Visual Basic : VB.NET
1ère année ENIM

Par : BENMILOUD et LEBBAR

2011-2012

Plan du cours

- Introduction au VB.NET
- Langage basic
- Les outils de création de l'interface
- Gestion d'erreurs
- Fichiers

Partie I : Introduction au VB.NET

Présentation générale du VB.NET

- Permet la réalisation des applications avec :
  - des interfaces graphiques et ergonomiques
  - peu de programmation

Plan

- Introduction
- Présentation de l'environnement
- Les objets
- La programmation événementielle
- La création d'applications en VB.NET
  - Interface
  - Code

Introduction

- Langage de programmation orientée objet
- Langage de programmation événementielle
- Langage de programmation visuelle

Environnement de VB.NET

- Barre de menus
- Boîte à outils : Toolbox
- Fenêtre de propriétés : Properties
- Explorateur de solution : Solution Explorer

Exemples d'objets de la boîte à outils

- Les **contrôles** sont des objets graphiques qui permettent de constituer des applications à interface Windows.
  - Feuilles, Boutons, Étiquettes, les combobox et les boutons radios etc
- Le formulaire est le composant de base de toute application VB.net
- Ajout d'autres contrôles par le menu **Project/Add Component**

Les objets

- Combinaison de :
  - données (propriétés)
  - code (méthodes)
- **Propriétés** : déterminent l'aspect et le comportement des objets.
  - **Objet.Propriété**
- **Méthodes** : appliquent des actions aux objets
  - **Objet.NomMéthode(liste d'arguments)**

Les propriétés

- Chaque objet de VB possède différentes propriétés qui définissent son aspect
- Modification des propriétés :
  - Lors de la création (conception/designer) :
    - utiliser la fenêtre propriétés
  - Lors de l'exécution :
    - nom_objet.nom_propriété = valeur

Les Méthodes

- Les méthodes appliquent des actions aux objets.
- Pour agir sur l'objet (appel de la méthode) :
  - Nom_Objet**.**Nom_Méthode()
- Exemple :
  - Form1.show()
  - Form1.Hide()
- Indication :
  - Pour chaque application, on peut définir quel formulaire s'ouvrira **le premier**:
    - Project/NomProjet properties/ onglet application
    - Sélectionner dans la liste : Startup form, la feuille de démarrage

Programmation événementielle

- L'exécution ne suit pas un ordre prédéfini contrairement à la programmation traditionnelle comme le C ou Pascal
- La procédure événementielle porte le nom du contrôle associé au nom de l'événement
  - Button1_Click() : appel de cette procédure lors du clic sur le contrôle Button1
  - Form1_load() : appel de cette procédure lors du chargement du formulaire 1
  - Text_box1_KeyUp() : survient quand on relâche la touche

Création d'une application en VB.NET

- Elle comprend 2 étapes
  - Conception de l'interface utilisateur de l'application
  - Écriture du code pouvant réagir aux actions effectuées à partir de l 'interface utilisateur

Éditeur de code

- Un éditeur syntaxique :
  - détecte les instructions erronées au fur et à mesure qu'on les entre (souligné en bleu)
- Bleu correspond aux mots clés
- Vert correspond aux commentaires (toute ligne commençant par un guillemet simple)
- On l'active avec la touche F7

Projet VB.NET

- Un ensemble de fichiers définissant une application
  - Les formulaires (ou windows form) :
    - MyForm.vb
    - MyForm.designer.vb
  - Les modules de code ou de classes : .vb
  - Le fichier général de description du projet : (.vbproj et .sln)

Sauvegarde / Exécution

- Sauvegarde
  - Fichiers .vb
  - Fichiers .designer.vb
  - Fichier .vbproj
  - Fichier .sln
- Important : crée automatiquement le dossier contenant l'application
- Compilation : Build
- Exécution : Start debugging ou F5

Partie II : Langage basic

Plan

- Introduction
- Les variables / constantes
- Dialogue avec l'utilisateur : input output
- Les structures de contrôles
- Les fonctions et procédures
- Portée des variables
- Les fonctions prédéfinies

