

Queste de savoir

Les interfaces en VBA

jeudi 18 juillet 2024

Table des matières

	Introduction	1
1.	Définition	2
2.	Utilisation	4
3.	Code final	6
	Conclusion	10
	Contenu masqué	10

Introduction

En conclusion du billet sur les [classes en VBA](#) , nous avons dit que le système de **POO** est moins poussé en **VBA** que dans d'autres langages. Nous pouvons citer l'absence d'héritage par exemple.

Néanmoins, il est possible de travailler avec des interfaces. 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 exemple, les feuilles **Worksheet** (feuille de calcul) et **Chart** (feuille de graphique) sont des objets de types différents, mais qui ont des éléments similaires (méthode **Activate**, propriété **Name**, etc.). Suivant cet exemple, il serait envisageable d'avoir une interface indiquant que ces deux classes doivent implémenter une méthode **Activate** ou encore une propriété **Name**.

Ce qu'il est important de noter ici, c'est que cette interface indique qu'elles doivent implémenter ces membres, mais pas *comment*, laissant la liberté d'implémentation aux classes. Les deux classes pourraient donc implémenter une méthode **Activate** avec un comportement différent (soyons imaginatifs 🍌) que ça ne dérangerait pas, car le contrat serait respecté !

Autre 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, etc.

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

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`.

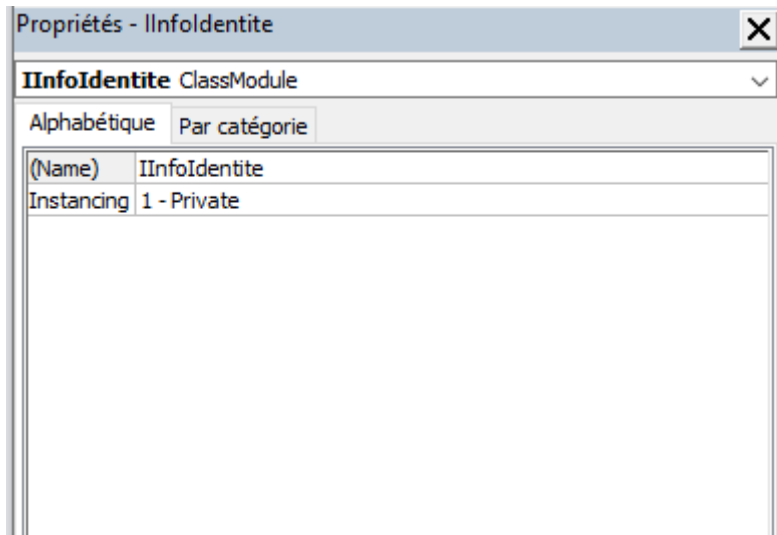


FIGURE 1.1. – Propriétés interface IInfoIdentite

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`.

1. Définition

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*.

