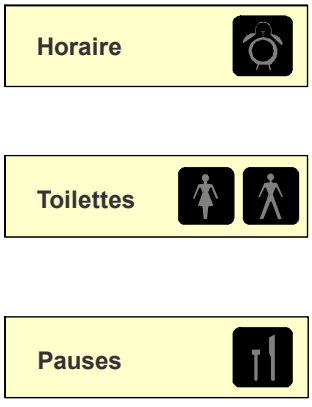
Visual Basic.NET

Agenda

Version 2.0

Agenda



Structure

- Introduction
- Base du langage
- Les formulaires Windows
- Programmation orientée objets
- Interface et collections
- Exercice « MasterMind »
- VB.NET 2 : Génériques et My
- Déploiement
- ADO.NET

Visual Basic.NET

Introduction

Version 2.0

Structure

- **Historique**
- **.NET Framework**
- **Visual Studio.NET**
- **Exercice**

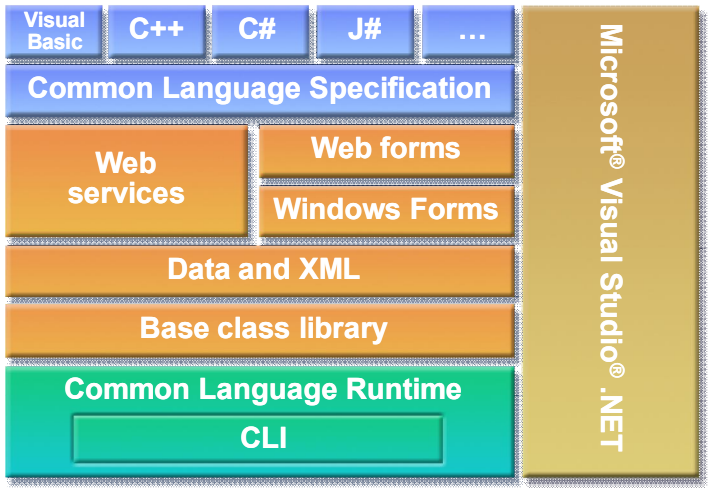
Historique

- Beginners All purpose Symbolic Instruction Code (BASIC)
- 1975 : Bill Gates & Paul Allen créent Microsoft
- 1981 : IBM diffuse BASIC avec le MS-DOS
- 1992 : Microsoft vend Visual Basic 1.0
- 1998 : Visual Basic 6.0 apparaît
- 2002 : Place à Visual Basic.NET 1.0
- 2005 : Place à Visual Basic.NET 2.0
- 2008 : Place à Visual Basic.NET 3.0
- 2010 : Place à Visual Basic.NET 4.0

Structure

- Historique
- .NET Framework
- Visual Studio.NET
- Exercice

.NET Framework



.NET Framework

- Objectifs de la plate-forme .NET
  - Concevoir les applications comme un ensemble de services via les Services Web XML
  - Simplifier le développement des logiciels
  - Création d'application robuste et durable
  - Création d'application multi-langages
    - C#, VB, C++, J#, Cobol, Eiffel, Fortran, Pascal, Perl, SmallTalk, Delphi, …
  - Compatibilité avec la technologie COM

.NET Framework

- Objectifs de la plate-forme .NET (suite)
  - Création d'une plate-forme extensible pour l'avenir
    - Microsoft Framework.NET
  - Portabilité des applications compilées (IL) sur différentes plates-formes
    - Windows 32, Windows 64
    - Windows Mobile
    - Windows XP, Windows Vista
    - Linux & Mac (Go-Mono)
      - http://www.mono-project.com

.NET Framework

- Objectifs de la plate-forme .NET (suite)
  - Common Language Runtime (CLR)
    - Coeur du .NET Framework
    - Moteur d'exécution qui charge, exécute et gère le code qui a été compilé en un pseudo-langage
      - Microsoft Intermediate Language (MSIL ou IL)
    - Peut appeler automatiquement le ramasse-miettes ou le Garbage Collector (GC)
  - Interfaces Utilisateurs
    - Application Web
    - Application Windows

.NET Framework

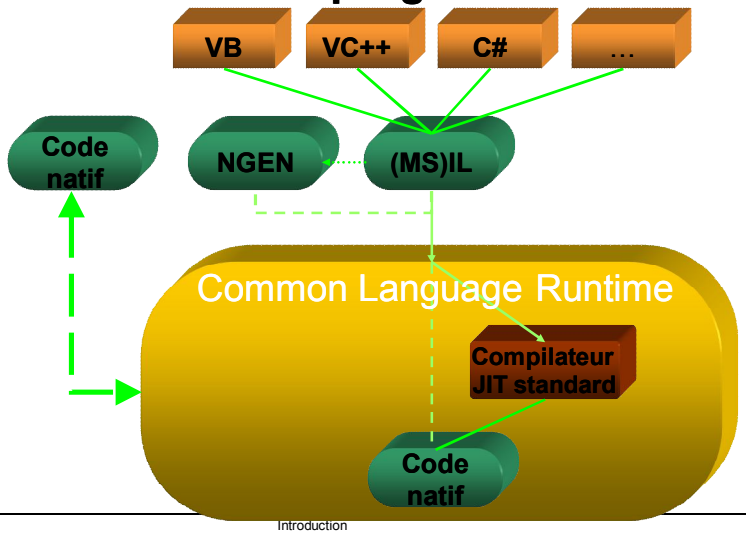
- **Le Classe de base du Framework.NET**
  - **System**
    - **Type de données primitifs, méthodes de conversions, méthodes mathématiques, etc.**
  - **System.Data**
    - **Connection avec les bases de données via Oledb ou Odbc ou avec des fournisseurs gérés (SQL Server, Oracle, DB2, MySQL, etc.)**
  - **System.IO**
    - **Lecture et écrire sur des flux ou des fichiers.**
  - **System.Net**
    - **Fournit une interface pour un grand nombre de protocoles réseaux**
  - **System.Threading**
    - **Création et gestion d'applications Multi-tâches**

.NET Framework

- **Le Classe de base du Framework.NET**
  - **System.Collections**
    - **Collections d'objet (ArrayList, Hashtable) ou spécialisées et fortement typées (StringCollection, BitArray)**
  - **System.Web.Services**
    - **Création de composantes applicatives à distance pouvant être appelées dans un environnement hétérogène**
  - **System.Web.UI**
    - **Création d'applications Web (ASP.NET)**
  - **System.Windows.Forms**
    - **Création d'applications Windows**
  - **System.Xml**
    - **Traitement des fichiers XML (eXtensible Markup Language)**

Microsoft.NET

- Exécution d'un programme



Structure

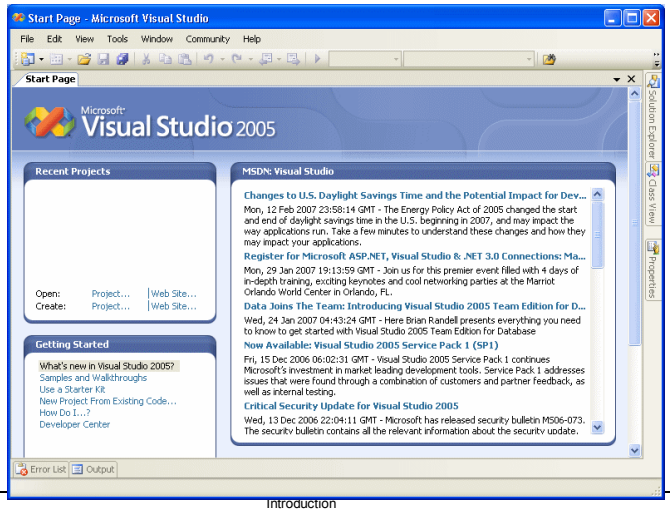
- Historique
- .NET Framework
- Visual Studio.NET
- Exercice

Visual Studio.NET

- **Programmation Windows**
  - **Outils RAD (Rapid Application Development) avec un ensemble de composant graphique**
  - **Modification du menu et ordonnancement des tabulations graphiquement**
  - **Gestion du redimensionnement par l'intermédiaire des ancres et gestion du docking**
  - **Modification des propriétés via l'interface graphique ainsi que par programmation**
  - **Déploiement simplifié**

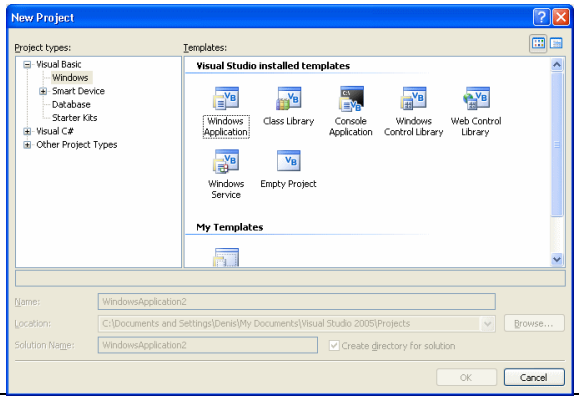
Visual Studio.NET

- **Exécuter Visual Studio .NET 2005**



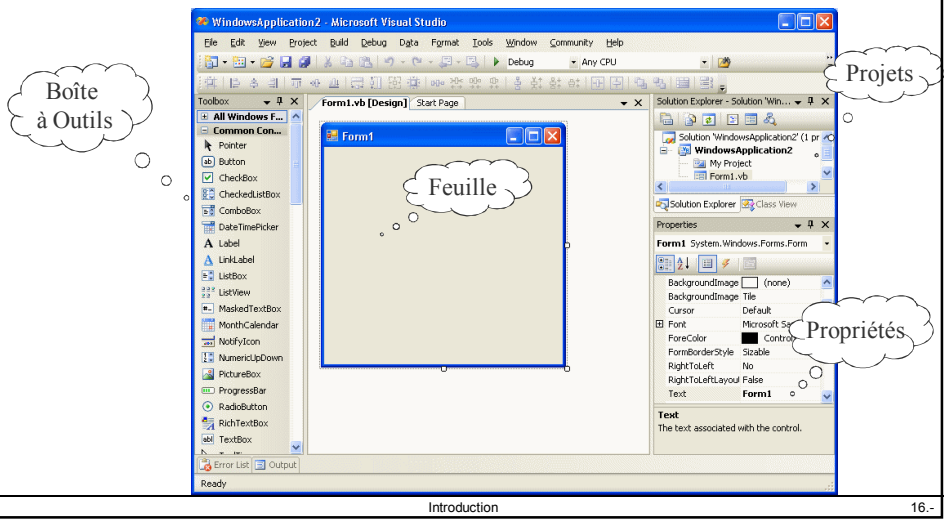
Visual Studio.NET

- **Création d'une application windows (Windows Application)**
  - **Fichier / Nouveau / Projet ...**



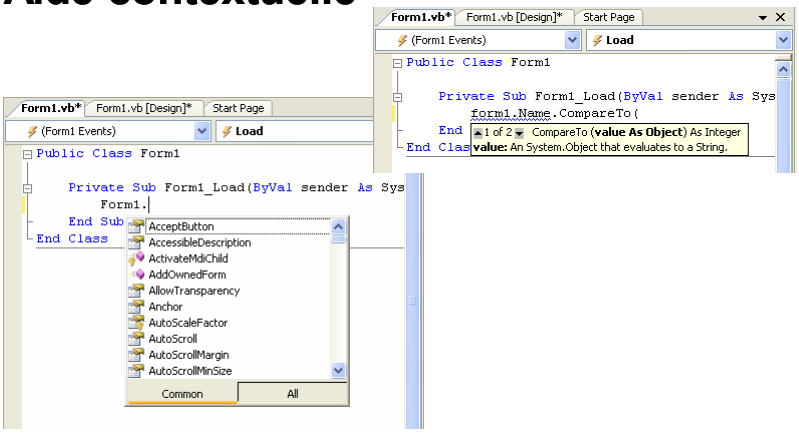
Visual Studio.NET

- **Le bureau se transforme …**



Visual Studio.NET

- **Aide complète : F1**
- **Aide contextuelle**

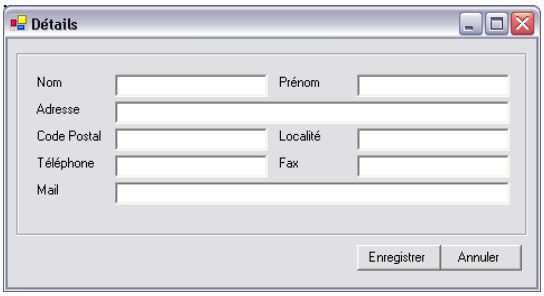


Structure

- **Historique**
- **Microsoft.NET**
- **Visual Studio.NET**
- **Exercice**

Exercice

- **Placez plusieurs composants sur votre feuille**



Visual Basic.NET

Base du Langage

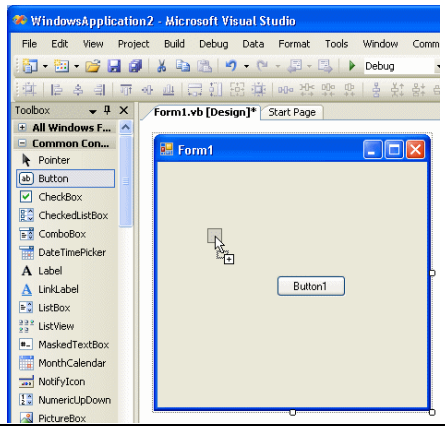
Version 2.0

Structure

- Exemple "Hello World"
- Variables
- Types primitifs
- Méthodes de conversions
- Tableaux
- Constantes
- Chaînes de caractères
- Enumérations
- Structures
- Opérateurs
- Instructions Conditionnelles
- Instructions Répétitives
- Méthodes
- Paramètres
- Surcharge des méthodes

Hello World

- Création d'un formulaire
- Ajout d'un bouton sur le formulaire



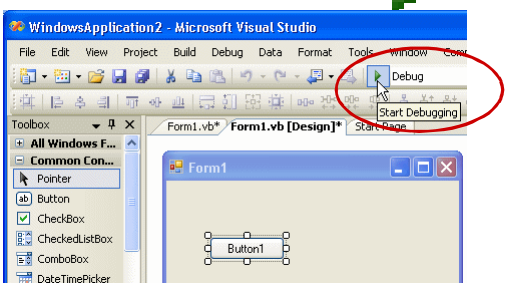
Hello World

- Ajoutez le code suivant

```
Public Class Form1

    ' Méthode qui s'éxécute lorsque l'on clique sur le bouton
    Private Sub Button1_Click(ByVal sender As System.Object, _
                              ByVal e As System.EventArgs) _
                Handles Button1.Click

        ' Permet d'afficher une information dans une fenêtre
        MessageBox.Show("Hello World")

    End Sub

End Class
```

Hello World

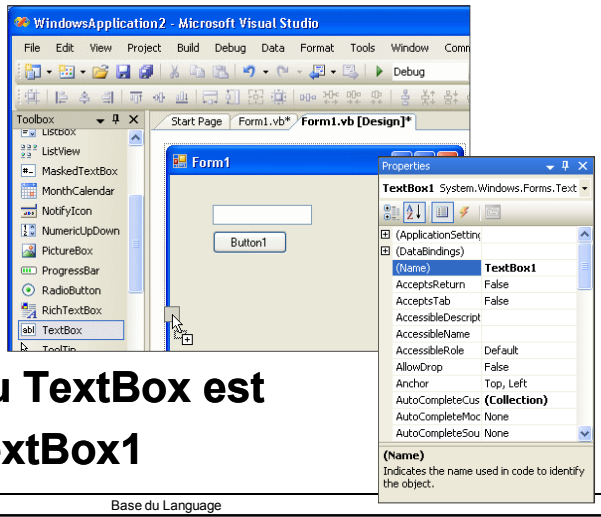
- Ensuite exécutez l'application en cliquant sur la F5 ou sur