Variables

- Une variable possède :
  - Un identificateur (Nom)
  - Un type
- Déclaration de variable :
  - **Dim** <Nom variable> **As** <Type>
  - Dim Taux as Single
  - Dim Réponse as String

Types de variables

- **Boolean** = true ou false
- **Integer** : de –32768 à 32767
- **Long** : -2147483648 à 2147483647
- **Single** : décimaux en simple précision
- **Double** : décimaux en double précision
- **String** : jusqu'à 65000 caractères
- Date : 1er Janvier 100 au 31 décembre 9999

Déclarations des constantes

- Const <Nom constante> = Expression
- Exemple : **Const** PI = 3.14

- VB.net Comporte des constantes prédéfinies.

Tableaux

- Dim NomTab(borneInf to BorneSup) As Type
  - Dim Semaine (1 to 7) as String
  - Semaine(1) = " Lundi "
- Dim NomTab(BorneSup) as Type
  - Dim Adresses(50) as String
    - Par défaut, l'indice minimum sera 0

Boîte de dialogue d'affichage

- Permet d'afficher un message destinés à l'utilisateur.
- Synatxes :
  - **Rep = MessageBox.**Show**(message,Caption,type_Button)**
  - **MessageBox.**Show**(message,Caption,type_Button)**
    - Show : Methode de la classe **MessageBox** permet d'afficher le message
    - Message : message que l'on veut écrire il peut être du texte ou du contenu d'une variable (Prompt)
    - Caption : le texte s'affichant dans la barre de titre (Caption)
    - Type_Button : Indique quel type de boîte va s'afficher (Style)
    - Rep : la valeur renvoyée de type **DialogResult** qui indique le numéro du bouton choisi par l'utilisateur.

Exemples de types de boîtes d'affichage

- MessageBoxButtons.AbortRetryIgnore :
  - Abandonner, Recommencer et Ignorer
- MessageBoxButtons.YesNoCancel :
  - Oui, Non et Annuler
- MessageBoxButtons.YesNo :
  - Oui et Non
- MessageBoxButtons.RetryCancel :
  - Recommencer et Annuler
- … etc

Valeurs de retour du click - boîtes d'affichage

- Windows.Forms.DialogResult.OK : 1
- Windows.Forms.DialogResult.Cancel : 2
- Windows.Forms.DialogResult.Abort : 3
- Windows.Forms.DialogResult.Retry : 4
- Windows.Forms.DialogResult.Ignore : 5
- Windows.Forms.DialogResult.Yes : 6
- Windows.Forms.DialogResult.No : 7

Boîte de saisie

- Permet de saisir une information par le biais d'une boîte de dialogue
- Syntaxe :
  - Réponse = InputBox(Invite,Titre,Valeur,Xpos,Ypos)
  - Invite = texte à afficher pour guider l'utilisateur (Prompt)
  - Titre = titre de la fenêtre (Title)
  - Valeur = valeur par défaut affiché dans la zone de saisie (DefaultResponse)
  - Réponse= contient la valeur saisie par l'utilisateur de type String

Opérateurs

- **Opérateurs arithmétiques :**
  - Opérateurs permettant d'effectuer des calculs mathématiques.
- **Opérateurs de comparaison :**
  - Opérateurs permettant d'effectuer des comparaisons.
- **Opérateurs de concaténation :**
  - Opérateurs permettant de combiner des chaînes.
- **Opérateurs logiques :**
  - Opérateurs permettant d'effectuer des opérations logiques.

