Arignar Anna Governement Arts and Science College

Department of Computer Science

www.aagasc.edu.in

VISUAL PROGRAMMING - QA

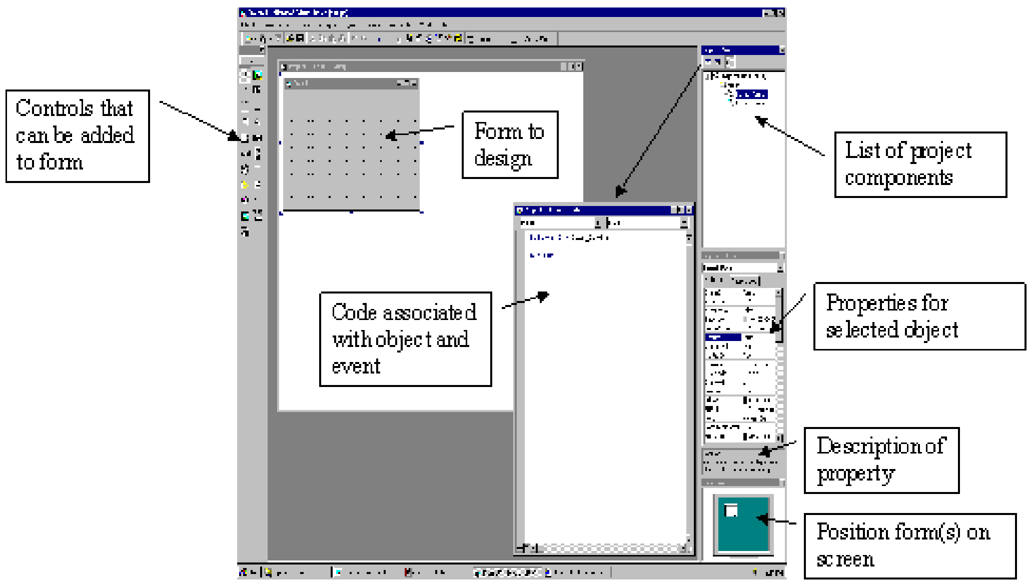
http://ecomputernotes.com/visual-basic

What do you mean by GUI?

Visual Basic provides a graphical user interface GUI that allows the developer drag and drop objects into the program as well as manually write program code.

Integrated Development Environment (IDE)

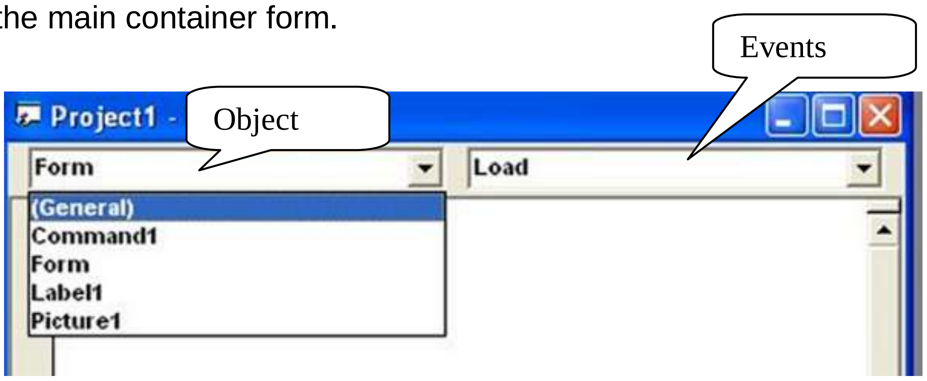
In Visual Basic 6.0 is the Integrated Development Environment (IDE). IDE is a term commonly used in the programming world to describe the interface and environment that we use to create our applications. It is called integrated because we can access virtually all of the development tools that we need from one screen called an interface. The IDE is also commonly referred to as the design environment, or the program.


Controls that can be added to form
Form to design
List of project components
Code associated with object and event
Properties for selected object
Description of property
Position form(s) on screen

Tha Visual Basic IDE is made up of a number of components

- Menu Bar
- Tool Bar
- Project Explorer
- Properties window
- Form Layout Window
- Toolbox
- Form Designer
- Object Browser

In previous versions of Visual Basic, the IDE was designed as a Single Document Interface (SDI). In a Single Document Interface, each window is a free-floating window that is contained within a main window and can move anywhere on the screen as long as Visual Basic is the current application. But, in Visual Basic 6.0, the IDE is in a Multiple Document Interface (MDI) format. In this format, the windows associated with the project will stay within a single container known as the parent. Code and form-based windows will stay within the main container form.



