Projet

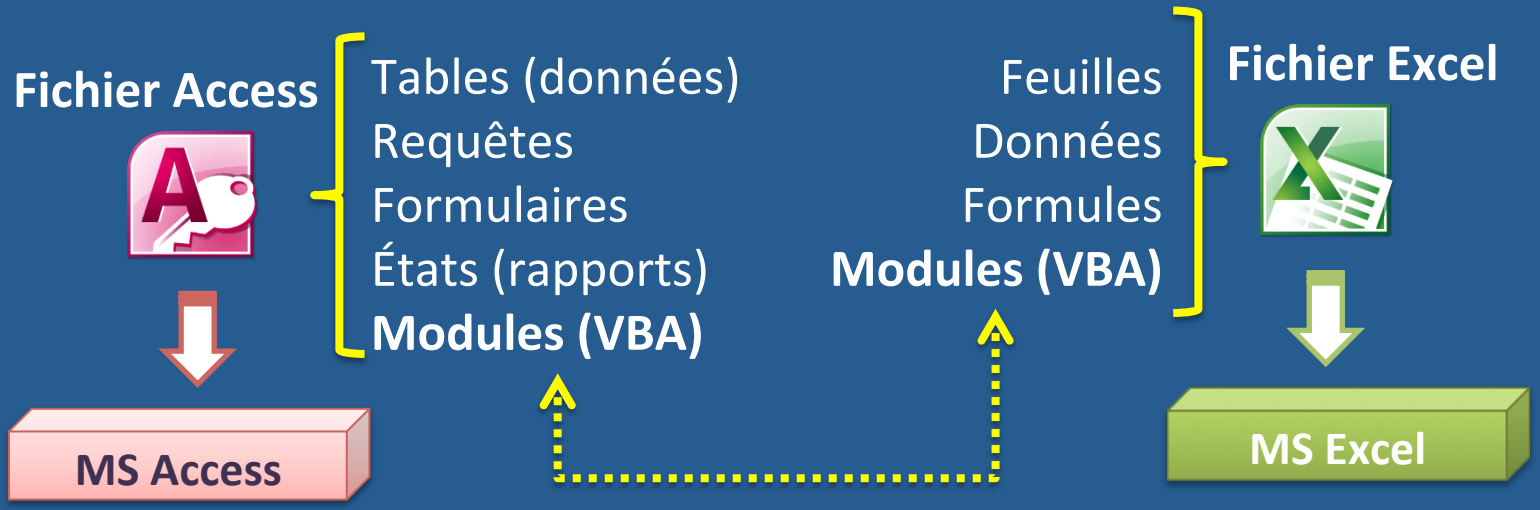
Rappel VBA

VBA : quoi & pourquoi ?

- VBA : quoi ?
  - **Langage** et **environnement** de programmation Orienté Objets
  - **Attaché aux documents** MS Office
- VBA : pourquoi ?
  - Associer **un comportement actif** à des documents Office
    - Calculs, vérifications, etc.

VBA : documents MS Office

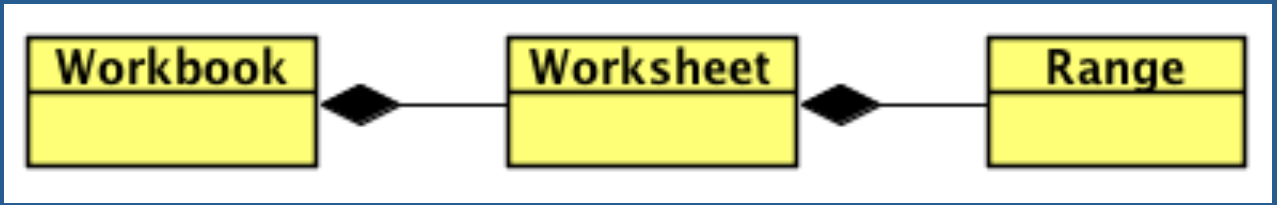
- Un document MS Office est composé des plusieurs éléments…



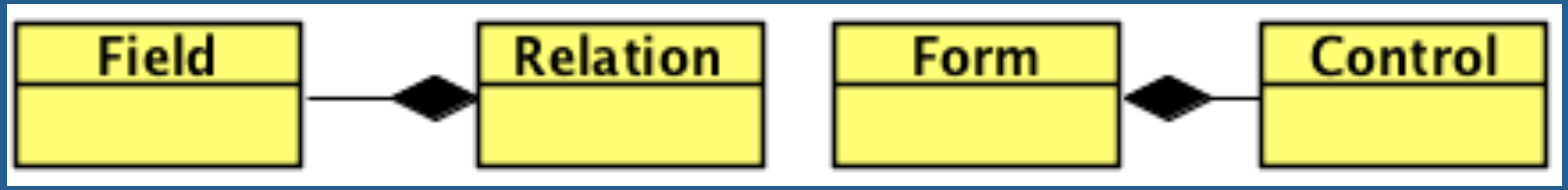
VBA : Modèle OO

- **Modèle Objets de VBA**
  - Les éléments manipulés par VBA sont des **objets**
    - **Worksheet** (feuille de calcul), **Range** (cellules), **Form** (formulaire), **Report** (état), **RecordSet** (requête)…
  - Il est possible de créer de nouvelles classes ➔ **modules de classe**

Modèle OO Excel

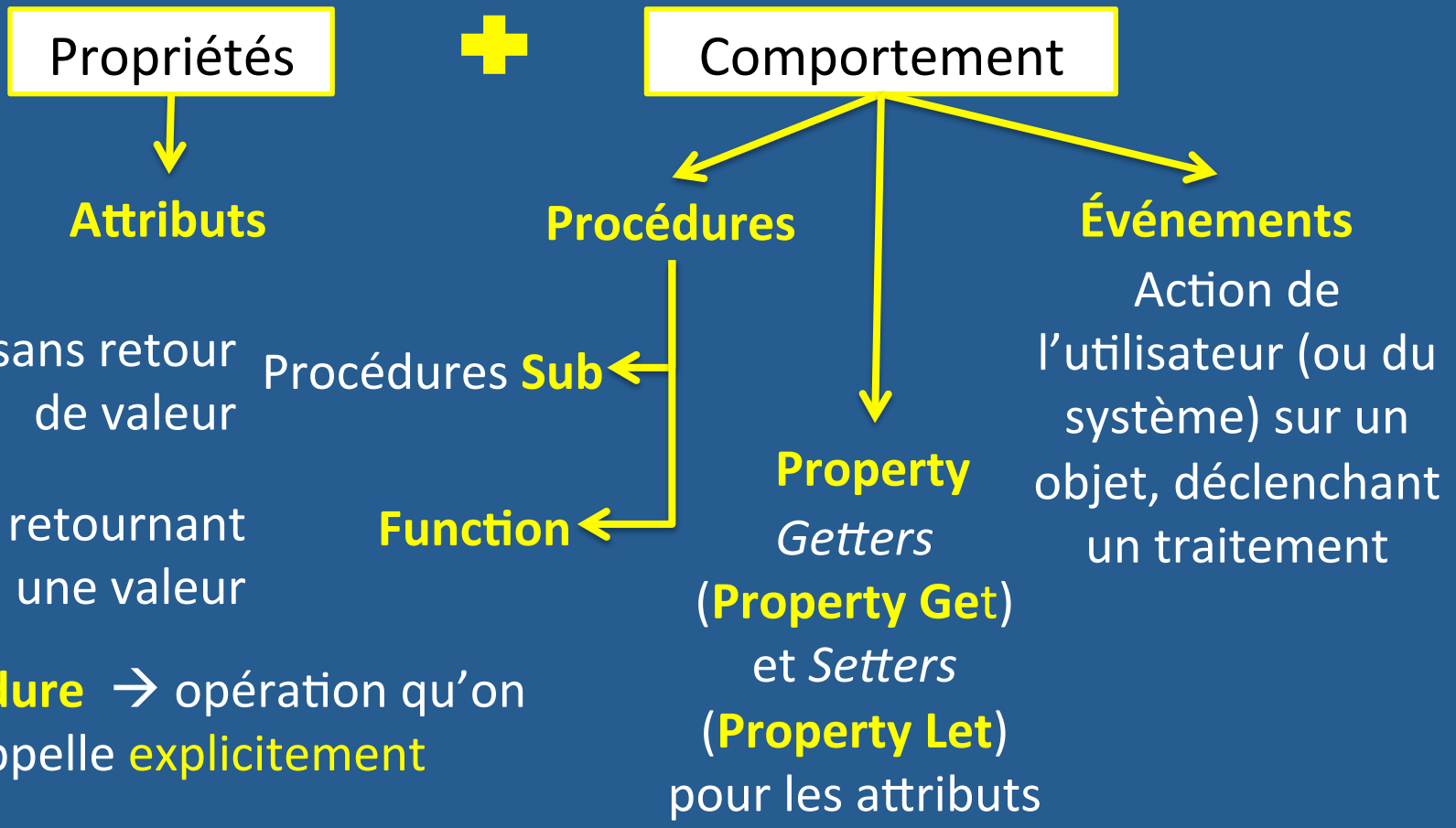


Modèle OO Access



VBA : Modèle OO

- Un objet VBA possède :



UNIVERSITÉ PARIS 1 – PANTHÉON - SORBONNE –

VBA : Modèle OO

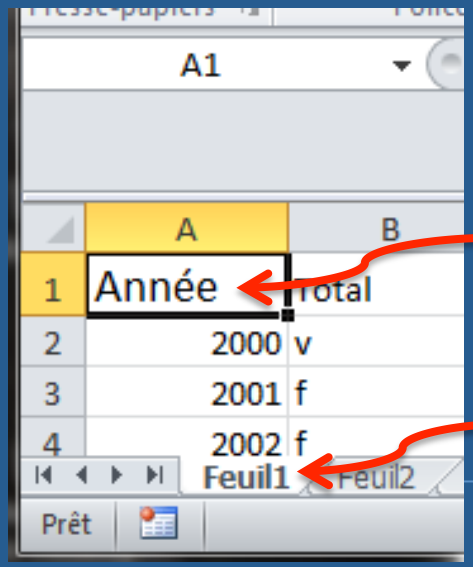
- **Collections**
  - Ensemble d'objets d'un même type
    - **Worksheets** (toutes feuilles de calcul), **Forms** (tous les formulaires), **Controls** (tous champs d'un formulaire) …

Collection`!["NomObjet"]` `Forms!["Employe"]`

Collection`("NomObjet")` `Forms("Employe")`

`Forms("Employe").Controls("Embauche")`

`Worksheets("Feuil1").Cells(1,1)`

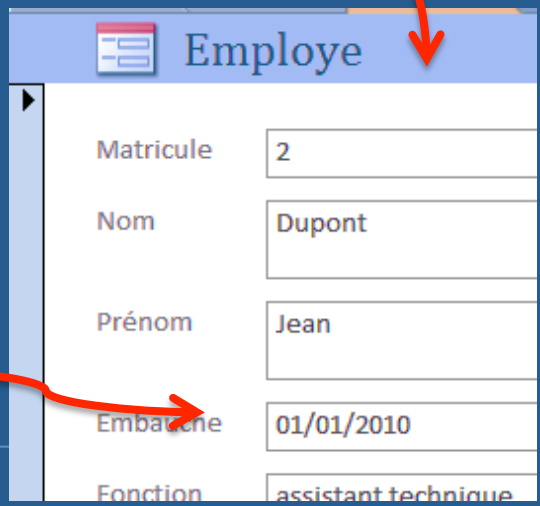


Cellule (1,1)

Feuille « Feuil1 »

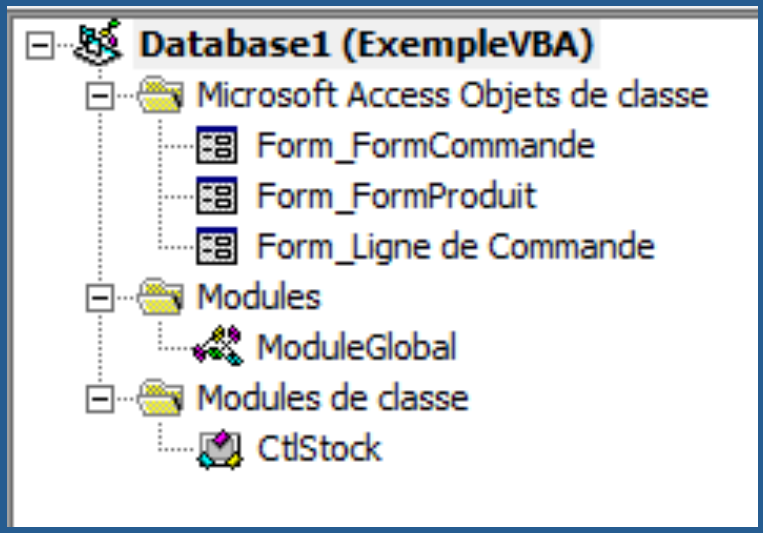
Formulaire « Employe »

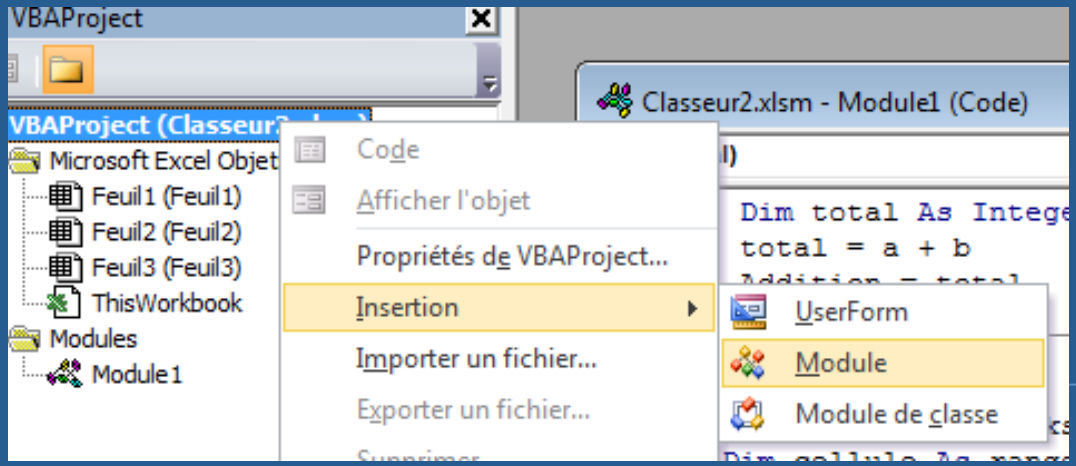
Contrôle « Embauche »



VBA : Modules

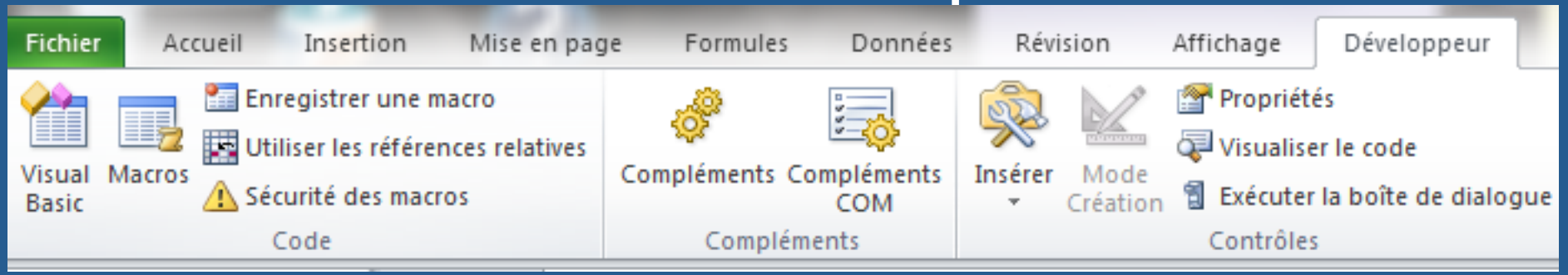
- Modules contiennent le code VBA pour un document
- Plusieurs types de modules
  - MS Objets
    - Evènements
  - Modules standards
    - Procédures utilitaires