Opérateurs arithmétiques

- **+**: addition classique
- **-** : soustraction
- ***** : multiplication
- ***/*** : diviser deux nombres et renvoie en résultat un nombre à ***virgule flottante*** (Double)
- **** : Diviser deux nombres et renvoie un nombre ***entier***
- **^** : Permet d'élever un nombre à une puissance.
- **Mod** : Permet de diviser deux nombres en ne renvoyant que le reste

Opérateurs de comparaison

- **<** : inférieur à**,**
- **<=** : inférieur ou égal à
- **>** : supérieur à
- **>=** : supérieur ou égal à
- **=** : égal à
- **<>** : différent de
- **Exemple:** ***prix_vente*** > 120
  - Peut être évalué à vrai **(True)** ou à faux **(False)** selon que la valeur de la variable *prix_vente* est supérieure à 120

Opérateurs de concaténation

- L'opérateur & permet de réunir à la suite des chaînes de caractères
- Si on l'utilise avec une chaîne et un nombre, le nombre est automatiquement converti en chaîne
- Exemple

```
MsgBox ("nom Faculté : "  & nomFaculté)
Str1 = nomFaculté & " "  & adresseFaculté
```

Opérateurs logiques

- **And**, **Or**, **Not**, **Xor**
- Condition composée :
  - une expression composée de conditions simples reliés par des opérateurs logiques.
- Exemple:
  - (*prix_vente* > 120) **Or Not** *bon_état*
  - Type de *prix_vente, bon_état ??*
- Priorité des opérateurs
  - **Not** > **And** > **Or**

Les structures de contrôle

- Branchement Conditionnel
- Choix multiple

Branchement Conditionnel

- If Condition Then
  [Instructions]
  End If
- If Condition Then
  [Instructions]
  Else
  [Instructions]
  End if
- **If** <condition1> **Then**
  <instructions 1>
  ElseIf <condition2> **Then**
  <instructions 2>
  Else
  <instructions 3>
  End If

Choix Multiple

- Select Case Expression
  Case Listevaleurs1
  [Instructions]
  Case Listevaleurs2
  [Instructions]
  Case Else
  [Instructions]
  End Select
- Listevaleurs peut être :
  - Une suite de valeurs : 1, 3,5,7
  - Une fourchette : 0 To 9
  - Une plage : **Is**>=10

Boucle For

- For Compteur = Début To Fin Step Pas
  [instructions]
  Next Compteur

- Par défaut, Pas = 1

Do While

- Do While Condition
  Instructions
  Loop
  - La condition est testée au début
- Do
  Instructions
  Loop While Condition
  - La condition est testée à la fin de la boucle

Do Until

- Do Until Condition
  Instructions
  Loop
  - La condition est testée au début
- Do
  Instructions
  Loop Until Condition
  - La condition est testée à la fin de la boucle

For Each Next

- For Each Élément In Ensemble
  Instructions
  Next Élément

Les enregistrements : Structure

- Permet de regrouper des données différentes.
- Exemple de définition d'un modèle de structure

```
Public Structure Eleve
        Dim ID As Integer
        Dim Nom As String
        Dim Prenom As String
        Dim Note As Double
End Structure
```

- Elle doit être déclarée dans la partie déclarations d'un module ou d'une classe et non pas dans une procédure

Structure (suite)

- Déclaration d'une variable de type Eleve
  - Dim Mon_eleve as Eleve
- Accès aux champs d'une structure
  - Mon_eleve.ID = 1238

ou

```
With Mon_eleve
    .ID = 1238
    .Nom =    "Tintin"
    .Prenom =   " Milou"
    .Note = 15
End With
```

Structure (suite)

- Déclaration d'un tableau de type Eleve
  - Dim ClasseA(100) as Eleve
- Accès aux champs des éléments du tableau
  - With ClasseA(0)

```
    .ID = 1238
    .Nom = "Dupond"
    .Prenom = "Milou"
    .Note = 15
End With
```

Procédures

- Ne retourne pas de valeur
- Synataxe :

Sub Nom_procédure([BYVAL] variable as Type,[BYREF] variable As Type)

déclarations

instructions

End sub

BYVAL : paramètre est passé par valeur

BYREF : paramètre est passé par adresse ou référence

Fonctions

- La fonction renvoie une valeur utilisable
- Syntaxe :

Function Nom_fonction([BYVAL] variable as Type,[BYREF] variable As Type)
déclarations
instructions
Return variable_calculée
End Function

} Nom_fonction =variable_calculée

Appel des procédures et fonctions

- L ’appel se fait simplement par son nom suivi de la liste des paramètres effectifs
  - Nom_procedure (valeur1,valeur2,..)
  - Variable = Nom_fonction(valeur1,valeur2,..)

