Leçon 2 : Notions de POO et .NET

SANE ARNAUD : Enseignant - Chercheur

Table des matières

1- Création de classes et d'objets

I

1.1- Création de classe

Une classe désigne un type plus complexe que les types standards.

Une classe regroupe en effet :

- des variables, encore appelées champs ou attributs de la classe ;
- des fonctions, encore appelées méthodes, qui portent généralement sur ces champs (c'est-à-dire qu'elles font intervenir ces champs dans leurs opérations) ;

Syntaxe :

Public Class Nom_classe

<Bloc d'instruction >

End Class

Le mot avant Class indique la portée de la classe:

- *Public* : Classe instanciable dans et hors du projet, utilisable par un autre programme
- *Private* : Classe instanciable que dans elle même!!
- *Friend* : Classe instanciable dans le projet
- *Protected* : Classe instanciable uniquement dans les classes dérivées

On parle de variable ou 'Attribut' permettant d'associer des données à une Classe.

- La variable peut être *'Privée'* et non visible à l'extérieur:

Private _mNom As String

NB : Il est conseillé de mettre un '_' au début du nom d'une variable privée puis une majuscule au début de chaque mot sauf le premier.

- Elle peut être *'Public'* , visible à l'extérieur et donc non encapsulée.

Public mNom As String

Exemple :

Public Class Pers

Public Nom As String

Public Age As Integer

End Class

1.2- Les objets

Un objet ne désigne rien d'autre qu'une variable d'une classe donnée . On parle aussi *d'instance (à la place d'objet) et d'instanciation d'une classe donnée,* pour *l'opération de création d'un objet.*

Création d'un objet :

Syntaxe :

Dim nom_objet *As New* nom_class

Exemple :

Dim P As New Pers

Accéder aux champs d'un objet

Syntaxe :

objet.Nom_champ

Remarque :

Une variable, comme tous les membres d'une classe peut être, *Private*, *Public*, mais aussi *Protected* (accessible par les seules méthodes de la classe ou des objets dérivés)

- *public* : Après ce mot clé, toutes les données ou fonctions membres sont accessibles.
- *private* : Après ce mot clé, toutes les données ou fonctions membres sont verrouillées et ne seront pas accessibles dans les classes dérivées.
- *protected* : Après ce mot clé, toutes les données ou fonctions membres sont verrouillées mais sont néanmoins accessibles dans les classes dérivées.

Exemple :

Public Class Pers

Public Nom As String

Public Age As Integer

End Class

*//********** Instanciation*

Dim P1 As New Pers

Dim P2 As New Pers

*//********** Assignation*

p1.Nom = "BLON Leticia"

p1.Age = 26

p2.Nom = "KASSELE Josiane"

p2.Age = 35

1.3- Les méthodes d'une classe

Une Classe peut contenir un comportement sous forme de procédures et fonctions contenant du code.

On peut ajouter une méthode à une Classe :

- Les méthodes d'une classe sont les procédures *Sub ou Function* déclarées dans la classe.
- Elles sont habituellement *'Public'*.

```
Public Class Personne
//**************Variables ou Attributs
Private _nom As String
//************** renvoie le _nom de l'attribut personne
Public Sub GetNom() As String
Return _nom
End Sub
//****************** Permet de donner une valeur à la variable _nom
Public Sub SetNom (ByVal N As String)
Me._nom= N
EndSub
End Class
```

1.3- Constructeur / Destructeur

- Un constructeur est une procédure exécutée lors de l'instanciation.

Dans le module de classe, elle est définie par :

Syntaxe :

```
Sub New()
End sub
```

On peut ajouter des paramètres qui serviront à instancier.

Exemple :

```
Sub New(ByVal LeNom As String)
_mNom = LeNom
End sub
```

- Un destructeur est la méthode membre appelée automatiquement lorsqu'une instance reçoit la valeur *Nothing*.

Syntaxe :

objet =Nothing

Exemple :

P = Nothing

1.4- Encapsulation

- Le fait de ne pas voir l'implémentation (le code), c'est l'encapsulation.
- Le code, les définitions de données sont privés à l'objet et non accessibles, ils sont enfermés, encapsulés dans une boite noire.
- L'encapsulation permet donc d'exposer aux applications clientes uniquement l'interface.
- Cela a une conséquence, si je modifie le code mais pas l'interface, l'application cliente n'a pas à être modifiée.

Application 1 :

II

Exercice

[*solution* n°1 p.11]

Énoncé :

Écrire classe etudiant avec les attributs nom, prenom. Ces attributs ne doivent pas être accessible en dehors de la classe. Puis instancier Et .

