Vk

Zeste de savoir

Les interfaces en VBA

vendredi 13 septembre 2024

Table des matières

Introduction

En conclusion du billet sur les classes, nous avions dit que le système de POO est moins poussé en VBA que dans d'autres langages.

Nous pouvons citer l'absence d'héritage notamment. L'héritage est un moyen de créer une classe spécialisée à partir d'une autre. Par exemple, nous pourrions imaginer une classe `ClientParticulier` et une classe `ClientEntreprise` héritant toutes deux d'une classe `Client` afin d'étendre ou de modifier son comportement.

Néanmoins, il est tout de même possible de travailler avec des interfaces en VBA. Une interface est une sorte de contrat qui stipule ce que chaque classe implémentant cette interface doit définir. Cela est utile lorsque des objets de types différents doivent pouvoir être utilisés, en partie ou totalement, de manière similaire, par des propriétés ou des méthodes ayant un nom et un sens commun.

Par exemple, un canard, *Superman* et un avion sont tous les trois capables de voler, mais pas tout à fait pareil…. Nous pourrions imaginer une interface stipulant le nécessaire pour voler tel qu'une propriété altitude, une méthode décoller, une méthode atterrir, ….

Ce qu'il est important de noter ici, c'est que cette interface indique aux classes qu'elles doivent implémenter ces éléments, mais pas *comment*, laissant la liberté d'implémentation à celles-ci. Ainsi, en reprenant notre exemple, nous pourrions donc implémenter une méthode décoller avec un comportement différent (le canard ferait coin-coin, battrait des ailes et s'envolerait à petite vitesse tandis que l'avion allumerait les moteurs, accélérerait sur la piste puis s'envolerait à moyenne vitesse, et que *Superman* passerait en tenue adéquate de façon discrète avant de lever le poing et de s'envoler à grande vitesse en poussant sur ses jambes). Nous pourrions aussi choisir de rendre privée l'altitude de vol de *Superman* pour des raisons de confidentialité. Il faut juste qu'en tout temps le contrat soit respecté.

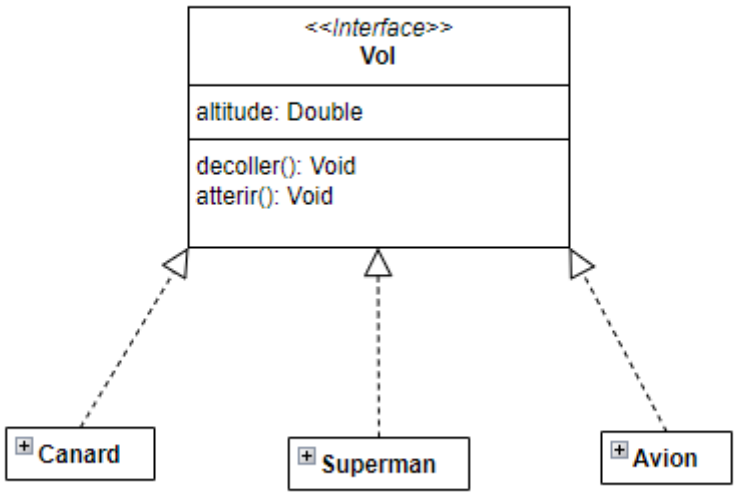

FIGURE 0.1. – Implémentation interface de vol par trois classes.

Au cours de ce billet, nous allons nous intéresser aux interfaces en VBA.

C'est parti !

Pour ce billet, nous nous placerons dans l'environnement Microsoft Office.

1. Définition

Comme indiqué en introduction, une interface est une sorte de contrat qui dit ce que doit définir les classes l'utilisant, mais pas comment.

Pour créer une interface, nous devons ajouter un module de classe. Pour son nom, il est d'usage en VBA de le préfixer par `I`.

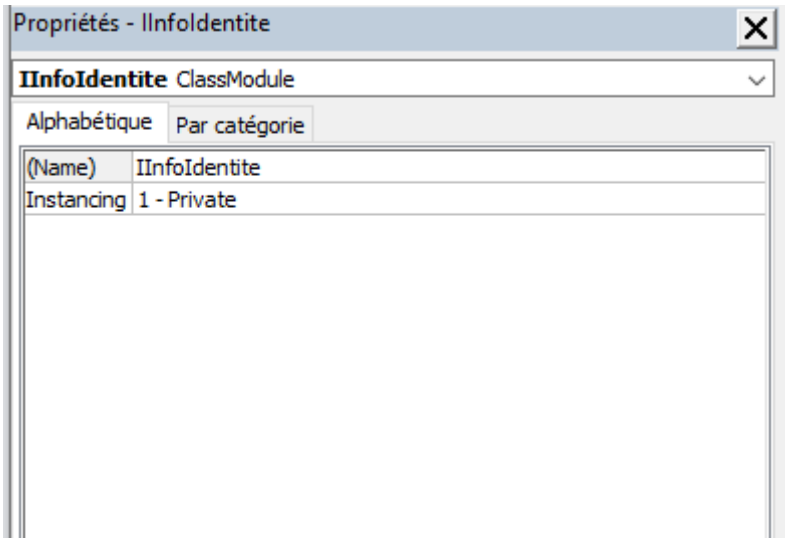

FIGURE 1.2. – Propriétés interface IInfoIdentite

Notre module de classe étant créé, nous pouvons ajouter des propriétés ou méthodes. Il suffit de les définir en `Public` pour qu'elles fassent partie de l'interface :

Interface IInfoIdentite

```
1  Option Explicit
2
3  Public Property Let sNom(ByVal sNom As String)  ' Propriété
       procédure property
4  End Property
5
6  Public Property Get sNom() As String ' Propriété procédure property
7  End Property
8
9  Public Property Set oAdresse(ByRef oAdresse As clsAdresse) '
       Propriété procédure property
10 End Property
```

Les propriétés en tant que variables membres (`mNom` par exemple) seront à définir dans les classes au moment de l'implémentation.