Portée des variables

- La portée des variables diffère selon le lieu de déclaration.
- On distingue trois possibilités :
  - Niveau Procédure
  - Niveau Formulaire
  - Niveau module

Niveau Procédure

- La variable est locale.
- Elle est valide que pour cette procédure.
- Elle est déclarée à l'intérieur de la procédure par la syntaxe suivante :
  - Dim <Nom variable > as <Type>

Niveau Formulaire

- Les variables déclarées dans la section *Déclarations* d'une feuille peuvent être **Private** ou **Public**
- **Private** : disponible pour toutes les procédures de la feuille, et ce par la syntaxe suivante :
  - Private <Nom variable> as <Type>
- **Public** : utilisables dans toute l'application, et ce par la syntaxe suivante :
  - Public <Nom variable> as <Type>
  - Il faut spécifier le nom de la feuille si on veut utiliser la variable en dehors de la feuille. Exemple :

    nom_formulaire.nom_variable

Niveau module

- Les variables déclarées dans la section *Déclarations* du module peuvent être **Private** ou **Public**
- Déclarations:
  - Public <Nom variable> as <Type>
    - Disponible dans toute l'application
  - Private <Nom variable> as <Type>
    - Disponible uniquement pour les procédures du module.

Portées des procédures et Fonctions

- **Private** : signifie que la fonction [procédure] est utilisable par toutes les procédures et les fonctions du même module. (lieu de sa définition).

- **Public** : signifie que la fonction [procédure] peut être appelée par toutes les procédures et les fonctions du projet.

Type String

- Les méthodes de l'objet String
  - ToUpper:
    - Mettre en majuscules une chaîne de caractères
  - ToLOwer :
    - Mettre en minuscule
  - Trim :
    - Permet de supprimer des caractères avant et après la chaîne de caractères
  - Length :
    - Taille d'une chaîne en nombre de caractères
  - Concat :
    - Mise bout à bout

Type String : Exemples

- Dim Message as string
- Message = " bonjour "
- Message = Message.ToUpper
  - => Message = " BONJOUR "
- Message=Message.Trim(" ")
  - => Message ="BONJOUR"
- Dim A,B,C as string
- A ="Dupond "
- B = "Anne"
- C = string.concat(A,B)
  - => C = "Dupond Anne"

Les fonctions mathématiques

- Dim R as double
- R = Math.Min(2,3) => R = 2
- R = Math.Max(2,3) => R = 3
- R = Math.Pow(2,3) => R = 8
- R = Math.sqrt(9) => R = 3
- R = Math.abs(-7) => R = 7

Autres fonctions

- IsNumeric : permet de déterminer si le contenu d'une variable est un nombre
- Exemple :
  - Dim MyVar as Object
  - Dim R as Boolean
  - MyVar ="45"
  - R = IsNumeric (MyVar) => R = True
  - MyVar = "45 kg"
  - R = IsNumeric(MyVar) => R = False
- Val() : renvoie un nombre si chaîne est composé de chiffres

Partie III: Outils de Création de l'interface

PLAN

- Les contrôles : objets visuels prédéfinis
  - Formulaire
  - Bouton de commande
  - Labels
  - Zones de texte
  - Cases à cocher
  - Listes
  - Création de menu
  - Grilles
- Gestion des erreurs

Rappels

- Un objet
- Une propriété
- Une méthode
- Une procédure événementielle

La procédure événementielle

- Appel de la procédure évènementielle :

  Private sub NomObjet_NomEvenement(Arguments)

  Instructions

  End Sub

Exemple :

- **Private Sub** BtOk_Click (ByVal **sender** As System.Objet, ByVal **e** As System.EventArgs) **Handles** BtOk.Click

  End Sub

Suite de Procédure événementielle

- **Private Sub :** La procédure évènement est **privée** donc accessible uniquement dans le module.