Visual Basic is initiated by using the Programs option > Microsoft Visual Basic 6.0

What are the features of VB?

Guo Interface: - VB is a Graphical User Interface language. This means that a VB program will always show something on the screen that the user can interact with to get a job done.

Modularization: - It is considered good programming practice to modularize your programs. Small modules where it is clearly indicated what comes into the module and what goes out makes a program easy to understand.

Object Oriented: - Object Oriented Programming is a concept where the programmer thinks of the program in "objects" that interact with each other. Visual Basic forces this good programming practice.

Debugging: - Visual Basic offers two different options for code debugging:- Debugging Managed Code Runtime Debugger The Debugging Managed Code individually debugs C and C++ applications and Visual Basic Windows applications. The Runtime Debugger helps to find and fix bugs in programs at runtime.

Data Access Feature: - By using data access features, we can create databases, scalable server-side components for most databases, including Microsoft SQL Server and other enterprise-level database.

Macros IDE: - The Macros integrated development environment is similar in design and function to the Visual Studio IDE. The Macros IDE includes a code editor, tool windows, the properties windows and editors.

Explain Objects and Modules? What are the types of Modules in VB

Objects : Objects are the real world entities. People, companies, employees, fan and ledger entries are all types of objects. In object-oriented terms, the word object is used to describe one specific thing like a car. Objects have an identity and this identity is defined with properties. A car has its name, model, cost and color. The same object-oriented concepts apply regardless of whether the object is based on the real world, on a concept, or on the implementation.

Modules: Modules are collection of code and data that function something like objects in objects oriented programming, but without defining OOP characteristics like inheritance, polymorphism etc. The concept behind modules is to enclose procedures and data in a way that hides them from the rest of the program.

Event and General : In VB event procedures are invoked in response to keyboard, mouse or system action. Your code can also explicitly invoke event procedures. The maximum number of events a control can have is fixed. Event procedures are stored in a form module and are private by default. A general procedure is not executed unless explicitly invoked. You can create a general procedure either by choosing the procedure from the insert menu or by typing the procedure heading Sub followed by the procedure name on a blank line.

What is the purpose of Visual Basic file types: .vbp, .frm, .bas, and .ocx?

Explain file extension in visual basic.

Visual Project consist of at least two, and usually more, files as follows:

.vbp file: This file is called the project file, is a small text file that holds the names of the other files in the project, as well as some information about the VB environment.

.frm file: Each of your form in the project is saved in a file with extension. To begin your project your project will have only one form. Later you can expect your projects to have several forms, with one .frm file for each form. A from holds the description of all objects and their properties for each form, as well as the basic code that you have written to respond to the events. These are also referred as form modules.

.bas file: Optionally your project can have this file. These file holds basic statements that can be accessed from any form. As soon as you begin .bas file are called standard code modules.

.ocx file: additional controls, called custom controls, are stored with a file .ocx extension. If you include controls in your projects that are not part of the standard control set, the .ocx file names will be included in the project.

.vbw file: After you save a project, Visual Basic automatically adds one more file to your project with extension of .vbw. This file holds information about each project's form.

Visual Basic is not only a language. It's an Integrated Development Environment in which you can develop, run, test and debug your applications.

What types of project that you can create in Visual Basic?

(i) Standard EXE : These are the typical applications that you develop with previous versions of Visual Basic.

(ii) ActiveX EXE, ActiveX DLL : These types of projects are available with the Professional edition. ActiveX components are OLE automation servers.

(iii) ActiveX Control : This type of project is also a feature of the Professional edition. We use it to develop your own ActiveX controls.

(iv) ActiveX Document EXE, ActiveX Document DLL : ActiveX documents are in essence Visual Basic applications that can run in the environment of the container that supports hyper-linking.

(v) VB Application Wizard, VB Wizard Manager : The Application Wizard takes you through the steps of setting up the skeleton of a new application. The Wizard Manager lets you build your own wizard.

(vi) Data Project : It's identical to the Standard EXE project type, but it automatically adds the controls that are used in accessing databases to the Toolbox.

(vii) DHTML Application : VB6 allows you to build Dynamic HTML pages that can be displayed in the browser's window on a client computer.

