de mon application

- Projet contenant nos classes utiles à la persistances + Utilisation des DataMember et DataContract sur nos propriétés et classes à sérialiser



- Utilisation des Hooks dans nos classes Listes et Compte.

```
[OnDeserialized]
0 références
void InitReadOnlyCollection(StreamingContext sc = new StreamingContext()) //Méthode qui est app
{
    CollectionManga = new ReadOnlyDictionary<Genre, SortedSet<Manga>>(collectionManga);
    List<Compte> lCompte = new List<Compte>();
    foreach(Compte c in listeCompte) //Permet d'éviter d'avoir un tableau dans notre IList
    {
        lCompte.Add(c);
    }
    listeCompte = lCompte;
    ListeCompte = new ReadOnlyCollection<Compte>(listeCompte);
    ChercherMeilleurManga();
}
```

Je sais coder une fonctionnalité qui m'est personnelle

- On a codé plusieurs fonctionnalités personnelles comme : GenreAuHasard, ChercherMeilleurManga, AjouterAvis contenues dans notre classe Listes. Voici un exemple avec GenreAuHasard

```
/// <summary>
/// Permet de choisir un genre au hasard
/// </summary>
2 références
public void GenreAuHasard()
{
    Array genreDispo = Enum.GetValues(typeof(GenreDispo));
    Random random = new Random();
    int index = random.Next(0, 4);
    GenreDispo gd = (GenreDispo)genreDispo.GetValue(index);
    GenreCourant = RecupererGenre(gd);
    ListeParGenre(GenreCourant);
}
```

Je sais documenter mon code

- Voir notre code

Je sais utiliser SVN

- Voir notre repository SVN sur la forge

Je sais développer une application qui compile

- Aucune erreur spécifiée par l'IDE et notre application compile

Je sais développer une application fonctionnelle

- Une fois lancée, notre application ne crash pas et nos fonctionnalités sont opérationnelles.

Je sais mettre à disposition un outil pour déployer mon application

- Publication + Installeur réalisés mais problème de crash une fois notre application installée et lancée.