- Cliquez sur le bouton

Hello World

- Ajoutez maintenant une zone de texte (TextBox)



- Le nom du TextBox est (Name) TextBox1

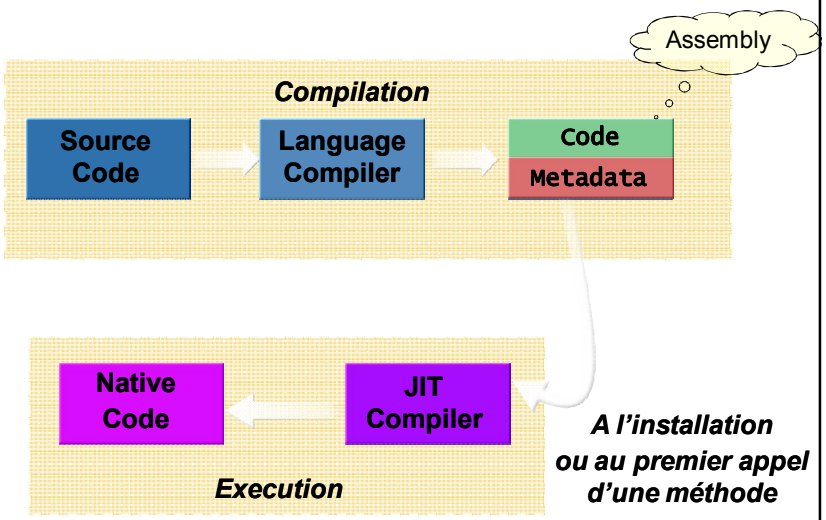
Hello World

- **Modifiez le code précédent**

```
Public Class Form1

    ' Méthode qui s'éxécute lorsque l'on clique sur le bouton
    Private Sub Button1_Click(ByVal sender As System.Object, _
                             ByVal e As System.EventArgs) _
                Handles Button1.Click

        ' Permet d'afficher une information dans une fenêtre
        MessageBox.Show(TextBox1.Text)

    End Sub

End Class
```

- **Exécutez et testez l'application**

Compilation et exécution



Variables

- **Variable locale à une procédure**

```
Dim Age As Integer = 25
```

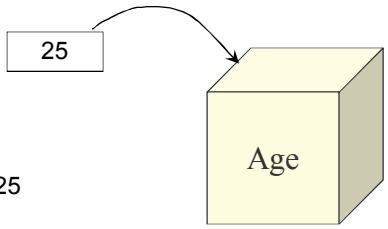
- **Variable globale à un objet**

```
Private Age As Integer = 25
```

Remarque :

Le type Integer n'est simplement qu'un abrégé de System.Int32

Dim Age As System.Int32 = 25



Type de données

Type VB.Net	Equivalent CLR	Plage
Boolean	System.Boolean	True ou False
Byte	System.Byte	0 à 255
Char	System.Char	0 à 65535
Date	System.DateTime	01/01/0001 à 31/12/9999
Decimal	System.Decimal	-2^{96} à 2^{96}
Double	System.Double	-10^{308} à 10^{308}
Single	System.Single	-10^{38} à 10^{38}
Short	System.Int16	-32 798 à 32 767
Integer	System.Int32	$-2\ 10^{9}$ à $2\ 10^{9}$
Long	System.Int64	$-9\ 10^{18}$ à $9\ 10^{18}$
String	System.String	0 à 2 milliards de caractères
Object	System.Object	NA

Remarque : chacun des types numériques correspond à une structure dans l'espace de noms System

Conversions de types

- **Deux types de conversion**
  - **Conversion Explicite**
    - **Possibilité d'avoir des pertes de données**

```
Dim a As Integer
a = CInt(123.45)
```

  - **Conversion Implicite**
    - **Aucune perte de données**

```
Dim a As Double
a = 123
```

- **si** `Option Strict = False`
  - **Certaine conversion se font automatiquement**
  - **Il peut y avoir des pertes de données**

Conversions de types

- **Obligation de convertir les données**
  - **Option Strict**
  - **Conversions étendues sécurisées**

Type VB.Net	Conversion en
Byte	Byte, Short, Integer, Long, Decimal, Single, Double
Short	Short, Integer, Long, Decimal, Single, Double
Integer	Integer, Long, Decimal, Single, Double
Long	Long, Single, Decimal, Double
Single	Single, Double
Date	Date, String

Conversions de types

Fonction de Conversion	Signification
CBool	Transforme une expression en un type Boolean
CByte	Transforme une expression en octet
CInt	Transforme une expression numérique en un entier en arrondissant
CLng	Transforme une expression numérique en un entier long en arrondissant
CSng	Transforme une expression numérique en précision simple
CDate	Transforme une expression date en date
CDbl	Transforme une expression numérique en précision double
CDec	Transforme une expression numérique du type Currency
CStr	Transforme toute expression en une chaîne
CChar	Convertit le premier caractère d'une chaîne en un type Char

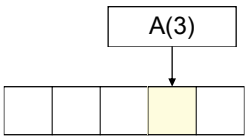
Conversions de types

- Chaque méthode de conversion en VB.NET possède un équivalent au niveau du CLR
- Les méthodes de conversion se trouvent dans System.Convert

Tableaux

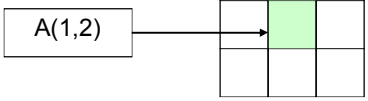
- **Collection d'éléments de données**
- **Peut contenir des objets**
- **Déclaration d'un tableau**
  - **Dimension unique**

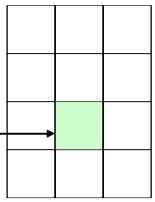
```
Dim A(4) As String
```



  - **Multidimensionnels**

```
Dim A(2,3) As String
```





Tableaux

- **System.Array**

Membre de la class Array	Signification
Clear()	**Cette méthode statique définit une plage d'éléments du tableau avec des valeurs vides**
CopyTo()	**Utilisée pour copier des éléments du tableau source dans le tableau**
Reverse()	**Inverse le contenu d'un tableau à une dimension**
Sort()	**Trie un tableau à une dimension de types intrinsèques (pour tri avancé il faut implémenter IComparer)**

Tableaux

- **Tableau dynamique**
  - ***Redim*** **redimensionne un tableau en perdant le contenu existant**
  - ***Redim Preserve*** **redimensionne un tableau en sauvegardant le contenu**

Constantes

- **Pourquoi ?**
  - **Un nom est plus explicite qu'une valeur**
  - **Pour des utilisations multiples**
- **Comment ?**

```
Private Const Pi As Double = 3.1416
Public Const Chaine As String = "Constante"
```

Chaînes de caractères

- **Manipulation de chaînes (String)**

Membre	Signification
Length()	Longueur de la chaîne
Concat()	Retourne une nouvelle chaîne provenant de deux chaînes
CompareTo()	Compare deux chaînes
Copy()	Retourne une copie d'une chaîne existante
Insert()	Insère une chaîne au sein d'une chaîne donnée
Remove() Replace()	Permet de modifier les éléments d'un chaîne (retourne une nouvelle copie)
ToUpper() ToLower()	Crée une copie d'une chaîne donnée en majuscules ou en minuscules

Chaînes de caractères

- **Remarques**
  - **"&" = String.Concat()**

```
Str = "Hello " & "World"
```

  - **En .NET, on ne peut pas modifier une chaîne de caractères : on manipule des copies de la chaîne.**
    - **Optimisation**

```
Dim str As New System.Text.StringBuilder

str.Append("Hello")
str.Append(" ")
str.Append("World")

Label1.Text = str.ToString()
```

Enumérations

- **Pourquoi ?**
  - **Un nom est plus explicite qu'une valeur**
  - **Fournir la liste des valeurs possibles**

- **Exemple :**

```
Public Enum Continent As Integer
    Amerique
    Afrique
    Europe
    Asie
    Oceanie
End Enum

Dim originePays As Continent = Continent.Europe
```


Enumérations

- **Remarques**
  - **Si aucune valeur n'est spécifiée, l'énumération des éléments commence à 0**
    - **Dans l'exemple précédent : Amerique = 0, Afrique = 1, …**
  - **Il est possible de modifier le type de l'énumération**
    - **Byte**
    - **Integer (valeur par défaut)**
    - **Long**
    - **Short**

Les structures

- **Pourquoi ?**
  - **Type personnalisé léger (par valeur)**
  - **Les types primitifs numériques (Integer, Boolean, Double, etc.) sont déclarés dans une structure**
- **Comment ?**

```
Structure Time
   Public Hour As Integer
   Public Min As Integer
   Public Sec As Integer
End Structure

Dim t As Time
t.Hour =  13
```


Opérateurs arithmétiques

Operateur	Opération
+	Addition
-	Soustraction
*	Multiplication
/	Division
\	Division entière
^	Exponentiation
Mod	Modulo (reste de la division)

Opérateurs relationnels

Operateur	Signification
=	Renvoie True si chaque expression est identique
<>	Renvoie True si chaque expression est différente
<	Renvoie True si A est inférieure que B
>	Renvoie True si A est supérieure que B
<=	Renvoie True si A est inférieure ou égale à B
>=	Renvoie True si A est supérieure ou égale à B

Opérateurs logiques

Opérateurs AND et AndAlso		
A	B	A AND B
True	True	True
True	False	False
False	True	False
False	False	False

Opérateur OR et OrElse		
A	B	A OR B
True	True	True
True	False	True
False	True	True
False	False	False

Opérateur XOR		
A	B	A XOR B
True	True	False
True	False	True
False	True	True
False	False	False

Opérateur NOT	
A	Not A
True	False
False	True

Opérateurs

- **Les opérateurs logiques And et Or**
  - **Contrairement au langage de la famille "C", l'opérateur And et Or analyse globalement l'expression. Il est possible d'utiliser AndAlso et OrElse pour effectuer une analyse partielle**
    - **Exemple :**

```
Dim i As Integer = 0

' La deuxième partie de l'expression ne sera pas vérifiée !!!
If ((i<>0) AndAlso (25/i)>5) The
   ‘ Traitement
End If
```


Instructions conditionelles

- **If … Then**
  …
  End If

```
If (A=1) Then
   msg = "A vaut 1"
Else
   msg = "A est différent de 1"
End If
```

- **Select Case …**
  …
  End Select

```
Select Case A
   Case 1
      msg = "A vaut 1"
   Case 2
      msg = "A vaut 2"
   Case Else
      msg = "Autre"
End Select
```

Instructions répétitives

- **Boucle For … Next**

```
For i = 1 To 10
   A += 1
Next
```

```
For i = 10 To 0 Step -1
   A += 1
Next
```


Instructions répétitives

- **For Each … Next**
  - **Itérer sur des collections (listes) de données**

```
Dim titres() As String = {"Visual Basic.Net",
                          "C Sharp",
                          "Framework.Net"}
Dim resultat As String
Dim element As String

For Each element In titres
   resultat &= element
Next
```

Instructions répétitives

- **Do While … Loop**

```
Do While A < 10
   A += 1
Loop
```

- **Do Until … Loop**

```
Do Until A > 10
   A += 1
Loop
```

- **Do … Loop While**

```
Do
   A += 1
Loop While A < 10
```

- **Do … Loop Until**

```
Do
   A += 1
Loop Until A > 10
```


Méthodes

- **Exécute une série de traitements**

```
Sub MaProcedure (monParam As Integer)

   Dim i As Integer
   i = monParam + 2

End Sub
```

```
Function MaFonction (monParam As Integer) As Integer

   Return monParam

End Function
```

Les paramètres

- En VB.NET, il est possible de passer des paramètres soit par référence (ByRef) soit par valeur (ByVal), dans ce dernier cas, il fournit une copie

- Il est possible de créer des paramètres optionnels (il est obligatoire de spécifier une valeur par défaut)

```
Sub MySub(Optional ByVal title As String = "VB")
  ' Traitement
End Sub
```


Surcharge des méthodes

- Il est possible d'avoir plusieurs fois la même méthode avec une signature (nombre et type de paramètres) différente

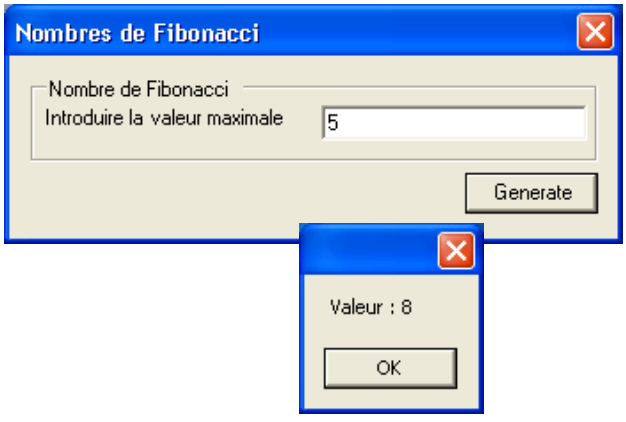
```
Public Sub MySub(ByVal i As Integer)
Public Sub MySub(ByVal s As String)
Public Sub MySub(ByVal p As String, ByVal i As Integer)
```

- Utilisation du mot-clé overloads pour spécifier explicitement la création d'une deuxième méthode avec une signature différente (mot-clé optionnel dans la plupart des cas)

Exercice 1

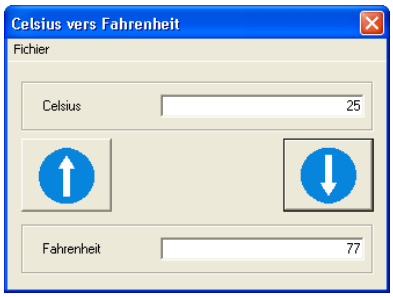
- **Calculez les nombres de Fibonacci**
  - **Fibo(0) = 1**
  - **Fibo(1) = 1**
  - **Fibo(2) = Fibo(0) + Fibo(1) = 2**
  - **…**
  - **Fibo(n) = Fibo(n-1) + Fibo(n-2)**

Exercice 1



Exercice 2

- **Développement d'un programme qui convertit des degrés** Celsius **en degrés** Fahrenheit **et inversément.**



Modifier les propriétés Name, Text et Image de manière adéquate.

