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

Documents

- Je sais concevoir un diagramme de classes qui représente mon application [sur 9 points]

Voir Documentation C#

- Je sais réaliser un diagramme de paquetages qui illustre bien l'isolation entre les parties de mon application [sur 2 points]

Voir Documentation C#

- Je sais décrire mes deux diagrammes en mettant en valeur et en justifiant les éléments essentiels [sur 9 points]

Voir Documentation C#

Programmation

- Je maîtrise les bases de la programmation C# (classes, structures, instances…) [sur 2 points]

Classes: voir n'importe quel fichier .cs
Enum: voir LauncherName.cs
Instance : voir n'importe quel fichier .cs
autres : voir n'importe quel fichier .cs

- Je sais utiliser l'abstraction à bon escient (héritage, interfaces, polymorphisme) [sur 3 points]

Jeu et Launcher hérite tout les deux de Element qui est abstract(heritage). Nous avons une collection d'Element dans notre manager (polymorphisme). Nous avons une interface IPersistance qui assure que les classes qui l'implemente ait les fonctions suivantes: Load et Save(interfaces)

- Je sais gérer des collections simples (tableaux, listes…) [sur 2 points]

La plupart de nos collections sont des collections simples
Exemple : ligne 24 de Data.cs

- Je sais gérer des collections avancées (dictionnaires) [sur 2 points]

Nous nous servons de dictionnaires dans les classes OriginSearcher et RiotSearcher (paquet Modele)
Exemple : ligne 13 de OriginSearcher.cs

- Je sais contrôler l'encapsulation au sein de mon application [sur 5 points]

Le paquet Modele expose à l'extérieur uniquement les types et la façade(manager), tout le reste est soit private,protected ou internal.
Exemple : ligne 11 de Loader.cs

•Je sais tester mon application [sur 4 points]
Nous disposons de jeux de test unitaires et fonctionnels (projets « Test Fonctionnels » et « Test Unitaires »)

•Je sais utiliser LINQ [sur 1 point]

Nous utilisons beaucoup LINQ l'endroit ou on l'utilise le plus est la fonction de filtration dans la classe GameSearcher.

```
protected string Filter(string[] executables, string nom = null, LauncherName launcher = LauncherName.Autre)
{
    int archi = Environment.Is64BitOperatingSystem ? 64 : 32;
    if (launcher == LauncherName.Riot)//Riot est un peu speciale (peu de jeux)(launcher....etc) donc hardcodage de ceux la
    {
        return executables.First(e => e.Contains("LeagueClient.exe") || e.Contains("VALORANT.exe") || e.Contains("LoR.exe"));
    }
    IEnumerable<string> res = executables.Where(e => FilterIndesirables(e)); //apllication du filtre
    if (!res.Any()) //si tout a disparu dans le filtre
    {
        return executables[0];
    }
    else if (nom != null && res.Count() > 1) //si un nom est defini on prend les executables contenant le nom (si il y en a)
    {
        var temp = res.Where(e => Path.GetFileName(e).Contains(nom, StringComparison.OrdinalIgnoreCase));
        res = temp.Any() ? temp : res;
        temp = res.Where(e => Path.GetFileName(e).Contains(nom.Replace(" ", ""), StringComparison.OrdinalIgnoreCase));
        res = temp.Any() ? temp : res;
    }
    if (res.Count() > 1 && res.Any(e => e.Contains("bin", StringComparison.OrdinalIgnoreCase))) //preferer les exe contenu dans un dossier bin ou binaries (si il y en a)
    {
        res = res.Where(e => e.Contains("bin", StringComparison.OrdinalIgnoreCase));
    }
    if (res.Count() > 1 && res.Any(e => e.Contains(archi.ToString()))) //preferer les exe contenu dans un dossier 64 ou 32 bit en fonction de l'architecture supporté (si il y en a) (on
    {
        res = res.Where(e => e.Contains(archi.ToString(), StringComparison.OrdinalIgnoreCase));
    }
    return res.First();
}

protected bool FilterIndesirables(string executable)
{
    return !(executable.Contains("Unins", StringComparison.OrdinalIgnoreCase) || executable.Contains("Crash", StringComparison.OrdinalIgnoreCase) || executable.Contains("Helper", String
        || executable.Contains("AntiCheat", StringComparison.OrdinalIgnoreCase) || executable.Contains("Downloader", StringComparison.OrdinalIgnoreCase) || executable.Contains("Upload")
        || executable.Contains("Unreal", StringComparison.OrdinalIgnoreCase) || executable.Contains("Redist", StringComparison.OrdinalIgnoreCase) || executable.Contains("egodumper", Str
        || executable.Contains("mono.exe", StringComparison.OrdinalIgnoreCase)); //traitement des cas communs
}
```