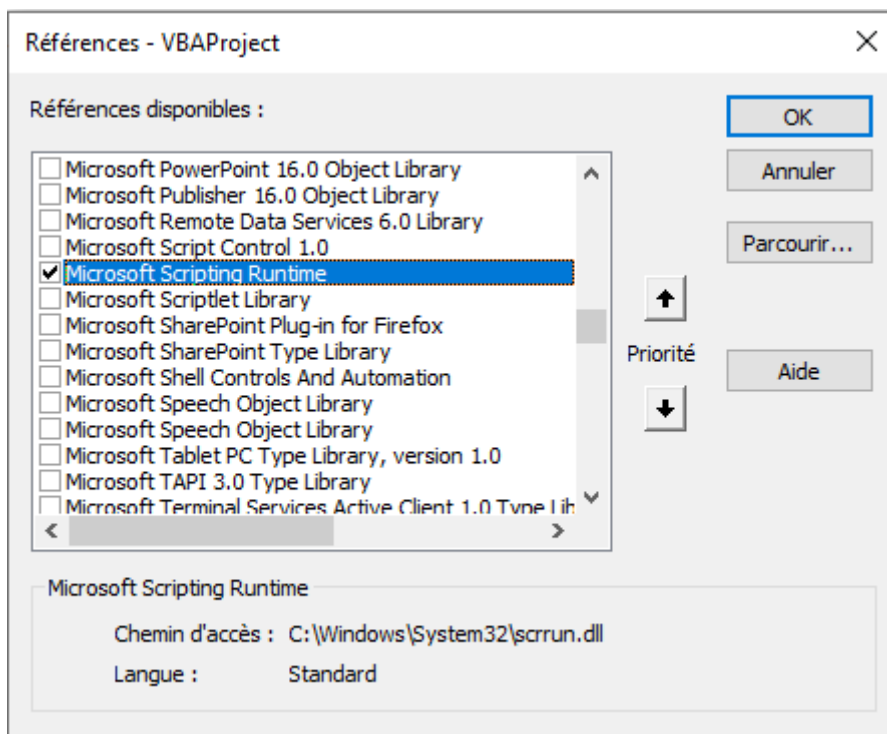


FIGURE 1.2. – Référence Microsoft Scripting Runtime

Interface ICommande

2. Utilisation

```
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.

2. Utilisation

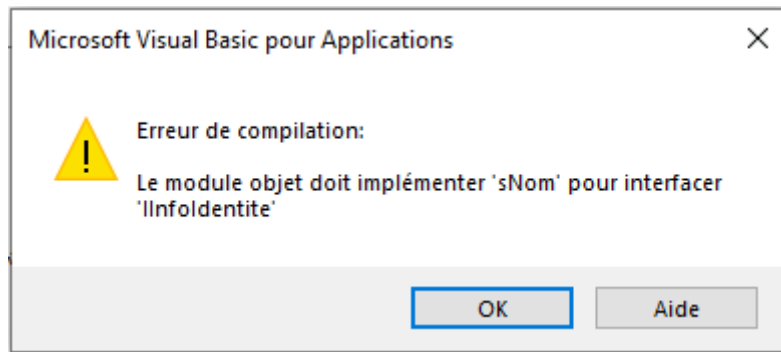


FIGURE 2.3. – Erreur compilation lorsque le contrat n'est pas respecté

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

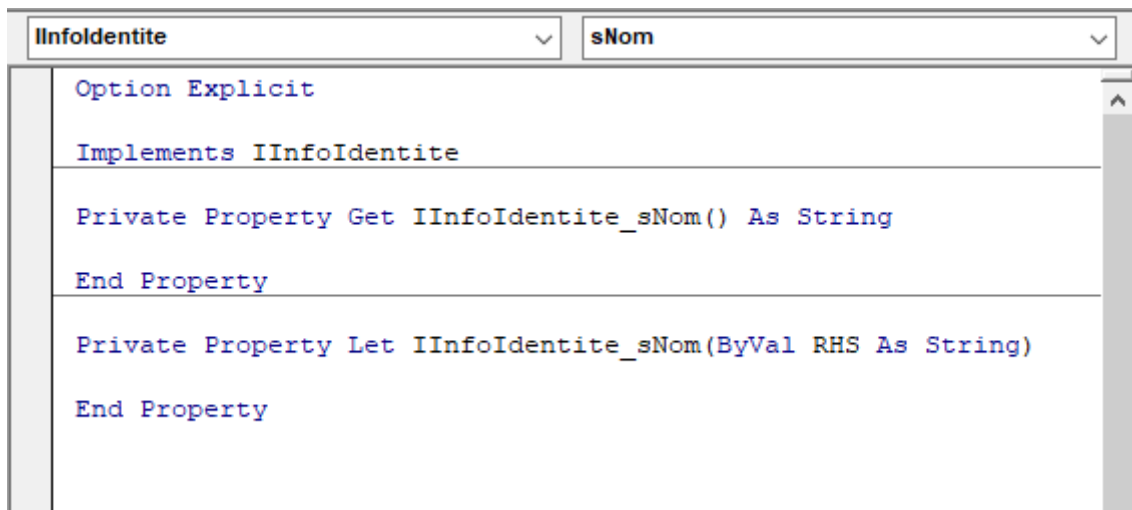


FIGURE 2.4. – Zone d'objet et zone de procédures/événements



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
```

