INTRODUCTION AU DEVELOPPEMENT EN VISUAL BASIC

H. TSOUNGUI, ISTV, 2017
v. 24.01.17

1. Démarche en programmation évènementielle

MS Visual Basic est un outil de développement rapide d'applications (RAD : Rapid Applications Development). Sa démarche est particulière :
– concevoir les interfaces (formulaires, grilles, etc), c-a-d dessiner les objets et préciser les **propriétés** et **méthodes** à utiliser pour chacun
– prévoir les **événements** (click sur un objet comme un bouton, perte du focus,etc)
- gérer les **événements dans le code** (appliquer les structures de contrôle)
- tester et contrôler la bonne exécution des actions (débuggage)

Un exemple
Développer une application basée sur la programmation évènementielle qui calcule le périmètre d'un rectangle, un carré ou un disque ainsi que la superficie (surface) de la figure choisie.

2. Les notions de base de la programmation « classique »

-Structure d'une procedure (sous-programme)

```
Sub MaProcedure()
        'Déclarations locales
          Dim I as integer
          Dim v(5) as string
        …
          … instructions diverses
          'appel d'autre procedure/fonction
           S = Surface(x)
        …
          … instructions diverses
        ...
End Sub
```

-Structure d'une fonction et valeur de retour

C'est en quelque sorte une procédure qui retourne une valeur. Voici un exemple :

```
Private Function SuperficieCercle( Rayon as Single) As Single
        Return 3.14*Rayon*Rayon
End Function
```

Utilisation dans un autre sous-programme (routine)

```
Dim ret as single   ' valeur de retour
Dim rayon as single
Rayon = inputbox(« Donnez le rayon »)
Ret = SuperficieCercle( Rayon)
Msgbox(« Superficie : «  & str(ret))
```

Exemples de procédures et fonctions

```
Public Class frmprocfonc
  ' Définition de la procédure addition
  Private Sub addition(ByVal x As Integer, ByVal y As Integer)
    Dim s As Integer
    s = x + y
    ' Pas de variable pour renvoyer le résultat s
     (mais c'est possible dans une variable globale)

    lblResult1.Text = Str(s)
    Me.Show()
  End Sub
```

Utilisation dans une autre procedure(appel)

```
  Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
    'Procédure ADDITION de deux valeurs
    Dim var1 As Integer : Dim var2 As Integer
    var1 = Val(txtvar1.Text) : var2 = Val(txtvar2.Text)
    addition(var1, var2)  ' Appel de la procédure On peut aussi utiliser call
addition(var1,var2)
  End Sub
```

```
  Private Function multiplication(ByVal x As Integer, ByVal y As Integer) As
Integer
    Dim p As Integer
    p = x * y
    Return p    ' Retour du résultat dans la variable "locale" p
  End Function

  Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button3.Click
    Me.Close()
  End Sub

  Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
    Dim var1 As Integer : Dim var2 As Integer : Dim k As Integer
    var1 = Val(txtvar1.Text) : var2 = Val(txtvar2.Text)

    k = multiplication(var1, var2)  ' Appel de la procédure multiplication
    'Affichage du résultat de la multiplication
    lblresult2.Text = k.ToString    ' au lieu de lblresult2.text = str(k)
  End Sub

End Class
```

-Types de données (variables et constantes)

-Constantes : données pour lesquelles on fixe une valeur ne pouvant être modifiée.
Déclaration : CONST pi as Integer = 3.14159
-Variables : leurs valeurs peuvent être modifiées au cours du programme.

3.Les variables (déclaration, types et portées)

* la déclaration de toutes les variables est obligatoire (voir options de projets).

Les types de variables

Tableau récapitulatif des possibilités offertes par VB.Net :

Type	Plage des valeurs	Taille
Short	-32 768 à 32 767	2 octets
Integer	- 2 147 483 648 à 2 147 483 647	4 octets
Long	-9 223 372 036 854 775 808 à -9 223 372 036 854 775 807 à	8 octets
Single	-3,4028235E+38 à -1,401298E-45 pour les valeurs négatives 1,401298E-45 à 3,4028235E+38 pour les valeurs positives	4 octets
Double	-1,79769313486231570E+308 à -4,94065645841246544E-324 pour les valeurs négatives -4,94065645841246544E-324 à -1,79769313486231570E+308 pour les valeurs positives	8 octets
Decimal	+/- 79 228 162 514 264 337 593 543 950 335 sans chiffre décimal +/- 7,922 816 251 426 433 759 354 395 033 5 avec 28 positions à droite de la virgule	12 octets
Byte	0 à 255	1 octets
Boolean	True / False	2 octets
String	de 0 à environ 230 caractères	
Char	1 caractère	2 octets
Date	01/01/1 à 31/12/9999	8 octets
Object	référence à un objet	4 octets + objet

On verra plus tard qu'il existe en réalité une pléthore d'autres types

Portée des variables(visibilité)

On distingue trois niveaux de portée pour les variables : procédure, form et projet

- Niveau procédure : la variable est créée à chaque exécution de cette procédure, puis détruite à chaque fois que la procédure se termine. Elle ne conserve donc sa valeur qu'à l'intérieur de cette procédure, et pour une seule exécution de cette procédure.

```
Sub Maprocedure()
  Dim toto as Integer        'cette variable est strictement locale
  … instructions …
  … instructions …
End Sub
```

- Niveau Form : la variable est créée dès que la Form est utilisée. Elle reste accessible par toutes les procédures contenues dans cette Form et conserve sa valeur tant que la Form est active en mémoire vive. Dans ce cas, il faut déclarer la variable n'importe où en-dehors d'une procédure de la Form .

```
Dim toto as Integer
Sub MaProcedure()
  ...
End Sub

Sub MonAutreProcedure()
  ...
End Sub
```

- Niveau projet : la variable est accessible à toutes les procédures du projet, quel que soit la Form, la classe ou la structure dans laquelle se trouvent lesdites procédures. Pour ce faire, la **déclaration doit être faite dans un module**, en employant le mot-clé **Public**.

```
Public toto as Integer

Public TABLO(5,4) As String
```

4. STRUCTURES DE CONTROLE

Les tests

La forme universelle du test est la suivante :

SI condition-booléenne
ALORS
Instructions …
SINON
Instructions
FIN-SI

IF *expression_booléenne* **THEN**
instructions_si_vrai
[**ELSE**
instructions_si_faux]
ENDIF

Autre expression des tests conditionnels

If *expression_booléenne* **Then**
instructions_si_vrai
Elseif *expression_booléenne* **Then**
instructions_si_vrai
Elseif *expression_booléenne* **Then**
instructions_si_vrai
Elseif *expression_booléenne* **Then**
instructions_si_vrai
etc.
Endif

Les sélections avec SELECT CASE

Quand on a un nombre important de tests à faire, il faut éviter les IF … ELSEIF … ENDIF en utilisant à la place, le commutateur SELECT CASE :

Select Case Valeur

Case condition-1 ou valeur1
Instructions (si le test est vrai)
Case condition-2
instructions
…
Case Else ‘ Autrement
instructions
End Select

Les structures répétitives et itératives (boucles)

TANT QUE

WHILE *expression_booléenne*
instructions
WEND (ou END WHILE)

POUR

FOR *variable* = *valeur_initiale* TO *valeur_finale* [step *pas*]
instructions
NEXT *variable*

Exemple d’imbrication de boucles FOR - NEXT

FOR *variable-**i*** = *valeur_initiale* to *valeur_finale* [step *pas*]
instructions
FOR *variable-**j*** = *valeur_initiale* to *valeur_finale* [step *pas*]
instructions
NEXT *variable-**j***

NEXT *variable-**i***

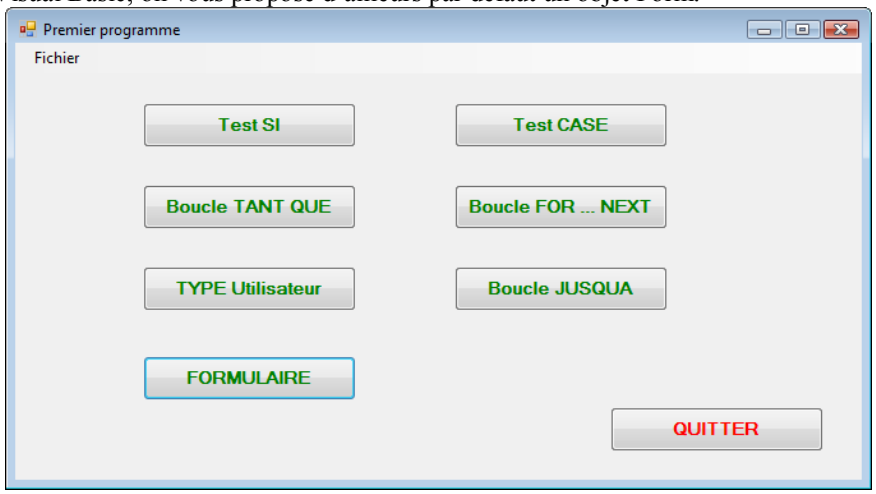
JUSQUA

DO [LOOP]
…
…
UNTIL condition booléenne

5. Les Premiers Contrôles (Boîte à outils de contrôles)

Le contrôle Form(formulaire)

C'est le contrôle de base, absolument universel, en Visual Basic est la feuille, ou formulaire, en anglais, **Form**. Cet objet est incontournable ; on ne peut créer et utiliser d'autres objets que si ceux-ci font partie d'une Form. A l'ouverture de Visual Basic, on vous propose d'ailleurs par défaut un objet Form.



Nous allons examiner deux propriétés dont il est essentiel de comprendre la signification.
Name : il s'agit du nom de l'objet tel qu'il est géré par l'application. Cela correspond en quelque sorte à un nom de variable (sauf que ce n'est pas une variable, c'est un objet !). Prendre l'habitude de bien nommer les objets form de manière significative : frmMenu pour le formulaire Menu, txtNomClient pour le champ textBox devant contenir le nom d'un client.
Ex : frmMenu, frmSuppression

Text : utile pour professionnaliser une application, lui donner un look fini.
Dans le code, on ne désigne donc jamais un objet par sa propriété Text, mais par son Name.

