



Software Application Tutorial



Introduction To Visual Basic 6.0

Visual Basic 6.0

Welcome to Microsoft Visual Basic, the fastest and easiest way to create applications for Microsoft Windows. Whether you are an experienced professional or brand new to Windows programming, Visual Basic provides you with a complete set of tools to simplify rapid application development.

The "Visual" part refers to the method used to create the graphical user interface (GUI). Rather than writing numerous lines of code to describe the appearance and location of interface elements, you simply add prebuilt objects into place on screen. If you've ever used a drawing program such as Paint, you already have most of the skills necessary to create an effective user interface.

The "Basic" part refers to the BASIC (Beginners All-Purpose Symbolic Instruction Code) language, a language used by more programmers than any other language in the history of computing. Visual Basic has evolved from the original BASIC language and now contains several hundred statements, functions, and keywords, many of which relate directly to the Windows GUI. Beginners can create useful applications by learning just a few of the keywords, yet the power of the language allows professionals to accomplish anything that can be accomplished using any other Windows programming language.

There are three main steps to creating an application in Visual Basic:



1. Create the interface.
2. Set properties.
3. Write code.

To see how this is done, use the steps in the following procedures to create a simple application that consists of a text box and a command button. When you click the command button, the message "Hello, world!" appears in the text box.

Creating the Interface

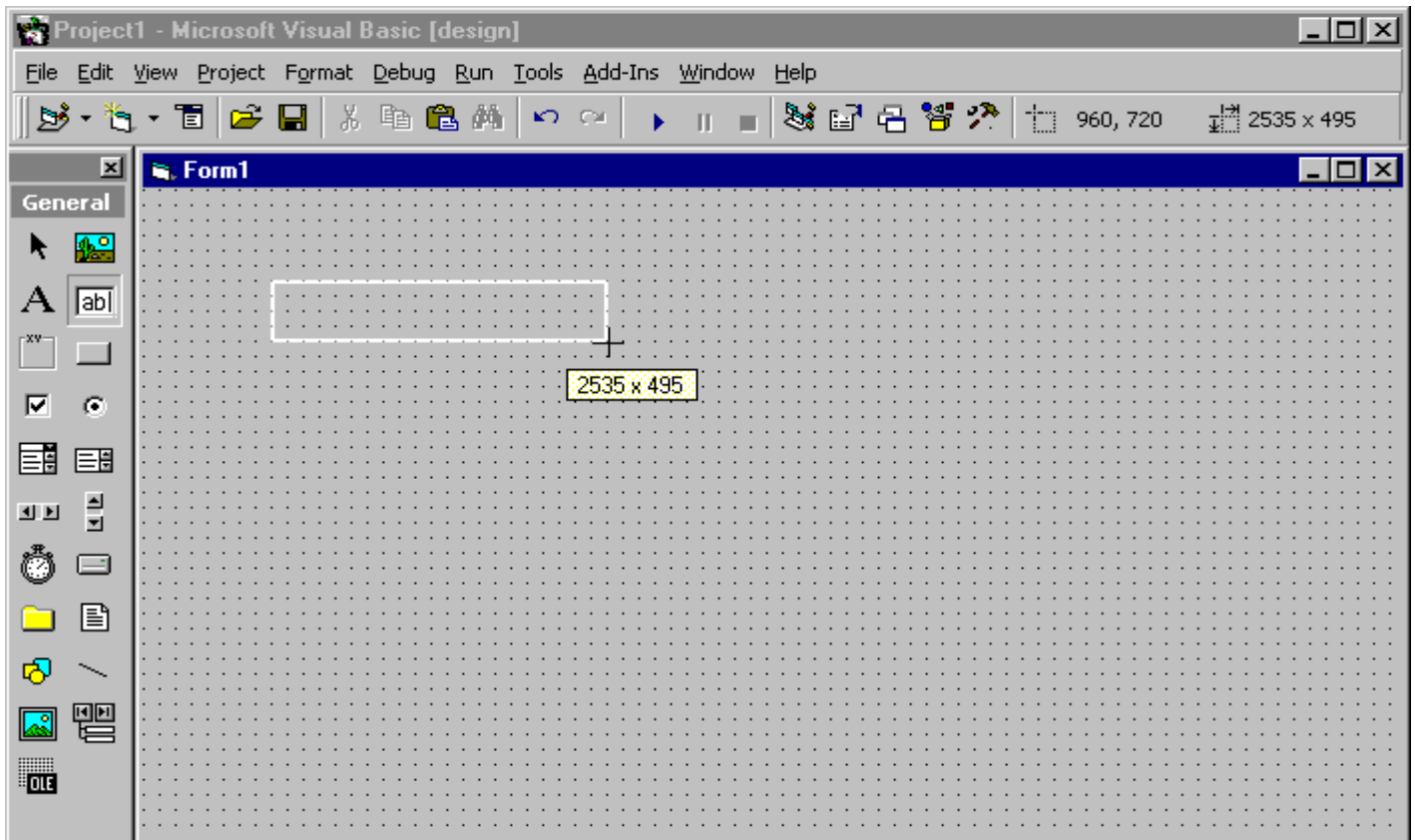
Forms are the foundation for creating the interface of an application. You can use forms to add windows and dialog boxes to your application. You can also use them as containers for items that are not a visible part of the application's interface. For example, you might have a form in your application that serves as a container for graphics that you plan to display in other forms.