•Je sais gérer les évènements [sur 1 point]

Nous utilisons les événements dans notre manager pour notifier la vue de changement, et dans Jeu pour notifier le changement d'une propriété (via l'événement NotifyPropertyChanged)
Exemple:ligne 84 de Manager.cs

IHM : Interface Homme-Machine (XAML, WPF)

Documents

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

Voir Documentation IHM

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

Voir Documentation IHM

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

Voir Documentation IHM

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

Voir Documentation IHM

•Je sais concevoir une application ergonomique [sur 2 points]

Explication dans la Documentation IHM

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

Explication dans la Documentation IHM

Programmation

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

Voir n'importe quel fichier .xaml
Exemple:ligne 20 de MainWindow.xaml

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

Voir n'importe quel fichier .xaml
Exemple:ligne 28 de MainWindow.xaml

•Je sais créer mon propre composant [sur 2 points]

Voir les fichiers contenus dans l'arborescente : « Projet\Vues\User Controls » et « Projet\Vues\ windowParts»

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

Voir App.XAML, ce fichier contient quelques styles utilisé dans des fenêtres comme AjoutDetailWindow ou AjoutJeuWindow
Exemple:ligne 17 de App.xaml

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

Nous n'avons pas de DataTemplate globaux
Nous utilisons des DataTemplate DetailLauncher.xaml, FolderExplorerView.xaml et MasterDetail.xaml

Exemple : ligne 18 de MasterDetail.xaml

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

Tous les boutons ont au moins une méthode abonné a l'événement click.
Une méthode est abonné à l'événement TextChanged de la TextBox de recherche afin d'actualiser la recherche a chaque caractère entré.
Exemple : ligne 59 MainWindow.xaml.cs

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

Nous utilisons les événements dans notre manager pour notifier la vue de changement, et dans Jeu pour notifier le changement d'une propriété (via l'événement NotifyPropertyChanged)
Exemple :ligne 84 de Manager.cs

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

Nous avons du DataBinding sur toutes nos fenêtres et UserControl
Exemple :ligne 15 de MasterDetail.xaml

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

Nous avons du DataBinding sur toutes nos fenêtres et UserControl
Exemple :Ligne 29 de DetailJeu.xaml

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

Nous utilisons pas de Dependency Property

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

Nous avons développé une application de type Master/Detail

Projet Tuteuré S2

Documents

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

Voir Documentation Projet Tutoré

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

Voir Documentation Projet Tutoré

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

Voir Documentation Projet Tutoré

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

Voir Documentation Projet Tutoré

Programmation

•Je sais coder la persistance au sein de mon application [sur 3 points]

Notre application sauvegarde les changement effectué par l'utilisateur.

•Je sais coder une fonctionnalité qui m'est personnelle [sur 3 points]

Voir Documentation Projet Tutoré

•Je sais documenter mon code [sur 2 points]

Nous avons documenter notre code
Exemple :GameSearcher.cs

•Je sais utiliser SVN. [sur 2 points]

Nous avons utiliser SVN

•Je sais développer une application qui compile [sur 3 point]

Notre application compile

•Je sais développer une application fonctionnelle [sur 5 point]

Notre application est fonctionnelle

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

Nous avons a disposition un outil pour déployer mon application