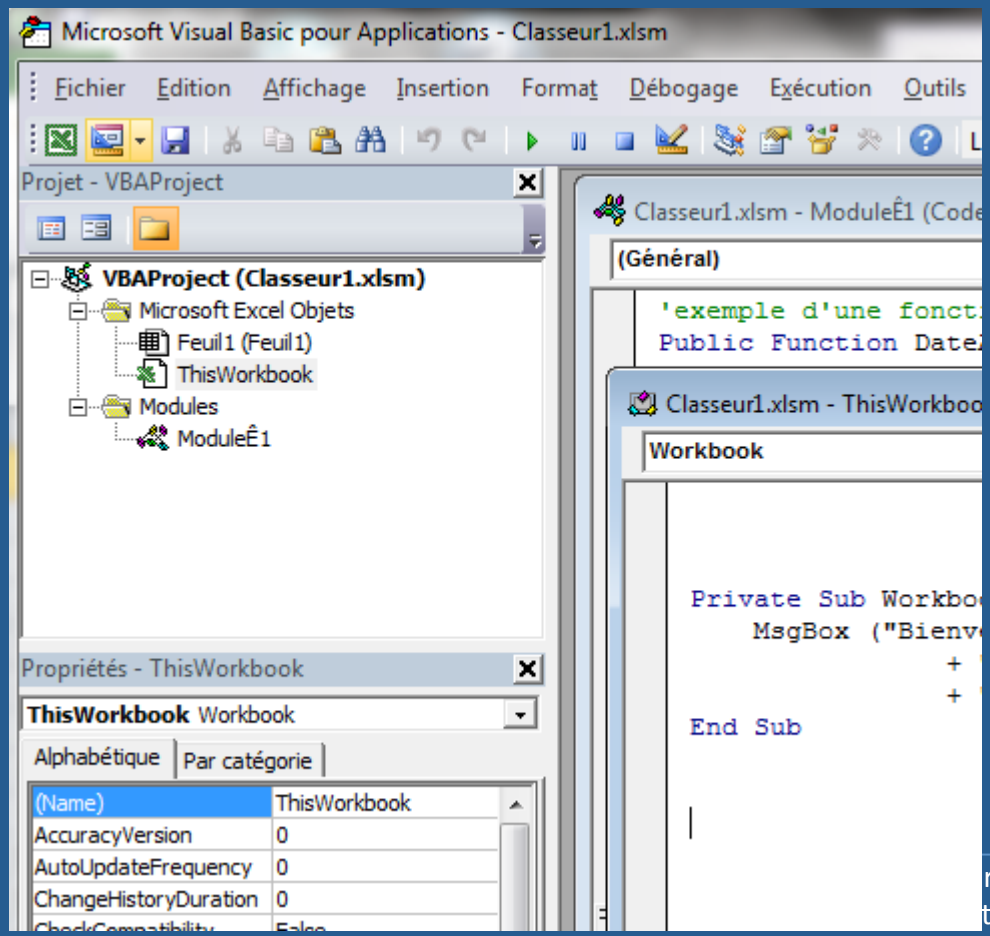




- Modules de classe
  - Classes crées par l'utilisateur

VBA : concepts de base





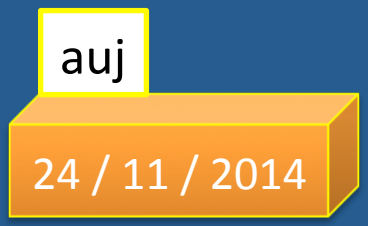
- Environnement de programmation (VBE)
  - Excel : Ruban « Développeur » (onglet à activer)
  - Access : Ruban « Outils de bases de données »

VBA : concepts de base

- **Variables**
  - Une variable est un **conteneur**, on y garde **une valeur**
  - Déclaration est fortement recommandée, mais **pas obligatoire**
    - **Dim** *variable* **AS** *Type*

Dim** auj **As Date

*auj = **Date*** *'date actuelle*



Types des données

AS **Integer** → entier
AS **Single** → numérique (*float*)
AS **Boolean** → booléen (*True* / *False*)
AS **String** → chaîne de caractères
AS **Variant** →
n'importe quel type de données
AS **Object** →
objet de n'importe quelle classe

VBA : concepts de base

- **Variables**
  - Une variable est un **conteneur**, on y garde **une valeur …**
  - … qu'on pourra utiliser plus tard !
  - Pour les objets, l'attribution se fait à l'aide de **Set**

```
Dim a As Integer
Dim b As Integer
a = 2

b = a * a
```

attribution

usage

```
Dim cellule As Range

Set  cellule = Worksheets("Feuil1").Cells(1,1)

MsgBox cellule.Valeur
```

Objet.Opération

Propriété **Valeur** de la classe **Range**

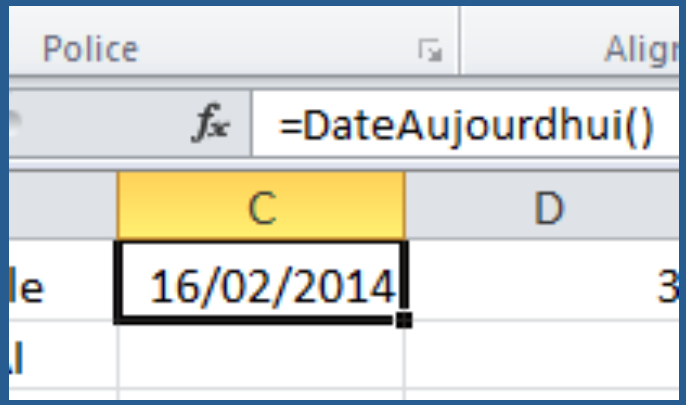
VBA : concepts de base

- **Fonctions (Function)**
  - Opération qui retourne une valeur…
  - …correspondant au nom de la fonction

Private | Public Function *MaFonction* **(***paramètres…* **) AS Type**

MaFonction = valeur_à_retourner

End Sub




```
Public Function DateAujourdhui() As Date
    Dim auj As Date
    auj = Date
    DateAujourdhui = auj
End Function
```

VBA : concepts de base

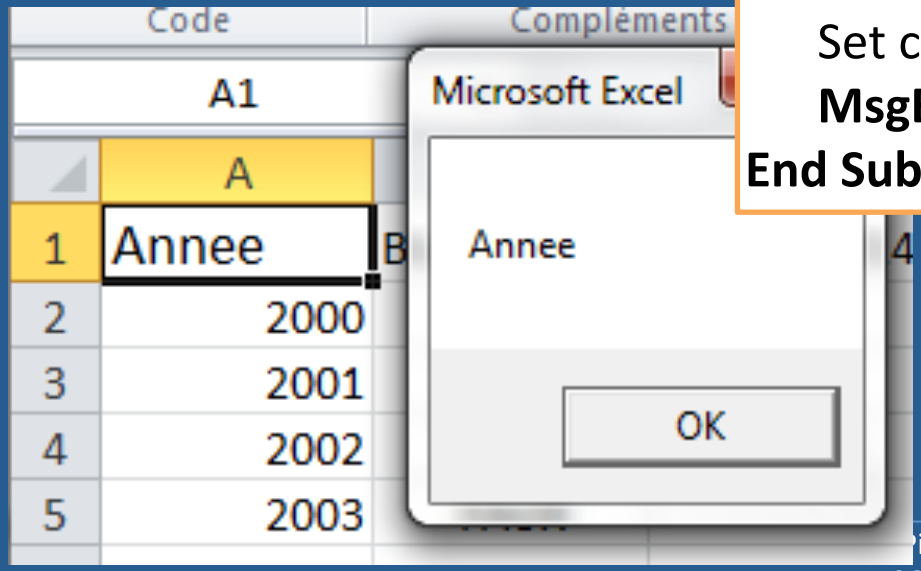
- **Procédures de type Sub**
  - Exécution d'une opération qui ne retourne aucune valeur

Private | Public Sub *NomProcédure* (*paramètre AS type* …)

. . .

End Sub

```
Sub ValeurA1()
    Dim cellule As Range
    Set cellule = Worksheets("Feuil1").Cells(1, 1)
    MsgBox (cellule.Value)
End Sub
```



UNIVERSITÉ PARIS 1 – PANTHÉON - SORBONNE –

VBA : concepts de base

- **Événements :**
  - Actions réalisées par l'utilisateur (ou le système) sur un élément (feuille, formulaire, contrôle…)
    - Exemples : ouverture d'un document, clique sur un bouton, fermeture de l'application, ouverture d'un formulaire…
  - Le type d'événement varie en fonction de l'élément qui reçoit l'action

Contrôle
- **Click** : lorsqu'on lui clique dessus

Sur une fenêtre (formulaire ou état)
- **Open** : à l'ouverture, avant d'afficher le 1er registre
- **Load** : lorsque le 1er registre est affiché
- **Close** : à la fermeture

Données (formulaire / contrôle associé à une source de données RecordSource /ControlSource)
- **BeforeInsert / AfterInsert**: avant/après insertion nouveau registre
- **BeforeUpdate/AfterUpdate**: avant/après mise à jour registre
- **Change** : lors de la modification d'un contrôle

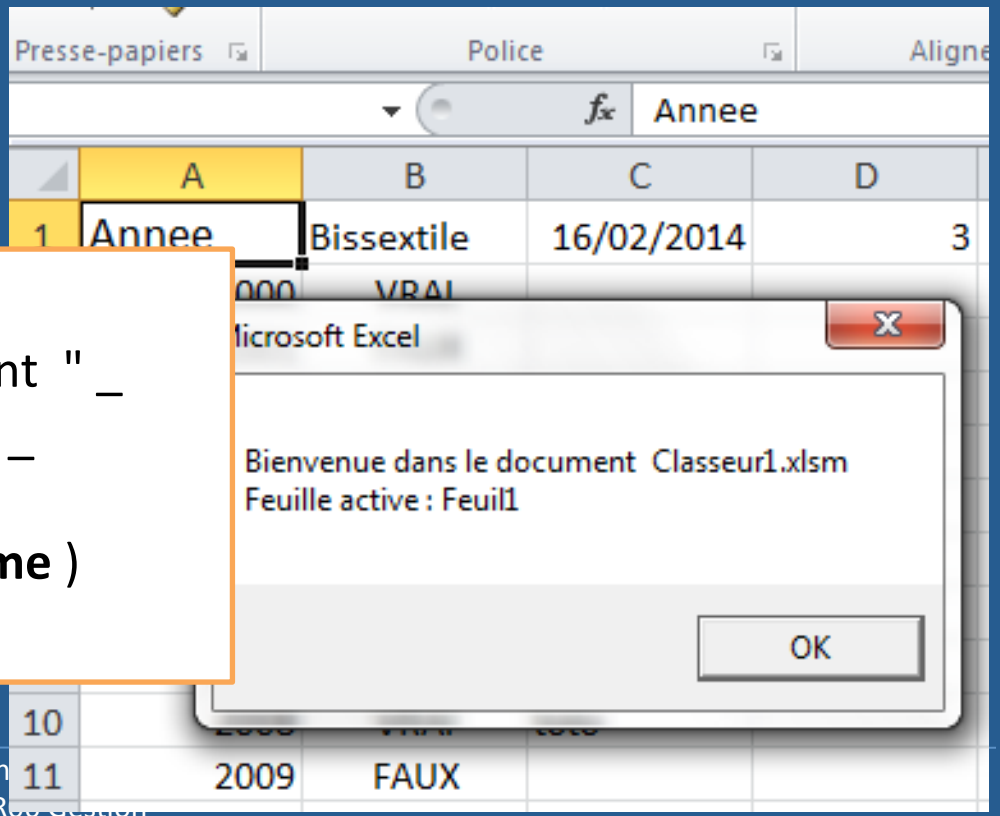
VBA : concepts de base

- Evénements
  - Exemple : affichage d'un message à l'ouverture d'un document Excel

élément qui reçoit l'action (« Workbook »)

quelle type d'action (« Open »)

```
Private Sub Workbook_Open()
  MsgBox ("Bienvenue dans le document  " _
      & ThisWorkbook.Name & Chr(10) _
      & "Feuille active : " _
      & ThisWorkbook.ActiveSheet.Name )
End Sub
```



VBA : évènements & procédures

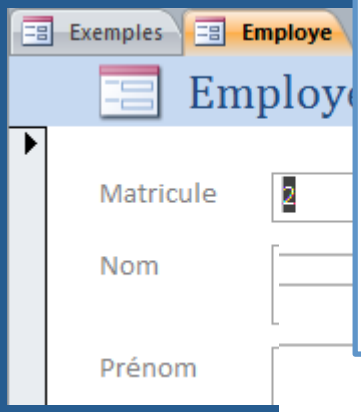
- **Evènements & Procédures**
  - A partir d'un événement, on peut invoquer une fonction ou une procédure Sub

```
Private Sub Ancien_Click()
    annees = Anciennete (Forms("Employe").Controls("Embauche"))
    MsgBox "Anciennete de " & annees & " ans " _
        & "à partir du " & Forms("Employe").Controls("Embauche"), vbInformation
End Sub
```

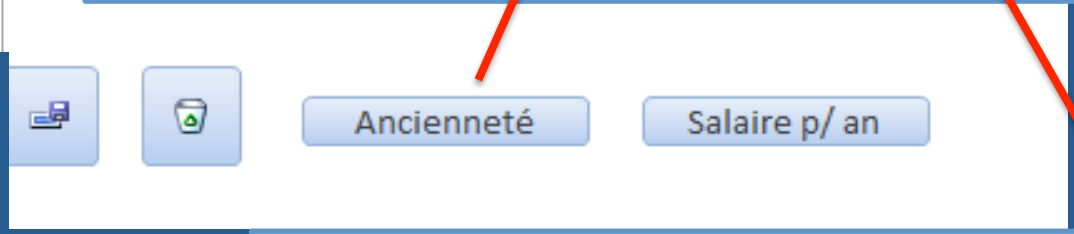
```
Public Function Anciennete(dateEntree As Date) As Integer
    Dim auj As Date
    auj = Date      'date système actuelle
    Anciennete = DateDiff("yyyy", dateEntree, auj)
End Function
```

& : concaténation
_ : instruction sur plusieurs lignes

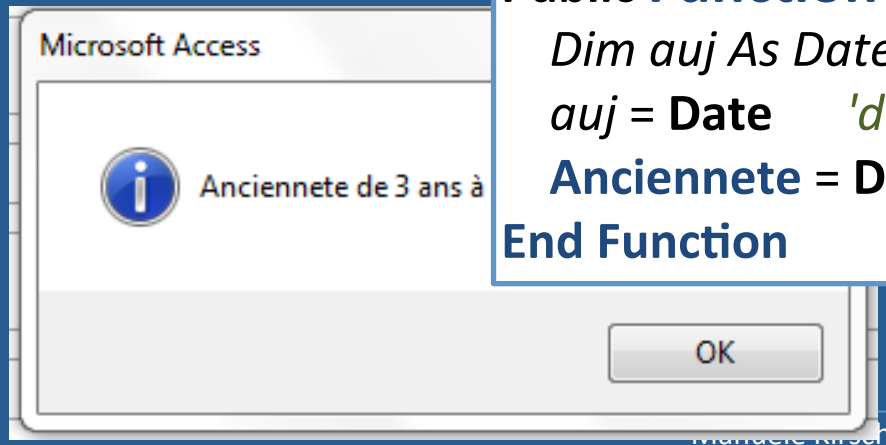
VBA : évènements & procédures



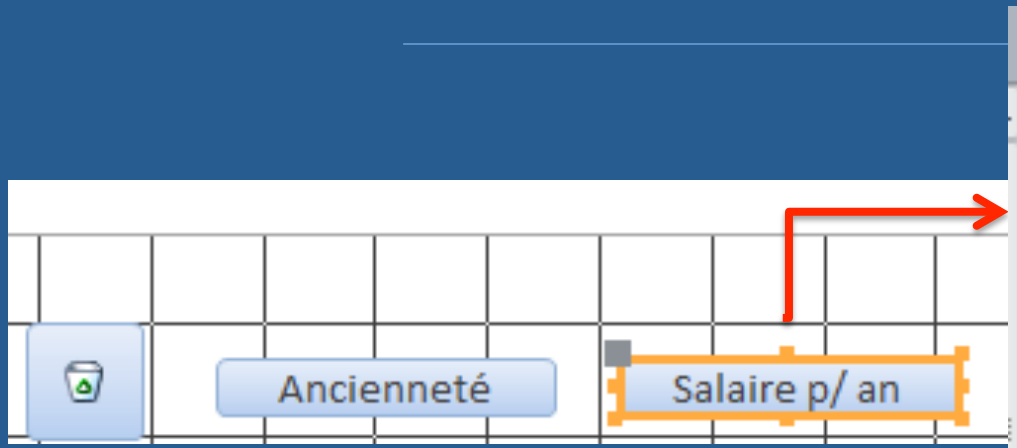

```
Private Sub Ancien_Click()
    annees = Anciennete (Forms("Employe").Controls("Embauche"))
    MsgBox "Anciennete de " & annees & " ans " _
        & "à partir du " & Forms("Employe").Controls("Embauche"), _
            vbInformation
End Sub
```