The first step in building a Visual Basic application is to create the forms that will be the basis for your application's interface. Then you draw the objects that make up the interface on the forms you create. For this first application, you'll use two controls from the Toolbox.

Button	Control
	Text box
	Command button

To draw a control using the Toolbox

1. Click the tool for the control you choose to draw — in this case, the **text box**.
2. Move the pointer onto your form. The pointer becomes a cross hair.
3. Place the cross hair where you want the upper-left corner of the control.
4. Drag the cross hair until the control is the size you want. (*Dragging* means holding the left mouse button down while you move an object with the mouse.)
5. Release the mouse button. The control appears on the form.



Another simple way to add a control to a form is to double-click the button for that control in the Toolbox. This creates a default-size control located in the center of the form; then you can move the control to another location on the form.

Resizing, Moving, and Locking Controls

Notice that small rectangular boxes called *sizing handles* appear at the corners of the control; you'll use these sizing handles in the next step as you resize the control. You can also use the mouse, keyboard, and menu commands to move controls, lock and unlock control positions, and adjust their positions.

To resize a control

1. Select the control you intend to resize by clicking it with the mouse. Sizing handles appear on the control.
2. Position the mouse pointer on a sizing handle, and drag it until the control is the size you choose. The corner handles resize controls horizontally and vertically, while the side handles resize in only one direction.
3. Release the mouse button –or– Use **SHIFT** with the arrow keys to resize the selected control.

To move a control

- Use the mouse to drag the control to a new location on the form –or– use the Properties window to change the **Top** and **Left** properties. When a control is selected, you can use the CTRL key with the arrow keys to move the control one grid unit at a time. If the grid is turned off, the control moves one pixel at a time.

To lock all control positions

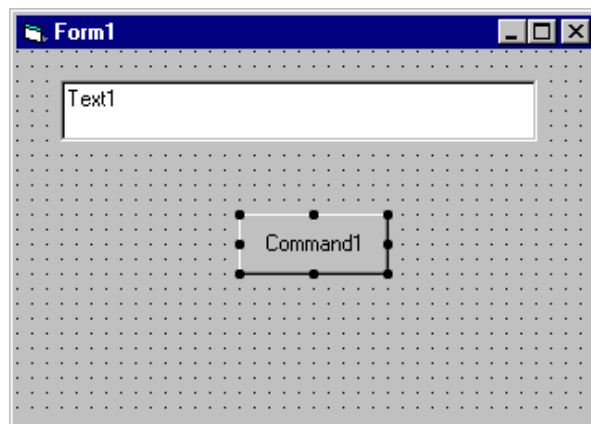
- From the **Format** menu, choose **Lock Controls** –or– click the **Lock Controls Toggle** button on the **Form Editor** Toolbar. This will lock all controls on the form in their current positions so that you don't inadvertently move them once you have them in the desired location. This will lock controls only on the selected form; controls on other forms are untouched. This is a toggle command, so you can also use it to unlock control positions.

