

Programmation Visual Basic

Contrôles et groupes / Fonctions et procédures personnalisées

Laure Tougne

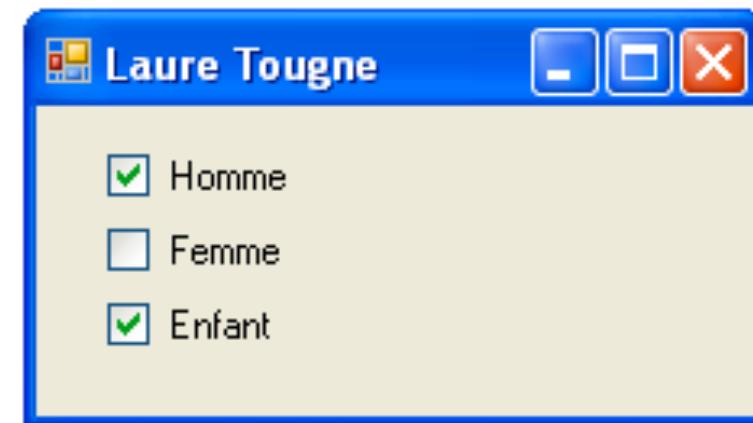
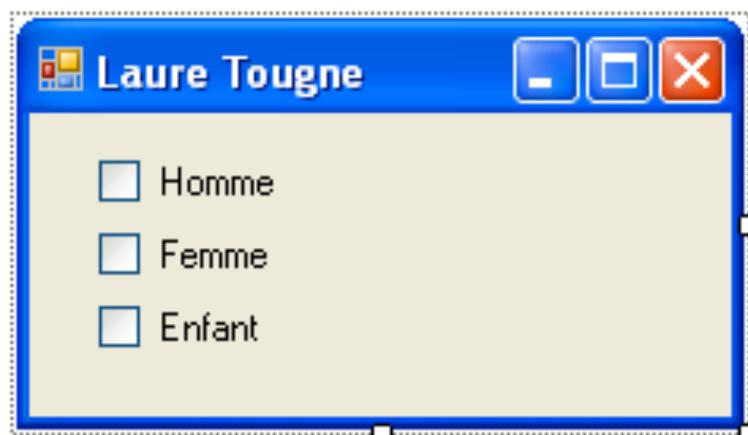
Plan

- Autres Contrôles
 - Cases, groupes de contrôles
 - Listes
- Un exemple pas à pas
- Fonctions et procédures personnalisées

AUTRES CONTRÔLES

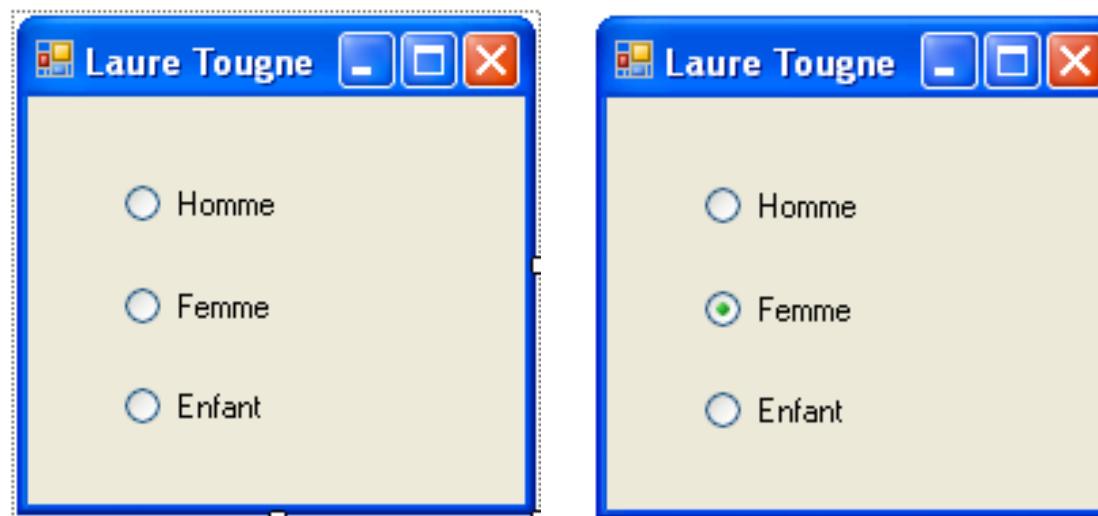
Les cases (1)

- Deux sortes de cases :
 - les cases dites "cases à cocher" (**Checkbox**): Elles sont carrées, et indépendantes les unes des autres, même si elles sont regroupées dans un cadre pour faire plus joli.



Les cases (2)

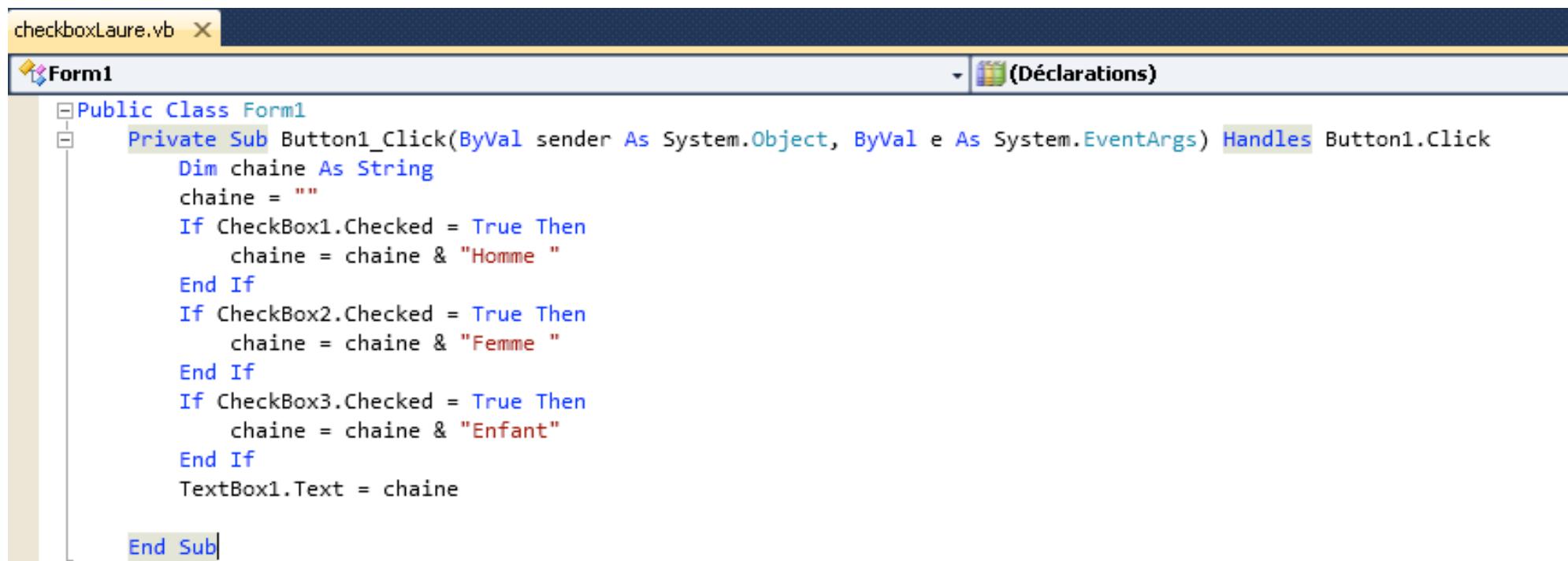
- Deux sortes de cases :
 - les cases dites "cases d'option", voire "boutons radio" (**OptionButton**). Elles sont rondes et font toujours partie d'un ensemble (dessiné par l'objet **Frame**). Au sein d'un ensemble de cases d'option, jamais plus d'une seule case ne peut être cochée à la fois.



Un exemple : interface



Un exemple : le code



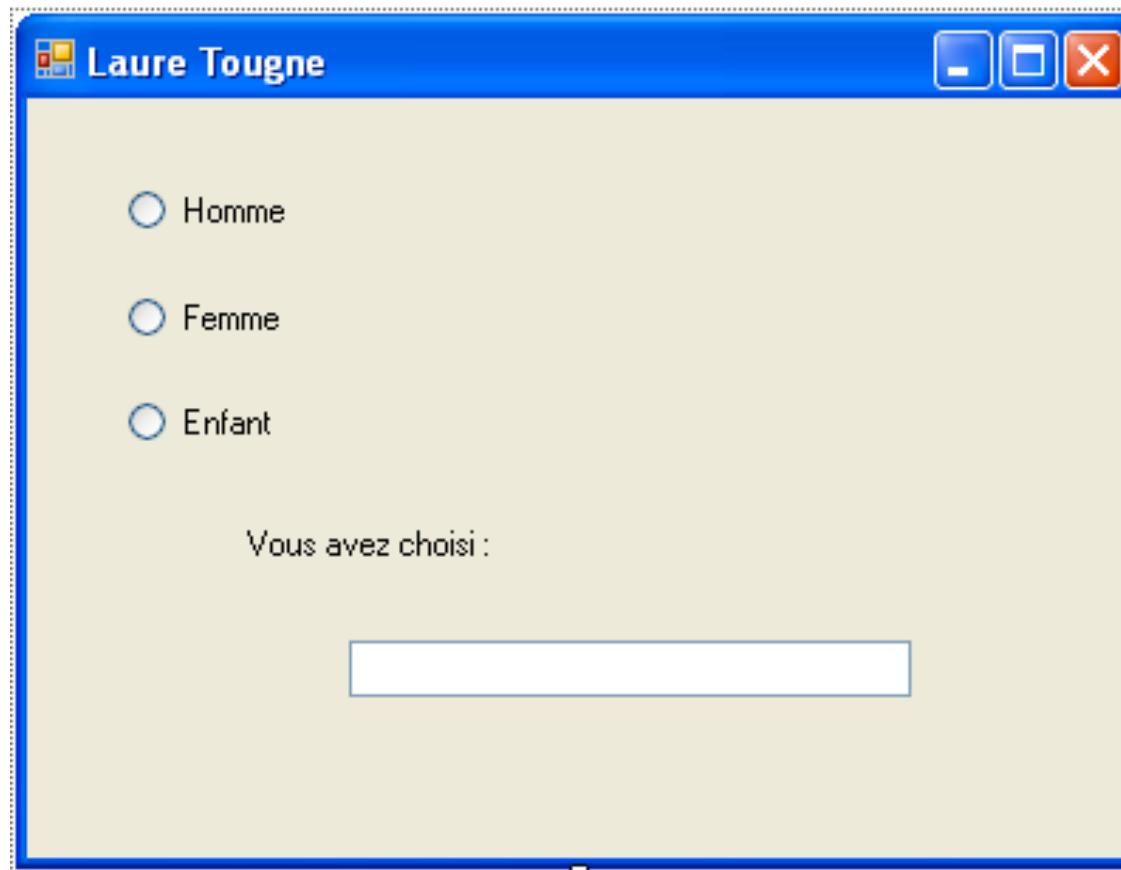
The screenshot shows a Microsoft Visual Studio interface with a dark theme. The title bar says "checkboxLaure.vb". The main window displays VBA code for a Windows application named "Form1". The code handles the "Button1_Click" event, which concatenates strings based on the checked status of three checkboxes ("CheckBox1", "CheckBox2", and "CheckBox3"). The resulting string is then displayed in a "TextBox1". The code uses standard VBA syntax with "Public Class Form1", "Private Sub Button1_Click", "If ... Then", and "End If" statements.

```
checkboxLaure.vb X
Form1 (Déclarations)
Public Class Form1
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        Dim chaine As String
        chaine = ""
        If CheckBox1.Checked = True Then
            chaine = chaine & "Homme "
        End If
        If CheckBox2.Checked = True Then
            chaine = chaine & "Femme "
        End If
        If CheckBox3.Checked = True Then
            chaine = chaine & "Enfant"
        End If
        TextBox1.Text = chaine
    End Sub
```