Visible : propriété booléenne qui gère, comme son nom l'indique, le caractère visible ou non de la Form (et partant, des autres contrôles qu'elle contient)
Procédure événementielles : à chaque objet créé peuvent correspondre autant de procédures que d'événements possibles survenant sur cet objet.

Ex : évènement sur un objet comme par ex. un click sur un bouton de commande

```
Private Sub NomObjet_Evenement()
…
End Sub
```

Private signifie que la procédure n'est utilisable (visible) que pour la Form considérée (et pas pour d'autres objets situés sur d'autres Form).

Evènement CLICK sur le bouton « FORMULAIRE »

Ex :
```
Private Sub btnFERMER_Click()
    Me.close
End Sub
```

Le contrôle CommandButton (Bouton de Commande)

Il s'agit du bouton type OK, Annuler, mais dont le texte apparent (en Anglais, propriété Text) et le rôle dans une application peuvent varier à l'infini. L'action que VB considère comme étant la plus commune pour les boutons de commande est le Click (en Français, clic).
Quelques propriétés intéressantes de la classe CommandButton :
Name : bien sûr !
Text : « FERMER » qui permet évidemment de fermer l'objet Form …
Visible : rend l'objet visible.
Enabled : cette propriété, booléenne, est comme on le verra est très commune à d'autres objets et permet de les rendre accessibles (activés). Elle permet (valeur True) à un contrôle d'être actif.

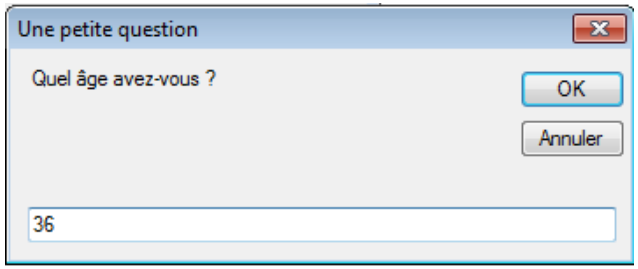
Le contrôle Listbox (Boîte de liste simple)

Très utile pour afficher des infos appelés **Items** sous forme de liste déroulante. Utilisation : **listBox1**.**Items.Add**(infos)

6. Fonctions d'interface

La fonction InputBox

La première de ces deux fonctions est celle qui nous permet de faire apparaître une petite **boîte de saisie** tout ce qu'il y a de plus ordinaire, comportant une zone dans laquelle l'utilisateur pourra entrer une valeur :



C'est une **fonction**, dont les deux arguments de type String correspondent respectivement à l'invite ("*Quel âge avez-vous ?*") et au titre ("Une petite question") de la boîte de saisie. Cette fonction renverra toujours une valeur de type **String**. Code à écrire :

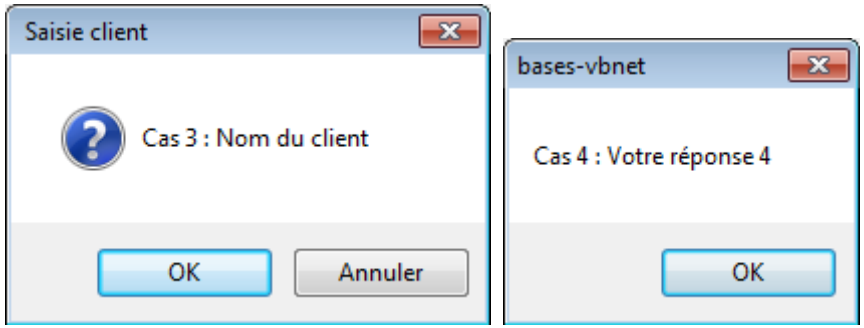
```
Dim Rep As String : Dim laquestion, Titre_fenetre As String
        laquestion = "Quel âge avez-vous ?"
        Titre_fenetre = "Une petite question"
        Rep = InputBox(laquestion, Titre_fenetre)
```

Ce code attendra que le bouton **OK** (ou **Annuler**) soit cliqué, et stockera alors le contenu de la zone de saisie dans la variable **Rep**.

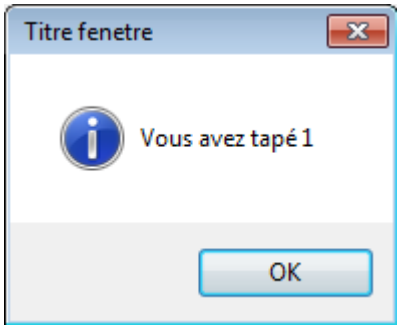
La fonction MsgBox

La fonction jumelle d'Inputbox, vouée à un usage un peu différent, est la fonction **MsgBox**. Elle déclenche l'apparition d'une **boîte SANS zone de saisie**, mais dotée d'un **jeu de boutons**, et éventuellement d'une illustration standard sous forme d'icône Point d'interrogation, exclammation, etc.

```
Dim rep1 As String = "" : Dim rep2, rep4 As Integer : Dim rep3
As Integer
        rep1 = InputBox("Cas 1 : Donnez une valeur")
        MsgBox(" Vous avez choisi " & rep1.ToString)
        rep2 = MsgBox("Cas 2 : Nom du client ", vbOKCancel)
        MsgBox(" Vous avez cliqué le bouton " & rep2.ToString)

        rep3 = MessageBox.Show("Cas 3 : Nom du client ", "
Saisie client", MessageBoxButtons.OKCancel,
MessageBoxIcon.Question)
        MsgBox(" Vous avez choisi le bouton " & rep3.ToString)

        rep4 = MsgBox("Cas 4 : Votre réponse 4 ")
        MsgBox("Vous avez tapé " & rep4.ToString,
MsgBoxStyle.Information, " Titre fenetre")
```



Là, on retrouve comme arguments l'**invite** ("*Missile thermonucléaire...*") et le **titre** ("*Dernière vérification*"). Mais entre ces deux paramètres, on doit fournir un argument d'un genre un peu spécial, le rôle de cet argument est double : il doit indiquer d'une part quel est le jeu de boutons souhaité, parmi les six disponibles ("**Abandonner, Réessayer, Ignorer**", "**OK**", "**OK, Annuler**", "**Réessayer, Annuler**", "**Oui, Non**", "**Oui, Non, Annuler**"). Il doit également stipuler quelle **illustration** (icône) viendra éventuellement égayer votre boîte à messages, parmi les quatre possibles (**Critical**, **Exclamation**, **Information**, **Question**). Mais nous les laisserons de côté provisoirement. Ce qui compte, c'est de comprendre comment ça marche.



Le Label (Etiquette)

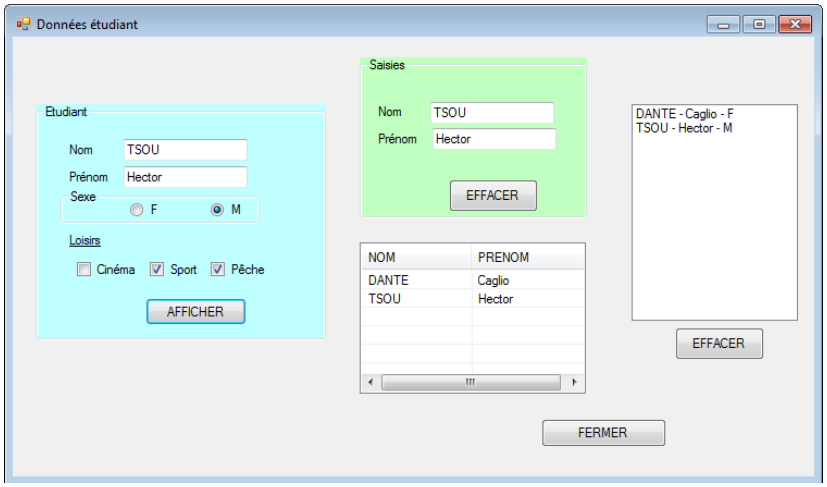
Un Label est un contrôle "inerte", qui sert à afficher un texte sur une Form.

Ce qu'il faut comprendre avec les Labels, c'est qu'ils ne peuvent jamais servir à effectuer une saisie par l'utilisateur.

La Zone de Texte (TextBox)

Ces zones (de la classe "TextBox" pour VB) peuvent servir à saisir une information. Il s'agit du seul contrôle **permettant une saisie au clavier** par l'utilisateur. En Visual Basic, il n'y a donc plus à proprement parler d'instruction **Lire**. A la place de cette instruction, on est contraint de passer par de telles zones.

La seule chose vraiment importante à savoir est que toute information contenue dans une zone de texte est obligatoirement de type… **texte** ! (autrement dit, cela inclut le cas où il s'agit d'un nombre, L'emploi de fonctions de conversion s'avèrera fréquemment indispensable
VAL(chaîne) pour convertir une chaîne en nombre
STR(numérique) pour convertir un nombre en chaîne.
On peut également utiliser les nombreuses **fonctions de conversion** prédéfinies **CSng, CStr, CDec, CInt**, etc.



Propriétés intéressantes des zones de texte
Multiline : autorise ou non l'écriture sur plusieurs lignes
Scrollbars : fait figurer dans la TextBox une barre de défilement horizontale ou verticale (ou les deux)

PasswordChar : crypte le texte entré par le caractère stipulé (généralement, on choisit le caractère *)
MaxLength : limite le nombre de caractères qu'il est possible de saisir dans la zone de texte.