To adjust the position of locked controls






















- You can "nudge" the control that has the focus by holding CTRL down and pressing the appropriate arrow key –or– you can change the control's **Top** and **Left** properties in the Property window.

You now have the interface for the "Hello, world!" application, as shown in Figure 2.4.

Figure 2.4 The interface for the "Hello, world!" application



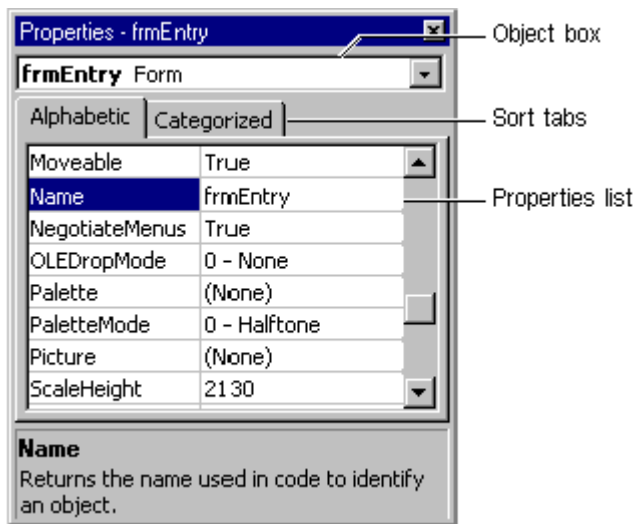
Tools in the Visual Basic Toolbox

Icon	Tool Name	What This Tool Does
	Pointer	Selects objects
	Picture box	Draws a box to display graphics
	Label	Draws a box to display text
	Text box	Draws a box that can display text and let the user type in text
	Frame	Groups two or more objects together
	Command button	Draws a command button
	Check box	Draws a check box
	Option (or radio) button	Draws a radio button
	Combo box	Draws a combo box
	List box	Draws a list box
	Horizontal scroll bar	Draws a horizontal scroll bar
	Vertical scroll bar	Draws a vertical scroll bar
	Timer	Places a timer on a form
	Drive list box	Draws a drive list box that displays all the disk drives available
	Directory list box	Draws a directory list box that displays a directory on a particular disk drive
	File list box	Draws a file list box that displays files in a specific directory
	Shape	Draws a geometric shape such as a circle or a square
	Line	Draws a line
	Image box	Draws a box to display graphics
	Data control	Draws a control to link a program to a database file
	OLE	Draws a box to insert an OLE object

Setting Properties

The next step is to set properties for the objects you've created. The Properties window (Figure 2.5) provides an easy way to set properties for all objects on a form. To open the Properties window, choose the Properties Window command from the View menu, click the Properties Window button on the toolbar, or use the context menu for the control.

Figure 2.5 The Properties window



The Properties window consists of the following elements:

- Object box — Displays the name of the object for which you can set properties. Click the arrow to the right of the object box to display the list of objects for the current form.
- Sort tabs — Choose between an alphabetic listing of properties or a hierarchical view divided by logical categories, such as those dealing with appearance, fonts, or position.
- Properties list — The left column displays all of the properties for the selected object. You can edit and view settings in the right column.

To set properties from the Properties window

1. From the **View** menu, choose **Properties**, or click the **Properties** button on the toolbar.

The **Properties** window displays the settings for the selected form or control.

2. From the Properties list, select the name of a property.
3. In the right column, type or select the new property setting.

Enumerated properties have a predefined list of settings. You can display the list by clicking the down arrow at the right of the Settings box, or you can cycle through the list by double-clicking a list item.

For the "Hello, world!" example, you'll need to change three property settings. Use the default settings for all other properties.