Un exemple : fonctionnement



Un autre exemple : interface



Un autre exemple : le code

The screenshot shows the Microsoft Visual Studio IDE interface. The title bar indicates the project is named "Form1.vb" and the file is "Form1.vb [Design]". The main window displays the source code for the Form1 class. The code uses event handlers for three radio buttons to update the text of a TextBox. The code is as follows:

```
Public Class Form1

    Private Sub RadioButton1_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles RadioButton1.CheckedChanged
        TextBox1.Text = "Homme"
    End Sub

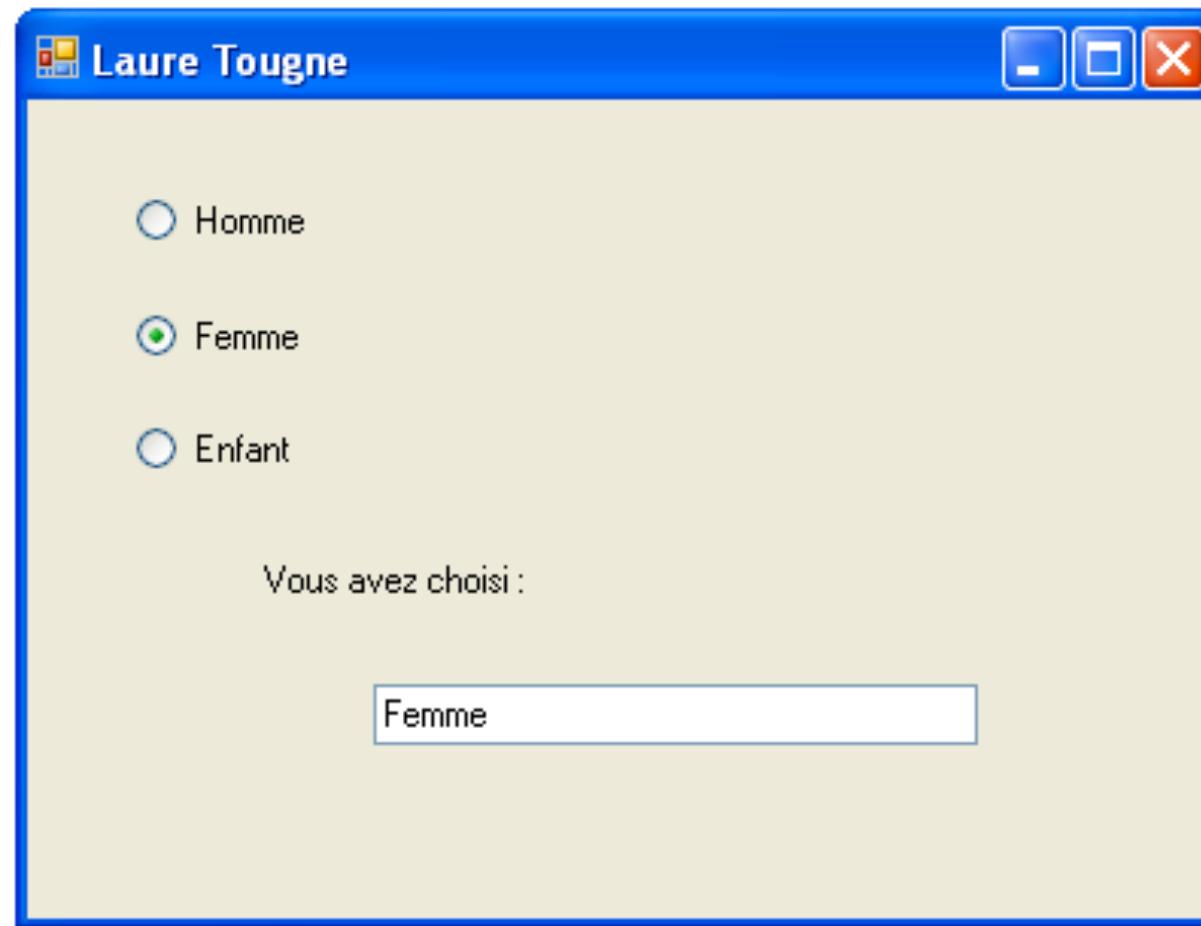
    Private Sub RadioButton2_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles RadioButton2.CheckedChanged
        TextBox1.Text = "Femme"
    End Sub

    Private Sub RadioButton3_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles RadioButton3.CheckedChanged
        TextBox1.Text = "Enfant"
    End Sub

End Class
```

The Solution Explorer on the right shows the project structure with a single file named "Form1.vb".

Un autre exemple : fonctionnement



Groupes de contrôles (1)

- Jusqu'à présent, chaque contrôle était indépendant
 - Lors de la création, choix dans la boîte à outils
 - Chaque contrôle à un Name
- Conséquence : alourdissement du code
- Créer un **groupe de contrôles** permet d'alléger le code
- Comment faire ? → **copié-collé** (les objets auront le même Name, assorti d'un Index)
- Exemple de code associé :
For i = 0 to 3
CaseCarrée(i).Value = 0
Next i

Groupes de contrôles (2)

- Une seule procédure pour l'ensemble des contrôles du groupe
- Exemple

Private Sub CaseCarrée_Click (Index as Integer)

End Sub

- Il suffit d'utiliser Index pour savoir quelle case est cliquée

Les listes (1)

- Une liste peut-être :
 - Modifiable ou pas : l'utilisateur peut, ou pas, ajouter un élément qui n'est pas dans la liste
 - Déroulante ou pas : l'utilisateur doit cliquer pour dérouler, ou pas, la liste.
- 2 types de listes : **ListBox** (non déroulante) et **ComboBox** (déroulante)
- L'option AllowDrop permet de spécifier si l'utilisateur peut modifier la liste ou pas.

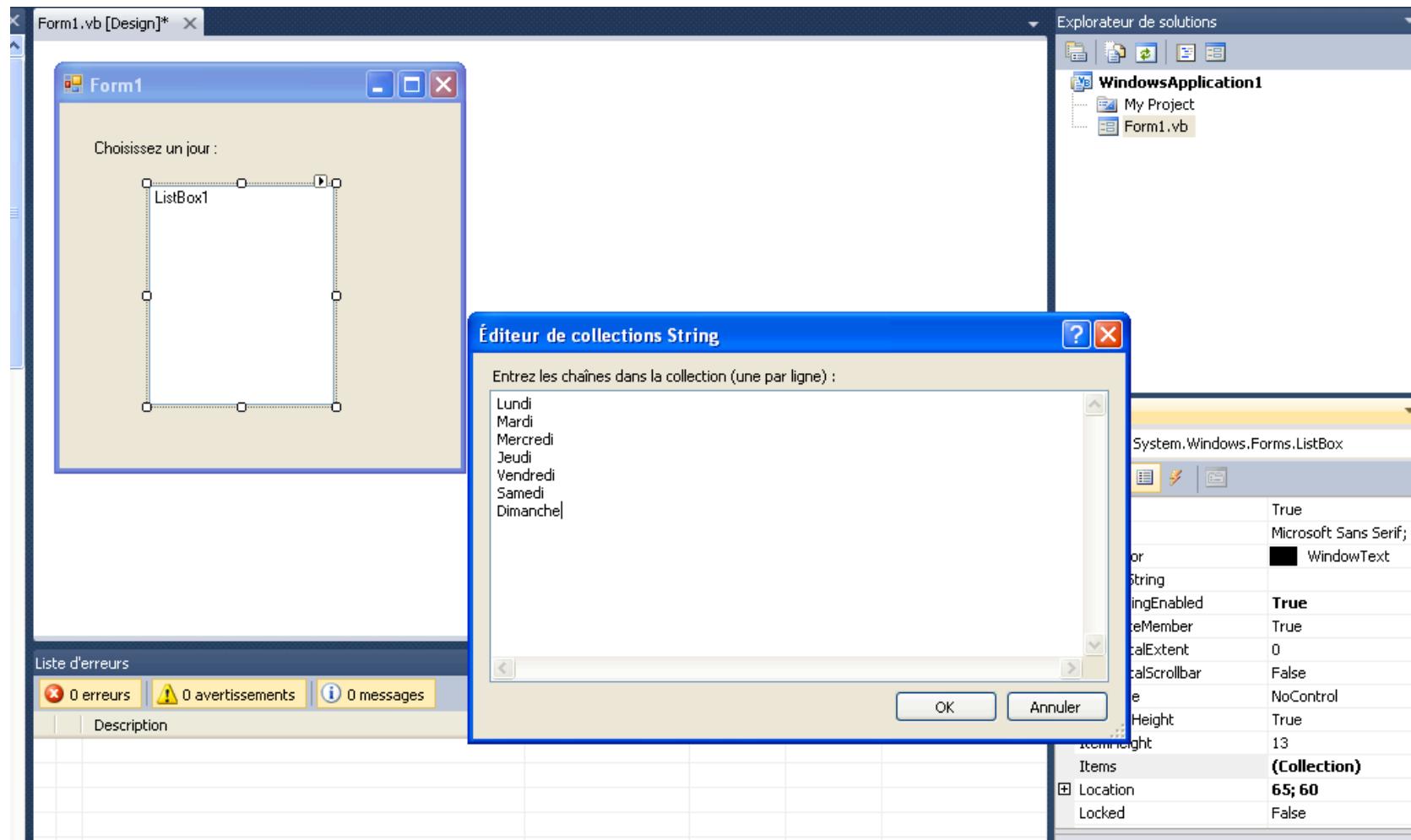
Les listes (2)