- **BtOk_Click :** Après le mot Sub il y a **un nom de procédure**. Ce nom est construit avec le nom du contrôle et l'évènement.
- **Sender :** indique le contrôle ayant déclenché l'évènement. c'est un Objet.

  sender.Name contient par exemple le nom du contrôle ayant déclenché l'évènement.

Suite de la procédure événementielle

- **e :** C'est une variable de type SystemEventArgs qui permet de connaître l'évènement qui à déclenché la procédure.

- **Handles** : Indique quel objet et évènement à déclenché la procédure.
- Handles BtOk.Click indique que c'est l'évènement Click sur l'objet BtOk qui déclenche la procédure.

Formulaires (ou feuilles)

- Le composant de base de toute application VB
- Une feuille principale Form1
- On peut ajouter d'autres fenêtres par :
  - Menu Project : Ajouter un formulaire (Add Windows Form)
  - Form1 est une fenêtre crée avec la classe WindowsForms.Form
- Suppression de feuille par Edit/Delete ou par menu contextuel
- Enregistrement de la feuille avec l'extension Form1.vb par le menu File/Save as

Propriétés du Formulaire

- Les propriétés essentielles concernant la feuille :
  - **Name** est commune à tous les objets, il sert à référencer l'objet dans le code. Accessible en mode conception uniquement.
  - **Text** : le texte associé à l'objet sur l'écran. C'est le texte qui apparaît à la barre de titre en haut. Accessible en mode conception ou dans le code.

Propriétés du Formulaire

- **Visible** : gère le caractère visible de l'objet
  - frmAcceuil.visible = false : fait disparaître le formulaire nommé frmAcceuil ⇔ frmAcceuil.hide()
- **WindowState** : donne l'état de la fenêtre
  - Plein écran : FormWindowState.Maximized
  - Normal : FormWindowState.Normal
  - Dans la barre des tâches : FormWindowState. Minimized

Propriétés du Formulaire (suite)

- **ControlBox** : gère la présence (True) ou non du menu système
- FormBorderStyle : permet de choisir le type des bords de la fenêtre
  - Me.FormBorderStyle = Windows.Forms.FormBorderStyle.Sizable
- Pour changer une propriété du formulaire courant / On utilise la syntaxe :
  - Me.Nom_propriété = valeur

Méthodes du Formulaire

- Exemples :
  - Show : charge la feuille en mémoire et la rend visible à l'écran :
    - **Form1.Show()**
  - Hide : rend la feuille invisible
    - **Form1.Hide()**

Procédures événementielles du Formulaire

- Exemples d'évènements :
  - **Load** (chargement) : cette procédure contient tout ce que l'application doit faire au moment où le formulaire en question doit se charger en mémoire vive.
  - **MouseDown**
  - **MouseMove**

Bouton de commande

- Le contrôle CommandButton
- Les propriétés essentielles :
  - **Name** : Nom de l'objet utilisé dans le code
  - **Text** : ce qui s'affiche sur le bouton de commande (Ok, Annuler ..)
  - **Enabled** : actif pour True et inactif pour False
  - **FlatStyle** : pour changer l'aspect du bouton
- L'événement principal est le **Click**

Label

- Contrôle « inerte » qui sert à afficher un texte sur une feuille (étiquette)
- L'utilisateur ne peut pas saisir du texte sur ce contrôle.
- Quelques propriétés : Name, Text, Alignement et BorderStyle Autosize, etc.