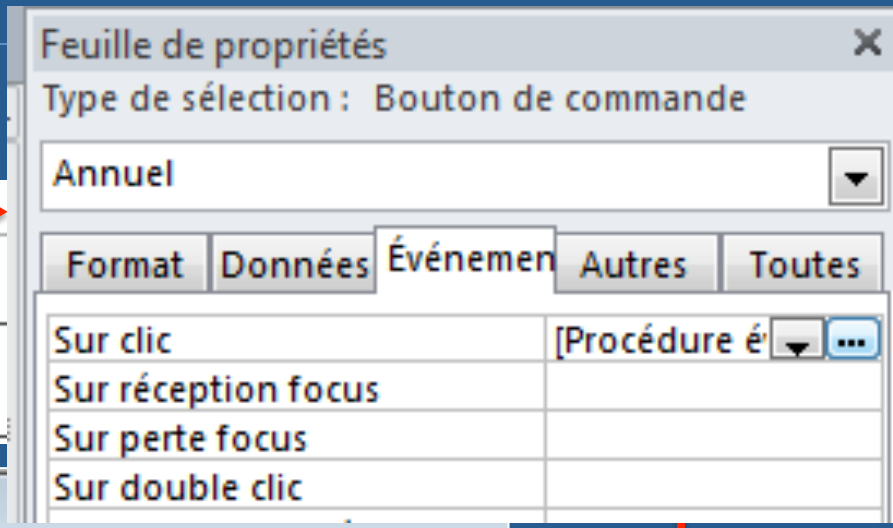



```
Public Function Anciennete(dateEntree As Date) As Integer
    Dim auj As Date
    auj = Date      'date système actuelle
    Anciennete = DateDiff("yyyy", dateEntree, auj)
End Function
```



VBA : évènements & procédures





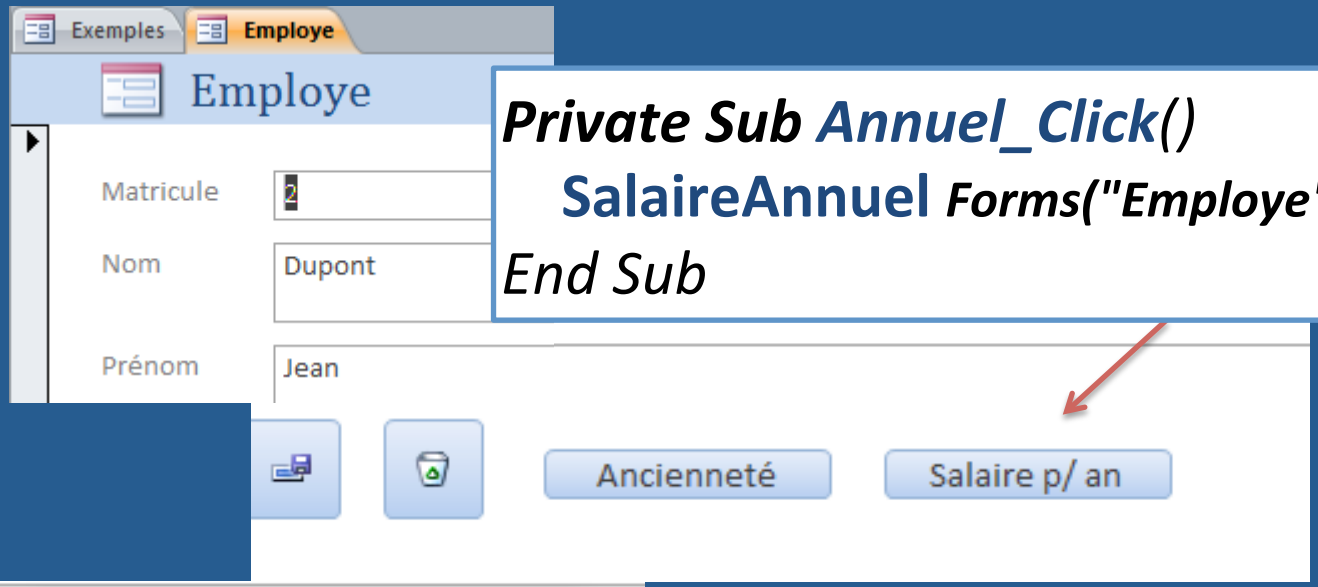
Database1 - Form_Employe (Code)

Annuel | Click

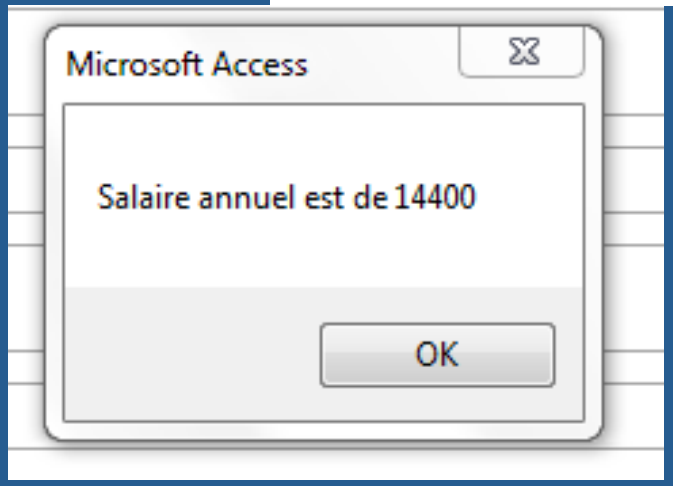
```
        MsgB                                annees & " ans " _
            &                               s("Employe").Controls("Eml
End Sub

Private Sub Annuel_Click()
    SalaireAnnuel Forms("Employe").Controls("Salaire")

End Sub
```

Objet_Evénement
NomBouton **_ Click**

VBA : évènements & procédures

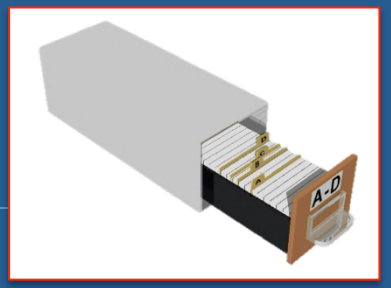



```
Private Sub Annuel_Click()
    SalaireAnnuel Forms("Employe").Controls("Salaire")
End Sub
```




```
Public Sub SalaireAnnuel(Salaire As Single)
    Dim Annuel As Single
    Annuel = Salaire * 12
    MsgBox "Salaire annuel est de " & Annuel
End Sub
```

VBA : tableaux



- **Tableaux**
  - Variables capables de garder **plusieurs valeurs**
  - Variable au **multiplicité > 0**
  - Première position est ***0***, ***sauf indication*** contraire

```
Dim tab_0_a_2(3) As Currency
tab_0_a_2(1) = 1.5
```

tab_0_a_2

0	1.5	0
0	**1**	2

```
Dim tab_1_a_3(1 To 3) As Currency
tab_1_a_3(1) = 1.5
```

tab_1_a_3

1.5	0	0
1	2	3

```
Dim tabBiDim(2, 2) As Currency
tabBiDim(0, 0) = 1.5
tabBiDim(1, 0) = 1.5
```

tabBiDim

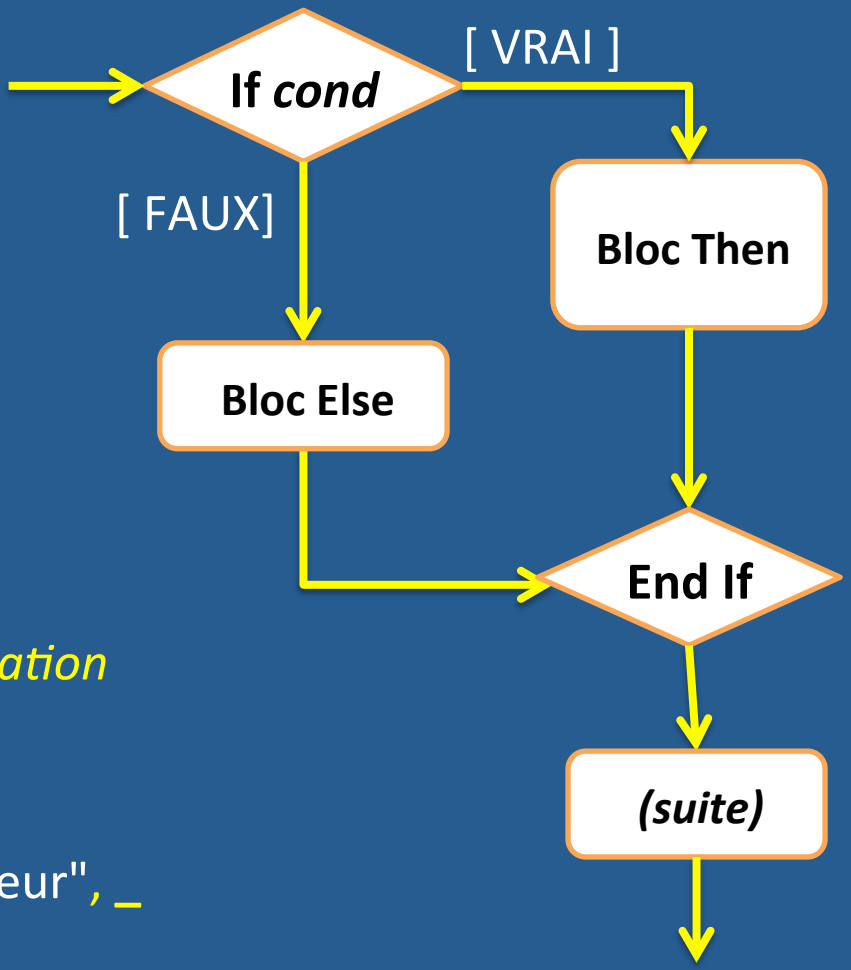
	0	1
0	**1.5**	0
1	**1.5**	0

VBA : Instructions de contrôle

- Instructions conditionnelles : **if… else…**

```
If condition Then
    'si condition vraie …
Else
    'si condition faux
End If
```

optionnel




```
If mont <= 100 Then
    MsgBox "Frais " & mont & _
        " inférieur à 100 : Validé !" , vbInformation
Else
    MsgBox "Frais " & mont & _
    " supérieur à 100 : Validation par le directeur", _
     vbOKOnly
End If
```

VBA : Instructions de contrôle

- Instructions conditionnelles : **if…elseIf…else…**

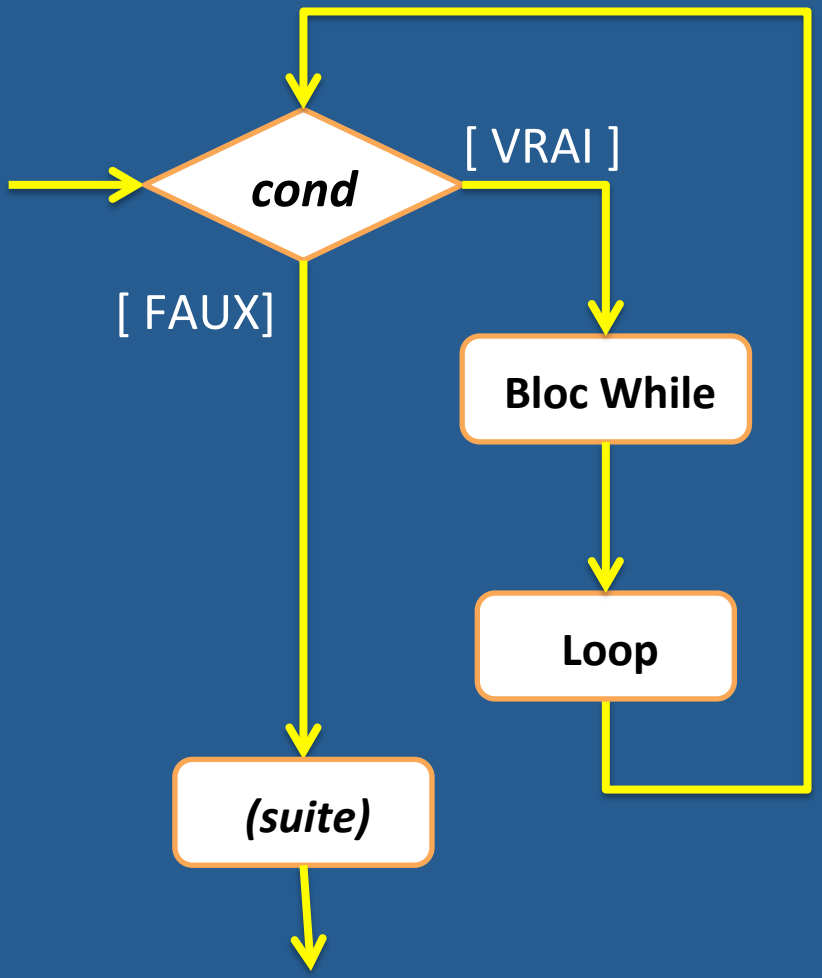
```
If condition Then
    'si condition vraie …
ElseIf condition2 Then
    'si condition2 vrai
Else
    'si aucune des
          ' précédentes
End If
```

```
If mont > 0 And mont <= 50 Then
    MsgBox "Frais " & mont & _
        " inférieur à 50 : Validé !", vbInformation
ElseIf mont <= 100 Then
    MsgBox "Frais " & mont & _
     " supérieur à 50 et inférieur à 100 : " _
        & " Validation par le RH",   vbOKOnly
Else
    MsgBox "Frais " & mont & _
    " supérieur à 100 : Validation par le directeur", _
     vbOKOnly
End If
```

VBA : Instructions de contrôle

- Boucles : **Do While… Loop**

```
Do While condition
    ' exécute tant que la
    ' condition est vraie
Loop
```



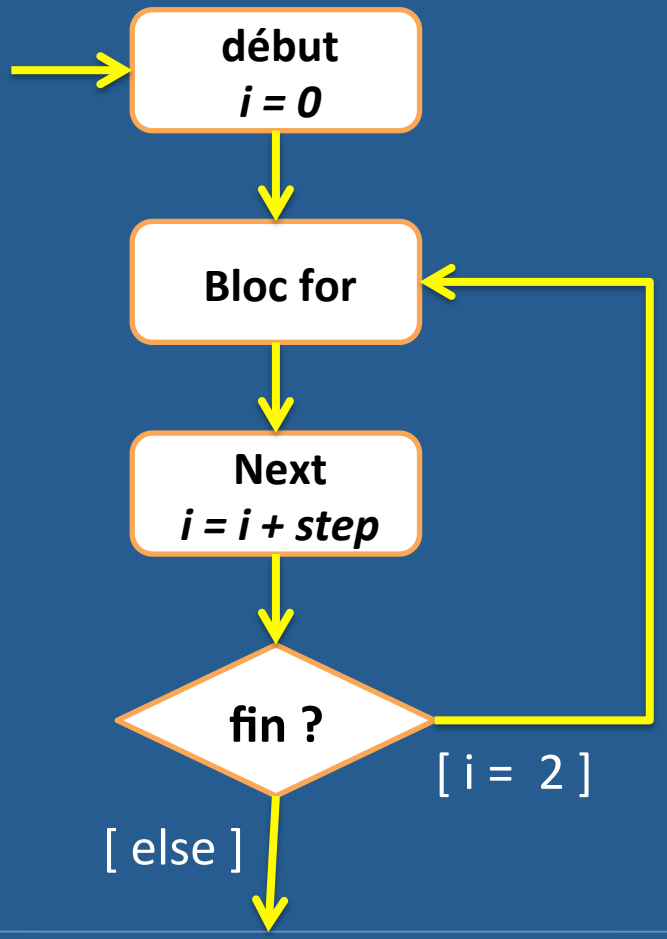

```
'tant qu'il reste des registres dans le
'RecordSet
Do While Not rsl.EOF
    tot = tot + rsl("Montant")
    rsl.MoveNext
Loop
```

VBA : Instructions de contrôle

- Boucles : **For…Next, For each … Next**

```
For début To fin Step
    ' exécute de début
    ' jusqu'à fin
    ' de n en n (step)
Next
```

```
For i = 0 To 2  Step 1
    somme = somme + tab_0_a_2 ( i )
Next
```



VBA : Instructions de contrôle

- Boucles : **For…Next, For each … Next**
  - On peut imbriquer plusieurs boucles…

```
For début To fin Step
    ' exécute de début
    ' jusqu'à fin
    ' de n en n (step)
Next
```

```
For i = 0 To 1
    For j = 0 To 1
        somme = somme + tabBiDim ( i , j )
    Next
Next
```

Pour chaque valeur de ***i***, on fait ***j*** de ***0 à 1***

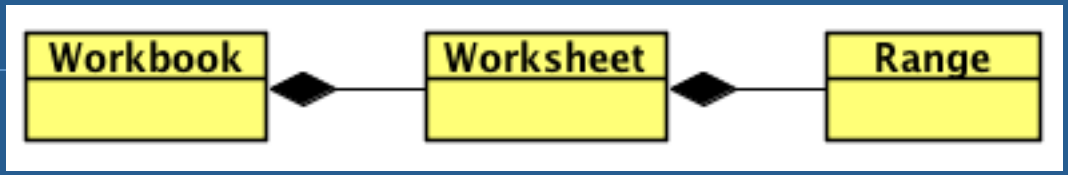
VBA : Instructions de contrôle

- Boucles : **For…Next, For each … Next**
  - On peut aussi parcourir toute une collection

```
For Each var IN collection
    ' pour chaque élément var
    ' dans la collection
    ' (ou tableau)
Next
```