- Propriétés indispensables
 - **SelectedIndex** : renvoie ou définit l'indice de l'élément actuellement sélectionné. Vaut 0 par défaut.
 - **SelectedItem** : renvoie le texte correspondant à l'élément sélectionné.
- Autres propriétés intéressantes
 - **ListCount**, qui renvoie le nombre d'éléments d'une liste (propriété numérique)
 - **Multiselect**, qui permet la sélection multiple (propriété booléenne)
 - **Sorted**, qui trie automatiquement les éléments d'une liste (propriété booléenne)

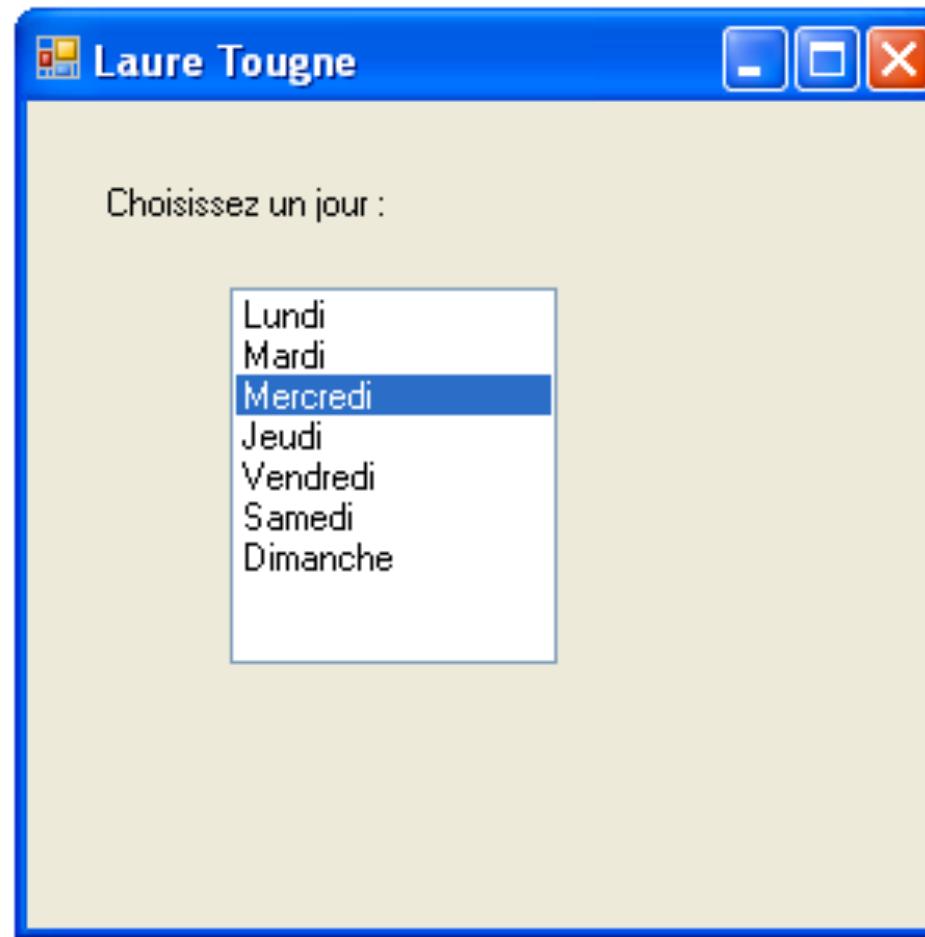
Les listes (3)

- Méthodes à connaître
 - **AddItem Chaîne** : ajoute l'élément Chaîne à une liste (un argument supplémentaire, facultatif, permet éventuellement de spécifier à quel indice l'élément doit être inséré).
 - **RemoveItem (indice)** : supprime de la liste l'élément possédant l'indice spécifié.
 - **Clear** : efface tous les éléments d'une liste

