Visual Basic.Net

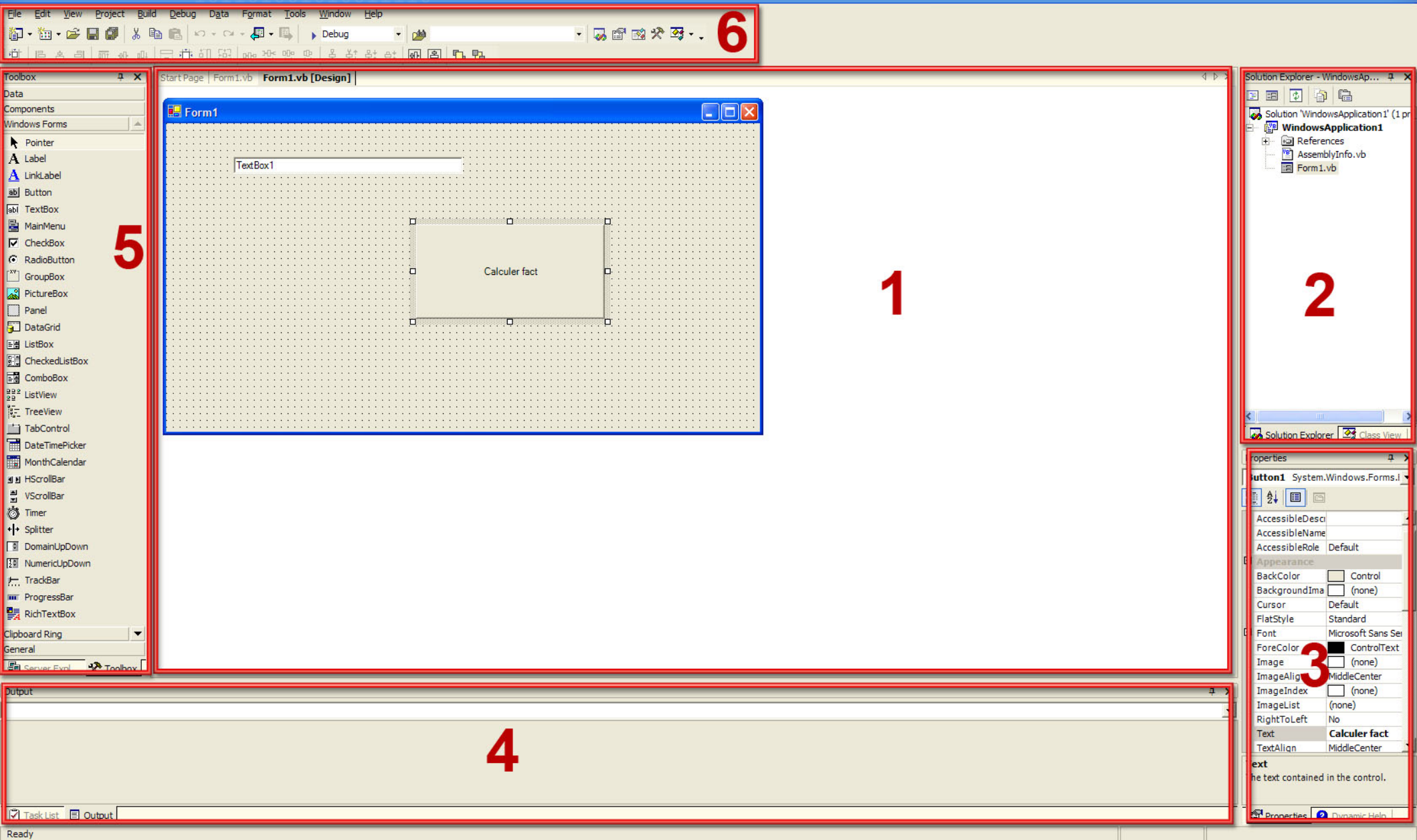
EISTI – Ing 2 IFI

Olivier Favre

Charles Villette

Sommaire

Visual Studio 2003
L'EDI



Visual Studio 2003
L'EDI

- 1 – Zone Visuel + code
- 2 – Votre projet – La hiérarchie des classes
- 3 – Propriétés des objets
- 4 – Output : débug et code d'erreurs
- 5 – Toolbox (uniquement en mode dessin)
- 6 – Boite générale : Tout est accessible depuis cette boite.

Démarrage rapide : Exemple !