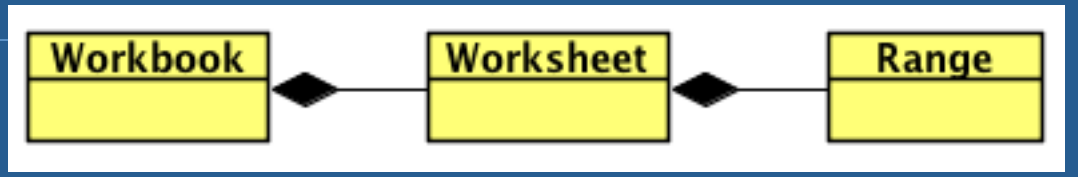
```
'on va afficher les nombre de registres dans
'chaque table
For Each t In CurrentDb.TableDefs
    MsgBox t.Name & " : " & t.RecordCount
Next
```

VBA : accès aux données EXCEL



- Les données dans un document Excel (objet ThisWorkbook) s'organisent en feuilles de calcul (objets **Worksheet**) et en cellules (objets **Range**)
- Quelques objets importants
  - **ThisWorkbook** : document courant
  - **ActiveWorkbook** : document en 1er plan (actif)
  - **ActiveSheet** : feuille de calcul active
- Les **collections** vont permettre l'accès aux différents objets de la classe correspondante
  - **Worksheets**("Feuil1") → accès à la feuille « Feuil1 »

VBA : accès aux données EXCEL



- Range
  - La notion de Range est essentielle dans Excel.
  - Un objet Range est une région dans une feuille contenant une ou plusieurs cellules
  - Les opérations de la classe Range vont nous permettre d'interagir ou de modifier les cellules

L'opération **Value** récupère la **valeur de la cellule**

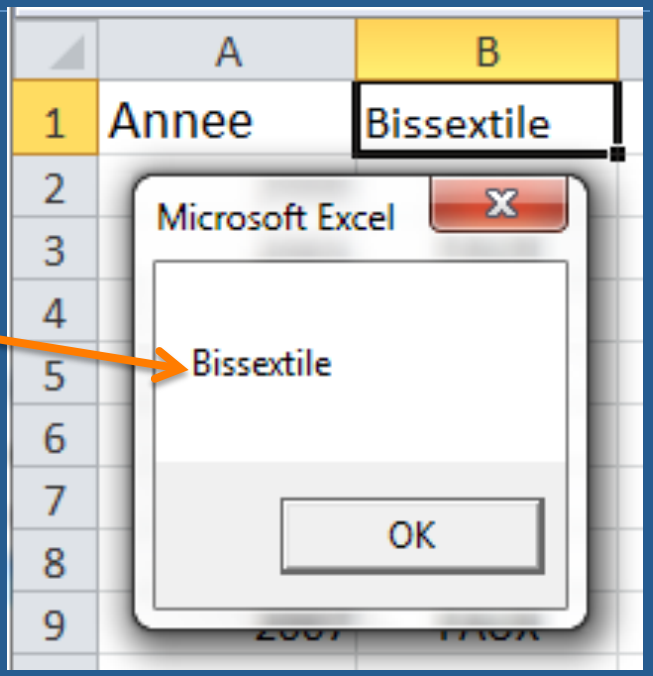
```
ActiveSheet.Range("A1").Value
ActiveSheet.Cells(1, 2).Value
```

La collection **Cells** contient un ensemble d'objets **Range**

Attention : les collections démarrent en **1 (et *pas 0*)**

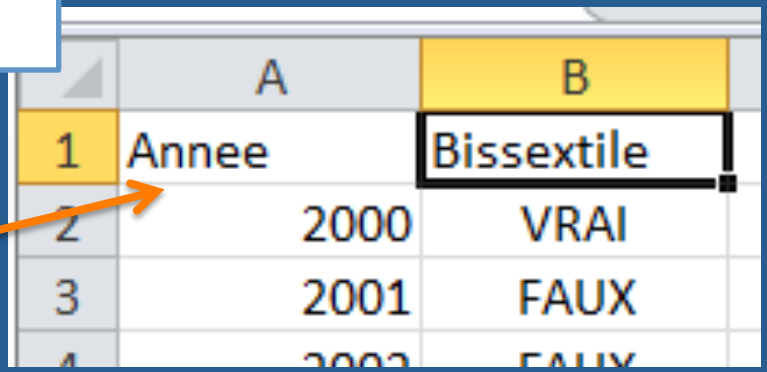
VBA : accès aux données EXCEL

```
Sub UnRange()
    MsgBox ActiveSheet.Range("A1").Value
    MsgBox ActiveSheet.Cells(1, 2).Value
End Sub
```



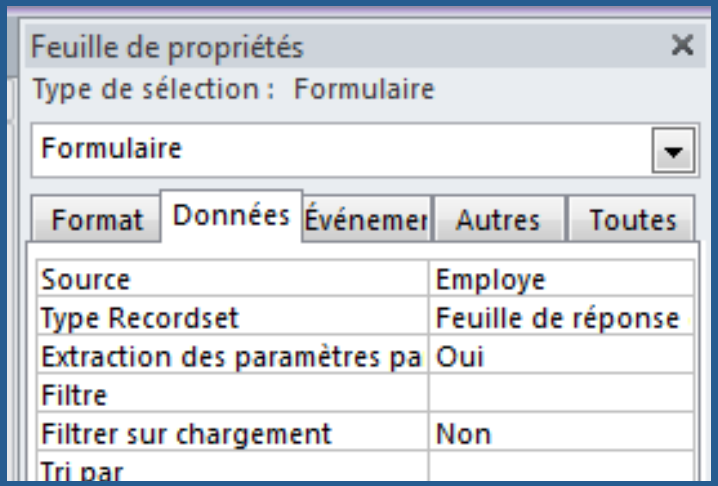

```
Private Sub ChangeA1()
    Worksheets("Feuil1").Cells(1, 1).Font.Size = 12
End Sub
```

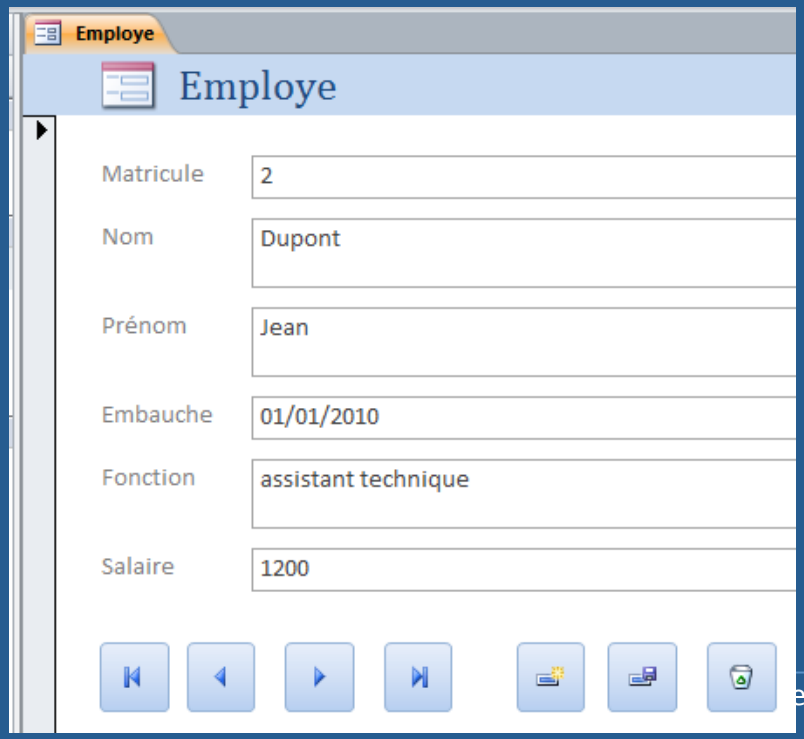
changement de la taille de la police



VBA : accès aux données ACCESS

- Accès aux données des tables peut se faire de différentes manières
  - Par formulaires, par requête...
    - **Source** : Form attaché à une table

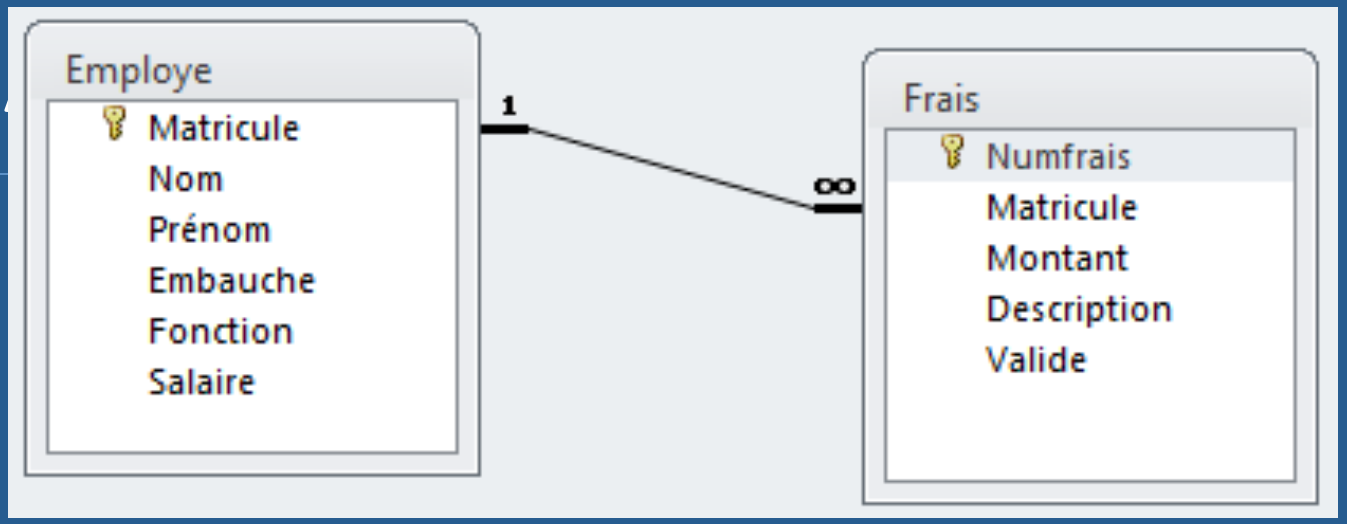


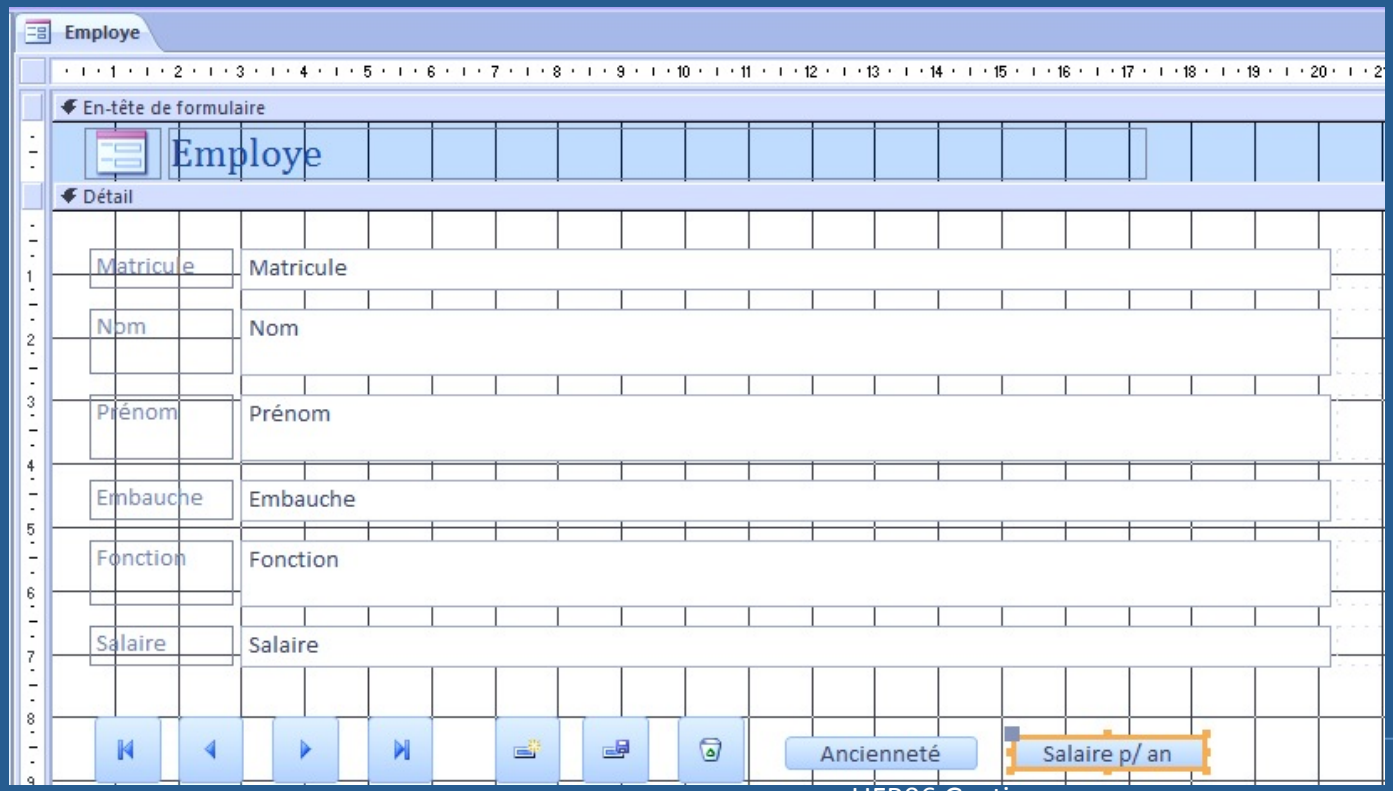


  - On peut même ouvrir des bases sur d'autres fichiers
    - Ex.: un fichier Excel qui importe les données d'un fichier Access

VB

- Exemple Employés





Formulaire « Employe »

(mode Création)

VBA : accès aux données ACCESS

- Quelques objets importants
  - **DBEngine** : objet qui gère l'accès à une BdD
  - **Database** : BdD ouverte

    *bds = **DBEngine.OpenDatabase**("c:\docs\clients.accdb")*

    *bds = **CurrentDb**()*
  - **RecordSet** : ensemble d'enregistrements
    - Manipulation des enregistrements

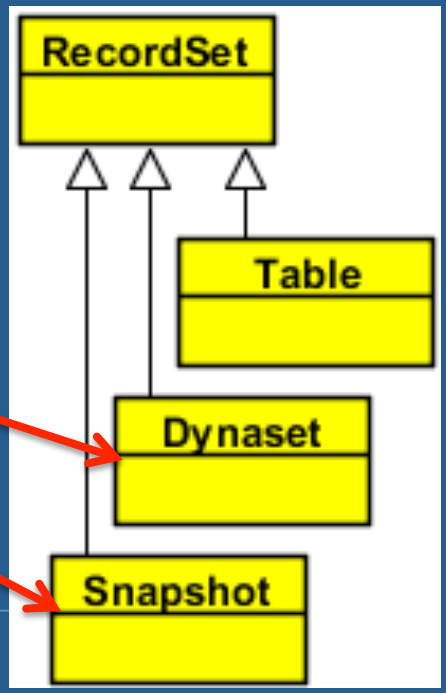
*Set rst = **dbs.OpenRecordSet**("Frais", **dbOpenTable**)*

*Set rst = **CurrentDb().OpenRecordset**(sql)*

rst.MoveFirst

rst.MoveNext

rst.Edit



VBA : Accès aux BdD

- Exemple : calculer les frais…

```
matr = Me.Controls("Matricule")
sql = "SELECT Frais.Montant FROM Frais WHERE Frais.Matricule=" & matr

Set rsl = CurrentDb().OpenRecordset(sql)

If rsl.RecordCount > 0 Then
   rsl.MoveFirst
   Do While Not rsl.EOF
      tot = tot + rsl("Montant")
      rsl.MoveNext
   Loop
End If

MsgBox "Total des frais = " & tot, vbOKOnly
```

Exécution de la requête SQL

S'il y a des enregistrements (***RecordCount***)

Avance au **premier** enregistrement

Tant qu'on n'arrive **pas** à la fin des enregistrements(**EOF**)