Solution :

Public

End

Et

2- L'héritage et le Polymorphisme

Définition : 1.1- L'héritage

L'héritage permet de former une nouvelle classe à partir de classes existantes :

- *La nouvelle classe (classe fille, sous-classe)* hérite des attributs et des méthodes *des classes à partir desquelles elle a été formée (classes mères, super-classes)*.
- *De nouveaux attributs et de nouvelles méthodes* peuvent être définis dans *la classe fille*.
- *Des méthodes des classes mères* peuvent être *redéfinies dans la classe fille*.

Il y a héritage quand on peut dire : « A *est un* B » on peut avoir comme exemple :

1. une voiture est un véhicule (Voiture hérite de Véhicule) ;
2. un bus est un véhicule (Bus hérite de Véhicule) ;
3. un moineau est un oiseau (Moineau hérite d'Oiseau) ;
4. un corbeau est un oiseau (Corbeau hérite d'Oiseau) ;
5. un chirurgien est un médecin (Chirurgien hérite de médecin) ;

Syntaxe : 1.2-Syntaxe héritage

L'héritage se traduit par le mot clé *Inherits*. On peut ensuite étendre et modifier une classe existante avec des propriétés et méthodes additionnelles.

Public Class Nom_classe_fille

Inherits Nom_classe_mere

*//***Liste des attributs*

*//******Liste des méthodes*

End Class

Exemple :

```
1 //***********Classe mère
2 Public Class Personne
3
4       Public Prenom As String
5       Public Nom As String
6       Public Sub recupdonne() As String
7        Nom =  inputbox("Entrer le nom")
8        Prenom =  inputbox("Entrer le prénoms")
```

```
9         End Sub
10    End Class
11 //************classe fille
12 Public Class Client
13        Inherits Personne
14        Public IDClient As String
15        Public TypeClient As String
16    End Class
```

1.3- Le Polymorphisme

Le nom de polymorphisme signifie *"qui peut prendre plusieurs formes"*.

Il y a 4 sortes de polymorphisme:

- *Le polymorphisme ad hoc* permet d'avoir des fonctions de même nom, avec des fonctionnalités similaires, dans des classes sans aucun rapport entre elles . Par exemple, la classe Integer, la classe Long et la classe Date peuvent avoir chacune une fonction ToString. Cela permettra de ne pas avoir à se soucier du type de l'objet que l'on a si on souhaite le transformer en String.
- *Le polymorphisme de surcharge ou en anglais overloading* est une méthode gère des paramètres de type et de nom différents. Ce polymorphisme représente la possibilité de définir plusieurs méthodes de même nom mais possédant des paramètres différents (en nombre et/ou en type). pour ouvrir une fenêtre MessageBox, la méthode Show a 12 signatures, en voici 2 :

Ici on donne 4 paramètres.

Reponse= MessageBox.show(TexteAAfficher,Titre, TypeBouton etIcone, BoutonParDéfaut)

Ici 1 seul paramètre.

Reponse= MessageBox.show(TexteAAfficher)

- *Le polymorphisme d'héritage (redéfinition, spécialisation ou en anglais overriding)*
  Quand une classe hérite d'une autre classe, elle hérite des méthodes. On peut redéfinir substituer une méthode de la classe parent par une méthode de même nom dans la classe enfant.
- L*e polymorphisme générique (en anglais template)* C'est la forme la plus naturelle du polymorphisme: elle permet d'appeler la méthode d'un objet sans devoir connaître son type. En VB 2005 cela correspond aux génériques.

Application 2 :

[solution n°2 p.11*]*

Exercice

Énoncé :

Écrire classe Client avec les attributs nom, prenom et une autre classe Grossiste qui hérite de Client avec un attribut txRemise n'est accessible que dans la classe grossiste

Solution :

Public

End

Public

txRemise

End

Solutions des exercices

> **Solution** n°1

Exercice p. 7

Énoncé :

Écrire classe etudiant avec les attributs nom, prenom. Ces attributs ne doivent pas être accessible en dehors de la classe. Puis instancier Et .

Solution :

```
Public Class etudiant
Private nom As String
Private prenom As String
End Class
Dim Et As New etudiant
```

> **Solution** n°2

Exercice p. 10

Énoncé :

Écrire classe Client avec les attributs nom, prenom et une autre classe Grossiste qui hérite de Client avec un attribut txRemise n'est accessible que dans la classe grossiste

Solution :

```
Public Class Client
Private nom As String
Private prenom As String
End Class
Public Class Grossiste
Inherits Client
Private txRemise As Double
End Class
```