Contents

CHAPTER 1: INTRODUCTION TO VISUAL BASIC

1.1 Concepts

a. Visual: The "Visual" part refers to the method used to create the graphical user interface (GUI).

b. BASIC: (Beginners All-Purpose Symbolic Instruction Code) language, a language used by more programmers than any other language in the history of computing.

c. Visual Basic: Visual Basic is a Microsoft Windows Programming language. Visual basic allows user to create GUI. Visual Basic programs are created in an Integrated Development Environmental (**IDE**). The **IDE** allows the programmer to create, run and debug Visual Basic programs conveniently. : It is Language developed in early 1960's at Dartmouth College

c. Objects: is the collection of controls (labels, command button, textbox, checkbox etc) and form that allow user to create a project.

d. Events: An event is a specific action that occurs on or with a certain object.

Examples of event used in VISUAL BASIC are: click, Dblclick, change, load, keypress, mouse down etc

e. Procedure: a procedure is the block of visual basic statement enclosed by declaration statement (s**ub, function, operator, get, and set**) and a matching end declaration**.**

Eg. Private sub **command1_click**

Statement

End sub

f. Application: is similar to project .application is the collection of object that work together to accomplish something useful. In visual basic is called project

g. Event procedures: are the code related to some object, this is code that is executed when a certain event occurs.

d) Method: method build –in procedure that can be invoked to impart some action to particular object.

1.2 The Importance of Visual Basic Program

1. It uses integrated development environment (IDE) which is easier for the user to minimize code writing.

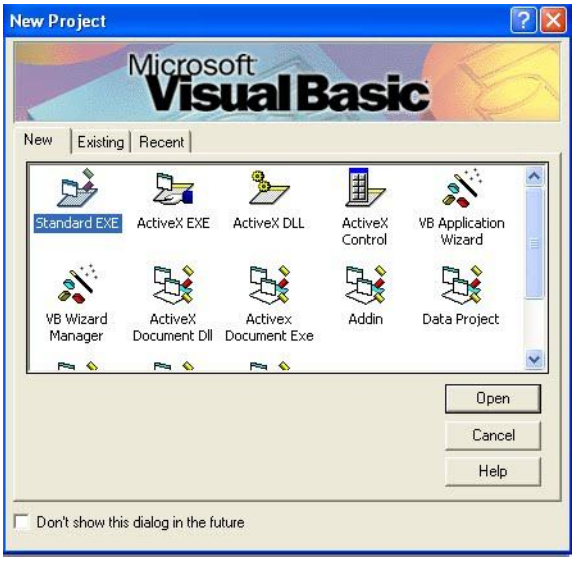
2. All visual programs follow the same concepts; therefore the user will become more familiar with visual approach for other visual languages.
3. It provides Input box and Output box as an interactive window with user.
4. It is able to connect to Internet, and to call Explorer.

1. 3 Visual BASIC environment

Visual Basic implements graphical user interface that allows the use of graphics for different applications. It provides visual interactive windows with user, like Dialogue box for (color, font) Input box, and Output box .Also it is able to create menu to simplify user application.
To run this program on user computer:

Start>programs>Microsoft Visual Studio 6.0>Microsoft Visual Basic 6.0.--> standard exe→open

It will appear on the computer screen as in the following picture.



To exit from Visual Basic and return to Windows is like exit from most Windows applications.

There are three ways to close the Visual Basic as stated below.
1- Click on close button icon that appears in the upper-left corner of the screen.
2- Press Alt+F4

3- Select File >Exit

1.4 Steps in Developing Application

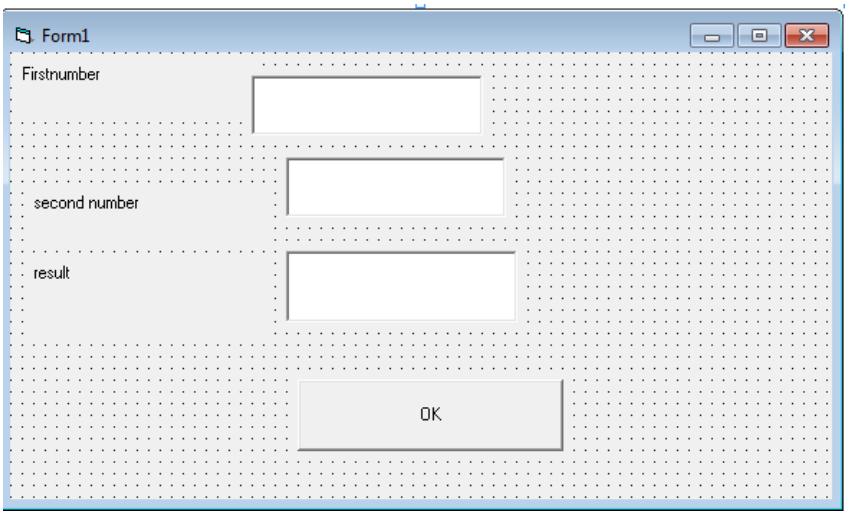
There are three primary steps involved in building a Visual Basic application:

1-**Draw** the user interface

2- **Assign** properties to controls

3- **Attach** code to control

Example: step1 and 2:



Step3: Private Sub Command1_Click()

a = Val(Text1.Text)

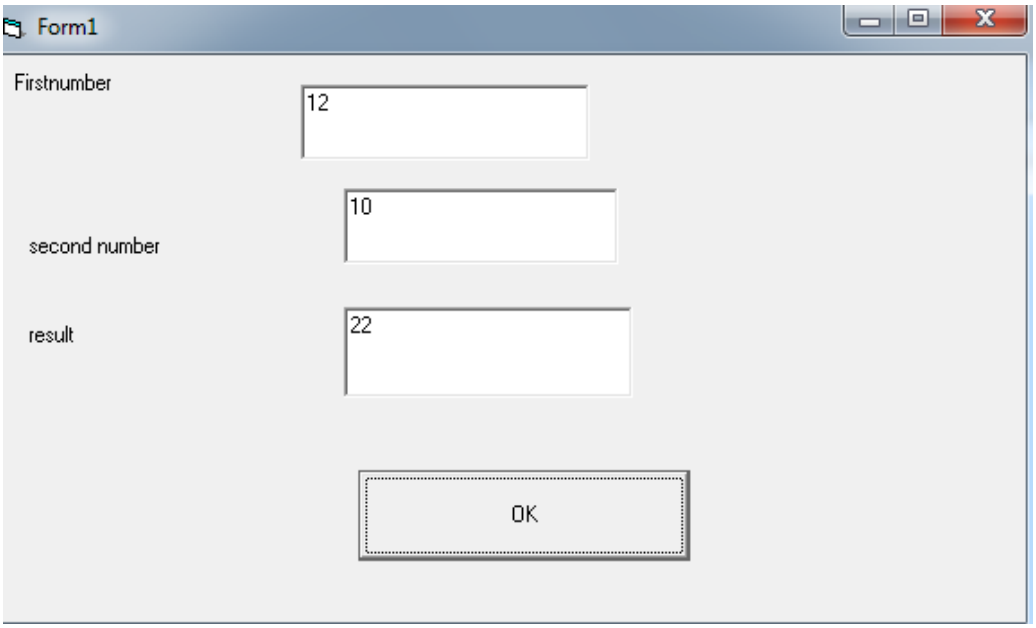
b = Val(Text2.Text)

result = a + b

Text3.Text = result

End Sub

Output



1.5 The modes used in VB6

There are three mode used in vb 6 those are:

Design mode: used to build application

Run mode: used to run application

Break mode: application halted(stop) and debugger is available.

1.6 Project

Project is a program designed to user application that may be simple (like calculator program) or complex (like word program). Visual basic program can create many types of projects. The most important or usual project is the standard project (for window applications) and the DHTML project (for internet).

Working with Standard Projects:

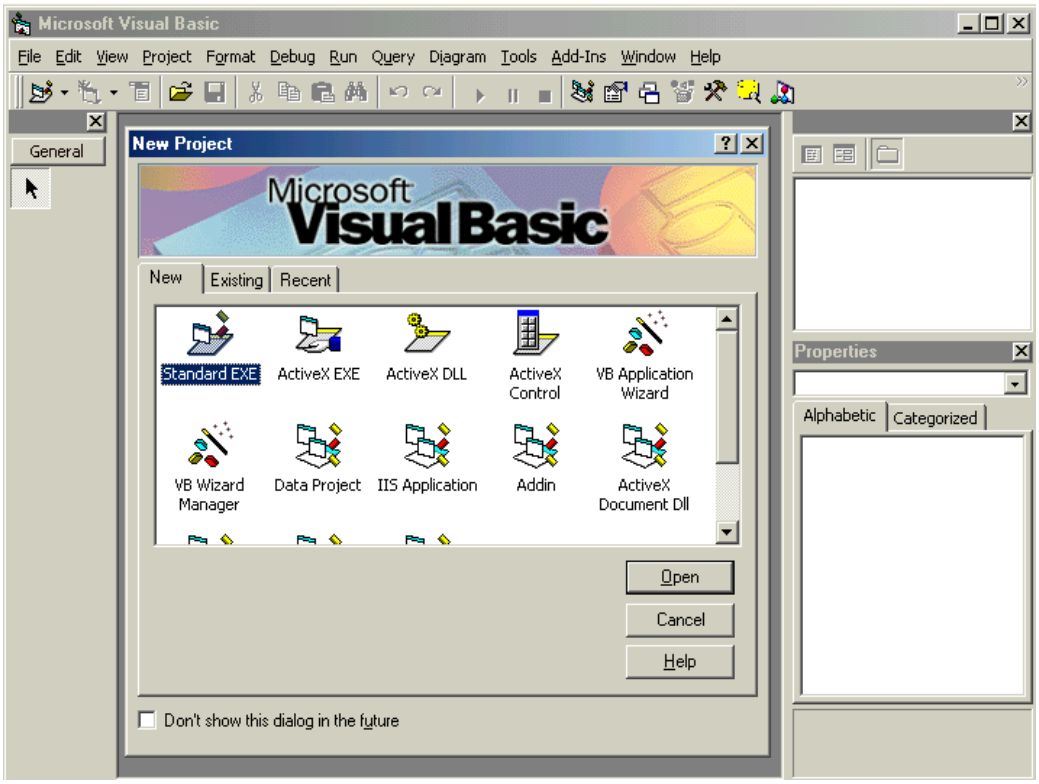
The following working steps (create, save, add, open and delete) could be done:

a) To create project:

When program starts, project box appears-select Standard EXE >

Project window appears.

OR: File> New project> Box (select Standard EXE)> Project window appears.



b) **To add project:** Any number can be added.

Project icon> Select Standard EXE> Project window appears.

Note: Usually first project runs first, but user can change that by:

Selecting project from project window > mouse list > Set as startup.

c) **To open an existing project:**

It is previously designed and saved on disc in a folder.

File> Open project> Box (select existing and look for the project) >

Project window

d) To delete a project:

Select project in Project window > Mouse list > Remove project.

e) To save project: The visual basic can save the project on disc in two ways, as an executable type or a non- executable type.

For project in non execution stage: There are many types of files summarized as follows:

1. **Project file**: it consists of all files which are related to specific project,also some other information with it. This could be saved with extension(.VBP)
2. **The form Files**: this contains form description and any Object or program related to it .This is saved with extension (.frm).

To save project for first time:

File>Save project (group) as>Box (project name)> forms saved then projects group saved.

To resave project: to save previously saved project in same place File>save project (group)

Note: If a form is modified it should be saved. To save a form:

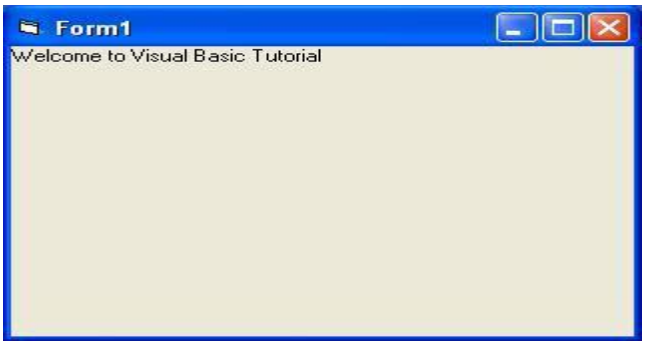
Select a form from Project window>File>Save project form1 as > Savebox (select form name). OR: File>Save project form1.

II- project for execution: This is the final stage so that it could be opened and run by Windows and no need for Visual Basic program. File> Make project.exe.

Item		Action steps	Remarks
Create project	New	File>New project	The user can open any number of projects.
	Exist	File>Open project	Project was already designed and saved.
	Recent	File>Open project	Project was recently designed and saved.
Save project		File>Save project group as	Visual Basic can deal with it (open and modify).
		File>Make project1.exe	For execution by window.
Delete project		File> Remove project	Select project before remove.

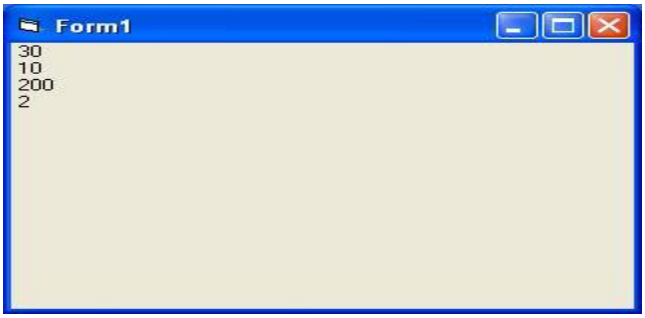
Example1 of application:

```
Private Sub Form_Load ( )
Form1.show
Print "Welcome to Visual Basic tutorial"
End Sub
```



Example2:

```
Private Sub Form_Activate ( )
Print 20 + 10
Print 20 - 10
Print 20 * 10
Print 20 / 10
End Sub
```



CHAPTER 2: INTEGRATED DEVELOPMENT ENVIRONMENT (IDE)

The IDE environment consists of many elements. Some elements are displayed when Visual Basic is started (By default) as in the following figure. Other elements are displayed if the user requires them. We will list some of these elements.

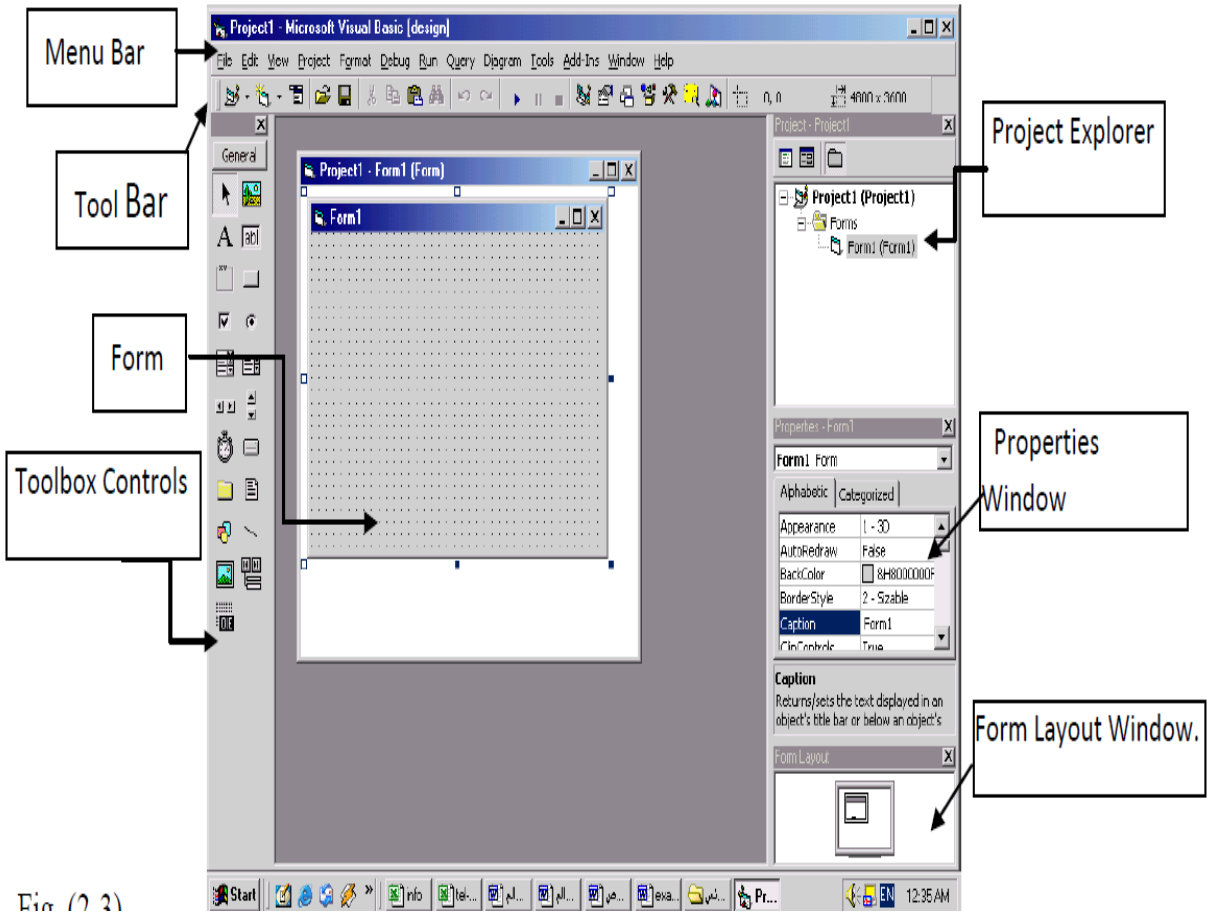


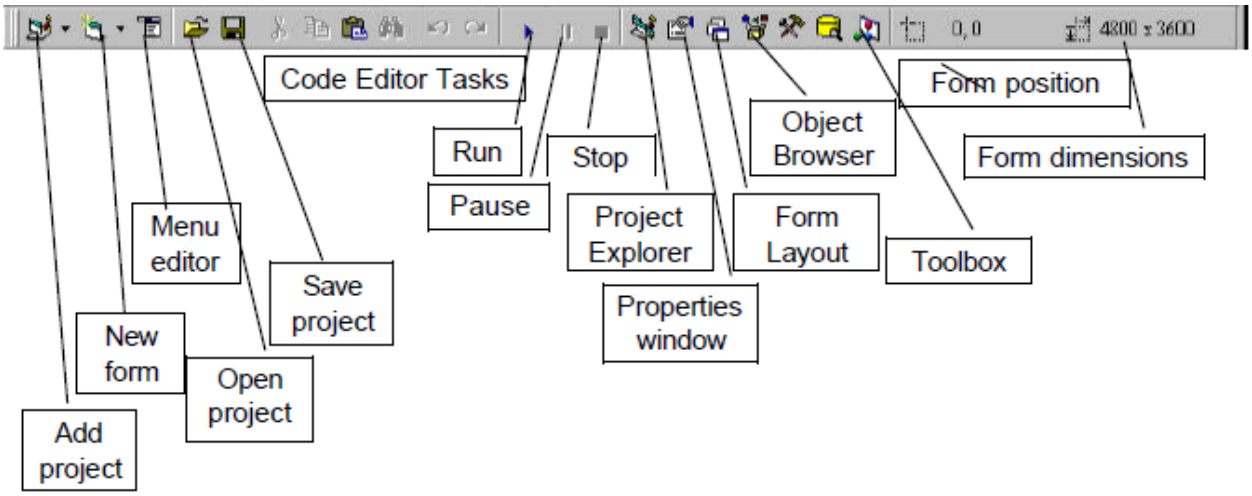
Fig. (2.3)

2.1 Menu Bar

Menu bar contains a standard command and specific command like (File, Edit, View, Project, Format, Debug, Run, etc.)

2.2 Toolbar

Tool bar contains several icons that provide quick access to commonly used features.



2.3 ToolBox

ToolBox contains a collection of tools (controls) that are needed for project design. To show the toolbox press View> toolbox icon. The user can place the tool on form, and then work with the tool. To place the tool on form: click on tool > draw tool to form > the tool appears on form or double click on tool then the tool appears on form.

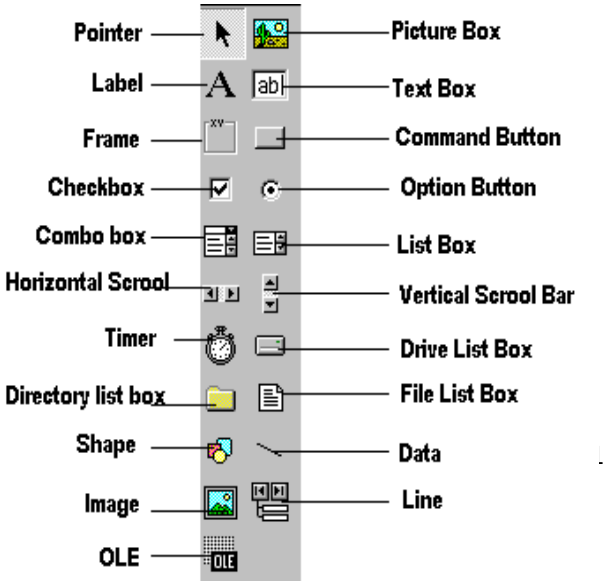
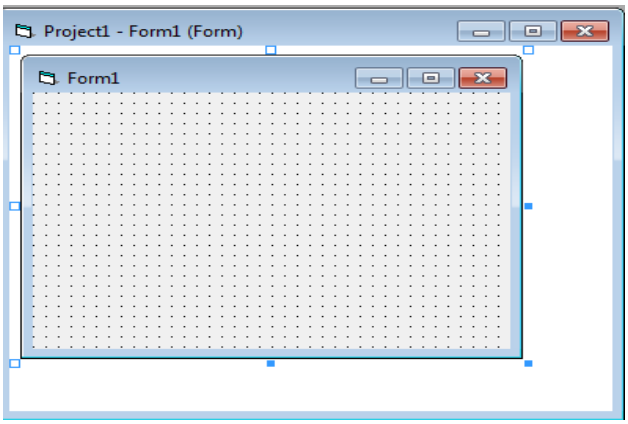


Table (1): Toolbox controls summary.