Récupère un champs ***rsl ("attribut")***

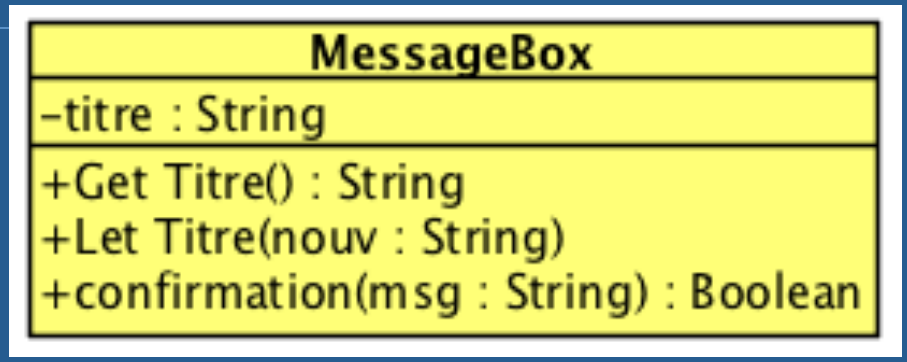
Avance au **prochain** enregistrement

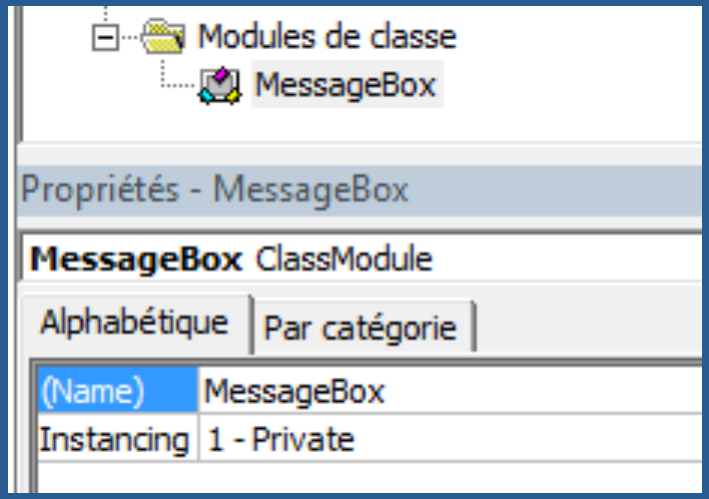
VBA : modules de classe

- Modules de classes
  - Définition d'une **classe**
    - **Attributs** → variables
    - **Opérations**
      - **Property Get / Let**
      - **Function / Sub**

*Les **attributs** sont « exposés » en tant que « **Property** » par les **Get et Let***

  - Usage :
    - Dim msg AS New MessageBox
    - Set msg = **New** MessageBox
    - **msg.Titre = "nouveau titre"** exécute **Let Titre**
    - … **msg.Titre** … exécute **Get Titre**





VBA : modules de classe

MessageBox
-titre : String
+Get Titre() : String +Let Titre(nouv : String) +confirmation(msg : String) : Boolean

Database1 - MessageBox (Code)

(Général)

```
Option Compare Database

Dim strTitre As String

Public Property Get Titre() As String
   Titre = strTitre
End Property

Public Property Let Titre(nouvTitre As String)
    strTitre = nouvTitre
End Property

Public Function confirmation(msg As String) As Boolean
    If MsgBox(msg, vbYesNo, strTitre) = vbYes Then
        confirmation = True
    Else
        confirmation = False
    End If
End Function
```

Attributs

Get : getter (***getTitre***)
Accès à la propriété
Titre

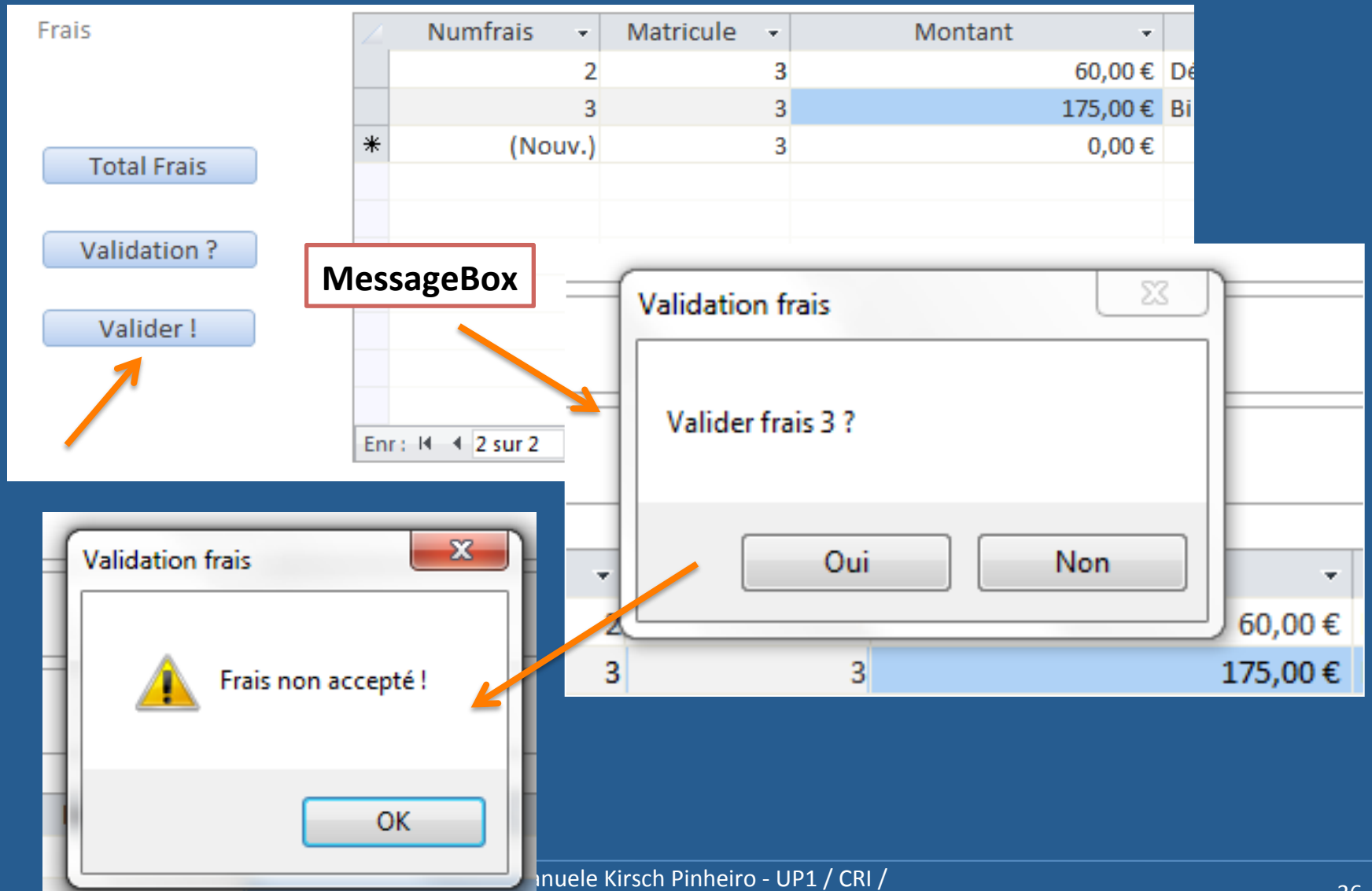
Let : setter (***setTitre***)
Modifie la propriété
Titre

Function confirmation
Opération

VBA : modules de classe

```
Private Sub btnValider_Click()
 'déclaration d'un objet de type MessageBox
   Dim msg As New MessageBox
 'usage de l'objet Msg
   msg.Titre = "Validation frais "     'VBA appelle le Let Titre
   If msg.confirmation("Valider frais ? ") = True Then
      valider Me.Frais.Controls("NumFrais"), True
      MsgBox "Frais confirmé !", vbOKOnly, msg.Titre
   Else
      valider Me.Frais.Controls("NumFrais"), False
      MsgBox "Frais non accepté !", vbOKOnly + vbExclamation, _
           msg.Titre                    'ici VBA appelle le Get Titre
   End If
End Sub
```

VBA : modules de classe



– PANTHÉON - SORBONNE –
UNIVERSITÉ PARIS 1

GESTION Sorbonne PARIS 1

Informatique
Modélisation UML

Objectifs :

Exemple de projet en VBA

Rappel VBA

- VBA est un langage de programmation (à l'instar de PHP) qui est attaché aux documents MS Office
- Un programme VBA n'existe que s'il est attaché à un document MS Office (une fiche Excel, **une base de données Acces**)
- Grâce au VBA, on peut faire des calculs avancés, de vérifications, etc. à l'intérieur des documents MS Office

Exemple en VBA

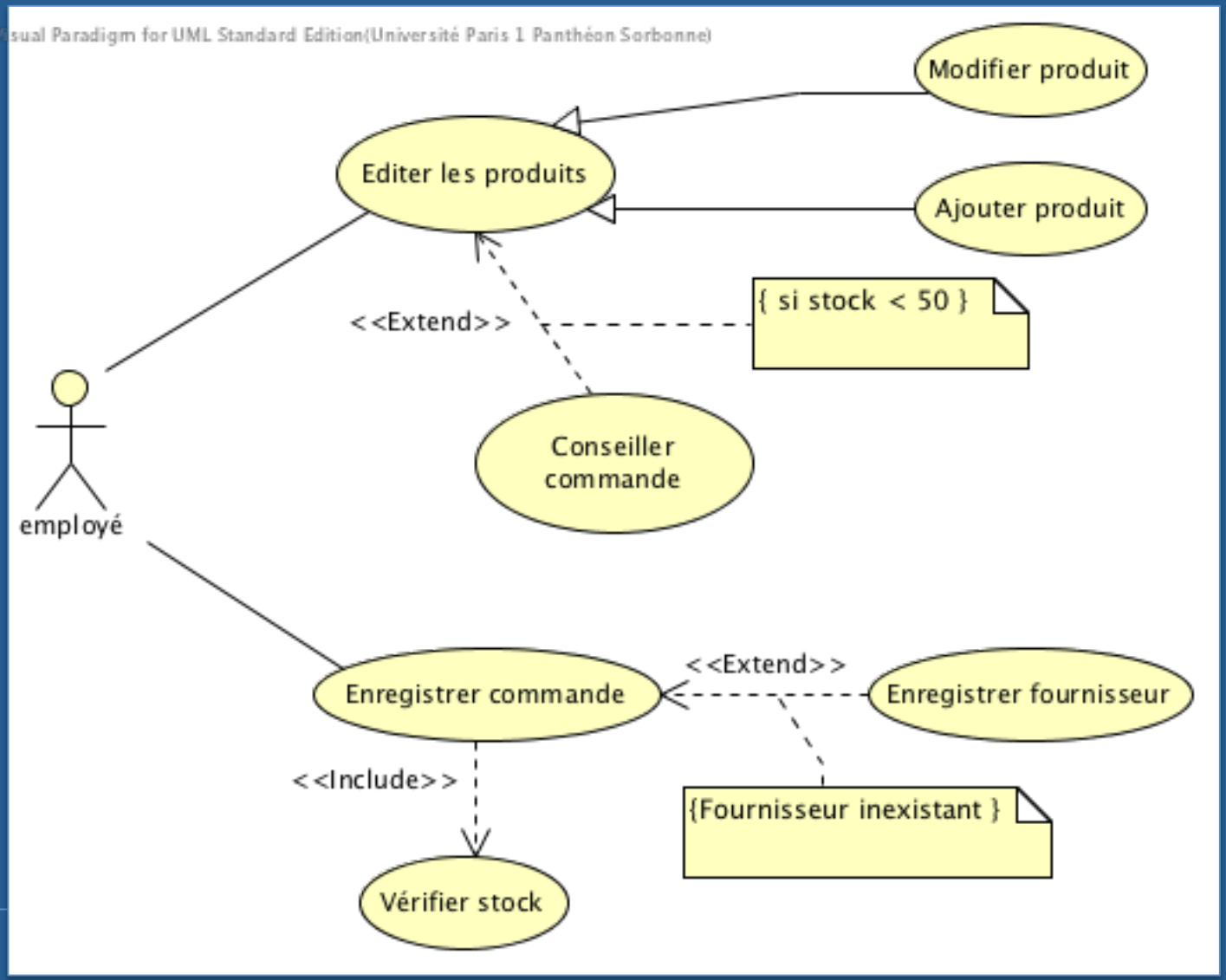
- **Exemple de projet**
  - Un employé d'une société de vêtements a besoin d'un petit système pour l'aider **à gérer les commandes** qu'il passe auprès des fournisseurs
  - Le système doit **enregistrer les produits** et les **commandes**
  - Si le **stock est inférieur à un seuil** (50 produits), le système doit **conseiller** l'employé à **passer une commande**
  - L'employé doit être capable d'**enregistrer** un nouveau **fournisseur** lors qu'il passe une commande

Exemple VBA

- **Démarche**
  - Établir les **besoins** : *pour **quoi** faire ?*
    - Diagrammes de **cas d'utilisation**
  - Établir les **données** : *que manipule-t-on ?*
    - Premier **diagramme de classes**
    - **Diagramme objets** pour illustrer le diagramme de classes
  - Établir les **processus** : *que faire ? Dans quel ordre ?*
    - Réfléchir aux processus mis en place
    - **Diagrammes d'activités**
  - Imaginer les **interactions** : ***Comment** procède-t-on ?*
    - Quels formulaires aurons-nous proposer ? Quels états (rapports) ?
    - Réfléchir aux classes « contrôle » et aux modules nécessaires
    - Améliorer / compléter le **diagramme de classes**