Pour que notre exemple compile, rajoutons une classe `clsAdresse`.

Classe clsAdresse

```
1  Option Explicit
2
3  ' Rien
```

Nous pouvons ajouter autant d'interfaces que nous le souhaitons. Une bonne pratique est d'avoir de petites interfaces spécifiques plutôt qu'une grosse interface générale. En effet, cela évite ainsi d'implémenter du code inutile, et il est plus simple de comprendre le champ d'actions des interfaces utilisées. C'est un des principes (le I) de l'approche SOLID .

Interface IInfoProfessionnelle

```
1 Option Explicit
2
3 Public sSiret As String ' Équivalent de Propriété procédure
      property Let + Get
```

Comme nous allons travailler avec des dictionnaires en liaison anticipée, nous ajoutons la référence *Microsoft Scripting Runtime*.

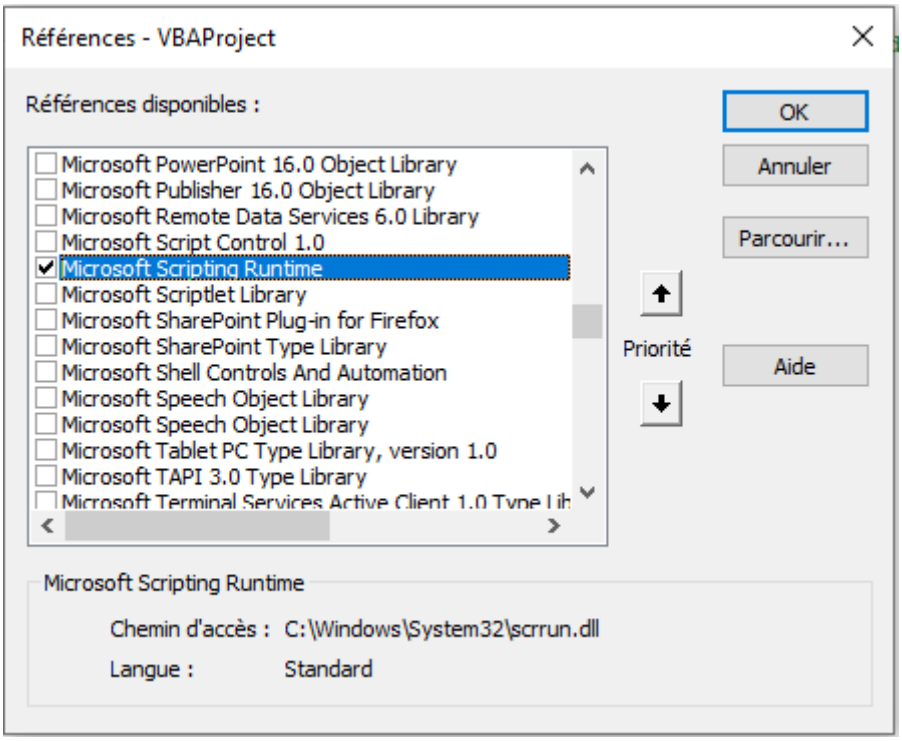

FIGURE 1.3. – Référence Microsoft Scripting Runtime

Interface ICommande

```
1 Option Explicit
2
3 Public Function NombreCommandes() As Integer ' Méthode
4 End Function
5
```

```
6 Public Sub AjouteCommande(ByRef dictCommande As
      Scripting.Dictionary)  ' Méthode
7 End Sub
```

Et une dernière pour la route :

Interface IAffichable

```
1 Option Explicit
2
3 Public Function ToString() As String ' Méthode
4 End Function
5
6 Public Sub Affiche() ' Méthode
7 End Sub
```

Au cours de cette première section, nous avons vu comment définir des interfaces en VBA.

2. Utilisation

Pour utiliser une interface, on dit qu'on l'implémente d'où le mot-clef `Implements` suivi du nom de l'interface.

Classe clsClient avant implémentation complète d'IInfoIdentite

```
1 Option Explicit
2
3 Implements IInfoIdentite
4 ' ...
```

Notons que toutes les classes implémentent implicitement une interface : celle par défaut de la classe.

À ce stade, si nous compilons le projet, nous obtiendrons une erreur de compilation. C'est normal, car nous n'avons pas donné la définition du contenu de l'interface. Autrement dit, nous n'avons pas respecté le contrat ! Cela veut aussi dire que lorsque notre interface évoluera, il faudra faire le nécessaire pour continuer à respecter ce contrat.