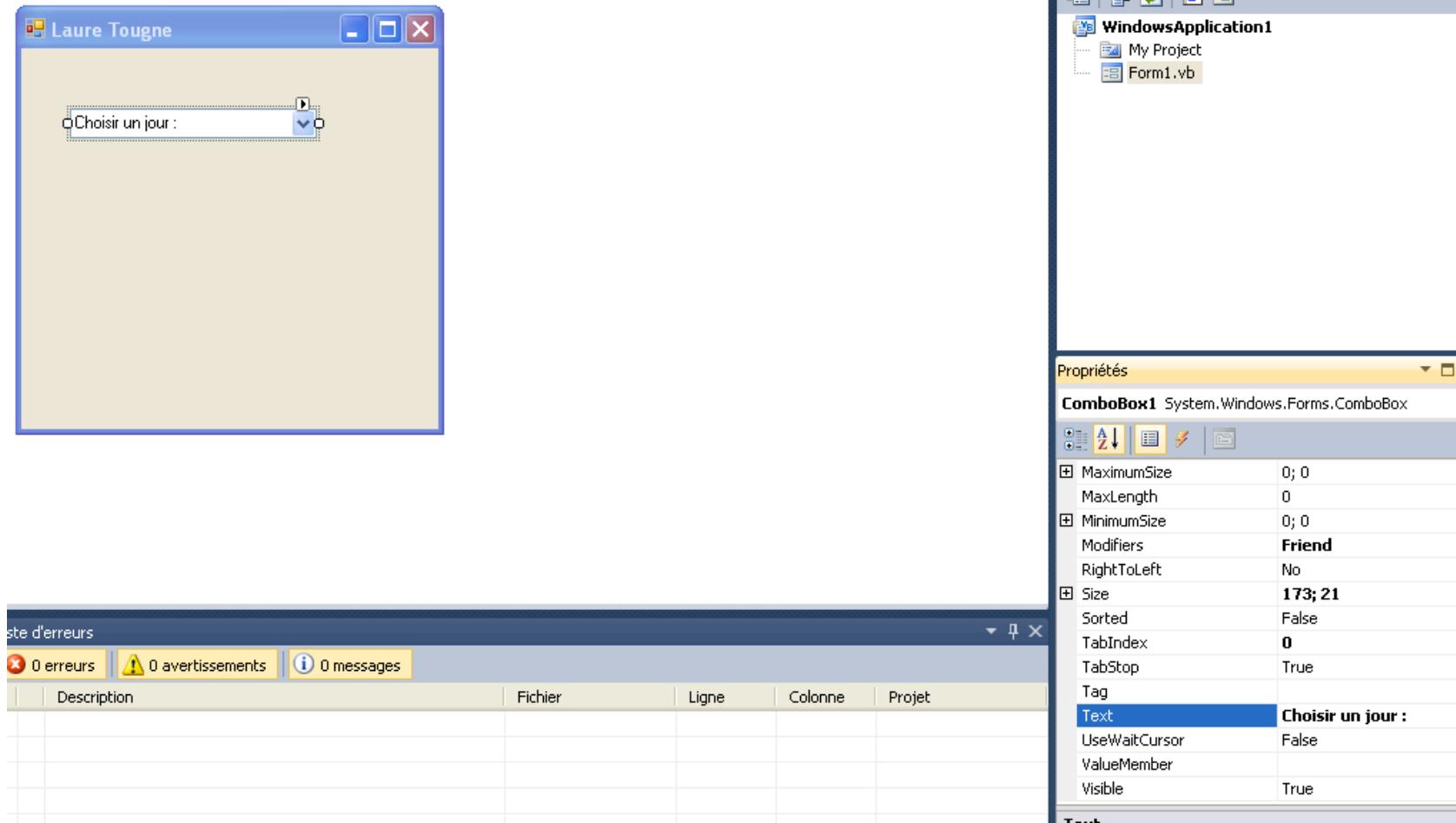
Un exemple de ListBox



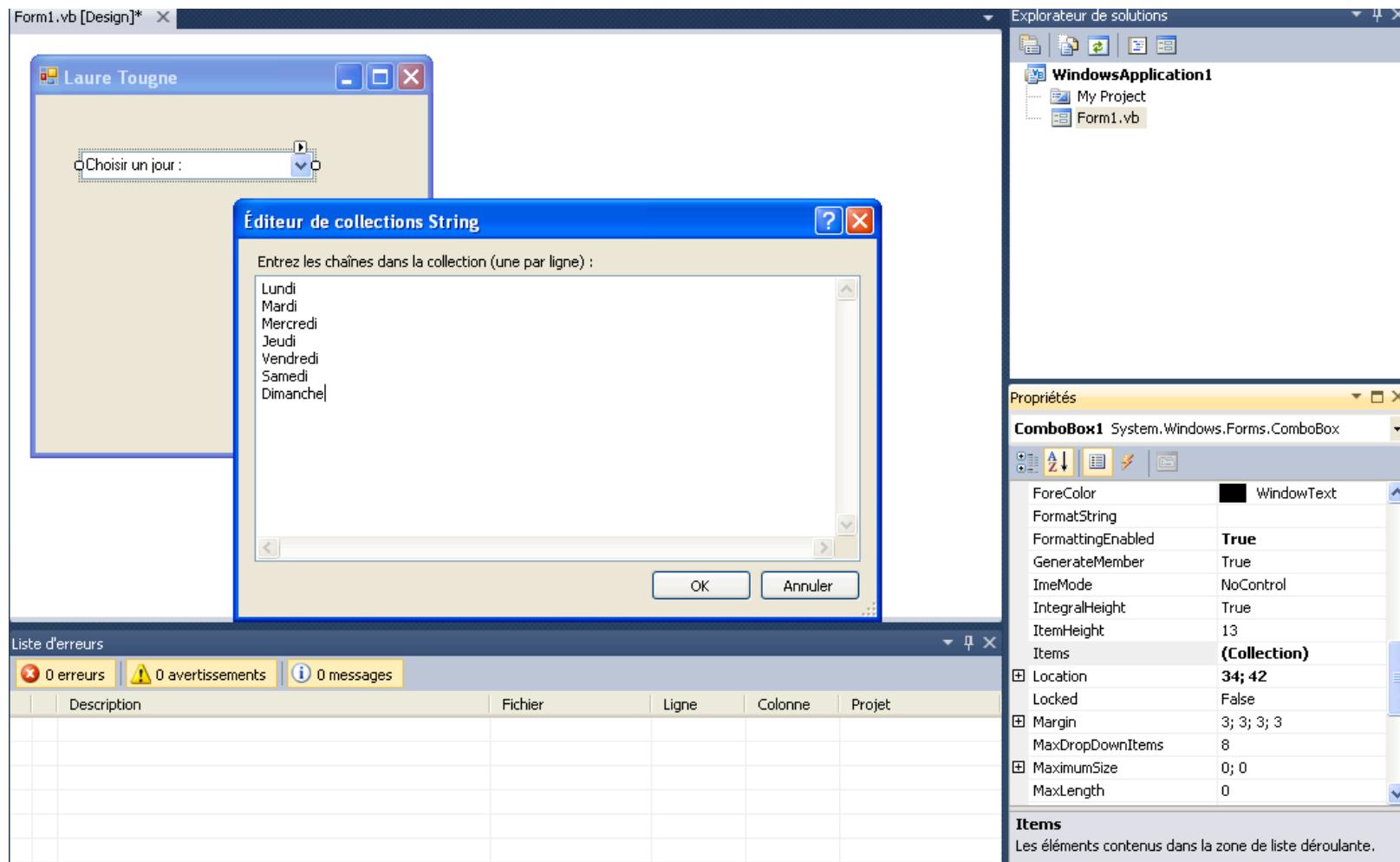
Fonctionnement de la ListBox



Un exemple de ComboBox (1)



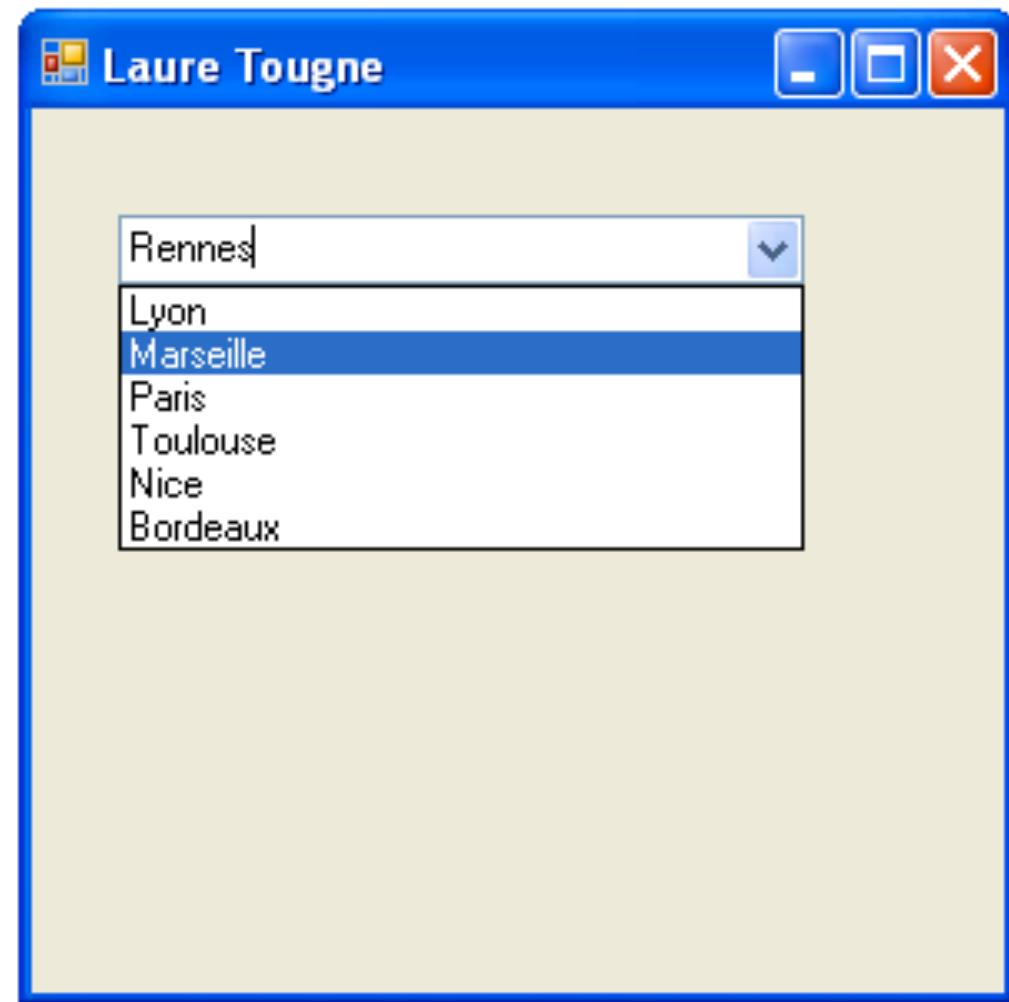
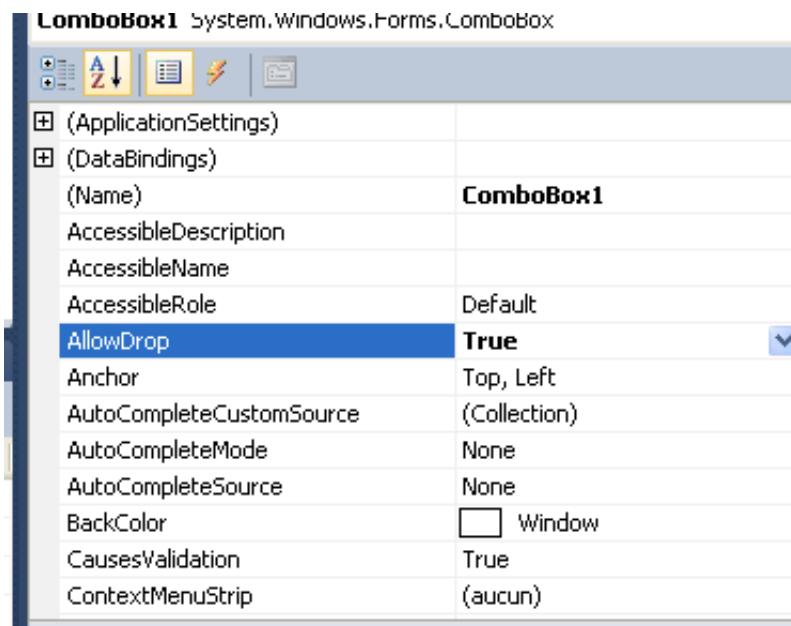
Un exemple de ComboBox (2)



Un exemple de ComboBox (3)



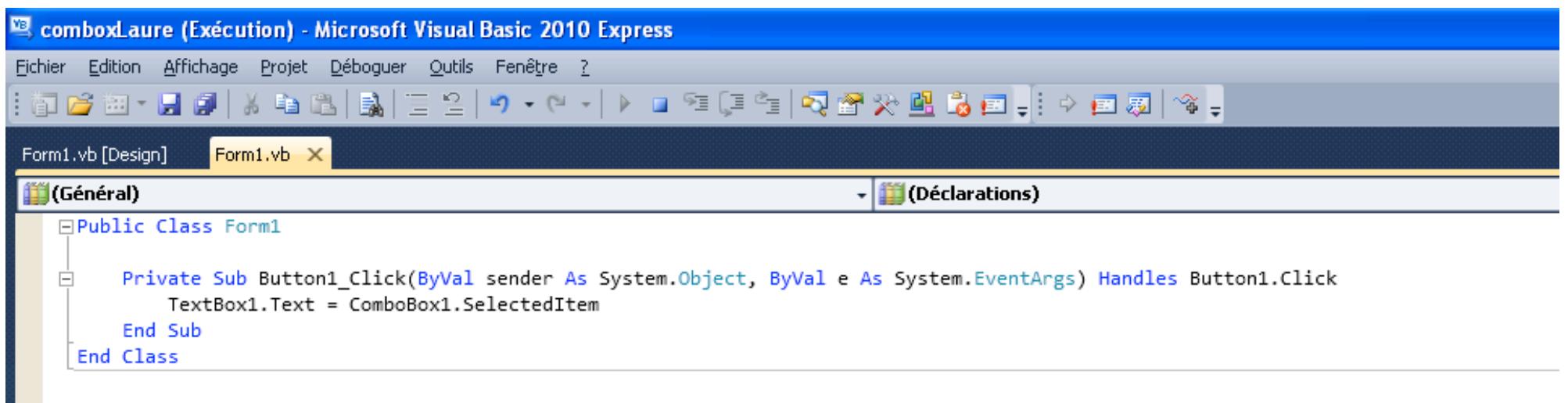
Un exemple de ComboBox (4)



Un exemple de ComboBox (5)



Un exemple de ComboBox (6)



The screenshot shows the Microsoft Visual Basic 2010 Express IDE. The title bar reads "comboxLaure (Exécution) - Microsoft Visual Basic 2010 Express". The menu bar includes Fichier, Edition, Affichage, Projet, Déboguer, Outils, Fenêtre, and ?.