Control	Description
Pointer	Used to interact with controls on the form(resize them, move them, etc.). The pointer is not a control
PictureBox	A control that display images or print the result.
Label	A control that displays uneditable text to the user.
TextBox	A control for accepting user input. Textbox can also display text.
Frame	A control for grouping other controls.
CommandButton	A control that represents a button. The user presses or clicks to initiate an action.
CheckBox	A control that provides the user with a toggle choice (checked or unchecked)
OptionButton	Option buttons are used in groups where only one at a time can be true.
ListBox	A control that provides a list of items.
ComboBox	A control that provides a short list of items.
HscrollBar	A horizontal scrollbar.
VscrollBar	A vertical scrollbar.
Shape	A control for drawing circles, rectangles, squares or ellipse
Line	A control for drawing line.
DrivelistBox	A control accessing the system disk drivers.
DirlistBox	A control accessing directories on a system
Filelistbox	A control accessing file in a directory
Image	A control for displaying images. The images control does not provide as many capabilities as a picturebox.
OLE	A control for interacting with other window applications.
Timer	A control that performs a task at programmer specified intervals. A timer is not visible to the user.

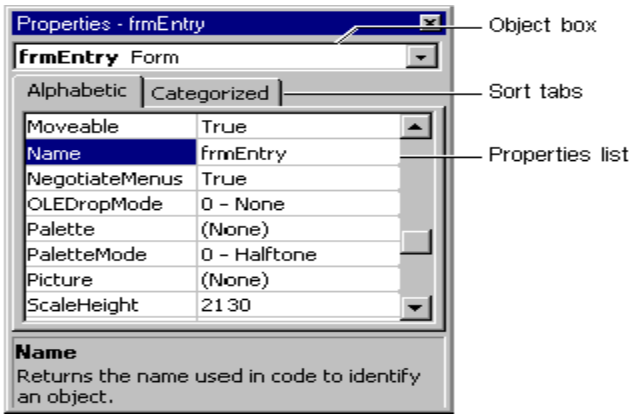
2.4 Form Designer

Form designer is a window for each form to customize the designed interface of the application. Using the form designer, the user can add controls, graphics, and text to create the desired form appearance.



2.5 Properties Window

Properties window is a List of properties settings for a selected form or a control. These properties are characteristics (such as size, visible, or color) of the selected object it provides an easy way to set properties. To show the properties window press View> properties window icon.

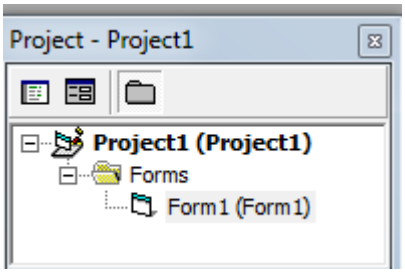


Properties name	Objective
Name	Used to represent name of object in code.
Caption	Name appears on object.
Back color	Background color for object.
Fore color	Color of text written on object.
Font	Font style type and size
Visible	The tool is visible or invisible.
Enable	The tool enable or disable
Height	Length of object
Width	Width of object
Top	Coordinates of top of object on screen
Left	Coordinates of left of object on screen
Text	Allows inputting and editing text in object.

2.6 Project Explorer Window

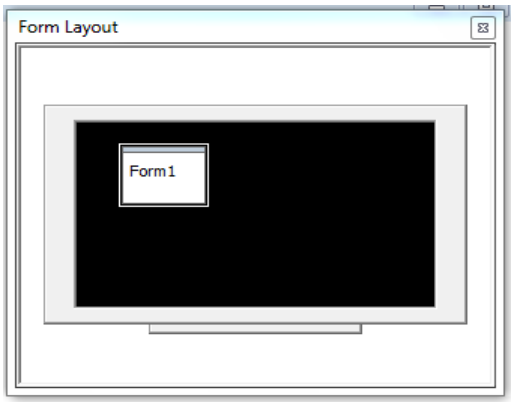
The window titled Project-Project1 is called the Project Explorer and contains the project files. The project explorer window's tool bar contains three buttons, namely view code, view object and toggle folders. When pressed, the view code button displays a window for writing Visual Basic code. View object, when pressed, displays the form. Double-clicking form1 (form1) also displays the form. The toggle folders button toggles (i.e., alternately hides or shows) the forms folder.

The forms folder contains a listing of all forms in the current project. To show the Project Explorer window press View> Project Explorer window icon



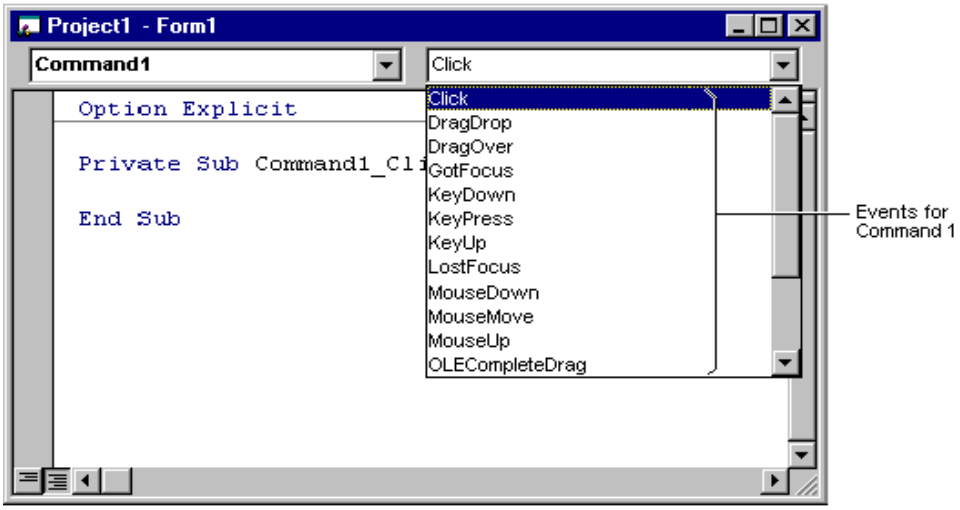
2.7 Form Layout Window

The Form Layout window specifies a form's position on the screen at runtime. The Form Layout window consists of an image representing the screen and the form's relative position on



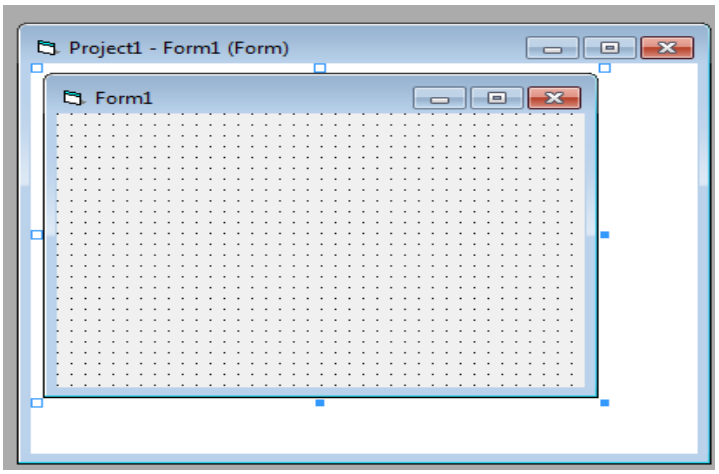
2.8 Code Editor Window:

Code Editor Window is used to write a VB code for an application. For each form there is a separate code editor window. It is displayed when user clicks on form or object in form.



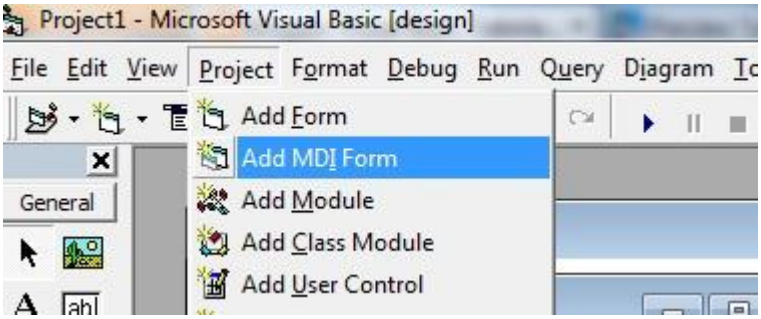
2.9 Environment options (SDI, MDI)

SDI: single document interface is one where all windows appear independently of one other without unification of a single parent window .



MDI: multiple document interfaces is one that allows you to view multiple windows within a larger window.

How to add an MDI form to the current project?



Project > Add MDI form. Click Project from the menu bar and click Add MDI form.

Restrictions of the MDI form

1. You can have only one MDI form per project.
2. You can't place most controls on an MDI form. The only controls that can be placed on the surface of the MDI form are Timer, Picture Box.

These restrictions are there because MDI forms are special type of forms, only used to handle multiple child forms.

How does the MDI form work?

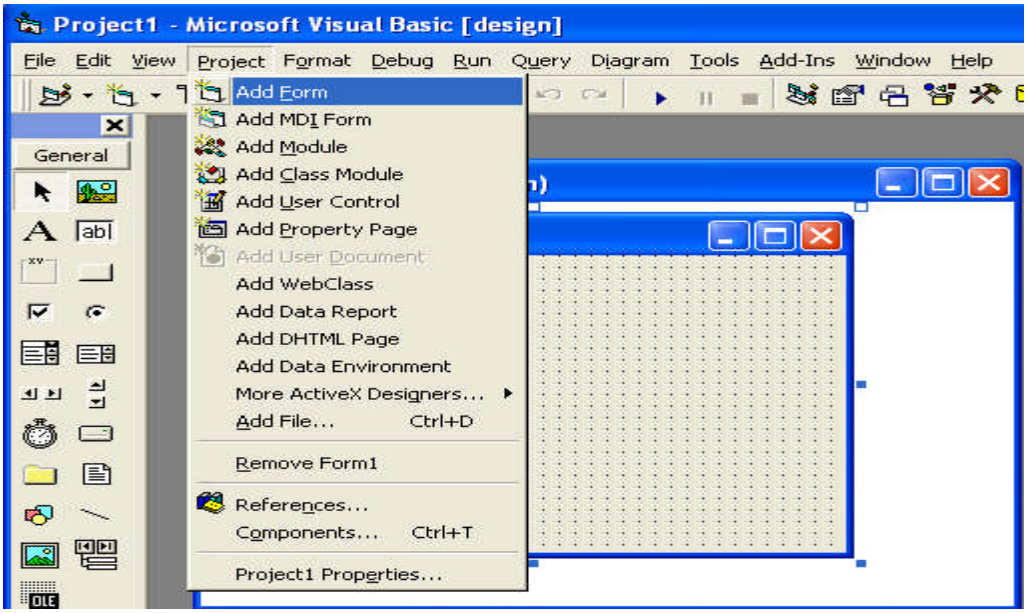
In your project, there will be only one MDI parent form with one or more MDI child forms (or simply child forms).

- ✓ **MDI child form:** To add a child form, you have to add a regular form, and set the MDIchild property to True. You can have many child forms. You can show an MDI child form using the Show method as same as the regular forms.
- ✓ **AutoShowChildren property of an MDI form:** The default value is True. When its True, the MDI child forms are displayed when they are loaded. When the value is False, only then you can keep it hidden after loading, otherwise not.

CHAPTER 3: VISUAL BASIC CONTROLS

Controls: is the graphical features drawn on forms that allow user interaction (text box, labels, command button)

3.1. **Forms:** The form is the most important visible object, without it no control can be displayed. It is a window that can be designed and modified to fit user applications. In the standard project the form Designer creates and modifies visual forms .When user starts visual Basic program a form is automatically displayed in Designer window. The designer can add any number of forms to the project of his application by pressing: add form from project menu.



The forms also have properties and events.

3.1.1 Form properties

Properties list has a predefined value (numeric or string) and could be changed, some properties could be rewritten like caption, and some could be selected from option list by pressing on down arrow on the side. Others could be rewritten or by browsing the computer files when the user clicks on the dotted button on the right side a dialogue box appears. The browsing button appears when the user clicks inside the box.

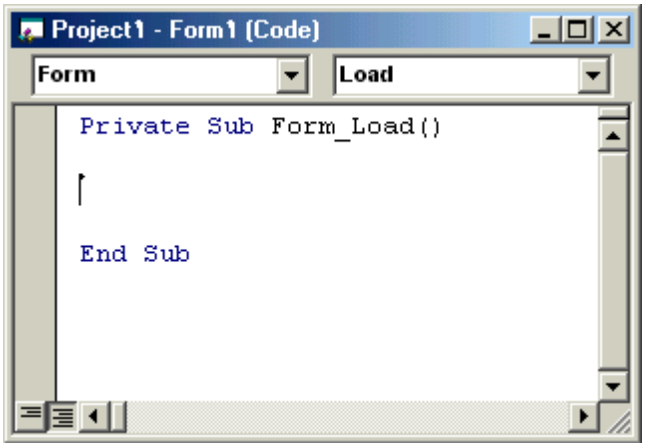
The most important properties of the form are listed in the following table:

3.1.2 Code form

The code is written in code Form and it will be edited quickly by code editor .The codes are of two categories:

1- Declaration is written before any procedure in the code

2- Statements. The user selects the required event then code statements are written inside these event procedures.



3.1.3 Form and event

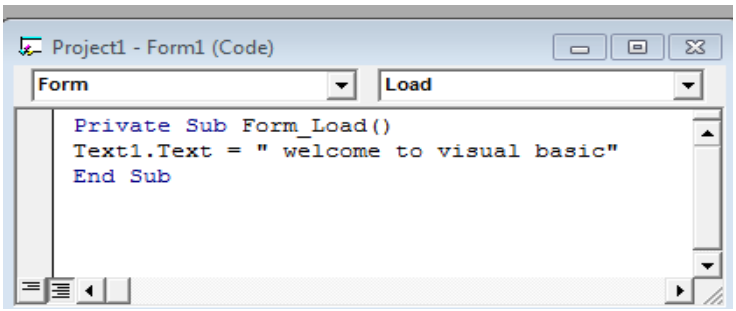
The forms and controls support events (generation, interaction with mouse and keyboard). The most important events for the form are described in the following table.

Event	Action taken when
Click	Single click on object.
DbClick	Double click on object.
load	Loading the object

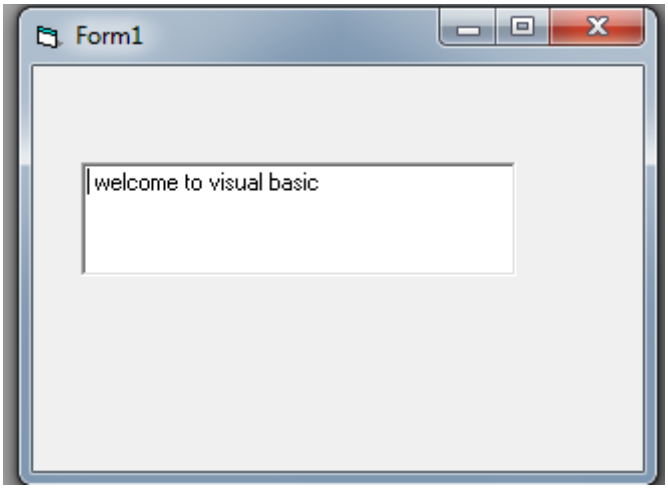
Examples:

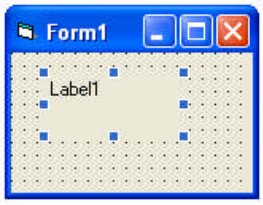
1- Design a form such that: in event load, when project runs, the form backcolor property changed (chose any color).

Eg: **code:**



Result





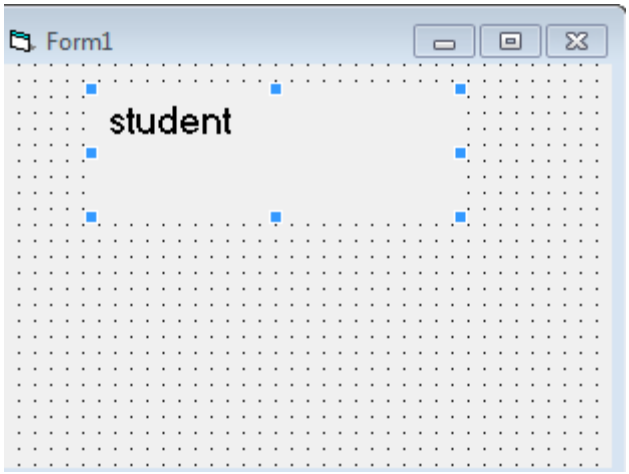
3.2 Labels:

Label is used to display fixed text on form

Property name	Objective	Code	Stage of Changing
Caption	String appear on label	$label_{no.}$.caption= "any name"	Design and run
Autosize	To resize tool to fit text	$label_{no.}$.autosize= true or false	Design and run
Backcolor	Background color for label	$label_{no.}$.Backcolor=Qbcolor(no.)	Design and run
Forecolor	Color of text written on label	$label_{no.}$.forecolor=Qbcolor(no.)	Design and run
Font	Font style, type and size	Size: $label_{no.}$.fontsize= no. Style: *font* { *italic*, *bold*, *underline* } Type: label.FontName = "arial"	Design and run
visible	The label appear or disappear	$Label_{no}$.visible= true or false	Design and run
Enabled	The label enable or disable.	label $_{no.}$. Enabled =true or false	Design and run

Example1: Design a form contains label "Student" in size 14.

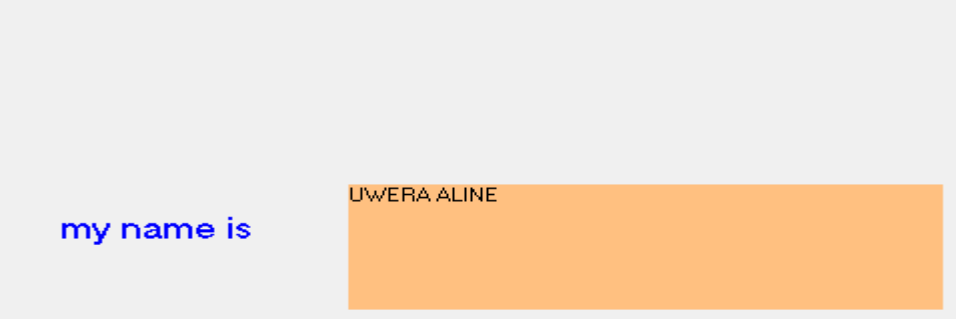
Draw label → Caption: student, Font:14



Example2: write program in VB that displays your name in label

Answer:

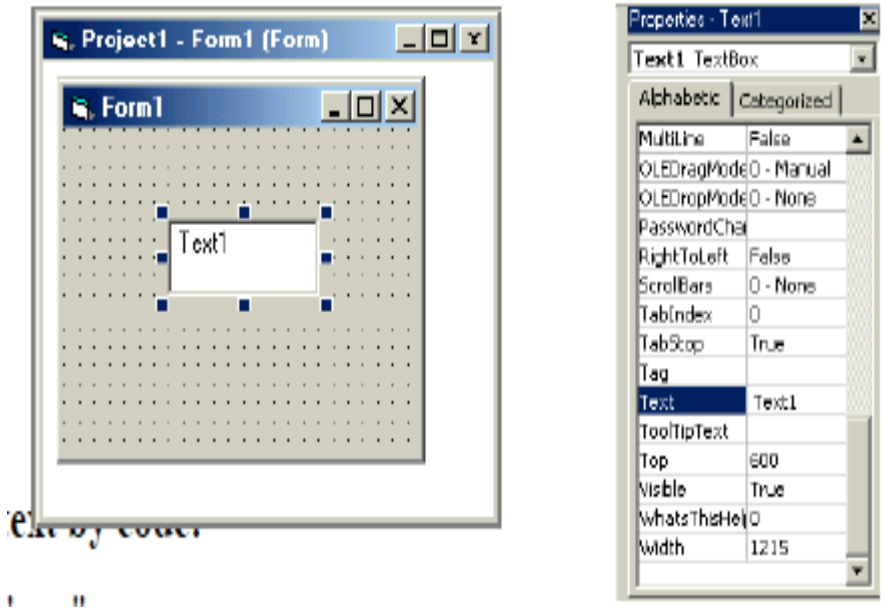
```
Private Sub Label2_Click()
Label2.Caption = "UWERA ALINE"
End Sub
```



3.3 Text Box

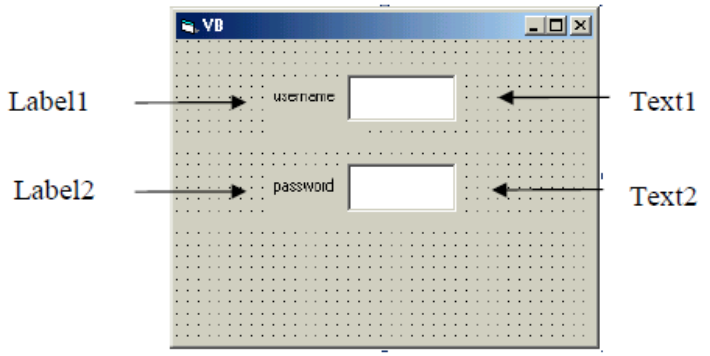
The textbox is a box for entering and displaying text (characters or values) in user project. This tool is used frequently in most of the application. The textbox has property window, with no caption, but with space for text. The most important property of this tool is the text content which is described in the following:

Property name	Objective	Code	Stage of Changing
Text	String appear on textbox	text $_{no.}$. text = "any name"	Design and run
multiline	To enter more than one line	true or false	Design
Backcolor	Background color for textbox.	text $_{no.}$.Backcolor=Qbcolor(no.)	Design and run
Forecolor	Color of text written on textbox.	text $_{no.}$.forecolor=Qbcolor(no.)	Design and run
Font	Font style, type and size.	Size: text $_{no.}$.fontsize= no. Style: *font* { *italic*, *bold*, *underline* } Type: label.FontName = "arial"	Design and run
visible	The textbox appear or disappear	text $_{no.}$.visible= true or false	Design and run
Enabled	The textbox enable or disable.	text $_{no.}$. Enabled =true or false	Design and run
passwordchar	A row of symbols appear instead of letters	Text$_{no.}$.passwordchar=(symbol)	Design and run
Setfocus	Put the focus on the specified textbox	Text$_{no.}$.setfocus	Run



Example: Design a form to enter username and password such that the title of the form is VB.

