TD 5_6 Héritage et Interface

Objectifs

L'objectif de ce TD est double, confirmer les acquis sur l'héritage approfondir la classe abstraite et être capable d'utiliser les interfaces proposées par C# et de créer ses propres interfaces

Exercice 1

On définit une figure par une aire, un périmètre et un point d'origine. En tant que telle, nous ne créerons jamais d'instance de la classe figure.

On définit un rectangle comme une figure, le point d'origine étant le coin haut droit.

On définit un cercle comme une figure dont le point d'origine est le centre.

Le rectangle et le cercle ont des aires et périmètres calculés de façon particulière.

- Définir le diagramme UML
- Ecrire les classes de telle sorte que nous puissions calculer l'aire et le périmètre d'un rectangle et d'un cercle dans le Main

Exercice 2 – Utiliser une interface prédéfinie

Reprendre la classe « salarié » vu au TD4 qui possède un numéro, un nom, un prénom, une date d'entrée dans l'entreprise et un salaire

Nous souhaitons créer une liste de salariés et être en mesure de pouvoir trier toute liste de salariés en fonction des noms. Vous retiendrez la méthode Sort de la collection générique List<T>.

Pour cela, la classe Salarie doit hériter de l'interface IComparable avec les obligations liées au contrat défini dans cette interface (int CompareTo(object val)).

Créer dans le programme principal, des salariés et une liste de salariés.

Trier les salariés par ordre des Noms

Exercice 3 : Créer sa propre interface avec un contrat

Créer une interface en vue de pouvoir l'appliquer sur toute classe nécessitant un calcul de moyenne.

Pour cela, vous allez créer une interface IMoyenne avec le contrat approprié.

Reprendre la classe Entreprise qui possède un nom et une collection de salariés.

Appliquer cette interface à la classe Entreprise pour calculer la moyenne des salaires de ses salariés.

Exercice 4 : Créer une interface avec un contrat sur une propriété

Il est souhaitable que chaque classe de notre application, afin de faciliter sa gestion, puisse identifier la clef de chaque instance de manière unique : à savoir le numéro du salarié pour la classe Salarié et l'identifiant de l'entreprise pour la classe Entreprise.

Pour cela, vous allez créer une interface IIdentifiable qui propose un contrat de propriété que vous ferez hériter aux 2 classes

Exercice 5 : Interface est type d'une variable

On propose de pouvoir comparer des objets de différentes classes au moyen d'une conversion vers les nombres entiers. Pour cela on va utiliser une interface avec la méthode de conversion.

```
interface IConvertible
{
    int Convert();
}
```

3-1 Ecrire la classe MaDate

avec ses 3 attributs jour, mois, année, son constructeur. La classe MaDate implémente cette interface. Celle-ci retourne le nombre de jours de l'instance courante (on suppose chaque année de 365 jours et chaque mois de 30 jours pour simplifier)

3-2 Dans la classe Program

- Ecrivez une méthode statique pour comparer si deux objets convertibles sont égaux
- Ecrivez une méthode statique pour rechercher le minimum d'un tableau d'objets convertibles

3-3 Dans le Main de la classe Program

Utilisez les méthodes écrites précédemment en utilisant les instances de la classe MaDate

Exercice 5 – Adapter ces interfaces sur une autre classe afin d'homogénéiser le comportement des classes d'une application

Pour comprendre l'intérêt des interfaces, vous les adapterez à la classe Voiture et la classe Garage

Chaque voiture sera définie par une plaque d'Immatriculation, une marque, un modèle et un prix d'achat.

Un garage sera défini par une N° de Siret et d'une flotte de voitures

Vous implémentez la méthode Moyenne sur les prix de la flotte de voitures et l'identification unique sur le N° de Siret pour le garage et la plaque d'immatriculation pour les voitures

Vous triez les voitures sur les prix par ordre décroissant