Exercice

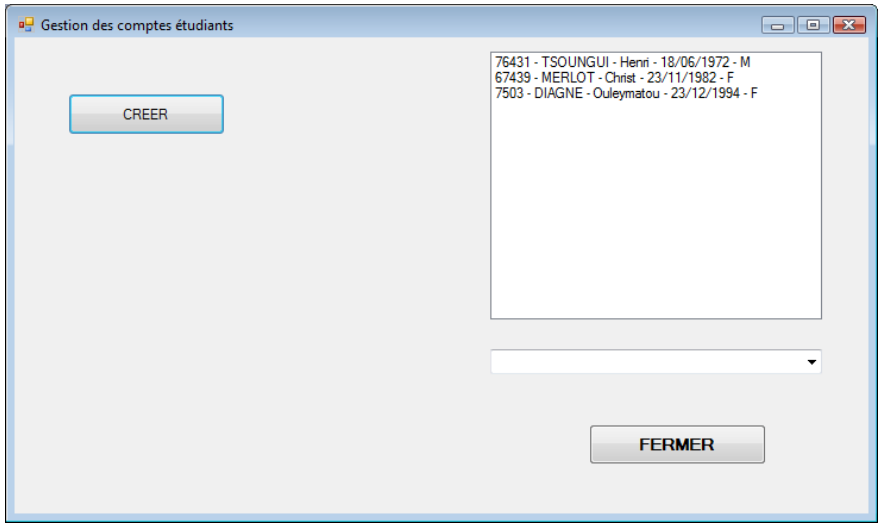
-Ecrire un programme VB permettant de saisir **n** valeurs numériques d'un tableau et de les afficher dans une liste(listbox).
-Rechercher la plus petite valeur du tableau ainsi que la plus grande.
-Calculer la somme des éléments du tableau.
-Ecrire une fonction retournant le nombre de fois qu'apparaît une valeur donnée dans le tableau.

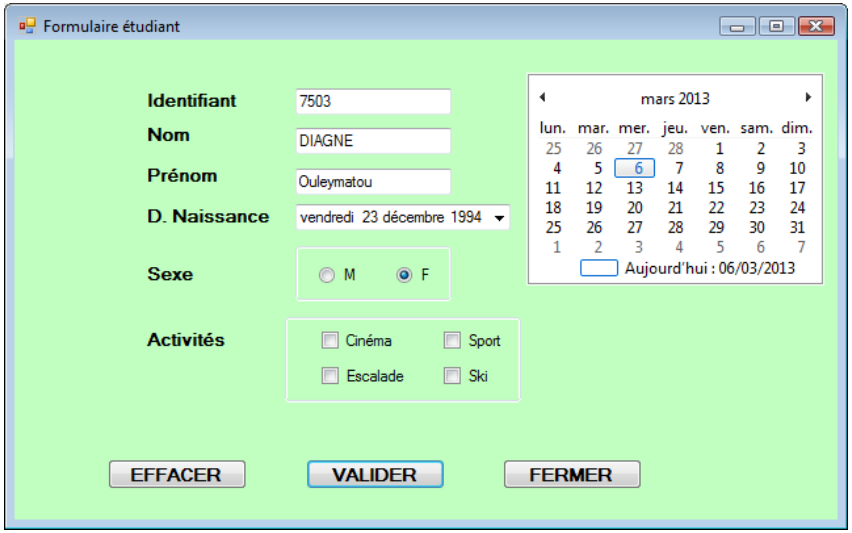
Autres contrôles

ListView permet d'afficher des données dans une liste avec grille et une meilleure organisation des titres

```
Private Sub Button2_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles Button2.Click
        Dim infos As String
        Dim sexe As String

        txtNomEtud2.Text = txtNomEtud.Text
        txtPrenomEtud2.Text = txtPreomEtud.Text
        'Boutons radio
        If radsexeF.Checked = True Then
            sexe = "F"
        Else
            sexe = "M"
        End If
       'Ce que l'on veut afficher dans la listView
        infos = txtNomEtud2.Text & " - " & txtPrenomEtud2.Text
& " - " & sexe
       'On le met dans la variable « ligne »
        Dim Ligne As ListViewItem = New ListViewItem(New
String() {txtNomEtud2.Text, txtPrenomEtud2.Text})

        'Affichage des NOMS des colonnes (entêtes: columnHeads)
        ListView1.GridLines = True  ' On affiche la grille du
lstView
        ' ListView1.View = View.Details
        lstBoxEtud.Items.Add(infos)'Affichage dans la "listBox"

        ListView1.Items.Add(Ligne)'Affichage dans la "listView"
        Me.Refresh()

    End Sub
```

ListBox , ListView, boutons radio (radio buttons) et **cases à cocher (checkboxes), Calendrier (MonthCalendar), Saisie date (DateTimePicker)**





Déclaration de structure dans le module « Modetu »

```
Module Modetu
   Public Structure etu
      Public id As String   ' identifiant étudiant
      Public nom As String   ' nom
      Public pren As String   ' prénom
      Public nais As Date   ' date de naissance
      Public sexe As Char    ' sexe
   End Structure

End Module
```

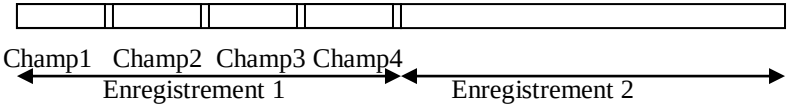
Code de la procédure CREER (étudiant)

```
  Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button3.Click
      Dim stud As etu
      stud.id = Me.txtIdetud.Text
      stud.nom = Me.txtNometud.Text
      stud.pren = Me.txtPrenetud.Text
      stud.nais = Me.dtpNaissance.Text

      'recup sexe dans le groupe des grpsexe des boutons radio
      If Me.radF.Checked Then  'Me.radF.Checked = true (sélectionné)
         stud.sexe = "F"
         MsgBox("Vous êtes une fille")
      Else                'Me.radF.Checked = false (non sélectionné)
         stud.sexe = "M"
         MsgBox("Vous êtes un garçon")
      End If
     'Affichage des infos dans la listBox du formulaire principal
      frmGestEtud.lstetudiants.Items.Add(stud.id & " - " & stud.nom & " - " &
stud.pren & " - " & stud.nais & " - " & stud.sexe)

      'recup activités dans le groupe grpactivites des cases à cocher
      If Me.chkCinema.Checked Then  (Cinema sélectionné)
         stud.activ1= "Cinéma"
       End Sub
```

GESTION DES FICHIERS à accès direct (Random Files)

Les fichiers RANDOM sont des fichiers séquentiels ie des des suites de données bien organisées : ils sont constitués de **champs** et d'**enregistrements**. Tous les enregistrements ont la même taille en octets. Pour les gérer, on utilise un **type structuré** clairement défini dans un module.

Pour manipuler un fichier, il faut d'abord l'**ouvrir,** on peut alors **lire** ses données, y **écrire** ou **modifier les données**, enfin il vaut mieux le **fermer** après l'avoir utilisé. Voici un exemple de structure d'un fichier à accès direct ou aléatoire :



Exemple de structure « client » déclaré dans un module.

```
Public Structure client
        <VBFixedString(4)> Public cliNum As String
        <VBFixedString(15)> Public cliNom As String
        <VBFixedString(15)> Public cliPrenom As String
        <VBFixedString(25)> Public cliAdresse As String
        <VBFixedString(5)> Public cliCpostal As String
        <VBFixedString(20)> Public cliVille As String
        <VBFixedString(10)> Public cliTelfixe As String
        <VBFixedString(10)> Public cliTelmob As String
End Structure
```

CREATION d'enregistrement (écriture dans le fichier)

```
Private Sub btnSauver_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnSauver.Click
          'Création de client
        ouvrir_clients2()
        Dim vcli As New client
        Dim nenreg As Integer
        nenreg = FileLen("C:\clients2.dat") \ Len(vcli)
'Nombre d'enregistrements déjà présents dans le fichier
'Récup contenus des champs (textBoxes) pour le stockage dans
la variable
' Remplissage de la structure
        With vcli
            .cliNum = zclinum.Text
            .cliNom = zclinom.Text
            .cliPrenom = zcliprenom.Text
            .cliAdresse = zcliadresse.Text
            .cliCpostal = zclicpostal.Text
            .cliVille = zcliville.Text
            .cliTelfixe = zclitelfixe.Text
            .cliTelmob = zclitelmob.Text
        End With

  'Ecriture de l'enregistrement
  nenreg = nenreg + 1   ' Position d'écriture du record on
écrit toujours à une position donnée

  FilePut(1, vcli, nenreg)         ' Ecriture de
l'enregistrement (record)

  fermer_clients2()
    End Sub
```

LECTURE d'enregistrement (lecture du fichier)

```
Private Sub ListeToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ListeToolStripMenuItem.Click

        Dim vcli As New client : Dim i As Integer : Dim
ligne As String

  ' Ouvrir le fichier en lecture/écriture

  FileOpen(1, "C:\CLIENTS2.DAT", OpenMode.Random,
OpenAccess.ReadWrite, OpenShare.LockWrite, Len(vcli))
        i = 1
While (Not EOF(1))  'Lecture d'un enregistrement du fichier
  FileGet(1, vcli, i)     ' Lecture enregistrement (record)
```

```
' Variable ligne d'affichage
 ligne = vcli.cliNum + " - " + vcli.cliNom + " - " +
vcli.cliPrenom + " - " + vcli.cliAdresse + " - " +
vcli.cliCpostal + " - " + vcli.cliVille + " - " +
vcli.cliTelfixe + " - " + vcli.cliTelmob
' Affichage d'une ligne d'infos d'enregistrement
   frm_cli_liste.lst_clients2.Items.Add(ligne)
   i = i + 1  ' On passe au suivant
End While

 frm_cli_liste.Show()   ' Onaffiche la liste des clients

'fermer_clients2()

    FileClose(1)
End Sub
```

Les procédures pour ouvrir et fermer le fichier

```
  Public Sub ouvrir_clients2()'On ouvre en lecture/écriture
        Dim vcli As New client
        ' Ouvrir le fichier en lecture/écriture.
        FileOpen(1, "C:\CLIENTS2.DAT", OpenMode.Random,
OpenAccess.ReadWrite, OpenShare.LockWrite, Len(vcli))
  End Sub

  Public Sub fermer_clients2()
       ' Fermer le fichier
        FileClose(1)
  End Sub
```

ACCES A DES BASES DE DONNEES par VB.NET

1) Base MS ACCESS

Accès à une BDD MS Access par OLEdb
Chaîne de connection ACCESS 2007 :
"Provider=**Microsoft.ACE.OLEDB.12.0**; Data Source=C:\Users\hector\Desktop\CLASSES\DEUST1\BDD-ACCESS**marketing.accdb**"

