TD1- Conception et implémentation de POO

Objectifs du TD

- Maîtriser les notions de base : classe, objet, attribut, portée, constructeur, méthode.
- Savoir modéliser un programme orienté objet grâce à UML
- Bonnes pratiques de développement :
  - ✓ Commentaires
  - ✓ Nomination des attributs
  - ✓ Un seul return par méthode
  - ✓ Utiliser un jeu de test pour vérifier les fonctionnalités d'un objet

Exercice 1:

Soit les exemples d'informations suivant décrivant une voiture :
"Mercedes C250 mise en circulation en 2000 de couleur blanche."
"Peugeot 106 mise en circulation en 1997 de couleur rouge."
"BMW série 1 mise en circulation en 2008 de couleur gris."
Identifiez les informations qui caractérisent une voiture. Proposez une classe Voiture, qui se compose uniquement d'attributs, permettant de stocker les informations décrivant une voiture. Pour respecter le principe d'encapsulation, quelle portée doit-on attribuer aux attributs de la classe ?
Implémentez la classe Voiture

Exercice 2:

Ajoutez un constructeur à la classe Voiture. Le constructeur permettra d'instancier un objet ainsi :
Voiture V1;
p1 = **new** Voiture("Mercedes", "C220", **false**, 2000, "rouge");

Exercice 3:

La 3ième ligne de ce code est-elle valide ? Justifiez.
Voiture V1;
p1 = **new** Voiture("Mercedes", "C220", **false**, 2000, "rouge");
Console.WriteLine(p1.marque);

Exercice 4:

Pour accéder aux valeurs des attributs, il est possible de déclarer des propriétés. Par exemple, définissons
la propriété Marque (M majuscule) :

```
public String Marque
{
get { return marque; }
}
```
La propriété Marque permet d'obtenir la valeur de l'attribut modele. Il est alors possible d'écrire :

```
Voiture V1;
p1 = new Voiture("Mercedes", "C220", false, 2000, "rouge");
Console.WriteLine(V1.Marque);
```

Le code suivant est-il valide :

```
Voiture V1;
p1 = new Voiture("Mercedes", "C220", false, 2000, "rouge");
V1.Marque= "ferrari" ;
```

Proposez une propriété pour le modèle et l'année de mise en circulation. Proposez une propriété NbrAnnee, dont la valeur est le nombre d'année de la voiture. Le calcul se fera uniquement à partir de l'année en cours (2018).

Exercice 5:

Ajoutez à la classe Voiture une méthode int RetournerAgeEn(int anneeRef) qui renvoie l'âge de la voiture par rapport à une année de référence anneeRef. Quelle visibilité doit-on attribuer à la méthode ?

Exercice 6:

Ajoutez une méthode Boolean PlusVieuxQue(int ageRef) qui renvoie true si la voiture est plus vieille que ageRef. Ajoutez une méthode Boolean PlusVieuxQue(Voiture pRef) qui renvoie true si la voiture est plus vieille que la voiture pRef.

Exercice 7:

L'information su la couleur est, pour l'instant, en visibilité private. On ne peut donc pas accéder à la valeur couleur depuis l'extérieur de la classe. Que proposez-vous pour permettre l'accès en lecture ? L'accès en écriture peut aussi se justifier (une voiture ne change pas de marque mais peut changer de couleur).
Proposez une solution.

Exercice 8:

Proposez un second constructeur, permettant de ne pas définir la couleur, qui sera alors mis à une valeur par défaut : "inconnu".

Exercice 9:

Proposez un exemple d'utilisation de toutes les possibilités de la classe Voiture (: dans main.cs).

Exercice 9:

Ajoutez une méthode String message() qui renvoie une chaine de caractères de la forme " Voiture 1 : Mercedes C250 mise en circulation en 2000 de couleur blanche."

Exercice 10:

On souhaite développer un programme orienté objet (POO) de gestion des ventes dans un magasin. Pour se faire, vous aller créer une classe Produit permettant de représenter les produits vendus. Un produit est caractérisé par une référence qui permet de l'identifier de façon unique, un libellé, un prix hors taxe et une quantité en stock.
Les objets produits créés par la classe Produit peuvent assurer les fonctionnalités suivantes :

- Augmenter la quantité disponible de l'article
- Enregistre la vente d'un certain nombre d'unités de produit, pour cela vous diminuerez la quantité disponible en stock. Si le nombre d'unités en stock est supérieure à la quantité vendue, la vente est impossible.
- Calculer et renvoyer le prix TTC du produit
- Décrire le produit en exprimant la référence, l'intitulé et le prix
- Comparer 2 produits (2 articles sont identiques si et seulement si leur numéro de référence sont Identiques)

La gestion des produits se fait dans le magasin. Pour cela, vous allez créer une classe Magasin qui est caractérisé par un nom et un ensemble de produit. Dans le magasin, on peut ajouter un produit, on peut supprimer un produit et on peut afficher la liste des produits qui existent (en précisant la quantité).
T.A.F
1- Tracer le digramme de classe modélisant le programme.
2- Implémenter les classes en C#.
3- Tester le programme en créant plusieurs produits et en les ajoutant au magasin. Tester la suppression et l'affichage de la liste des produits.

Exercice 11:

On se propose de de créer en mode console un jeu de type MasterMind.
Le but du jeu :
Deux joueurs s'affrontent : un humain ou/et un ordinateur.
Un joueur choisit un code secret de plusieurs chiffres, l'autre doit le deviner en faisant le moins de propositions possibles.
Chaque chiffre (0 à 9) peut apparaître plusieurs fois ou non dans le code.
Quand le joueur qui joue fait une proposition de code, l'autre lui indique combien de chiffres sont bien placés et combien de chiffres sont mal placés dans le code secret.
Le jeu se joue en plusieurs manches, le rôle des joueurs est échangé à chaque nouvelle manche. A la fin de la partie, le gagnant est celui qui a trouvé les codes avec le moins de propositions.
T.A.F
1- Tracer le digramme de classe modélisant le jeu.
2- Implémenter les classes en C#.
3- Tester le jeu.

Exercice 12:

Dans un parc d'élevage de chien, le directeur veut créer un POO qui permet le suivi de l'état de santé des chiens.

Un chien est caractérisé par un nom, un poids et un poids de référence. Si le poids dépasse le poids de référence, il faut que l'animal fasse un régime.

Ecrire un constructeur pour la classe Chien qui a pour paramètre le nom de l'animal et le poids de référence. Le poids doit être initialisé de façon aléatoire (poids compris entre 3 et 15 Kg). Si le poids de l'animal dépasse le poids de référence, alors l'animal doit faire un régime, sinon l'animal ne doit pas faire un régime

Ecrire les propriétés permettant d'accéder aux différents attributs. Tester.

Ecrire une propriété qui permet de modifier le poids de l'animal. Tester.

Proposer une redéfinition de ToString() pour la classe Animal. Tester.

Ecrire une méthode Nourriture qui retourne le poids de la ration journalière d'un chien. La ration de nourriture est fonction du poids de l'animal. Il faut compter 20g de nourriture par kilos. Néanmoins si l'animal est au régime sa ration doit être diminuée de 15%. Tester.

Pour la gestion du parc, on se propose de créer une classe Parc qui contient plusieurs chiens. On peut ajouter un chien ou supprimer un chien. On veut également lister les chiens qui doivent suivre un régime et les chiens qui ne doivent pas suivre un régime

T.A.F

1- Tracer le digramme de classe modélisant la gestion du parc d'élevage.

2- Implémenter les classes en C#.

3- Tester le programme.