Interface et Implements

Les **Interface**, qui peuvent être considérées comme des ***Class* Interface**, afin de ne pas les confondre avec les interfaces de classes ou les interfaces utilisateurs (ou **IHM** pour Interface Homme Machine), permettent la déclaration d'ensembles de membres sans code. Ces déclarations, qui ne peuvent être que des **Property**, **Sub**, **Function** et **Event**, ne peuvent pas être assorties d'attributs de portée, ni des fins de blocs usuels (**End Property**, **End Sub**, …). Les codes des membres ainsi déclarés doivent être tous produits dans les classes qui implémentent l'**Interface**. Cela signifie notamment que plusieurs classes implémentant les mêmes **Interface** peuvent offrir une interface de classe identique, ou contenant au moins une même liste de membres, ceux définis dans les **Interface** implémentés.

```
Public Interface MonInterface
  Sub UneProcedure(ByVal X As Integer)
  Function UneFonction(ByVal S As String) As Integer
  Property UnePropriete()
End Interface

Public Class UneClasse
  Implements MonInterface
  Public Sub UneProcedure(ByVal Entier As Integer)
                                              Implements MonInterface.UneProcedure
    ' … … …
  End Sub
  Public Function UneFonction(ByVal Chaine As String) As Integer
                                               Implements MonInterface.UneFonction
    ' … … …
  End Function

  Public Property UnePropriete() Implements MonInterface.UnePropriete
    Get
      ' … … …
    End Get
    Set(ByVal Value)
      ' … … …
    End Set
  End Property
  ' … … …
End Class
```

L'héritage multiple n'est pas possible en VB, mais l'utilisation des **Interface** permet de résoudre sa programmation le cas échéant. Ainsi, l'illustration de l'héritage multiple présentée précédemment est réalisable par l'usage des **Interface**. Les classes définies dans l'illustration ne possèdent pas de membres homonymes et les **Interface** les représentant peuvent être écrites comme ceci :

```
Public Interface IVehicule
    Property Proprietaire()
    Property Marque()
    Property NoPlaque()
    Sub Avancer(ByVal Distance As Integer)
    Sub Reculer(ByVal Distance As Integer)
    Sub Arreter()
End Interface

Public Interface IVoiture
    Inherits IVehicule
    Property NbPlaces()
    Function Liste_Passagers() As String()
End Interface

Public Interface ICamion
    Inherits IVehicule
    Property Tonnage()
    Property TypeChargement()
End Interface
```

```
Public Interface ICamionnette
    Inherits IVoiture
    Inherits ICamion
    Property Type_Utilisation()
End Interface

Public Class ClCamionnette
    Implements ICamionnette
    ' … et il n'y plus qu'à programmer chaque membre de chaque Interface …
```

Il est permis de donner aux membres implémentés et à leurs arguments, des noms locaux différents de ceux déclarés dans l'**Interface** d'origine.

```
Public Interface MonInterface
  Sub UneProcedure(ByVal X As Integer)
End Interface

Public Class UneClasse
  Implements MonInterface
  Public Sub MaSub(ByVal Entier As Integer) Implements MonInterface.UneProcedure
     ' … … …
  End Sub
```

Mais l'**Interface** n'a pas pour vocation première de combler l'absence de mécanisme d'héritage multiple en VB.Net. La création d'une procédure de tri capable de trier n'importe quel objet est un exemple plus approprié de son usage.

…