- ----------

I-Cas 1

```
Imports System.Data.OleDb     ' Bilio importées pour OLE

' ------------------ Début de la classe --------------------

Public Class frmOledbase

    Public con As New OleDbConnection

' ------------------------------------------------------------

    Private Sub Fermer_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click
        Me.Close()
    End Sub

' ------------------------------------------------------------

    Private Sub Executer_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Autre.Click

        Try
            ' Définition de la chaîne de connection
            con.ConnectionString =
"provider=microsoft.jet.oledb.4.0 ; data
source=C:\VBdev\spa.mdb"
            con.Close()   ' Fermeture pour le cas où elle
serait restée ouverte
            con.Open()    ' Ouverture connection à la BDD

            voir_results()   'Exécuter une requete et voir
les enregistrements résultats

            con.Close()   ' Fermeture connection à la BDD

        Catch ex As Exception
            MessageBox.Show(ex.ToString)
        End Try

    End Sub
' ------------------------------------------------------------

    Public Sub voir_results ()
        Dim dt As New DataTable ' Préparation de la table
"résultats" de la requête
        Dim ds As New DataSet
        ds.Tables.Add(dt)

        ' Exemples de requêtes à tester - Les en-tête de
colonnes sont à modifier

        Dim da As New OleDbDataAdapter("select * from maitre
where mVille <>'LILLE' ;", con)
        'Dim requete As String = "select * from maitre where
mVille ='LILLE' ;"
        'Dim requete As String = InputBox("Tapez votre
requête ")
        'Dim da As New OleDbDataAdapter(requete, con)
        'Dim da As New OleDbDataAdapter("select * from
maitre;", con)

        da.Fill(dt)   ' Remplissage du Data_adaptateur (da)
```

```
Dim enreg As DataRow
   lstviewOle.Items.Clear()
        'Affichage des données dans un listview
        For Each enreg In dt.Rows
         lstviewOle.Items.Add(enreg.Item(0))
        lstviewOle.Items(lstviewOle.Items.Count -
1).SubItems.Add(enreg.Item(1))
        lstviewOle.Items(lstviewOle.Items.Count -
1).SubItems.Add(enreg.Item(2))
        lstviewOle.Items(lstviewOle.Items.Count -
1).SubItems.Add(enreg.Item(3))
        lstviewOle.Items(lstviewOle.Items.Count -
1).SubItems.Add(enreg.Item(4))
        lstviewOle.Items(lstviewOle.Items.Count -
1).SubItems.Add(enreg.Item(5))
        Next

    End Sub
' -------------------------------------------------------

    Private Sub Button2_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnRAZ.Click
        lstviewOle.Items.Clear()
    End Sub

' -------- Fin de la classe -------------------------
End Class
```

II-Cas 2

```
' ------------------ Début de la classe -------------------
    Dim con As New OleDbConnection
    Public Sub voir_results()
        Dim dt As New DataTable ' Préparation de la table
"résultats" de la requête
        Dim ds As New DataSet
        ds.Tables.Add(dt)
        ' Exemples de requêtes à tester - Les en-tête de
colonnes sont à modifier

        Dim da As New OleDbDataAdapter("select * from maitre
", con)
        'Dim requete As String = "select * from maitre where
mVille ='LILLE' ;"
        'Dim requete As String = InputBox("Tapez votre
requête ")
        'Dim da As New OleDbDataAdapter(requete, con)
        'Dim da As New OleDbDataAdapter("select * from
maitre;", con)

        da.Fill(dt)   ' Remplissage du Data_adaptateur (da)

        Dim enreg As DataRow
        lstviewOle.Items.Clear()
        'Affichage des données dans un listview
        For Each enreg In dt.Rows
            lstviewOle.Items.Add(enreg.Item(0))
            lstviewOle.Items(lstviewOle.Items.Count -
1).SubItems.Add(enreg.Item(1))
            lstviewOle.Items(lstviewOle.Items.Count -
1).SubItems.Add(enreg.Item(2))
            lstviewOle.Items(lstviewOle.Items.Count -
1).SubItems.Add(enreg.Item(3))
            lstviewOle.Items(lstviewOle.Items.Count -
1).SubItems.Add(enreg.Item(4))
            lstviewOle.Items(lstviewOle.Items.Count -
1).SubItems.Add(enreg.Item(5))
        Next

    End Sub

    Private Sub btnExecuter_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnExecuter.Click
```

```
 Try
            ' Définition de la chaîne de connection
            con.ConnectionString =
"provider=microsoft.jet.oledb.4.0 ; data
source=C:\VBdev\spa.mdb"
            con.Close()    ' Fermeture pour le cas où elle
serait restée ouverte
            con.Open()     ' Ouverture connection à la BDD

            voir_results()   'Exécuter une requete et voir
les enregistrements résultats

            con.Close()    ' Fermeture connection à la BDD

        Catch ex As Exception
            MessageBox.Show(ex.ToString)
        End Try
    End Sub
```

2) Base MYSQL

```
Imports MySql.Data
Imports MySql.Data.MySqlClient

Public Class frmMysqlproj
    'NE PAS OUBLIER DE DEMARRER LA BASE MySQL
    Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs)
        'Connection
        ' Dim connStr As String =
"SERVER='localhost';DATABASE='employes';UID='henri';PASSWORD
='henri'"
        Dim connStr As String = "Database=employes;" & "Data
Source=127.0.0.1;" & "User Id=henri;Password=henri;" &
"Connection Timeout=20"
        'Fin Connexion
        Dim connection As New MySqlConnection(connStr)

        Dim connexion As MySqlConnection = New
MySqlConnection()
        Dim adaptateur As MySqlDataAdapter = New
MySqlDataAdapter()
        Dim com As MySqlCommand = New MySqlCommand
        Dim requete As String
        'Dim tablereacteur As New DataTable

        'Dim maliste As New List(Of String)
        Dim myDataTable As New DataTable

        Try
            connexion.Open()
            requete = "SELECT * FROM salarie;"

            com.Connection = connexion
            com.CommandText = requete
            adaptateur.SelectCommand = New
MySqlCommand(requete, connexion)

         For i As Integer = 0 To myDataTable.Rows.Count - 1
                'Affichage des valeurs des champs dans les
textBoxes
                txtNum.Text =
myDataTable.Rows(i)("salNum").ToString
                txtNom.Text =
myDataTable.Rows(i)("salNom").ToString
                txtSite.Text =
myDataTable.Rows(i)("salSite").ToString
            Next

            connexion.Close()

        Catch ex As Exception
            MsgBox("erreur : " + ErrorToString())
```

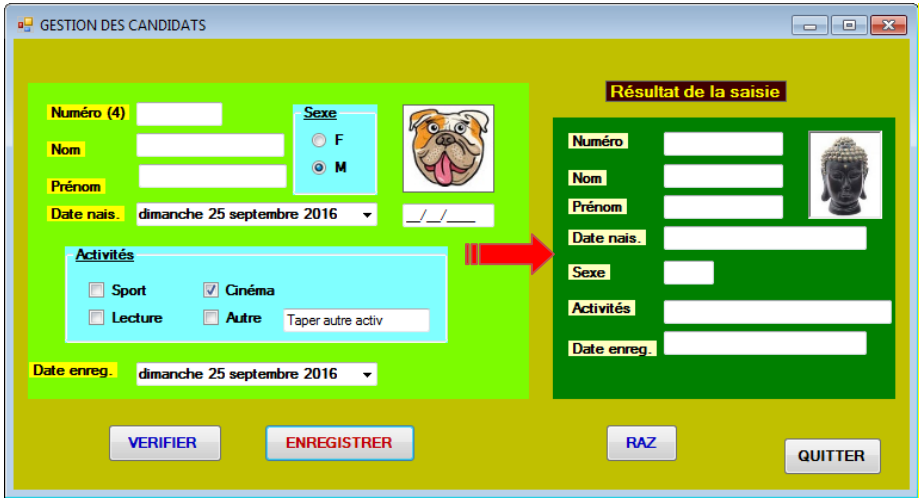
```
        End Try
    End Sub
    Private Sub Button2_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button2.Click
        'Chaîne de Connexion
       Dim ConnexionSql As String = "Database=voyages;" &
"Data Source=192.168.20.44;" & "User Id=root;
Password=pass;" & "Connection Timeout=20"
        'Dim myDataTable As New DataTable
        Dim vSal As String

        Try                    'Intéressant pour gérer les
erreurs
            vSal = txtSalarie.Text
            Dim requete As String = "SELECT * FROM employe
where empNum = " & "'" & txtSalarie.Text & "'" 'Requête SQL
            Dim connection As New
MySqlConnection(ConnexionSql)
            Dim cmd As New MySqlCommand(requete, connection)

            connection.Open()   'Ouverture connection

            Dim reader As MySqlDataReader
            reader = cmd.ExecuteReader()     'Execution du
Reader, plus rapide

            While (reader.Read())
                'Les champs du dernier enreg. Les indices
commencent à 0.
                txtNum.Text = ((reader.GetString(0)))
                txtNom.Text = ((reader.GetString(1)))
                txtSite.Text = ((reader.GetString(2)))

                'Affichage dans le comboBox
                ' ComboBox1.Items.Add((reader.GetString(0)))

            End While
            'Affichage champ "nom" dans le listBox
            While (reader.Read())   'Tant qu'il y a un
enregistrement

ListBox2.Items.Add(reader.GetString("empNom"))
            End While

            reader.Close()                  'Fermeture du
Reader et de la connection à la base
            connection.Close()
        Catch ex As Exception               'Gestion
d'erreurs
            MessageBox.Show(ex.Message)     'Message de
l'exception
        End Try
    End Sub

    Private Sub Button1_Click_1(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Button1.Click
        Me.Close()
    End Sub

    Private Sub Button3_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button3.Click
        Dim ConnexionSql As String = "Database=voyages;" &
"Data Source=192.168.20.44;" & "User Id=root;Password=pass;"
& "Connection Timeout=20"
        Dim myDataTable As New DataTable

        Dim query As String = "SELECT * FROM employe"
'Requête SQL
        Dim connection As New MySqlConnection(ConnexionSql)
        Dim cmd As New MySqlCommand(query, connection)

        connection.Open()
```

