Unit 03

WORKING WITH FORMS

Content

1. Appearance of forms
2. Form properties
3. Designing menu structure
4. Building dynamic forms at run time
5. Introduction to MDI forms.

Introduction to Form

In Visual Basic, the *form* is the container for all the controls that make up the user interface. When a Visual Basic application is executing, each window it displays on the desktop is a form.

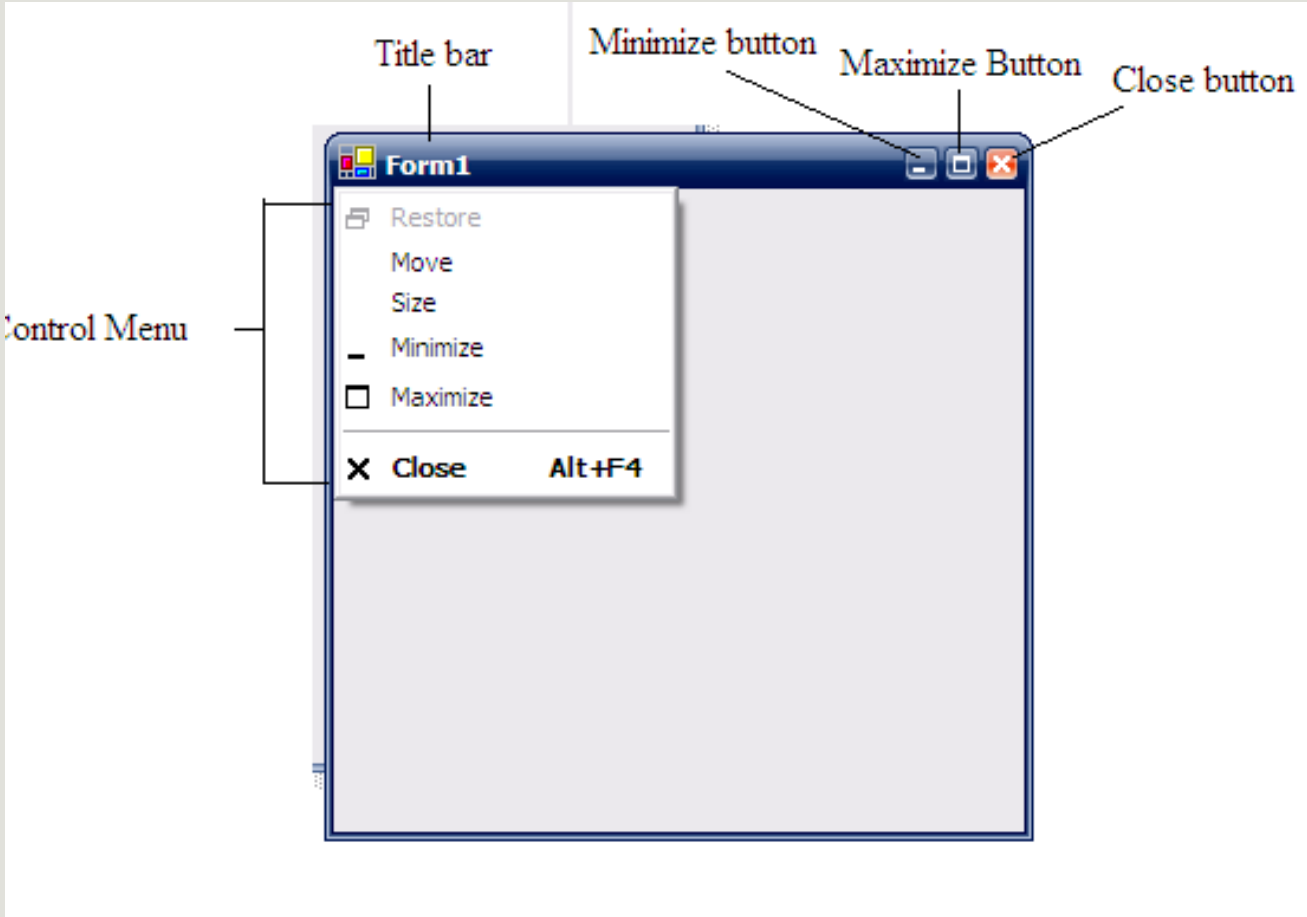
The form is the top-level object in a Visual Basic application, and every application starts with the form. Forms have built-in functionality that is always available without any programming effort on your part. You can move a form around, resize it, and even cover it with other forms. You do this with the mouse, or with the keyboard through the Control menu.