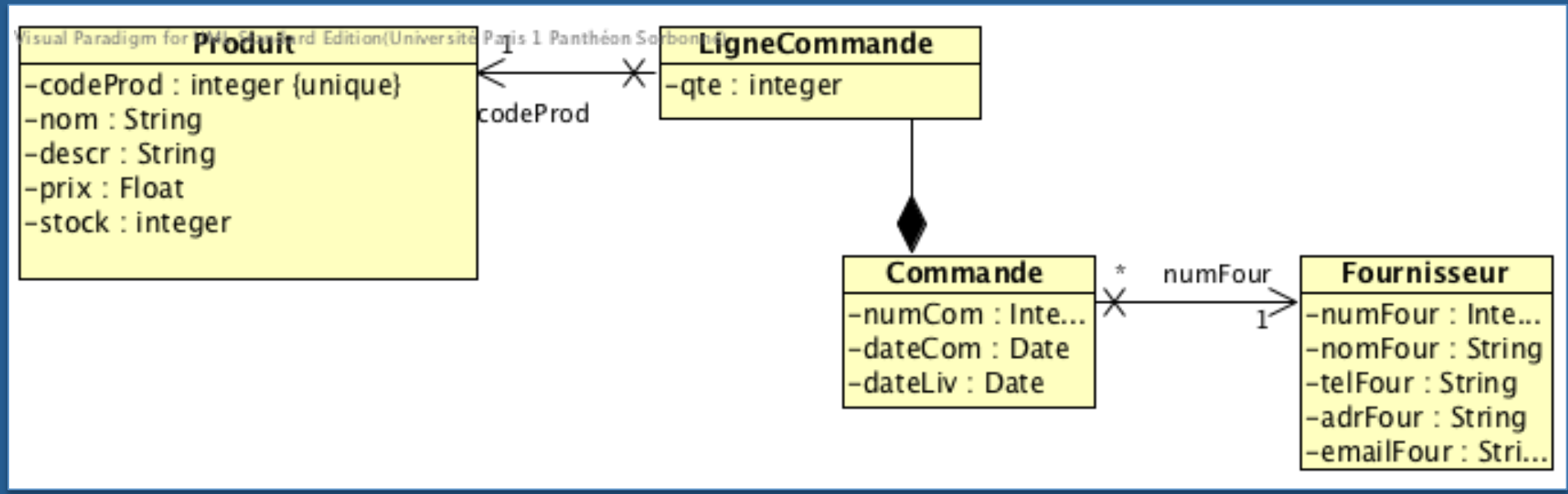
Exemple en VBA

- Diagramme de Cas d'Utilisation : **fonctionnalités**



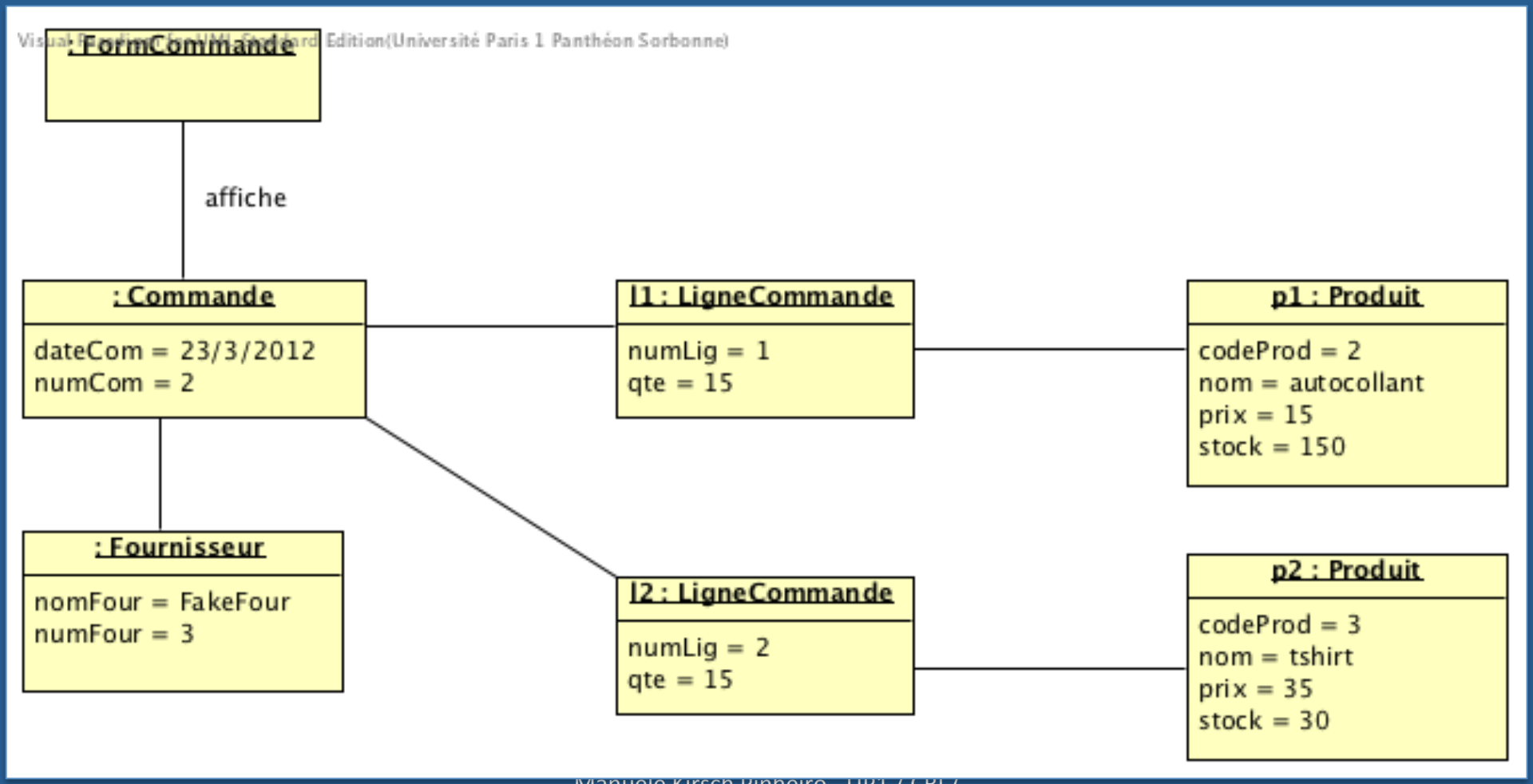
Exemple VBA

- Diagramme de classes : données manipulées



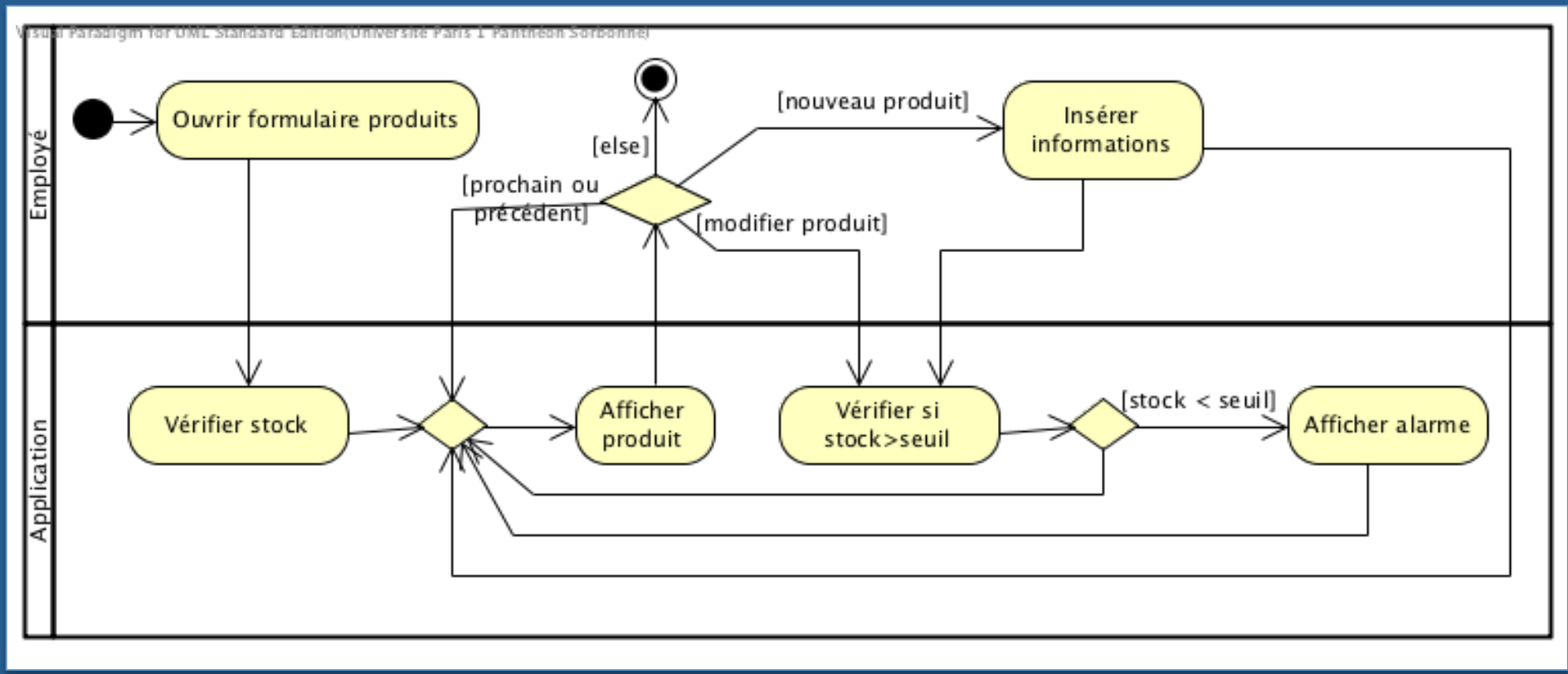
Exemple VBA

- Diagramme objets : une commande



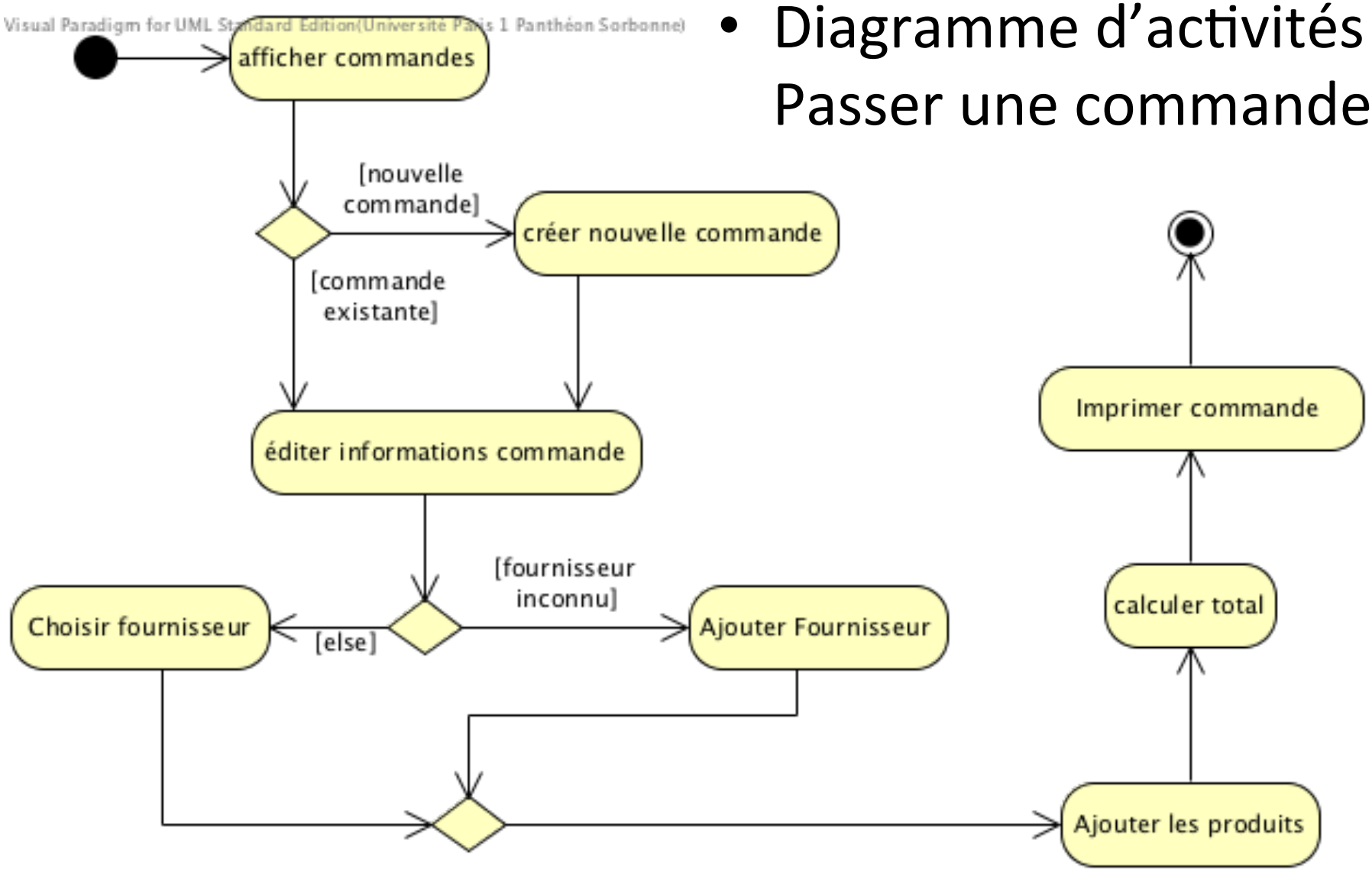
Exemple VBA

- Diagramme d'activités : éditer produits



Exemple VBA

- Diagramme d'activités : Passer une commande

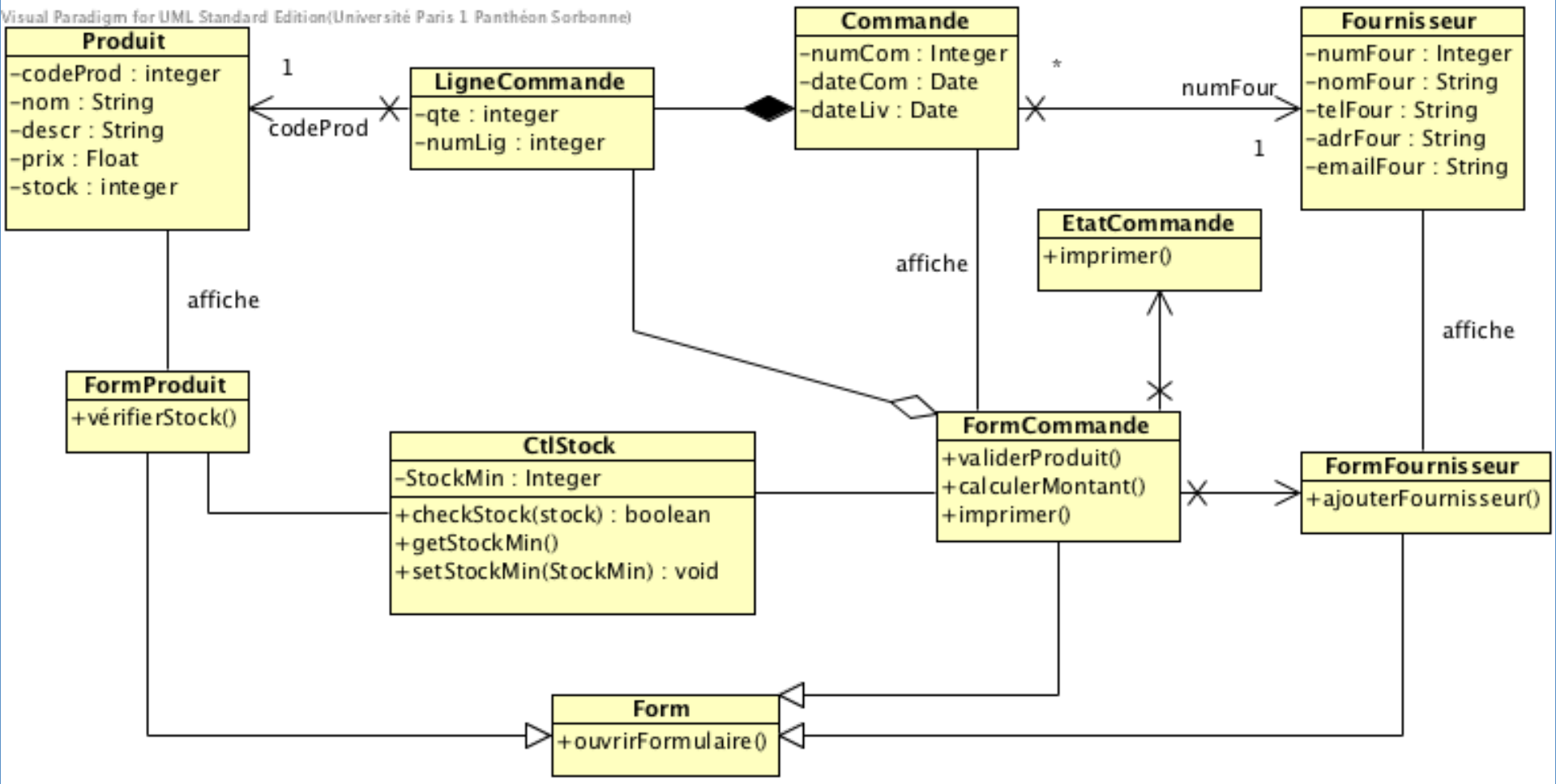


Exemple VBA

- A partir des diagrammes d'activités, nous pouvons établir les **formulaires** et **états** souhaités
  - Formulaire pour nouveau produit
  - Formulaire pour une nouvelle commande
  - Rapport (état) sur une commande (pour l'imprimer)
- Enrichir diagramme de classes :
  - Ajouter les formulaires et les états
  - Ajouter les éléments de « contrôle » (module de classes)
    - contrôle de stock