(viii) IIS Application : VB6 allows you to build applications that run on the Web server and interact with clients over the Internet with the Internet Information Server.

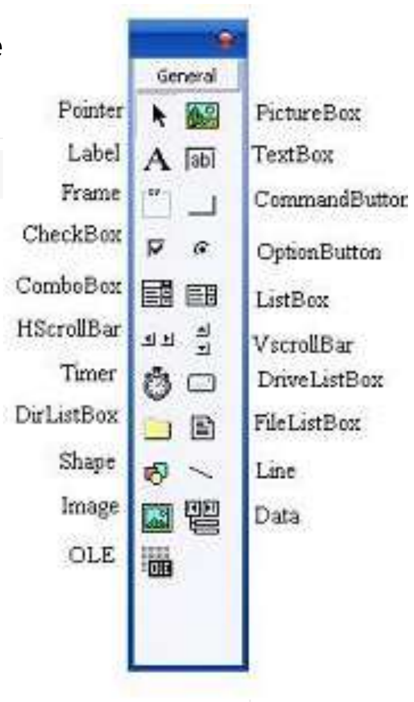
(ix) Addin : You can create your own add-ins for the VB IDE. These are special commands you can add to Visual Basic's menus.

(x) VB Enterprise Edition Controls : It simply creates a new Standard EXE project and loads all the tools of the Enterprise Edition of Visual Basic.

What are the tools in Tool box in VB?

Toolbox

The Toolbox contains a set of controls that are used to place on a Form at design time thereby creating the user interface area. Additional controls can be included in the toolbox by using the Components menu item on the Project menu. A Toolbox is represe in figure 2 shown below.



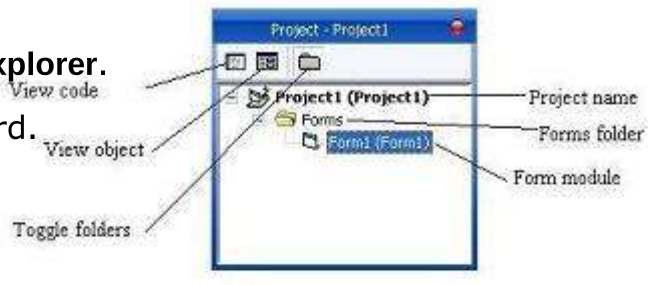
Control	Description
Pointer	Provides a way to move and resize the controls form
PictureBox	Displays icons/bitmaps and metafiles. It displays text or acts as a visual container for other controls.
TextBox	Used to display message and enter text.
Frame	Serves as a visual and functional container for controls
CommandButton	Used to carry out the specified action when the user chooses it.
CheckBox	Displays a True/False or Yes/No option.
OptionButton	OptionButton control which is a part of an option group allows the user to select only one option even it displays mulitiple choices.
ListBox	Displays a list of items from which a user can select one.
ComboBox	Contains a TextBox and a ListBox. This allows the user to select an ietm from the dropdown ListBox, or to type in a selection in the TextBox.
HScrollBar and VScrollBar	These controls allow the user to select a value within the specified range of values
Timer	Executes the timer events at specified intervals of time
DriveListBox	Displays the valid disk drives and allows the user to select one of them.
DirListBox	Allows the user to select the directories and paths, which are displayed.
FileListBox	Displays a set of files from which a user can select the desired one.
Shape	Used to add shape (rectangle, square or circle) to a Form
Line	Used to draw straight line to the Form
Image	used to display images such as icons, bitmaps and metafiles. But less capability than the PictureBox
Data	Enables the use to connect to an existing database and display information from it.
OLE	Used to link or embed an object, display and manipulate data from other windows based applications.
Label	Displays a text that the user cannot modify or interact with.

Write short notes about Project Explorer

Project Explorer shows a list of all files (forms, modules) that make up a project. The list also includes files (modules) created or loaded in Visual Basic Editor. For information on modules, refer to A Project and Three Types of Module.

To display the project explorer, do one of the following:

1. On the **View** menu, click **Project Explorer**.
2. Press **Ctrl** + **R** keys on the keyboard.
3. On the toolbar, click .



Explain Property window

The Properties Window exposes the various characteristics of selected objects. Each and every form in an application is considered an object. Now, each object in Visual Basic has characteristics such as color and size. Other characteristics affect not just the appearance of the object but the way it behaves too. All these characteristics of an object are called its properties. Thus, a form has properties and any controls placed on it will have propeties too. All of these properties are displayed in the Properties Window.