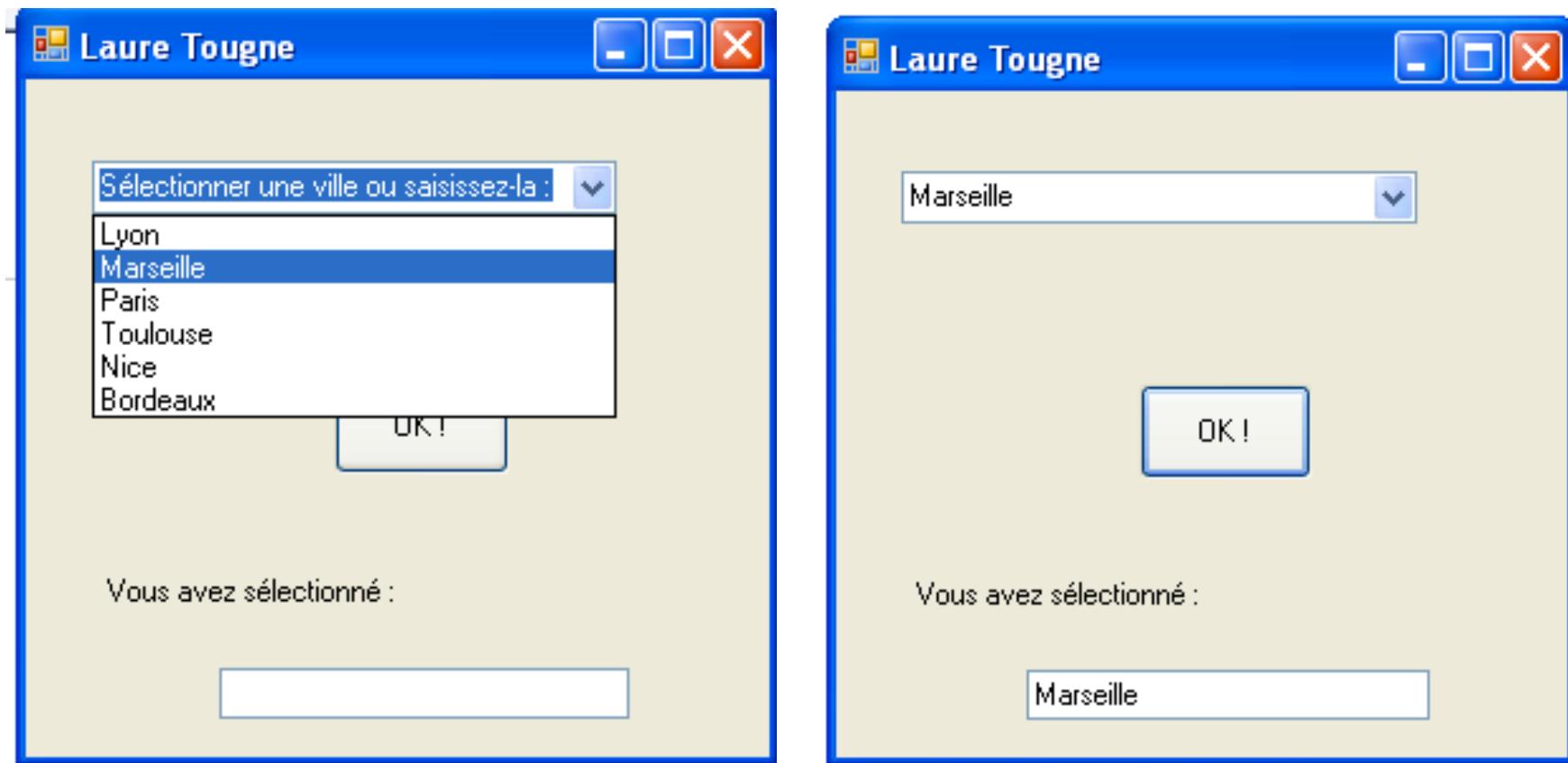
The toolbar below the menu bar contains various icons for file operations, such as Open, Save, Print, and Build.

The main window displays the code for Form1.vb [Design]. The code is as follows:

```
Public Class Form1
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        TextBox1.Text = ComboBox1.SelectedItem
    End Sub
End Class
```

The code defines a class named Form1 with a single event handler for the Button1_Click event. The event handler sets the text of a TextBox1 control to the selected item of a ComboBox1 control.

Un exemple de ComboBox (7)