Sol: design stage



Form1	
caption	V.B
Text1	
text	
Text2	
Text	
Label1	
caption	username
Label2	
caption	password

Example: Design a form with one textbox, set the text properties so that this massage appears when project runs (welcome to visual basic world).

Eg: There are two methods:

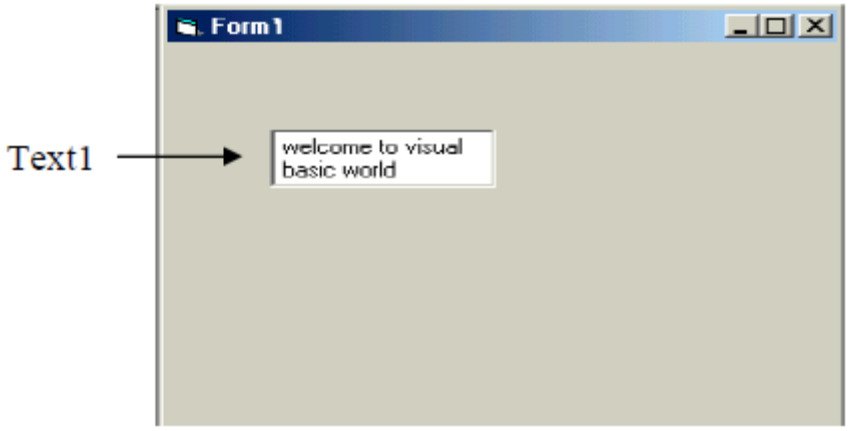
First method: changing property by code:

```
Private Sub Form_Load()
Text1.Text = "welcome to visual basic world"
End Sub
```

Second method: by properties window

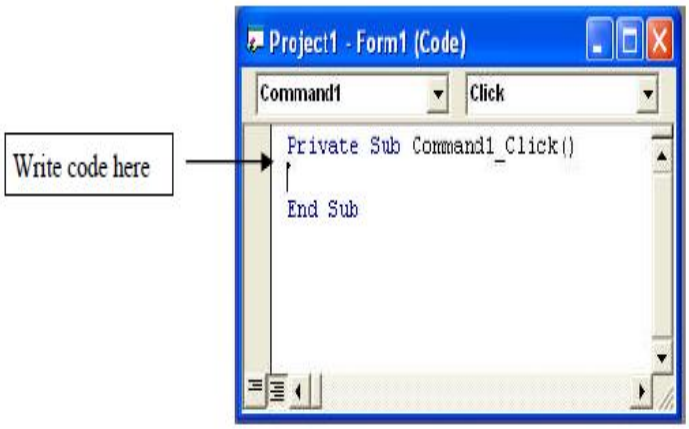
Text1	
text	Welcome to visual basic world

Running stage



3.4 Command button

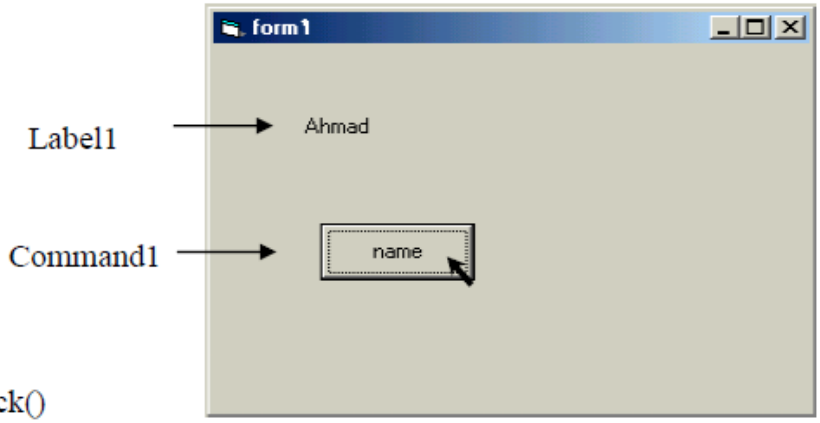
It acts as a switch. To deal with tool property> click on command button> property window appear> change setting of any desired property. Usually change set its caption property to a suitable string. To make the button functional; the user should add some code. To do this: click on command tool> code form appears with click event procedure. Write code in this event or other events like keypress event.



Example: Design a form with label, such that when click on the command button "name" your name appears on label (at running stage).

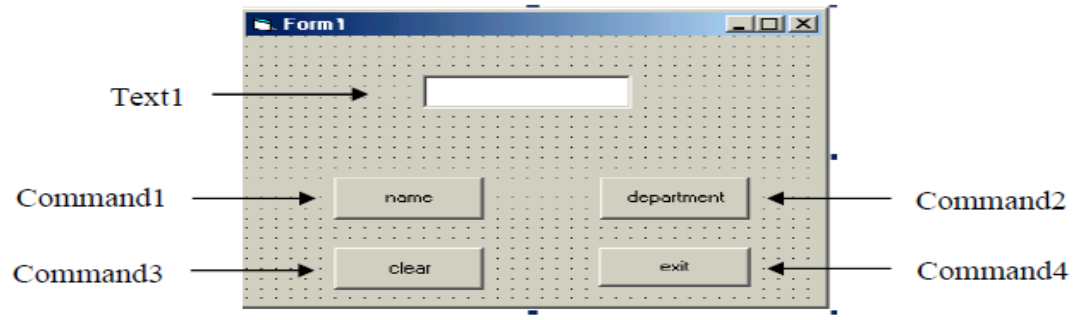
sol:

Label1	
caption	
Command1	
caption	name




```
Private Sub Command1_Click()
Label1.Caption = "Ahmad"
End Sub
```

Example: Design a form to appear your name and department in textbox, when click on command button "name" and "department" respectively so that you can clear these informations when click on command "clear" and stop project when click on command "exit".



Text1	
text	
Command1	
caption	name
Command2	
caption	department
Command3	
caption	Clear
Command4	
caption	exit

```
Private Sub Command1_Click()
Text1.text="Muna"
End Sub
Private Sub Command2_Click()
Text1.text="Science"
End Sub
Private Sub Command3_Click()
Text1.text=" "
End Sub
Private Sub Command4_Click()
end
End Sub
```

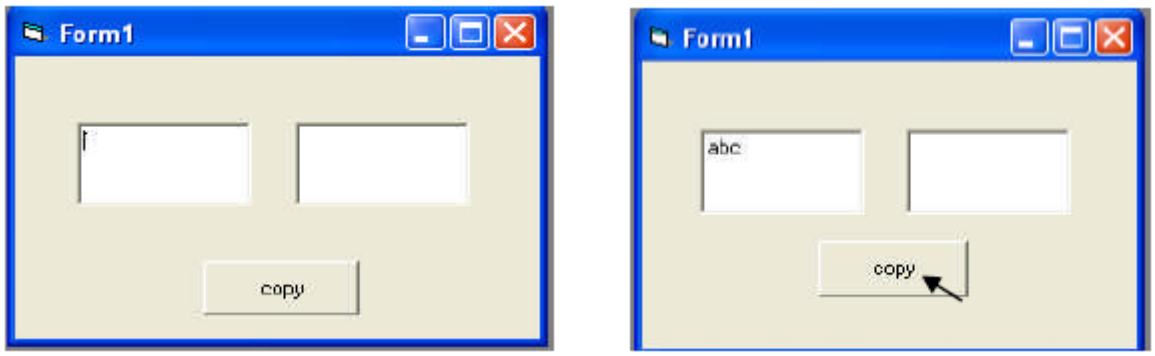
Example: Design a form contains two textbox so that when click on command button "copy" the text copied from first textbox to the second textbox but in size (28).

Sol:

Text1	
text	
Text2	
Text	
Command1	
caption	copy

```
Private Sub Command1_Click()
Text2.Text = Text1.Text
Text2.FontSize = 28
End Sub
```

At run stage this window appear

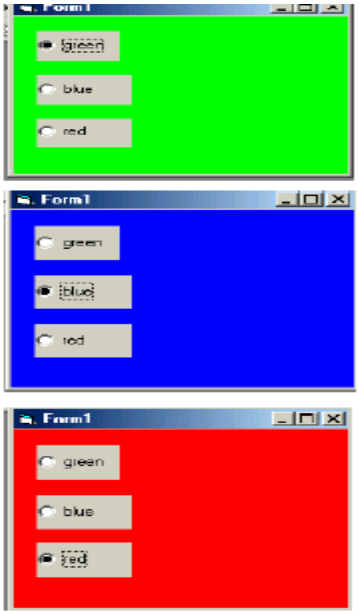


3.5 Option button

Option button is used only as a group of buttons. When the user selects one of them the others are deselected automatically. All other properties of this control are similar to those in form and command button where they are fully discussed which are caption, font, enabled, backcolor and visible beside an important property which is value that takes true or false and it used with if statement. The option button usually takes click event.

Example: Design a form with three option buttons " red ", " green " and " blue " such that when we click on options the color of the form colored by red, green and blue respectively.

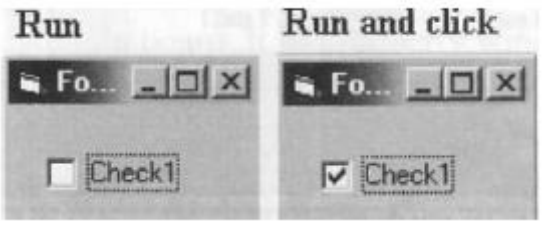
option1:caption	green
option2: caption	blue
option3: caption	red




```
Private Sub Option1_Click()
Form1.BackColor = vbGreen
End Sub

Private Sub Option2_Click()
Form1.BackColor = vbBlue
End Sub

Private Sub Option3_Click()
Form1.BackColor = vbRed
End Sub
```

3.6 Check box

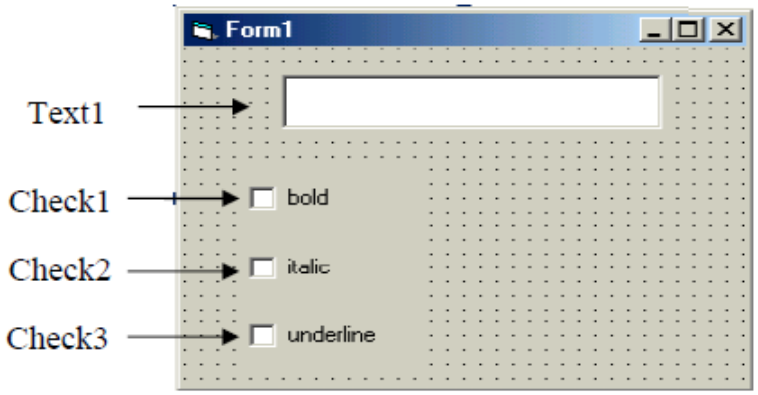
Any number of check boxes can be used on a form. They work independently. Its Property value could be changed in design stage manually, or in running stage by code.



Example: Design a form with one text box and three check boxes such that when click on boxes the following is done: change typing to bold, italic, underline.

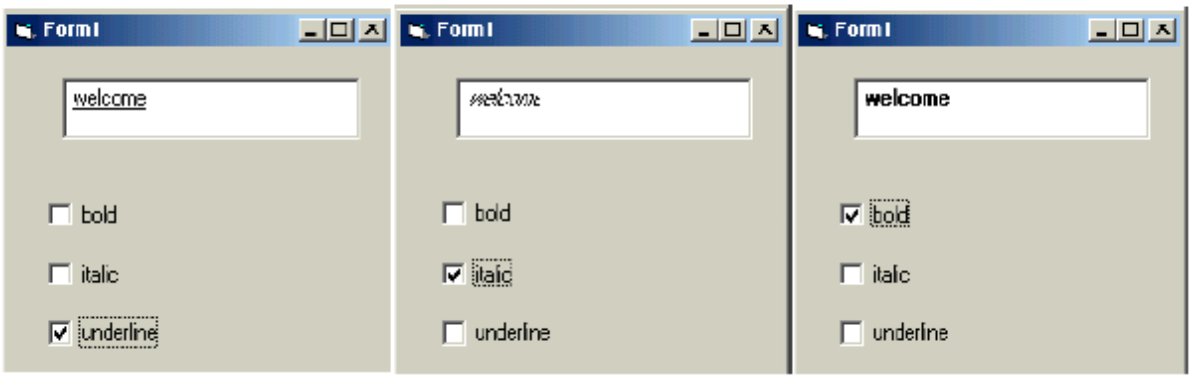
Sol:

Text1	
Text	• • •
Check1	
caption	Bold
Check2	
caption	Italic
Check3	
caption	underline




```
Private Sub Check1_Click()
Text1.FontBold = Check1.Value
End Sub
Private Sub Check2_Click()
Text1.FontItalic = Check2.Value
End Sub
Private Sub Check3_Click()
Text1.FontUnderline = Check3.Value
End Sub
```

Run stage:



3.7 Timer

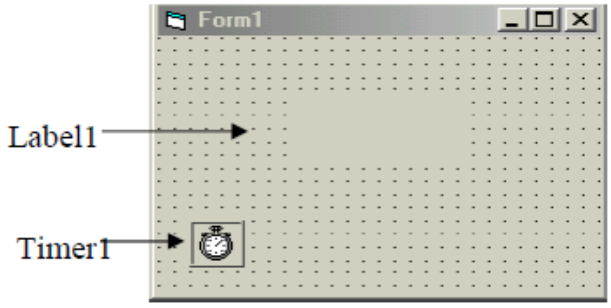
Timer returns the time in millisecond. It may be used to measure execution time of code (program efficiency).

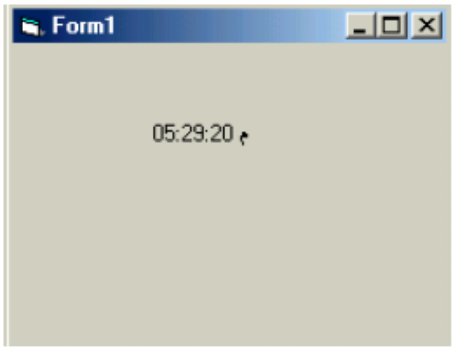
Property name	Objective and code
interval	To repeat the code according to event. It takes an integer values (0-65535) and measured in millisecond
enabled	$timer_{no.}$. Enabled =true or false

Ex: design electronic clock to display the time in seconds.
sol:

Timer1	
interval	1000
Label1	
Caption	

```
Private Sub Timer1_Timer()
Label1.Caption = Time
End Sub
```





3.8 List Box

What is a list control in Visual Basics 6.0?

A list control in Visual Basics 6.0 is a control which helps to display data vertically to the user on the form of a list.

A list box has many entries and is selected by the user. These entries are not generated automatically, there are entered by methods **AddItem**, the method **RemoveItem** help to delete these entries from a list. To clear the whole content of a list we use **Listname.clear** , To refresh list content we use Listname.Refresh

Example:

List the following department names in a Visual Basics 6.0 list control:

Computer Electronics

Computer science

Accounting

Secretary

Construction To add these elements on a list we write the following codes:

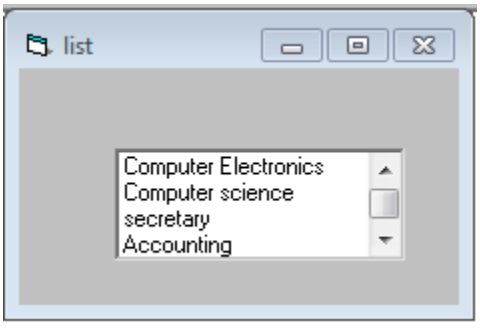
ListA.AddItem “Computer Electronics”

ListA.AddItem “Computer science”

ListA.AddItem “secretary”

ListA.AddItem “Accounting”

ListA.AddItem “Construction”



3.9 ComboBox

A comboBox control is a combination of TextBox control and ListBox control on only one entity. It allows to add entries or to select from a list predefined values.

List the following department names in a Visual Basics 6.0 ComboBox control:

Computer Electronics

Computer science

Accounting

Secretary

Construction

To add these elements on a ComboBox we write the following codes:

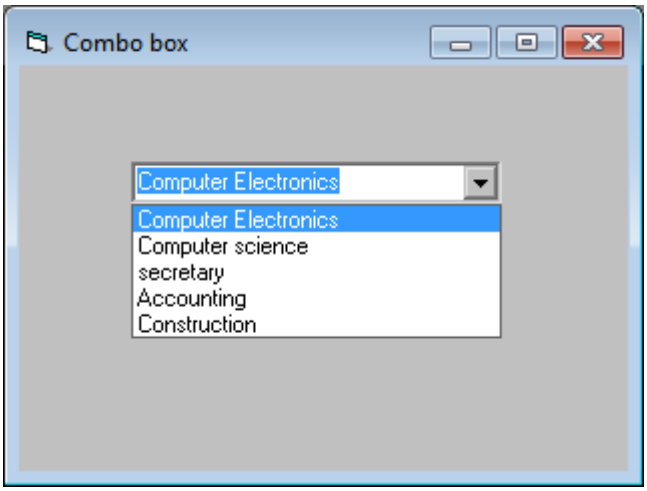
Combo1.AddItem “Computer Electronics”

Combo**1**.AddItem “Computer science”

Combo1.AddItem “secretary”

Combo1.AddItem “Accounting”

Combo1.AddItem “Construction”



Note: the procedure for add a list box and combo box must be placed in form not in control itself, and they are two way to add it.

- ✓ One by using procedure

  Private sub form_Load()

  Combo1.AddItem “programming”

  Combo1.AddItem “networking”

  End sub

- ✓ Second by pass to list an combo box control properties

CHAPTER 4: CODE ELEMENT

4.1 Declaration of variables and constants and data types

4.1.1 Data types used in visual basic

Visual Basic is rich in its data types. Data types are used to declare the variables. At the time of declaration, memory is allocated for the variables. Different data types are used to store different types of values.

Table: Memory storage for data types

Data Type	Storage	Data Type	Storage
Byte	1 byte	String (variable-length)	Length + 10 bytes
Boolean	2 bytes	String (Fixed-Length)	Length of string
Integer	2 bytes	Currency	8 bytes
Long	4 bytes	Decimal	12 bytes
Single	4 bytes	Object	4 bytes
Double	8 bytes	Variant (numeric)	16 bytes
Date	8 bytes	Variant (text)	length +22 bytes

Table: Data types & their value range

Data Type	Value Range
Byte	0 to 255
Boolean	True/False
Integer	-32,768 to 32,767
Long	-2,147,483,648 to 2,147,483,647
Single	-3.402823*10^3 to -1.401298*10^45 for negative values 1.401298*10^-45 to 3.402823*10^38 for positive values
Double	-1.79*10^308 to -4.94*10^-324 for negative values 4.94*10^-324 to 1.79*10^308 for positive values
Date	January 1, 100 to December 31, 9999
String (Variable length)	0 to approximately 2 billion characters
String (Fixed length)	1 to 65,400 characters
Currency	-922,337,203,685,477.5808 to 922,337,203,685,477.5807
Decimal	+,-79,228,162,514,264,337,593,543,950,335 if no decimal is used +,-7.9228162514264337593543950335 (28 decimal places)
Object	Any object
Variant (numeric)	Any value as 16px as Double
Variant (text)	Same as variable length string

4.1.2 Declaration of variable and constant

4.1.2.1 Variables

Variable is used to store value. The value of the variable may vary during the program execution.

Declaration of a variable

The declaration means defining the variable type. When you declare a variable, memory space for the variable is reserved. This is called memory allocation. Different amount of memory space is reserved for different data types.

You can declare a variable with the Dim keyword.

Syntax:

Dim variable As [Type]

Example:

```
Private Sub cmdSum_Click()
Dim m As Integer
Dim n As Integer
Dim sum As Integer

m = 10 'm is a variable, 10 is a constant
n = 30
sum = m + n

Print "The sum is " & sum

End Sub
```

Output: The sum is 40

You can declare many variables in one line as follows and assign multiple variables in one line using ':' operator.

```
Private Sub cmdSum_Click()
Dim m As Integer, n as Integer, sum as Integer
m = 10 : n = 30
sum = m + n

Print "The sum is " & sum

End Sub
```

Output: The sum is 40

Explicit declaration: is the variable declared in declaration section or at the beginning of a procedure.

Example:

Dim mynumber as integer.

Implicit declaration: The variable is declared "on the fly" its data type is deduced from other variable:

Example: Dim firstnumber As integer ' explicit declaration

Dim secondnumber As integer 'explicit declaration

Result= firstnumber + secondnumber '**Implicit declaration**

Rules of naming variable

- ✓ A variable name must begin with an alphabet.
- ✓ It cannot be more than 255 characters.
- ✓ The variable name must not contain any special character like %,&,!,#,@ or $.
- ✓ It has to be unique within the same scope.
- ✓ They may include letters, numbers, and underscore(_)
- ✓ You cannot use a reserved word or keyword

Dim X As Integer
Dim Balance As Currency
Dim Y As Long
Dim A AS Double,
B As Double
Dim Month As Date
Dim Max As Single
Dim Name As String
Dim Z,V,C

4.1.2.2 Declaration of Constants

Constant: Constant is a fixed value that does not change during the program execution. You can define your own constant to use it in your program.

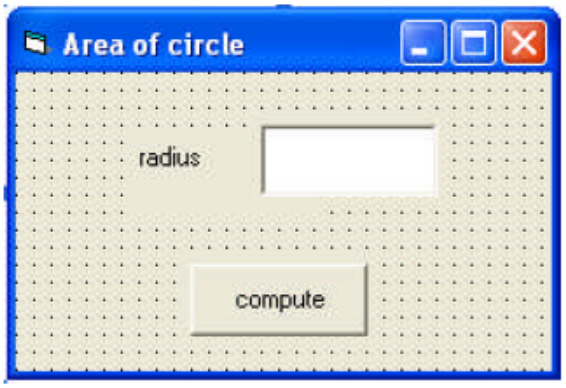
It is a space in memory filled with fixed value that will not be changed.