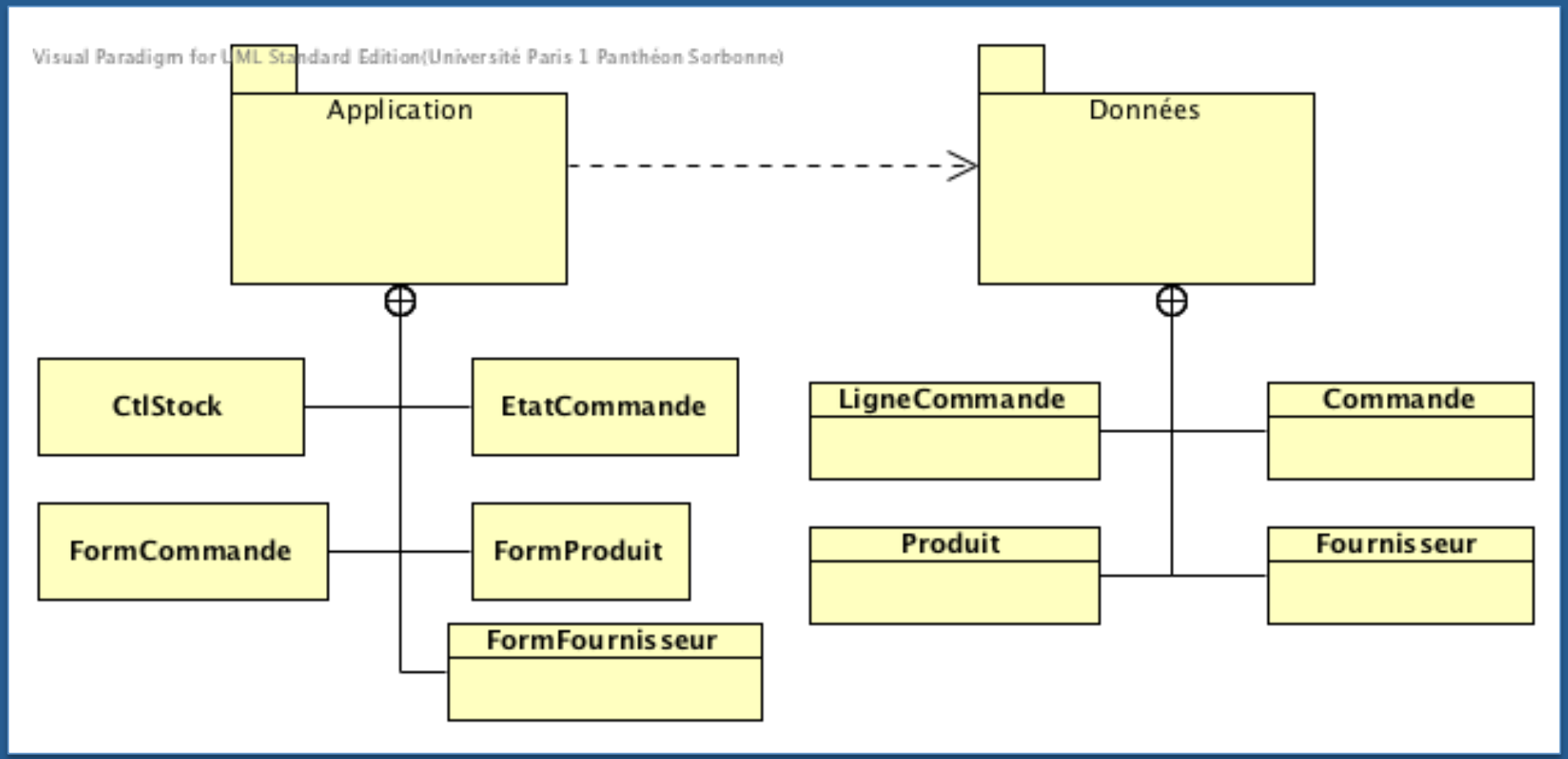
Exemple VBA

- Diagramme de classes revisité



Exemple VBA

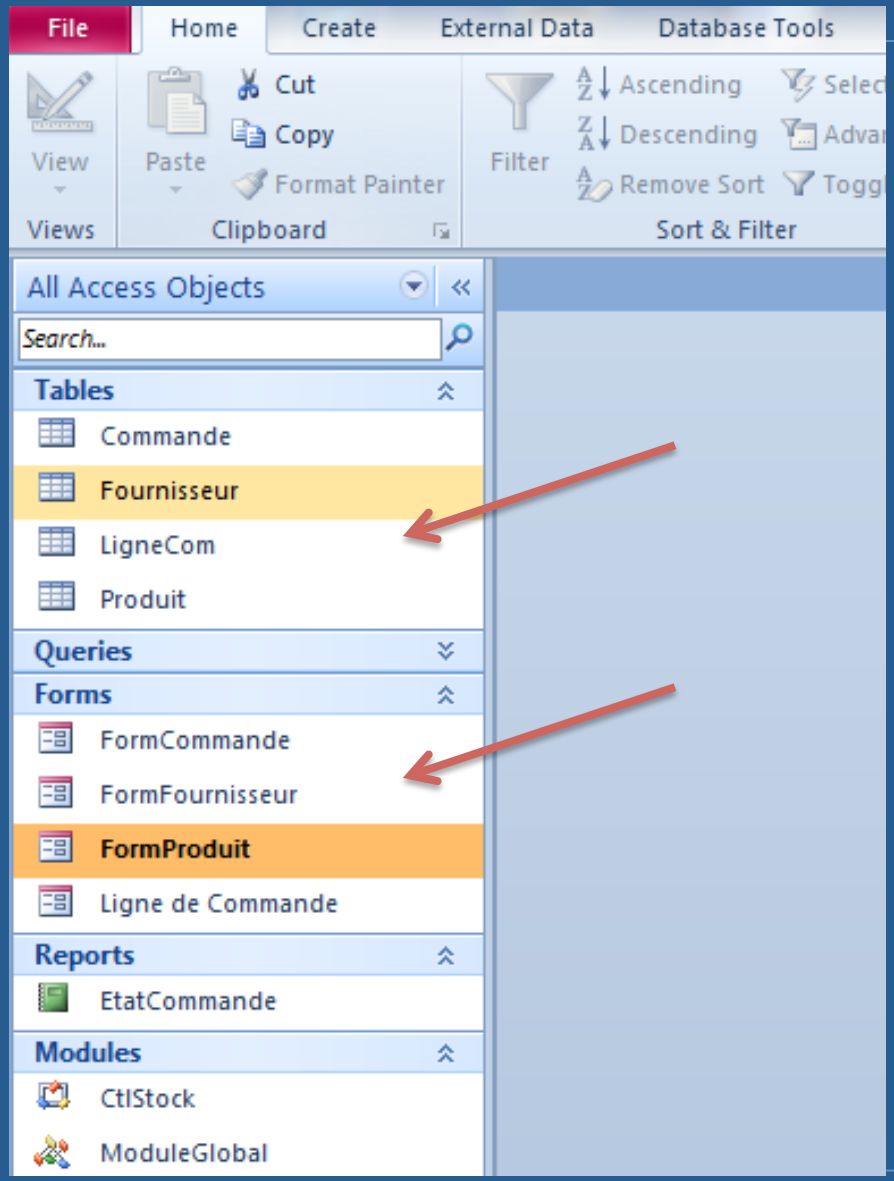
- Diagramme de paquetages



Exemple VBA

- Passage du projet à VBA
  - Création de la base de données
    - Les **tables** correspondent aux classes de données
  - Création des **formulaires**
    - Correspondant aux classes **Form**
  - Création des modules de programmation
    - **Module de classe** → correspond à la notion de classe
      - **CtlStock** : définition du seuil et vérification du stock
    - **Module** → ensemble d'objets et d'opérations
      - **ModueGlobal** : calculs variés (ex. : calculer total commande)

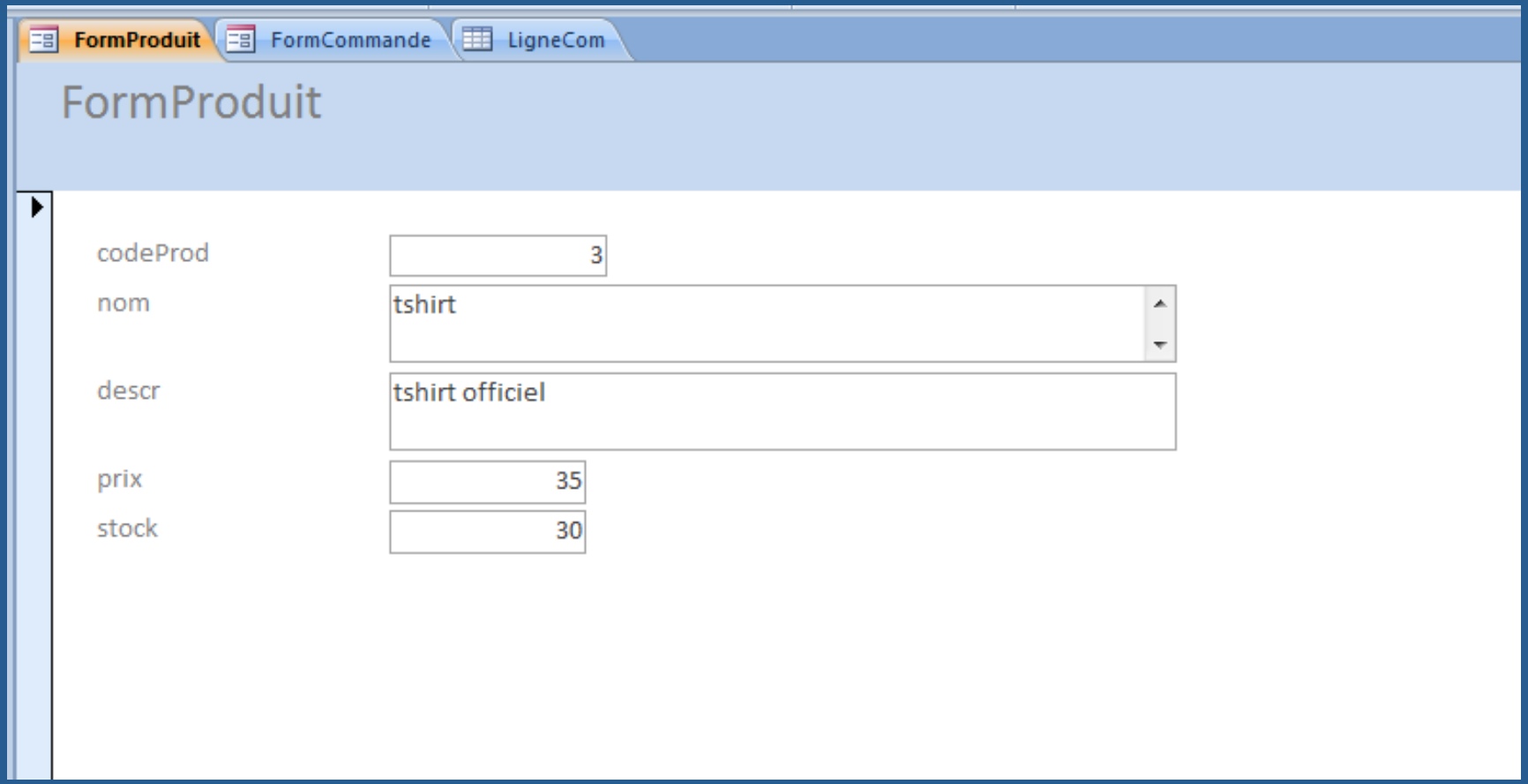
Exemple VBA



- Une « application » VBA s'utilise un document MS Office, une base MS Access dans ce cas
  - Création des tables et formulaires prévus dans le diagramme de classes dans Access