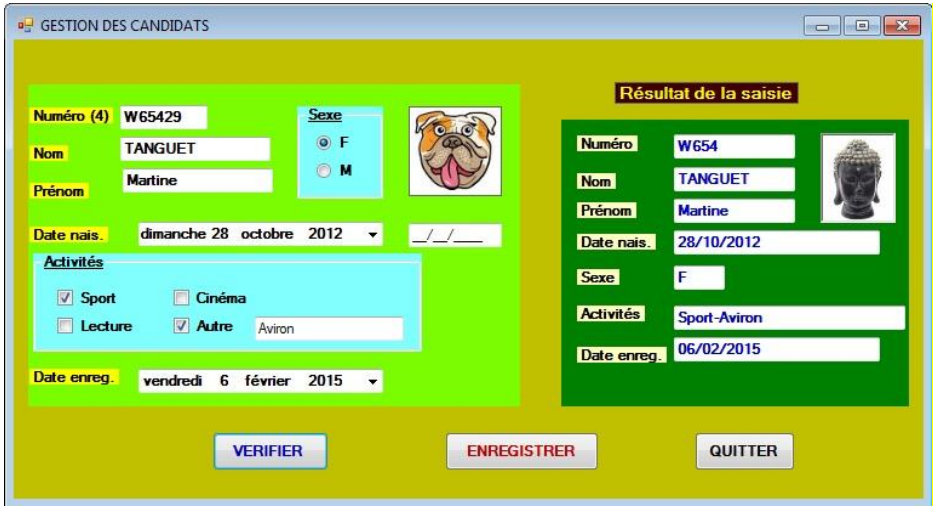
```
        Dim reader As MySqlDataReader
        reader = cmd.ExecuteReader()
        'Dim i As Integer = 0
        ListBox1.Items.Clear()
        ListBox2.Items.Clear()
        ComboBox1.Items.Clear()
        While reader.Read()
            'Les champs à recup. Les tables commencent à 0.
            ListBox1.Items.Add(reader.GetString(0) & " - " _
                    & reader.GetString(1) & " - " &
reader.GetString(2) & " - " _
                    & reader.GetString(3) & " - " &
reader.GetString(4) & " - " _
                    & reader.GetString(5))
            ComboBox1.Items.Add(reader.GetString(0) & " - "
& reader.GetString(1))
            ListBox2.Items.Add(reader.GetString(1) & " - " &
(reader.GetString(4)))
        End While

        reader.Close()
        connection.Close()

    End Sub

    Private Sub frmMysqlproj_Load(ByVal sender As Object,
ByVal e As System.EventArgs) Handles Me.Load
        'Jouer un peu de musique au chargement de la feuille

    End Sub

    Private Sub Button4_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button4.Click
        ListBox1.Items.Clear()
    End Sub

    Private Sub Button5_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button5.Click
        Dim ConnexionSql As String = "Database=voyages;" &
"Data Source=192.168.20.44;" & "User Id=root;Password=pass;"
& "Connection Timeout=20"
        ' Insertion d'un enregistrement dans les agences
        Dim query As String = "insert into agence values(" &
"'" & txtAgCode.Text & "'" & "," & "'" & txtAgNom.Text & "'"
& ")"
        Dim connection As New MySqlConnection(ConnexionSql)
        Dim cmd As New MySqlCommand(query, connection)

        connection.Open()

        Dim reader As MySqlDataReader
        reader = cmd.ExecuteReader()
    End Sub

    Private Sub Button6_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button6.Click
        Dim ConnexionSql As String = "Database=voyages;" &
"Data Source=192.168.20.44;" & "User Id=root;Password=pass;"
& "Connection Timeout=20"
        Dim myDataTable As New DataTable

        Dim query As String = "SELECT * FROM agence"
'Requête SQL
        Dim connection As New MySqlConnection(ConnexionSql)
        Dim cmd As New MySqlCommand(query, connection)

        connection.Open()

        Dim reader As MySqlDataReader
        reader = cmd.ExecuteReader()
        'Dim i As Integer = 0
        ListBox3.Items.Clear()

        While reader.Read()
            'Champs de la table Agence.
            ListBox3.Items.Add(reader.GetString(0) & " - " _
```

```
                    & reader.GetString(1))
        End While

        reader.Close()
        connection.Close()
    End Sub
End Class
```

EXEMPLE PROJET

Gestion des candidats





```
Module modCandidats
    Public Structure candidat  'Déclaration de type
        <VBFixedString(4)> Public candNum As String
        <VBFixedString(15)> Public candNom As String
        <VBFixedString(15)> Public candPrenom As String
        <VBFixedString(1)> Public candSexe As String
        <VBFixedString(10)> Public candDnaiss As String
        <VBFixedString(10)> Public candDenreg As String
        <VBFixedString(50)> Public candLoisirs As String
    End Structure
End Module
      -   ------------------------------------------------

Public Class frmMenuCandidat           'Classe principale

      -   ------------------------------------------------

    Private Sub Button3_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button3.Click

        RAZ_verif()
        'Transfert des infos candidat dans la fenêtre résultat
        txtRNum.Text = Mid(txtNumero.Text, 1, 4)
        txtRNom.Text = txtNom.Text
        txtRPrenom.Text = txtPrenom.Text
        txtRDnaiss.Text = CDate(dtPickerDnaiss.Text)    'A
saisir au format mm/jj/aaaa
        'txtRDnaiss.Text = maskedDNaiss.Text
        txtRDenreg.Text = CDate(dtPickerDenreg.Text)

        MsgBox("La date de naissance se présente ainsi : " &
txtRDenreg.Text)


        'tests sexe saisi
        If radbtnF.Checked = True Then
            txtRSexe.Text = "F"
        Else
            txtRSexe.Text = "M"
        End If

        'test activites
        Dim activ As String = ""
        If chkBoxSport.Checked = True Then
            activ = activ & "-" & "Sport"
        End If

        If chkBoxCinema.Checked = True Then
            activ = activ & "-" & "Cinéma"
        End If
        If chkBoxLecture.Checked = True Then
            activ = activ & "-" & "Lecture"
        End If

        If chkBoxAutre.Checked = True Then
            activ = activ & "-" & txtAutre.Text
        End If

        txtRActivites.Text = Mid(activ, 2, 50) ' On laisse
tomber le premier "-"

        Me.Refresh()

    End Sub
      -   ------------------------------------------------
    Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click
        Dim rep As Integer

        rep = MsgBox("Confirmez-vous la sauvegarde après
vérification", vbOKCancel)
        MsgBox("Valeur du bouton cliqué : " & rep.ToString)
        If rep = 1 Then
            sauver_candidat()
        Else
            MsgBox("Sauvegarde annulée - Aurevoir ! ",
vbExclamation)
        End If
    End Sub
```

```
    Private Sub sauver_candidat()
        Dim vcandidat As New candidat : Dim longueur As Integer
        Dim nbenreg As Integer : Dim position As Integer

        ouvrir_fichier_candidats()

        longueur = Len(vcandidat)
        nbenreg = FileLen("C:\candidats.dat") \ Len(vcandidat)
        ' nbenreg est le nombre actuel d'enregistrements

        'Calcul de la position d'écriture
        position = nbenreg + 1

        'Recup champs
        With vcandidat
            .candNum = txtRNum.Text
            .candNom = txtRNom.Text
            .candPrenom = txtRPrenom.Text
            .candSexe = txtRSexe.Text
            .candDnaiss = txtRDnaiss.Text
            .candDenreg = txtRDenreg.Text
            .candLoisirs = txtRActivites.Text
        End With

        'Ecriture d'un enregistrement à la fin du fichier
        FilePut(2, vcandidat, position)

        fermer_fichier_candidats()

    End Sub
    -   ------------------------------------------------------
        Private Sub ouvrir_fichier_candidats()
        Dim longueur As Integer : Dim vcandidat As New candidat
        longueur = Len(vcandidat)   ' Longueur d'un
enregistrement en octets
        'Ouverture du fichier en écriture
        FileOpen(2, "C:\candidats.dat", OpenMode.Random,
OpenAccess.ReadWrite, OpenShare.LockRead, longueur)
    End Sub
    -   ------------------------------------------------------
    Private Sub fermer_fichier_candidats()
        FileClose(2)
    End Sub
    -   ------------------------------------------------------
    Private Sub RAZ_verif()
        txtRNum.Text = ""
        txtRNom.Text = ""
        txtRPrenom.Text = ""
        txtRSexe.Text = ""
        txtRDnaiss.Text = ""
        txtRDenreg.Text = ""
        txtRActivites.Text = ""
    End Sub
    -   ------------------------------------------------------
    Private Sub frmMenuCandidat_Load(ByVal sender As Object,
ByVal e As System.EventArgs) Handles Me.Load
        'RAZ des champs de vérification
        txtRNum.Text = ""
        txtRNom.Text = ""
        txtRPrenom.Text = ""
        txtRSexe.Text = ""
        txtRDnaiss.Text = ""
        txtRDenreg.Text = ""
        txtRActivites.Text = ""

    End Sub
    -   ------------------------------------------------------
    Private Sub Button4_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button4.Click
        RAZ_verif()
    End Sub

End Class
    -   ------------------------------------------------------
        Private Sub Button2_Click(ByVal sender As
        System.Object, ByVal e As System.EventArgs) Handles
        Button2.Click
        Me.Close()
    End Sub
```

TRAVAUX DIRIGES Visual Basic

H. TSOUNGUI. 2016

Exo-1 Programme de calcul de TVA

Pour ce premier exercice, il vous est proposé un formulaire. Par la suite, vous concevrez vous-mêmes vos propres interfaces.

Ce formulaire comporte des champs de saisie de valeurs (montant hors taxes, taux de TVA) ainsi qu'un premier bouton de commande permettant de déclencher une procédure de calcul et affichage des résultats. Deux derniers boutons permettent de remettre à « blanc » toutes les zones de saisie ou de fermer le formulaire.