The forms that constitute the visible interface of your application are called ***Windows forms;*** this term includes both the regular forms and dialog boxes, which are simple forms you use for very specific actions, such as to prompt the user for a specific piece of data or to display critical information.

A ***dialog box*** is a form with a small number of controls, no menus, and usually an OK and a Cancel button to close it.

Appearance of forms

Applications are made up of one or more forms, and the forms are what users see. You should create your forms carefully, make them functional, and keep them simple and spontaneous. The main characteristic of a form is the title bar on which the form's caption is displayed (see Figure).



Appearance of forms

Title bar opens the Control menu, which contains the commands shown in Table

Table Commands of the Control Menu

Command	**Effect**
Restore	Restores a maximized form to the size it was before it was maximized; available only if the form has been maximized
Move	Lets the user move the form around with the mouse
Size	Lets the user resize the form with the mouse
Minimize	Minimizes the form
Maximize	Maximizes the form
Close	Closes the current form

Forms Properties

Some important properties of form are as follows.

Name:

Name used to identify form. It is the name by which the form is referred to in your code. Visual Basic by default names the forms Form1, Form2, Form3, etc. three letter prefix for form names is ***frm***

Appearance:

Return or set whether or not an object is painted at run time with 3D effects. There are two values of this property 0-Flat & 1- 3D. the default vale is 3D.

BackColor:

This property is used to sets the form background color.

BorderStyle

The BorderStyle property determines the style of the form's border and the appearance of the form; it takes one of the values shown in Table

Value	Effect
None	Borderless window that can't be resized; this setting should be avoided.
Sizable (default)	Resizable window that's used for displaying regular forms.
FixedDialog	A fixed window, used to create dialog boxes.
FixedSingle	A fixed window with a single line border.
FixedToolWindow	A fixed window with a Close button only. It looks like the toolbar displayed by the drawing and imaging applications.
SizableToolWindow	Same as the FixedToolWindow but resizable. In addition, its caption font is smaller than the usual.

Forms Properties

Caption:

It is the text that appears in the title bar of form. A caption can be changed at runtime. The caption must be informative and meaningful.

ControlBox:

This property is also True by default. Set it to False to hide the icon and disable the Control menu. Although the Control menu is rarely used, Windows applications don't disable it. When the ControlBox property is False, the three buttons on the title bar are also disabled.

Enabled:

If True, allows the form to respond to mouse and keyboard events; if False, disables form. Default value is true.

Font:

This property is used to sets font type, style, size.

ForeColor:

This property is used to sets color of text or graphics.

Forms Properties

Height:

This property is used to sets the height of the form.

Left:

This property is used to sets the distance from left side of computer screen to left side of form.

Min Button, Max Button:

These two properties are True by default. Set them to False to hide the corresponding buttons on the title bar.

Movable:

The default value is true. If set to false, the form cannot be moved by the user during the runtime.

Picture:

This property is used to places a bitmap picture in the form.

Forms Properties

StartUpPosition:

This property determines the initial position of the form when it's first displayed; it can have one of the values shown in Table 3.3.

Table 3.3: The FormStartPosition Enumeration

Value	Effect
Manual	The location and size of the form will determine its starting position.
CenterOwner	The form is centered in the area of its parent form.
CenterScreen	The form is centered on the monitor.
WindowsDefaultBounds	The form is positioned at the default location and size determined by Windows.

Forms Properties

Top:

This property is used to sets the distance from top side of computer screen to top side of form.

Visible:

This property is used to sets a value that determines whether an form is visible or hidden. Default value is true.

Width:

This property is used to sets the width of the form.

Designing menu structure

Menus are one of the most common and characteristic elements of the Windows user interface. Even in the old days of character-based displays, menus were used to display methodically organized choices and guide the user through an application. Despite the visually rich interfaces of Windows applications and the many alternatives, menus are still the most popular means of organizing a large number of options. Many applications duplicate some or all of their menus in the form of icons on a toolbar, but the menu is a standard fixture of a Form. You can turn the toolbars on and off, but not the menus.