Constant may be declared as:

Const constant name = value

Example: Declare x as a constant (P), then compute the area of a circle. Put suitable design.

Sol:



Form1	
caption	Area of a circle
label1	
Caption	radius
Text1	
text	فارغ
Command1	
caption	compute
Enabled	false

code stage:

```
Const p = 3.14159
Dim a, r As Single

Private Sub Text1_Change()
Command1.Enabled = True
End Sub

Private Sub Command1_Click()
r = Val (Text1.Text)
a = r ^ 2 * p
MsgBox ("area=" & a)
Text1.Text = " "
Text1.SetFocus
End Sub
```

Example2:

You can create your own constant to use it in your program. The value of the constant remains unchanged throughout the program.

Example3:

```
Private Sub cmdCalculate_Click()
Const pi = 3.1415926 'or Const pi As Double = 3.1415926
Dim area As Double, r As Double
r = 2
area = pi * r * r
Print area
End Sub
```

Output: 12.5663704

4.2 Visual Basic operators

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

Operation Code	Operation
^	Exponent
*,/	Multiplication and division
\	Integer division
Mod	Modulus – rest of division
-,+	Subtraction and addition

2- To Concatenate two strings, use the **&** symbol or the **+** symbol

4 There are six Comparison operators in Visual Basic.

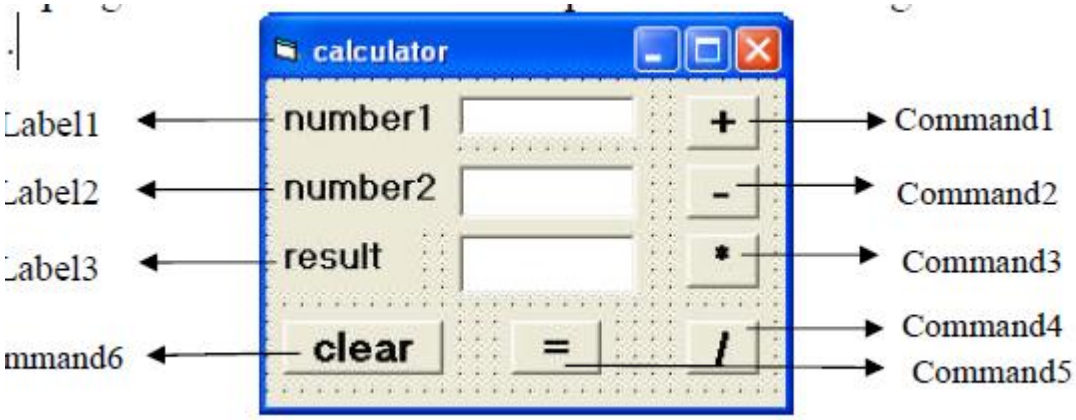
Operation Code	Comparison
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
=	Equal to
< > or > <	Not equal to

4-There are three logical operators:

Operation Code	Operation
Not	Logical not
And	Logical and
Or	Logical or

Note: Logical operators follow arithmetic operators in precedence.

Example3: write a program to execute the four operations according to the following design.



Code stage:

```
Dim a,b, c as single
Private sub command1_click ()
a = val(text1.text)
b = val(text2.text)
c = a + b
 text3.text = c
End sub

Private sub command2_click ()
a = val(text1.text)
b = val(text2.text)
c = a – b
text3.text = c
End sub
```

```
Private sub command3_click ()
a = val(text1.text)
b = val(text2.text)
c = a * b
text3.text = c
End sub

Private sub command4_click ()
a = val(text1.text)
b = val(text2.text)
c = a / b
text3.tex t= c
End sub

Private sub command5_click ()
Text3.text=cstr(c)
End sub
Private sub command6_click ()
Text1.text=" "
Text2.text=" "
Text3.text=" "
End sub
```

CHAPTER5: CONDITIONAL STATEMENTS

There are two types of conditional statements:

1- If statement

2- Select case

5.1. If statement

The comparison operations are used with conditional statements. The comparison operations are: (<, <=, >, >=, =, <>, and, or)

There are four structures for if statement.

a) Simple structure If.. then:

Used for running one programming statement only if the required condition satisfied.

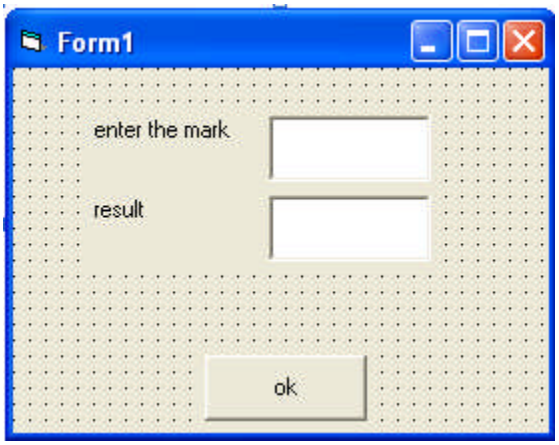
The general form is:

If *condition* ***then*** *statement*

Example 1: write a program to enter a mark of a student then print (pass) if he successful.

Sol:

```
Dim x as integer
Private sub command1_click()
X= val (text1.text)
If x>= 50 then text2.text= "pass"
End sub
```



b) **If block structure:** Used for running many programming statements if the required condition satisfied.

The general form is:

If condition **then**

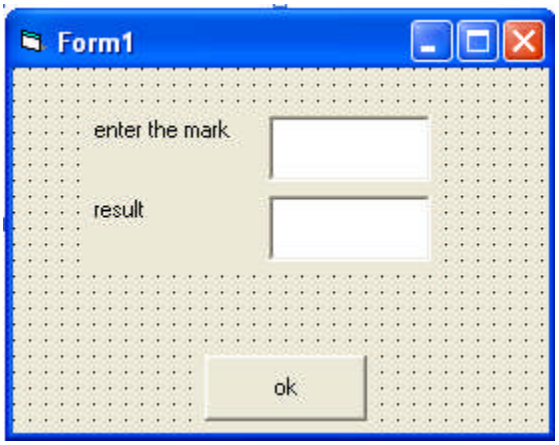
Statements

End if

Example 2: write a program to enter a mark of a student then print (pass) in size 18 if he successful.

Sol:

```
Dim x as integer
Private sub command1_click()
X= val(text1.text)
If x>= 50 then
text2.text= "pass"
text2.fontsize=18
end if
End sub
```



c) **If.. Then.. Else structure:** Used for running many programming statements if the required condition satisfied. And running another programming statements (after else) if the required condition not satisfied.

The general form is:

If condition **then**

Statements

Else

Statements

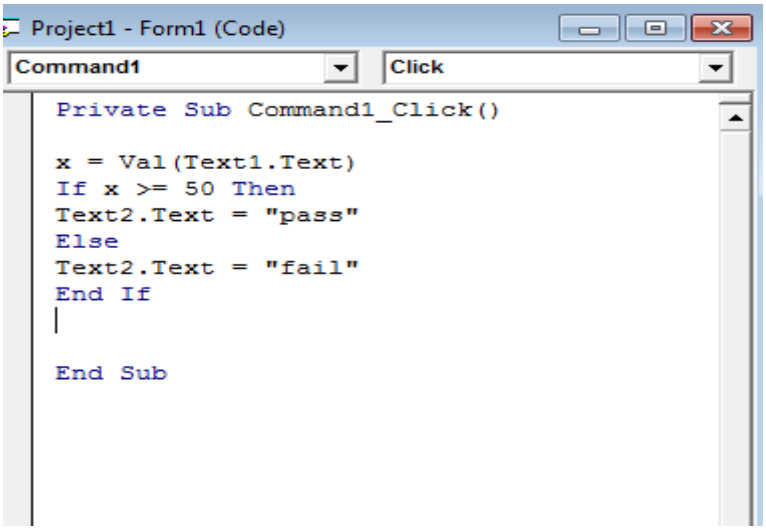
End if

Example3: write a program to enter a mark of a student then print (pass) if he successful and print (fail) otherwise.

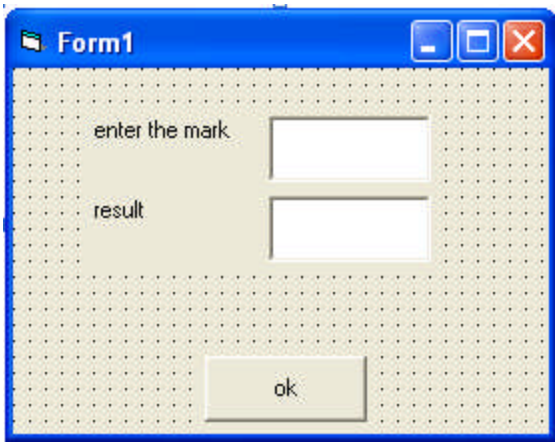
Sol:

Dim x As Integer

Private Sub command1_click()



End Sub



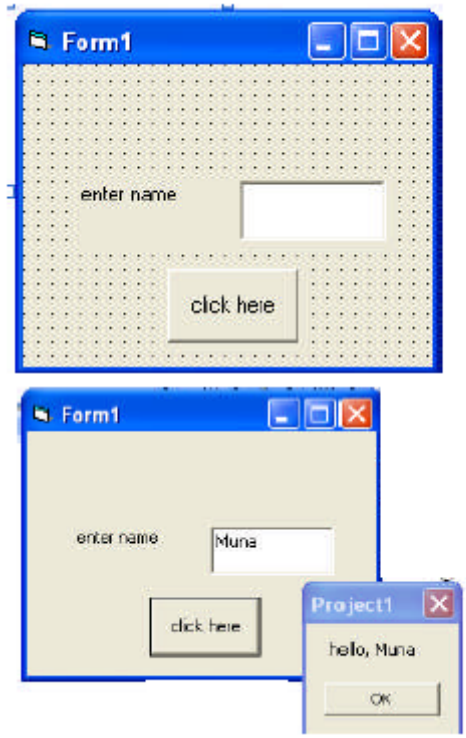
d) **If.. Then.. Elseif.. Else structure:**

Used if we have many conditions to be satisfied

Example 4: write a program to enter a user name and display the message (hello) three times. The first one for (Muna), the second one for (Maha) and the third for any user as a guest.

Sol:

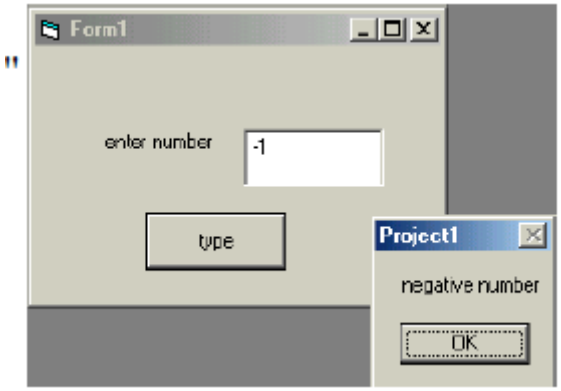
```
Dim x As String
Private Sub command1_click()
x = Text1.Text
If x = "Muna" Then
MsgBox "hello, Muna"
ElseIf x = "Maha" Then
MsgBox "hello,Maha"
Else
MsgBox "hello, guest"
End If
End Sub
```



Example 5: Write a program to classify any entered number according to its sign and display the phrase (negative number) when the number is negative and the phrase (positive number) when the number is positive, otherwise display the phrase (neither positive nor negative).

Sol:

```
Dim x As Single
Private Sub command1_click ()
x = Val (Text1.Text)
If x > 0 Then
MsgBox "positive number"
ElseIf x < 0 Then
MsgBox "negative number"
Else
MsgBox "neither positive nor negative"
End If
End Sub
```



Nested If statement:

It can be takes the following structure:

```
If condition then
    If condition then
        Statements
    End if
End if
```

structure2 *structure 1*

Note: Any structure of if structures can be used insteade of structure 1 and 2 above.

5.2 Select case statement

If you have a lot of conditional statements, using If..Then..Else could be very messy. For multiple conditional statements, it is better to use Select Case .

The format is :

```
Select Case expression
Case value1
Block of one or more VB statements
Case value2
Block of one or more VB Statements
Case value3
Block of one or more VB statements
Case value4
Case Else
Block of one or more VB Statements
End Select
```

Example 7: write a program to print the days of the week when we enter its number

Sol:

```
Dim x As Integer
Private Sub Command1_Click()
x = val(Text1.Text)
Select Case x
Case 1
MsgBox ("Sunday")
Case 2
MsgBox ("Monday")
```

Case 3

MsgBox ("Tuesday")

Case 4

MsgBox ("Thursday")

Case 5

MsgBox ("Wednesday")

Case 6

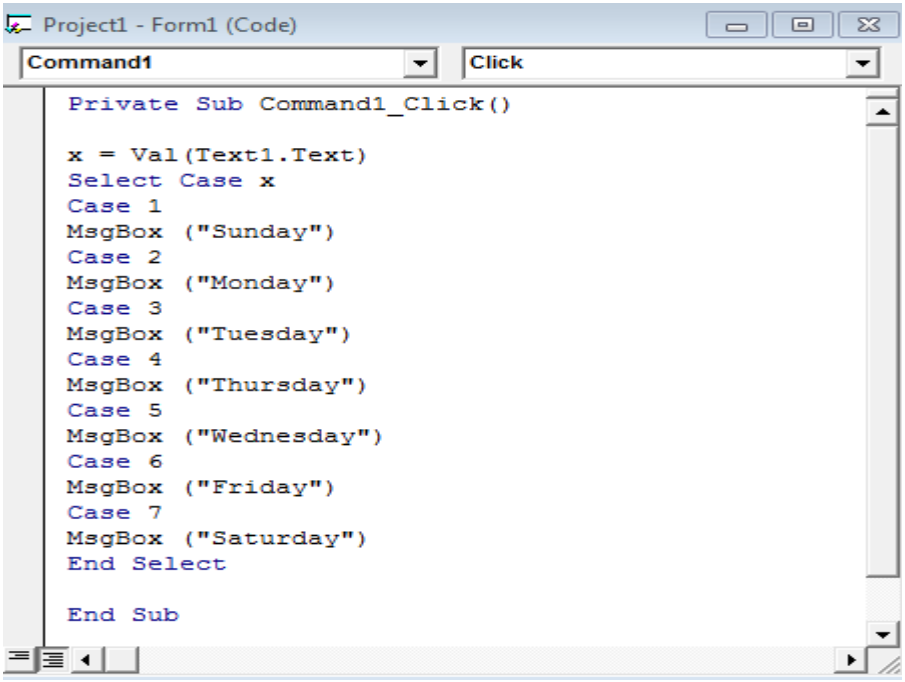
MsgBox ("Friday")

Case 7

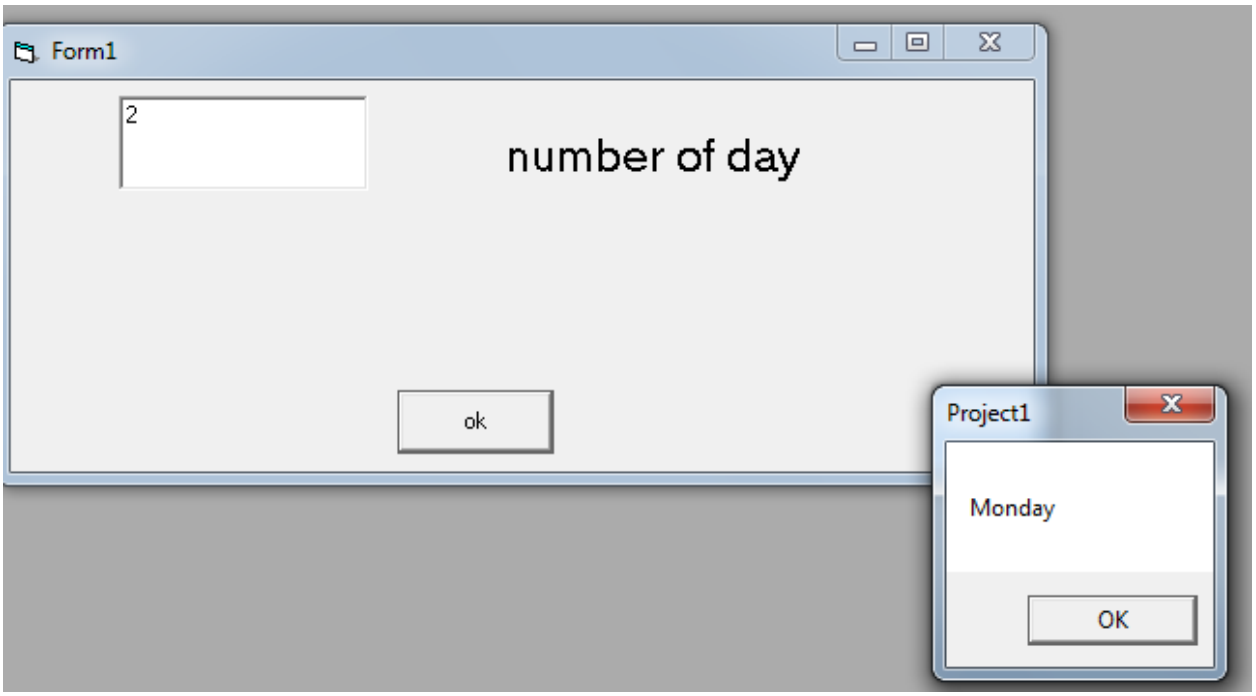
MsgBox ("Saturday")

End Select

End Sub



Running



Example2: Examination Grades

```
Dim grade As String
Private Sub Compute_Click( )
Grade = txtgrade.Text
Select Case grade
Case "A"
result.Caption = "High Distinction"
Case "A-"
result.Caption = "Distinction"
Case "B"
result.Caption = "Credit"
Case "C"
result.Caption = "Pass"
Case Else
result.Caption = "Fail"
End Select
```

*Please note that grade is a string, so all the case values such as "A" are of String data type.

Example3

```
Dim mark As Single
Private Sub Compute_Click()
'Examination Marks
mark = mrk.Text
Select Case mark
Case Is >= 85
comment.Caption = "Excellence"
Case Is >= 70
comment.Caption = "Good"
Case Is >= 60
comment.Caption = "Above Average"
Case Is >= 50
comment.Caption = "Average"
Case Else
comment.Caption = "Need to work harder"
End Select
```

NB: * Note we use the keyword Is here to impose the conditions. This is generally used for numeric data.

CHAPTER 6: LOOPS STATEMENTS

6.1 Looping

The format are:

i) While – wend

ii) Do - while

iii) For - next

While – wend

While – wend first checks whether the condition is true. The statements will be executed as long as the condition remains true.

The general form is

WHILE <condition>

Statements

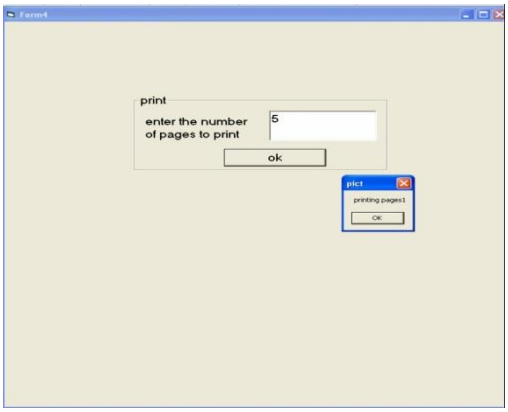
WEND

Example

Aim: To print the number of pages

Properties

Control	Property	Value
Frame1	*Caption*	*Print*
Label1	*Caption*	*Enter the number of pages to print*
Text1	*Text*	*Empty*
Command1	*Caption*	*ok*



Code

```
Private Sub Command1_click ()

While i < Val(Text1.Text)

i = i + 1
```

MsgBox ("printing pages" & i) Wend

End Sub

DO – WHILE

Do –loop executes a block of statements as long as the condition is true. If the expression is false the program continues to the statement following the loop statement and that will be executed

The general format is

DO WHILE <condition>
Statements

LOOP

Aim: To find out number of words in a sentence

Properties

Control	**Property**	**Value**
Frame1	*Caption*	*Frame1*
Text1	*Text*	*Empty*
Label1	*Caption*	*Enter a line of text*

Code

Private Sub General-declarations() Dim d
As String

Private Sub Text1_MouseDown()

d = Text1.Text
position = 1

Do While position > 0

position = InStr(position + 1, d, " ")

words = words + 1

Loop

MsgBox ("the number of words is " & words) End
Sub

Instr() function finds space in the text and returns positive number. When no more spaces, it returns 0 and instr(start point, line of text to be counted, the parameter to count i.e. space)

Example .1

Do while counter <=1000
num.Text=counter
counter =counter+1
Loop
* The above example will keep on adding until counter >1000.
The above example can be rewritten as
Do
num.Text=counter
counter=counter+1
Loop until counter>1000

6.2 For....Next Loop

The format is:

For counter=startNumber to endNumber (Step increment)

One or more VB statements

Next

Example:

(a) For counter=1 to 10

display.Text=counter

Next

(b) For counter=1 to 1000 step 10

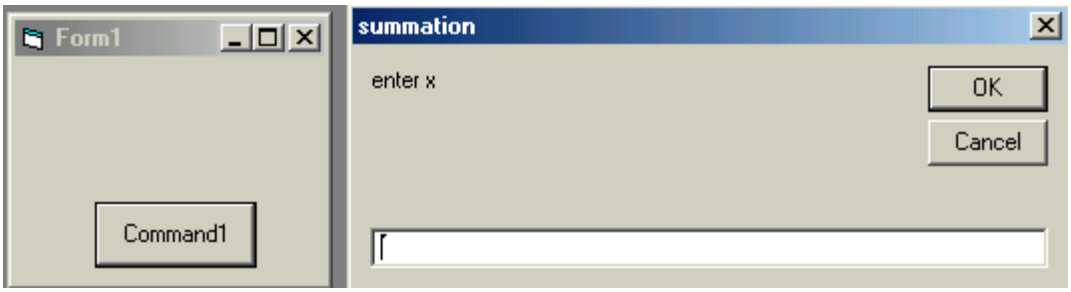
counter=counter+1

Next

Example6: Write a program to find the summation of undetermined number of positive numbers such that the program will be stopped when we enter negative number.