3. Code final

```
10 End Property
11
12 Public Property Let IInfoIdentite_sNom(ByVal sNom As String)
13     mNom = sNom
14 End Property
15
16 Public Property Set IInfoIdentite_oAdresse(ByRef oAdresse As
17     clsAdresse)
18     Set mAdresse = oAdresse
19 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`.

3. Code final

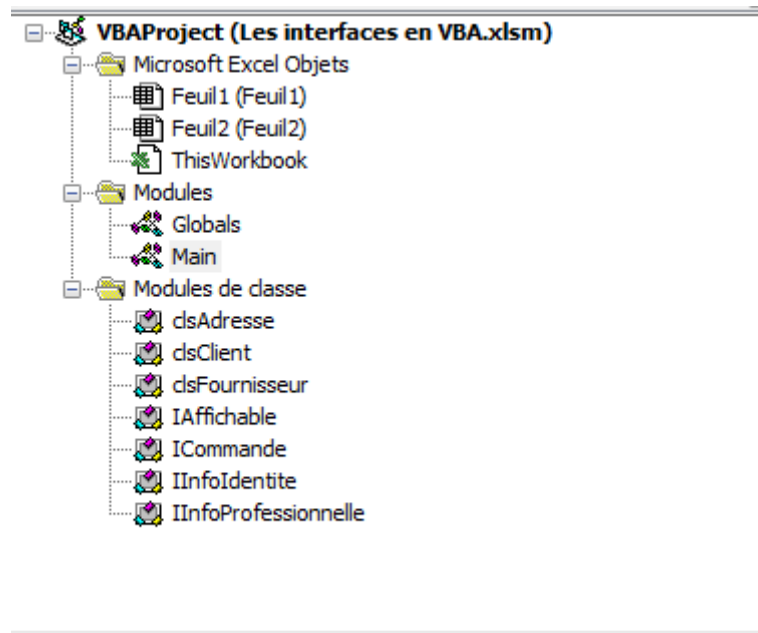


FIGURE 3.5. – Structure projet

