TD2 - Classes et objets C#

Exercice 1

Les exemples suivants illustrent les informations décrivant une personne.
"Holly Pierre est né en 1945, il est célibataire."
"Jeanne Marie est née en 1941, elle est mariée."
"Doe John est né en 1972, il est veuf."

Identifiez les informations qui caractérisent une personnne. Proposez une classe Personne, constituée uniquement d'attributs, permettant de stocker les informations décrivant une personne. Pour respecter le principe d'encapsulation, quelle visibilité doit-on attribuer aux attributs de la classe ?
Implémentez la classe Personne

Exercice 2

Ajoutez un constructeur à la classe Personne. Le constructeur permettra d'instancier un objet ainsi :

```
Personne p1;
p1 = new Personne("Holly", "Pierre", false, 1945, "celibataire");
```

Exercice 3

La 3ième ligne de ce code est-elle valide ? Justifiez.

```
Personne p1;
p1 = new Personne("Holly", "Pierre", false, 1945, "celibataire");
Console.WriteLine(p1.nom);
```

Exercice 4

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

```
public String Nom
{
  get { return nom; }
}
```

La propriété Nom permet d'obtenir la valeur de nom. Il est alors possible d'écrire :

```
Personne p1;
p1 = new Personne("Holly", "Pierre", false, 1945, "celibataire");
Console.WriteLine(p1.Nom);
```

Par contre, Nom est une propriété permetant seulement d'obtenir la valeur, pas de la définir. Le code suivant n'est donc pas valide :

```
Personne p1;
p1 = new Personne("Holly", "Pierre", false, 1945, "celibataire");
p1.Nom = "autrenom"
```

Proposez une propriété pour le prénom et l'année de naissance. Proposez une propriété Age, dont la valeur est l'âge de la personne. Le calcul se fera uniquement à partir de l'année en cours (2017).

Exercice 5

Ajoutez à la classe Personne une méthode int RetournerAgeEn(int anneeRef) qui renvoie l'âge de la personne à l'année anneeRef. Quelle visibilité doit-on attribuer à la méthode ?

Exercice 6

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

Exercice 7

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

Exercice 8

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

Exercice 9

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

Exercice 10

Ajoutez une méthode String message() qui renvoie une chaine de caractères de la forme "Jeanne Marie est née en 1941, elle est mariée."

Exercice 11

Proposez une redéfinition de ToString() pour la classe Personne. Testez.

Exercice 12

Codez la classe Point suivante en redéfinissant ToString et en ajoutant deux propriétés en lecture.

La classe Point contient 2 attributs de type double, **x** et **y**, et 2 méthodes :
Translater(dx : double, dy : double) ; retour de type void
DistanceOrigine() ; retour de type double

Exercice 13

Une personne positionnée est une personne qui a une position dans le plan. Proposez une classe adaptée puis codez-la.

Exercice 14

NB *Exercice optionnel*

1. Ecrire une classe Chat qui a pour attributs :
   — nom
   — poids
   — doitFaireUnRegime
2. Ecrire un constructeur pour la classe Chat qui a pour paramètre le nom du chat. Le poids du chat doit être initialisé de façon aléatoire (poids compris entre 3 et 9 Kg). Si le poids du chat est supérieur à 6Kg, alors le constructeur doit initialiser la variable DoitFaireUnRegime à vrai, sinon la variable doit être initialisée à faux.
3. Ecrire les propriétés permettant d'accéder aux différents attributs. Tester.
4. Ecrire une propriété qui permet de modifier le poids du chat. Tester.
5. Proposer une redéfinition de ToString() pour la classe Chat. Tester.
6. Ecrire une méthode Croquettes qui retourne le poids de la ration journalière d'un chat. La ration de croquettes est fonction du poids du chat. Il faut compter 20g de croquettes par kilos. Néanmoins si le chat est au régime sa ration doit être diminuée de 15%. Tester.

Indication : Nombres aléatoire en C#

```
Random rndNumbers = new Random();
//A initialiser 1 seule fois ...

double rndNumber = rndNumbers.NextDouble();
//pour generer un nombre reel compris entre 0 et 1
```

Pour en savoir plus : `https://msdn.microsoft.com/fr-fr/library/system.random(v=vs.11`