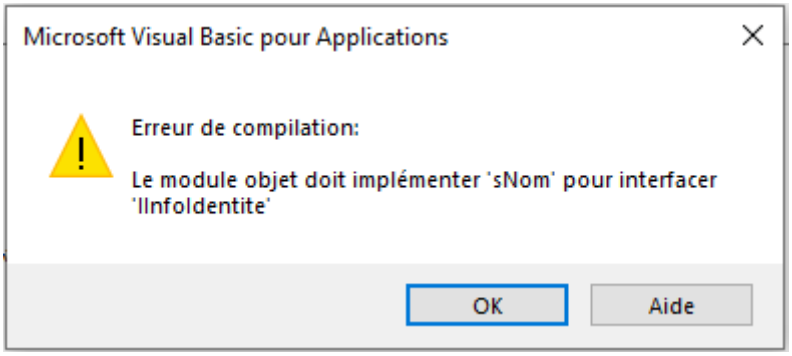

FIGURE 2.4. – Erreur compilation lorsque le contrat n'est pas respecté

Une petite astuce pour rajouter les éléments à implémenter est de passer par les zones de sélection en haut de l'éditeur de code.

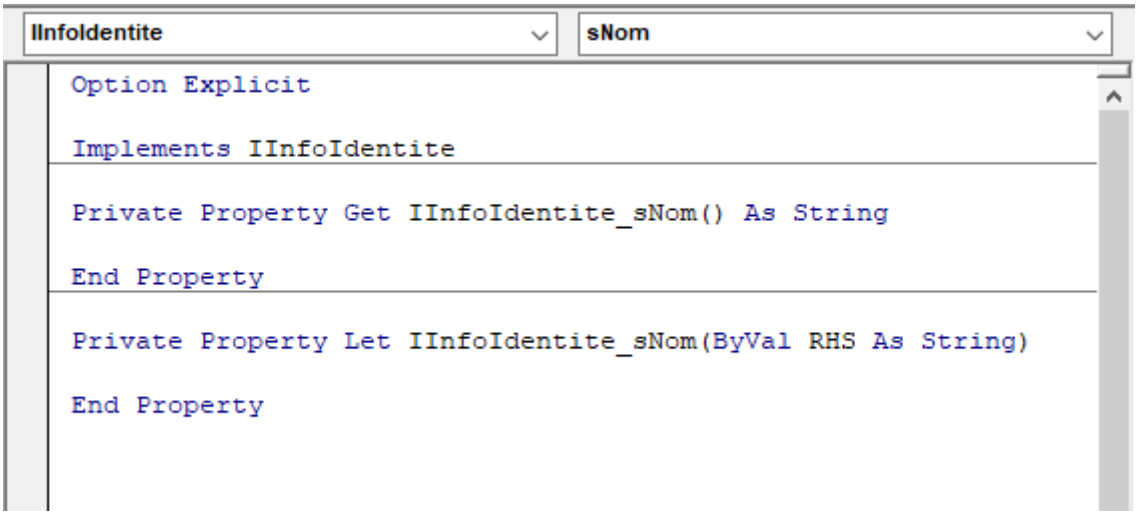

FIGURE 2.5. – Zone d'objet et zone de procédures/événements

Nous pouvons bien entendu écrire le code nous-mêmes, mais il faut faire attention à la syntaxe. Comme nous pouvons le voir dans l'image ci-dessus, le nom est construit sous la forme `NomInterface_NomProprieteOuMethodeADefinir`.

Classe clsClient après implémentation complète d'IInfoIdentite

```
1  Option Explicit
2
3  Implements IInfoIdentite
4
5  Private mNom As String
6  Private mAdresse As clsAdresse
7
8  Public Property Get IInfoIdentite_sNom() As String
9      IInfoIdentite_sNom = mNom
```

```
End Property

Public Property Let IInfoIdentite_sNom(ByVal sNom As String)
    mNom = sNom
End Property

Public Property Set IInfoIdentite_oAdresse(ByRef oAdresse As
    clsAdresse)
    Set mAdresse = oAdresse
End Property
```

Nous pouvons continuer sur notre lancée et implémenter notre interface `ICommande`. Pour cela, nous utiliserons également le code suivant placé dans un module standard :

Module Globals

Contenu masqué n°1

Classe clsClient après implémentation complète d'ICommande

Contenu masqué n°2

Nous allons maintenant implémenter ces deux interfaces pour une classe fournisseur. Ce qu'il est intéressant de remarquer ici, c'est la divergence que nous pouvons faire par rapport à `clsClient`. Nous pouvons donner une visibilité différente ou un comportement différent.

clsFournisseur

Contenu masqué n°3

Au fil de cette section, nous avons vu comment mettre en œuvre une interface au sein d'une classe.

3. Code final

Pour terminer, voici le code final du projet.

Nous implémentons l'interface `IAffichage` (dans `clsClient` et `clsFournisseur`) ainsi que l'interface `IInfoProfessionnelle` (seulement dans `clsFournisseur`) et nous testons notre programme au sein d'une procédure `Main`.