Object	Property	Setting
Form	Caption	Hello, world!
Text box	Text	(Empty)
Command button	Caption	OK

Setting the Icon Property

All forms in Visual Basic have a generic, default icon that appears when you minimize that form. However, you will probably change this icon to one that illustrates the use of the form or your application. To assign an icon to a form, set the Icon property for that form. You can use 32 x 32 pixel icons that were standard in 16-bit versions of Microsoft Windows and are also used in Windows 95/98 and Windows NT, as well as the 16 x 16 pixel icons used in Windows 95/98.

Writing Code

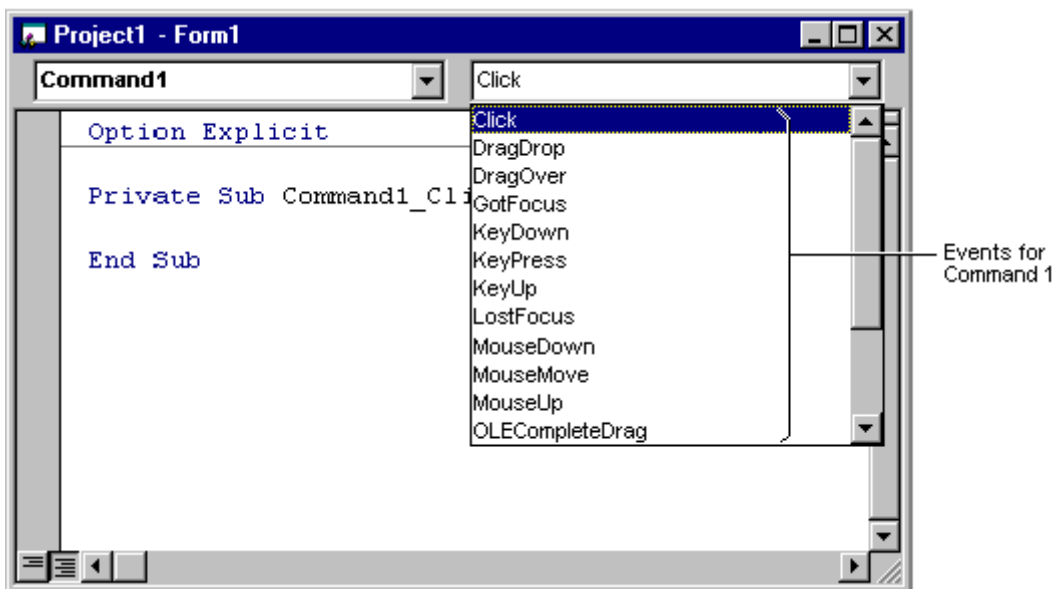
The *Code Editor window* is where you write Visual Basic code for your application. Code consists of language statements, constants, and declarations. Using the Code Editor window, you can quickly view and edit any of the code in your application.

To open the Code window

- Double-click the form or control for which you choose to write code –or– from the Project Explorer window, select the name of a form or module, and choose the **View Code** button.

Figure 2.6 shows the Code Editor window that appears when you double-click the Command button control, and the events for that command.

Figure 2.6 The Code Editor window



You can choose to display all procedures in the same Code window, or display a single procedure at a time.

To display all procedures in the same Code window

1. From the **Tools** menu, select the **Options** dialog box.
2. On the **Editor** tab in the **Options** dialog box, select the check box to the left of **Default to Full Module View**. The check box to the left of **Procedure Separator** adds or removes a separator line between procedures –or– click the **Full Module View** button in the lower left corner of the Code Editor window.

To display one procedure at a time in the Code window

1. From the **Tools** menu, select the **Options** dialog box.
2. On the **Editor** tab in the **Options** dialog box, clear the check box to the left of **Default to Full Module View** –or– click the **Procedure View** button in the lower left corner of the Code Editor window.

The Code window includes the following elements:

- Object list box — Displays the name of the selected object. Click the arrow to the right of the list box to display a list of all objects associated with the form.
- Procedure list box — Lists the procedures, or events, for an object. The box displays the name of the selected procedure — in this case, Click. Choose the arrow to the right of the box to display all the procedures for the object.

