TRAVAIL PRATIQUE 1

420-4C4-JR

Évolution des applications

Travail individuel

30% de la note finale

Date limite de remise

27 mars 2021 à 23h59min

Faites la remise ***sur Léa***

Des pénalités de 10% par jour de retard seront imposées. En cas de retard, la remise doit se faire par message Mio.

Présentation

Dans ce travail, vous ferez la migration d'une application initiale, en mode console et utilisant des fichiers texte comme source de données, en la faisant doter d'interfaces graphiques utilisant des fichiers « .json » comme source de données. Vous allez donc créer une nouvelle version de cette application en vous basant sur l'ancienne.

Les fichiers de l'ancienne application console, d'extensions « .cs » et « .txt », en plus des fichiers « .json » ont été fournis dans le dossier contenant le présent énoncé (sous-dossier « Fichiers de travail »).

Consignes de migration

L'objectif est que l'application reste la plus fidèle possible à la version originale. Évidemment, les choses qui peuvent être améliorées doivent l'être, mais les fonctionnalités déjà présentent dans la version originale doivent être conservées dans la nouvelle application. Autrement dit, il ne faut pas réinventer l'application à votre goût.

Ainsi, vous devez faire des choix au niveau de la présentation des interfaces utilisateurs et de rendre l'application la plus fonctionnelle possible : choix des contrôles et de leurs propriétés, leurs positionnements, leurs organisations, la séquence d'exécution à suivre, les boutons à cliquer, les boites de messages à afficher, etc.

Vos choix doivent aussi respecter les restrictions suivantes :

1. Faites les modifications nécessaires pour que les données soient lues à partir des deux fichiers « .json » fournis : ListeJeuSociete.json et ListeCategorieJeu.json
    - Commentez clairement chaque modification afin d'expliquer ce qui est modifié et pourquoi. Commencez chaque modification avec l'expression « MODIF Q1 ».
2. Les classes « JeuSociete » et « CategorieJeu » doivent être conservées et utilisées dans la nouvelle application.
    - Il faudra donc les modifier le moins possible et seulement lorsque c'est nécessaire.
    - Commentez clairement chaque modification de ces classes afin d'expliquer ce qui est modifié et pourquoi. Pour les repérer facilement, commencez chaque commentaire avec le mot « MODIF ».
3. La fenêtre principale doit permettre de gérer les jeux :
    - Elle doit contenir 3 onglets. C'est au travers de cette fenêtre que les opérations de MAJ d'un jeu (par insertion, modification ou suppression) seront possibles.
    - Onglet 1 : permet d'afficher la liste des jeux dans un contrôle « DataGrid » et doit utiliser du « DataBinding ».
        - Utilisez un DataTemplate dans le DataGrid pour afficher la liste de catégories d'un même jeu.
        - Lorsqu'un jeu est sélectionné dans le DataGrid, ses détails sont affichés, dans le même onglet de la fenêtre, à l'aide de divers contrôles d'interface graphique sous une forme qui ressemble à un formulaire. C'est dans ce formulaire qu'il est possible de modifier les données.
        - Avant de finaliser la modification d'un jeu existant, l'application doit s'assurer que le nouveau nom n'existe pas déjà dans la liste des jeux, elle doit aussi vérifier le nombre minimal et maximal ainsi que le nombre de catégories associées au jeu (entre 1 et 5).
    - Onglet 2 : permet d'ajouter un nouveau jeu dans la liste des jeux et dans le fichier « .json ».

        - Avant de finaliser l'ajout d'un nouveau jeu, l'application doit s'assurer qu'il n'y a pas déjà un jeu qui porte le même nom, que le nombre minimal de joueurs est inférieur au nombre maximal et que le nombre de catégories associées au jeu est entre 1 et 5.
    - Onglet 3 : permet de supprimer un jeu de la liste des jeux et du fichier « .json ».
    - Les opérations de mises à jour (MAJ) par insertion, modification et suppression ne sont pas permises directement dans le DataGrid affichant la liste des jeux.
    - Utilisez des styles pour au moins 3 types de contrôles dont 1 avec clé.
        - Définissez un style global accessible dans toute l'application.
        - Définissez un style local à la fenêtre principale.
        - Définissez un style local au contrôle dans lequel il a été défini.

      Chacun de ces styles doit définir au moins 3 propriétés du contrôle visé.

      Ajoutez un commentaire commençant avec le mot « STYLE » avant la définition de chaque style.