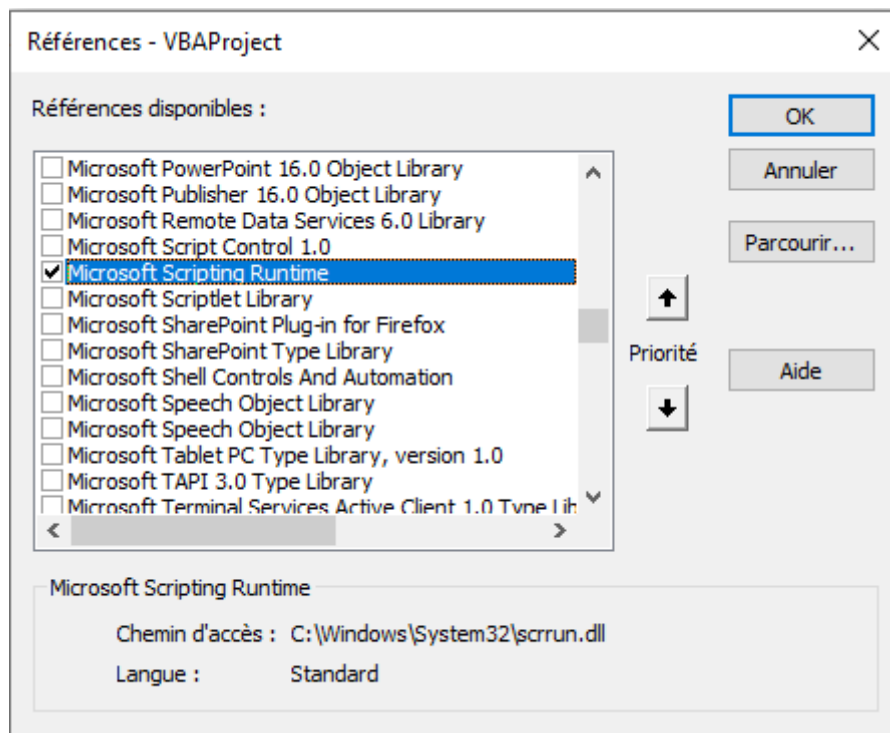


FIGURE 3.6. – Référence Microsoft Scripting Runtime

3. Code final

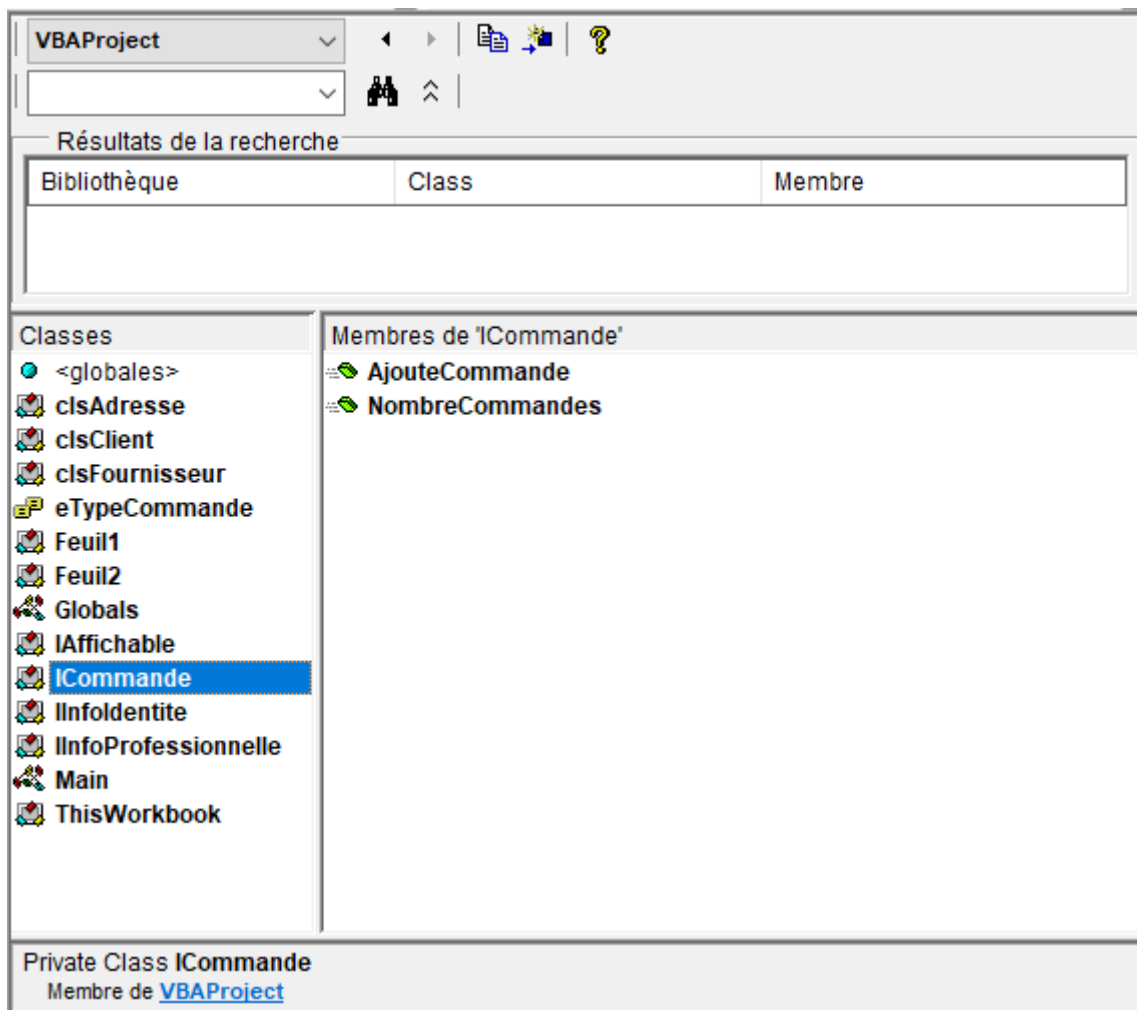


FIGURE 3.7. – Explorateur d'objets final

Module Globals

☉ Contenu masqué n°4

Interface IAffichable

☉ Contenu masqué n°5

Interface ICommande

☉ Contenu masqué n°6

Interface IInfoIdentite

☉ Contenu masqué n°7

3. Code final

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)