Creating Event Procedures

Code in a Visual Basic application is divided into smaller blocks called *procedures*. An *event procedure*, such as those you'll create here, contains code that is executed when an event occurs (such as when a user clicks a button). An event procedure for a control combines the control's actual name (specified in the Name property), an underscore (_), and the event name. For example, if you want a command button named Command1 to invoke an event procedure when it is clicked, use the procedure Command1_Click.

To create an event procedure

1. In the **Object** list box, select the name of an object in the active form. (The *active* form is the form that currently has the focus.)

For this example, choose the command button, Command1.

2. In the **Procedure** list box, select the name of an event for the selected object.

Here, the **Click** procedure is already selected, because it's the default procedure for a command button. Note that a *template* for the event procedure is now displayed in the Code window.

3. Type the following code between the **Sub** and **End Sub** statements:

```
4. Text1.Text = "Hello, world!"
```

The event procedure should look like this:

```
Private Sub Command1_Click ()  
  
    Text1.Text = "Hello, world!"  
  
End Sub
```

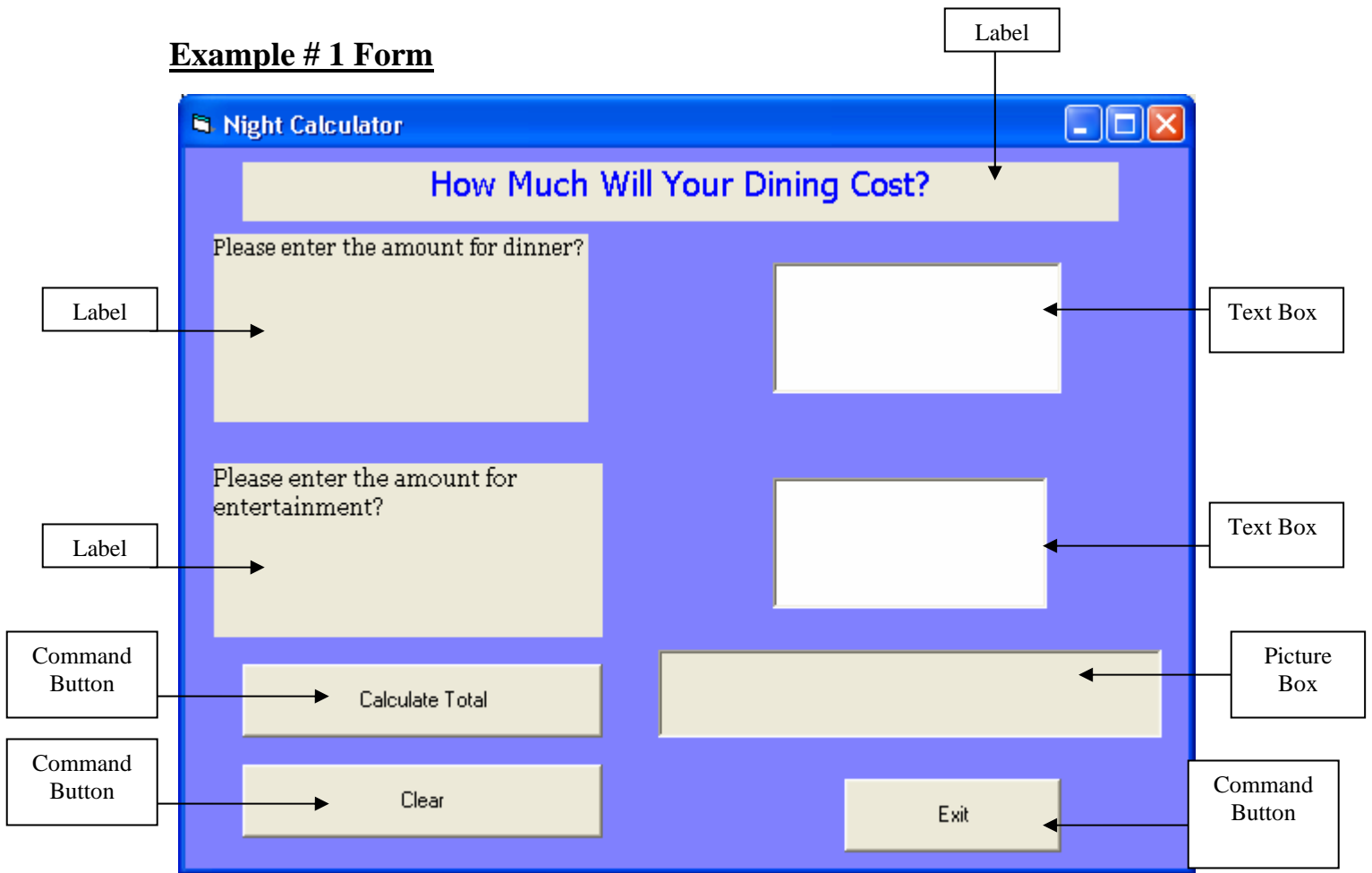
You'll note here that the code is simply changing the Text property of the control named Text1 to read "Hello, world!" The syntax for this example takes the form of *object.property*, where Text1 is the object and Text is the property. You can use this syntax to change property settings for any form or control in response to events that occur while your application is running.

