C# - Les séries Animées

Consignes :

- Fournissez des solutions en code C# claires, concises, bien structurées et commentées
- Respectez les principes de la conception orientée objet et les bonnes pratiques de programmation en C# que nous avons analysée dans le cours
- Dessinez les diagrammes UML avec des outils comme looping
- N'hésitez pas à enrichir votre projet de vos propres connaissances sur le domaine

Rendu :

Le rendu sera transmis sous forme de dossier (Drive) qui devra contenir :

- Un document PDF donnant vos réponses (schéma et texte)
- Un fichier looping ou autre du ou des diagrammes
- Un sous-dossier permettant de lancer votre projet avec des exemples d'utilisation

Partie 1 : Conception orientée objet et modélisation

1- Modélisation des personnages

a. Concevez une hiérarchie de classes pour représenter différents types de personnes de séries animées. Commencez par une classe de base abstraite **Protagoniste** avec les propriétés **Nom**(string) et **SerieOrigine**(string) ainsi qu'une méthode abstraite **Interagir (Protagoniste autreProtagoniste**)

b. Créez au moins deux classes concrètes héritant de **Protagoniste** qui devront posséder au moins 2 propriétés et une méthode spécifique en plus de la méthode **Interagir**

c. Réalisez le diagramme de classe correspondant

2- Gestion des Saisons et des Episodes

a. Créez une classe **Saison** avec une propriété **Numéro**(int) et une liste d'objets **Episode**

b. Créez une classe **Episode** avec les propriétés **Titre**(string) et **DureeMinutes**(int)

c. Créez une classe **Serie** avec les propriétés **Titre**(string), **AnneeDebut**(int)**,** **Genre**(string) et une liste d'objets **Saison**

d. Ajoutez une méthode à la classe **Serie** nommée **CalculerDureeTotale ()** qui retourne la durée totale en minutes de tous les épisodes de toutes les saisons de la série

e. Réalisez le diagramme de classe correspondant

3- Interfaces

a. Définissez une interface **IDevenirPuissant** avec une méthode **GagnerEnPuissance (string raison**)

b. Faites en sorte qu'au moins une de vos classes de personnages implémente l'interface **IDevenirPuissant**. Implémentez la méthode **GagnerEnPuissance** pour modifier la propriété de puissance du personnage et afficher un message d'écrivant la raison de l'augmentation

Partie 2 : Manipulation de données et LINQ

1- Collections et Recherche

   a. Créez une **List<Serie>** contenant au moins cinq séries animées différentes, chacune avec plusieurs saisons et épisodes

   b. Utilisez LINQ pour trouver toutes les séries du genre « Action » dont l'année de début est antérieure à 2010. Affichez les titres de ces séries

   c. Utilisez LINQ pour trouver le titre de l'épisode le plus long de toutes les séries de votre collection

2- Regroupement et agrégation

   a. En utilisant la collection de séries, utilisez LINQ pour regrouper les séries par genre et afficher le nombre de séries dans chaque genre

   b. Utilisez LINQ pour calculer la durée moyenne des épisodes pour une série spécifique de votre collection

3- Sélection et projection

   a. En utilisant LINQ, créez une nouvelle collection anonyme contenant uniquement le titre de la série et le nombre total d'épisodes pour chaque série de votre collection ; Affichez le contenu de cette nouvelle collection

Partie 3 : Gestion des Exceptions et robustesse

1- Gestion d'exceptions spécifiques

   a. Considérez une méthode qui prend en entrée le titre d'une série et un numéro de saison et tente de retourner le nombre d'épisodes de cette saison. Gérez les exceptions suivantes de manière spécifique :

      i. Une exception sir la série avec le titre donnée n'existe pas

      ii. Une exception si le numéro de saison demandé est invalide (inférieur à 1 ou supérieur au nombre total de saisons de la série)

      iii. Une exception si la saison spécifie ne contient aucun épisode

   b. Pour chaque type d'exception, capturez-la et affichez un message informatif et retournez une valeur par défaut appropriée

2- Création et lancement d'exceptions personnalisées

   a. Définissez une classe d'exception personnalisée nommée **EpisodeManquantException** qui hérite de **Exception** et inclut une propriété **TitreSerie**(string) et **NumeroSaison**(int)

   b. Modifiez la méthode de la question précédente. Si la saison spécifiée ne contient aucun épisode, lancez une instance de votre exception personnalisée en fournissant le titre de la série et le numéro de la saison concernée. Dans le code utilisant cette méthode capturez cette nouvelle exception et affichez un message pertinent

Barème

L’examen est noté sur 80 ramené à 20.

Partie	Question	Sous-question	Points
1. Conception orientée objet et modélisation (24 points)	1. Modélisation des personnages (9 points)	a.	2
		b.	7
		c.	2
	2. Gestion des Saisons et des Épisodes (12 points)	a.	2
		b.	2
		c.	2
		d.	4
		e.	2
	3. Interfaces (3 points)	a.	1
		b.	1
		c.	1
2. Manipulation de données et LINQ (24 points)	1. Collections et Recherche (8 points)	a.	3
		b.	3
		c.	2
	2. Regroupement et agrégation (8 points)	a.	4
		b.	4
	3. Sélection et projection (8 points)	a.	8
3. Gestion des Exceptions et robustesse (32 points)	1. Gestion d’exceptions spécifiques (20 points)	a.	4
		b.	5
		c.	5
		d.	6
	2. Création et lancement d’exceptions personnalisées (12 points)	a.	5
		b.	5
		c.	2
TOTAL			80