Exercice 2

- **Formules de conversion**
  - **Celsius >> Fahrenheit** **F = 32 + 9/5 * C**
  - **Fahrenheit >> Celsius** **C = 5/9 * (F - 32)**

Visual Basic.NET

Les Formulaires Windows

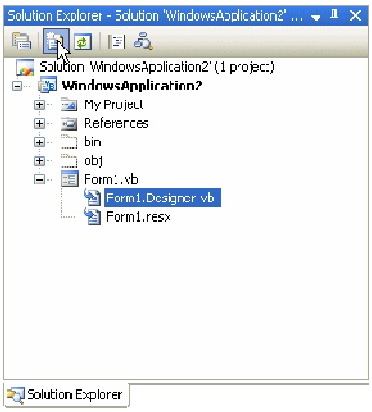
Version 2.0

Structure

- Formulaires
- Propriétés générales des contrôles
- Ordre de tabulation des contrôles
- Les contrôles courants
- Notations hongroises

Formulaires

- **Les propriétés peuvent être définies via l'interface graphique ou directement dans le code**
  - **Possibilité de voir le code généré par VB.NET dans le fichier .Designer.vb (afficher tous les fichiers)**



Formulaires

- **Propriétés**
  - **FormBorderStyle**

None	**Aucune Bordure, on ne peut redimensionner le formulaire**
FixedSingle	**Une seule bordure 3D, on ne peut redimensionner le formulaire**
Fixed3D	**Bordure 3D, on ne peut redimensionner le formulaire**
FixedDialog	**Bordure de style boîte de dialogue, on ne peut redimensionner le formulaire**
Sizeable	**Identique en apparence à FixedSingle, on peut redimensionner le formulaire**
FixedToolWindow	**Une seule bordure, on ne peut redimensionner le formulaire**
SizeableToolWindow	**Une seule bordure, on peut redimensionner le formulaire**

Formulaires

- **Propriétés**
  - `TopMost`
    - **Faculté de rester visible, même sans le focus**
  - `StartPosition`
    - **Position de départ**
  - `Opacity`
    - **Permet de rendre un formulaire complètement transparent**
  - `MaximumSize` **et** `MinimumSize`
    - **Empêcher de réduire ou d'agrandir un formulaire au-delà d'une certaine taille**
  - `WindowState`
    - **Taille lors de l'ouverture**

Formulaires

- **Propriétés**
  - `AutoScroll`
    - **Barre de défilement d'un formulaire**
  - `AcceptButton` **(Enter)**
  - `CancelButton` **(Esc)**
    - **Bouton par défaut (Exemple : Wizard)**
  - `Menu`
    - **Spécifie le menu à afficher**
  - `ControlBox`, `MaximizeBox`, `MinimizeBox`
    - **ToolBox à afficher**

Formulaires

- **Propriétés**
  - `TransparencyKey`
    - **Spécifie la couleur de fond à mettre en transparent, peut-être utilisé avec `BackgroundColor` ou `BackgroundImage` (uniquement par programmation)**

```
Dim img As Bitmap = Bitmap.FromFile("c:\Example.bmp")
'The color at Pixel(10,10) is rendered as transparent
'for the complete background.

img.MakeTransparent(Img.GetPixel(10, 10))
Me.BackgroundImage = Img
Me.TransparencyKey = Img.GetPixel(10, 10)
```


Evénements

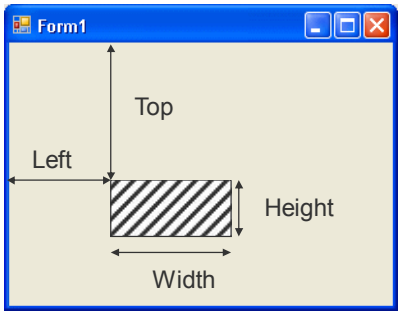
- **Evénements**
  - `Load`
    - **Chargement du formulaire**
  - `Closed`
    - **Fermeture du formulaire**
  - `Disposed`
    - **Libération des ressources**
  - `Resize`
    - **Redimensionnement du formulaire**

Structure

- Formulaires
- Propriétés générales des contrôles
- Ordre de tabulation des contrôles
- Les contrôles courants
- Notations hongroises

Propriétés générales

- (Name)
- Location
  - X, Y
- Size
  - Width
  - Height
- Dock
  - Permet de positionner un élément à un endroit de l'écran



Propriétés générales

- Anchor
  - Gestion du redimensionnement
- Tag
  - Information liée à l'objet
- Enabled
  - Contrôle graphique utilisab 

s
- Visible
- TabStop
  - True : l'utilisation de la touche TAB arrête le curseur sur le contrôle

Structure

- Formulaires
- Propriétés générales des contrôles
- Ordre de tabulation des contrôles
- Les contrôles courants
- Notations hongroises

Ordre de tabulation

- Définir l'ordre de tabulation des contrôles sur un formulaire
- `View|Tab Order`
  - L'indice de tabulation s'affiche dans le coin supérieur gauche de chaque contrôle
- Cliquer sur chaque contrôle dans l'ordre souhaité pour le parcours de tabulation
- L'ordre de tabulation peut-être modifié manuellement via le `TabIndex`

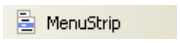
Structure

- Formulaires
- Propriétés générales des contrôles
- Ordre de tabulation des contrôles
- Les contrôles courants
- Notations hongroises

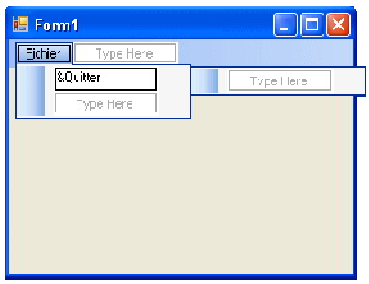
Les contrôles courants

- Menu
- Label
- TextBox
- Button
- CheckBox
- RadioButton
- ListBox
  ComboBox
- Timer
- DateTimePicker
- ToolTip
- NumericUpDown
- TabControl
- ProgressBar
- ImageList
- ListView
- TreeView

Menu (mnu)



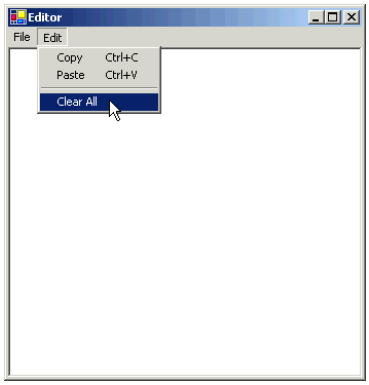
- **Menu d'un formulaire**
  - Text
  - Checked
  - ShortCut
  - ShowShortCut



- **Possibilité d'utiliser le caractère "&" pour la touche de raccourci via ALT**
- **Possibilité d'utiliser le caractère "-" comme séparateur**

Menu (mnu)

- **Exemple**
  - **&File**
    - **E&xit (CTRL+Q)**
  - **&Edit**
    - **Copy (CTRL+C)**
    - **Paste (CTRL+V)**
    - **Separator**
    - **Clear All**



Menu (mnu)

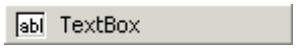
- **Insérez le code suivant :**

```
Sub exitApplication()
   Me.Close()
End Sub

Sub copyText()
  txtEditor.Copy()
End Sub

Sub pasteText()
  txtEditor.Paste()
End Sub

Sub clearText()
  txtEditor.Text = String.Empty
End Sub
```

Label (lbl)

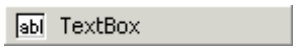
- **Libellé d'affichage et non modifiable par l'utilisateur**
- **Propriétés**
  - `Text` **... Contenu du libellé**
  - `BorderStyle` **... None | FixedSingle | Fixed3D**
  - `ForeColor` **... Couleur des caractères**
  - `TextAlign` **... TopLeft | TopRight | MiddleCenter | …**
- **Particularité**
  - **Utilisation du & pour définir une touche d'accès rapide en combinaison avec ALT**
  - **Exemple : lblExample.Text = "No&m :"**

TextBox (txt)



- **Affiche une boîte d'encodage de texte pour l'utilisateur**
- **Propriétés**
  - `Text` **... Contenu de la zone**
  - `BackColor` **... Couleur de fond**
  - `ReadOnly` **... Zone non modifiable**
  - `MaxLength` **... Nombre de caractères maximum**
  - `PasswordChar` **... Caractère d'affichage « invisible »**

TextBox (txt)



- `MultiLine` ... Une seule ligne | Plusieurs lignes
- `ScrollBars` ... Barre horizontale et/ou verticale

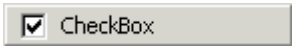
- **Evénements**
  - `TextChanged` ... Le contenu (.Text) a changé
  - `GotFocus` ... Le curseur est entré
  - `LostFocus` ... Le curseur est sorti
  - `KeyPress` ... Une touche a été enfoncée
  - `KeyDown` … Une touche spéciale a été enfoncée (ex: DEL)

Button (btn)

- **Bouton que l'utilisateur peut enfoncer**
- **Propriétés**
  - `Text` ... Contenu texte du bouton (&)
  - `BackColor` ... Couleur de fond
  - `Image` ... Image du bouton
  - `TextAlign` … Position du texte sur le bouton

- **Evénement**
  - `Click` ... L'utilisateur a enfoncé le bouton

CheckBox (chk)

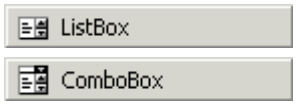


- **Sélectionne un élément**
- **Propriétés**
  - `Text` **... Libellé de la zone (&)**
  - `Checked` **… True | False**
  - `ThreeState` **… True | False**
  - `CheckState` **... Checked | Unchecked| Intermediate**
  - `TextAlign` **… Position du texte et de la case**
- **Evénement**
  - `Click`

RadioButton(rad)

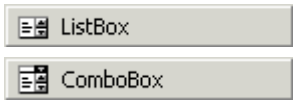
- **Sélectionne un élément dans une liste de** `RadioButton`
- **Propriétés**
  - `Text` **... Libellé de la zone (&)**
  - `Checked` **... True | False**
  - `CheckAlign` **… Position de la case**
  - `TextAlign` **… Position du texte et de la case**
- **Evénement**
  - `Click`

ListBox (lst)
ComboBox (cbo)



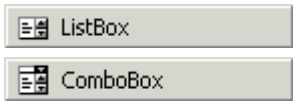
- Liste d'éléments (Texte)
- Propriétés
  - `Count` … Retourne le nombre d'élément
  - `BackColor` … Couleur de fond
  - `Items` … Éléments
  - `SelectedItem` … Élément sélectionné

ListBox (lst)
ComboBox (cbo)



- Méthodes
  - `Items.Clear()` ... Efface toute la liste
  - `Items.Add()` ... Ajoute un élément
  - `Items.Remove()` ... Retire un élément

ListBox (lst)
ComboBox (cbo)




```
Sub fillComboBox
   cbo.Items.Add("Pierre Dupont")
   cbo.Items.Add("Benoit Dubois")
End Sub
```

```
Sub clearComboBox
   cbo.Items.Clear()
End Sub
```


Timer (tmr)

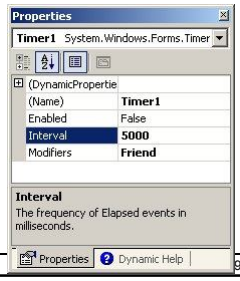
- **Contrôle qui ne fait pas partie de l'interface utilisateur**
- **Désactivé par défaut**
  - **Méthode `Start()` permet l'activation du `Timer`**
- **Propriété `Interval`**
  - **Intervalle en millisecondes du déclenchement du `Timer`**

Timer (tmr)

- **Exemple**

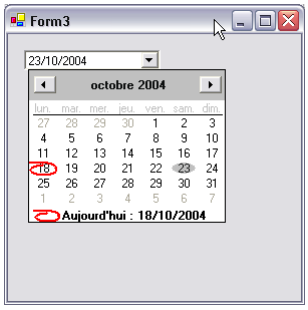
```
Sub tickHandler()
   MessageBox.Show("Déclenchement du timer")
End Sub
```

- **Toutes les 5 secondes, une boîte de dialogue s'ouvrira et contiendra « Déclenchement du timer »**



DateTimePicker (dtp)

- **Propriétés**
  - **Checked**
  - **ShowCheckBox**
  - **CustomFormat**
  - **Format**
  - **ShowUpDown**
  - **Value**

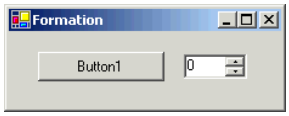


ToolTip (tip)

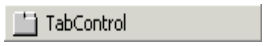
- **Contrôle qui ne fait pas partie de l'interface utilisateur**
- **L'ajout de ce contrôle crée une nouvelle propriété à chaque contrôle du formulaire**
  - **`Tooltip` on ToolTip1**
- **Une valeur à cette propriété fera apparaître une infobulle lorsque la souris sera sur le contrôle**

NumericUpDown (nud)

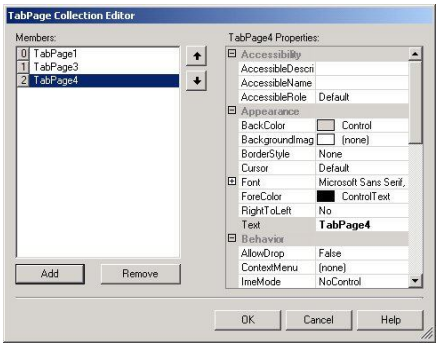
- **Zone de texte d'une seule ligne**
  - **un nombre**
  - **un bouton haut/bas qui permet d'incrémenter ou de décrémenter le nombre**
- **Propriété `Value`**



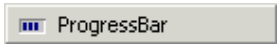
TabControl (tab)



- **Propriété (TabPages)**
  - Add
  - Remove
  - Text
    - « Onglet 1 »
  - ToolTipText
- Appearance



ProgressBar



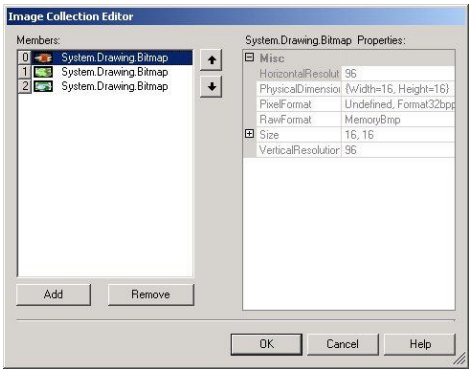
- **Propriétés**
  - Minimum = 0
  - Maximum = 100

  - Value = 50

```
ProgressBar.Minimum = 1
ProgressBar.Maximum = 250

For t = 1 To 250
   ProgressBar.Value = t
Next
```

ImageList (iml)

- **Enregistre une bibliothèque d'images généralement utilisé par d'autres contrôles tels que `ListView`, `TreeView` ou `ToolBar`**



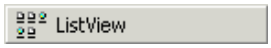
- **Méthodes**
  - `Add()`
  - `Remove()`
- **Propriétés**
  - `Images`

ImageList (iml)

- **Exemple :**

```
Sub changeImage()
    PictureBox1.Image = ImageList1.Images(1)
End Sub
```

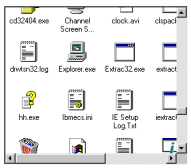
ListView (lvw)

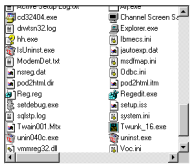


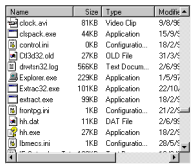
- **Affiche des éléments dans une liste du style Windows Explorer**
- **Propriétés**
  - `View`
    - `LargeIcon`
    - `SmallIcon`
    - `List`
    - `Details`
  - `FullRowSelect`
    - `True-False`

ListView (lvw)

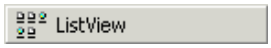
- **LargeIcon**
- **SmallIcon**
- **List**
- **Details**