```
1  _____ Client : Dupont _____
2  Adresse :
3
4  Commandes :
5      CLIENT00000001
6          1 X Casquette
7          5 X Crème solaire
8      CLIENT00000002
9          1 X Parasole
10 _____
11
12 _____ Client : Etdupont _____
13 Adresse :
14 Aucune commande enregistrée
15 _____
16
17 _____ Fournisseur : global factory _____
18 Adresse :
19 N° siret : 123456784512
20 Qté minimale / article : 10
21
```

Conclusion

```
22 Commandes :
23     ACHAT000000001
24         10 X Casquette
25         10 X Crème solaire
26 -----
27
28 _____ Fournisseur : supplier _____
29 Adresse :
30 N° siret : 123456784514113
31 Qté minimale / article : 0
32
33 Commandes :
34     ACHAT000000002
35         1 X Parasole
36 -----
37
```

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

```
1 Option Explicit
2 Option Private Module ' Pour empêcher la fonction d'être
   utilisable côté Excel
3
4 Public Enum ETypeCommande
5     eTypeCommandeClient
6     eTypeCommandeFournisseur
7 End Enum
```

```
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 =
23         lCompteurCommandeFournisseur + 1
24       lCompteur = lCompteurCommandeFournisseur
25       sDebut = "ACHAT"
26   End Select
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
```

```
18 End Sub
19
20 Public Property Get IInfoIdentite_sNom() As String
21     IInfoIdentite_sNom = mNom
22 End Property
23
24 Public Property Let IInfoIdentite_sNom(ByVal sNom As String)
25     mNom = sNom
26 End Property
27
28 Public Property Set IInfoIdentite_oAdresse(ByRef oAdresse As
    clsAdresse)
29     Set mAdresse = oAdresse
30 End Property
31
32 Private Function ICommande_NombreCommandes() As Integer
33     ICommande_NombreCommandes = mCommandes.Count
34 End Function
35
36 Public Sub ICommande_AjouteCommande(ByRef dictContenuCommande As
    Scripting.IDictionary)
37     Dim dictCommande As New Scripting.Dictionary
38
39     dictCommande.Add "Num", GenereNumCommande(eTypeCommandeClient)
40     dictCommande.Add "Contenu", dictContenuCommande
41
42     mCommandes.Add dictCommande
43 End Sub
```

[Retourner au texte.](#)

Contenu masqué n°3

```
1 Option Explicit
2
3 Implements IInfoIdentite
4 Implements ICommande
5
6 Private mNom As String
7 Private mAdresse As clsAdresse
8 Private mQteMinimaleParArticle As Integer ' Spécifique
9
10 Private mCommandes As Collection
11
12 Private Sub Class_Initialize()
13     Set mCommandes = New Collection
14 End Sub
```

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

Contenu masqué

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

[Retourner au texte.](#)

Contenu masqué n°4


```
1 Option Explicit
2 Option Private Module ' Pour empêcher la fonction d'être
   utilisable côté Excel
3
4 Public Enum ETypeCommande
5     eTypeCommandeClient
6     eTypeCommandeFournisseur
7 End Enum
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 =
23                lCompteurCommandeFournisseur + 1
24            lCompteur = lCompteurCommandeFournisseur
25            sDebut = "ACHAT"
26    End Select
27    GenereNumCommande = sDebut & Format(lCompteur, "00000000")
28 End Function
```

[Retourner au texte.](#)

Contenu masqué n°5

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

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

[Retourner au texte.](#)

Contenu masqué n°10