The following properties apply to most of the VB tools:

• Name : It sets the name of the control, through which you can access the control's properties and methods.

• Appearance : It can be 0 for a flat look or 1 for a 3-D look.

• Back Color : It sets the background color on which text is displayed or graphics are drawn.

• Fore Color : It sets the foreground color

• Font : It sets the face, attribute, and size of the font used for the text on the control.

• Caption : It sets the text that is displayed on many controls that don't accept input, for example, the text on a Label control, the caption of a Command Button control.

• Text : It sets the text that is displayed on the controls that accept user input, for example, the TextBox control.

• Width & Height : These properties set the control's dimensions.

• Left & Top : These properties set the coordinates of the control's upper-left corner, expressed in the units of the container.

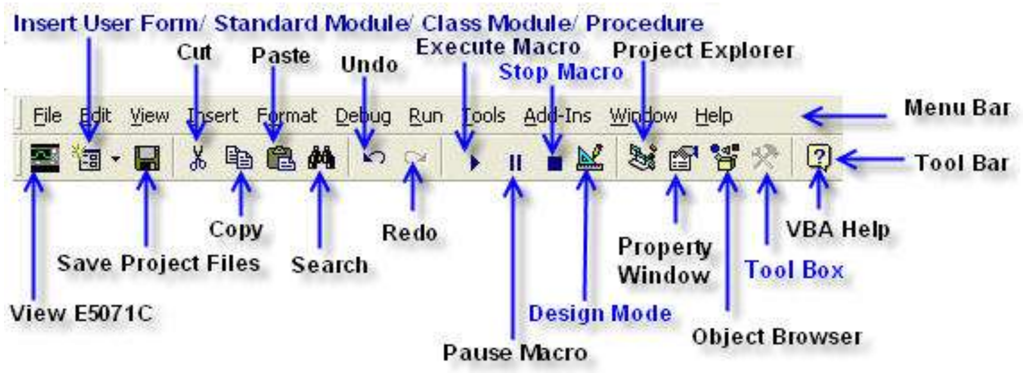
• Enabled : By default, this property's value is True, which means that the control can get the focus.

To display the properties window, do one of the following:

1. On the **View** menu, click **Properties Window**.
2. Press **F4** key on the keyboard.
3. On the toolbar, click .

Explain Standard Tool Bar in Visual Basic

The toolbar provides access to commonly used commands via icon buttons; these commands are a subset of the commands accessible from the menu bar.



Explain Program Structure

In Code window

1. Declaration
   In which we can declare object and other variables related with this project.
2. Form Load Event
   This event consist of code which run before the form load on screen
3. Component Events
   In this section we can use corresponding events and codes.
4. Form Unload
   In this event we can write the code which is used after close the forms.

What is Visual Basic Event? Give Some Examples of Events

Visual Basic programs are built around events. Events are various things that can happen in a program. this will become clearer when studied in contrast to procedural programming. In procedural languages, an application is written is executed by checking for the program logically through the program statements, one after another. For a temporary phase, the control may be transferred to some other point in a program. An event procedure is "attached" to an object and is executed when the relevant event impinges on that object.

- Change
- Click
- DragDrop
- GotFocus
- KeyDown
- KeyPress
- KeyUp
- LostFocus
- MouseDown
- MoseMove
- MouseUp
- OLECompleteDrag
- Paint
- Resize
- Load
- Unload

Unit – 2
Constant and Literal Data Types

Literal

A literal is a value that is expressed as itself rather than as a variable's value or the result of an expression, such as the number 3 or the string "Hello". A constant is a meaningful name that takes the place of a literal and retains this same value throughout the program, as opposed to a variable, whose value may change.

Dim age as Integer
Age=20

Where 20 is the numeric literal

Dim cname as String
cname="Praveen"

Where "Praveen" is the string literal

Data Types

Visual Basic classifies the information mentioned above into two major data types, they are the numeric data types and the non-numeric data types.

Numeric Data Types

Numeric data types are types of data that consist of numbers that can be computed mathematically with standard operators. Examples of numeric data types are height, weight, share values, the price of goods, monthly bills, fees and others. In Visual Basic, numeric data are divided into 7 types, depending on the range of values they can store. Calculations that only involve round figures can use Integer or Long integer in the computation.