1-Ecrire le code de chacun des boutons de commande de l'interface.

Exo-2 Types de données complexes : vecteurs (tableaux à 1 dimension) - Structures de contrôle

-Ecrire un programme VB permettant de manipuler un vecteur (tableau à une dimension) d'entiers.

1-Déclarer le vecteur V de 5 nombres entiers
2-Saisir les 5 valeurs numériques du tableau avec **InputBox**
3-Afficher les valeurs saisies dans une liste (**listbox**).
4-Rechercher la plus petite valeur ainsi que la plus grande.
5-Calculer la somme des éléments du tableau
6-Calculer la moyenne des éléments du tableau

Exo-3

- Refaire ***l'exo-2*** en utilisant la fonction de génération des nombres aléatoires RND() pour automatiser l'obtention des valeurs du tableau à une dimension.
- Remplir un vecteur de valeurs tirées au hasard entre 1 et 9.
- Lire une valeur à rechercher les occurrences dans le vecteur par exemple 2.
- Ecrire le code donnant le nombre de fois qu'apparaît cette valeur dans le vecteur.

Exo-4 Tableaux à plusieurs dimensions (matrices) – structures de contrôle

-Manipulation d'un tableau à plusieurs dimensions

Tableau des clients **CLIENTS (4 lignes x 4 colonnes)**

Numéro-client	Nom-client	Prénom-client	Chiffre-affaires
C1	DEMON	Jean	945.34
C2	MOULOUD	Aziz	2304.85
C3	MARTIN	David	4963.57
C4	LEROUGE	Alain	3945.34

1-Déclarer dans un **module** un tableau CLIENTS(4,4) de chaines ;
2-Saisir les données ci-dessus par client ;
3-Afficher les données de ce tableau dans un formulaire ou une liste déroulante ;
4-Créer une interface et écrire le code permettant de rechercher un client et d'afficher ses informations.
5-Rechercher le meilleur client et afficher ses infos (Numéro, Nom et Chiffre d'affaires).

Variations sur le thème

---suite guide de résolution

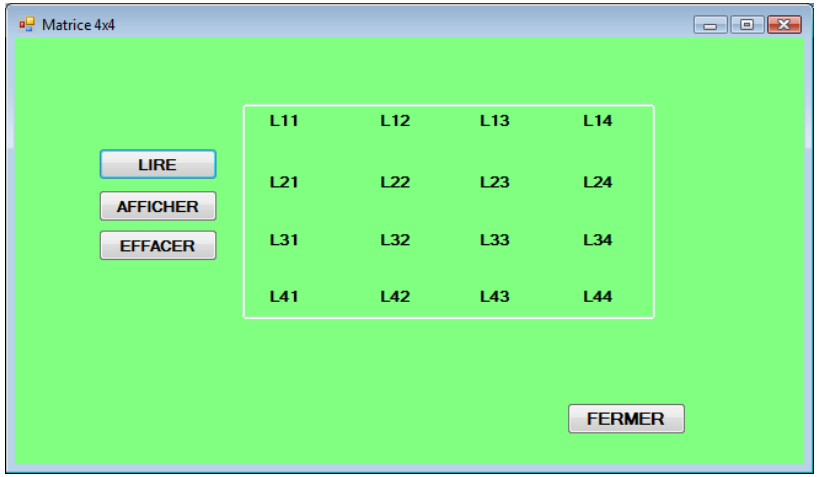
-Manipulation d'un tableau à plusieurs dimensions

Tableau des clients CLIENTS (4 lignes x 4 colonnes)

1-Déclarer dans un **module** un tableau CLIENTS(4,4) de chaines ;

```
Module Modmatrice
    Public clients(4, 4) As String
End Module
```

2-Saisir les données ci-dessus par client ;

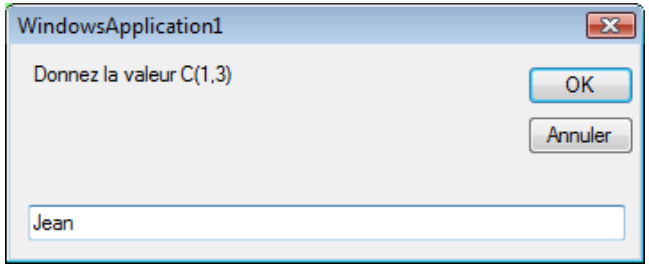


- Quand on clique sur le bouton « LIRE », il faut saisir les données dans deux boucles imbriquées

```
Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click
        Dim i As Integer : Dim j As Integer
        'Lecture des valeurs de la matrice
        For i = 1 To 4
            For j = 1 To 4
                clients(i, j) = InputBox("Donnez la
valeur C(" & i & "," & j & ")")
            Next j
        Next i
        'Affichage des valeurs lues
        l11.Text = clients(1, 1) : l12.Text = clients(1,
2) : l13.Text = clients(1, 3) : l14.Text = clients(1, 4)
   …    …    …    …
    End Sub
```

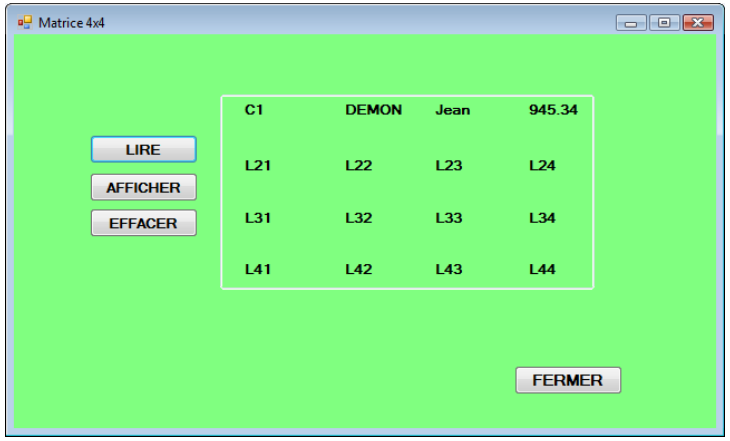
NB : On peut aussi initialiser les éléments du tableau une fois pour toutes avec desaffectations du genre suivant

```
clients(1, 1) = « C1 » : clients(1, 2) = « DEMON » :
clients(1, 2) = ...
```



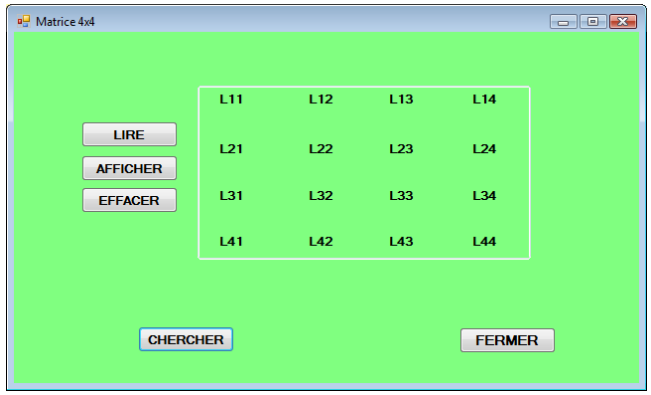
3-Ensuite, il faut afficher les valeurs lues dans les labels qui ont pour nom (propriété name L11, L12 tout en minuscules)

```
'Affichage des valeurs lues
   l11.Text = clients(1, 1) : l12.Text = clients(1,
2)
   l13.Text = clients(1, 3) : l14.Text = clients(1,
4)
   … …
```

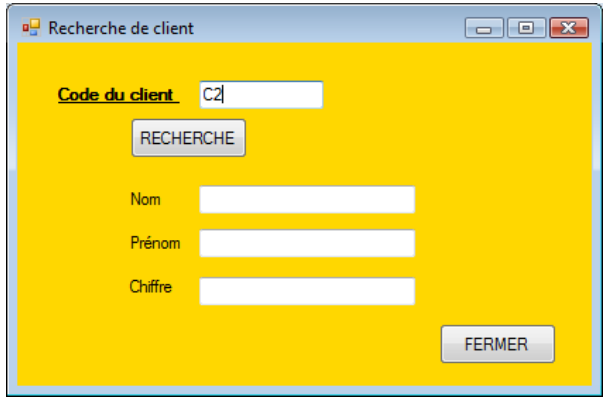


4-Créer une interface et écrire le code permettant de rechercher un client et d'afficher ses informations.

On ajoute un bouton « CHERCHER » dans le premier formulaire



En cliquant dessus, le formulaire suivant s'ouvre :



On écrit le code de la recherche du bouton « RECHERCHE » :

… … … … Algorithme ……. ….
Il faut parcourir les client(i,1) 'parcours des lignes de la colonne 1
comparer les codes, etc.
Si égalité entre le code client saisi et la valeur de client(i,1),
on a trouvé,
alors on sait que le client cherché est sur la ligne k=i ;
Il suffit alors d'afficher client(k,1) pour son code, client(k,2) pour son nom, client(k,3) son prénom et
Client (k,4) pour son chiffre d'affaires.
Fin du parcours

5-Rechercher le meilleur client et afficher ses infos (Numéro, Nom et Chiffre d'affaires).
… … … … Algorithme ……. ….
On fixe comme meilleur client le premier de la liste ; puis on parcoure les autres lignes à partir de i=2 dans une boucle
Pour i=2 à 4 ,
on compare le chiffre d'affaires avec le meilleur du moment
si on trouve plus grand,
on refixe le « nouveau meilleur » en prenant soin de récupérer la position kpos = i.
Fin Pour ;

-La procédure est la même si on veut trouver le « plus mauvais » client.
Les mauvais clients sont ceux qui auraient un chiffre négatif …