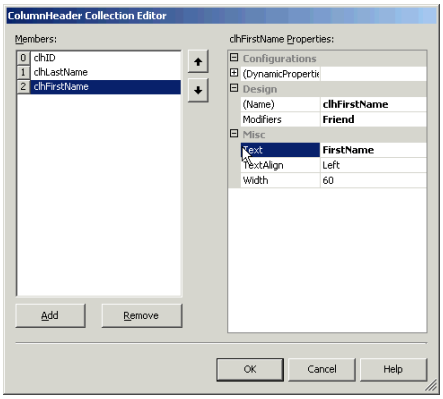




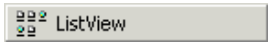
ListView (lvw)



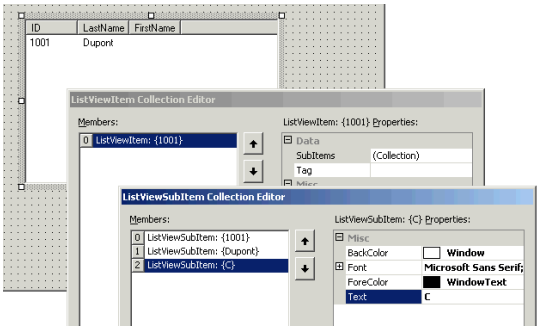
- **Propriété : `Columns`**
  - `Add()`
  - `Remove()`
  - `Text`
  - `TextAlign`



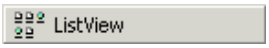
ListView (lvw)

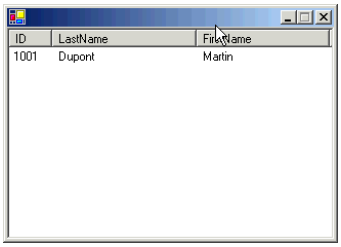


- **Propriété : `Items`**
  - `SubItems`
  - `Text`



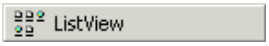
ListView (lvw)





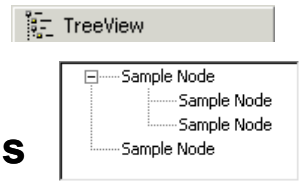

```
Private Sub fillListView()
   lvwCustomers.Items.Clear()
   lvwCustomers.Items.Add("1001")
   lvwCustomers.Items(0).SubItems.Add("Dupont")
   lvwCustomers.Items(0).SubItems.Add("Martin")
End Sub
```


ListView (lvw)



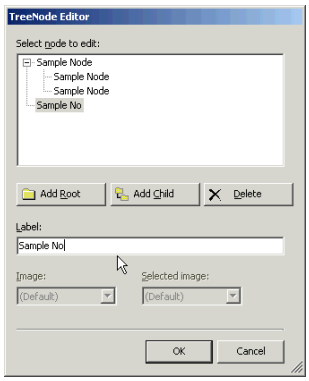
- **Evénements**
  - `ColumnClick`
  - `Click`
  - `...`

TreeView (tvw)



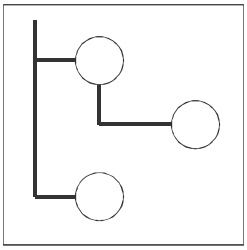
- **Affiche des éléments dans une liste arborescente**

- **Propriétés**
  - `Nodes`
  - `ShowLines`
  - `ShowPlusMinus`
  - `ShowRootLines`

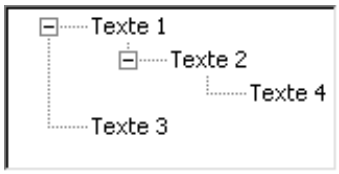


TreeView (tvw)

- **Méthodes**
  - `Clear`
    - `tvw.Nodes.Clear`
  - `Add`

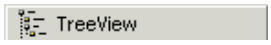


TreeView (tvw)




```
Private Sub fillTreeView
   tvw.Nodes.Clear()
   tvw.Nodes.Add("Texte1")
   tvw.Nodes(0).Nodes.Add("Texte2")
   tvw.Nodes(0).Nodes(0).Nodes.Add("Texte4")
   tvw.Nodes.Add("Texte3")
End Sub
```


TreeView (tvw)



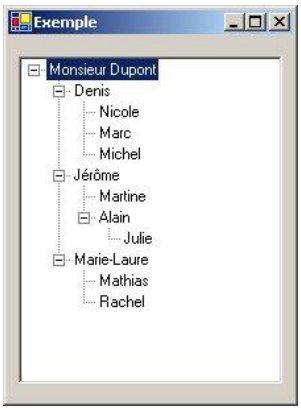
- **Evénements**
  - AfterCollapse
  - BeforeCollapse
  - AfterExpand
  - BeforeExpand
  - Click
  - DoubleClick

Exercice

- Monsieur Dupont a 2 fils et 1 fille (Denis, Jérôme et Marie-Laure)
- Denis a 1 fille (Nicole) et 2 fils (Marc et Michel)
- Jérôme a une fille (Martine) qui n'a pas d'enfant et 1 fils (Alain) qui a 1 fille (Julie)
- Marie-Laure a un fils (Mathias) et 1 fille (Rachel)
- Afficher l'arbre généalogique

Exercice

- Ce que vous devez obtenir :



Préfixage des contrôles

Contrôle	Préfixe
Button	btn
CheckBox	chk
CheckedListBox	clb
ColorDialog	col
ComboBox	cbo
ContextMenu	cmn
DataGrid	dgrid
GroupBox	grp
ImageList	iml
Label	lbl
LinkLabel	llb
ListBox	lst

Préfixage des contrôles

Contrôle	Préfixe
ListView	lvw
Menu	mnu
MonthCalendar	cal
NumericUpDown	nud
OpenFileDialog	ofd
Panel	pnl
PictureBox	img
RadioButton	rad
SaveFileDialog	sfd
Splitter	spl
StatusBar	Stb
TextBox	txt

Visual Basic.NET

Programmation
Orientée
Objets

Version 2.0

Structure

- Classe et Objet
- Constructeur
- Encapsulation
- Héritage
- Polymorphisme
- Evénement
- Exception
- Garbage Collector
- Exercices

Introduction à l'orienté objet

- **Modélisation informatique d'un ensemble d'éléments, d'une partie du monde réel en un ensemble d'entités informatiques.**
- **Ces entités informatiques sont appelées objet.**
- **Il s'agit de données informatiques regroupant les principales caractéristiques (taille, la couleur, ...) ainsi que leurs fonctionnements.**

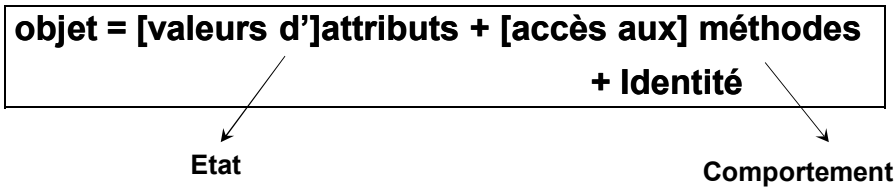
Introduction à l'orienté objet

- **Problématique**
  - **Le monde du logiciel ne suit pas la même évolution que le monde du matériel informatique**
  - **Le logiciel a beaucoup de peine à suivre une évolution de complexité simplement linéairement croissante**
    - **Faible résistance aux changements**
- **Question ?**
  - **Comment rendre un logiciel plus résistant aux changements sans devoir repartir de zéro**
- **Solutions**
  - **Une des solutions pourrait être le principe d'encapsulation**
    - **Permet de cacher les éléments modifiables aux sein d'un programme**

Objet

- **Un objet**
  - **Entité réelle (ex : voiture) ou abstraite (ex : temps),**
  - **Possède une identité unique,**
  - **Contient un état (attributs) et un comportement propre (méthodes),**
  - **Un objet = "instance d'une classe"**

Objet



Objet

- **Les attributs :**
  - **Il s'agit des données caractérisant l'objet. Ce sont des variables stockant des informations d'état de l'objet.**

Objet

- **Les méthodes :**
  - **Les méthodes d'un objet caractérisent son comportement, c'est-à-dire l'ensemble des actions (appelées opérations) que l'objet est à même de réaliser. Ces opérations permettent de faire réagir l'objet aux sollicitations extérieures (ou d'agir sur les autres objets). De plus, les opérations sont étroitement liées aux attributs, car leurs actions peuvent dépendre des valeurs des attributs, ou bien les modifier**

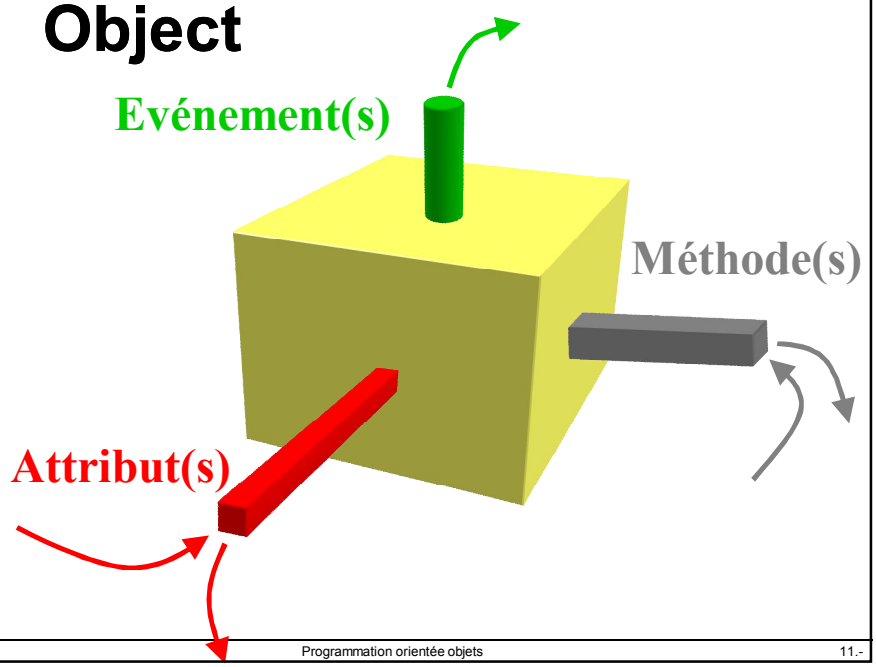
Objet

- **L'identité**
  - **L'objet possède une identité, qui permet de le distinguer des autres objets, indépendamment de son état.**
  - **Une adresse mémoire (un pointeur)**

Objet

- **Cycle de vie d'un objet**
  - **Création d'un objet (représente, dans un langage de programmation la réservation d'un espace mémoire via le mot clef New ou new)**
  - **Appel automatique d'une méthode particulière appelée par le constructeur**
  - **Utilisation de l'objet**
  - **Destruction de l'objet (suppression de l'espace mémoire réservé)**
  - **Appel d'une méthode particulière pour libérer les ressources (peut se faire automatiquement – "Garbage Collector")**

Object

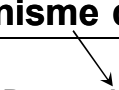


Classe

- Une classe est un modèle, un «moule», qui permet de fabriquer des objets de même structure, de même comportement
- Caractérisé par
  - des attributs (états) et
  - des méthodes (comportements)
- L'objet est donc issu d'une classe
- Une classe peut être considérée comme un type utilisateur (type personnalisé)

Classe

classe = attributs + méthodes
+ mécanisme d'instanciation



Permet de créer des instances (objets)

Classe

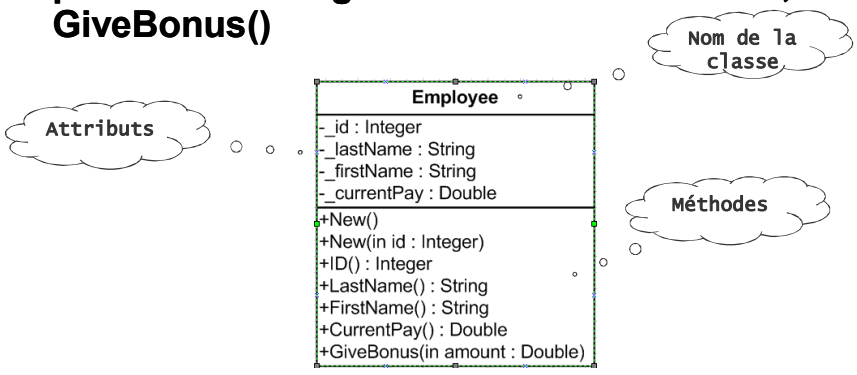
- **Avantage**
  - **Modèle plus simple**
    - **Il y a beaucoup moins de classes que d'objets**
    - **les objets d'une même classe sont similaires**
    - **L'ensemble des classes ne change pas pendant l'exécution**
  - **Modèle plus général**
    - **Le modèle est indépendant de l'état des objets**
    - **Le modèle est indépendant du nombre d'objets**
  - **Modèle réutilisable**
    - **Une classe est un composant logiciel réutilisable**

Classe et Objet

- Il est important de faire la différence entre
  - concept ou classe (abstrait)
    - Voiture
    - Vacances
    - Editeur de texte
  - ... et l'objet (concret)
    - Audi
    - Egypte
    - UltraEdit
- "Audi" est un représentant (concret) du concept (abstrait) de "Voiture"

Classe et Objet

- Création d'une classe pour les employés ; chaque employé possède un nom, un salaire et un numéro d'employé. Supposons que chaque employé possède une méthode permettant d'augmenter le bonus salarial, GiveBonus()



Création d'un objet

- On crée un objet via le mot clé New
- New gère un nouvel espace mémoire pour l'objet
- Appel automatique du constructeur

```
Dim employee as Employee
employee = New Employee()
```

```
Dim employee As Employee = New Employee()
```

```
Dim employee As New Employee()
```


Structure

- Classe et Objet
- Constructeur
- Encapsulation
- Héritage
- Polymorphisme
- Evénement
- Exception
- Garbage Collector
- Exercices

Constructeur

- Initialisation de toutes les données d'état avec une valeur correcte lors de la création d'un objet
- Possibilité de créer un nombre illimité de constructeurs (via la surcharge des méthodes)
  - Exemple :

```
Sub New()
  ' Constructor
End Sub
```


Constructeur

- Surcharge des membres, c'est-à-dire possibilité d'avoir plusieurs fois la même méthode avec une signature différente
- Possibilité d'appeler le constructeur de la classe courante via le mot clé "Me"

Classe Employee

```
Public Class Employee

	Private _id As Integer
	Private _lastName As String
	Private _firstName As String
	Private _currentPay As Double

	Public Sub New(ByVal id As Integer)
		_id = id
	End Sub

	Public Sub GiveBonus(ByVal amount As Double)
		_currentPay += amount
	End Sub

End Class
```


Structure

- Classe et Objet
- Constructeur
- Encapsulation
- Héritage
- Polymorphisme
- Evénement
- Exception
- Garbage Collector
- Exercices

Encapsulation

- Toutes les données et les méthodes sont rassemblées au niveau de l'objet et seules les informations qui sont utiles sont accessibles de l'extérieur (Interface)
- Les données internes d'un objet ne doivent pas être accessibles directement à partir d'une instance de cet objet
- Si l'utilisateur veut modifier les données, il doit le faire de manière indirecte
  - Par l'intermédaire des propriétés
  - Les données d'état doivent être "Private"