Type	Storage	Range of Values
Byte	1 byte	0 to 255
Integer	2 bytes	-32,768 to 32,767
Long	4 bytes	-/+ 10 Digits
Single	4 bytes	-3.4E+38 to 3.4E+38
Double	8 bytes	-1.7E+308 to 1.7E+308
Currency	8 bytes	-/+ 15 Digits
Decimal	12 bytes	+/- 29 Digits (28 decimal places).

Non-numeric Data Types

Nonnumeric data types are data that cannot be manipulated mathematically. Non-numeric data comprises string data types, date data types, boolean data types that store only two values (true or false), object data type and Variant data type .The following table shows nonnumeric data types.

Table 5.2: Nonnumeric Data Types

Data Type	Storage	Range
String(fixed length)	Length of string	1 to 65,400 characters
String(variable length)	Length + 10 bytes	0 to 2 billion characters
Date	8 bytes	January 1, 100 to December 31, 9999
Boolean	2 bytes	True or False
Object	4 bytes	Any embedded object
Variant(numeric)	16 bytes	Any value as large as Double
Variant(text)	Length+22 bytes	Same as variable-length string

How to Declare Variable

After declaring various variables using the Dim statements,

Dim Variable1,Variable2…VariableN As Data types

Example

```
Dim  a,b,c As Integer
```

we can assign values to those variables. The syntax of an assignment is

Variable=Expression

The variable can be a declared variable or a control property value. The expression could be a mathematical expression, a number, a string, a Boolean value (true or false) and more.

The following are some examples variable assignment:

f = 100
s = f - 99
uname = "John Kennady"
text1.text = uname

Visual Basic Operators

The simplest operators carry out arithmetic operations. These operators in their order of precedence are:

Arithmetic Operator	Operation
^	Exponentiation
* /	Multiplication and division
\	Integer division (truncates)
Mod	Modulus
+ -	Addition and subutraction

- **Parentheses** around expressions can change precedence.
- To **concatentate** two strings, use the **&** symbol or the **+** symbol:

lblTime.Caption = "The current time is" & Format(Now, “hh:mm”)

txtSample.Text = "Hook this “ + “to this”

- There are six **comparison** operators in Visual Basic:

Relational Operator for Comparison

>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
=	Equal to
<>	Not equal to

- The result of a comparison operation is a Boolean value (**True** or **False**).

We will use three **logical** operators

Not	Logical not
And	Logical and
Or	Logical or

- The **Not** operator simply negates an operand.
- The **And** operator returns a True if both operands are True. Else, it returns a False.
- The **Or** operator returns a True if either of its operands is True, else it returns a False.
- Logical operators follow arithmetic operators in precedence.

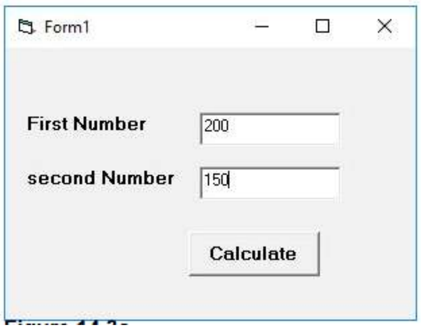
Explain Sub Routine or Sub Procedure.

A sub procedure(also call subroutine) is a procedure that is called from the main procedure to perform a specific task. It is different from function in the sense that it does not return a value as a function does.A sub procedure is usually used to accept input from the user, display information, print information, manipulate properties or perform some other tasks. It is a program code by itself and it is not an event procedure because it is not associated with a runtime procedure . It is called by other code whenever it is required to perform a certain task. Sub procedures help to make programs smaller and seamless to manage. A sub procedure begins with a Sub ProcedureName keyword and ends with an End Sub keyword.

The structure of a sub procedure is as follows:

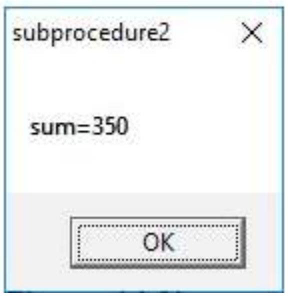
```
Sub ProcedureName (arguments)
        Statements

End Sub
```

In this example, we create a sub procedure to sum up two values that are specified by the arguments . The main program can reference a procedure by using its name together with the arguments in the parentheses.