```
1 Option Explicit
2
3 Implements IInfoIdentite
4 Implements ICommande
5 Implements IAffichable
6
7 Private mNom As String
8 Private mAdresse As clsAdresse
9
10 Private mCommandes As Collection
11
12 Private Sub Class_Initialize()
13     Set mCommandes = New Collection
14 End Sub
15
16 Private Sub Class_Terminate()
17     ' Les dictionnaires contenus ne sont pas supprimés
18     ' explicitement ici
19     Set mCommandes = Nothing
20 End Sub
21
22 Public Property Get IInfoIdentite_sNom() As String
23     IInfoIdentite_sNom = mNom
24 End Property
25
26 Public Property Let IInfoIdentite_sNom(ByVal sNom As String)
27     mNom = sNom
28 End Property
29
30 Public Property Set IInfoIdentite_oAdresse(ByRef oAdresse As
31     clsAdresse)
32     Set mAdresse = oAdresse
33 End Property
34
35 Private Function ICommande_NombreCommandes() As Integer
```

```

34     ICommande_NombreCommandes = mCommandes.Count
35 End Function
36
37 Public Sub ICommande_AjouteCommande(ByRef dictContenuCommande As
    Scripting.IDictionary)
38     Dim dictCommande As New Scripting.Dictionary
39
40     dictCommande.Add "Num", GenereNumCommande(eTypeCommandeClient)
41     dictCommande.Add "Contenu", dictContenuCommande
42
43     mCommandes.Add dictCommande
44 End Sub
45
46 Private Function IAffichable_ToString() As String
47     Dim sResult As String
48
49     sResult = "_____ Client : " & mNom & " _____" &
        vbCrLf
50     sResult = sResult & "Adresse :" & vbCrLf
51
52     If ICommande_NombreCommandes Then
53         sResult = sResult & vbCrLf
54         sResult = sResult & "Commandes :" & vbCrLf
55
56         Dim vCommande As Variant
57         Dim vReference As Variant
58         For Each vCommande In mCommandes
59             sResult = sResult & "   " & vCommande("Num") &
                vbCrLf
60             For Each vReference In vCommande("Contenu")
61                 sResult = sResult & "   " &
                    vCommande("Contenu")(vReference) & " X " &
                    vReference & vbCrLf
62             Next vReference
63
64         Next vCommande
65     Else
66         sResult = sResult & "Aucune commande enregistrée" &
            vbCrLf
67     End If
68
69     sResult = sResult & " _____" & vbCrLf
70
71     IAffichable_ToString = sResult
72 End Function
73
74 Public Sub IAffichable_Affiche()
75     Debug.Print IAffichable_ToString()
76 End Sub

```

Contenu masqué n°11

```
1 Option Explicit
2
3 Implements IInfoIdentite
4 Implements IInfoProfessionnelle
5 Implements ICommande
6 Implements IAffichable
7
8 Private mNom As String
9 Private mAdresse As clsAdresse
10 Private mSiret As String 'Spécifique
11 Private mQteMinimaleParArticle As Integer ' Spécifique
12
13 Private mCommandes As Collection
14
15 Private Sub Class_Initialize()
16     Set mCommandes = New Collection
17 End Sub
18
19 Private Sub Class_Terminate()
20     ' Les dictionnaires contenus ne sont pas supprimés
    explicitement ici
21     Set mCommandes = Nothing
22 End Sub
23
24 Public Property Let QteMinimaleParArticle(ByVal
    iQteMinimaleParArticle As Integer)
25     ' Spécifique
26     mQteMinimaleParArticle = iQteMinimaleParArticle
27 End Property
28
29 Private Property Get IInfoIdentite_sNom() As String
30     ' Spécifique visibilité
31     IInfoIdentite_sNom = mNom
32 End Property
33
34 Public Property Let IInfoIdentite_sNom(ByVal sNom As String)
35     ' Spécifique pour empêcher la modification du nom une fois
    défini
36     If mNom = "" Then
37         mNom = sNom
38     End If
39 End Property
40
```