Encapsulation

- L'encapsulation permet de définir des niveaux de visibilité des éléments de la classe. Ces niveaux de visibilité définissent les droits d'accès aux données selon que l'on y accède par une méthode de la classe elle-même, d'une classe héritière, ou bien d'une classe quelconque.
  - Public
  - Private
  - Friend
  - Protected (cfr.Héritage)
  - Protected Friend (cfr.Héritage)

Encapsulation

Méthode d'accès	Signification
Public	Marque une méthode comme étant accessible à partir d'une instance d'objet ou de n'importe quelle sous-classe. Niveau de visibilité par défaut
Private	Marque une méthode comme étant accessible seulement par la classe qui a défini la méthode
Friend	Définit une méthode comme étant utilisable par n'importe quel type du même assembly

Encapsulation

- Accès aux attributs par l'intermédiaire des propriétés Get et Set
- Méthodologie :
  - 1. Création d'un attribut privé
  - 2. Création de propriété en lecture et/ou écriture
    - `Public Property Name As String (R/W)`
    - `Public ReadOnly Property Name As String (R)`
    - `Public WriteOnly Property Name As String (W)`
  - `Bloc Get = Lecture`
  - `Bloc Set = Ecriture`

Encapsulation

- **Exemple :**

```
Private _fields As String

Public Property Fields() As String
       Get
              Return _fields
       End Get
       Set (ByVal Value As String)
              _fields = Value
       End Set
End Property
```


Encapsulation

```
Private _fields As String
Public ReadOnly Property Fields() As String
   Get
      Return _fields
   End Get
End Property
```

```
Private _fields As String
Public WriteOnly Property Fields() As String
   Set(ByVal Value As String)
      _fields = Value
   End Set
End Property
```

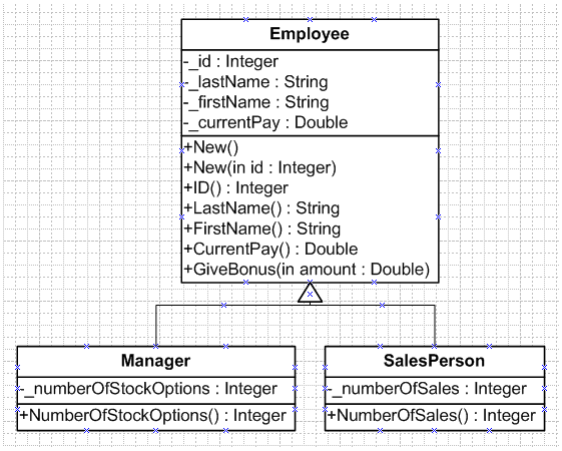
Structure

- Classe et Objet
- Constructeur
- Encapsulation
- Héritage
- Polymorphisme
- Evénement
- Exception
- Garbage Collector
- Exercices

Héritage

- Création sur base d'une classe existante, en héritant de l'interface et des fonctions de la classe initiale
- Permet d'étendre le comportement d'une classe de base (ou classe parent) en héritant des principales fonctionnalités dans une sous-classe (ou classe enfant)
- Créer une classe de base
- Créer une sous-classe
- Utilisation du mot-clé "Inherits"

Héritage



Un vendeur "est-un" employé tout comme un manager

Les classes de base sont utilisées pour définir les caractéristiques générales qui sont communes à tous les descendants. Les sous-classes étendent ces fonctionnalités généralement en ajoutant des comportements spécifiques de la classe

Héritage

- **Exemple**

```
Public Class Manager
   Inherits Employee

   Private _numberOfStockOptions As Integer

   Public Property NumberOfStockOptions() As Integer
      Get
         Return _numberOfStockOptions
      End Get
      Set(ByVal Value As Integer)
         _numberOfStockOptions = Value
      End Set
   End Property
End Class
```

Héritage

- **Exemple**

```
Public Class SalesPerson
   Inherits Employee

   Private _numberOfSales As Integer

   Public Property NumberOfSales() As Integer
      Get
         Return _numberOfSales
      End Get
      Set(ByVal Value As Integer)
         _numberOfSales = Value
      End Set
   End Property
End Class
```


Héritage

- **Encapsulation**

Méthode d'accès	Signification
Protected	Marque une méthode comme étant utilisable par la classe qui l'a définit, ainsi que par toute classe enfant, mais elle est Private pour les autres utilisateurs
Protected Friend	Marque une méthode comme étant utilisable par la classe qui l'a définit, ainsi que par toute classe enfant ainsi que par n'importe quel type du même assembly , mais elle est Private pour les autres utilisateurs

Contrôle de la création de la classe de base

- Lorsque l'on crée un objet "Manager", VB fait appel au constructeur de la classe parent et ensuite celui de la classe enfant
- Pour une optimisation des appels, il faut dans la classe enfant faire appel, explicitement, au constructeur de la classe parent grâce au mot-clé "MyBase"

Me, MyBase

- Interaction avec des représentations importantes d'objets et de classes
- Me
  - Met à disposition une référence de l'instance de l'objet courant
  - Pas obligatoire

Me, MyBase

- **MyBase**
  - **Référence la classe parent**
  - **Appel des méthodes avec MyBase comme si nous avions une référence à un objet du type de données de notre classe parent**
  - **Pas de moyen de naviguer directement dans la chaîne d'héritage (cas de redéfinition) au-delà de notre parent immédiat**

Interruption de l'héritage

- **Il peut arriver un moment où l'on souhaite ne plus pouvoir hériter d'une classe (sealed class)**
- **Utilisation du mot-clé "NotInheritable"**
- **Optimisation**
- **Exemple :**
  - **Dans le Framework.NET la class String a été déclarée comme NotInheritable,**
  - **Création d'une classe pour les vendeurs à temps partiel :**

```
Public NotInheritable Class PartialTimeSalesPerson
   Inherits SalesPerson

End Class
```

Classes Abstraites

- **Problématique :**
  - **Nous remarquons que nous avons dans Employee des méthodes et des propriétés. Ces données ont un sens lors de la création d'un objet vendeur ou manager. Or nous remarquons que l'on peut créer un objet Employé …**
    - **Que signifie ?**
      - `Private _employee As New Employee`

```
Public MustInherit Class Employee
      …
End Class
```


Propriétés et Méthodes Partagées

- **Créer des propriétés et des méthodes qui appartiennent aux classes plutôt qu'à n'importe quel objet spécifique**
- **Ces méthodes appartiennent à tous les objets d'une classe donnée et sont partagées par toutes les instances de la classe**
- **Mot-clé "Shared"**

Propriétés et Méthodes Partagées

- **Exemple : Création d'une propriété qui définit le nom de l'entreprise pour tous les employés**

```
Private Shared _companyName As String

Public Shared Property CompanyName() As String
   Get
      Return _companyName
    End Get
    Set(ByVal Value As String)
       _companyName = Value
    End Set
End Property
```


Structure

- **Classe et Objet**
- **Constructeur**
- **Encapsulation**
- **Héritage**
- **Polymorphisme**
- **Evénement**
- **Exception**
- **Garbage Collector**
- **Exercices**

Polymorphisme

- Possibilité pour une méthode de réagir différemment
- Exemple
  - Méthode GiveBonus() au niveau de la classe parent
  - La méthode doit réagir différemment en fonction des stocks options pour les employés et des ventes pour les vendeurs
- Le polymorphisme permet à une sous-classe de redéfinir une méthode

Polymorphisme

- Utilisation des mots-clés "Overridable" et "Overrides"
- Exemple :

```
Public Overridable Sub GiveBonus(ByVal amount As Double)
  _currentPay += amount
End Sub
```

Polymorphisme

```
Private Class SalesPerson : Inherits Employee
   Public Overrides Sub GiveBonus(ByVal amount As Double)
      ...
      MyBase.GiveBonus(amount * (_numberOfSales * 0.3))
    End Sub
End Class
```

```
Private Class Manager : Inherits Employee
   Public Overrides Sub GiveBonus(ByVal amount As Double)
       ...
       MyBase.GiveBonus(amount * (_numberOfStockOptions * 1.2))
   End Sub
End Class
```


Polymorphisme

- **Shadows**
  - **Permet de redéfinir la méthode de la classe parent en cachant la définition de départ**
    - **On peut redéfinir la méthode en ayant une signature différente !**

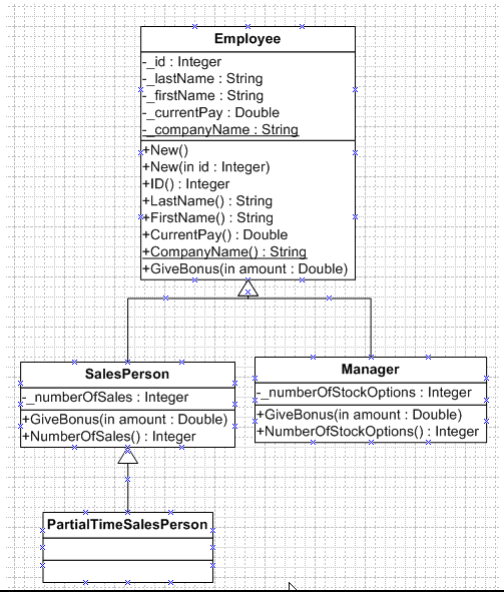
```
Private Class Parent
   Public Sub M1()
      ...
   End Sub
End Class
```

```
Private Class Child : Inherits Parent
   Public Shadows Sub M1(ByVal p As Byte)
       ‘ La signature parent n’existe plus
   End Sub
End Class
```

Polymorphisme

- **Overloads**
  - **Permet de surcharger la méthode dans la classe courante**
  - **Si le mot clé n'est pas spécifié, il utilise Shadows par défaut**

Diagramme final



Structure

- Classe et Objet
- Constructeur
- Encapsulation
- Héritage
- Polymorphisme
- Evénement
- Exception
- Garbage Collector
- Exercices

Gestion des événements

- Liaison entre une méthode et un événement par programmation grâce au mot-clé "Handles"
- Exemple :

```
 Private Sub btn_Click(ByVal sender As System.Object, _
         ByVal e As System.EventArgs) Handles btn.Click
   ...
End Sub
```

Gestion des événements

- Dans le FCL (Framework Class Library), tous les évènements possèdent deux paramètres
  - Le premier paramètre, `sender`, est l'objet qui a enclenché l'évènement
  - Le second, `e`, est une liste de paramètres
- Il est fortement conseillé de respecter ce schéma lors de la création d'évènement

Gestion des événements

- Création d'un événement
  - `Public Event CurrentPayChanged(sender As Object, e As EventArgs)`
- Enclenchement d'un événement
  - `RaiseEvent CurrentPayChanged(Me,e)`
- Exemple :

```
Public Class Employee
    Public Event CurrentPayChanged (sender As Object, _
                                            e As EventArgs)
    Protected Overridable Sub OnCurrentpayChanged( _
                                            e As EventArgs)
         RaiseEvent CurrentPayChanged(Me, e)
    End Sub
End Class
```

Gestion des événements

- Création d'un objet avec la prise en charge des événements

```
Private WithEvents _manager As New Manager()
```

- Il est possible de gérer les évènements de manière indépendante en utilisant AddHandler … AddressOf et RemoveHandler … AddressOf

Structure

- Classe et Objet
- Constructeur
- Encapsulation
- Héritage
- Polymorphisme
- Evénement
- Exception
- Garbage Collector
- Exercices

Gestions des exceptions

Propriété System.Exception	Signification
HelpLink	Cette propriété renvoie une URL vers un fichier qui décrit l'erreur en détail.
Message	Cette propriété en lecture seule renvoie la description textuelle d'une erreur donnée
Source	Cette propriété renvoie le nom de l'objet (ou éventuellement de l'application) qui a envoyé l'erreur
StackTrace	Cette propriété en lecture seule contient une chaîne qui identifie la séquence d'appels qui a déclenché l'erreur
InnerException	Permet de préserver les détails de l'erreur entre une série d'exception

Gestions des exceptions

- **Interception d'exceptions**

```
Try
   ...
Catch ex As ArgumentException
   ...
   MessageBox.Show(ex.Source)
Catch ex As ApplicationException
   ...
   MessageBox.Show(ex.Source)
Finally
   ...
End try
```

Gestions des exceptions

- Le mot clé Throw Permet d'enclencher une exception
- Si rien n'est spécifié après le Throw, il réenclenche l'exception interceptée

```
Try
   ...
Catch ex As ArgumentException
   Throw New ApplicationException("Erreur !!!")
Catch ex As Exception
   Throw
End try
```


Structure

- Classe et Objet
- Constructeur
- Encapsulation
- Héritage
- Polymorphisme
- Evénement
- Exception
- Garbage Collector
- Exercices

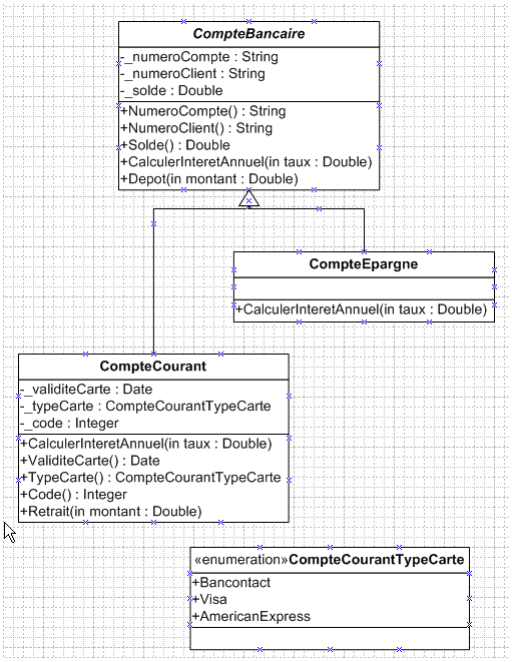
Durée de vie des objets

- Un objet est automatiquement détruit (par le ramasse-miettes) lorsque aucune référence relative à un objet n'est présentée au sein de la portée en cours (ou lorsqu'une référence a été explicitement initialisée à Nothing)

Structure

- Classe et Objet
- Constructeur
- Encapsulation
- Héritage
- Polymorphisme
- Evénement
- Garbage Collector
- Exercices

Exercice



Exercice

- **Le but de l'exercice est de faire la structure (pas de code dans les méthodes) des classes suivante :**
- **Création d'une class "CompteBancaire"**
  - **Class Abstraite**
  - **Attributs :**
    - **_numeroCompte**
    - **_numeroClient**
    - **_solde**
  - **Méthodes :**
    - **CalculerInteretAnnuel(ByVal taux As Double)**
    - **Depot(ByVal montant As Double)**

Exercice

- **Créer deux classes enfant**
  - **CompteCourant**
    - **Attributs supplémentaires**
      - **_validiteCarte**
      - **_typeCarte (définir un enum "CompteCourantTypeCarte": Bancontact, Visa et AmericanExpress)**
      - **_code**
    - **Methodes supplémentaires**
      - **Overrides - CalculerInteretAnnuel(taux As Double)**
      - **Retrait(montant As Double)**
  - **CompteEpargne**
    - **Méthode**
      - **Overrides - CalculerInteretAnnuel(taux As Double)**

Visual Basic.NET

Interfaces et collections

Version 1.3

Interface

- **Exprime un comportement que vous tentez de modéliser c'est-à-dire qu'une classe donnée peut pouvoir prendre en charge**
- **Peut prendre en charge un nombre illimité de propriétés**
- **Exemple**

```
Public Interface IDisposable
      Sub Dispose()
End Interface
```