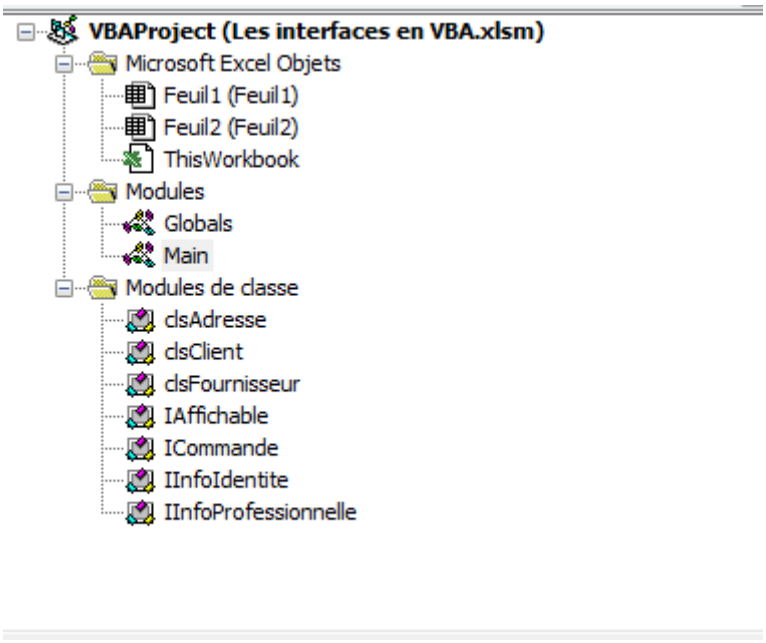

FIGURE 3.6. – Structure projet

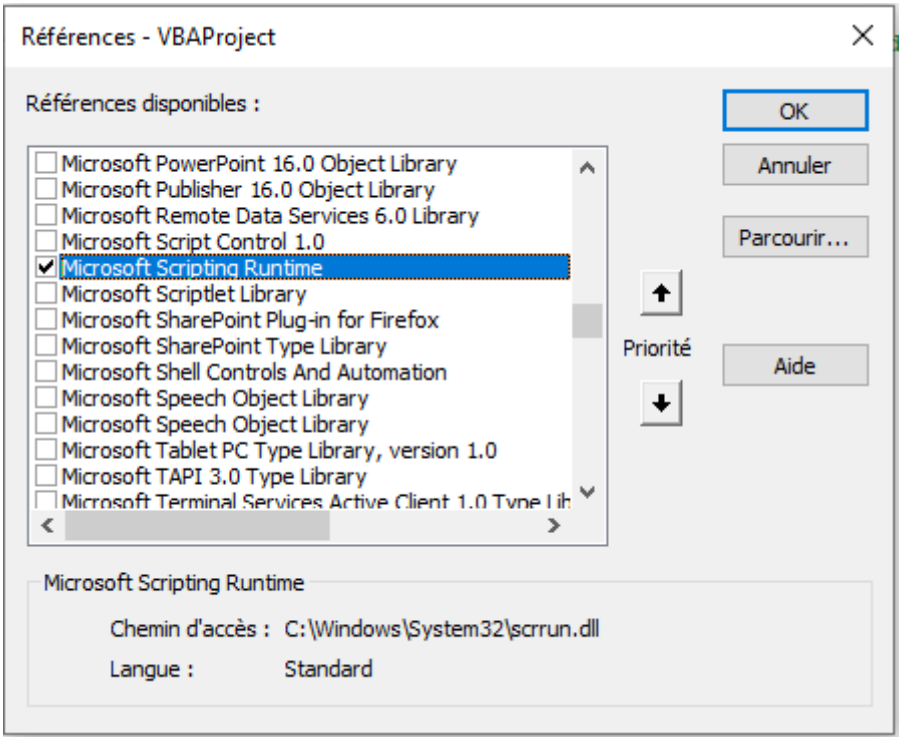

FIGURE 3.7. – Référence Microsoft Scripting Runtime

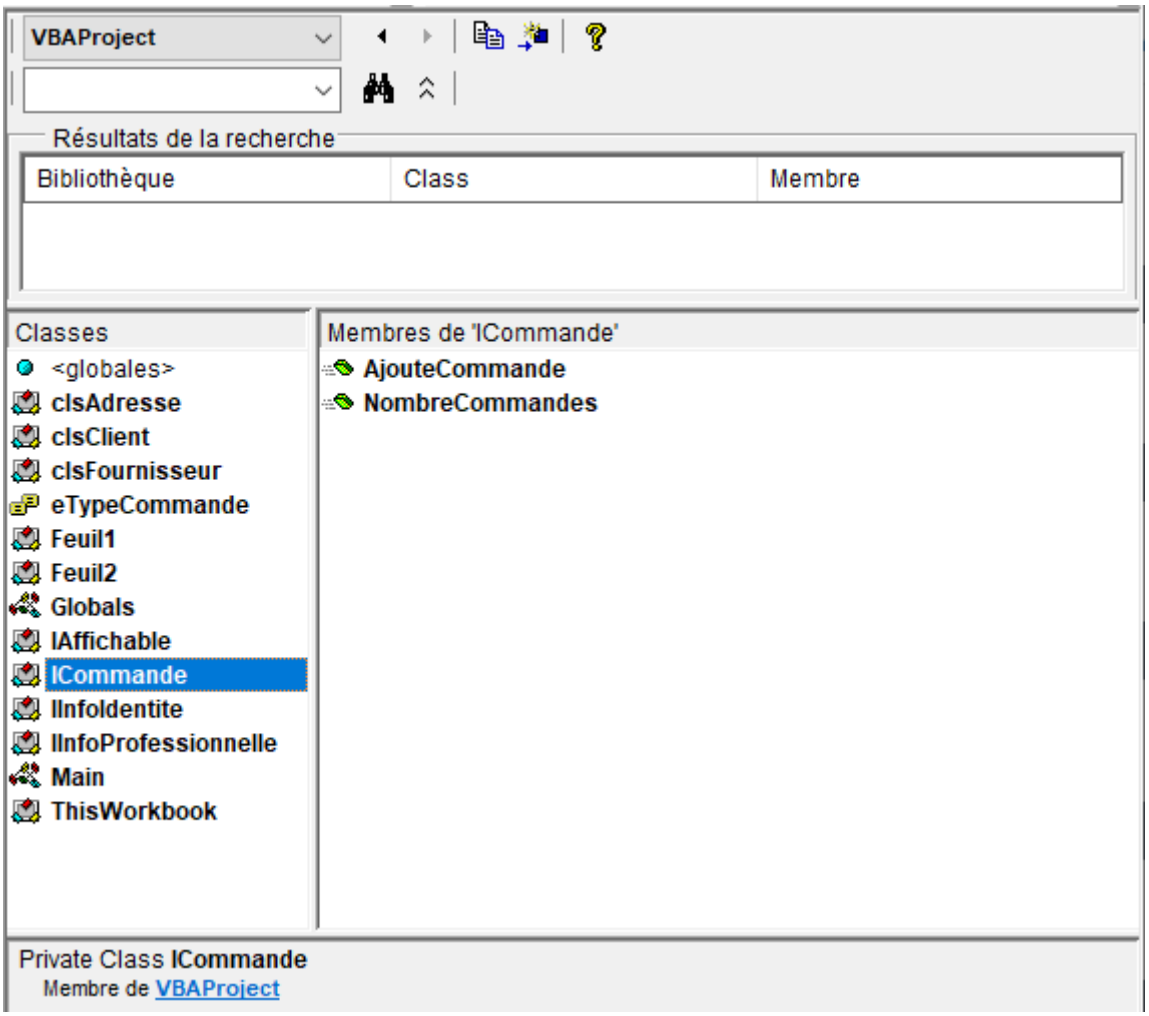

FIGURE 3.8. – Explorateur d'objets final

Module Globals

Contenu masqué n°4

Interface IAffichable

Contenu masqué n°5

Interface ICommande

Contenu masqué n°6

Interface IInfoIdentite

Contenu masqué n°7

Interface IInfoProfessionnelle

Contenu masqué n°8

Classe clsAdresse

Contenu masqué n°9

Classe clsClient

Contenu masqué n°10

Classe clsFournisseur

Contenu masqué n°11

Module Main

Contenu masqué n°12

Fenêtre d'exécution (CTRL + G)

```
_________ Client : Dupont _________
Adresse :

Commandes :
    CLIENT00000001
        1 X Casquette
        5 X Crème solaire
    CLIENT00000002
        1 X Parasole
________________________

_________ Client : Etdupont _________
Adresse :
Aucune commande enregistrée
________________________

_________ Fournisseur : global factory _________
Adresse :
N° siret : 123456784512
Qté minimale / article : 10

```

```
Commandes :
    ACHAT00000001
        10 X Casquette
        10 X Crème solaire
________________________

_________ Fournisseur : supplier _________
Adresse :
N° siret : 123456784514113
Qté minimale / article : 0

Commandes :
    ACHAT00000002
        1 X Parasole
________________________

```

Conclusion

C'est déjà la fin de ce billet.