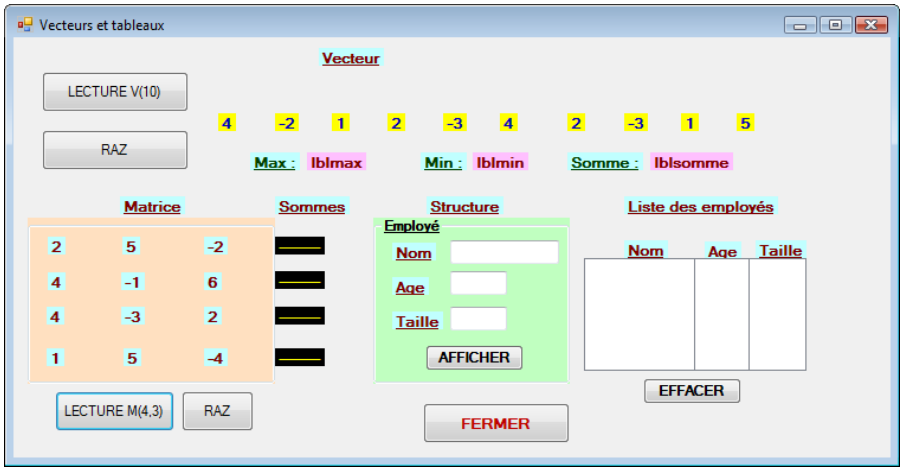
Exo-5 Gestion simplifiée de stock de produits
De manière similaire à l'exo précédent, cet exercice propose de gérer un stock de marchandises. Il s'agit
D'utiliser un tableau PRODUITS(Référence, Libellé, Prix-HT, Quantité-en-stock), limité à 7 produits.
Créer les interfaces et écrire le code permettant de gérer le stock :
1-Entrée des produits
2-Sortie des produits
3-Inventaire du stock
4-Recherche d'un produit
5-Afficher la valeur du stock actuel
6-Saisie d'une commande et édition d'une facture

Exo-6 Jeu Loto-5 Simulation du jeu de loterie
Développer un programme en VB simulant une loterie à cinq nombres. Après avoir effectué une mise de **5 euros** au plus, on propose 5 nombres compris entre 1 et 10. On clique sur un bouton pour tirer au hasard cinq nombres **différents** dans le même intervalle. Chaque tirage diminue la mise de 1 euro.

-Afficher le résultat du tirage et gérer les gains. On peut jouer tant que la mise est positive.
-Gestion des gains :
*Si le joueur a trouvé 1 ou 2 nombres, il perd 1 euro de mise (gain : 0 euro).
*Trois nombres trouvés, gain de 2 euros
*Quatre nombres trouvés, gain de 7 euros
*Cinq nombres trouvés, gain de 10 euros

Exo-7

Utilisation des types de données complexes : vecteurs, tableaux à plusieurs dimensions et structures de données (records).



Reproduire le formulaire et écrire le code nécessaire pour les boutons de commande.

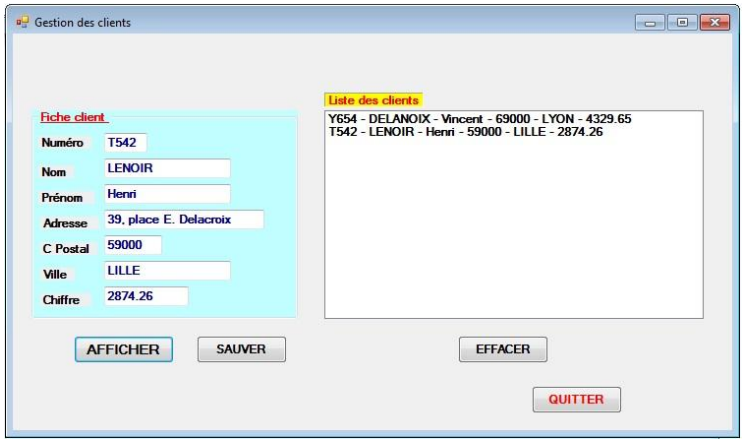
Exo-8 Utilisation de types structurés - Gestion des clients

Description de la structure à utiliser (à déclarer dans un module) :

```
Module modstruct    'Module public
    Public Structure client  'Déclaration de type
        <VBFixedString(3)> Public cliNum As String
        <VBFixedString(15)> Public cliNom As String
        <VBFixedString(15)> Public cliPrenom As String
        <VBFixedString(20)> Public cliAdresse As String
        <VBFixedString(5)> Public cliCpostal As String
        <VBFixedString(15)> Public cliVille As String
        <VBFixedString(8)> Public cliChiffre As String
    End Structure
End Module
```

Affichage des données dans un contrôle listBox

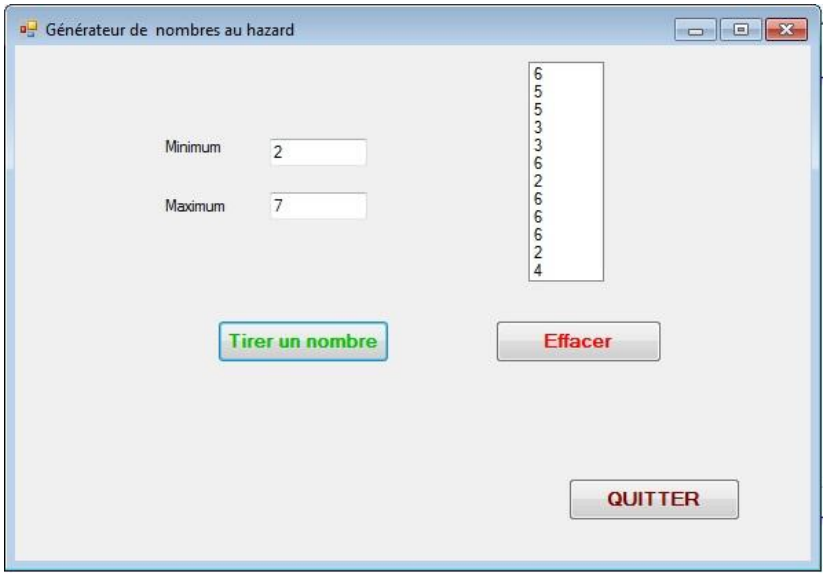
```
  Private Sub btnAfficher_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnAfficher.Click
        'Ajout dans la liste des clients
        Dim vcli As client
        Dim ligne As String

        With vcli
            .cliNum = txtNum.Text
            .cliNom = txtNom.Text
            .cliPrenom = txtPrenom.Text
            .cliAdresse = txtAdresse.Text
            .cliCpostal = txtCpostal.Text
            .cliVille = txtVille.Text
            .cliChiffre = txtChiffre.Text
        End With
        ligne = vcli.cliNum + " - " + vcli.cliNom + " - " +
vcli.cliPrenom + " - " + vcli.cliCpostal + _
            " - " + vcli.cliVille + " - " + vcli.cliChiffre
        lstClients.Items.Add(ligne)
    End Sub
```



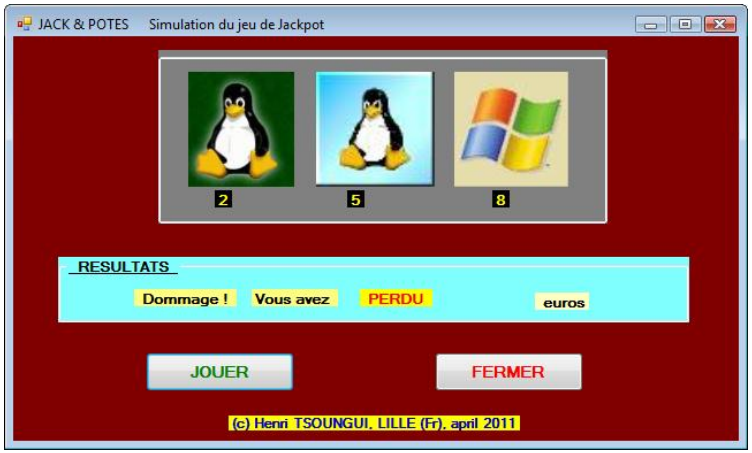
Autre exemple : gestion des étudiants

Exo-9

Ecrire un programme permettant de générer des nombres aléatoires compris dans un intervalle choisi (entre les valeurs min et max). Utiliser l'interface graphique ci-dessous pour afficher les nombres générés.

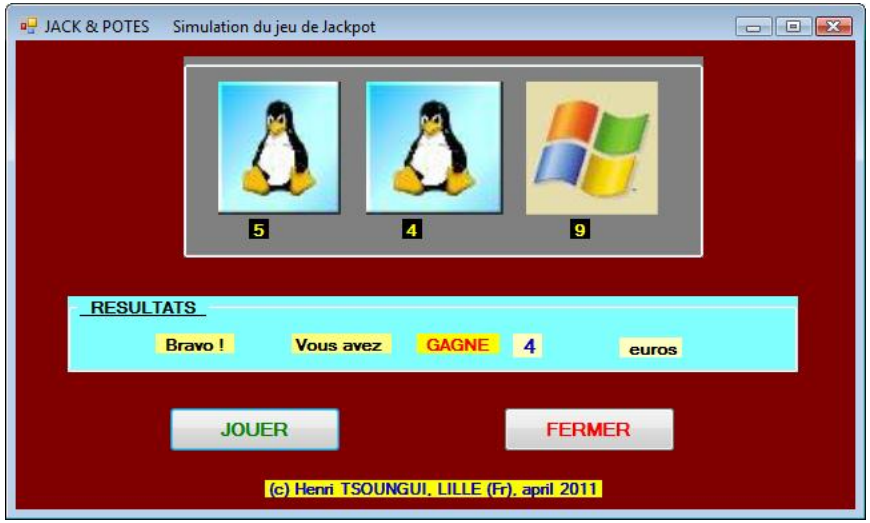


Exo-10 Simulation du jeu de Jackpot version légère 1.0 (un seul joueur)