Form1

First Number 200

second Number 150


subprocedure2

sum=350

```
Private Sub Command1_Click()
Dim x,y as Integer
  x = Val(TxtNum1.Text)
  y = Val(TxtNum2.Text)
  sum x, y
End Sub

Sub sum(a As Integer, b As Integer)
```

```
  MsgBox ("sum=" & a + b)
End Sub
```

User Defined Functions

The structure of a function is as follows:

    Public Function functionName(Arg As dataType,.......) As dataType

Public indicates that the function is applicable to the whole project and private indicates the function is application to a module or sub procedure

Example

```
Public Function areaOfCircle(r As Double) As Double
    areaOfCircle = 3.145 * r * r
End Function
```

In the above function we can call text1.text = areaOfCircle(5)

This will return the value after calculating the value.

Built in Visual Basic Functions

Visual Basic offers a rich assortment of built-in functions. The on-line help utility will give you information on any or all of these functions and their use. Some examples are:

Function Value Returned

Abs	Absolute value of a number
Asc	ASCII or ANSI code of a character
Chr	Character corresponding to a given ASCII or ANSI code
Cos	Cosine of an angle
Date	Current date as a text string
Format	Date or number converted to a text string
Left	Selected left side of a text string
Len	Number of characters in a text string
Mid	Selected portion of a text string
Now	Current time and date
Right	Selected right end of a text string
Rnd	Random number
Sin	Sine of an angle
Sqr	Square root of a number
Str	Number converted to a text string
Time	Current time as a text string
Timer	Number of seconds elapsed since midnight
Val	Numeric value of a given text string

Loop statements

Loop statements allow us to execute one or more lines of code repetitively. Visual Basic supports the following loop statements:

Do……Loop : The Do……Loop executes a block of statements for as long as a condition is True. Visual Basic evaluates an expression, and if it's True, the statements are executed. If the expression is False, the program continues and the statement following the loop is executed.

There are two variations of the Do……Loop statement. A loop can be executed either while the condition is True or until the condition becomes True. These two variations use the keywords While and Until to specify how long the statements are executed. To execute a block of statements while a condition is true, use the following syntax :

```
Do While condition
        statement block
Loop
```

Example

```
Do while counter <=100
     Text1.Text=counter
     counter =counter+1
Loop
```

* The above example will keep on adding until counter > 100

To execute a block of statements until the condition becomes True, use the following syntax :

```
Do Until condition
        statement block
Loop
```

Another variation of the Do loop executes the statements first and evaluates the condition after each execution. This Do…Loop has the following syntax :

For…..Next : The For …..Next loop requires that you know how many times the statements in the loop will be executed. The For…Next loop uses a variable(it's called the loop's counter) that increases or decreases in value during each repetition of the loop. The For…..Next loop has the following syntax :

```
For counter=start to end  [Step increment]
        statements
Next  [counter]
```

The keywords in the square brackets are optional. The arguments counter, start, end and increment are all numeric. The loop is executed as many times as required for the counter to reach the end value.

Example :

```
Dim counter As Integer
For counter = 1 To 10
        List1.AddItem counter
Next
```

While ………Wend : The while……wend loop executes a block of statements while a condition is True. The while…..wend loop has the following syntax :

```
While condition
        statement - block
Wend
```

If condition is true, all statements are executed and when the Wend statement is reached, control is returned to the While statement which evaluates condition again. If condition is still True, the process is repeated. If condition is False, the program resumes with the statement following the Wend statement.

The Label

This is probably the first control you will master. It is used to display static text, titles and screen output from operations. The important properties are

Caption - the text that is displayed in the label

BackColor and ForeColor - colors of the background and the text

BackStyle - Opaque or Transparent - whether the background is visible or not

Font - font and size of text Alignment - text centered, left or right

Multiline- True or False - if True, you can have several lines of text, delimited by <CR> in the label - by default, it is set to False

TextBox & CommandButton

The TextBox is like a Label but, it is used to input data into the program. The data typed in is in the **Text** property of the control. When the program is Run, only the controls that can be manipulated will be activated.

TextBox control supports various events such as Change, Click, MouseMove and many more that will be listed in the Properties dropdown list in the code window for the TextBox control. We will look into a few of them as given below.