Au fil de celui-ci, nous avons vu comment définir et utiliser des interfaces en VBA.

C'est une notion supplémentaire qu'il peut être intéressant de connaître afin de construire des projets plus faciles à faire évoluer.

À bientôt !

Quelques ressources :

— La documentation
— Cette page

Contenu masqué

Contenu masqué n°1

```vba
Option Explicit
Option Private Module ' Pour empêcher la fonction d'être
    utilisable côté Excel

Public Enum ETypeCommande
    eTypeCommandeClient
    eTypeCommandeFournisseur
End Enum
```

Contenu masqué

```
 8
 9  Public Function GenereNumCommande(ByVal eTypeCommande As
        ETypeCommande) As String
10      ' Static pour conserver les valeurs au fil des appels pour
            incrémenter
11      Static lCompteurCommandeClient As Long
12      Static lCompteurCommandeFournisseur As Long
13
14      Dim lCompteur As Long
15      Dim sDebut As String
16      Select Case eTypeCommande
17          Case eTypeCommandeClient
18              lCompteurCommandeClient = lCompteurCommandeClient + 1
19              lCompteur = lCompteurCommandeClient
20              sDebut = "CLIENT"
21          Case eTypeCommandeFournisseur
22              lCompteurCommandeFournisseur =
                    lCompteurCommandeFournisseur + 1
23              lCompteur = lCompteurCommandeFournisseur
24              sDebut = "ACHAT"
25      End Select
26
27      GenereNumCommande = sDebut & Format(lCompteur, "00000000")
28  End Function
```

Retourner au texte.