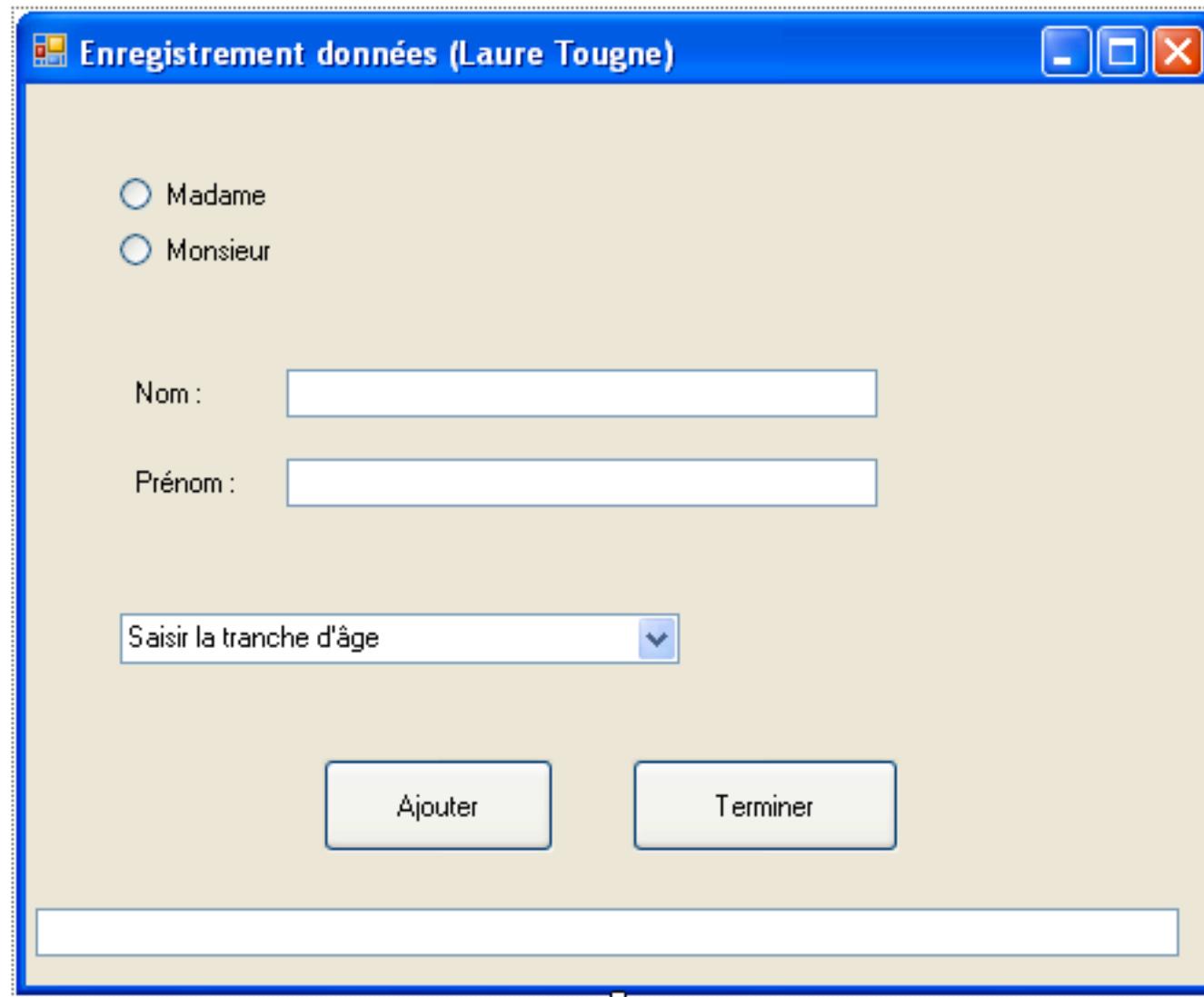


UN EXEMPLE PAS À PAS

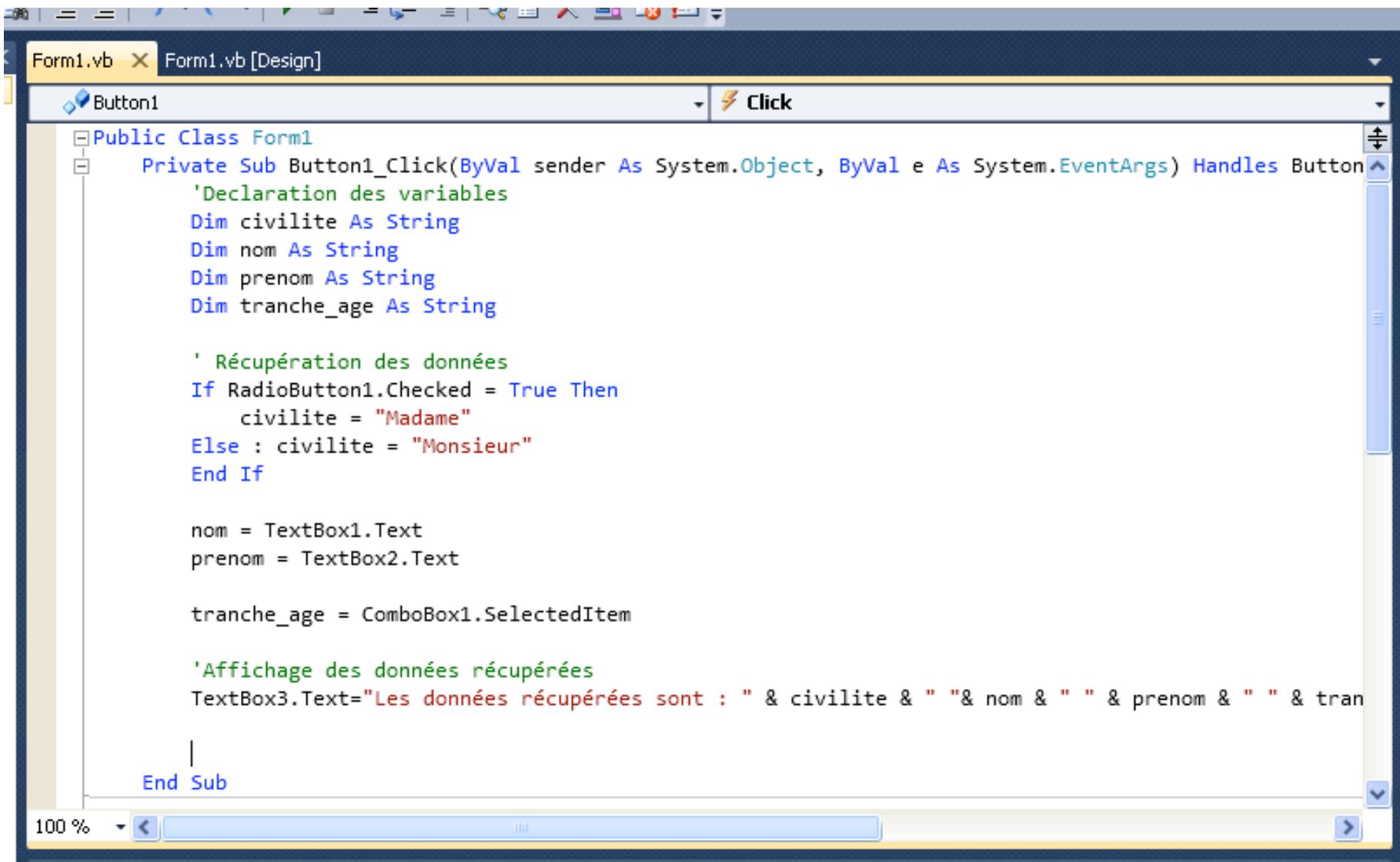
Enoncé

- On se propose de créer un programme pour enregistrer des coordonnées d'individus dans un fichier.
- L'interface devra comprendre au moins :
 - 2 boutons (exclusifs) permettant de sélectionner « Madame » ou « Monsieur »
 - 2 zones de texte permettant la saisie respectives des noms et prénoms
 - 1 liste déroulante, non modifiable, permettant de sélectionner une tranche d'âge « Moins de 10 ans », « 10-25 », « 25-50 », « 50-75 », « Plus de 75 »
 - Un bouton « Ajouter » pour enregistrer l'individu dans le fichier et effacer les 2 zones de texte
 - Un bouton « Terminer » pour quitter le programme

Etape 1 : création de l'interface



Etape 2 : début du code



```
Form1.vb [Design] | Click
Button1
Public Class Form1
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        'Déclaration des variables
        Dim civilite As String
        Dim nom As String
        Dim prenom As String
        Dim tranche_age As String

        ' Récupération des données
        If RadioButton1.Checked = True Then
            civilite = "Madame"
        Else : civilite = "Monsieur"
        End If

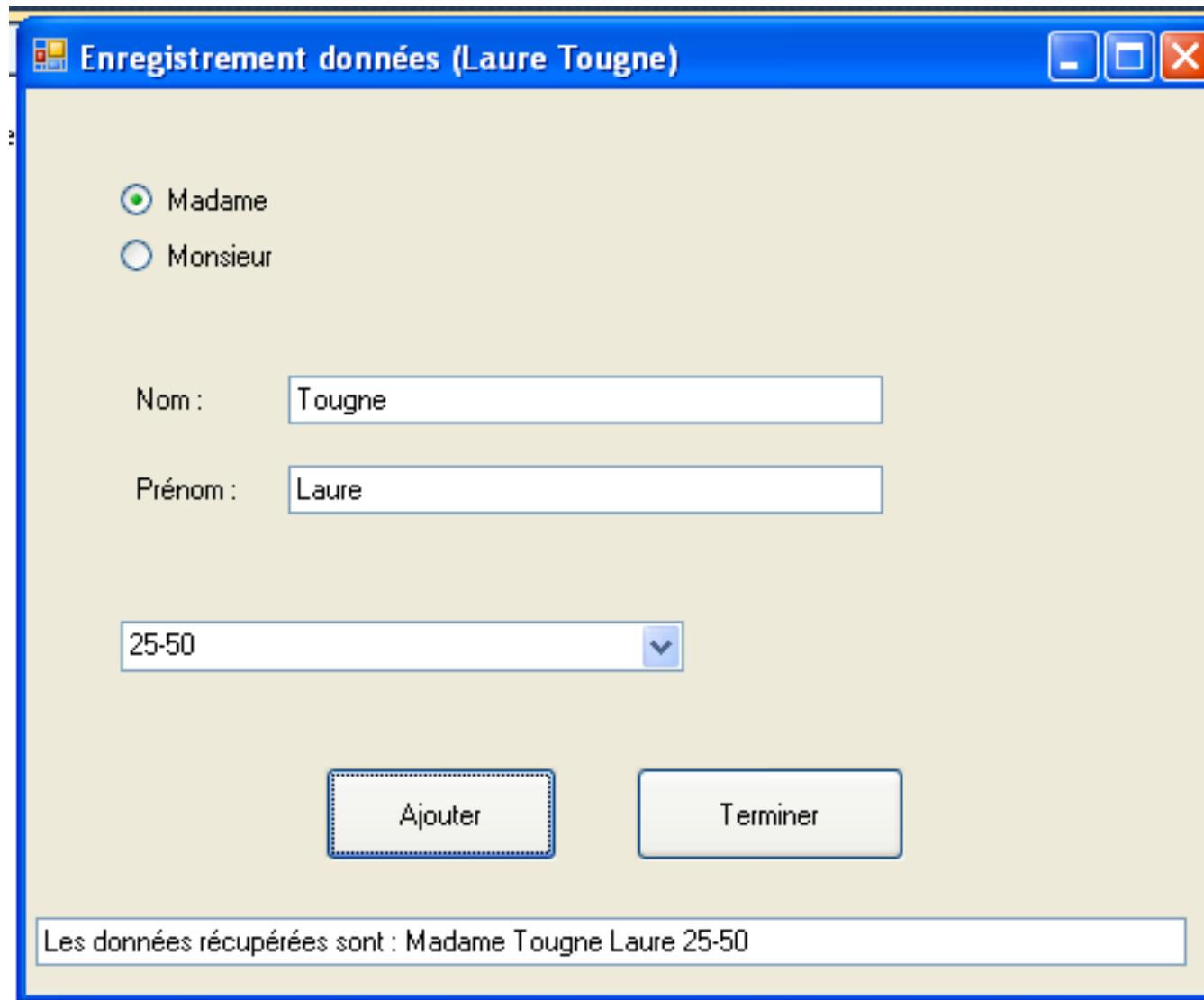
        nom = TextBox1.Text
        prenom = TextBox2.Text

        tranche_age = ComboBox1.SelectedItem

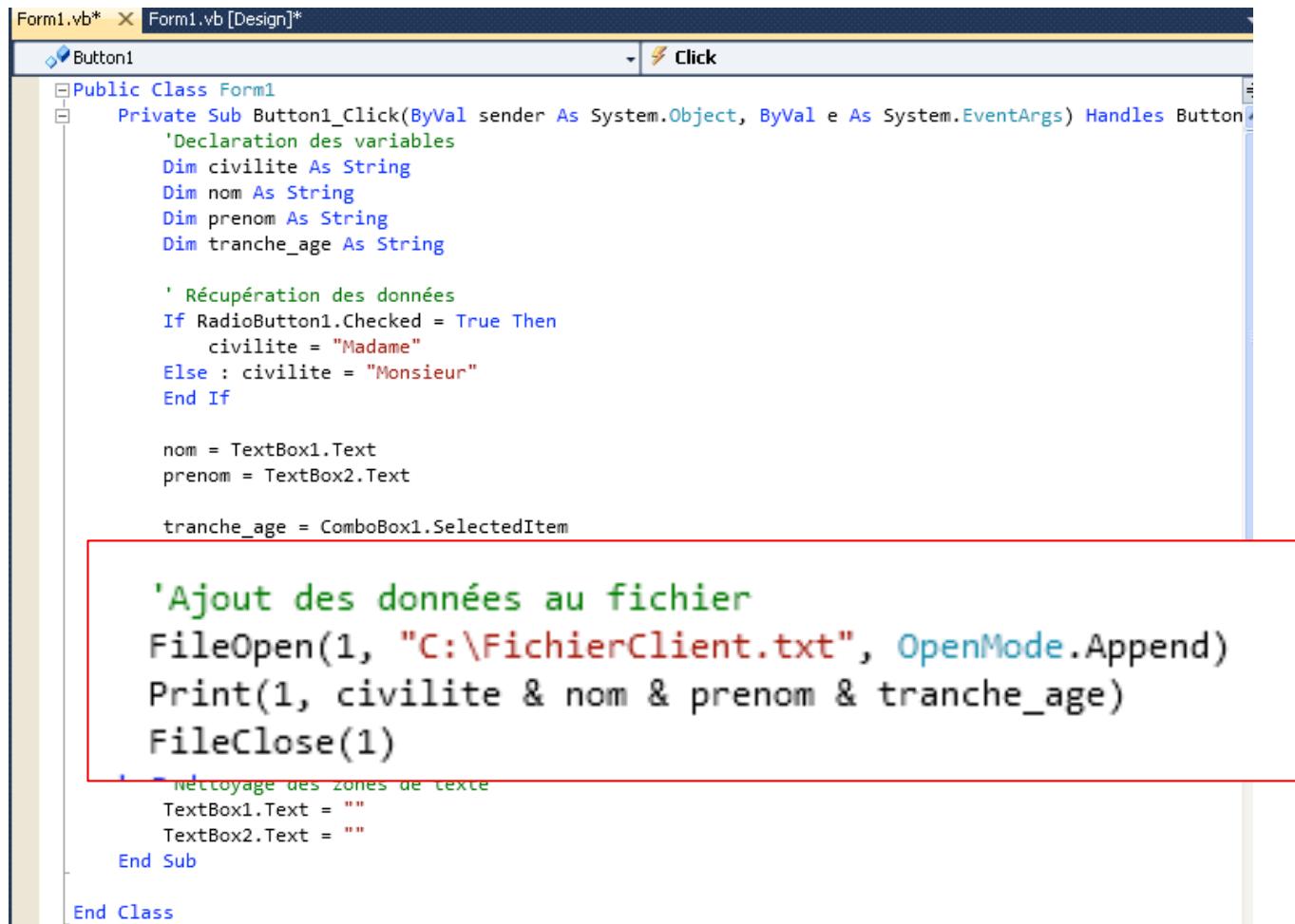
        'Affichage des données récupérées
        TextBox3.Text = "Les données récupérées sont : " & civilite & " " & nom & " " & prenom & " " & tranche_age
    End Sub