The code entered in the Change event fires when there is a change in the contents of the TextBox The Click event fires when the TextBox control is clicked. The MouseMove event fires when the mouse is moved over the TextBox

On the form there is only one control at any given time that has the cursor on it - it is said to have **Focus**. If you type data, the control with Focus will receive it. You change the Focus with Tab or by clicking on a different control.

The Command Button is used to initiate actions, usually by clicking on it. The Caption property determines the text to display on the face of the button. The **Default** property, if set to true, means that the button will be activated (same as Clicked) if the <Enter> key is hit anywhere in the form.

```
Private Sub Command1_Click()
    Unload Me
End Sub
```

ListBox and ComboBox controls present a set of choices that are displayed vertically in a column. If the number of items exceed the value that be displayed, scroll bars will automatically appear on the control. These scroll bars can be scrolled up and down or left to right through the list.

The following Fig lists some of the common ComboBox properties and methods.

Property/Method	Description
Properties	
Enabled	By setting this property to True or False user can decide whether user can interact with this control or not
Index	Specifies the Control array index
List	String array. Contains the strings displayed in the drop-down list. Starting array index is 0.
ListCount	Integer. Contains the number of drop-down list items
ListIndex	Integer. Contains the index of the selected ComboBox item. If an item is not selected, ListIndex is -1
Locked	Boolean. Specifies whether user can type or not in the ComboBox
Methods	
AddItem	Add an item to the ComboBox
Clear	Removes all items from the ComboBox
RemoveItem	Removes the specified item from the ComboBox
SetFocus	Transfers focus to the ComboBox

Event Procedures	
Change	Called when text in ComboBox is changed
DropDown	Called when the ComboBox drop-down list is displayed
GotFocus	Called when ComboBox receives the focus
LostFocus	Called when ComboBox loses it focus

Adding items to a List

It is possible to populate the list at design time or run time

Design Time : To add items to a list at design time, click on List property in the property box and then add the items. Press CTRL+ENTER after adding each item as shown below.

Run Time : The AddItem method is used to add items to a list at run time. The AddItem method uses the following syntax.

Object.AddItemitem, Index

The item argument is a string that represents the text to add to the list

The index argument is an integer that indicates where in the list to add the new item. Not giving the index is not a problem, because by default the index is assigned.

Following is an example to add item to a combo box. The code is typed in the Form_Load event

```
Private Sub Form_Load()
        Combo1.AddItem 1
        Combo1.AddItem 2
        Combo1.AddItem 3
        Combo1.AddItem 4
        Combo1.AddItem 5
        Combo1.AddItem 6
End Sub
```

Removing Items from a List

The RemoveItem method is used to remove an item from a list. The syntax for this is given below.

Object.RemoveItem index

The following code verifies that an item is selected in the list and then removes the selected item from the list.

```
Private Sub cmdRemove_Click()
    If List1.ListIndex > -1 Then
    List1.RemoveItem List1. ListIndex
    End If
End Sub
```

Sorting the List

The Sorted property is set to True to enable a list to appear in alphanumeric order and False to display the list items in the order which they are added to the list.

Using the ComboBox

A ComboBox combines the features of a TextBox and a ListBox. This enables the user to select either by typing text into the ComboBox or by selecting an item from the list. There are three types of ComboBox styles that are represented as shown below.

The Simple Combo box displays an edit area with an attached list box always visible immediately below the edit area. A simple combo box displays the contents of its list all the time. The user can select an item from the list or type an item in the edit box portion of the combo box.

The Dropdown list combo box turns the combo box into a Dropdown list box. At run time , the control looks like the Dropdown combo box. The user could click the down arrow to view the list.

Control Array

A control array is a group of controls that share the same name type and the same event procedures. Adding controls with control arrays uses fewer resources than adding multiple control of same type at design time. To create a control array, you must put a single control of the desired type on the form. With the control selected, press [Ctrl]C and then [Ctrl]V.

For example the component name is text1 the control array will creates the name as text1(0), text1(2), text1(3)…etc.

Using index value we can control set of objects easily.

Tab order

The tab order is the order in which a user moves focus from one control to another by pressing the TAB key. Each form has its own tab order. By default, the tab order is the same as the order in which you created the controls. Tab-order numbering begins with zero.

Tab order can be set in the Properties window using the TabIndex property. The TabIndex property of a control determines where it is positioned in the tab order. By default, the first control drawn has a TabIndex value of 0, the second has a TabIndex of 1, and so on.