Contenu masqué n°2

```
 1  Option Explicit
 2
 3  Implements IInfoIdentite
 4  Implements ICommande
 5
 6  Private mNom As String
 7  Private mAdresse As clsAdresse
 8
 9  Private mCommandes As Collection
10
11  Private Sub Class_Initialize()
12      Set mCommandes = New Collection
13  End Sub
14
15  Private Sub Class_Terminate()
16      ' Les dictionnaires contenus ne sont pas supprimés
            explicitement ici
17      Set mCommandes = Nothing
```

Contenu masqué

```
End Sub

Public Property Get IInfoIdentite_sNom() As String
    IInfoIdentite_sNom = mNom
End Property

Public Property Let IInfoIdentite_sNom(ByVal sNom As String)
    mNom = sNom
End Property

Public Property Set IInfoIdentite_oAdresse(ByRef oAdresse As
    clsAdresse)
    Set mAdresse = oAdresse
End Property

Private Function ICommande_NombreCommandes() As Integer
    ICommande_NombreCommandes = mCommandes.Count
End Function

Public Sub ICommande_AjouteCommande(ByRef dictContenuCommande As
    Scripting.IDictionary)
    Dim dictCommande As New Scripting.Dictionary

    dictCommande.Add "Num", GenereNumCommande(eTypeCommandeClient)
    dictCommande.Add "Contenu", dictContenuCommande

    mCommandes.Add dictCommande
End Sub
```

Retourner au texte.

Contenu masqué n°3

```
Option Explicit

Implements IInfoIdentite
Implements ICommande

Private mNom As String
Private mAdresse As clsAdresse
Private mQteMinimaleParArticle As Integer ' Spécifique

Private mCommandes As Collection

Private Sub Class_Initialize()
    Set mCommandes = New Collection
End Sub
```

```
15
16 Private Sub Class_Terminate()
17     ' Les dictionnaires contenus ne sont pas supprimés
         explicitement ici
18     Set mCommandes = Nothing
19 End Sub
20
21 Public Property Let QteMinimaleParArticle(ByVal
       iQteMinimaleParArticle As Integer)
22     ' Spécifique
23     mQteMinimaleParArticle = iQteMinimaleParArticle
24 End Property
25
26 Private Property Get IInfoIdentite_sNom() As String
27     ' Spécifique visibilité
28     IInfoIdentite_sNom = mNom
29 End Property
30
31 Public Property Let IInfoIdentite_sNom(ByVal sNom As String)
32     ' Spécifique pour empêcher la modification du nom une fois
         défini
33     If mNom = "" Then
34         mNom = sNom
35     End If
36 End Property
37
38 Public Property Set IInfoIdentite_oAdresse(ByRef oAdresse As
       clsAdresse)
39     Set mAdresse = oAdresse
40 End Property
41
42 Private Function ICommande_NombreCommandes() As Integer
43     ICommande_NombreCommandes = mCommandes.Count
44 End Function
45
46 Public Sub ICommande_AjouteCommande(dictContenuCommande As
       Scripting.IDictionary)
47     Dim vPrix As Variant
48
49     ' Spécifique où l'on s'assure que la quantité commandée au
         fournisseur est
50     ' au moins égale à la quantité minimale lorsqu'une quantité
         minimale est paramétrée
51     Dim vReference As Variant
52     For Each vReference In dictContenuCommande.Keys
53         If dictContenuCommande(vReference) <
               mQteMinimaleParArticle And mQteMinimaleParArticle > 0
               Then
54             dictContenuCommande(vReference) =
                   mQteMinimaleParArticle
```

```
55          End If
56      Next vReference
57
58      Dim dictCommande As New Scripting.Dictionary
59      dictCommande.Add "Num",
            GenereNumCommande(eTypeCommandeFournisseur)
60      dictCommande.Add "Contenu", dictContenuCommande
61
62      mCommandes.Add dictCommande
63  End Sub
```

Retourner au texte.

Contenu masqué n°4

Contenu masqué

```
Option Explicit
Option Private Module ' Pour empêcher la fonction d'être utilisable côté Excel

Public Enum ETypeCommande
    eTypeCommandeClient
    eTypeCommandeFournisseur
End Enum

Public Function GenereNumCommande(ByVal eTypeCommande As ETypeCommande) As String
    ' Static pour conserver les valeurs au fil des appels pour incrémenter
    Static lCompteurCommandeClient As Long
    Static lCompteurCommandeFournisseur As Long

    Dim lCompteur As Long
    Dim sDebut As String
    Select Case eTypeCommande
        Case eTypeCommandeClient
            lCompteurCommandeClient = lCompteurCommandeClient + 1
            lCompteur = lCompteurCommandeClient
            sDebut = "CLIENT"
        Case eTypeCommandeFournisseur
            lCompteurCommandeFournisseur = lCompteurCommandeFournisseur + 1
            lCompteur = lCompteurCommandeFournisseur
            sDebut = "ACHAT"
    End Select

    GenereNumCommande = sDebut & Format(lCompteur, "00000000")
End Function
```

Retourner au texte.

Contenu masqué n°5

```
Option Explicit

Public Function ToString() As String ' Méthode
End Function

Public Sub Affiche() ' Méthode
End Sub
```

Retourner au texte.