```

Etape 3 : fonctionnement (1)



Etape 4 : enregistrement dans le fichier



The screenshot shows the Microsoft Visual Studio IDE with the code editor open. The title bar says "Form1.vb* Form1.vb [Design]*". The code is written in VB.NET. A red box highlights the section of code that handles the file operations:

```
Public Class Form1
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        'Déclaration des variables
        Dim civilite As String
        Dim nom As String
        Dim prenom As String
        Dim tranche_age As String

        ' Récupération des données
        If RadioButton1.Checked = True Then
            civilite = "Madame"
        Else : civilite = "Monsieur"
        End If

        nom = TextBox1.Text
        prenom = TextBox2.Text

        tranche_age = ComboBox1.SelectedItem

        'Ajout des données au fichier
        FileOpen(1, "C:\FichierClient.txt", OpenMode.Append)
        Print(1, civilite & nom & prenom & tranche_age)
        FileClose(1)

        ' Nettoyage des zones de texte
        TextBox1.Text = ""
        TextBox2.Text = ""
    End Sub
End Class
```

→ Nous y reviendrons plus tard...

Etape 5 :

```
Form1.vb*  Form1.vb [Design]*
```

```
Button1 Click
```

```
Public Class Form1
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        'Déclaration des variables
        Dim civilite As String
        Dim nom As String
        Dim prenom As String
        Dim tranche_age As String

        ' Récupération des données
        If RadioButton1.Checked = True Then
            civilite = "Madame"
        Else : civilite = "Monsieur"
        End If

        nom = TextBox1.Text
        prenom = TextBox2.Text

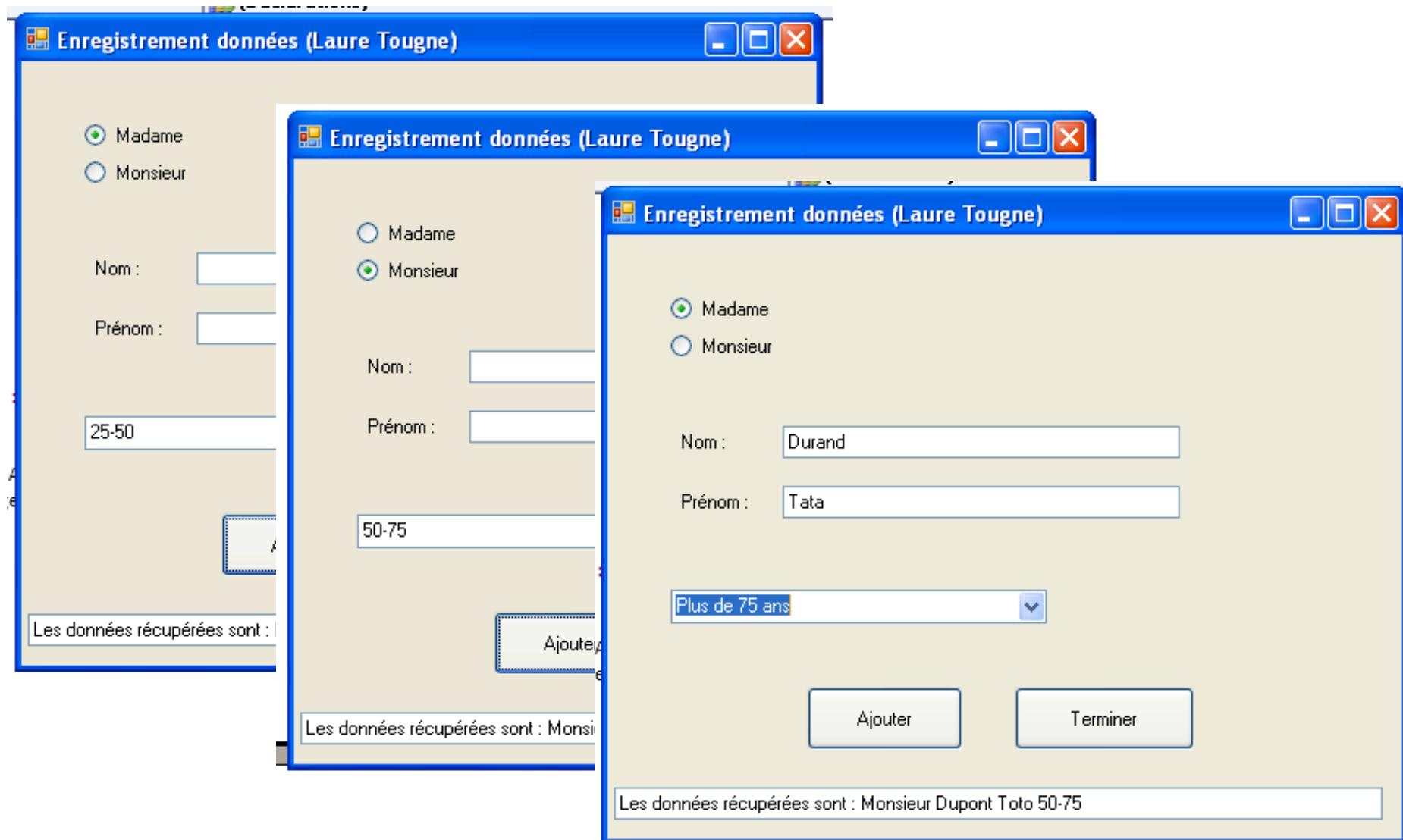
        tranche_age = ComboBox1.SelectedItem

        'Affichage des données récupérées
        TextBox3.Text = "Les données récupérées sont : " & civilite & " " & nom & " " & prenom & " " & t

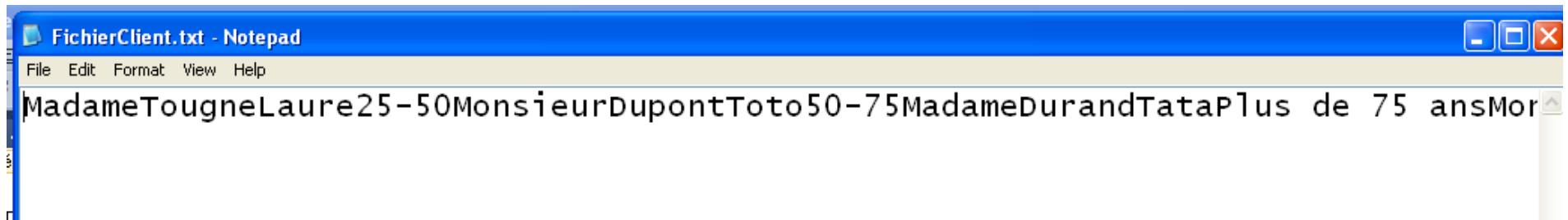
        'Ajout des données au fichier
        'Nettoyage des zones de texte
        TextBox1.Text = ""
        TextBox2.Text = ""

    End Sub
End Class
```

Etape 6 : quelques étapes



Etape 7 : fichier résultat



Remarques

- Ce n'est pas la méthode la plus optimale : ouverture et fermeture du fichier à chaque fois...
- Nous verrons plus tard comment gérer des variables globales et faire des procédures utilisables dans les procédures événementielles...

FONCTIONS ET PROCÉDURES PERSONNALISÉES

Introduction

- Jusque-là nous n'avons vu que les événements par défaut associés aux contrôles, et les procédures associées à ces événements.
- Mais...
 - il existe d'autres événements...
 - Et surtout la possibilité de faire vos propres procédures (ou fonctions)...
- L'objectif ici est de voir comment définir vos propres procédures et fonctions.

Intérêt des fonctions et procédures personnalisées

- Permet de découper un problème en sous-problèmes
- Permet de ré-utiliser éventuellement certaines parties de codes à partir de différents modules
- Certaines seront publiques et d'autres privées :

Sub exemple()

...

End Sub

ou

Public Sub exemple()

...

End Sub

Private Sub exemple()

...

End Sub

Procédure / Fonction

- La différence principale entre une procédure (Sub) et une fonction (function) est la valeur renournée par la fonction.
- Exemple :

The screenshot shows two windows side-by-side. The left window displays a portion of a Visual Studio code editor with C# syntax. The right window shows a standard Windows message box titled "Windows Application" with the message "Attention !!!" and an "OK" button.