4. Créez une deuxième fenêtre pour gérer les catégories des jeux.
    - Cette fenêtre devrait être modale, c'est-à-dire que tant qu'elle est affichée, l'écran principal ne sera pas accessible.
    - L'affichage de la liste de catégories doit se faire dans un contrôle « ListBox » et vous devez utiliser aussi le « DataBinding » pour le remplir.
    - Lorsqu'une catégorie est choisie dans la « ListBox », d'autres contrôles de l'interface utilisateur permettent d'afficher le nom et le nombre d'utilisations (voir plus bas) de chaque catégorie et de faire les mises à jour de cette catégorie (modification du nom, suppression d'une catégorie ou ajout d'une nouvelle catégorie).
    - Avant de finaliser l'ajout d'une nouvelle catégorie, votre application doit s'assurer qu'il n'y a pas déjà une catégorie qui a le même nom.
    - Avant de finaliser la modification d'une catégorie existante, votre application doit s'assurer que le nouveau nom n'existe pas déjà dans la liste des catégories.
5. À la fin de chaque opération (ajout, modification ou suppression) sur un jeu ou une catégorie, les fichiers « .json » sont réécrits.

Améliorations à apporter

On va profiter de la migration pour apporter des améliorations à l'application.

1. Certaines informations sont représentées sous forme d'un intervalle de valeur minimale et maximale.
    - Utilisez un nouveau type de données qui permettra de définir un intervalle de 2 entiers.
    - Seulement le minimum doit être obligatoire. Un maximum vide signifie qu'il n'a pas de limite.
    - Évidemment, la valeur du minimum doit être plus petite ou égale à la valeur du maximum.
    - Ce type de données doit être utilisé partout où c'est nécessaire.
2. Si on modifie une catégorie, cette modification doit être appliquée dans les jeux qui l'utilisent. De la même façon, si on supprime une catégorie, cette catégorie ne doit plus apparaître dans la liste des catégories associées à un jeu.
   Puisqu'il n'est pas possible d'avoir un jeu sans avoir au moins une catégorie, assurez-vous avant de supprimer une catégorie que cette suppression ne brise pas notre règle.
3. Dans la fenêtre de gestion des catégories, la ListBox doit afficher, pour chaque nom de catégorie, le nombre de fois qu'une catégorie est utilisée par les jeux. Cette information sera disponible pour chaque catégorie sélectionnée dans la liste et sera toujours correcte (mise à jour du nombre après chaque ajout, modification ou suppression d'un jeu).

Consignes de remise

- Au début de chaque fichier au format « .cs », « .xaml » ou « .xaml.cs » de votre projet, ajoutez un commentaire contenant votre nom et votre matricule.
- Les fichiers de données, de format « .json », remis avec la solution finale doivent contenir la même information reçue au début du TP et avoir le même format de données.
- Les interfaces graphiques doivent être claires et doivent respecter les standards habituels pour un GUI d'application dans l'environnement Windows. Toutes les fonctionnalités doivent être faciles à comprendre et à utiliser.
- La remise doit être faite sur Léa. Remettez un fichier « .zip » contenant le projet en entier (avec même les fichiers de données).

Barème

Migration :	
1- Utilisation du fichier « ListeJeuSociete.json » Utilisation du fichier « ListeCatgorieJeu.json » 2- Modifications et commentaires dans la classe « JeuSociete » Modifications et commentaires dans la classe « CategorieJeu »	10%
3- Fenêtre principale des jeux avec 3 onglets. - Onglet 1 : Affichage et modification fonctionnels ▪ DataGrid et Binding ▪ DataTemplate pour afficher les catégories ▪ Formulaire pour affichage et modification ▪ Vérification de la cohérence des nouvelles données d'un jeu avant de valider sa modification. - Onglet 2 : Ajout fonctionnel ▪ Contrôles nécessaires pour l'ajout d'un nouveau jeu ▪ Vérification de la cohérence des données d'un nouveau jeu avant de valider son ajout. - Onglet 3 : suppression fonctionnelle ▪ Contrôles nécessaires pour la sélection puis la suppression d'un jeu - Le contenu du DataGrid est toujours mis à jour en temps réel. - Au moins 3 Styles dont 1 avec clé (un style global, un style de fenêtre et un style local)	30%
4- Deuxième fenêtre pour les catégories - Fenêtre modale - Affichage des catégories dans un ListBox et utilisation du Binding - Le contenu du ListBox est toujours MAJ en temps réel. - Contrôles nécessaires pour afficher les données et opérations de MAJ - Vérification de la cohérence des données avant des MAJ - Ajout de catégorie fonctionnel - Modification de catégorie fonctionnelle - Suppression de catégorie fonctionnelle	20%
5- Sauvegarde des fichiers à la fin de chaque MAJ.	5%

Améliorations :	
1- Solution au problème d'intervalles Solution au problème de minimum et maximum	10%
2- Correction du problème de modification de catégories Correction du problème de suppression de catégories Vérification de la contrainte d'au moins une catégorie par jeu avant de supprimer un jeu	15%
3- Solution pour l'affichage du nombre d'utilisations d'une catégorie dans les jeux tout en respectant les critères de la POO. Ce nombre est toujours mis à jour.	10%

NB :

- Attention à la qualité du français, des pénalités peuvent être appliquées jusqu'à concurrence de 10% de la note.
- Le code doit être commenté adéquatement :
  - Commentez les parties les plus importantes de votre code.
  - Une pénalité allant jusqu'à 10% en cas de manque ou d'absence de commentaires dans les parties importantes du code.