TextBox

- C’est une zone qui sert à afficher une information saisie au clavier par l’utilisateur
- Les zones de texte sont de type texte
- Pour convertir en valeur numérique on utilise la fonction :
  - **Dim** X **as double**
  - X = **Val**(textbox1.text)

TextBox : propriétés

- Quelques propriétés :
  - **Text** : désigne son contenu
  - **Multiline** : autorise ou non l'écriture sur plusieurs lignes
  - **Scrollbar**, **Passwordchar** et **Maxlength**

Quelques événements

- **Keypress : survient quand la touche est enfoncée**
  - Permet de récupérer la touche pressée par e.KeyChar (sauf F1, F2 ..)
- **KeyUp : survient quand on relâche la touche**
- **KeyDown :survient quand on appuie sur une touche**
  - Permettent de savoir si ALT, CTL on été pressée

Comment interdire la frappe de certains caractères dans un TextBox ?

- On utilise KeyPress, il retourne un objet **e**
- **E.KeyChar :** contient la caractère pressé mais il est en lecture seule.
- Pour annuler la frappe : **e.Handled = True**
- Exemple pour écrire que des valeurs numériques :

```
If IsNumeric(e.KeyChar) Then
        e.Handled = False
    Else
        e.Handled = True
    End If
```

Contrôles Cases

- Cases à cocher (**CheckBox**) : permettent des choix indépendants même s'ils sont regroupés dans un même cadre (GroupBox)

- Cases d'options (**RadioButton**) : permettent de choisir une seule option parmi un groupe de choix exclusif

Contrôles Cases (Suite)

- Propriété :
  - **Checked** : permet de savoir si la case est cochée
- Événement :
  - **CheckedChange**() : permet d'intercepter le changement d'état du bouton ou case

Utiliser un GroupBox

- Pour cocher toutes les cases

```
Dim x As CheckBox
        For Each x In GroupBox1.Controls
            x.Checked = True
        Next
```

- Pour décocher toutes les cases

```
Dim x As CheckBox
        For Each x In GroupBox1.Controls
            x.Checked = False
        Next
```

Contrôles Listes

- **ListBox** : non modifiable et non déroulante
- **ComboBox** : modifiable et déroulante

Propriétés

- ListBox1.Items.Count : indique le nombre d'éléments dans la liste
- Le 1er élément de la liste est d'index 0

Méthodes de ListBox

- **Pour ajouter dans une liste :**
  - Listbox1.items.add("Rabat")
  - ListBox1.items.insert(4,"Safi") 'ajoute à la 4ème position
- **Pour supprimer des éléments d'une liste :**
  - ListBox1.items.removeat(5) :Enlève l'élément d'index 5
  - ListBox1.items.remove("Salé") :Enlève 'Salé' de la liste
  - Listbox1.items.remove(listbox1.selecteditem) : Enlève l'élément sélectionné
- **Pour vider une liste**
  - ListBox1.items.clear

Création de menu : Objet MainMenu

- La commande Tools/Choose ToolBox/ Items
- Pour chaque Commande de menu, on utilise sa fenêtre de propriétés
  - Enabled, Checked, ShortCut
- L'événement essentiel : Click

L'objet : Grilles

- Ajouter un objet de type : DataGridView
- Changer sa propriété Name : "Grille"
- Pour mettre 3 colonnes et 20 lignes dans la grille :
  - Grille.RowCount = 20
  - Grille.ColumnCount = 3

Grilles : suite

- Grille.Item(i, j).Value = 12
  - Met la valeur 12 à la colonne i et ligne j
- Pour nommer les en têtes des colonnes :
  - .Columns(0).Name = "Nom"
  - .Columns(1).Name = "Prénom"
- .RowHeadersVisible = False : pas de première colonne d'en tête des lignes
- .Columns(2).Width = 30 : dimensionner la largeur de la colonne 2

Grilles : suite

- Empêche les modifications de lignes, colonnes, l'ajout, la suppression :
  - .AllowUserToAddRows = False
  - .AllowUserToDeleteRows = False
  - .AllowUserToOrderColumns = False
  - .AllowUserToResizeColumns = False
  - .AllowUserToResizeRows = False

Grilles : suite

- Une ligne sur 2 en bleue
  - .RowsDefaultCellStyle.BackColor = Color.White
  - .AlternatingRowsDefaultCellStyle.BackColor = Color.AliceBlue
- Interdire la sélection de plusieurs cellules
  - .MultiSelect = False