Top Example (Procedure):

```
Public Class Form1
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        avertissement()
    End Sub
    Private Sub avertissement()
        MsgBox("Attention !!!")
    End Sub
End Class
```

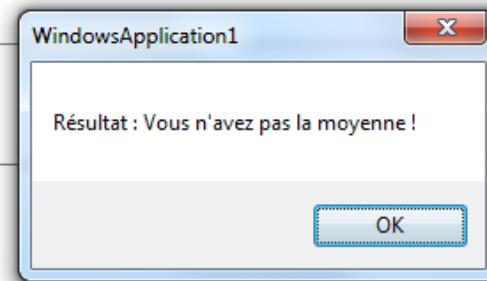
Bottom Example (Function):

```
Public Class Form1
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        Dim res As Double
        res = carre(5.84)
        MsgBox(res)
    End Sub
    Function carre(ByVal nombre)
        carre = nombre ^ 2
    End Function
End Class
```

The bottom example demonstrates a function named "carre" which takes a parameter "nombre" and returns its square. When the application runs, it displays a message box with the value "34,1056".

Les arguments

- Ils permettent d'utiliser des valeurs d'une procédure dans une sous-procédure ou dans une fonction.
- Exemple :

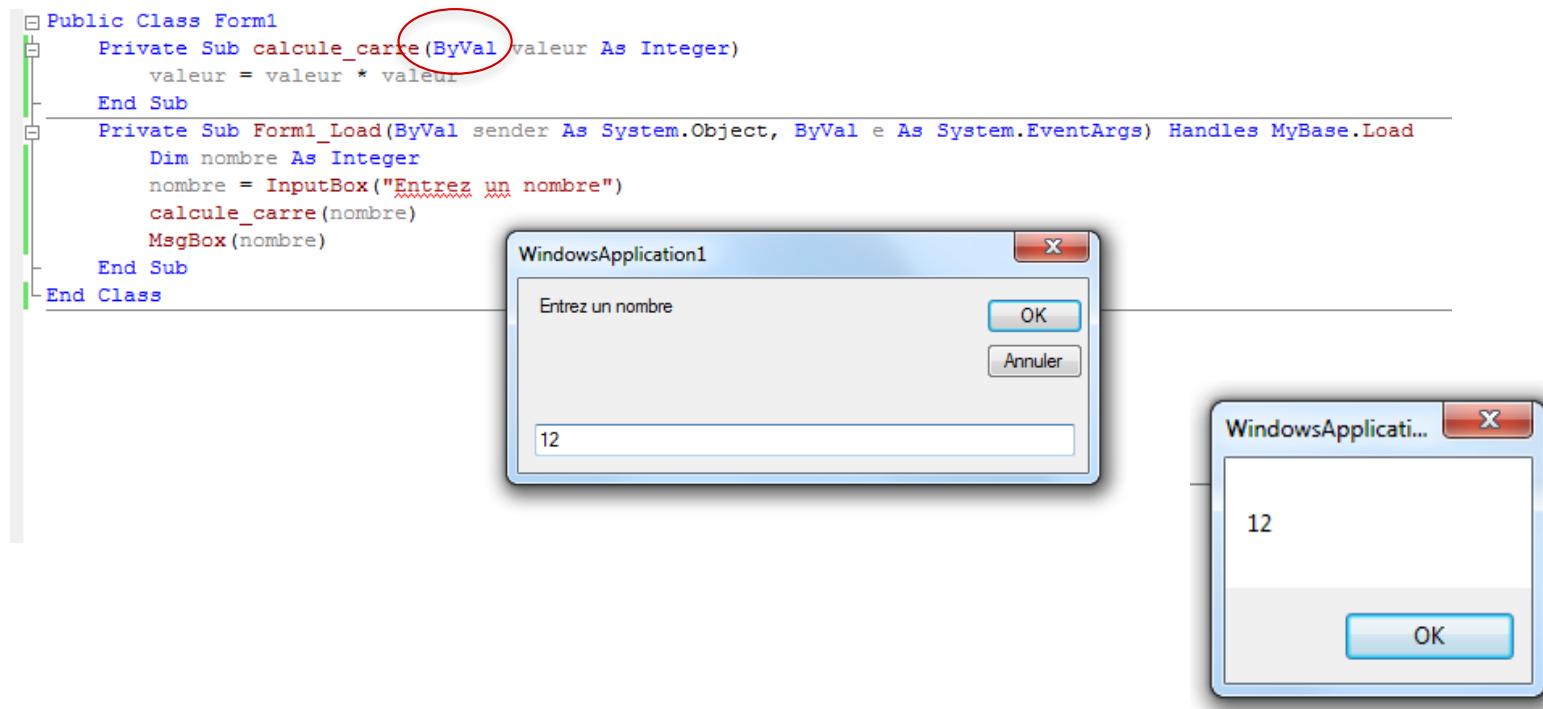


```
Public Class Form1

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        Dim moyenne As Double
        moyenne = InputBox("Entrez votre moyenne")
        If moyenne < 10 Then
            notification("Vous n'avez pas la moyenne")
        Else
            notification("Vous avez la moyenne")
        End If
    End Sub
    Private Sub notification(ByVal texte As String)
        MsgBox("Résultat : " & texte & " !")
    End Sub
End Class
```

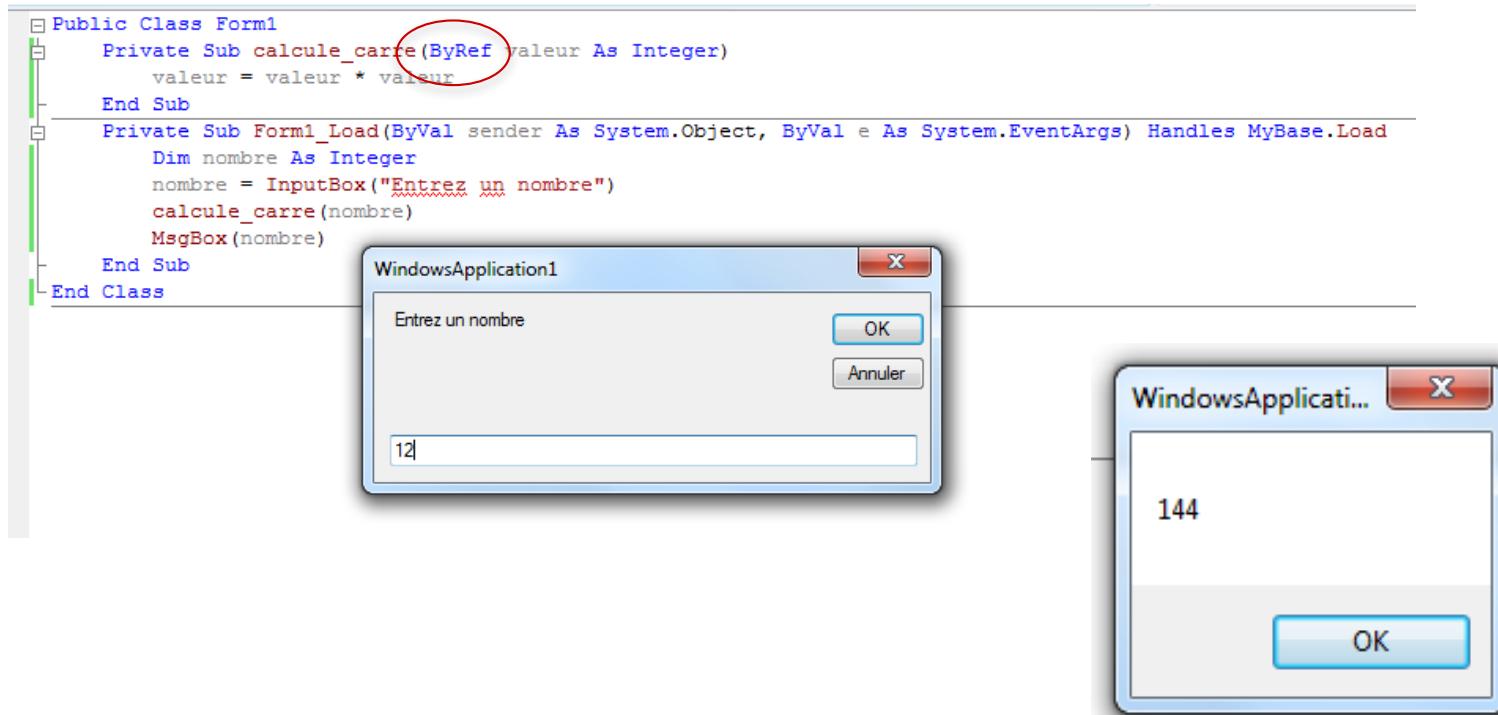
Arguments par référence / par valeur (1)

- Par défaut, un argument est passé par valeur (**ByVal**) : sa valeur est inchangée en dehors de la procédure.
- Exemple :



Arguments par référence / par valeur (2)

- Pour que la valeur soit transmise en dehors de la procédure, il faut utiliser **ByRef**
- Exemple :



Les fonctions

- Un autre moyen de transmettre une valeur est d'utiliser une fonction.
- Exemple :

The screenshot shows a Windows application interface. On the left, the code editor displays the following VB.NET code:

```
Public Class Form1
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        Dim nombre As Double
        Dim nombre_au_cube As Double
        nombre = InputBox("Entrez un nombre")
        nombre_au_cube = cube(nombre)
        MsgBox("Le cube de " & nombre & " est " & nombre_au_cube)
    End Sub
    Function cube(ByVal nombre)
        cube = nombre * nombre * nombre
    End Function
End Class
```

Two lines of code are circled in red: the function declaration 'Function cube(ByVal nombre)' and the assignment statement 'cube = nombre * nombre * nombre'. To the right of the code editor, there are two overlapping message boxes. The top message box is titled 'WindowsApplication1' and contains the instruction 'Entrez un nombre' with an input field containing the value '3'. The bottom message box is also titled 'WindowsApplication1' and contains the message 'Le cube de 3 est 27' with an 'OK' button.