Interface

- Toute classe qui hérite d'une interface doit implémenter tous ses éléments
- L'interface n'est qu'un ensemble nommé de membres abstraits
- On peut définir des données Public, Private et Protected ainsi qu'un certain nombre de méthodes concrètes
- Chaque membre de l'interface est abstrait
- Permet l'héritage multiple

Implémentation

- Utilisation du mot-clé "Implements" (après le mot clé "Inherits") pour prendre en charge les fonctionnalités de l'interface
  - Exemple :

```
Public Class MyCollection
   Inherits CollectionBase
   Implements IDisposable

   Public Sub Dispose() Implements IDisposable.Dispose

   End Sub
End Class
```

Interface

- **Remarques :**
  - **Le fait de séparer l'interface de son implémentation se trouve au coeur même de la programmation et de la conception orientée objet**
  - **Par convention, le nom d'une interface commence par la lettre "I".**

Collection

- **Permet de stocker un ensemble d'éléments (objets, variables, …)**
- **System.Array**
  - **Un tableau est une collection simplifiée**
- **System.Collections**
  - **Ensemble d'interfaces standards**

Collection

Interface de System.Collections	Signification
ICollection	Définit des caractéristiques génériques (par exemple, lecture seule, thread, sécurisé, etc.) pour une classe de collection
IComparer	Permet à deux objets d'être comparés
IDictionnary	Permet à un objet de représenter son contenu en utilisant des paires nom/valeur
IDictionnaryEnumerator	Utilisée pour énumérer le contenu d'un objet qui prend en charge Idictionnary
IEnumerable	Renvoie l'interface IEnumerator pour un objet donné
IEnumerator	Utilisée généralement pour prendre en charge chaque itération de sous-type
IHashCodeProvider	Renvoie le code de hachage pour le type qui implémente en utilisant un algorithme de hachage personnalisé
IList	Fournit le comportement pour ajouter, supprimer et indexer des éléments dans une liste d'objets

Collection

Class de System.Collections	Signification
ArrayList	Tableau d'objets dimensionnés dynamiquement
Hashtable	Représente une collection de clés associées et de valeurs qui sont organisées en fonction du code de hachage de la clé.
Queue	Représente une file d'attente stantard FIFO
SortedList	Comme un dictionnaire. On peut aussi accéder aux éléments selon leur position ordinale (index)
Stack	File d'attente LIFO fournissant des fonctionnalités d'empilement, et de dépilement

Collection

- **Création d'une collection**

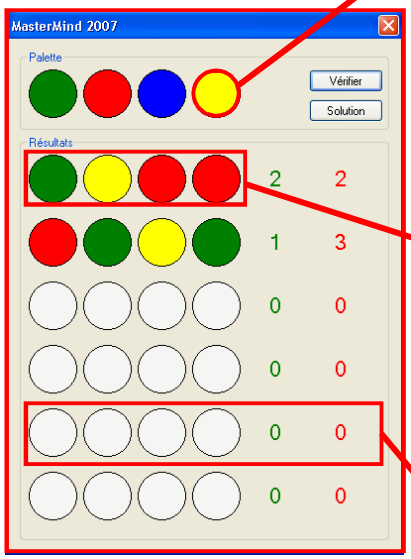
```
Sub showToString()

   Dim collection As New ArrayList
   collection.Add(New Point)
   collection.Add(DateTime.Now)
   collection.Add(New Button)
   collection.Add("Hello")
   collection.Add(New Employee)

   For Each o As Object In collection
        MessageBox.Show(o.ToString())
   Next

End Sub
```

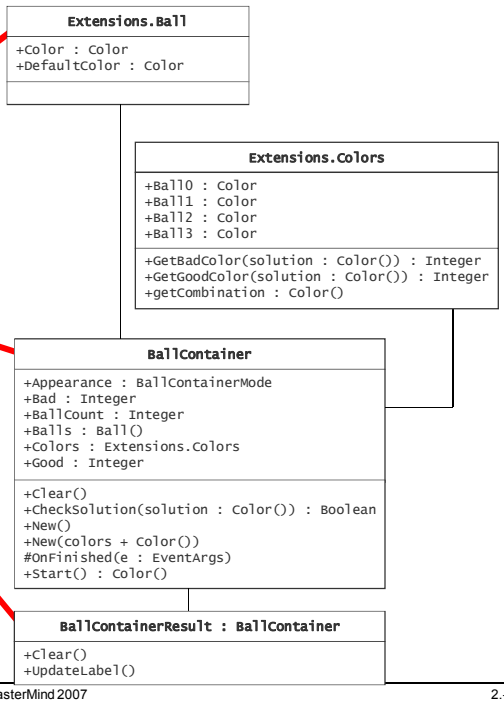
Visual Basic.NET

MasterMind

Version 2.0

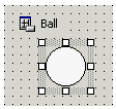
MasterMind



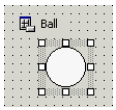


MasterMind

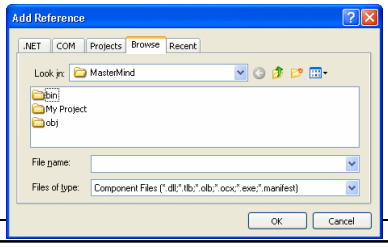
- **Analyse**
  - **Création de deux UserControls**
    - **L'objet "Ball" (voir MasterMind.Extensions)**
    - **L'objet "BallContainer"**
  - **Création de deux fenêtres**
    - **La fenêtre Principale**
    - **La fenêtre Solution**

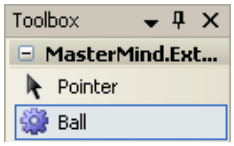


Ball

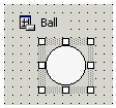


- **Utilisation d'un composant**
  - **Copier la bibliothèque sur votre disque**
  - **Ajouter l'Assembly *MasterMind.Extensions***
    - **Menu Project / Add Reference**



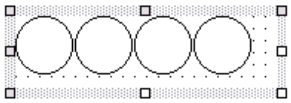


BallContainer



- **Création d'un composant UserControl**
  - **Un UserControl est un objet graphique**
  - **Il hérite de *System.Windows.Forms .UserControl***

BallContainer



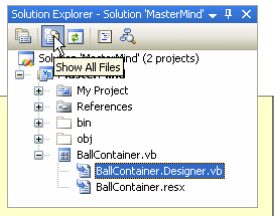
- **Création d'un UserControl qui contient 4 composants « Ball »**
- **Description**

BallContainer
+Appearance : BallContainerMode +Bad : Integer +BallCount : Integer +Balls : Ball() +Colors : Extensions.Colors +Good : Integer
+Clear() +CheckSolution(solution : Color()) : Boolean +New() +New(colors + Color()) #OnFinished(e : EventArgs) +Start() : Color()

BallContainerMode
+Inactive +InitialColor +UserChoice

BallContainer

- **Constructeur**
  - **Le constructeur se trouve dans BallContainer.Design.vb**




```
Public Sub New()

    ' This call is required by the Windows Form Designer.
    InitializeComponent()

    ' Add any initialization after the InitializeComponent() call.

End Sub
```


BallContainer

- **Propriété « Balls »**

```
<Browsable(False)> _
Public ReadOnly Property Balls() As Ball()
    Get
        Return _balls
    End Get
End Property
```

BallContainer

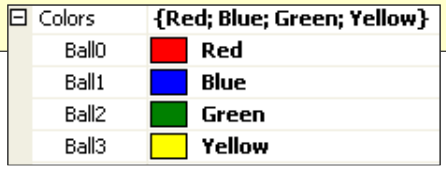
- **Propriété « BallCount »**

```
<Browsable(False)> _
Public ReadOnly Property BallCount() As Integer
   Get
      Return _balls.Length
   End Get
End Property
```


BallContainer

- **Propriété « Colors »**

```
Private _colors As MasterMind.Extensions.Colors

<DesignerSerializationVisibility(DesignerSerializationVisibility.Content), _
Category("Appearance"), Description("Couleurs des boules du Container.")> _
Public ReadOnly Property Colors() As MasterMind.Extensions.Colors
    Get
        Return _colors
    End Get
End Property
```



BallContainer

- **Constructeurs**

```
Public Sub New()
   ' This call is required by the Windows Form Designer.
   InitializeComponent()
   ' Add any initialization after the InitializeComponent() call.
   _balls = New MasterMind.Extensions.Ball() {Ball1, Ball2, Ball3, Ball4}
   [redacted]
End Sub
```

```
Public Sub New(ByVal colors() As Color)
    Me.New()
    _colors.Ball0 = colors(0)
    _colors.Ball1 = colors(1)
    _colors.Ball2 = colors(2)
    _colors.Ball3 = colors(3)
End Sub
```


BallContainer

- **Propriété « Appearance »**

```
Private _mode As BallContainerMode

<Category("Appearance"), Description("Mode du composant.")> _
Public Property Appearance() As BallContainerMode
    Get
        Return _mode
    End Get
    Set(ByVal Value As BallContainerMode)
        _mode = Value
    End Set
End Property
```

```
Public Enum BallContainerMode
    Inactive
    InitialColor
    UserChoice
End Enum
```

BallContainer

- **Méthode « GetCombination »**
  - **Dans MasterMind.Extensions, la méthode *Colors.GetCombination* retourne des couleurs choisies aléatoirements.**

```
Private _solution As Color()

Public Function Start() As Color()
    If Appearance = BallContainerMode.InitialColor Then
        _solution = _colors.GetCombination()
        return _solution
    End If
 Else
    Throw New MethodAccessException("Cette méthode n'est " & _
              accessible que si Appearance = InitialColor.")
End Sub
```


BallContainer

- **Méthode « Clear »**
  - **Remet toutes les boules à la couleur par défaut**

```
Public Sub Clear()
    Dim b As Ball

    For Each b In Me.Balls
        b.Color = Ball.DefaultColor
    Next

    Me.Appearance = BallContainerMode.Inactive

End Sub
```

BallContainer

- **Propriétés « Good » et « Bad »**

```
<Browsable(False)> _
Public ReadOnly Property Good() As Integer
    Get
        Return _colors.GetGoodColors(_solution)
    End Get
End Property

<Browsable(False)> _
Public ReadOnly Property Bad() As Integer
    Get
        Return _colors.GetBadColors(_solution)
    End Get
End Property
```


BallContainer

- **Méthode « CompareTo »**

```
Public Function CheckSolution(ByVal solution As Color()) As Boolean

    _solution = solution

    If Me.Good = Me.BallCount Then
        OnFinished(EventArgs.Empty)
        Return 0
    Else
        Return 1
    End If

End Function
```

BallContainer

- Evénement « Finished »

```
Public Event Finished(ByVal sender As System.Object, _
                      ByVal e As System.EventArgs)


Protected Sub OnFinished(ByVal e As EventArgs)
    RaiseEvent Finished(Me, e)
End Sub
```


BallContainer

- Drag ‘n Drop

1. Activer la propriété « AllowDrop »
2. Enclencher le Drag dans « MouseDown »
3. Modifier le curseur dans « DragEnter »
4. Gérer le Drop dans « DragDrop »

BallContainer

- Drag ‘n Drop
  - Enclencher le Drag dans « MouseDown »
  - Uniquement pour InitialColor

```
Private Sub Ball_MouseDown(ByVal sender As Object, _
                           ByVal e As MouseEventArgs) _
                [redacted]

  If Me.Appearance = BallContainerMode.InitialColor Then
    Dim boule As Ball = CType(sender, Ball)
    boule.DoDragDrop(boule.Color, DragDropEffects.Copy)
  End If

End Sub
```


BallContainer

- Drag ‘n Drop
  - Modifier le curseur dans « DragEnter »
  - Uniquement pour UserChoice

```
Private Sub Ball_DragEnter(ByVal sender As Object, _
                           ByVal e As DragEventArgs) _
            Handles Ball1.DragEnter, Ball2.DragEnter, _
                    Ball3.DragEnter, Ball4.DragEnter

  If Me.Appearance = BallContainerMode.UserChoice Then
    If e.Data.GetDataPresent(GetType(Color)) = True Then
      e.Effect = DragDropEffects.Copy
    End If
  End If

End Sub
```

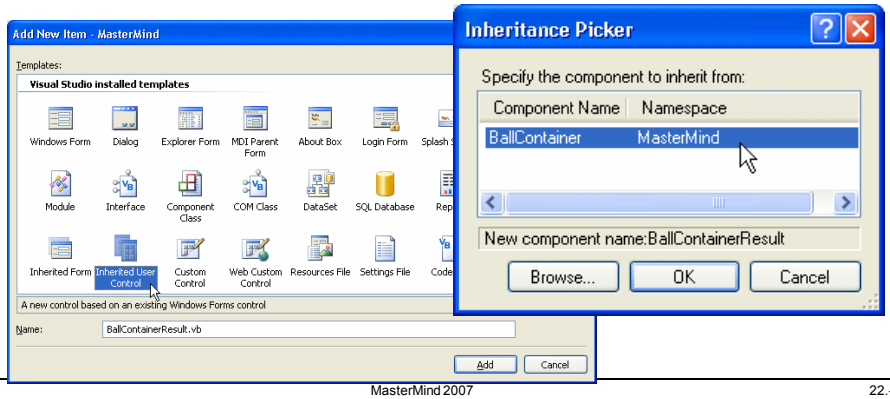
BallContainer

- Drag ‘n Drop
  - Gérer le Drop dans « DragDrop »
  - Uniquement pour UserChoice

```
Private Sub Ball_DragDrop(ByVal sender As Object, _
                          ByVal e As DragEventArgs) _
            Handles Ball1.DragDrop, Ball2.DragDrop, _
                    Ball3.DragDrop, Ball4.DragDrop

  If Me.Appearance = BallContainerMode.UserChoice Then
    Dim boule As Ball = CType(sender, Ball)
    boule.Color = e.Data.GetData(GetType(Color))
  End If

End Sub
```

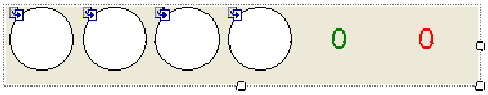

BallContainerResult

- Pour gérer graphiquement les résultats
  - Créer un UserControl
  - Hériter de BallContainer



BallContainerResult

- Pour gérer graphiquement les résultats
  - Ajouter deux Labels (lblGood et lblBad)
  - Propriétés
    - AutoSize = False
    - Font.Size = 16
    - ForeColor = Green | Red
    - TextAlign = MiddleCenter
    - Text = 0



BallContainerResult

- Pour gérer graphiquement les résultats
  - Modifier BallContainer

```
Public [redacted] Sub Clear()

Public [redacted] ReadOnly Property Good() As Integer

...

Public [redacted] ReadOnly Property Bad() As Integer

...
```