Contenu masqué n°6

```
1  Option Explicit
2
3  Public Function NombreCommandes() As Integer ' Méthode
4  End Function
5
6  Public Sub AjouteCommande(ByRef dictCommande As
       Scripting.Dictionary)  ' Méthode
7  End Sub
```

Retourner au texte.

Contenu masqué n°7

```
1  Option Explicit
2
3  Public Property Let sNom(ByVal sNom As String)  ' Propriété
       procédure property
4  End Property
5
6  Public Property Get sNom() As String ' Propriété procédure property
7  End Property
8
9  Public Property Set oAdresse(ByRef oAdresse As clsAdresse) '
       Propriété procédure property
10 End Property
```

Retourner au texte.

Contenu masqué n°8

```
1  Option Explicit
2
3  Public sSiret As String ' Équivalent de Propriété procédure
       property Let + Get
```

Retourner au texte.

Contenu masqué n°9

```
Option Explicit

' Rien
```

Retourner au texte.

Contenu masqué n°10

```
Option Explicit

Implements IInfoIdentite
Implements ICommande
Implements IAffichable

Private mNom As String
Private mAdresse As clsAdresse

Private mCommandes As Collection

Private Sub Class_Initialize()
    Set mCommandes = New Collection
End Sub

Private Sub Class_Terminate()
   ' Les dictionnaires contenus ne sont pas supprimés
      explicitement ici
    Set mCommandes = Nothing
End Sub

Public Property Get IInfoIdentite_sNom() As String
    IInfoIdentite_sNom = mNom
End Property

Public Property Let IInfoIdentite_sNom(ByVal sNom As String)
    mNom = sNom
End Property

Public Property Set IInfoIdentite_oAdresse(ByRef oAdresse As
    clsAdresse)
    Set mAdresse = oAdresse
End Property

Private Function ICommande_NombreCommandes() As Integer
```

Contenu masqué

```
    ICommande_NombreCommandes = mCommandes.Count
End Function

Public Sub ICommande_AjouteCommande(ByRef dictContenuCommande As
    Scripting.IDictionary)
    Dim dictCommande As New Scripting.Dictionary

    dictCommande.Add "Num", GenereNumCommande(eTypeCommandeClient)
    dictCommande.Add "Contenu", dictContenuCommande

    mCommandes.Add dictCommande
End Sub

Private Function IAffichable_ToString() As String
    Dim sResult As String

    sResult = "_________ Client : " & mNom & " _________" &
        vbNewLine
    sResult = sResult & "Adresse :" & vbNewLine

    If ICommande_NombreCommandes Then
        sResult = sResult & vbNewLine
        sResult = sResult & "Commandes :" & vbNewLine

        Dim vCommande As Variant
        Dim vReference As Variant
        For Each vCommande In mCommandes
            sResult = sResult & "    " & vCommande("Num") &
                vbNewLine
            For Each vReference In vCommande("Contenu")
                sResult = sResult & "        " &
                    vCommande("Contenu")(vReference) & " X " &
                    vReference & vbNewLine
            Next vReference

        Next vCommande
    Else
        sResult = sResult & "Aucune commande enregistrée" &
            vbNewLine
    End If

    sResult = sResult & "_________________________" & vbNewLine

    IAffichable_ToString = sResult
End Function

Public Sub IAffichable_Affiche()
    Debug.Print IAffichable_ToString()
End Sub
```

Retourner au texte.

Contenu masqué n°11

```
Option Explicit

Implements IInfoIdentite
Implements IInfoProfessionnelle
Implements ICommande
Implements IAffichable

Private mNom As String
Private mAdresse As clsAdresse
Private mSiret As String 'Spécifique
Private mQteMinimaleParArticle As Integer ' Spécifique

Private mCommandes As Collection

Private Sub Class_Initialize()
    Set mCommandes = New Collection
End Sub

Private Sub Class_Terminate()
   ' Les dictionnaires contenus ne sont pas supprimés
      explicitement ici
    Set mCommandes = Nothing
End Sub

Public Property Let QteMinimaleParArticle(ByVal
    iQteMinimaleParArticle As Integer)
    ' Spécifique
    mQteMinimaleParArticle = iQteMinimaleParArticle
End Property

Private Property Get IInfoIdentite_sNom() As String
    ' Spécifique visibilité
    IInfoIdentite_sNom = mNom
End Property

Public Property Let IInfoIdentite_sNom(ByVal sNom As String)
   ' Spécifique pour empêcher la modification du nom une fois
      défini
    If mNom = "" Then
        mNom = sNom
    End If
End Property