- Création d'un nouveau projet
- Double clic sur la fenêtre → du code !

Syntaxe : Variables

- Règles habituelles :
  - commence par une lettre
  - Pas d'espaces ni caractères spéciaux sauf _
- `Dim` *`NomDeVariable`* `As` *`Type`* = *`affectation facultative`*
- `NomDeVariable = `*`valeur`*
- `NomDeVariable = Nothing`

Syntaxe :
Fonctions et procédures

- Différence : une fonction renvoie une valeur
- `Public|Private|Protected Sub Nom(param As Type)`
  ```
  ' Code
  End Sub
  ```
- `Public|... Function Nom(Optional param As Type = defaut) As TypeDeRetour`
  ```
  ' Code
  End Function
  ```

Syntaxe :
Valeur de retour

- ```
  Public Successeur(n As Integer) As Integer
      Successeur = n+1
      Successeur = Successeur * 10
      Exit Function ' ou Return, sans valeur
      ' Commandes non éxecutées
  End Function
  ```
- ```
  Public Successeur(n As Integer) As Integer
      Return 10*(n+1)
  End Function
  ```

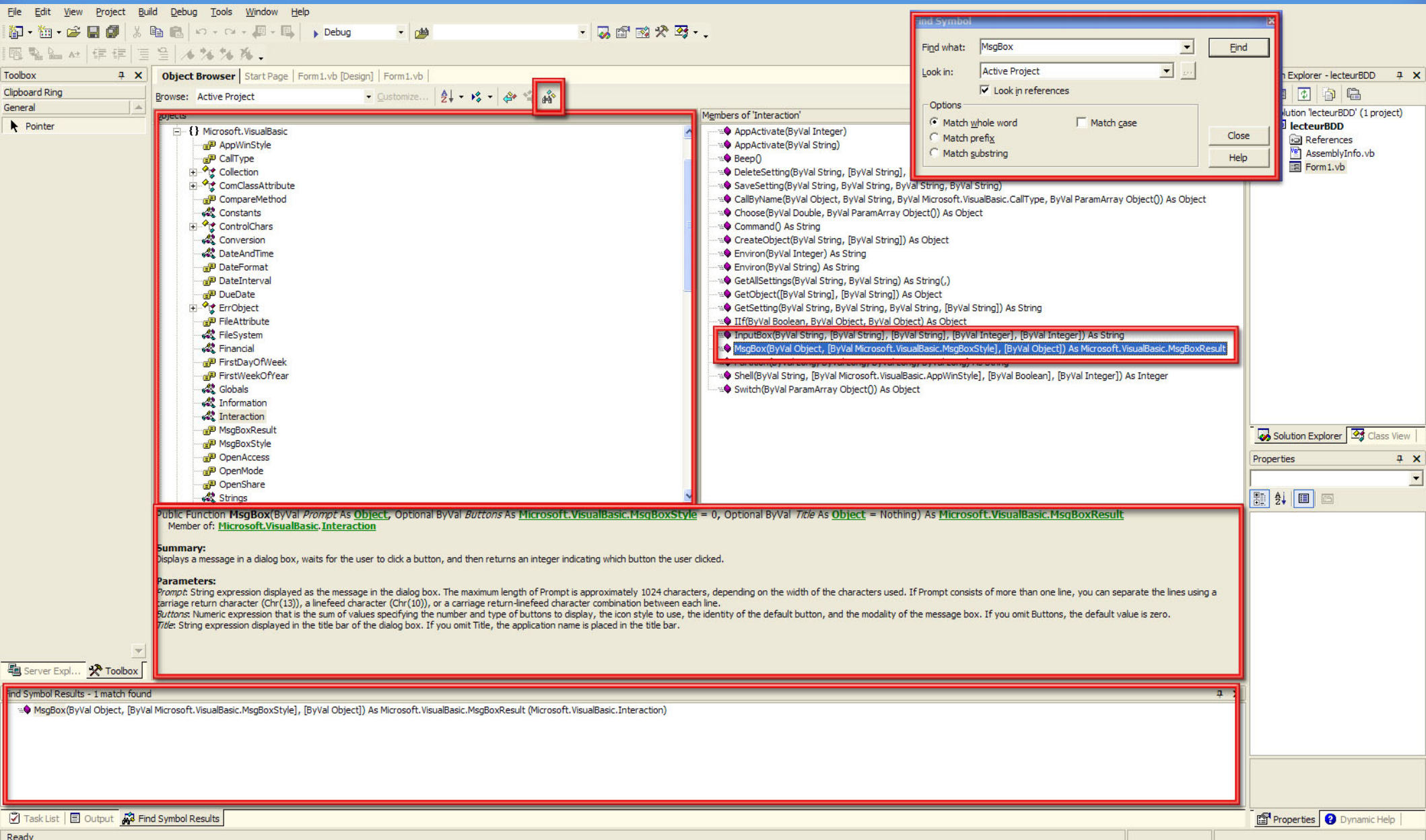
Un peu d'IHM

- Dépôt d'un bouton
- Changement d'une propriété
- Double clic → un événement « `Click` »