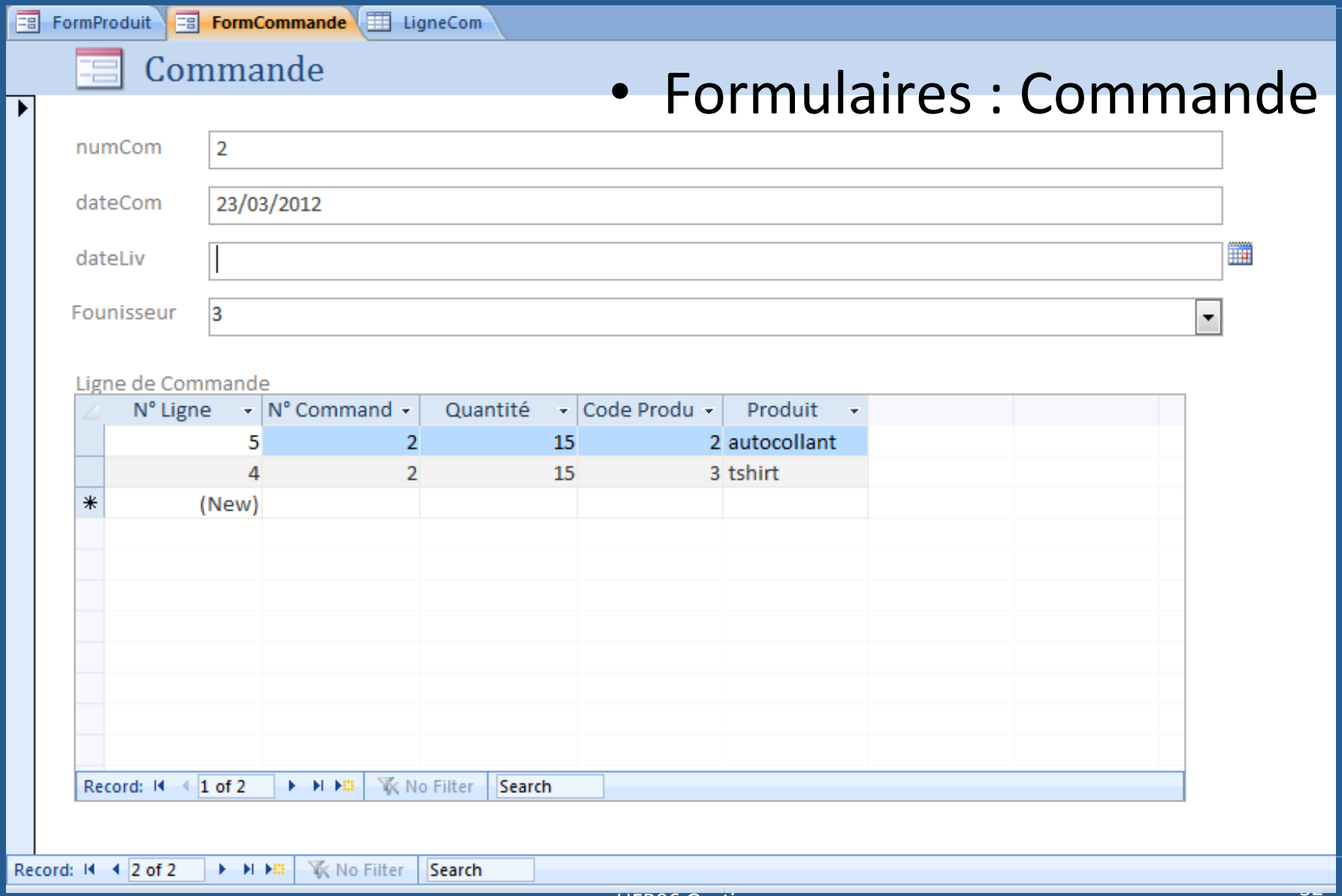
Exemple VBA

- Formulaires : Produit



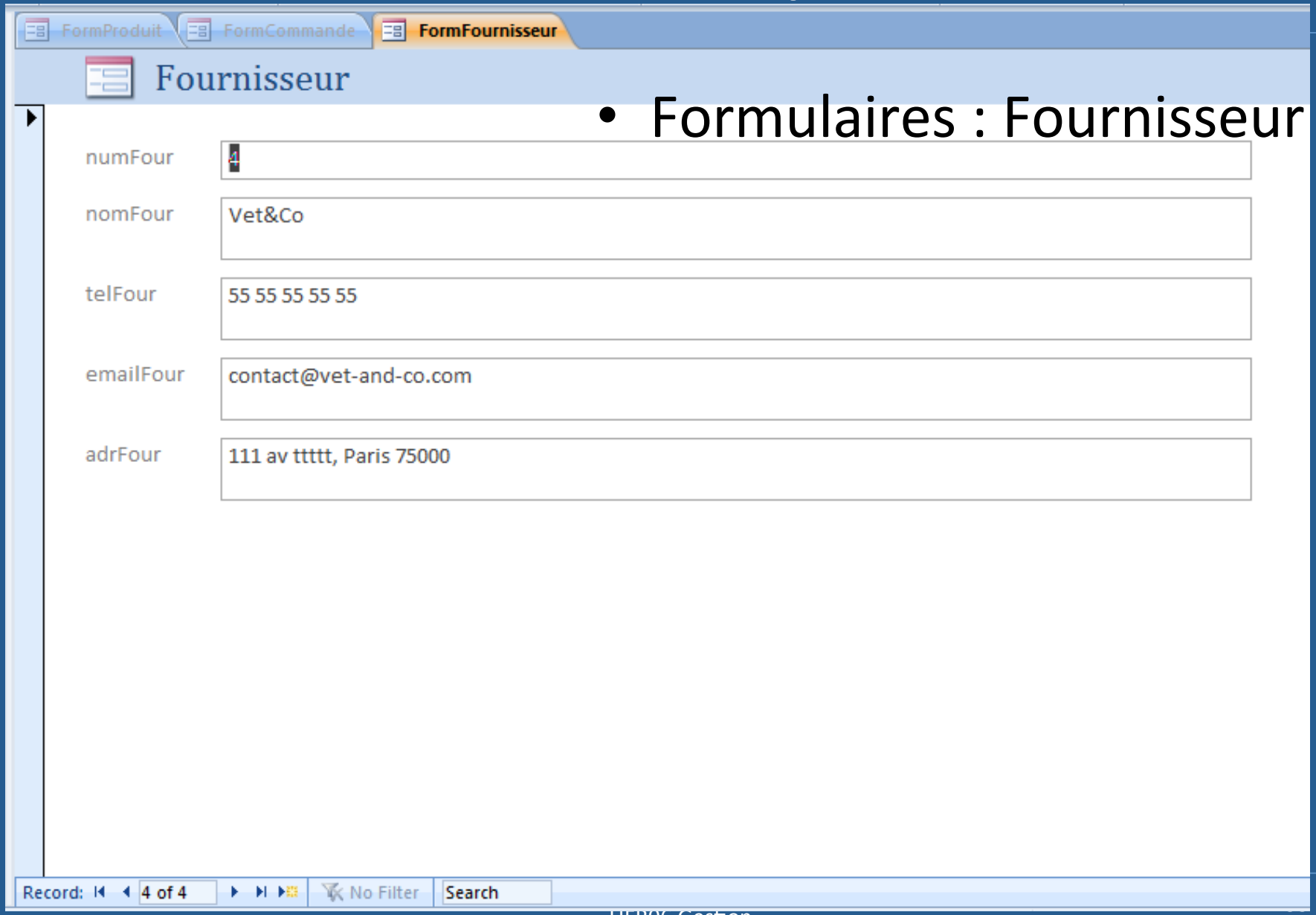
Exemple VBA

- Formulaires : Commande



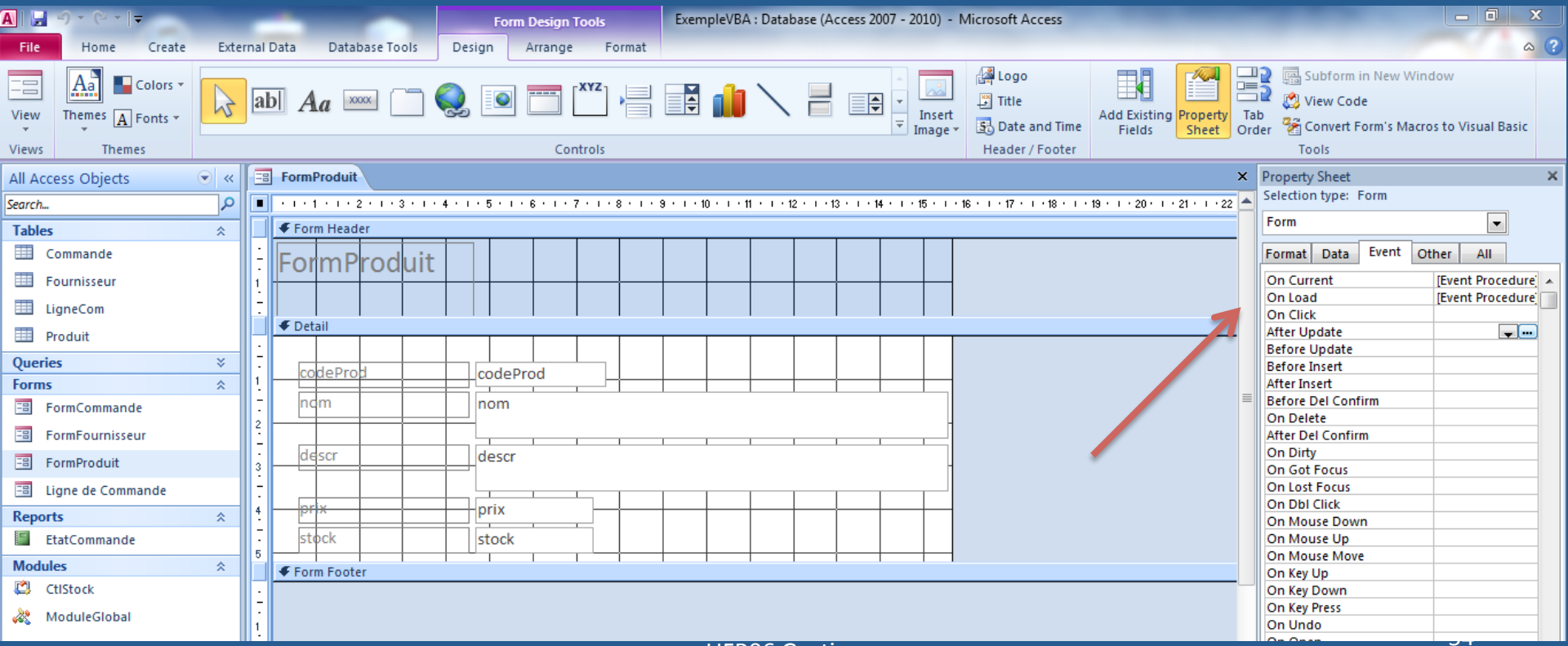
Exemple VBA

- Formulaires : Fournisseur



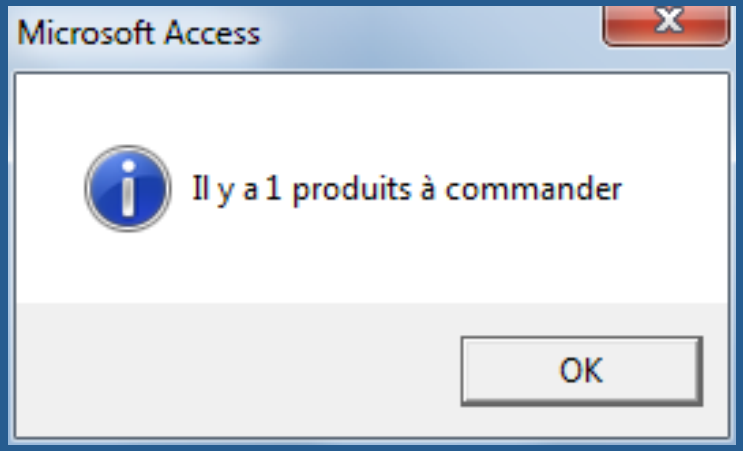
Exemple VBA

- VBA va nous permettre de programmer les vérifications qu'on a indiqué dans les diagrammes d'activités (par des évènements)

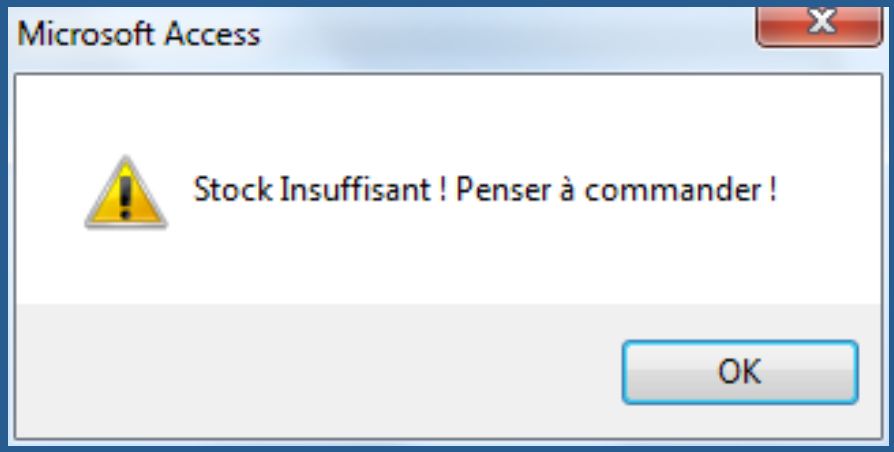


Exemple VBA

- Vérification stock lors de l'édition des produits



Avertissement à l'entrée de FormProduit



Avertissement dans FormProduit si stock < seuil

Exemple VBA

- **Code VBA pour FormProduit**

ExempleVBA - Form_FormProduit (Code)

Form | Load

```
Private Sub Form_Current()
    Dim stk As Integer
    stk = Forms("FormProduit").Controls("stock").Value
    If ModuleGlobal.controle.CheckStock(stk) <> True Then
        MsgBox "Stock Insuffisant ! Penser à commander !", vbExclamation
    End If
End Sub
```

Pour chaque produit on vérifie le stock

```
Private Sub Form_Load()
    Dim r As Recordset
    Dim sql As String

    sql = "Select * from Produit where stock < " _
       & ModuleGlobal.controle.StockMin

    Set r = CurrentDb().OpenRecordset(sql)
    MsgBox "Il y a " & r.RecordCount & " produits à commander", vbInformation

End Sub
```

À l'ouverture du formulaire, on vérifie également

```
Private Sub stock_AfterUpdate()
    Dim stk As Integer

    stk = Forms("FormProduit").Controls("stock").Value
    If stk < 0 Then
       MsgBox "Le stock doit être suppérieur à zéro", vbCritical
       Forms("FormProduit").Controls("stock").Value = 0
    End If
    If ModuleGlobal.controle.CheckStock(stk) <> True Then
        MsgBox "Le stock est inférieur à 50. Penser à passer une commande !", vbExclamation
    End If
End Sub
```

Et lorsqu'on modifie un produit, on avertit aussi si stock<seuil

Exemple VBA

ExempleVBA - ModuleGlobal (Code)

(General) (Declarations)

```
Option Compare Database
'ce module contient les objets et les variables nécessaires à tous

Public controle As New ctlStock


Public Function CalculerTotal(num As Integer)
    Dim r As Recordset
    Dim total As Double
    Dim sql As String

    sql = "Select Produit.Prix, LigneCom.codeProd, LigneCom.qte "
    sql = sql & "FROM Produit, LigneCom WHERE LigneCom.codeProd=Produit.codeProd AND "
    sql = sql & " LigneCom.numCom = " & num  'Forms("FormCommande").Controls("numCom")

    'MsgBox sql

    Set r = CurrentDb().OpenRecordset(sql)

    Do While Not r.EOF
        total = total + (r.Fields("prix") * r.Fields("qte"))
        'MsgBox total
        r.MoveNext
    Loop

    CalculerTotal = total

End Function
```

Le seuil est le même pour toute l'application :

Objet « **controle** » dans **ModuleGlobal**

Dans les **modules**, on peut aussi ajouter les **fonctions** (opérations) qu'on n'arrive pas à mettre dans les formulaires

Exemple VBA

ExempleVBA - CtlStock (Code)

(General) (Declaratic

```
Option Compare Database

    Private stkMin As Integer

    Public Sub Class_Initialize()
        stkMin = 50
    End Sub

    Property Get StockMin() As Integer
         StockMin = stkMin
    End Property

    Property Let StockMin(seuil As Integer)
        If seuil >= 0 Then     'Stock min doit être sup ou égal à
            stkMin = seuil
        End If

    End Property

    Function CheckStock(stock As Integer) As Boolean
        If stock >= stkMin Then
            CheckStock = True
        Else
            CheckStock = False
        End If

    End Function
```

Les « **Modules de Classe** » correspondent à la notion de **classe**.
On y définit des **propriétés** et des **opérations**

Getter (Get) et Setter (Let) pour la propriété stockMin
Si on fait
controle.StockMin = 30
on passe par le Let

Opération CheckStockMin

Exemple VBA

ExempleVBA - Form_Ligne de Commande (Code)

(General) | (Declarations)

```
Private Sub Form_AfterUpdate()
    Forms("FormCommande").Refresh

End Sub

Private Sub Form_BeforeUpdate(Cancel As Integer)
    Dim r As Recordset
    Dim sql As String
    Dim stk As Integer


    sql = "Select stock from Produit where codeProd="
    sql = sql & Me.Controls("codeProd")

    Set r = CurrentDb().OpenRecordset(sql)

    r.MoveFirst
    stk = r.Fields("stock").Value

    If ModuleGlobal.controle.CheckStock(stk) Then
         MsgBox "Produit possède déjà un stock de " & stk _
                & ". Commande inutile !", vbExclamation
    End If

End Sub

Private Sub numCom_Enter()
    Me.numCom = Forms("FormCommande").Controls("numCom")

End Sub
```

- **Code VBA pour la ligne de commande**

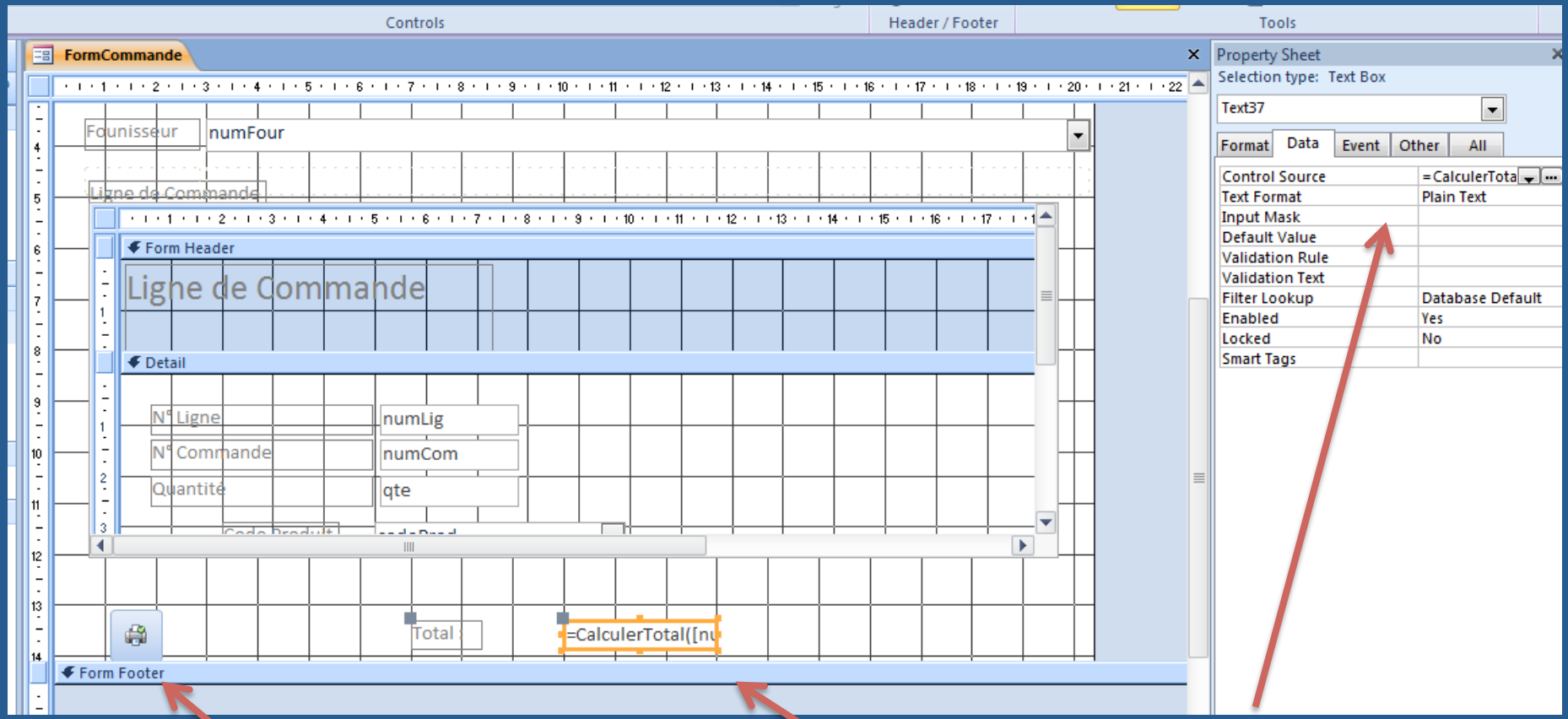
À chaque mise à jour, on actualise la commande

On vérifie le stock à chaque nouvelle ligne et on averti l'utilisateur si le stock>seuil

Les lignes de commande concernent la commande affichée

Exemple VBA

- **Code VBA pour FormCommande**

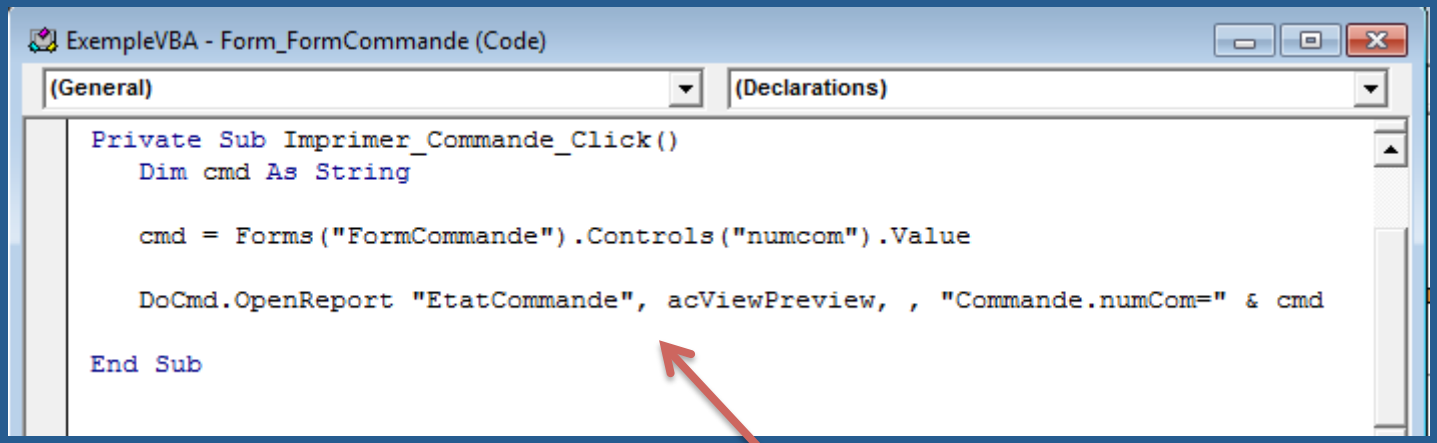


Possibilité d'impression

Calcul total de la commande, effectué concrètement dans le ModuleGlobal

Exemple VBA

- **Code VBA pour FormCommande**



Pour imprimer, on va utiliser un état capable d'afficher la commande

Exemple VBA

- **Code VBA pour FormCommande**