For More Information For information on creating other types of procedures, see "Introduction to Procedures" in "Programming Fundamentals."

Running the Application

To run the application, choose Start from the Run menu, or click the Start button on the toolbar, or press F5. Click the command button you've created on the form, and you'll see "Hello, world!" displayed in the text box.

Example # 1 Form



Command Button – Most Visual Basic applications have command buttons that allow the user to simply click them to perform actions. When the user chooses the button, it not only carries out the appropriate action, it also looks as if it's being pushed in and released and is therefore sometimes referred to as a push button.



Text Box – In general, the text box control should be used for editable text, although you can make it read-only by setting its Locked property to True. Text boxes also allow you to display multiple lines, to wrap text to the size of the control, and to add basic formatting. By default, you can enter up to 2048 characters in a text box. If you set the MultiLine property of the control to True, you can enter up to 32K of text.



Picture Box - Think of the picture box as a blank canvas upon which you can paint, draw or print. A single control can be used to display text, graphics or even simple animation.

Example #1 Code

```
Private Sub Command1_Click()  
totdin = Text1.Text  
totent = Text2.Text  
total = Val(totdin) + Val(totent)  
  
Picture1.Print "Your night will cost "; FormatCurrency(total); "."  
  
End Sub  
  
Private Sub Command2_Click()  
End  
End Sub  
  
Private Sub Command3_Click()  
Picture1.Cls  
Text1.Text = ""  
Text2.Text = ""  
  
End Sub
```

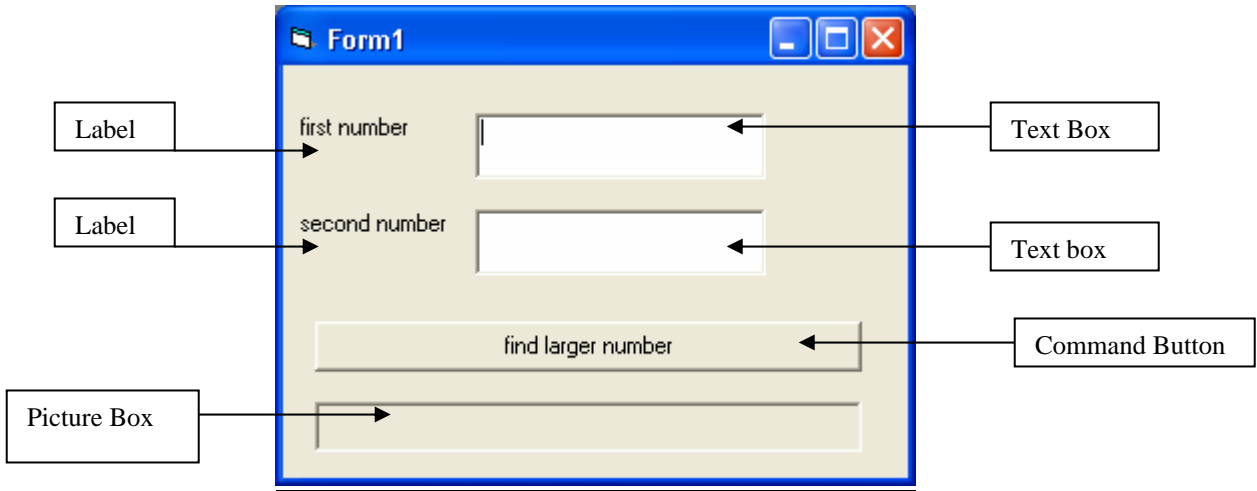
Sub – the beginning of an event procedure. It is an abbreviation of Subprogram