Se débrouiller tout seul, déjà !

- IntelliSense
  - Ctrl+Space : Auto-complétion
  - Ctrl+Shift+Space
- L'Explorateur d'Objets

Onglet Object Explorer (Ctrl+Alt+J)



Gestion d'erreurs : On Error

- `On Error Resume Next`
- `On Error Goto EtiquetteGestionErreur`

```
' Code
Exit Function|Sub ' à ne pas oublier !
EtiquetteGestionErreur:
If MsgBox("Erreur n°" & Err.Number & " : "
& Err.Description, MsgBoxStyle.Critical +
MsgBoxStyle.AbortRetryIgnore, "Erreur !")
= MsgBoxResult.Ignore Then
    Resume Next
End If
```

Gestion d'erreurs : Try – Catch, Throw

- Plus rapide que `On Error`, mais pas utilisables dans une même méthode
-
```
Try
    y = 1 / x ' attention à la division par 0
Catch ex As ArithmeticException
    MsgBox("Erreur : " & ex.Message())
Catch ex As Exception
    MsgBox("Erreur Inconnue : " & ex.Message())
Finally
    ' Code toujours exécuté !
End Try
```
-
```
Throw New Exception("Mon exception !")
```

Éléments d'IHM

- Button
- Label
- TextBox
- Menu
- CheckBox, RadioButton
- ListBox
- ProgressBar
- ...

Se débrouiller seul bis : Le Débogage

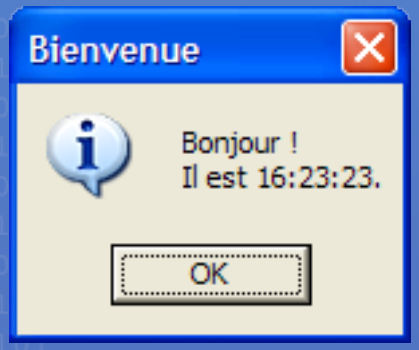
- Breakpoints
  - Définition, Désactivation, Conditions
- Watches
  - Auto
  - Locals
  - Watch
- Call Stack

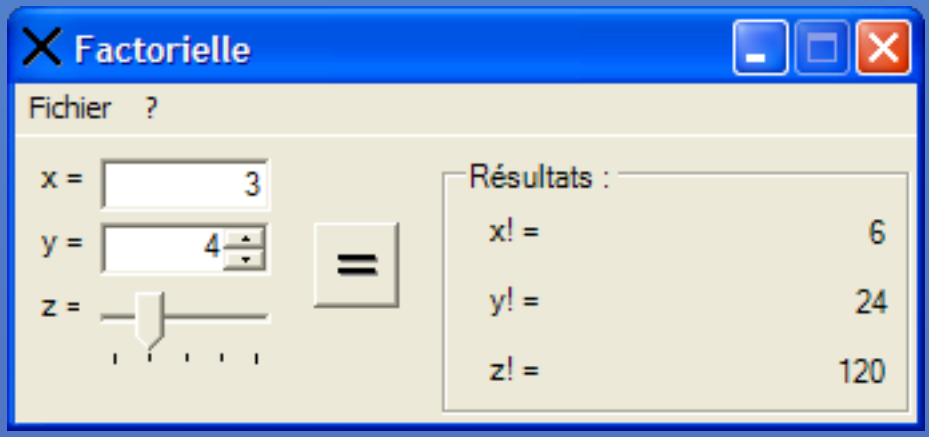
Exercice 1

- Calculatrice de factorielles
  - `TextBox`, `NumericSpinner`, `TrackBar`
  - `Button` de calcul
  - `Labels` d'affichage
  - `MsgBox` de bienvenue avec l'heure
  - `Menu` Fichier/Quitter (Ctrl+Q) et ?/À propos (F1)
  - Tester avec « toto » et « 100 »
  - Gestion des erreurs