Partie IV :
- Gestion d'erreurs
- Gestion des fichiers

Gestion des erreurs

- Erreurs de syntaxe :
  - soulignés en ondulé bleu le code lors de la validation
- Erreurs de logique :
  - à détecter par le mode d'exécution pas à pas
- Erreurs d'exécution :
  - surviennent en mode Run dans l'IDE ou lors de l'utilisation de l'exécutable

Erreurs de logique

- **Outils de débogage :**
  - ✓ Points d'arrêt
  - ✓ Options d'exécution pas à pas
  - ✓ Fenêtre **Exécution : debug.print**

Exemples d'erreurs d'exécution

- Soit une erreur de conception. Exemples :
  - Ouvrir un fichier qui n'existe pas (On aurait du vérifier qu'il existe avant de l'ouvrir!).
  - Division par zéro.
  - Utiliser un index d'élément de tableau supérieur au nombre d'élément.

- Soit une erreur de l'utilisateur. Exemples :
  - On lui demande de taper un chiffre, il tape une lettre ou rien puis valide ou vice versa

Erreurs d'exécution

```
Try
    'Instruction susceptible de provoquer une erreur.

Catch ex as exception
    'Traitement de l'erreur

End Try
```

L'objet '**Exception**'

- Généré par l'erreur.
- Catch ex As Exception
- Cet objet Exception à des propriétés:
  - **Message** qui contient le descriptif de l'erreur.
  - **Source** qui contient l'objet qui a provoqué l'erreur....
  - ex.Message contient donc le message de l'erreur.

Gestion de fichiers

- Ouverture du fichier
  - En lecture
  - En écriture
- Fonctions de lecture et écriture dans un fichier
- Fermeture du fichier
- Les boites de dialogues prédéfinis

Ouverture d'un fichier séquentiel (texte)

- FileOpen(numéro, nom du fichier, mode)
  - **numéro** est un entier compris entre 1 et 255 **nom du fichier** est une chaîne de caractères
  - **mode** est une valeur choisie parmi **OpenMode.Input** (ouverture en lecture), **OpenMode.Output** (ouverture en écriture) ou **OpenMode.Append** (ouverture en ajout)

Lecture / Ecriture pour un fichier séquentiel

- LineInput (**NuméroFichier)** : Lit une ligne en entrée depuis le fichier texte
- PrintLine (**NuméroFichier, texte) : Écrit le contenu de la variable texte dans le fichier**

Fichiers séquentiels : généralités

- **EOF (NuméroFichier) :**
  - **'End Of File',** (Fin de Fichier) il prend la valeur True si on est à la fin du fichier et qu'il n'y a plus rien à lire.
- **LOF (NuméroFichier)** :
  - '**Lenght Of File**', il retourne la longueur du fichier en octets.
- FileClose(**NuméroFichier)** :
  - Permet de fermer le fichier

Objet : OpenFileDialog

- ShowDialog () : permet d'ouvrir la fenêtre d'explorateur des fichiers de windows
- Exemple de code :

```
With OpenFileDialog1
     .Title= "Ouvrir"                 'Titre de la barre de titre
    .InitialDirectory = "c:\"         'pour choisir le répertoire de départ
    .Filter="Fichiers txt|*.txt"      'pour limiter les types de fichiers .txt
                                      's'il y a plusieurs filtres les séparer par ;
                                      'FilterIndex  indique le filtre en cours
    .Multiselect=False                'limite la sélection à 1 seul fichier
End With
```

- Filename() : permet de récupérer le nom du fichier sélectionné

Objet : SaveFileDialog

- ShowDialog () : permet d'ouvrir la fenêtre de l'explorateur des fichiers de type sauvegarde
- On utilise les même propriétés que l'objet OpenFileDialog

Références

- http://plasserre.developpez.com/cours/vb-net/
- http://www.siteduzero.com/tutoriel-3-134798-visual-basic-net.html