```

Contenu masqué

```
Public Property Set IInfoIdentite_oAdresse(ByRef oAdresse As clsAdresse)
    Set mAdresse = oAdresse
End Property

Public Property Let IInfoProfessionnelle_sSiret(ByVal sSiret As String)
    mSiret = sSiret
End Property

Public Property Get IInfoProfessionnelle_sSiret() As String
    IInfoProfessionnelle_sSiret = mSiret
End Property

Private Function ICommande_NombreCommandes() As Integer
    ICommande_NombreCommandes = mCommandes.Count
End Function

Public Sub ICommande_AjouteCommande(dictContenuCommande As Scripting.IDictionary)
    Dim vPrix As Variant

    ' Spécifique où l'on s'assure que la quantité commandée au fournisseur est la
    ' au moins égale à la quantité minimale lorsqu'une quantité minimale est paramétrée
    Dim vReference As Variant
    For Each vReference In dictContenuCommande.Keys
        If dictContenuCommande(vReference) < mQteMinimaleParArticle And mQteMinimaleParArticle > 0 Then
            dictContenuCommande(vReference) = mQteMinimaleParArticle
        End If
    Next vReference

    Dim dictCommande As New Scripting.Dictionary
    dictCommande.Add "Num", GenereNumCommande(eTypeCommandeFournisseur)
    dictCommande.Add "Contenu", dictContenuCommande

    mCommandes.Add dictCommande
End Sub

Private Function IAffichable_ToString() As String
    Dim sResult As String

    sResult = "_________ Fournisseur : " & mNom & " _________" & vbNewLine
    sResult = sResult & "Adresse :" & vbNewLine
```

```
81      sResult = sResult & "N° siret : " & mSiret & vbNewLine
82      sResult = sResult & "Qté minimale / article : " &
            mQteMinimaleParArticle & vbNewLine
83
84      If ICommande_NombreCommandes Then
85          sResult = sResult & vbNewLine
86          sResult = sResult & "Commandes :" & vbNewLine
87
88          Dim vCommande As Variant
89          Dim vReference As Variant
90          For Each vCommande In mCommandes
91              sResult = sResult & "    " & vCommande("Num") &
                    vbNewLine
92              For Each vReference In vCommande("Contenu")
93                  sResult = sResult & "        " &
                        vCommande("Contenu")(vReference) & " X " &
                        vReference & vbNewLine
94              Next vReference
95
96          Next vCommande
97      Else
98          sResult = sResult & "Aucune commande enregistrée" &
                vbNewLine
99      End If
100
101     sResult = sResult & "_________________________" & vbNewLine
102
103     IAffichable_ToString = sResult
104 End Function
105
106 Public Sub IAffichable_Affiche()
107     Debug.Print IAffichable_ToString()
108 End Sub
```

Retourner au texte.

Contenu masqué n°12

```
1  Option Explicit
2
3  Private Sub Main()
4      Dim cEntites As New Collection
5
6      ' Ajout de clients / fournisseurs
7      Dim oClient1 As New clsClient
8      oClient1.IInfoIdentite_sNom = "Dupont"
9      Set oClient1.IInfoIdentite_oAdresse = New clsAdresse
```

Contenu masqué

```
    ' Debug.Print (oClient1.IInfoIdentite_sNom) ' Dupont
    cEntites.Add oClient1

    Dim oClient2 As New clsClient
    oClient2.IInfoIdentite_sNom = "Etdupont"
    Set oClient2.IInfoIdentite_oAdresse = New clsAdresse
    cEntites.Add oClient2

    Dim oFournisseur1 As New clsFournisseur
    oFournisseur1.IInfoIdentite_sNom = "global factory"
    ' Test spécifique modification : le nom reste global factory
    oFournisseur1.IInfoIdentite_sNom = "global factory2"
    ' Debug.Print (oFournisseur1.IInfoIdentite_sNom) ' Erreur car
        inaccessible

    Set oFournisseur1.IInfoIdentite_oAdresse = New clsAdresse
    oFournisseur1.IInfoProfessionnelle_sSiret = "123456784512"
    oFournisseur1.QteMinimaleParArticle = 10
    cEntites.Add oFournisseur1

    Dim oFournisseur2 As New clsFournisseur
    oFournisseur2.IInfoIdentite_sNom = "supplier"
    Set oFournisseur2.IInfoIdentite_oAdresse = New clsAdresse
    oFournisseur2.IInfoProfessionnelle_sSiret = "123456784514113"
    cEntites.Add oFournisseur2

    ' Ajout de commandes
    Dim dictCommandeClient1_1 As New Scripting.Dictionary
    dictCommandeClient1_1.Add "Casquette", 1
    dictCommandeClient1_1.Add "Crème solaire", 5
    oClient1.ICommande_AjouteCommande dictCommandeClient1_1

    Dim dictCommandeClient1_2 As New Scripting.Dictionary
    dictCommandeClient1_2.Add "Parasole", 1
    oClient1.ICommande_AjouteCommande dictCommandeClient1_2

    Dim dictCommandeFournisseur1_1 As New Scripting.Dictionary
    dictCommandeFournisseur1_1.Add "Casquette", 1
    dictCommandeFournisseur1_1.Add "Crème solaire", 5
    oFournisseur1.ICommande_AjouteCommande
        dictCommandeFournisseur1_1

    Dim dictCommandeFournisseur2_1 As New Scripting.Dictionary
    dictCommandeFournisseur2_1.Add "Parasole", 1
    oFournisseur2.ICommande_AjouteCommande
        dictCommandeFournisseur2_1

    ' Affichage en tirant parti du polymorphisme
    Dim vEntite As Variant
    For Each vEntite In cEntites
```

Contenu masqué

```
57          vEntite.IAffichable_Affiche
58      Next vEntite
59  End Sub
```

Retourner au texte.

Liste des abréviations

POO Programmation Orientée Objet. 1

VBA Visual Basic for Applications. 1, 2