Bonus :

- Icône dans un `Button`
- Icône de fenêtre
- Icône d'exécutable





Syntaxe : Tableaux

- `Dim MonTableau(n) As Type`
- `' De 0 à MonTableau.Length-1`
- `Dim DimensionImpliciteObligatoirement() As Type = { liste, de, valeurs, facultatives }`
- `MonTableau(x) = MonTableau(y)`
- `Redim [Preserve] MonTableau(x)`
- `Erase MonTableau`
- `Dim MultiDimensionnel(n1, n2) As Type`

Encore un peu de syntaxe : While, Do – Loop

- While *Condition*
  ```
  ' Code
  Exit While
  ' Non exécuté !
  ```
  End While
- Do While|Until *Condition*
  ```
  ' Code
  Exit Do 'pour l'exemple
  ```
  Loop
- Do
  ```
  ' Code au moins exécuté une fois
  Exit Do 'pour l'exemple
  ```
  Loop While|Until *Condition*

Encore un peu de syntaxe : For, For Each

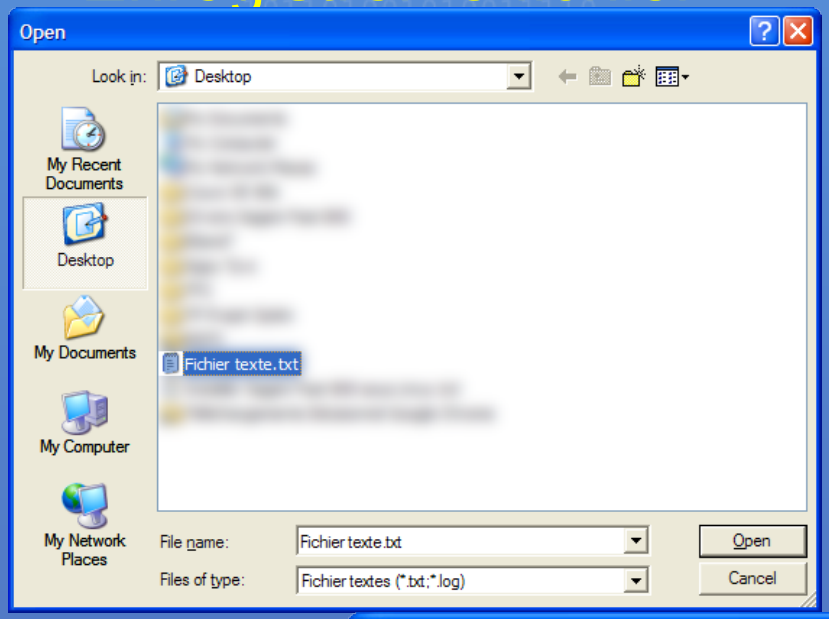
- ```
  Dim i As Integer
  For i = 0 To 10
    ' Exit For, encore pour l'exemple
  Next [i] 'on peut remettre le nom de la variable
  ```
- ```
  Dim e As TypeElement
  For Each e In CollectionDElements
    ' Exit For, idem
  Next [e] 'nom facultatif aussi
  ```