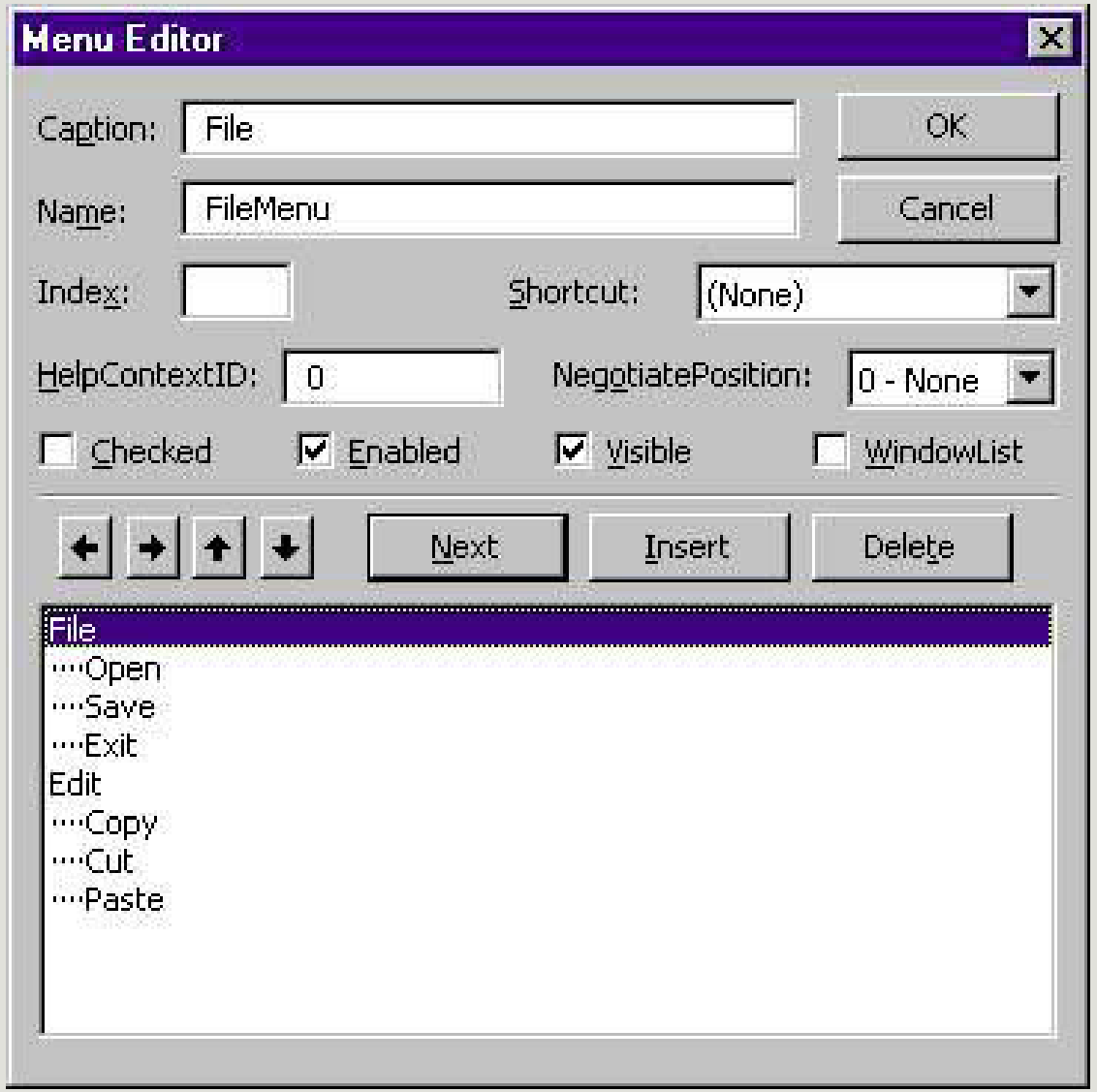
The Menu Editor

Menus can be attached only to Forms, and you design them with the Menu Editor. To see how the Menu Editor works,

start a new project, and when Form1 appears in the design window.

Choose Tools Ø Menu Editor to open the Menu Editor, as shown in following Figure.

Alternatively, you can click the Menu Editor Button on the toolbar.