Private – indicates that the event procedure could not be invoked by an event or another form.

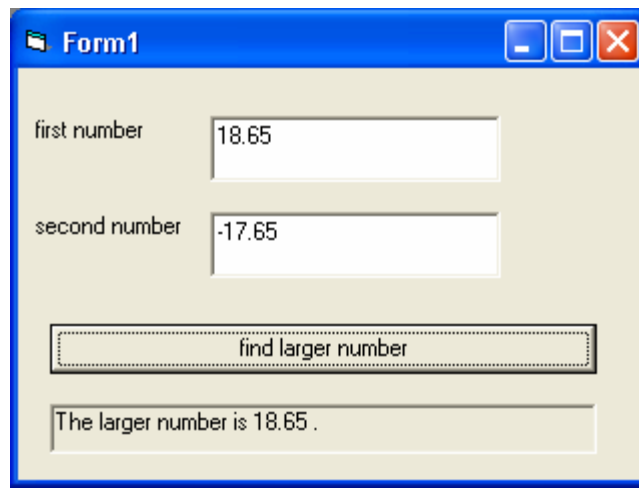
End Sub – Signals the end of the Private Subprogram.

Val – gives the string a numeric value.

Example # 2 Form



Run Program





Command Button – Most Visual Basic applications have command buttons that allow the user to simply click them to perform actions. When the user chooses the button, it not only carries out the appropriate action, it also looks as if it's being pushed in and released and is therefore sometimes referred to as a push button.



Text Box – In general, the text box control should be used for editable text, although you can make it read-only by setting its Locked property to True. Text boxes also allow you to display multiple lines, to wrap text to the size of the control, and to add basic formatting. By default, you can enter up to 2048 characters in a text box. If you set the MultiLine property of the control to True, you can enter up to 32K of text.



Picture Box - Think of the picture box as a blank canvas upon which you can paint, draw or print. A single control can be used to display text, graphics or even simple animation.

Example # 2 Code

```
Private Sub Command1_Click()  
  
Picture1.Cls  
If Val(Text1.Text) > Val(Text2.Text) Then  
    largernum = Val(Text1.Text)  
Else  
    largernum = Val(Text2.Text)  
End If  
Picture1.Print "The larger number is"; largernum; "."  
  
End Sub
```

Sub – the beginning of an event procedure. It is an abbreviation of Subprogram

Private – indicates that the event procedure could not be invoked by an event or another form.

End Sub – Signals the end of the Private Subprogram.

Shortcut Keys for Using Visual Basic

<u>Key(s)</u>	<u>What the shortcut does</u>
Alt + Q	Exits out of Visual Basic
Ctrl + N	Creates a new project
Ctrl + O	Opens an existing project
Ctrl + P	Prints a project
Ctrl + S	Saves a file
F7	Displays the code window for the currently selected object
Shift + F7	Displays the object for a particular event procedure.
Shift + F2	Jumps to the definition of the word identified by the cursor.
F2	Displays the Object Browser window
Ctrl + R	Displays the Project Explorer window
F4	Displays the properties window
Ctrl + E	Displays the menu editor window
Ctrl + D	Adds a file to a Visual Basic project
F5	Runs a Visual Basic Project