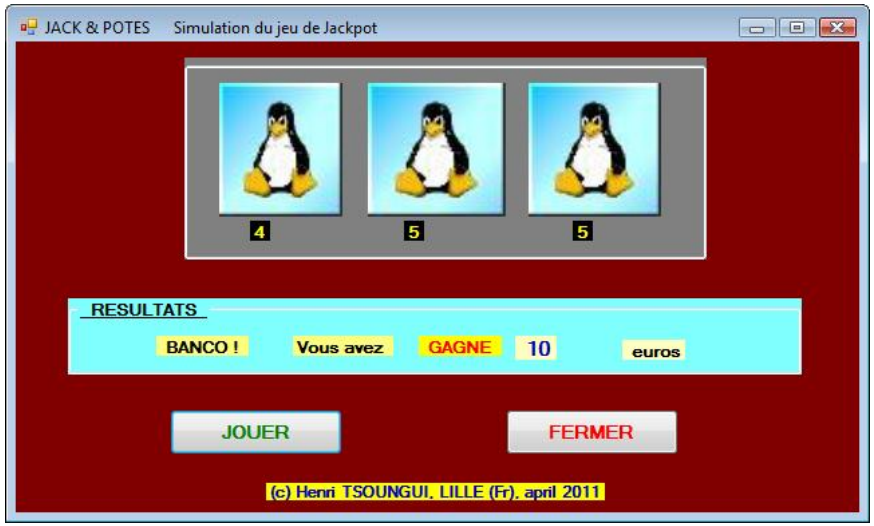


Le jeu : on clique sur « JOUER ». Le système tire au hasard 3 nombres X, Y et Z compris entre 1 et 9. Le premier nombre concerne la première roue, le deuxième la seconde et le troisième, la troisième roue. On dispose de 3 imagesimg1, img2 et img3.
-Si le nombre tiré est dans [1-2-3], on affiche l'image img1 pour la roue concernée.
-Si le nombre tiré est dans [4-5-6], on affiche l'image img2 pour la roue concernée.
-Si le nombre tiré est dans [7-8-9], on affiche l'image img3 pour la roue concernée.
Les gains :
-si on obtient trois images différentes => on perd (gain de 0 euro)
-si on obtient seulement deux images successives identiques (img1-img1-W) ou (W-img2-img2)
ou (img3-img3-W), W étant différente de l'image à côté => gain de **5** euros.
-si on obtient trois images identiques, deux cas peuvent se présenter
-les trois nombres générés sont différents => c'est un BANCO : gain de **10** euros.
-les trois nombres générés sont identiques => c'est un SUPER-BANCO : gain de **20** € !
Développez ce jeu en respectant les règles proposées.

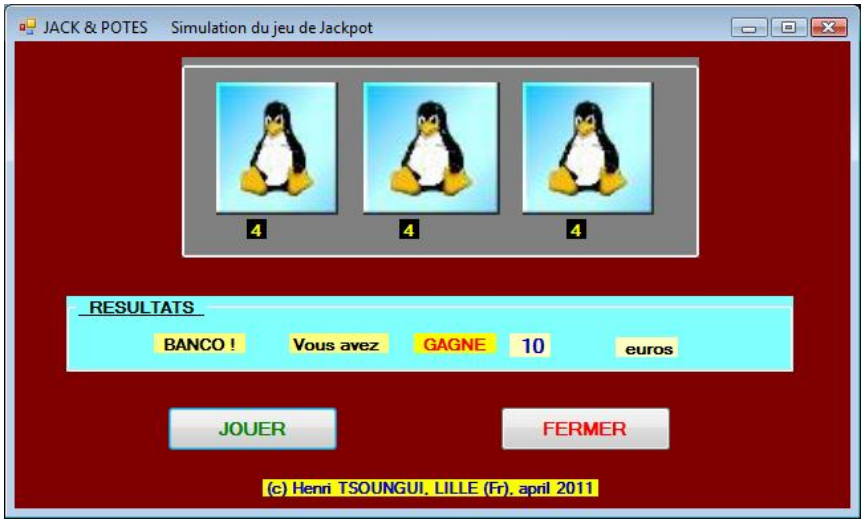
Exemples d'illustrations de situations



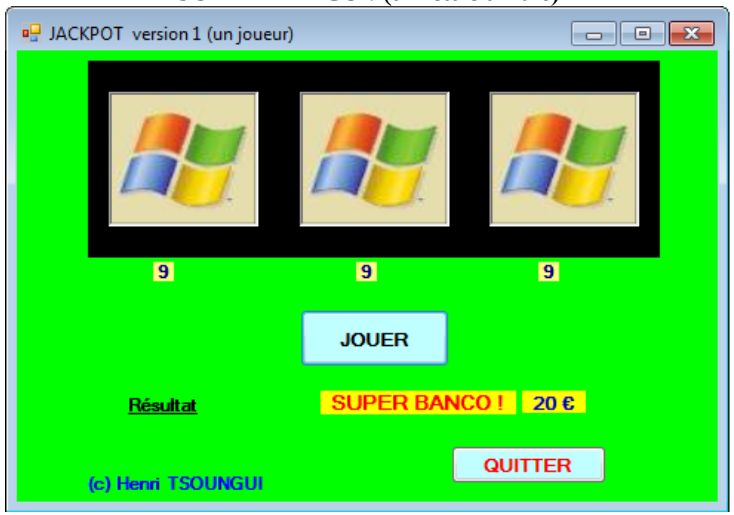
Gain simple (**5 €**)



BANCO ! (**10€**)



SUPER-BANCO ! (à mettre à **20 €**)



Exo-11 JACK&POTES Simulation du jeu de jackpot version 2.0

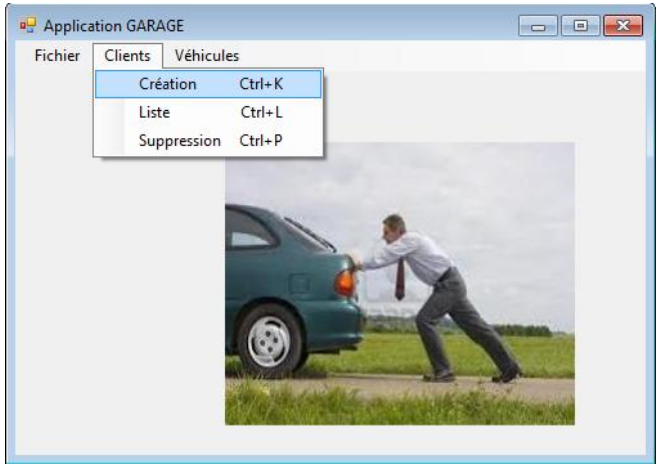
Développer une application en Visual Basic utilisant les fichiers séquentiels à accès direct pour stocker les gains.

Contraintes

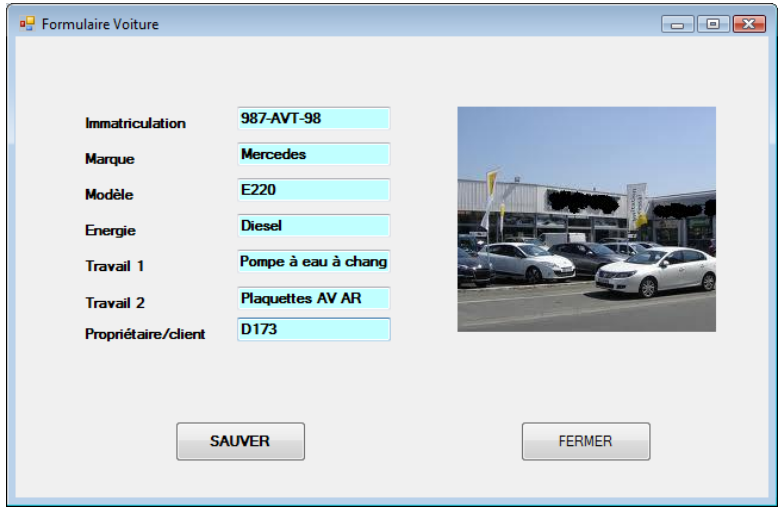
- L'application doit permettre à au moins un joueur de tenter sa chance.
- Pour jouer, tout joueur doit miser une somme minimale. Le jeu ne démarre que si la mise est suffisante.
- Pour gagner on doit avoir obtenu au moins deux *images successives identiques* :
  2 images successives identiques => gain de 10 euros
  3 images identiques => gain de 50 euros (Banco)
- La génération de *sons* est obligatoire pour distinguer les résultats.
- Les résultats des joueurs (Nom et gain total) seront sauvés dans un *fichier séquentiel* à accès direct et un bouton de commande permettra de les visualiser dans l'ordre croissant des gains (Tri des résultats).

Exo-12 Gestion des véhicules reçus dans un garage (extrait DS DEUST – 1)

Dans cet exercice, il vous est demandé de gérer des voitures qui entrent dans un garage pour qu'on y effectue certains travaux. Vous devrez également gérer la création et la suppression des clients. Vous développerez donc
-la création, la mise à jour et la suppression des véhicules
-la création et la suppression des clients
La première partie utilise le type utilisateur (structure) **voiture** déclaré dans le module **ModGarage** ainsi que le formulaire **frmVoiture** ci-dessous dont les champs de saisie sont nommés txtImmat, txtMarque, txtModèle, txtEnergie, txtTravail1, txtTravail2, txtClient. Faire de même pour la gestion des clients.



Ci-dessus, le formulaire **frmGarage** à améliorer avec des **menus déroulants**



Les structures de données à utiliser sont décrites ci-après.

```
Module Modgarage
    Public Structure client
        <VBFixedString(4)> Public cliNum As String
        <VBFixedString(15)> Public cliNom As String
        <VBFixedString(15)> Public cliPrenom As String
        <VBFixedString(25)> Public cliAdresse As String
        <VBFixedString(5)> Public cliCpostal As String
        <VBFixedString(20)> Public cliVille As String
        <VBFixedString(10)> Public cliTelfixe As String
        <VBFixedString(10)> Public cliTelmob As String
    End Structure



    Public Structure voiture
        <VBFixedString(8)> Public voitImmat As String
        <VBFixedString(15)> Public voitMarque As String
        <VBFixedString(10)> Public voitModele As String
        <VBFixedString(12)> Public voitEnergie As
String
        <VBFixedString(60)> Public voitTravail1 As
String
        <VBFixedString(60)> Public voitTravail2 As
String
        <VBFixedString(4)> Public voitClient As String
    End Structure
End Module
```

NB : cet exercice peut être fait en utilisant une petite base de données ACCESS ou SQLServer et **ADO.NET**.

Exemples d'exécution du Jackpot simple