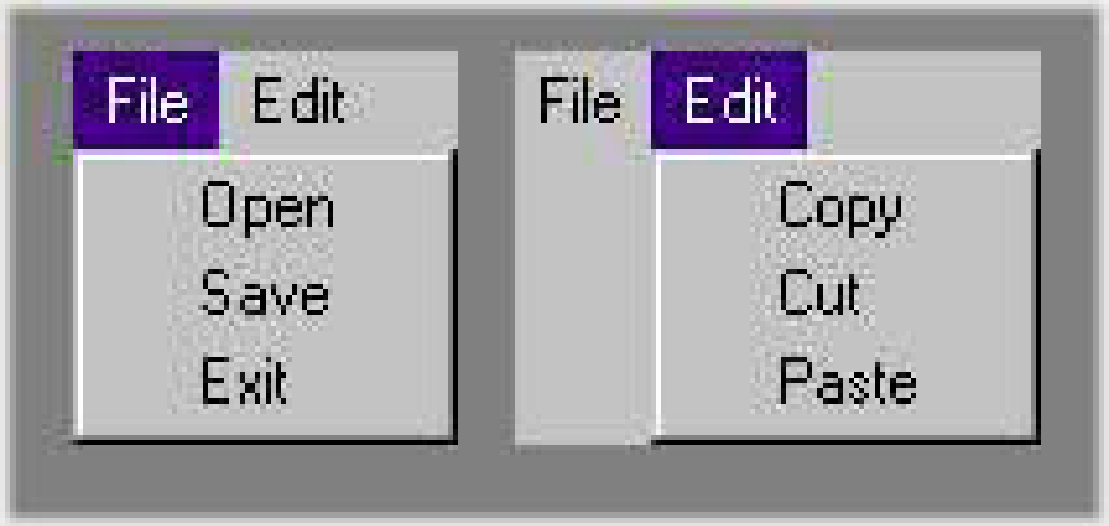
In the Menu Editor window you can specify the structure of your menu by adding one command at a time. Each menu command has two mandatory properties:

Caption This is the string that appears on the application's menu bar.

Name This is the name of the menu command. This property doesn't appear on the screen, but your code uses it to program the menu command.

The Caption and Name properties of a menu item are analogous to the properties that have the same name as the Command button or Label control. Caption is what the user sees on the Form, and Name is the means of accessing the control from within the code. As far as your code is concerned, each menu command is a separate object, just like a Command button or Label control.

To add commands to the Form's menu bar, enter a caption and a name for each command. As soon as you start typing the command's caption, it also appears in a new line in the list at the bottom of the Menu Editor window. Let's create the menu structure shown in above figure. This menu contains two commands, File and Edit. When the user clicks on either one, the submenus are displayed.



To design the menu, follow these steps:

1. Open a new Form in the Design pane, and choose Tools > Menu Editor to open the Menu Editor window.

2. In the Caption box, type the caption of the first command (**File**).

3. In the Name box, enter the command's name (**FileMenu**).

4. Press Enter or click Next to enter the next command. Repeat steps 1 through 3 for all the commands listed.

The Index Field

Use the Index field to create an array of menu commands. All the commands of the array have the same name and a unique index that distinguishes them, just like arrays of controls.

The Checked Field

Some menu commands act as toggles, and they are usually checked to indicate that they are on or unchecked to indicate that they are off. To display a checkmark next to a menu command initially, select the command from the list by clicking its name, and then check the Checked box in the Menu Editor window. You can also access this property from within your code to change the checked status of a menu command at runtime.

The Enabled Field

Some menu commands aren't always available. The Paste command, for example, has no meaning if the Clipboard is empty. To indicate that a command can't be used at the time, you set its Enabled property to False. The command then appears grayed in the menu, and although it can be highlighted, it can't be activated. You can set the initial status of a command by checking or clearing the Enabled box in the Menu Editor window. You can also toggle the status of a menu command from within your code by manipulating its Enabled property, similar to the Checked command.

The Visible Field

To remove a command temporarily from the menu, set the command's Visible property to False. The Visible property isn't used frequently in menu design. In general, you should prefer to disable a command to indicate that it can't be used (some other action is required to enable it). Making a command invisible frustrates users, who may try to locate the command in another menu.

Introduction to MDI forms.