Sol:

```
Dim x, sum As Single
Private Sub command1_click()
sum = 0
x = Val(InputBox("enter x", "summation"))
Do While x >= 0
sum = sum + x
x = Val(InputBox("enter x", "summation "))
Loop
MsgBox (CStr(sum))
End Sub
```



Example7: Write a program to find the summation of the numbers from 5 to 15.

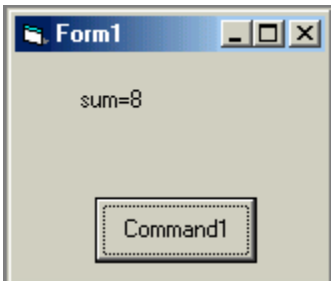
Sol:

```
Dim I, sum as integer
Private Sub command1_click ()
sum = 0
For i = 5 to 15
Sum = sum + i
Next i
Label1.caption = "sum ="&cstr(sum)
End Sub
```

Example8: Write a program to find the summation of 10 numbers.

Sol:

```
Dim i as integer
Dim x, sum as double
Private Sub command1_click ()
sum = 0
For i = 1 to 10
x = val(inputbox ("enter number"))
Sum = sum + x
Next i
Label1.caption = "sum="& cstr(sum)
End Sub
```



Running stage:

For example if we entered the numbers: 1, 5, -1, 3, 2, 0, -1, 3, 0, -4 then sum=8

Example9: Write a program to find the average of n numbers.

Sol:

```
Dim i as integer
Dim x, sum, av as Double
Private Sub command1_click ()
i = 1: sum = 0
n = val (text1.text)
Do while i <= n
x = val(inputbox ("enter number"))
Sum = sum + x
i = i + 1
Loop
Av = sum/n
Text2.text = av
End Sub
```

Example10: Write a program to print multipliers of 5 (from 5 to 50)

Sol:

```
Dim i as integer
Private Sub Command1_Click ()
i = 5
Do until i > 50
Print i
```

```
i = i + 5
Loop
End Sub
```

Example11: write a program to find the average of numbers that dividable by 3 (without remainder) from 3 to 99.

Sol:

```
Dim I, n, sum as integer
Dim av as Double
Private Sub command1_click ()
i = 3 : n = 0
sum = 0
Do while i <= 99
Sum = sum + i
i = i + 3
n = n + 1
Loop
Av = sum/n
Print "av ="; av
End Sub
```

Example12: write a program to print (welcome) ten times, the first one with the ordinary size and color. Then make the color changed and the size bigger at each
time.

Sol:

```
Dim i As Integer
Private Sub Command1_Click()
Print "welcome"
For i = 1 To 9
FontSize = 10 + i
ForeColor = QBColor(i)
Print "welcome"
```

Next i

End Sub

Series:

To compute the value of series, we use suitable loop statements according to the boundaries (limits) of each series.

Example13: Find

Sum=1+x+x2+x3+…+xn , where x is an integer.

Sol:

Dim I, n, x, sum as integer

Private Sub command1_click ()

sum = 1

n=val(text1.text)

x=val(text2.text)

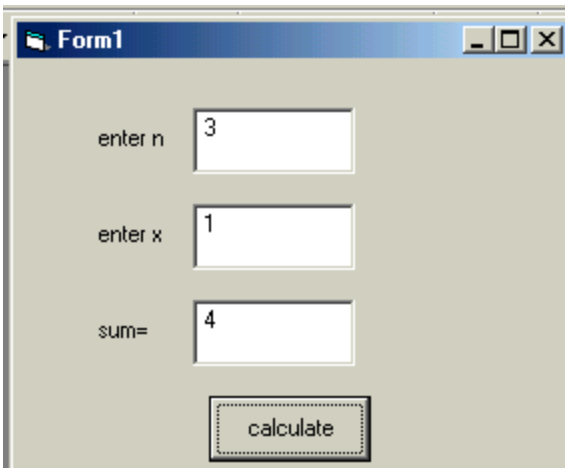
For i = 1 To n

Sum = sum +x^i

Next i

Text3.text=sum

End Sub



6.3: Nested loop

The nested loops are the loops that are placed inside each other. The most inner loop will be executed first, then the outer ones. These loops should neither intersect, nor have the same index. As follows:

For i = 1 To n

For j = 1 To m

Statements

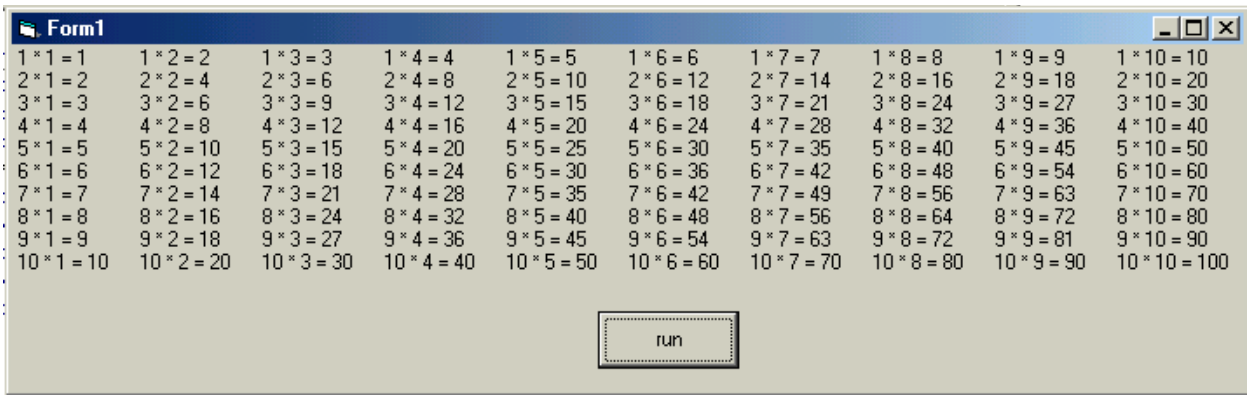
Next j

Next i

Example14: write a program to print the multiplication table.

Sol:

```
Dim I, j As Integer
Private Sub command1_click()
For I = 1 To 10
For j = 1 To 10
p = I * j
Print I; "*"; j; "="; p,
Next j
Print
Next I
End Sub
```



Example15: write a program to generate the numbers in the following form.

1

2

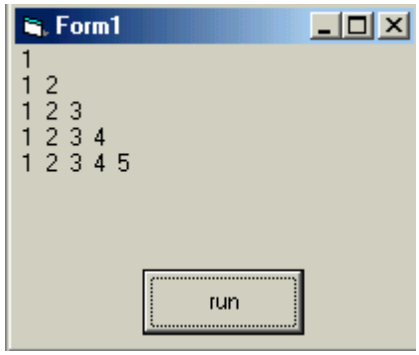
1 2 3

1 2 3 4

1 2 3 4 5

Sol:

```
Dim I, j As Integer
Private Sub command1_click()
For I = 1 To 5
For j = 1 To i
Print j;
Next j
Print
Next I
End Sub
```



CHAPTER 7: FUNCTIONS IN VISUAL BASIC

Functions are similar to normal procedures but the main purpose of the functions is to accept certain inputs and pass them on to the main program to finish the execution. They are two types of function, the built-in functions (or internal functions) and the functions created by the programmers.

The general format of a function is functionName(arguments) where arguments are values that are passed on to the functions.

We are going to learn two very basic but useful internal functions, i.e. the MsgBox( ) and InputBox ( ) functions.

7.1 MsgBox ( ) Function

The objective of MsgBox is to produce a pop-up message box and prompt the user to click on a command button before he /she can continues. This message box format is as follows:

yourMsg=MsgBox(Prompt, Style Value, T itle)

The first argument, Prompt, will display the message in the message box. The Style Value will determine what type of command buttons appear on the message box, please refer Table 7.1 for types of command button displayed. The Tit le argument will display the title of the message board.

Table 7.1: Style Values

Style Value	Named Constant	Buttons Displayed
0	vbOkOnly	Ok button
1	vbOkCancel	Ok and Cancel buttons
2	vbAbortRetryIgnore	Abort, Retry and Ignore buttons.
3	vbYesNoCancel	Yes, No and Cancel buttons
4	vbYesNo	Yes and No buttons
5	vbRetryCancel	Retry and Cancel buttons

We can use named constant in place of integers for the second argument to make the programs more readable. In fact, VB6 will automatically shows up a list of names constant where you can select one of them.

example: yourMsg=MsgBox("Click OK to Proceed", 1, "Startup Menu")

and yourMsg=Msg("Click OK to Proceed". vbOkCancel,"Startup Menu")

are the same.

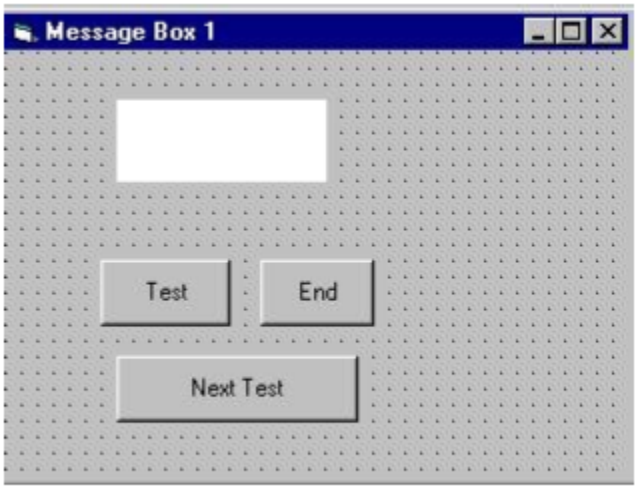
yourMsg is a variable that holds values that are returned by the MsgBox ( ) function. The values are determined by the type of buttons being clicked by the users. It has to be declared as Integer data type in the procedure or in the general declaration section. Table 7.2 shows the values, the corresponding named constant and buttons.

Table 7.2 : Return Values and Command Buttons

Value	Named Constant	Button Clicked
1	vbOk	Ok button
2	vbCancel	Cancel button
3	vbAbort	Abort button
4	vbRetry	Retry button
5	vbIgnore	Ignore button
6	vbYes	Yes button
7	vbNo	No button

i. The Interface:

You draw three command buttons and a label as shown in Figure 7.1



ii. The procedure for the test button:

```
Private Sub Test_Click()
Dim testmsg As Integer
testmsg = MsgBox("Click to test", 1, "Test message")
If testmsg = 1 Then
Display.Caption = "Testing Successful"
Else
Display.Caption = "Testing fail"
End If
End Sub
```

When a user click on the test button, the image like the one shown in Figure 10.2 will appear.As the user click on the OK button, the message "Testing successful" will be displayed and when he/she clicks on the Cancel button, the message "Testing fail" will be displayed.

To make the message box looks more sophisticated, you can add an icon besides the message. There are four types of icons available in VB as shown in Table 7.3

Value	Named Constant	Icon
16	vbCritical	
32	vbQuestion	
48	vbExclamation	
64	vbInformation	

In this example, the following message box will be displayed:

You could draw the same Interface as in example 10.1 but modify the codes as follows:

```
Private Sub test2_Click()
Dim testMsg2 As Integer
testMsg2 = MsgBox("Click to Test", vbYesNoCancel + vbExclamation, "Test Message")
If testMsg2 = 6 Then
display2.Caption = "Testing successful"
ElseIf testMsg2 = 7 Then
display2.Caption = "Are you sure?"
Else
display2.Caption = "Testing fail"
End If
End Sub
```

7.2 The InputBox( ) Function

An InputBox( ) function will display a message box where the user can enter a value or a message in the form of text. The format is

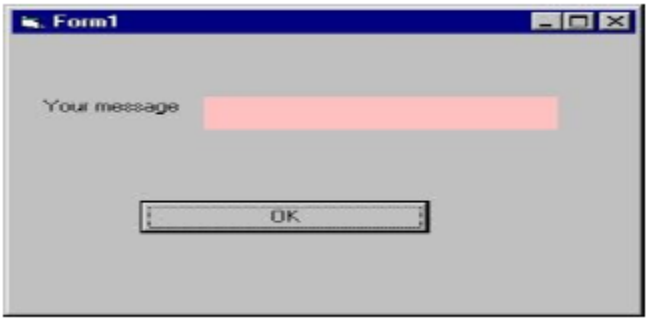
myMessage=InputBox(Prompt, Tit le, default_text, x-posit ion, y-posit ion)

myMessage is a variant data type but typically it is declared as string, which accept the message input by the users. The arguments are explained as follows:

- Prompt: The message displayed normally as a question asked.
- Title: The title of the Input Box.
- default-text: The default text that appears in the input field where users can use it as his intended input or he may change to the message he wish to key in.
- x-position and y-posit ion - the posit ion or the coordinate of the input box.

Example

i. The Interface

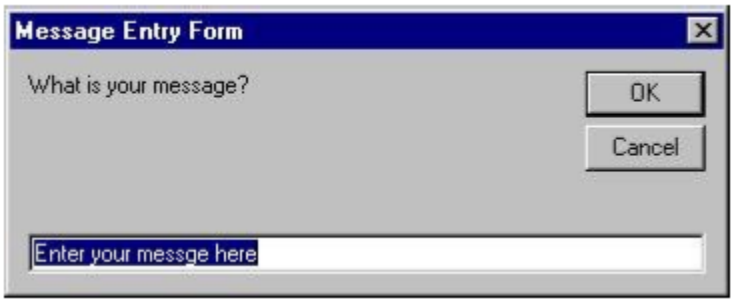


ii. The procedure for the OK button

```
Private Sub OK_Click()
Dim userMsg As String
userMsg = InputBox("What is your message?", "Message Entry Form", "Enter your messge
here", 500, 700)
If userMsg <> "" Then
message.Caption = userMsg
Else
message.Caption = "No Message"
End If
```

End Sub

When a user click the OK button, the input box as shown in Figure 10.5 will appear. After user entering the message and click OK, the message will be displayed on the caption, if he click Cancel, "No message" will be displayed.



7.3 VAL() Function

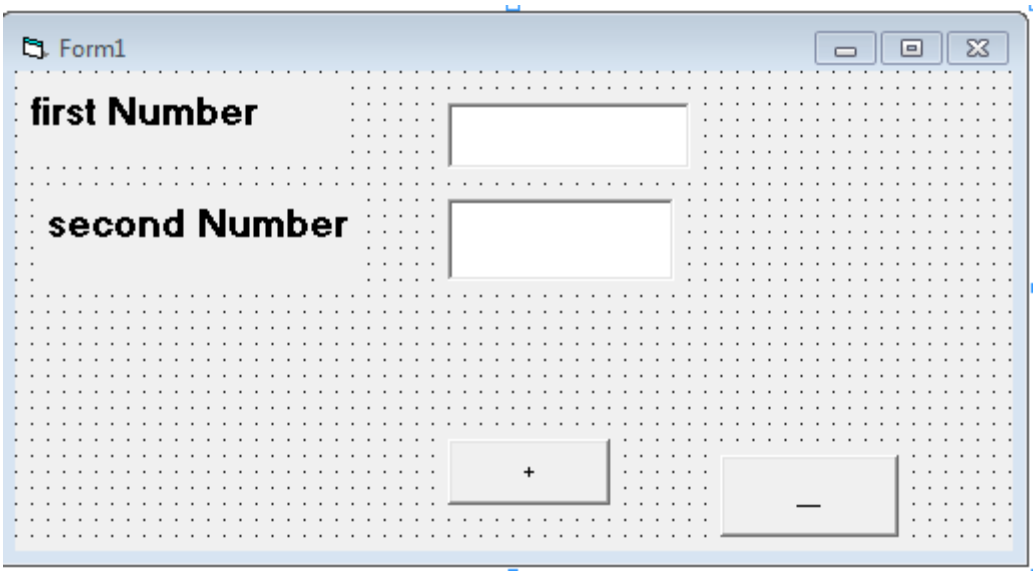
Visual Basic provides number of built in functions. The Val function is one of it. Val function converts the text data into a numeric value.

The general form of Val function is as follows:

Val(Expression to Convert)

The **Str** is the function that converts a number to a string while the **Val** function converts a string to a number. The two functions are important when we need to perform mathematical operations.

Example2: Write a program to add and subtract two integer numbers after putting a suitable design. Use message box for outputting.

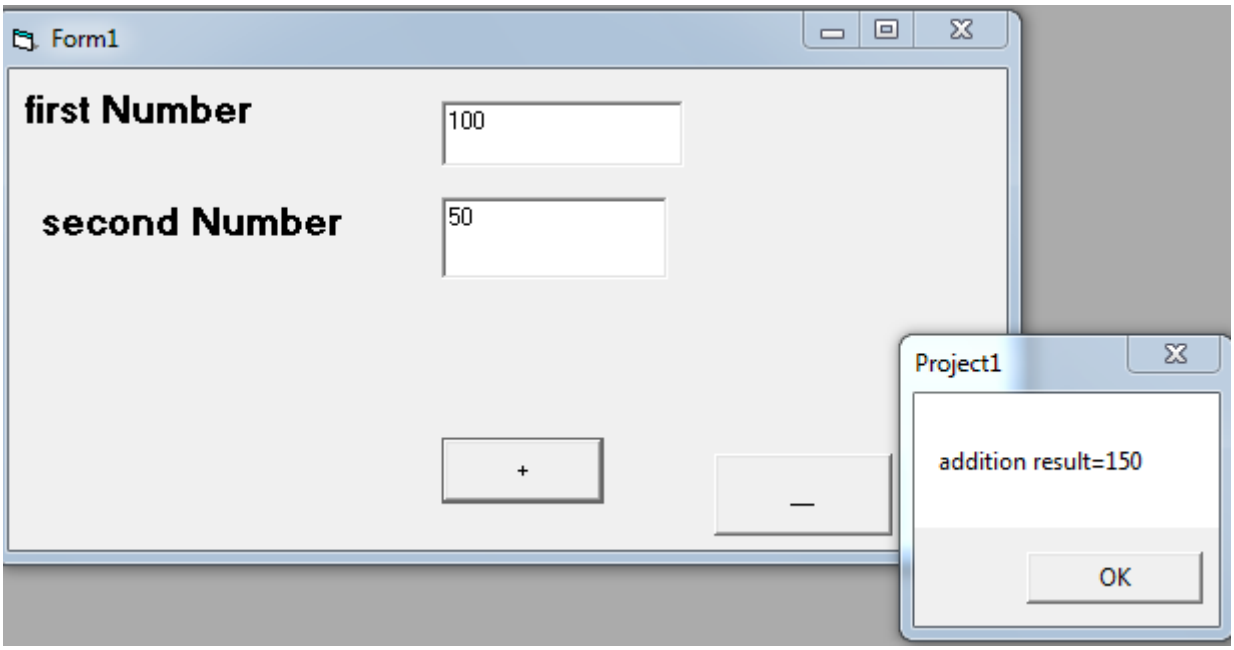


Code:

```
Dim x, y, z As Integer

Private Sub Command1_Click()

x = Val(Text1.Text)

y = Val(Text2.Text)

z = x + y

MsgBox ("addition result=" & z)

End Sub

Private Sub Command2_Click()

x = Val(Text1.Text)

y = Val(Text2.Text)

z = x - y

MsgBox ("subtraction result=" & z)

End Sub
```

Running stage

Enter two values in text1 and text2. When click on command (+) or (-) the addition or subtraction result appears in message box.



7.4 Sqr() Function

The Sqr function returns the square root of a number. This function takes a Double type argument and returns a double implying that you can work with large numbers using the Sqr function.

Example:

Print Sqr(9)

Output: 3

7.5 Ucase() Function

The **Ucase** function converts all the characters of a string to capital letters.

Syntax: **UCase(*string*)**

Example: strNew = UCase("Visual Basic") ' strNew = "VISUAL BASIC"

7.6 Lcase() Function

The **Lcase** function converts all the characters of a string to small letters.

Syntax: **LCase(*string*)**

Lcase("Visual Basic") =visual basic

7.7 Len () Function

Returns a Long containing the length of the specified string

Synthax: Len(*string*)

Where *string* is the string whose length (number of characters) is to be returned.

Examples:lnglen=Len("visual basic" 'lnglen=12

CHAPTER 8: VISUAL BASIC MENUS

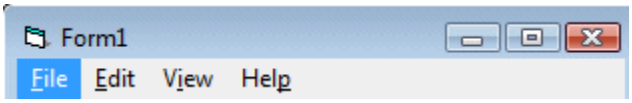
8.1 Menu creation

Definition

An on-screen list of available functions, or operations, that can be performed currently.

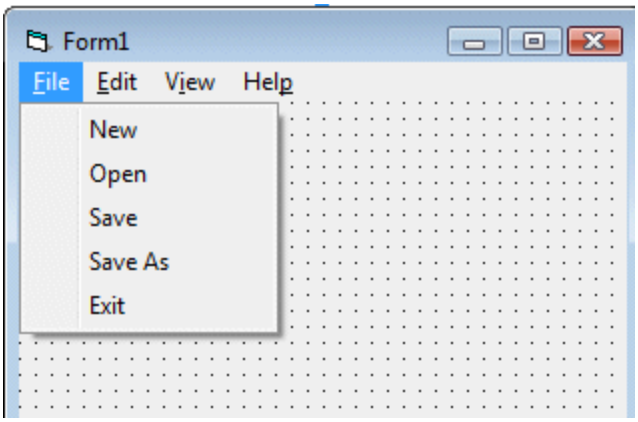
Menu bar

A row of menu titles typically located at the top of the application's window on screen. The File menu is often the first menu title on the menu bar. See menu bar:



Menu panel

The formal name for the list of options that is displayed when you select a menu from the menu bar. See meu pane.