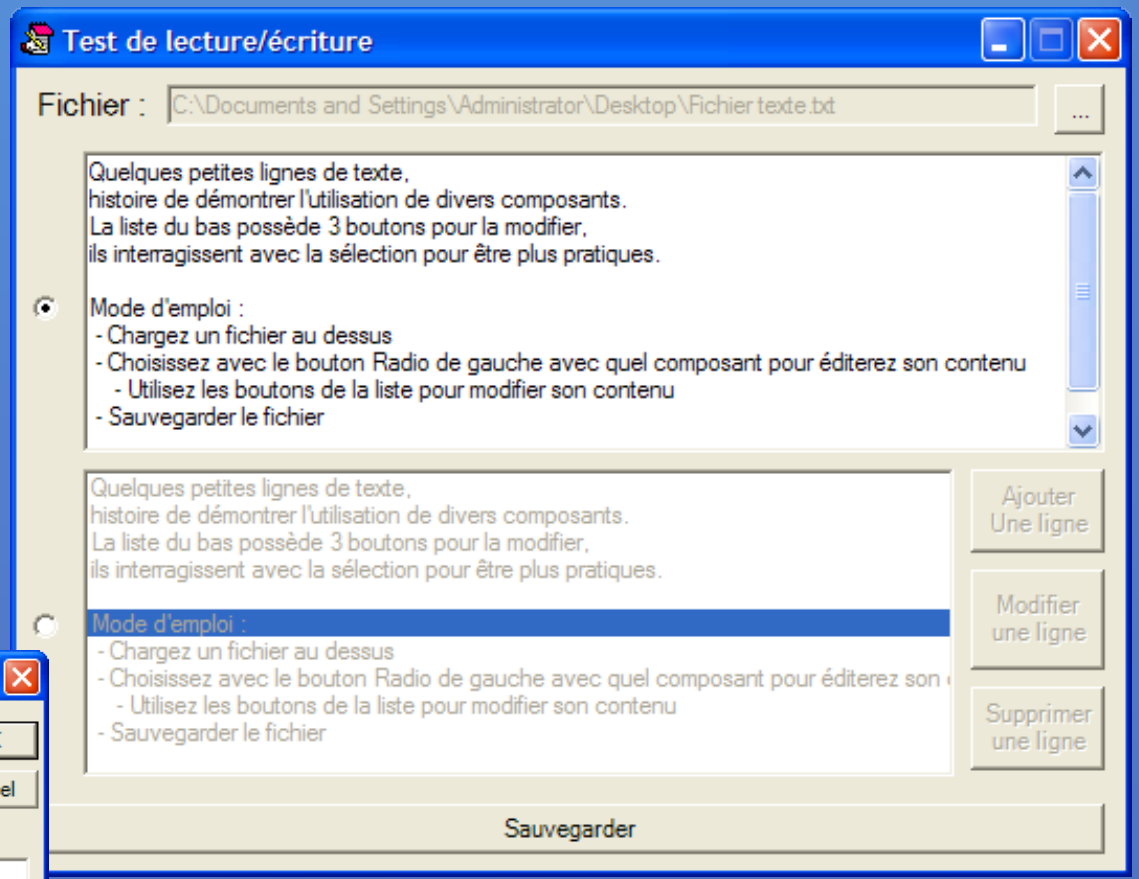
Exercice 2

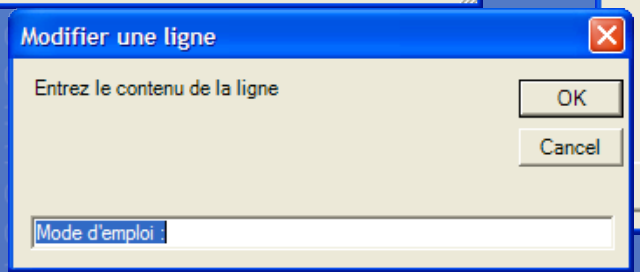
- Ouvrir un fichier
- Visualisation et modification dans un :
  - `TextBox` : Activer le MultiLine et le ScrollBars Both
  - `List` : Utiliser 3 boutons : Ajouter/Modifier/Supprimer une ligne
- Enregistrer le fichier

Bonus :

- Mettre une icône à la fenêtre et à l'application
- Faire une procédure Main
- Dossier par défaut : Desktop







Accès aux données : Connexion

- Se connecter à une BDD Access :
-
```
Dim cn As New OleDbConnection
cn.ConnectionString =
"Provider=Microsoft.Jet.OLEDB.4.0;"
cn.ConnectionString &= "DataSource=" _
                     &  "C:\maBdd.mdb"
cn.Open()
```

Accès aux données : Requête

- Lire une source de données :
-
```
Dim Reader As OleDbDataReader
Reader = New OleDbCommand( _
  "SELECT * FROM client",cn _
).ExecuteReader()
Do While Reader.Read
    MsgBox(Reader.getValue(0))
Loop
Reader.Close()
```

Accès aux données : Requête

- Modifier/Ecrire une source de données

```
New OleDbCommand("INSERT INTO Client VALUES(...)", _
cn).ExecuteNonQuery()
```

- N'ayant pas besoin de retour (hormis une erreur éventuelle), il n'y a pas plus à faire pour modifier/écrire.

Accès aux données : Fermeture

- **NE JAMAIS OUBLIER** de fermer la connexion à la base de données :
- `cn.Close()`

Exercice 3

Bonus :
- Traitement dans une procédure déléguée
- Barre d'état
- Icônes à partir des ressources

- Connexion à une base Access :
  - Connexion à une base de données depuis un fichier
  - Création d'une table
  - Ajout de données
  - Exécution d'une requête