The Multiple Document Interface (MDI) was designed to simplify the exchange of information among documents, all under the same roof. With an MDI application, you can maintain multiple open windows, but not multiple copies of the application. Data exchange is easier when you can view and compare many documents simultaneously.

We used Windows applications that can open multiple documents at the same time and allow the user to switch among them with a mouse-click. Microsoft Word is a typical example, although most people use it in single document mode. Each document is displayed in its own window, and all document windows have the same behavior.

The main form, or MDI Form, is not duplicated, but it acts as a container for all other windows, and it's called the ***parent window***. The windows in which the individual documents are displayed are called ***child windows*** (or document windows). When you reposition the parent window on the Desktop, its child windows follow.

Introduction to MDI forms.

Child windows, however, exist independently of the parent window. You can open and close child windows as you want, and child windows can even have different functions.

MDI applications aren’t very common; not too many applications lend themselves to MDI implementation. Most of them are easier to implement with multiple Forms, but some applications should be implemented with an MDI interface. These are the applications that can open multiple documents of the same type and use a common menu structure that applies to all open documents.

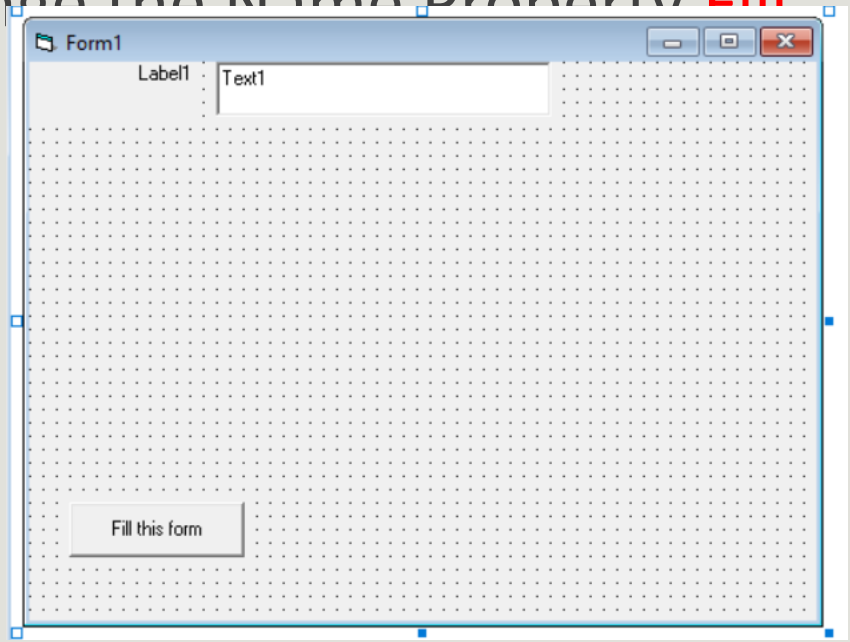
Building dynamic forms at run time

There are situations when you won't know in advance how many how many instance of a given control may be required on a form.

It is possible to design dynamic Forms, which are populated at runtime.