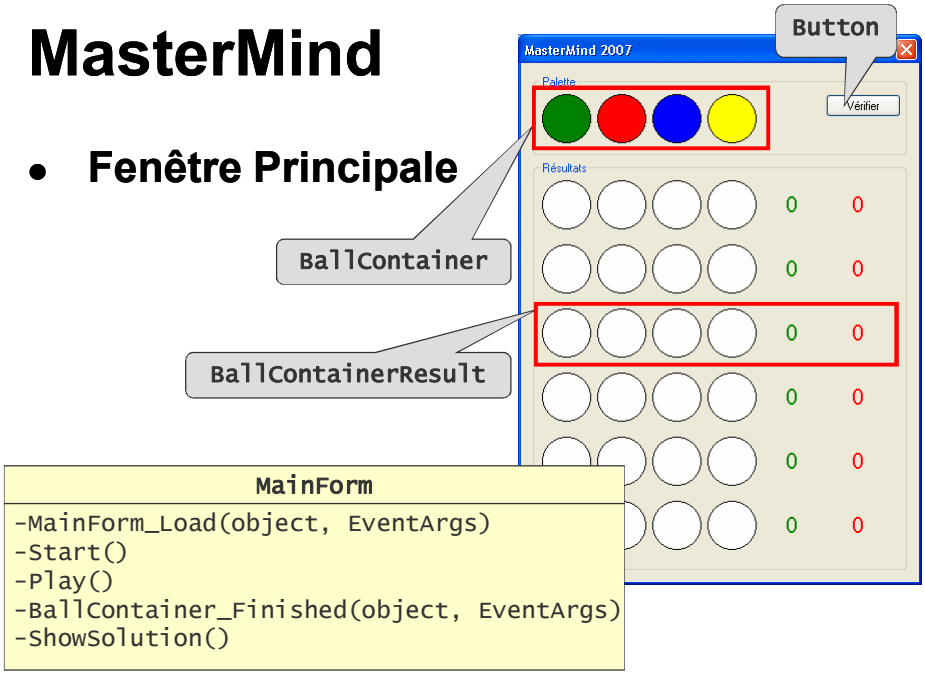
BallContainerResult

- **Pour gérer graphiquement les résultats**
  - **Surcharger *Clear***
  - **Créer UpdateLabel**

```
Public [redacted] Sub Clear()
    MyBase.Clear()
    lblGood.Text = "0"
    lblBad.Text = "0"
End Sub

[redacted]
```


MasterMind

- **Fenêtre Principale**



MainForm

```
-MainForm_Load(object, EventArgs)
-Start()
-Play()
-BallContainer_Finished(object, EventArgs)
-ShowSolution()
```

MasterMindForm

- **Champs**

```
Private _currentPosition As Integer
Private _bcResult(0 To 5) As BallContainerResult
Private _solution As Color()
```


MasterMindForm

- **Initialisation des composants**

```
Private Sub MasterMindForm_Load(ByVal sender As System.Object,
                                ByVal e As System.EventArgs) Handles MyBase.Load
    _bcResult(0) = BallContainerResult1
    _bcResult(1) = BallContainerResult2
    _bcResult(2) = BallContainerResult3
    _bcResult(3) = BallContainerResult4
    _bcResult(4) = BallContainerResult5
    _bcResult(5) = BallContainerResult6

    [redacted]

End Sub
```

MasterMindForm

- **Méthode « Start »**

```
Private Sub Start()
    Dim bc As BallContainer

    'Remise à blanc des BallContainers
    For Each bc In _bcResult
        bc.Clear()
        bc.Appearance = BallContainerMode.Inactive
    Next

    ' Activation du premier BallContainer
    _currentPosition = 0
    _bcResult(0).Appearance = BallContainerMode.UserChoice

    ' Choix des couleurs aléatoires
    _solution = bcInitialColor.Start()
End Sub
```


MasterMindForm

- **Evénement « btnCheck.Click »**

```
Private Sub btnCheck_Click(ByVal sender As System.Object, _
                           ByVal e As System.EventArgs) Handles btnCheck.Click
    'Comparaison
    _bcResult(_currentPosition).CheckSolution(_solution)
    _bcResult(_currentPosition).UpdateLabel()

    'Résultat
    If _currentPosition >= 5 Then
        MessageBox.Show("Vous avez perdu.")
        Start()
    Else
        'Activation de la position suivante
        _currentPosition += 1
        _bcResult(_currentPosition).Appearance = BallContainerMode.UserChoice

        'Remise en état de la position précédente
        _bcResult(_currentPosition - 1).Appearance = BallContainerMode.Inactive
    End If
End Sub
```

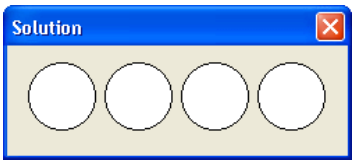
MasterMindForm

- **Evénement « Finished »**

```
Private Sub BallContainer_Finished(ByVal sender As Object, ByVal e As EventArgs) _
            Handles BallContainerResult1.Finished, BallContainerResult2.Finished, _
                    BallContainerResult3.Finished, BallContainerResult4.Finished, _
                    BallContainerResult5.Finished, BallContainerResult6.Finished

    MessageBox.Show("Félicitation... vous avez gagné")

    Start()

End Sub
```


SolutionForm

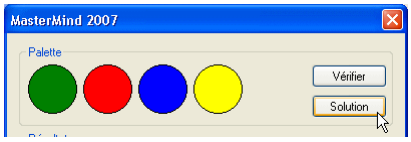
- **Méthode « ShowSolution »**
  - **Créer un formulaire *SolutionForm***
  - **Placer un BallContainer**
  - **Créer un constructeur**




```
Public Sub New(ByVal solution As Color())

    InitializeComponent()

    BallContainer1.Colors.Ball0 = solution(0)
    BallContainer1.Colors.Ball1 = solution(1)
    BallContainer1.Colors.Ball2 = solution(2)
    BallContainer1.Colors.Ball3 = solution(3)

End Sub
```

SolutionForm

- **Afficher la solution**
  - **Si perdu**
  - **Pour *tricher***

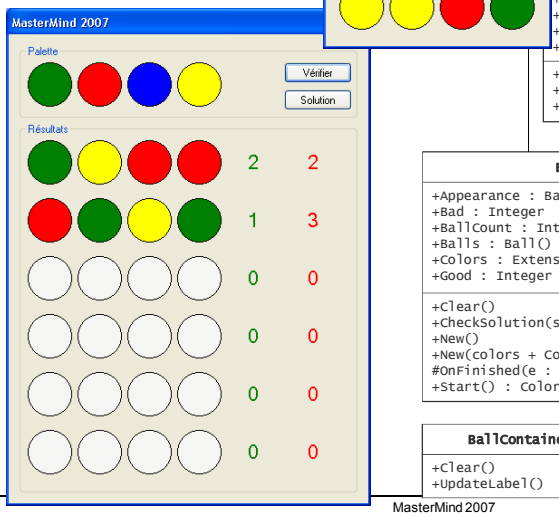


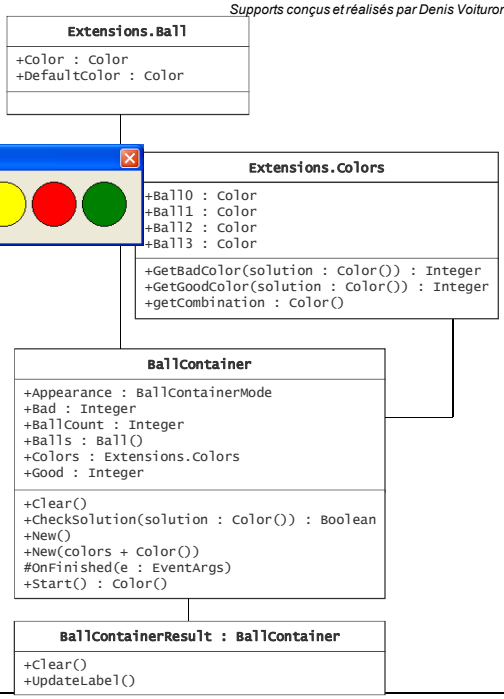

```
Private Sub btnCheck_Click(ByVal sender As System.Object, _
                           ByVal e As System.EventArgs) Handles btnCheck.Click
    ...

    'Résultat
    If _currentPosition >= 5 Then
        MessageBox.Show("Vous avez perdu.")
        [redacted]
        Start()
    Else
        ...
    End If
End Sub
```

```
Private Sub btnSolution_Click ...
    Dim frm As New SolutionForm(_solution)
    frm.Show()
End Sub
```


MasterMind

- **Conclusion**





Visual Basic .Net

Particularités
de VB.Net 2.0

Version 2.0

Nouveautés de VB.Net 2.0

- Les classes partielles
- Les types unsigned
- La surcharge d'opérateur
- Les instances par défaut
- Les blocs Using
- L'opérateur IsNot
- L'objet My
- Le mot-clé Continue
- Les generics
- Autres nouveautés

Les classes partielles

- **Principe : diviser une même classe en plusieurs fichiers.**

```
' Fichier Person1.vb
Partial Public Class Person

    Private _firstName As String = String.Empty
    Private _lastName As String = String.Empty
    Private _age As Integer = 0

End Class

' Fichier Person2.vb
Partial Public Class Person

    Public Overrides Function ToString() As String
        Return _firstName + " " + _lastName.ToUpper()
    End Function

End Class
```


Les classes partielles

- **Caractéristiques**
  - **Les membres des classes partielles sont partagés entre tous les fichiers.**
  - **« Fusion » des classes à la compilation.**
- **Avantages**
  - **Le travail collaboratif est facilité.**
  - **Améliore la lisibilité du code.**
- **Conseils**
  - **Nécessité d'adopter une convention de nom pour les fichiers !**
  - **Ne pas définir d'autres classes dans les divers fichiers de classes partielles !**

Les types unsigned

- VB.Net peut maintenant utiliser les types entiers non signés.
  - UShort, UInteger, ULong et SByte.
- Avantages
  - Permettent d'obtenir de meilleures performances sur les plateformes 32 bits.
  - Étendent la plage de valeurs (positives) possibles.

La surcharge d'opérateur

- La surcharge d'opérateur permet de donner aux opérateurs un comportement spécifique quand ils sont appliqués à des types spécifiques.

```
Public Shared Operator +(ByVal p1 As Person, ByVal p2 As Person) _
As Integer

        Return p1.Age + p2.Age

End Operator
```

- Opérateurs surchargeables :
  - +, -, *, /, ^, &, <, >, =, <<, >>, And, Or, Xor, Not, etc.
- Il faut « selon les cas » surcharger l'inverse !
  - Exemple : surcharger > quand on surcharge <

Les instances par défaut

- Les programmeurs venant de VB6 se plaignaient souvent de devoir instancier une Form (en VB.Net) avant de l'afficher.

```
Dim myForm As New Form2
myForm.Show()
```

- Dorénavant, il est possible d'utiliser l'instance par défaut.

```
Form2.Show()
```


Les blocs Using

- Le bloc Using ... End Using permet de s'assurer de la libération des ressources allouées par un objet défini.

```
Public Sub MyMethod()

Using conn As New SqlConnection("myConnectionString")

        conn.Open()

        Dim mySqlCommand As New SqlCommand("mySqlCommand", conn)
        mySqlCommand.ExecuteNonQuery()

    End Using

End Sub
```

- A la fin du bloc Using, appel implicite de la méthode Dispose() de l'objet.

L'opérateur IsNot

- Le nouvel opérateur IsNot est un mélange des opérateurs Is et Not.

```
' VB.Net 1.x
If Not (myObject Is Nothing) Then

MessageBox.Show("myObject n'est pas nul")

End If

' VB.Net 2.0
If myObject IsNot Nothing Then

MessageBox.Show("myObject n'est pas nul")

End If
```


L'objet My

- L'objet My permet d'accéder à un très grand nombre de fonctionnalités.
- My.Application
  - Contexte d'exécution de l'application, culture,
  - Arguments, logs, déploiement (ClickOnce),
  - Splash screen, etc.
- My.Computer
  - Informations sur le système (mémoire, etc.),
  - Travailler sur le registre, le presse-papier,
  - Accès à la souris, au clavier, aux ports, à l'écran,
  - Jouer des sons Wav, etc.

L'objet My

- My.Forms
  - Permet d'accéder à l'ensemble des formulaires du projet.
- My.Resources
  - Permet de travailler sur différents types de ressources (audio, icône, bitmap, etc.).
- My.Settings
  - Permet de travailler sur les paramètres de l'application (fichier app.config).
- My.User
  - Permet d'obtenir des informations sur l'utilisateur (nom, authentification, etc.).
- My.WebServices
  - Fournit une instance de chaque Service Web du projet.

Le mot-clé Continue

- Le mot-clé Continue permet de passer à l'itération suivante d'une boucle sans exécuter le code "restant" dans cette boucle. C'est une rupture de séquence.

```
For i As Integer = 0 To 10

' Si i=3, alors pas de message dans la console.
If i = 3 Then Continue For

Console.WriteLine(i.ToString())

Next
```

- Il existe également le "Continue While" et le "Continue Do".

Les generics

- **Principe : permettent de définir des classes et méthodes sans indiquer un type particulier (généralisation).**

```
Dim a As Integer = 1, b As Integer = 5, c As Integer = 3, _
        total As Integer = 0
Dim myIntegerList As New List(Of Integer)

myIntegerList.Add(a)
myIntegerList.Add(b)
myIntegerList.Add(c)

For Each value As Integer In myIntegerList

        total += value

Next

MessageBox.Show("Total : " + total.ToString())
```


Les generics

- **Avantages :**
  - **Le type est connu à l'avance.**
    - **Sécurisation des opérations sur la collection.**
  - **Plus de boxing / unboxing.**
  - **Facilité d'utilisation par rapport aux tableaux.**
  - **Possibilité de créer des classes et méthodes réutilisables à volonté (quel que soit le type).**

Autres

- Support du 64 bits.
- Utilisation du protocole FTP
  - FtpWebRequest, FtpWebResponse, etc. (System.Net).
- Contrôle du réseau
  - Obtenir de nombreuses informations du réseau (System.Net.NetworkInformation).
- Support SMTP
  - Envoi de mails personnalisés (System.Net.Mail, System.Net.Mime).

Autres

- Amélioration de la console.
- Communication avec un port série : System.IO.Ports.SerialPort.
- Etc.