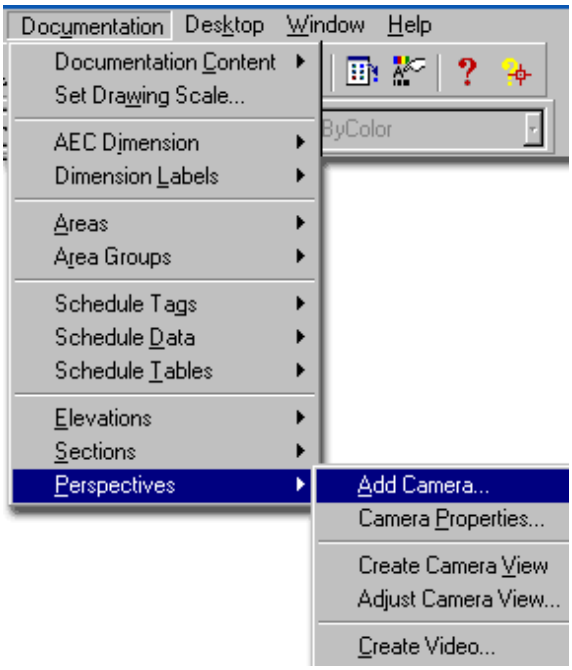


Pull-down menu

Also called a "drop-down menu" or "pop-down menu," the common type of menu used with a graphical user interface (GUI). Clicking a menu title causes the menu items to appear to drop down

from that position and be displayed. Options are selected either by clicking the menu item or by continuing to hold the mouse button down and letting go when the item is highlighted.

See drop down menu



How to create menu in VB?

VB programmers can create menus by first selecting the **form** that will host the menu and then using the **VB Menu Editor.**

- The Menu Editor is available only when a form is being designed. It is located on the Tools menu in VB6.

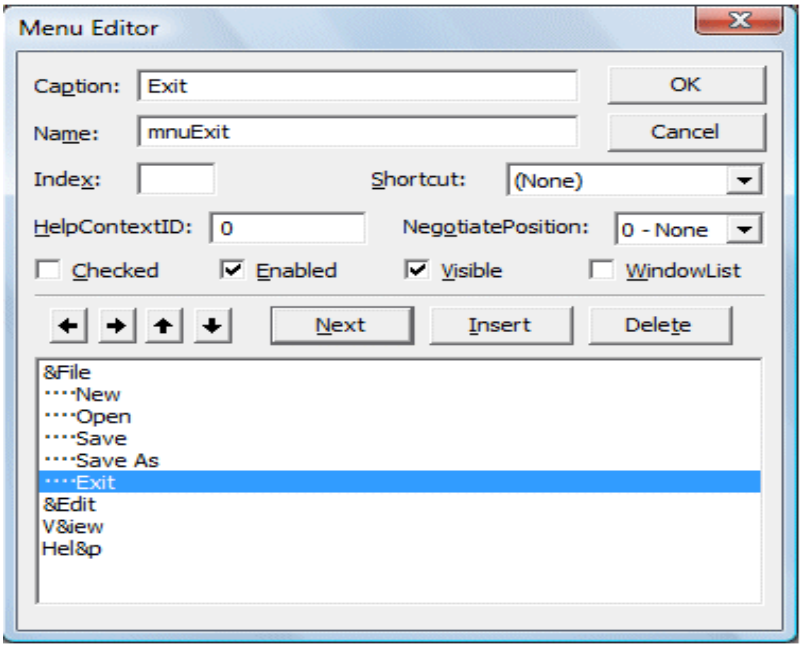
The first step in creating a menu is to enter the menu item's **caption.** The caption can incorporate the **ampersand (&)** for an access key designation, also known as an accelerator key. This enables the user to see an underlined letter in the menu and use the Alt key along with the accelerator key.

After the caption of the menu item has been set, the menu requires an **object name** for programmatic reference. The **menu names** can be any valid object name. Naming conventions are

preferred to allow for easier reading of source code and quick identification of objects Menus use the three-**letter prefix "mnu"** before the selected name.

You can implement a **separator bar** in a menu (separator bar is: **a horizontal line that visually defines the boundary between two groups of menu items)** by putting a single dash **(-)** as the entire caption for a menu item. This item then becomes the separator bar on the runtime menu. You must remember that even separator items in a menu must each have their own unique Name property.

- Menu items can also have their appearance properties set at design time through the Menu Editor. Properties such as **Checked**, **Enabled**, **Visible**, **WindowList**, and a **shortcut key** can all be specified.



In addition a drop-down list of possible shortcut key combinations allows the programmer to assign a shortcut key to this particular menu item. Unlike the accelerator or access key mentioned above, the shortcut key can be pressed by the user without its corresponding menu item being visible. Windows automatically displays the shortcut key assignment when the menu item is displayed. All of these properties are also available at runtime except for the shortcut key.

How to change properties of menu?

To change the desired property of the menu object, just use the object **name** followed by the **property name** and **the value.** Examples of such runtime changes are given in the following section.

e.g.: file. Visible = True

Attaching Code to a Menu Item's Click Event Procedure

A menu control has a single event procedure: the **Click** event procedure. The Click event procedure is the place where you write or call the code that you want to execute when the user chooses the menu item.

E.g.

```
Private sub mnuExit_click()

End

End sub
```

You can access a menu control's Click event procedure by single clicking the menu item from the design time copy of the menu.

8.2 Submenu creation

The submenu refers to the level in the menu structure the items will appear. An object may be a top-level menu item, sub-menu item, or a sub–sub-menu item.

Sub-level menu items have four small dots (**….**) preceding the menu caption; sub– sub-menu items have eight dots (**……..**). The menu items and sub menu can also be reorganized according to position in the menu.

To control the level and position of the menu item being entered, just use the four direction arrows in the Menu Editor. The up and down arrows reposition menu items for order. The left and right arrows allow menu items to be top-level, sub-level or sub–sub-level.

***Aim**: to create a menu of Edit and do the operations of Cut, Copy, Paste.*

Properties

Control	Property	Value
Text1	*Caption*	*Empty*
Text2	*Caption*	*Empty*

Menu editor

Menu	Property	Value
Edit	*Caption*	*Enter the day of the week*
	Name	*ed*
Cut	*Caption*	*Cut*
	Name	*edcu*
Copy	*Caption*	*Copy*
	Name	*edcop*
Paste	*Caption*	*Paste*
	Name	*edpas*

Code

```
Private Sub edcop_Click()

Clipboard.SetText  Screen.ActiveControl.SelText

End Sub

Private Sub edcu_Click()

Clipboard.SetText  Screen.ActiveControl.SelText

Text1.text=" "  End Sub



Private Sub edpas_Click()

Screen.ActiveControl.SelText = Clipboard.GetText()
```

End Sub

The code explanation

Seltext means selected text in the active control will be set as text for copying or cutting and it is sent to the clip board. The text in the clipboard will be put on the active control of the screen. The active control here is the text box control.

You will be getting output appear on the second text box

CHAPTER 9: DATABASE PROGRAMMING

This is the most important part of visual basic whereby we are going to produce a back

end where we are going to store the large volume of data.

Database

A database is a collection of records that can be manipulated with ease. The manipulation

of data is done by structured query language (SQL).the SQL is a simple English like language that user's can use to request a database for details, Often abbreviated as DB. A collection of information organized in such a way that a computer program can quickly select desired pieces of data. You can think of a database as an electronic filing system.

Traditional databases are organized by fields, records, and files. A field is a single piece of information; a record is one complete set of fields; and a file is a collection of records. For example, a telephone book is analogous to a file. It contains a list of records, each of which consists of three fields: name, address, and telephone number.

To access information from a database, you need a database management system (DBMS). This is a collection of programs that enables you to enter, organize, and select data in a database.

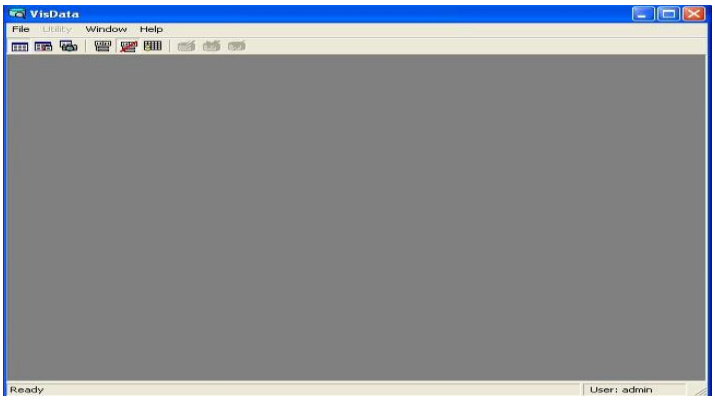
Increasingly, the term database is used as shorthand for database management system. In Visual Basic it is incorporated in the form of Visual Data Manger.

For designing database from visual basic u choose *Add-Ins* menu→ *Visual Data Manger*.

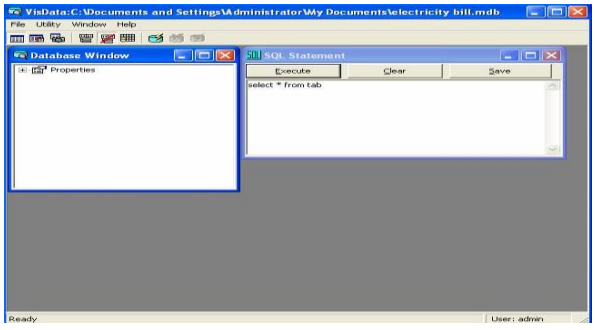
You will be getting a Visdata screen like one shown below.

In that go to *file* menu → *new* → *Microsoft access* →*version 7.0 mdb*

It will ask for the name of the database .give name (user defined).for example I m going to give as electricity bill

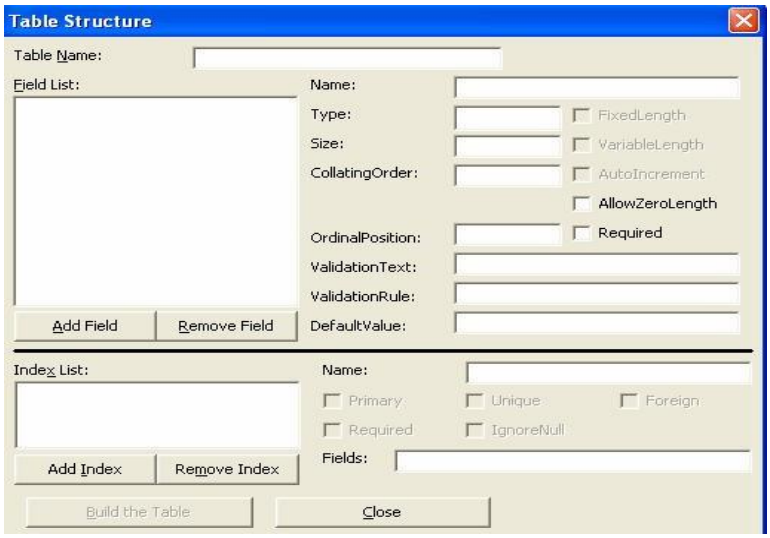


Then you will be getting a screen like this below



Next

In this right click on the properties you will be getting a menu, in that select a new table.



Give the *table name* and click on the *add field*.

Include all the fields that is necessary (for ex here customer's name, bill no, price/unit, unit, total amount) and then click build the table. When you want to include values to the records then double click on the table name and it will open for input of data and give values and then click *update*. When once giving values is finished then click close.

Note down the path of the database where you have stored. Now designing the database has been finished.

Now it is the turn of the front end. Take up a form.

Data control 

Data control provides access to databases through controls on your form. It makes the job of the developer easier when data has to be manipulated in a database.

Example

Aim: we are going to produce an electricity bill

Properties

Control	Property	Value
Label1	*Caption*	*Electricity bill*
Label2	*Caption*	*Customer's name*
Label3	*Caption*	*Bill no*
Label4	*Caption*	*Unit*
Label5	*Caption*	*Price per unit*
Label6	*Caption*	*Total*
Text1	*Caption*	*Empty*
Text2	*Caption*	*Empty*
Text3	*Caption*	*Empty*
Tect4	*Caption*	*Empty*
Text5	*Caption*	*Empty*
Data1	*Caption*	*Empty*

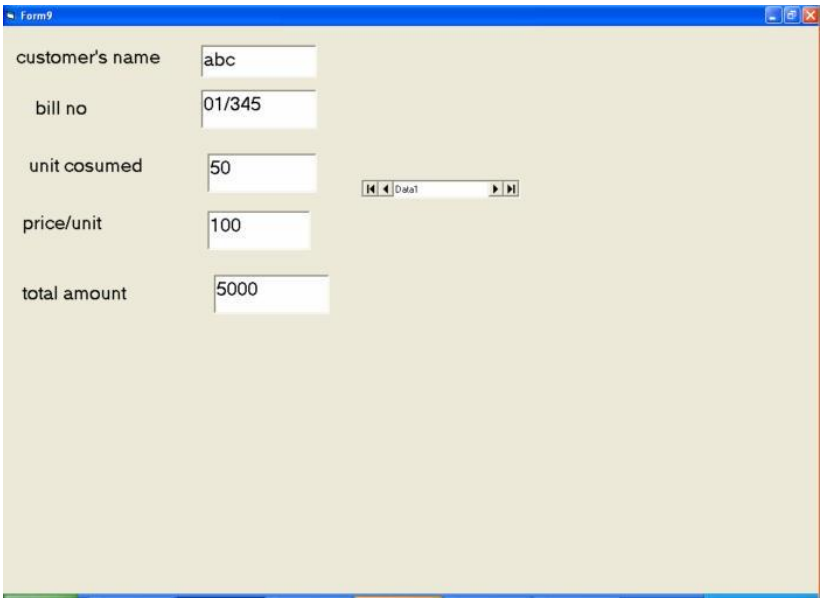
Connecting database and displaying corresponding fields to the form do the following

Control	Property	Value
Data1	*Data source*	*Select the Database name*
	Record source	*Select the record name*
Text1	*Data source*	*Data1*
	Data field	*Customer's name*
Text2	*Data source*	*Data1*
	Data field	*Bill no*

Text3	*Data source*	*Data1*
	Data field	*Unit*
Text4	*Data source*	*Data1*
	Data field	*Price per unit*
Text5	*Data source*	*Data1*
	Data field	*Total amount*

Now run your application it will show the records in the database. You can navigate between the records.

Output



To make the database work thro' code

Example

Aim: To have the information of employees

Properties:

Control	Property	Value
Label1	*Caption*	*Name*
Label2	*Caption*	*Id*
Label3	*Caption*	*age*
Label4	*Caption*	*department*
Label6	*Caption*	*Total*
Text1	*Caption*	*Empty*
Text2	*Caption*	*Empty*
Text3	*Caption*	*Empty*
Tect4	*Caption*	*Empty*
Data1	*Caption*	*Empty*

To set up the database

Control	Property	Value
Data1	*Data source*	*Select the Database name*
	Record source	*Select the record name*
	Visible	*False*
Text1	*Data source*	*Data1*
	Data field	*Name*
Text2	*Data source*	*Data1*
	Data field	*Id*
Text3	*Data source*	*Data1*
	Data field	*Age*
Text4	*Data source*	*Data1*
	Data field	*Dept*

Code:

```
Private Sub Command1_Click()
Data1.Recordset.MoveFirst

End Sub


Private Sub Command2_Click()
Data1.Recordset.MoveLast

End Sub


Private Sub Command3_Click()
Data1.Recordset.MoveNext

If Data1.Recordset.EOF Then MsgBox
"this is last record" End If

End Sub


Private Sub Command4_Click()
Data1.Recordset.MovePrevious If
Data1.Recordset.BOF Then MsgBox "this
is first record" End If

End Sub


Private Sub Command5_Click() End

End Sub
```

Private Sub Command6_Click()

Data1.Recordset.AddNew

End Sub

Private Sub Command7_Click()

Data1.Recordset.Delete

MsgBox "record deleted" End Sub

Private Sub Command8_Click()

Data1.Recordset.Update MsgBox "data

saved"

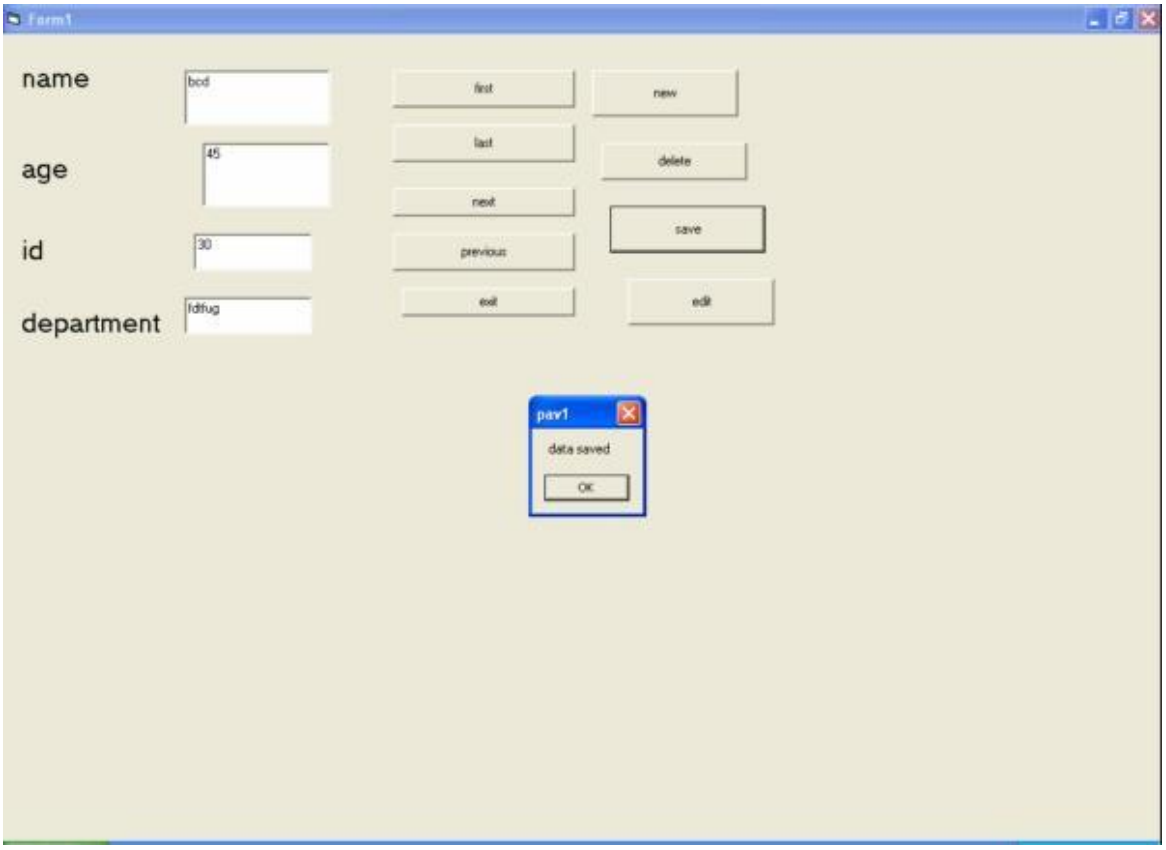
End Sub

Private Sub Command9_Click()

Data1.Recordset.Edit

End Sub

Output: navigate thro' the buttons provided


Form1
name
bcd
age
45
id
30
department
fdfug
first
new
last
delete
next
save
previous
exit
edit
pav1
data saved
OK

ADODC

The Background to ADO

In the early days of computing, dumb terminals were wired to powerful mainframe computers. The centralized Information Services (IS) department of a company ran the computing show. The mainframe gurus told us what we could and could not do. Then in

August of 1981, the first IBM personal computer was released and the world changed. Control was eventually wrested from the centralized IS department at companies and flowed to the every individual with a personal computer.

Each personal computer had its own CPU and hard drive to run programs and store data. Centralized computing for many day to day activities disintegrated and each person with a personal computer took individual control if their data destiny. But there were problems too. For example, lots of individual computers sitting on people's desks, and all wanted to share information and common data. Out of the many desktop solutions to this need to access data by distributed computing stations was Data Access Objects (**DAO**).

Now, with VB6.0, a brand new Jet database engine 3.51 just made this solution stronger. In fact, Jet 3.51 is faster and more robust than Jet 3.5 that shipped with VB 5.0. Microsoft wanted to enhance a proven, strong database access solution and, when developing desktop database solutions using .mdb or ISAM files, Microsoft says the combination of Jet and DAO is definitely the way to go. Microsoft has upgraded DAO and will continue to support this approach for the foreseeable future. And don't forget, DAO is the most popular desktop database access method around applications using DAO is tremendous.

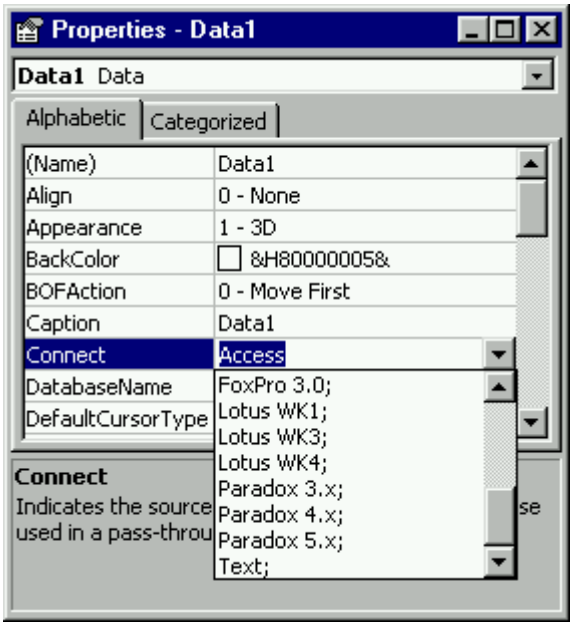
The Limitations of DAO

As seen two needs of modern business rapidly emerge that require a new and more sophisticated approach to gathering data.

- The first need is that of **accessing legacy data** - that is, information that is stored around the business enterprise in disparate forms in various types of computers.
- The second need is that of accessing **non-relational** data.