```

41 Public Property Set IInfoIdentite_oAdresse(ByRef oAdresse As
    clsAdresse)
42     Set mAdresse = oAdresse
43 End Property
44
45 Public Property Let IInfoProfessionnelle_sSiret(ByVal sSiret As
    String)
46     mSiret = sSiret
47 End Property
48
49 Public Property Get IInfoProfessionnelle_sSiret() As String
50     IInfoProfessionnelle_sSiret = mSiret
51 End Property
52
53 Private Function ICommande_NombreCommandes() As Integer
54     ICommande_NombreCommandes = mCommandes.Count
55 End Function
56
57 Public Sub ICommande_AjouteCommande(dictContenuCommande As
    Scripting.IDictionary)
58     Dim vPrix As Variant
59
60     ' Spécifique où l'on s'assure que la quantité commandée au
        fournisseur est la
61     ' au moins égale à la quantité minimale lorsqu'une quantité
        minimale est paramétrée
62     Dim vReference As Variant
63     For Each vReference In dictContenuCommande.Keys
64         If dictContenuCommande(vReference) <
            mQteMinimaleParArticle And mQteMinimaleParArticle > 0
65             Then
66                 dictContenuCommande(vReference) =
                    mQteMinimaleParArticle
67             End If
68         Next vReference
69
70     Dim dictCommande As New Scripting.Dictionary
71     dictCommande.Add "Num",
        GenereNumCommande(eTypeCommandeFournisseur)
72     dictCommande.Add "Contenu", dictContenuCommande
73
74     mCommandes.Add dictCommande
75 End Sub
76
77 Private Function IAffichable_ToString() As String
78     Dim sResult As String
79
80     sResult = "_____ Fournisseur : " & mNom & " _____" &
        vbNewLine
81     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
```

```
10     ' Debug.Print (oClient1.IInfoIdentite_sNom) ' Dupont
11     cEntites.Add oClient1
12
13     Dim oClient2 As New clsClient
14     oClient2.IInfoIdentite_sNom = "Etdupont"
15     Set oClient2.IInfoIdentite_oAdresse = New clsAdresse
16     cEntites.Add oClient2
17
18     Dim oFournisseur1 As New clsFournisseur
19     oFournisseur1.IInfoIdentite_sNom = "global factory"
20     ' Test spécifique modification : le nom reste global factory
21     oFournisseur1.IInfoIdentite_sNom = "global factory2"
22     ' Debug.Print (oFournisseur1.IInfoIdentite_sNom) ' Erreur car
        inaccessible
23
24     Set oFournisseur1.IInfoIdentite_oAdresse = New clsAdresse
25     oFournisseur1.IInfoProfessionnelle_sSiret = "123456784512"
26     oFournisseur1.QteMinimaleParArticle = 10
27     cEntites.Add oFournisseur1
28
29     Dim oFournisseur2 As New clsFournisseur
30     oFournisseur2.IInfoIdentite_sNom = "supplier"
31     Set oFournisseur2.IInfoIdentite_oAdresse = New clsAdresse
32     oFournisseur2.IInfoProfessionnelle_sSiret = "123456784514113"
33     cEntites.Add oFournisseur2
34
35     ' Ajout de commandes
36     Dim dictCommandeClient1_1 As New Scripting.Dictionary
37     dictCommandeClient1_1.Add "Casquette", 1
38     dictCommandeClient1_1.Add "Crème solaire", 5
39     oClient1.ICommande_AjouteCommande dictCommandeClient1_1
40
41     Dim dictCommandeClient1_2 As New Scripting.Dictionary
42     dictCommandeClient1_2.Add "Parasole", 1
43     oClient1.ICommande_AjouteCommande dictCommandeClient1_2
44
45     Dim dictCommandeFournisseur1_1 As New Scripting.Dictionary
46     dictCommandeFournisseur1_1.Add "Casquette", 1
47     dictCommandeFournisseur1_1.Add "Crème solaire", 5
48     oFournisseur1.ICommande_AjouteCommande
        dictCommandeFournisseur1_1
49
50     Dim dictCommandeFournisseur2_1 As New Scripting.Dictionary
51     dictCommandeFournisseur2_1.Add "Parasole", 1
52     oFournisseur2.ICommande_AjouteCommande
        dictCommandeFournisseur2_1
53
54     ' Affichage en tirant parti du polymorphisme
55     Dim vEntite As Variant
56     For Each vEntite In cEntites
```



```
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