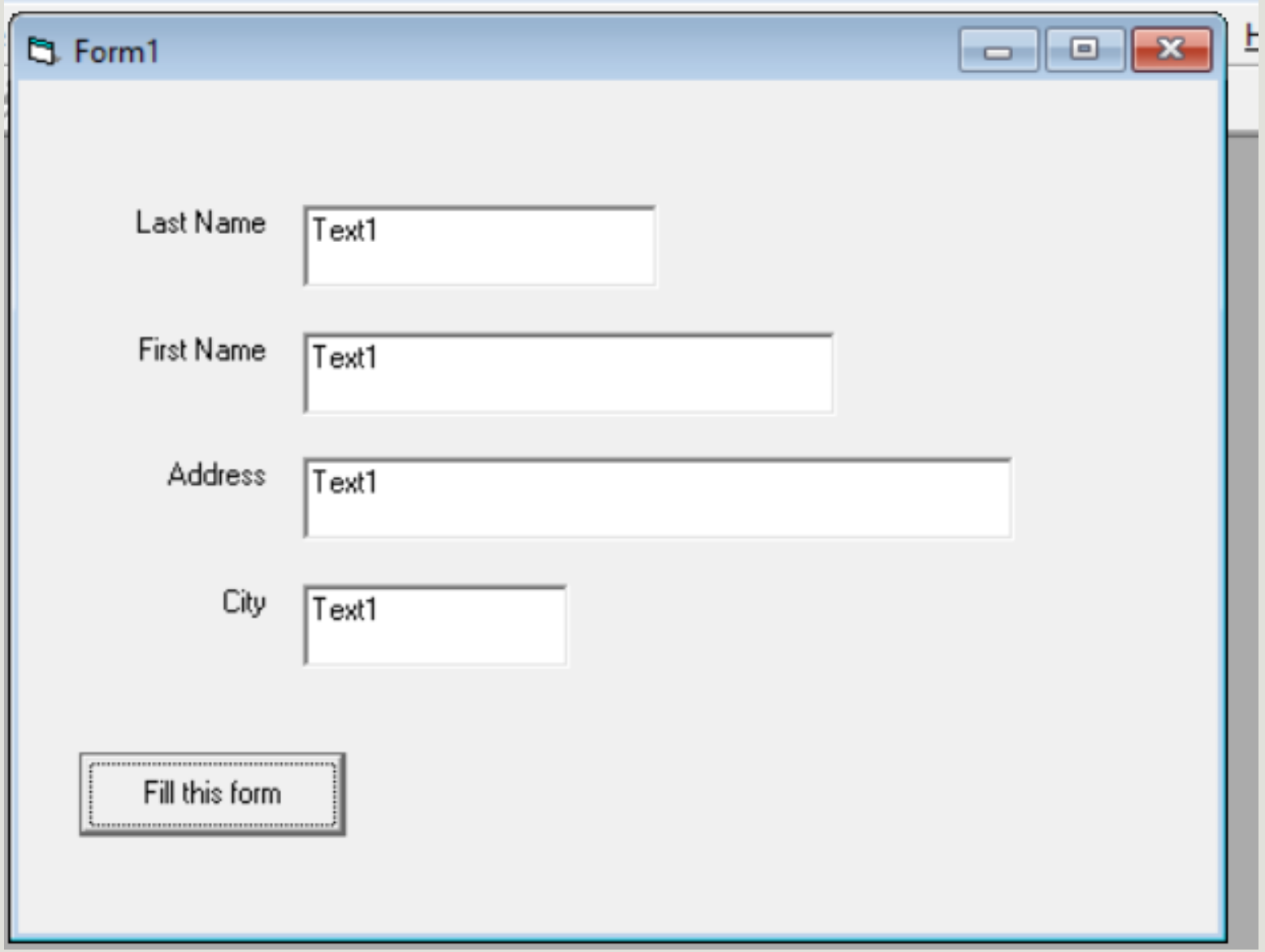
To Design the data entry form, You must place a label and a TextBox control on the form. Place them near the top of the Form, so that they can be used as guides for aligning the remaining controls. These two controls will remain invisible at all times.

1. Name the Label control Labels and set its Index Property to 0
2. Set its Alignment Property to 1(Right Justify)
3. Name the TextBox Control TextBoxes and set its Index to 0
4. Then Select both controls with the mouse and set their Visible property to False
5. Add a Command button on Form and change the Name Property Fill this Form



```
Private Sub Command1_Click()
Dim Captions(10) As String
Dim Sizes(10) As Integer
Captions(1) = "Last Name"
Captions(2) = "First Name"
Captions(3) = "Address"
Captions(4) = "City"

Sizes(1) = 20
Sizes(2) = 30
Sizes(3) = 40
Sizes(4) = 15
For i = 1 To 4
Load Labels(i)
Load TextBoxes(i)
Labels(i).Top = Labels(i - 1).Top + 1.5
* Labels(0).Height
Labels(i).Left = Labels(o).Left
TextBoxes(i).Top = TextBoxes(i -
1).Top + 1.5 * TextBoxes(i).Height
TextBoxes(i).Left = TextBoxes(0).Left
TextBoxes(i).Width = Sizes(i) *
TextWidth("A")
Labels(i).Caption = Captions(i)
Labels(i).Visible = True
TextBoxes(i).Visible = True
Next
End Sub
```


Form1
Last Name
Text1
First Name
Text1
Address
Text1
City
Text1
Fill this form