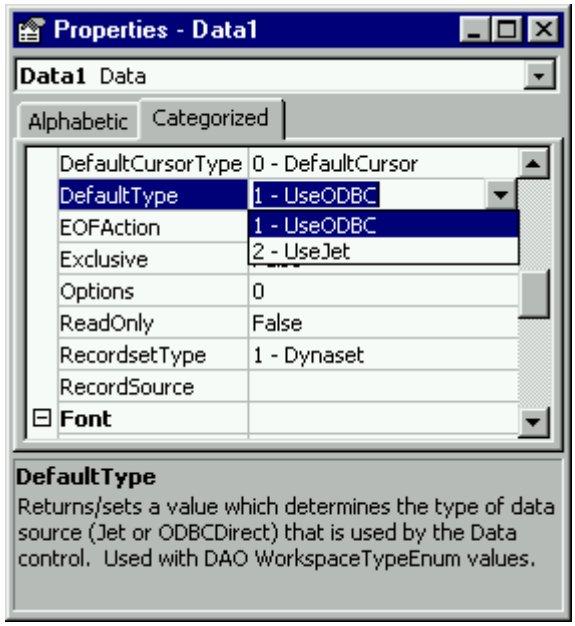
The Quest for Data

While .mdb databases (native to the Access database system) are easy, there are times when programmers need to get data from other desktop sources. For example, we might need to read or write data from dBASE, Paradox, FoxPro, or other databases. We might also need to retrieve information from Excel or Lotus spreadsheets, or even text files. If you take a look at the intrinsic (built-in) data control in VB 6.0, you will notice that there are several additional data sources that the control can talk to in addition to its native Access.



The real benefit of connecting to one of the external **ISAM** (Indexed Sequential Access Method) file types listed in the connect property of the data control is that we can work on the data - as is - without changing its structure. We can leave the data where it is and use VB6.0 to connect with the various data source types. So any applications that created these files can continue to operate unchanged. We just go in and read or write data to and from these sources.

Microsoft's standard for providing data access to various data sources is **Open Database Connectivity** (ODBC). Essentially, this is a SQL approach to retrieving data. ODBC is supported by all sorts of software applications from spreadsheets to word processors to databases. ODBC provides **database interoperability**, which really means that it gives us methods by which data can be exchanged among different databases.



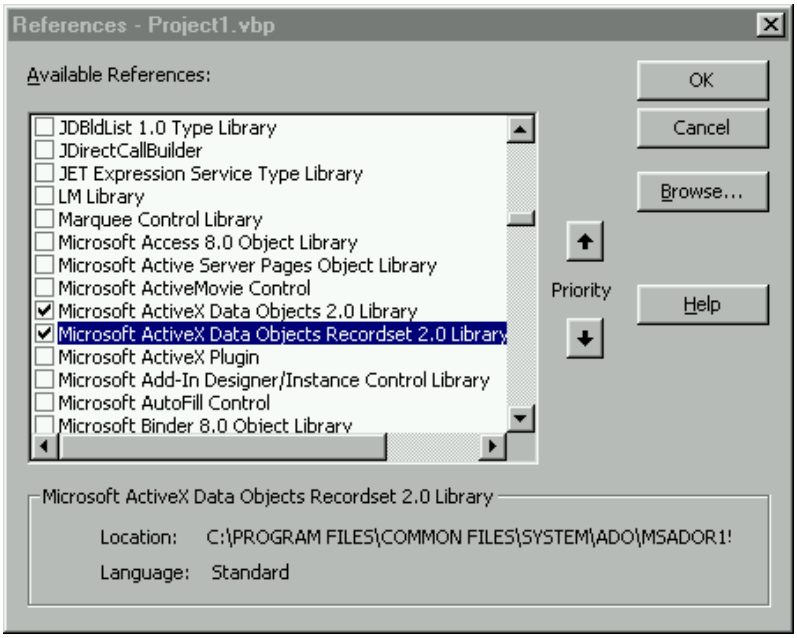
Using ODBC Direct allows you to deploy client/server applications using Microsoft Access without using Microsoft Jet as the middle layer. ODBC Direct is an interface directly to ODBC. So in most cases, it is faster.

If your application is hitting a Microsoft Jet .mdb or any other file-share databases it supports, you should use the Microsoft Jet path. This is because ODBC Direct was created specifically to access *remote* data. You should also use Jet if you want to join data in tables stored in different back-ends such as Oracle and SQL Server. You would need Jet in this case because it provides heterogeneous joins. You can create tables using ODBC Direct by executing SQL statements, but it's more convenient to use the Jet *TableDef* object.

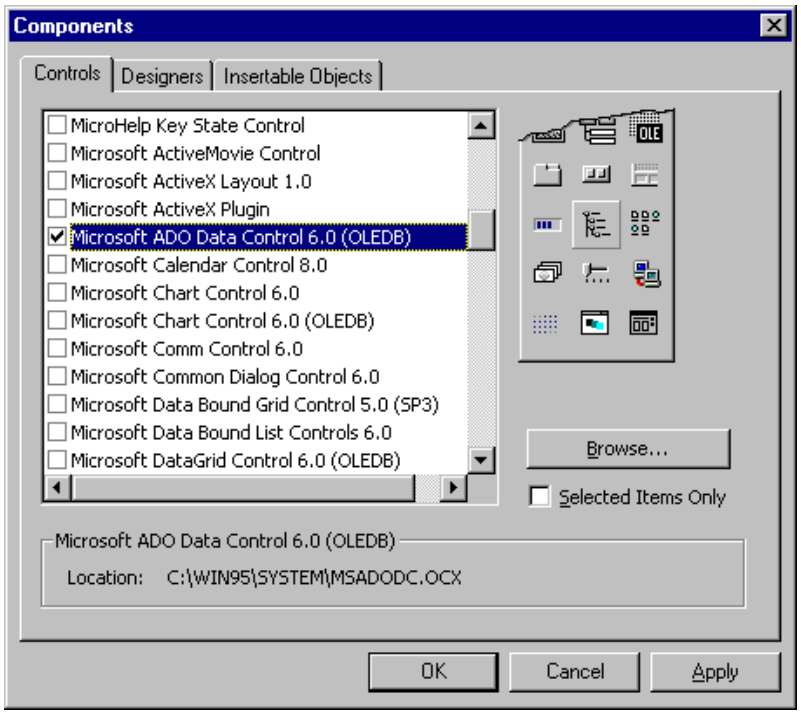
Active X Data Objects Data Control (ADODC)

With the ADO data control, we can be connected to more databases of different types. Depend upon the code it can be accessed from the same interface.

1. Start a new project. Now go into the Project | References dialog and add the Microsoft ActiveX Data Objects 2.0 Library and ActiveX Data Objects Record set 2.0 Library references to your project. Now VB 6.0 knows about the ADO components we want to use.

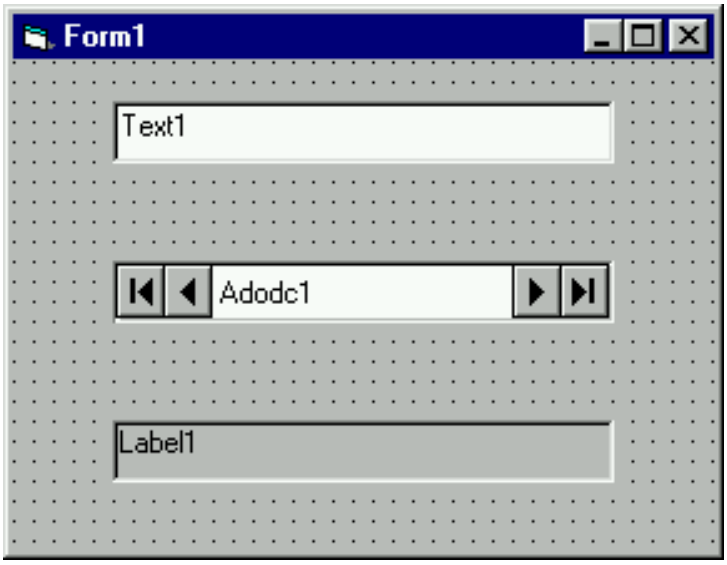


Then right click on your tool palette and select Components. Select the Microsoft ADO Data Control 6.0 (OLEDB):



Click OK. This will add an ADO data control to your palette.

2. Name the default form in the project frmADO. Draw an ADO Data Control (ADODC) on the form. Next, draw a textbox and label as shown on the form as well. We are going to create a simple bound text box program like our first data control program. And we will use the label to show where we are in the record set.

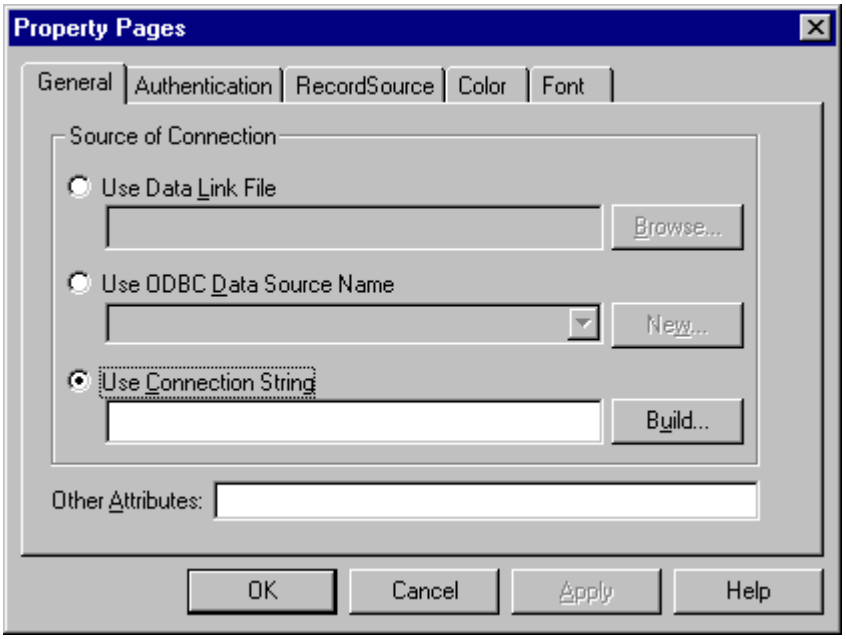


In order to hook up the ADODC to ourBiblio.mdb database, we must first set some properties.

Right click on the ADODC and select ADODC Properties. This will bring up a Property Page dialog box for the control. The first thing we must do is tell the control some important information. Unlike the singular *Database Name* property we need to set on the standard data control, the ADO data control requires a *connection string*. The connection string consists of the specific OLE DB provider to use, as well as the data source we want to access. The connection string is the critical piece of information the ADODC control needs to find the data source.

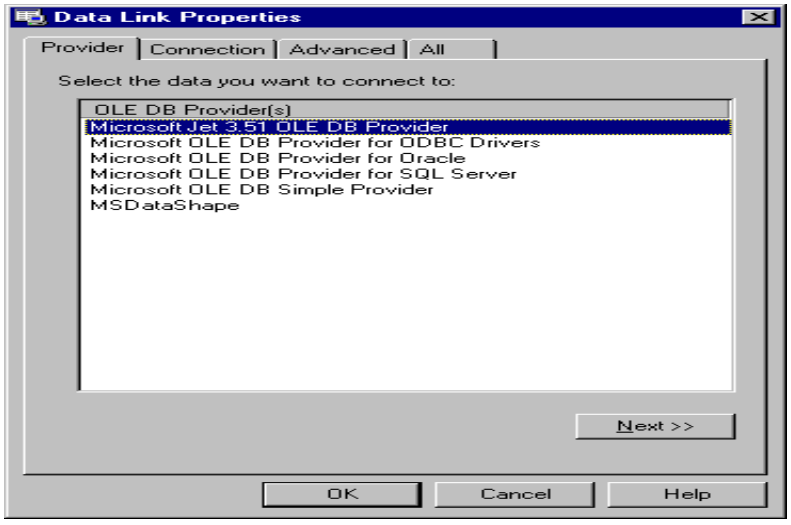
Steps to Set Up the ADODC Connection String

Since setting up the Connection String must contain just about every piece of information required connecting to our data source, this comes in very handy indeed! Recall that the connection string needs to know things like the location and name of the database, any passwords that might be required, and the OLE DB data provider.



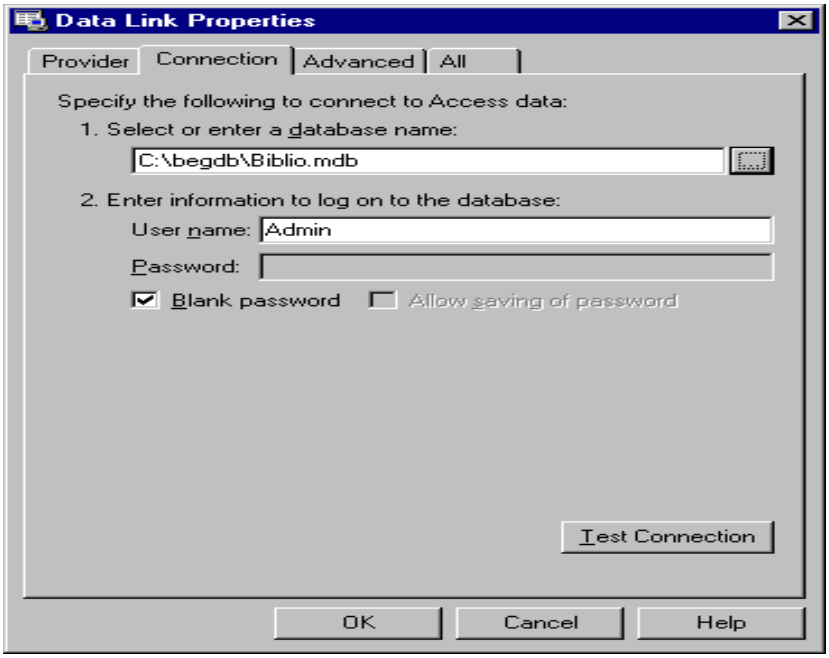
Click the Build button and let's step through the process.

We are presented with another set of property pages for the Data Link. Notice the list of OLE DB Providers that are shipped with VB6.0. If we wanted to connect to a generic ODBC source, we have a provider for ODBC Drivers. Notice that we have providers for Oracle and SQL Server. And as time goes on, all of the major database providers will ship their own OLE DB providers.

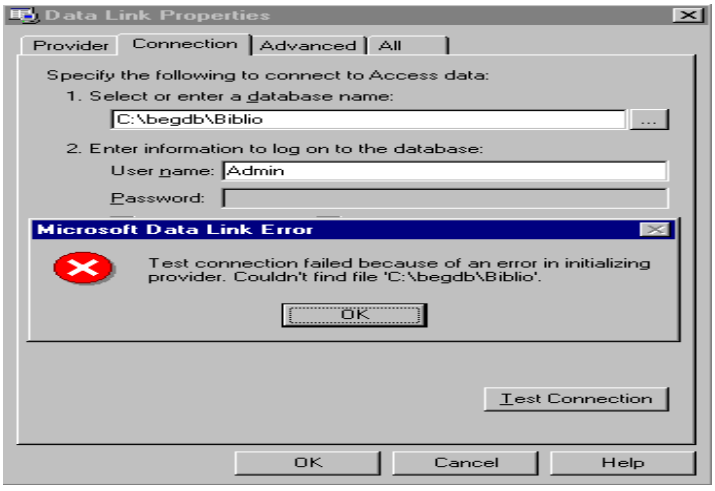


Select the Microsoft Jet 3.51 OLE DB Provider.

Click the *Next* >> button. This brings up the Connection tab. Here is where we must tell VB the location and name of the database we will be using. Click the button with the ellipsis and locate the usual\BegDB\Biblio.mdb database. Since the database does not require a password, don't change the entries for logging on to the database.



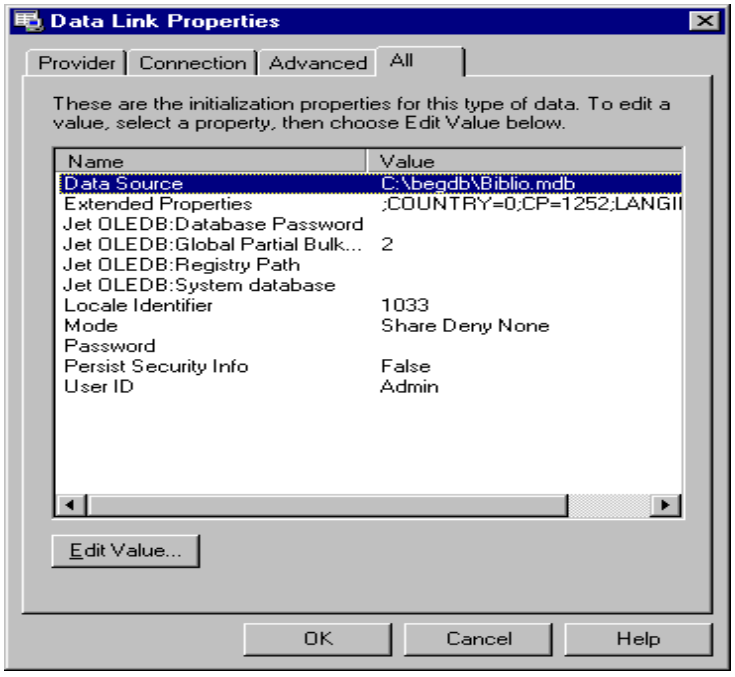
It is always a good idea to use the Test Connection option. This way, if there was something wrong with the location or name of the database, we would get an error advising us of this. Let's say that you entered the name of the database but forgot to add the .mdb extension. By testing the Data Link, we would know immediately:



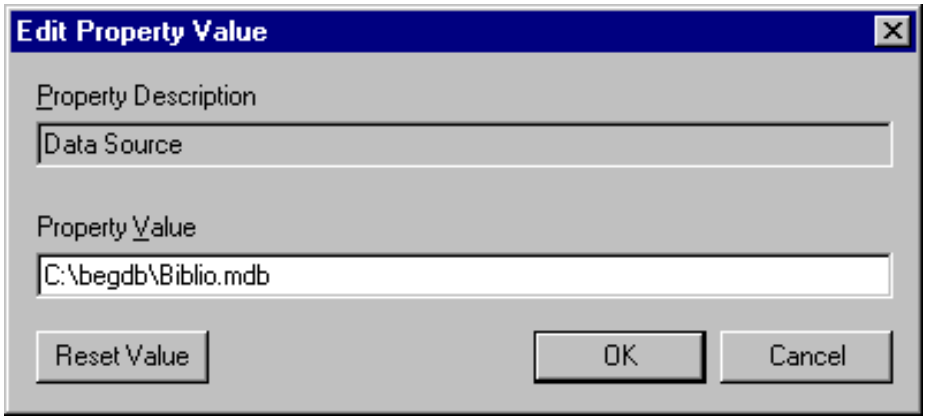
We can then correct the error in the name and location of the database and press Test Connection once again.

There, that's better. Now click the *Advanced* tab just to see what options are available to us. Leave the default *Share Deny None* (if you needed to open the database in a read- only, exclusive mode you would check the Read box)

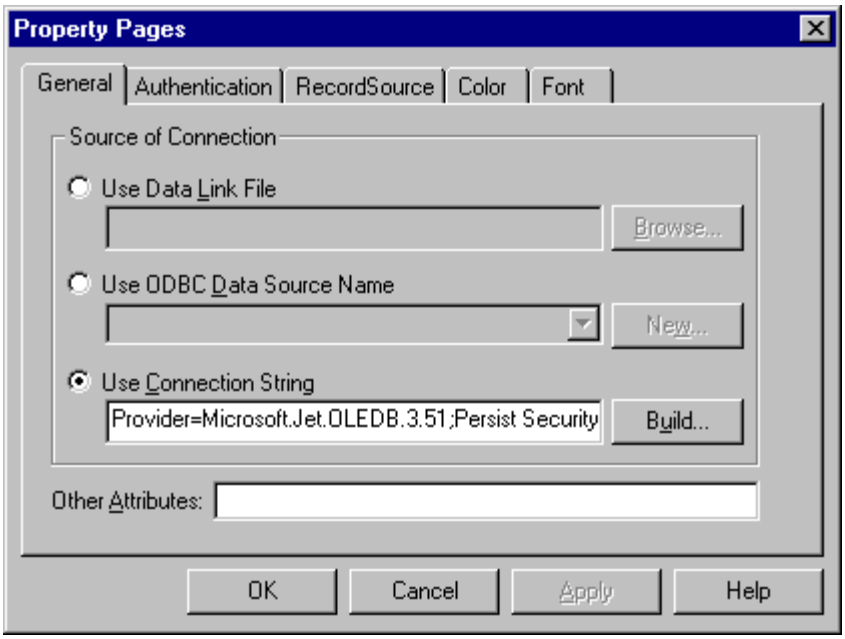
Now click the *All* tab. Here you can see all of the information the Data Link property box garnered for us:



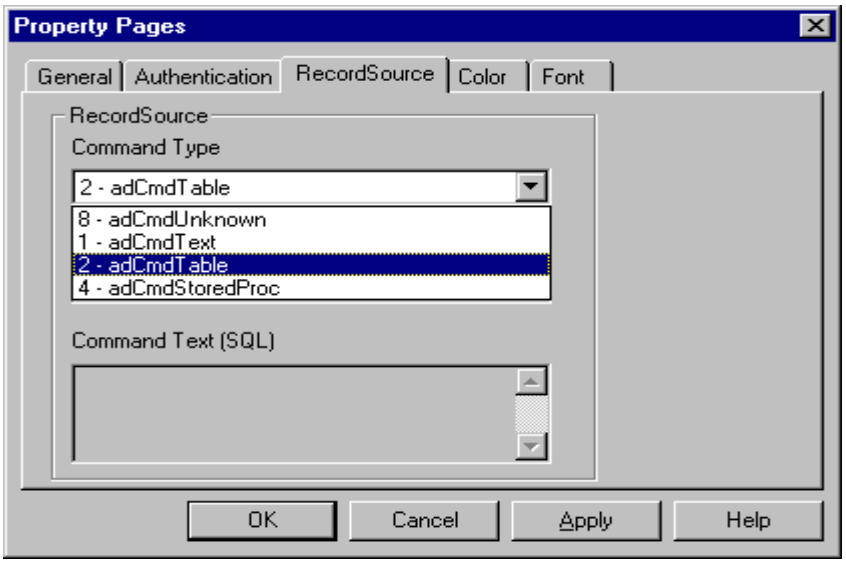
This is all of the information that will be used to create the connection string. If you need to modify any of the properties, simply click the *Edit Value*…button. Highlight any value you wish to edit and press the Edit Value button. This will give you a chance to modify any value in the connect string prior to clicking the *OK* button.