Support réalisé par Gaëtan Monseu

Visual Basic.NET

Active Data Object
.NET

Version 2.0

Structure

- Introduction
- Architecture Client/Server
- Objets principaux de l'ADO.NET
- Connection
- Command
- DataReader
- DataAdapter et DataSet
- Mise à jour du DataSet
- CommandBuilder

Introduction

- **Active Data Objects – ADO**
  - **Accéder aux données d'un serveur**
  - **Manipuler des données d'un serveur**
  - **A travers un provider OLE DB ou directement vers SQL Server**
- **Avantages**
  - **Facilité d'utilisation**
  - **Grande vitesse d'accès**
  - **Faible consommation de mémoire**

Introduction

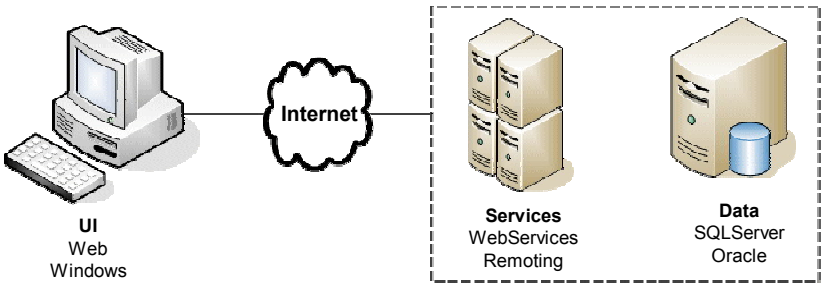
- **ADO.NET**
  - **est un modèle d'objets**
  - **pour accéder et modifier les données d'une base de données d'un serveur**
- **Objets**
  - **Propriétés**
  - **Méthodes**
  - **Evénements**

Structure

- Introduction
- Architecture Client/Server
- Objets principaux de l'ADO.NET
- Connection
- Command
- DataReader
- DataAdapter et DataSet
- Mise à jour du DataSet
- CommandBuilder

Introduction

- Exploitation d'un Client / Serveur



Server

- **Une large plage de solutions**

➢ Microsoft SQL Server ➢ Microsoft Data Engine – MSDE	`System.Data.SqlClient`
➢ Microsoft Access ➢ Sybase ➢ DB2	`System.Data.OleDb`
➢ Oracle Server	`System.Data.OracleClient`

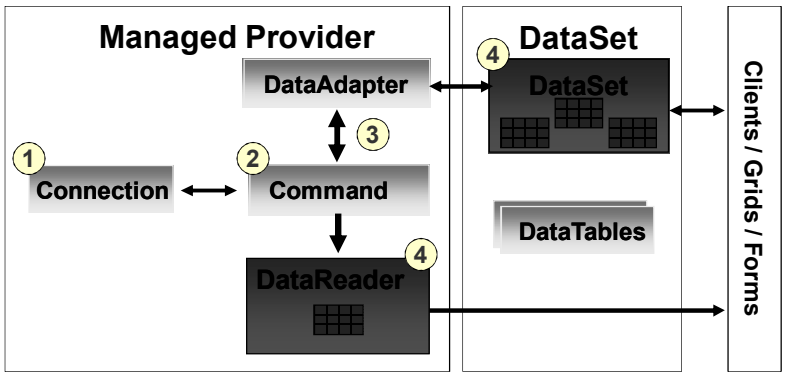
Structure

- **Introduction**
- **Architecture Client/Server**
- **Objets principaux de l'ADO.NET**
- **Connection**
- **Command**
- **DataReader**
- **DataAdapter et DataSet**
- **Mise à jour du DataSet**
- **CommandBuilder**

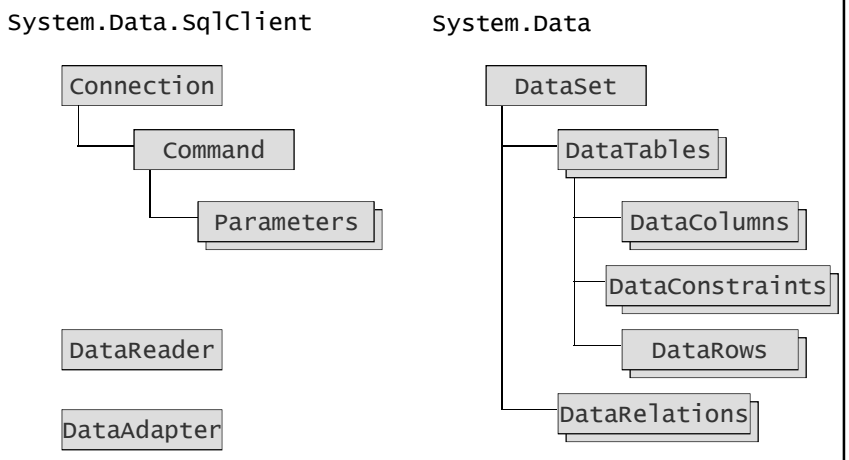
Connexion vers une BDD

- **Pour gérer des données :**
  - **Se connecter vers une source de données**
  - **Spécifier une commande (language SQL)**
  - **Exécuter la commande**
  - **Si la commande retourne des lignes, récupérer ces données dans le DataSet ou le DataReader**

Objets de l'ADO.NET



Objets de l'ADO.NET



Structure

- Introduction
- Architecture Client/Server
- Objets principaux de l'ADO.NET
- Connection
- Command
- DataReader
- DataAdapter et DataSet
- Mise à jour du DataSet
- CommandBuilder

Connexion

- Pour établir une connexion d'échange de données vers le serveur
- Définir un objet de type Connection
- Définir une chaîne de connexion
- Ouvrir la connexion

Connexion

- Chaîne de connexion
  - Paramètres de connexion vers SQL Server
    - Data Source = Localhost;
    - Initial Catalog = Northwind;
    - User ID = sa;
    - Password = xyz123;

Connexion - Exemple

```
Dim con As New SqlConnection
con.ConnectionString = "Data Source=localhost;" & _
                       "Initial Catalog=Northwind;" & _
                       "User ID=sa;" & _
                       "Password=;"
con.Open()
MessageBox.Show(con.ServerVersion)
con.Close()
con.Dispose()
```


Structure

- Introduction
- Architecture Client/Server
- Objets principaux de l'ADO.NET
- Connection
- Command
- DataReader
- DataAdapter et DataSet
- Mise à jour du DataSet
- CommandBuilder

Command

- **Commande SQL pour**
  - **Manipuler les données**

```
INSERT INTO Customers(CustomerID, CompanyName)
VALUES('MS', 'Microsoft')
```

ExecuteNonQuery()

  - **Récupérer des données**

```
SELECT COUNT(*)
FROM Customers
```

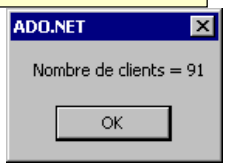
ExecuteScalar()

```
SELECT *
FROM Customers
```

ExecuteReader()

Command - Exemple

```
...
Dim com As New SqlCommand
com.Connection = con
com.CommandText = "SELECT COUNT(*) FROM Customers"
MessageBox.Show("Nombre de clients = " & _
cmd.ExecuteScalar())
...
```



Structure

- Introduction
- Architecture Client/Server
- Objets principaux de l'ADO.NET
- Connection
- Command
- DataReader
- DataAdapter et DataSet
- Mise à jour du DataSet
- CommandBuilder

DataReader

- DataReader
  - C'est un tableau de lignes
  - Passer à la ligne suivante par .Read()
  - On ne peut pas modifier les données de DataReader
  - La connexion vers le serveur est permanente
  - Très rapide d'utilisation

DataReader - Exemple

```
...
Dim dr As SqlClient.SqlDataReader
dr = com.ExecuteReader()
ListBox1.Items.Clear()
Do While (dr.Read())
   ListBox1.Items.Add(dr("CompanyName"))
Loop
dr.Close()
...
```


Structure

- Introduction
- Architecture Client/Server
- Objets principaux de l'ADO.NET
- Connection
- Command
- DataReader
- DataAdapter et DataSet
- Mise à jour du DataSet
- CommandBuilder

DataAdapter

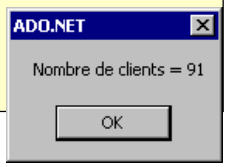
- **DataAdapter**
  - **Il représente un ensemble de commandes configurables ainsi que la connexion vers la base, afin de remplir l'objet DataSet**
  - **La connexion vers le serveur n'est pas permanente**

DataAdapter

- **Plusieurs commandes sont disponibles**
  - **SelectCommand** — `Utilisée avec DataSet.Fill`
  - **InsertCommand**
  - **UpdateCommand** — `Utilisée avec DataSet.Update`
  - **DeleteCommand**

DataAdapter - Exemple

```
...
Dim da As New SqlDataAdapter
Dim ds As New Data.DataSet
da.SelectCommand = com
da.Fill(ds, "MesClients")
MessageBox.Show("Nombre de lignes = " & _
      ds.Tables("MesClients").Rows.Count)
ds.Dispose()
da.Dispose()
...
```



DataAdapter - Exemple

```
...
Dim r As DataRow
ListBox1.Items.Clear()
For Each r In ds.Tables("MesClients").Rows
   ListBox1.Items.Add(r("CompanyName"))
Next
...
```

DataBinding

- **Liaison automatique vers des Win Forms**

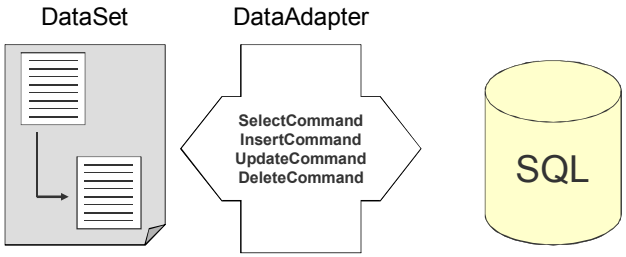
```
da.SelectCommand = com
da.Fill(ds, "MesClients")

DataGrid1.SetDataBinding(ds,"MesClients")
```


Structure

- **Introduction**
- **Architecture Client/Server**
- **Objets principaux de l'ADO.NET**
- **Connection**
- **Command**
- **DataReader**
- **DataAdapter et DataSet**
- **Mise à jour du DataSet**
- **CommandBuilder**

Modification des données

- **Utiliser les requêtes SQL**
  - **INSERT, UPDATE, DELETE**
- **Utiliser le DataSet**



DataSet.UpDate

- **Pour modifier des données du DataSet**
  - **Définir « UpdateCommand »**
    - **CommandText**
    - **Parameters**
  - **Modifier les données**
  - **Utiliser la méthode « Update »**
  - **Exploiter les événements RowUpdating et RowUpdated**

DataSet.UpDate -Exemple

```
Dim con As New SqlConnection()
Dim da As New SqlDataAdapter()
Dim ds As New DataSet()
'Définition de la commande SELECT
da.SelectCommand = New SqlCommand("SELECT CompanyName, CustomerID FROM" & _
                                                    " Customers", con)
'Définition de la commande UPDATE
da.UpdateCommand = New SqlCommand("UPDATE Customers SET
   CompanyName=@CompanyName WHERE CustomerID=@CustomerID", con)
'Définition des paramètres
da.UpdateCommand.Parameters.Add("@CompanyName", SqlDbType.NVarChar, 40,
   "CompanyName")
da.UpdateCommand.Parameters.Add("@CustomerID", SqlDbType.NChar, 5,
   "CustomerID")
da.Fill(ds, "MesClients")
ds.Tables(0).Rows(0)(0) = "Dupont"
MessageBox.Show("Nb modifiées : " & da.Update(ds, "MesClients"))
```

1

2

3

DataSet.RowUpDated

- **Utiliser l'événement pour**
  - **Vérifier les mises à jour (ligne par ligne)**
  - **Gérer les erreurs (lock, …)**

```
Private WithEvents da As New SqlDataAdapter()
Public Sub da_RowUpdated(ByVal sender As Object, ByVal e As _
                SqlClient.SqlRowUpdatedEventArgs) Handles da.RowUpdated
   If (e.RecordsAffected=0) Then
      MessageBox.Show("Update Error")
      e.Status = UpdateStatus.SkipCurrentRow
   End If
End Sub
```

DataSet.Tables.NewRow

- **Pour ajouter une ligne à la table d'un DataSet**
  - **Définir « InsertCommand »**
    - **CommandText**
    - **Parameters**
  - **Créer un objet DataRow sur base d'une ligne existante dans la table et y définir les valeurs des champs**
  - **Ajouter cet objet à la table**
  - **Utiliser la méthode « Update »**

DataSet.Tables.NewRow - Exemple

```
Dim r As DataRow
...
'Définition de la commande INSERT                                          1
da.InsertCommand = New SqlClient.SqlCommand("INSERT INTO Customers" & _
      "(CompanyName, CustomerID) VALUES (@CompanyName , @CustomerID)", con)

'Définition des paramètres
da.InsertCommand.Parameters.Add("@CompanyName", SqlDbType.NVarChar, 40,
      "CompanyName")
da.InsertCommand.Parameters.Add("@CustomerID", SqlDbType.NChar, 5,
      "CustomerID")

da.Fill(ds, "MesClients")                                                  2
r = ds.Tables(0).NewRow()
r("CompanyName") = "MS"
r("CustomerID") = "Microsoft"
ds.Tables(0).Rows.Add(r)
MessageBox.Show("Nb modifiées : " & da.Update(ds, "MesClients"))           3
```

DataSet.Tables.Rows.Delete

- **Pour Supprimer une ligne à la table d'un DataSet**
  - **Définir « DeleteCommand »**
    - **CommandText**
    - **Parameters**
  - **Rechercher la ligne à supprimer**
  - **Supprimer la ligne**
  - **Utiliser la méthode « Update »**

DataSet.Tables.Rows.Delete - Exemple

```
Dim r AsDataROw
...
'Définition de la commande DELETE                                    (1)
da.DeleteCommand = New SqlClient.SqlCommand("DELETE Customers WHERE" & _
        " CustomerID=@CustomerID", con)
'Définition des paramètres
da.DeleteCommand.Parameters.Add("@CustomerID", SqlDbType.NChar, 5, _
        "CustomerID")
da.Fill(ds, "MesClients")

For Each r In ds.Tables(0).Rows                                      (2)
   If r("CompanyName") = "Microsoft" Then
      r.Delete()                                                     (3)
   End If
Next
MessageBox.Show("Nb modifiées : " & da.Update(ds,"MesClients"))
```

Structure

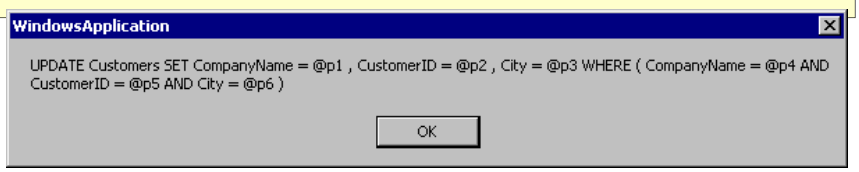
- Introduction
- Architecture Client/Server
- Objets principaux de l'ADO.NET
- Connection
- Command
- DataReader
- DataAdapter et DataSet
- Mise à jour du DataSet
- CommandBuilder

CommandBuilder

- Afin de simplifier la construction des commandes, ADO.NET propose un constructeur automatique de commandes
  - Associer le DataAdapter
  - Rafraîchir la construction des commandes:
    - GetUpdateCommand
    - GetInsertCommand
    - GetDeleteCommand

CommandBuilder - Exemple

```
Dim cb As New SqlCommandBuilder()
...
'Définition de la commande SELECT
da.SelectCommand = New _
     SqlClient.SqlCommand("SELECT CompanyName, CustomerID, _
        City FROM Customers", con)

cb.DataAdapter = da                                    (1)
cb.RefreshSchema()                                     (2)
MessageBox.Show(cb.GetUpdateCommand().CommandText)
```



Visual Basic.NET

Modificateurs

Version 2.0

C# vs VB.NET

C#	VB.NET	Explication
public	**Public**	Accessible dans et hors de l'assembly
internal	**Friend**	Accessible uniquement dans l'assembly
private	**Private**	Accessible uniquement dans la classe
protected	**Protected**	Accessible dans la classe courante et dans les classes dérivées
static	**Shared**	Partagé par toutes les instances de la classe
virtual	**Overridable**	Membre surchargeable
new	**Shadows**	Surcharge d'un membre non prévue (par virtual ou Overridable)
sealed	**NotOverridable**	Membre qui ne peut plus être hérité
abstract	**MustOverride**	Membre qui doit être implémenté dans la classe dérivante

Public – Private – Protected