VB.NET

Programmation Orienté Objet

Abdelali Saidi

saidi.a@ucd.ac.ma

Plan

Introduction

La POO

La programmation orienté objet est une manière de traiter un ensemble de propriétés et de méthodes comme une seule entité.

Introduction

Terminologie

Modules

Un module est un groupe de fonctions. En général, on implémente des fonctions publiques dans un module afin de pouvoir les appeler à partir de n'importe quel endroit du code.

Classes

Une classe est la définition d'un objet qu'on souhaite modéliser.

Introduction

Terminologie

Objets

Un objet est une instance d'une classe donné.

Espace de noms

Un espace de noms est un ensemble de classes et de structures de données. En général, les classes d'un même espace de noms appartiennent à un domaine spécifique. Pour utiliser les classes d'un espace de noms, il suffit d'importer ce dernier avec le mot clé import

Introduction

Terminologie

Exemple

```
Namespace Animeaux
    Class Chien
        Function aboie() As Integer
            Console.WriteLine("Le chien aboie !!")
            Return 0
        End Function
    End Class
End Namespace
```

```
Module Module1

    Sub Main()
        maFonction()
        Console.Read()
    End Sub

    Function maFonction() As Integer
        Dim jimmy As Animeaux.Chien
        jimmy = New Animeaux.Chien()
        jimmy.aboie()
        Return 0
    End Function

End Module
```

Plan

Constructeurs

Définition

Un constructeur est une procédure spéciale appelée lors de l'instanciation d'un objet. Elle permet d'allouer la mémoire nécessaire à l'objet et d'initialiser ses attributs.

Exemple

```
Public Sub New()
    Age = 0
End Sub
```

Plan

Les Getters et les Setters

Présentation

Les Getters et les Setters permettent d'avoir accès sur les attributs private d'une classe donné.

Les Getters et les Setters

Exemple

```
Imports System

Public Class Chien
    'Utilisons une variale privée pour l'âge :
    Private mAgeDuChien as Integer

    Public Property Age() As Integer
    'Lecture de l'âge :
    Get
        Console.Writeline ("Getting Property")
        Return mAgeDuChien
    End Get

    'MAJ de l'âge :
    Set(ByVal Value As Integer)
        Console.Writeline ("Setting Property")
        mAgeDuChien = Value
    End Set

    End Property
End Class
```

Les Getters et les Setters

Exemple (suite)

```
Class MainClass
    'Début du programme :
    Shared Sub Main()

    'Créons un objet :
    Dim Jimmy as Chien
    Jimmy = New Chien

    'Nous ne pouvons accéder directement à la variable mAgeDuChien,
    'nous devons utiliser la procédure Age().
    'Affectons-lui une valeur. La procédure Set Age sera mise à contribution :
    Jimmy.Age = 30

    'Récupérons maintenant cette valeur : c'est au tour de la procédure Get
    Dim curAge = Jimmy.Age()                                    Age de travailler :

    End Sub
End Class
```

Plan

La portés des déclarations

Les principales portées

Une classe peut contenir des attributs ou des méthodes avec les portées suivantes :

- Public : accessibles suite à la création de l'objet
- Private : accessibles qu'à l'intérieur même du module de classe.
- Protected : similaires aux Private, mais ils ont une particularité en cas d'héritage.
- Friend : accessibles qu'à l'intérieur du projet

La portés des déclarations

L'encapsulation

- L'encapsulation est la gestion des données et des procédures associées à une classe
- Les attributs sont généralement déclarés en tant que Private
- Pour accéder à ces données il faudra passer par les Property
  - Cela permet une protection des données face à des modifications accidentelles

La portés des déclarations

Exemple

```
Imports System
Namespace Animeaux
    Public Class Chien
        Public ageChein As Integer
        Public Function aboie() As Integer
            Console.WriteLine("Le chien aboie !!")
            Return 0
        End Function
        Private Function marche() As Integer
            Console.WriteLine("Le chien marche !!")
            Return 0
        End Function
    End Class
End Namespace
```

La portés des déclarations

Exemple (suite)

```
Module Module1
    Sub Main()
        maFonction()
        Console.Read()
    End Sub
    Function maFonction() As Integer
        Dim jimmy As Animeaux.Chien
        jimmy = New Animeaux.Chien()
        jimmy.aboie()
        jimmy.ageChein = 5
        jimmy.marche()
        Return 0
    End Function
End Module
```

Plan

Fonctions partagées

Le mot clé Shared

Dans une classe, les membres partagés peuvent être appelés directement (sans passer par l'instanciation d'un objet)

- Une méthode peut être appelé directement via sa classe
- Une contient une valeur commune à tous les objets d'une classe

Fonctions partagées

Modifications au niveau du dernier exemple

- La méthode aboie() sera partagée
  - Public Shared Function aboie()
- La méthode marche deviendra publique
  - Public Function marche()
- Au niveau de la fonction principale, on peut faire les appels suivants :

```
Animaux.Chien.Aboie()
Dim Jimmy As Animaux.Chien
Jimmy = New Animaux.Chien()
Jimmy.Marche()
```

NB : Qu'en est-il de la méthode writeLine() ?

Plan

La surcharge

Présentation

La surcharge est une technique qui permet d'utiliser le même nom de fonction avec des paramètres de différents types.

Exemple

On crée dans une classe (Addition) les deux méthodes suivantes :

```
Overloads Public Sub Add(A as Integer, B as Integer)
    Console.Writeline ("Adding Integers: " + Convert.ToString(a + b))
End Sub
Overloads Public Sub Add(A as String, B as String)
    Console.Writeline ("Adding Strings: " + a + b)
End Sub
```

Question : Implémentez la fonction Main() pour appeler ces deux méthodes.

Plan

L'héritage

Définition

L'héritage permet de créer des classes à partir d'une classe existante. Il suffit d'utiliser le mot clé Inherits.

Exemple

- Créez une classe Humain possédant une méthode marche()
  - La méthode marche affiche "Je marche ..."
- Créez une deuxième classe Developpeur
  - Cette classe hérite la classe Humain et possède une méthode code()
  - code() affiche "Je code ..."
- En créant un objet Humain, qu'est-ce qu'on peut appelé comme méthodes ?
- En créant un objet Developpeur, qu'est-ce qu'on peut appelé comme méthodes ?

Plan

La redéfinition

Présentation

Ce principe permet à une classe fille de redéfinir une propriété ou une méthode héritée qui doit avoir un comportement différent par rapport à celui de la classe mère.

- Overridable : est utilisé pour préciser qu'une méthode peut être redéfinie
- Overrides : est utilisé pour indiquer quelles méthodes sont redéfinies.

La redéfinition

Exercice

- À partir de la classe Humain, créez une classe Marocain qui l'hérite
- Redéfinissez la méthode marche()
- Appelez la méthode marche() des deux classes.

Plan

Le polymorphisme d'héritage

Présentation

Le polymorphisme est un mécanisme via lequel un objet peut prendre plus d'une forme.

Exemple

On suppose que nous avons créez une classe Indien à partir de la classe Humain et que nous avons redéfinit la méthode parle()

```
Class MainClass
    'Notre fonction principale :
    Shared Sub Main()
        'Tom est un humain (classe de base)
        Dim Tom as Humain

        'Affectons-lui une variable de la classe fille Indien
        Tom = New Indian
        'Cette affectation est correcte, car la classe Indien
        'est fille de la classe Humain.

        'Puis faisons appel à la méthode Parle comme ceci :
        Tom.Parle()
    End Sub
End Class
```