After the connection string is built, click OK to dismiss the property pages for the Data Link. Now the control has the information it needs to connect to the data source. However, we still need to inform the data control which table(s) we wish to access. Right click on the ADODC data control again and select ADODC Properties. Notice that the connection string text box is now filled in:



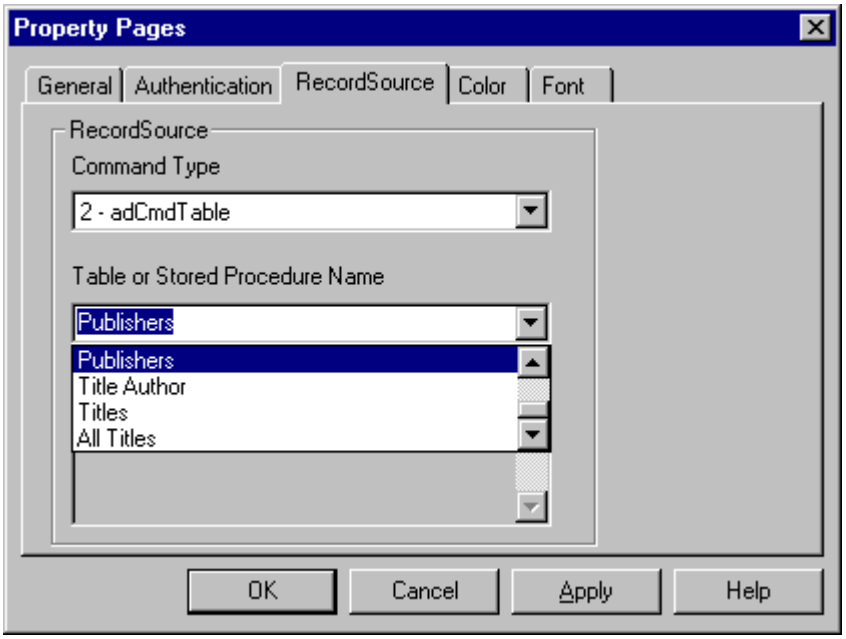
Click on the *Record Source* tab and click the drop down list box for the *Command Type*:



Select *2 – adCmdTable*. Now the control knows we want to access records from a table directly. If the *DataSource* is not known in advance, then *adCmdUnknown* is selected. If we were going to issue a SQL command, the *adCmdText* would be selected, and the bottom text box, Command Text (SQL), would become enabled. Finally, if we have stored, pre-compiled procedures, we would choose *adCmdStoredProc*. This time, be sure to select choice2 - adCmdTable.

Now the control knows that we want to access records from a table, and it knows the name of database from when we set up the Data Link. Now, the *Table or Stored*

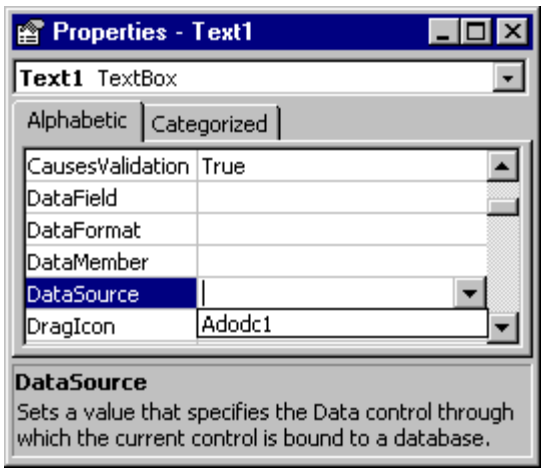
Procedure Name list box becomes enabled. Click the list box and all of the tables in the database are shown:



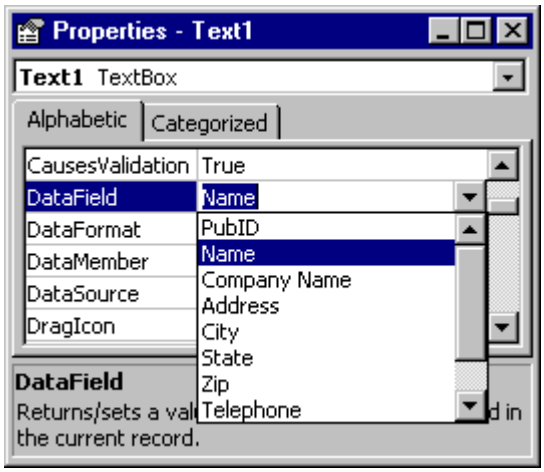
Select the *Publishers* table and click *OK*.

The data control now has the connection string built, and will be able to retrieve a recordset for us from the data source. Double-click on the data control to bring up the code window. You might notice that the *Adodc1 data control* has a few new event procedures.

3. Now that the ADODC data control has been set up, let's bind the *Text1 textbox*. Bring up the property dialog box forText1. Set *the DataSource property* to *Adodc1*.



Now click the drop down box for the *DataField*. Notice that just like the DAO data control; all of the valid fields are displayed:



Select the *Name field*.

Go ahead and *run the program*. You can see that it works as follows

To the user, there is absolutely no difference between the intrinsic data control and our new ADO data control.

CHAPTER 10: REPORT

Data report

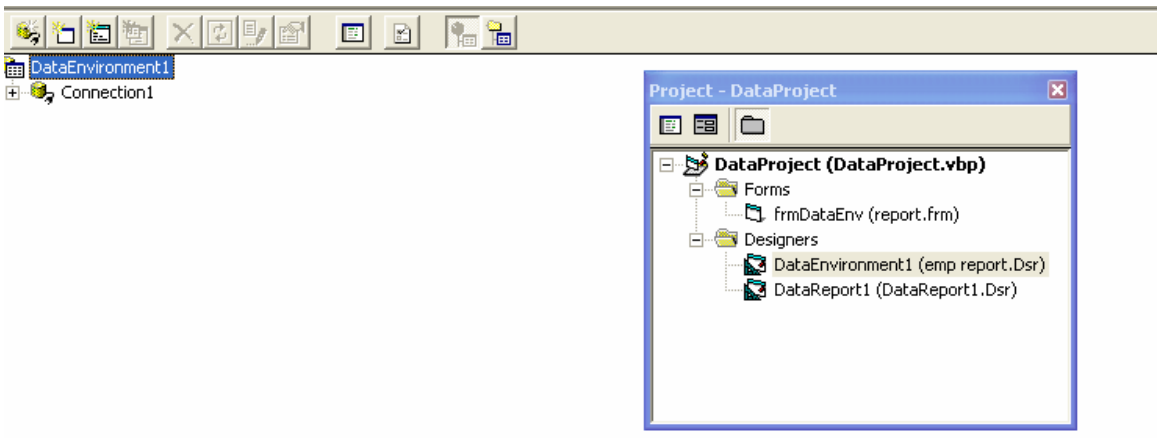
An equally important operation in database application development is that of generating reports. Generating hierarchical multi page reports with running totals and printing them properly has been a headache for developers. For getting a data report, *close* your current *Project*.

To establish a connection follow these steps

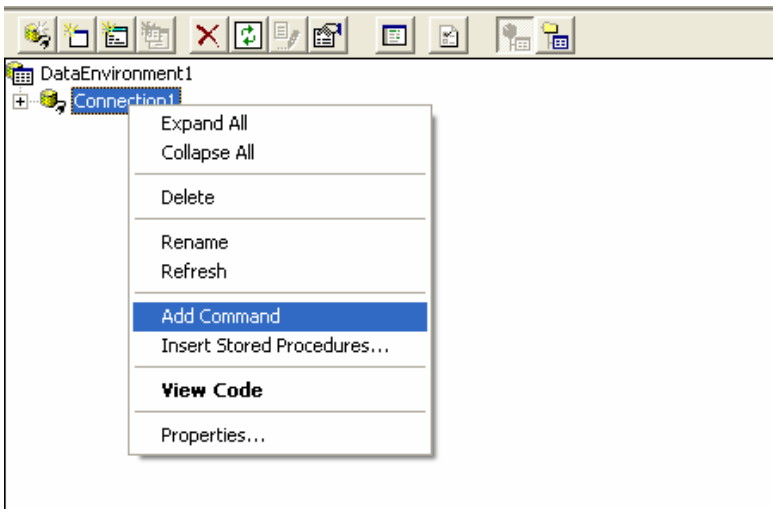
- From the *file* menu click *New*

- Select *data project.*

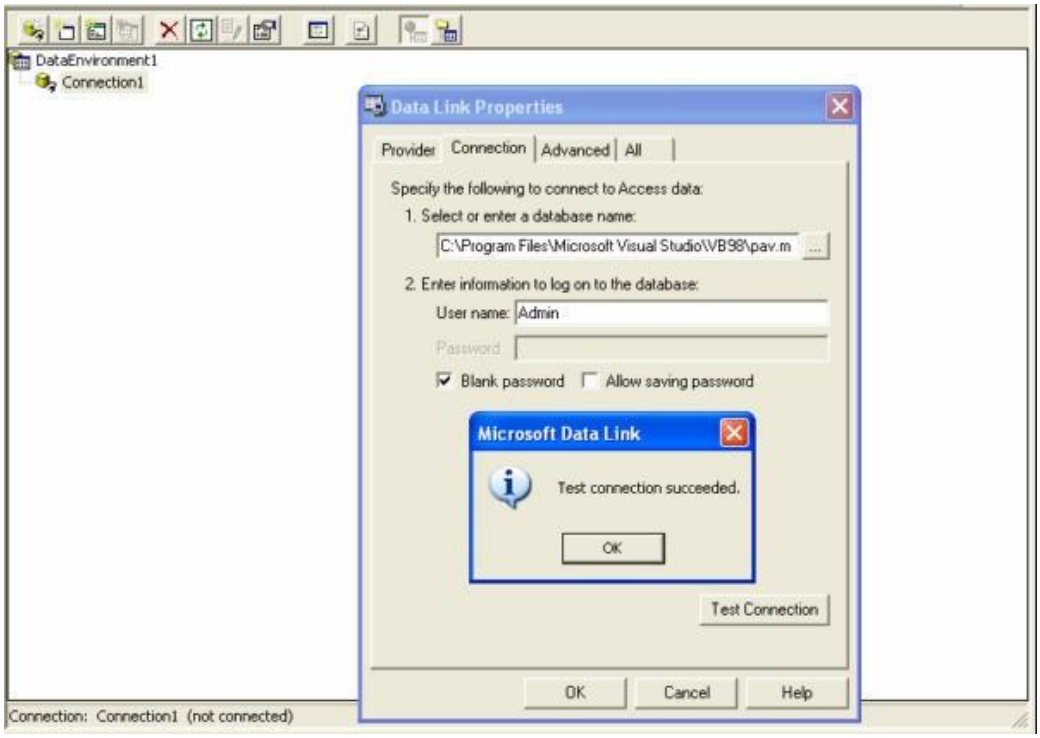
- Double click on the *data environment1* object in the *project explorer* to open the *data environment* window.



- Right click on the *data environment1* and from the shortcut menu, select *add command.*



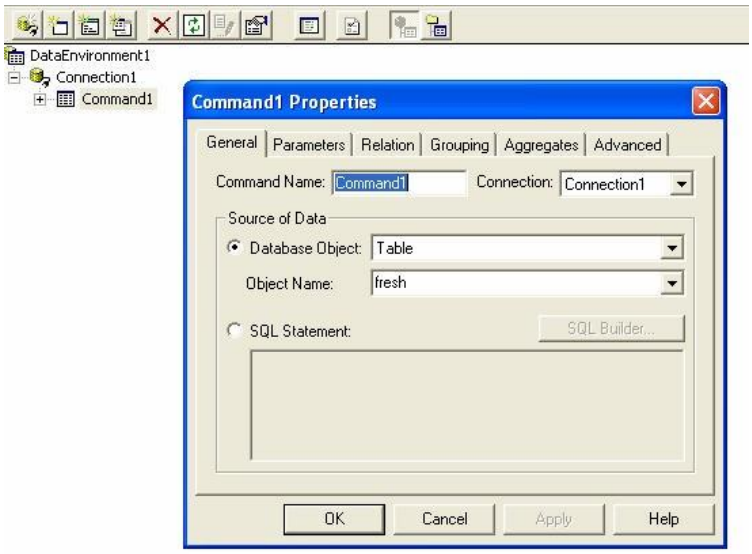
- *Right click on the connection1 object, and from the short cut menu, select properties to open the data link properties window. Specify the driver in the provider.*
- *Specify the database to which you want to connect, in the connections tab.*
- *Click the test connection button to make sure the connection works.*



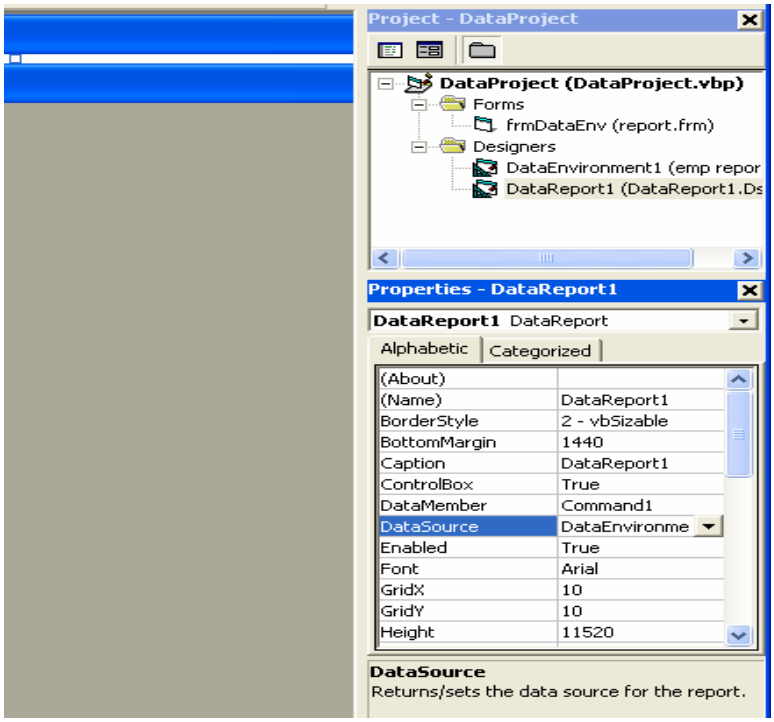
- Click *ok* to return to the *data environment* window.

To retrieve all the records from the database, follow these steps

- Right click on the *command1* object, select *properties* to open *command1 properties* window.
- In the *general* tab, set the *command name* to *command1* and *connection* to *connection1*
- In the *database* object, select *table* and in the *object name* box, select *name of the table* you want.

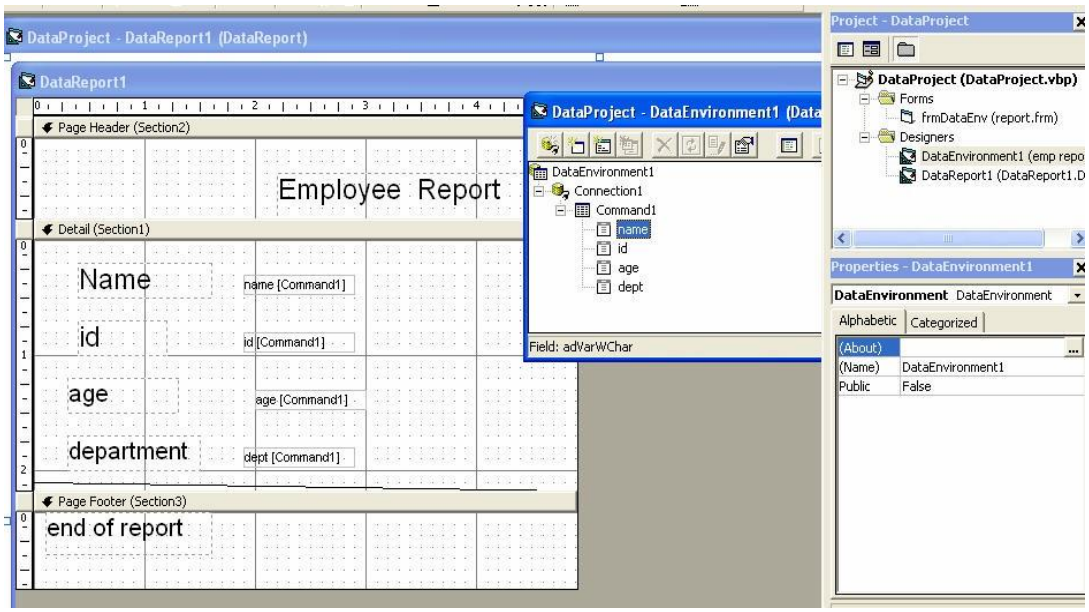


- Click *ok* to return to *data environment1* window. To prepare report follow these steps
- Double click on the *report designer*
- In the *properties window*, select *data source* as *data environment1* and *data member* as *command1.*

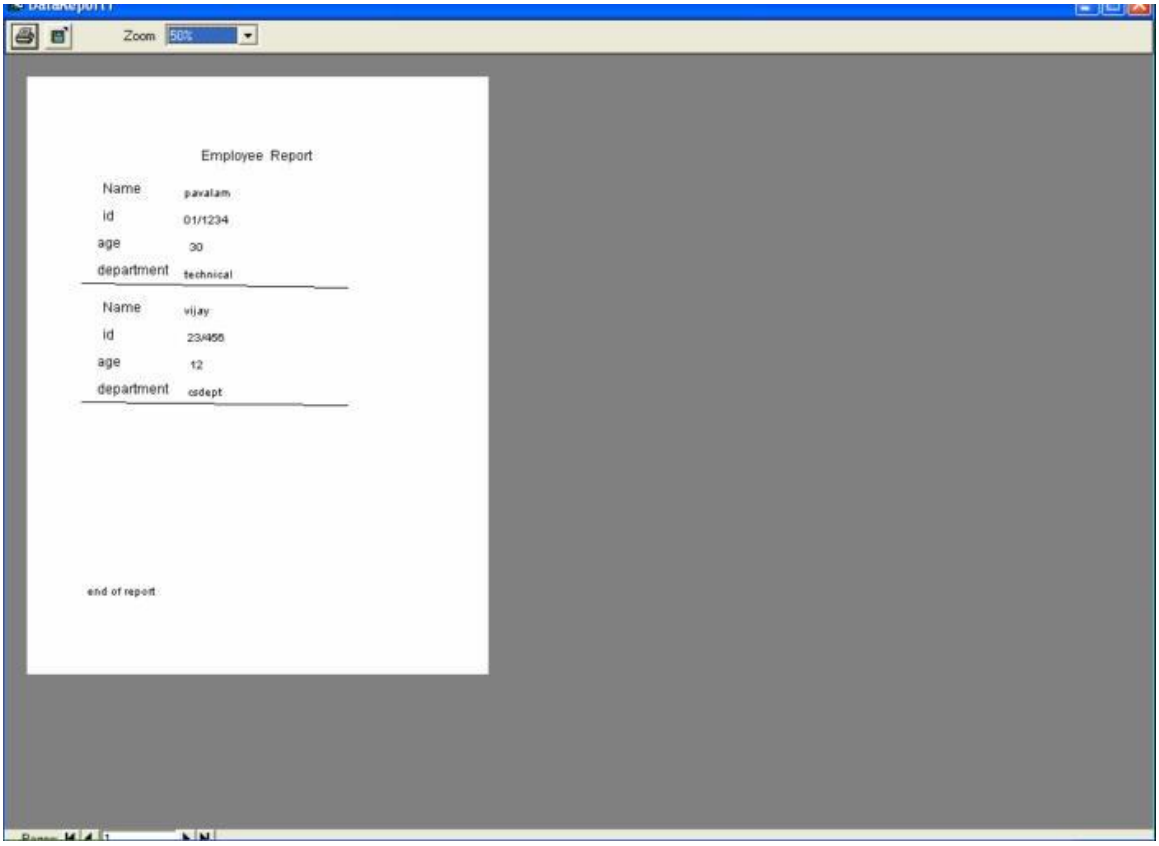

Project - DataProject
DataProject (DataProject.vbp)
Forms
frmDataEnv (report.frm)
Designers
DataEnvironment1 (emp repor
DataReport1 (DataReport1.Ds
Properties - DataReport1
DataReport1 DataReport
Alphabetic
Categorized
(About)
(Name) DataReport1
BorderStyle 2 - vbSizable
BottomMargin 1440
Caption DataReport1
ControlBox True
DataMember Command1
DataSource DataEnvironme
Enabled True
Font Arial
GridX 10
GridY 10
Height 11520
DataSource
Returns/sets the data source for the report.

- The template which is shown below contains a report, which is subdivided into *header*, *section*, *footer.*

- *Header* – what are the headings of the report you want put it there.
- *Section* - contains information's which are retrieved from the database.
- *Footer* – contains the information for the end of the report.
- In the *page header* right click, from the short cut menu select *insert control* and in that click *label control*. Write the name of the report.
- In the *detail* right click, from the short cut menu select *insert control* and in that click *label control*. Write the field name
- Go to the *data environment* window resize it and bring the fields from data environment window to report's detail section. Likewise you continue until all fields are finished.



- In the *page footer* right click *insert control* and from the sub menu select *label control*. Write its *caption* as end of report.
- Run the *data project*, the output will be in format as follows


Zoom
Employee Report
Name pavalam
id 01/1234
age 30
department technical
Name vijay
id 23/456
age 12
department csdept
end of report