Programming in VB
Unit 2

1. Define event? Give two examples of event in visual basic?

- An event is an action initiated by the user.
- When the user clicks on command button or mouse moving/dragging & drop or pressing a key on the keyboard.. This action is called an Event.
- Eg: Mouse up, Mouse Down , Click , Key press etc.

- When an event access, VB searches for the basic code connected to that event.
- The basic code responding to a specific event is kwon as Event Procedure.

- The programming code responding to these events is known as Event Driven Programming

Event Procedures

- Identify the object in the interface which responds
- Open the code window
- Identify the event to which VB must respond
- Write BASIC CODE to process the event

Objects of a user interface which can respond to events are:

FORMS

OBJECTS (button, option Box, Textbox, checkbox.. Etc)

Pull – Down Menus

Types of Events are:

1)Key Board Event: occur when the user presses a key such as Tab, enter etc.

1) Key Down: occurs when a key is pressed

2) Key up: when the user releases the key

3) Key press: to find out which key is pressed.
it occurs when the user press and releases the key
i.e enter, backspace, characters etc

4)Change: occurs when the contents of textbox, listbox, combobox
picturebox changes

2) Mouse Event: occurs when the user takes action using a mouse.

1. Click: when the left mouse button is clicked on the object
2. Dblclick: double click the left mouse button on the object
3. Mouse Down: when a mouse button is pressed
4. Mouse Up: occurs when the user releases the mouse button
5. Mouse Move: continuously as the mouse is moved over the object.

3) **Focus Event**: when data entry via keyboard the focus event can be used to move the focus from one textbox to another

1) Got Focus: It can be used to validate the data entered by the user.

2) Lost Focus: when focus moves from one object to the another, the first object which had the focus receives the lost Focus,

3) Set Focus: Focus can also be moved to another object by using the set focus method in the code.

4) Tab order: Tan key can be used to move focus from one object to another

2) What is Variables? How to declare variables in VB

- Variables are used to store values during execution of a program.
- They take different values at different time.
- They can access the value of the variable by using there variable name.

RULES:

1)Variable names must begin with a letter, digits or both

2)No special characters are allowed except underscore (_) eg: tot_marks

3)VB is not case sensitive

4)Appropriate variable name must be chosen to indicate their role in the program

5)Max length should not exceed 255 characters.

Declaring variables

1) Explicit Declaration

- A variable can be declared using the **DIM** statement

- Eg: Dim Num1 As Integer
  Dim welcome As strings

Num1 = 25
Welcom = " welcome to NHCK"

- Multiple variable can be declared as:

Dim num1 As Integer,Num2 As Integer

2) Implicit Declarations

- A variable are used without declaring them, it called as Implicit declaration

- When VB encounters an undeclared variable, It immediately creates a new variable type **Variant**

- **Variant :** can store all type of data values & its extremely flexiable

Eg: Dim myval

- **Myval can be variant**

3) Explain the data types used in VB 6.0?

1) Numeric Data type:

Data Type	Description	Size
Integer	Integer Value	2 bytes
Long	Long integer	4 bytes
Single	Single precision	4 bytes
Double	Double precision floating point	8 bytes

2) Byte Data Type : None of the numeric data type cant store in a single bytes

Byte datatype stores an integer range of 0 to 255

Eg: Dim n1 As byte

n1 = 25

3) String Variable:- Can store text/string of characters with double quotes “ ”

Eg: “new_Horizon”

Dim stdname As String

Stdname = “shalini”

Stdname = “”

*Fixed length strings:- Dim stdname As string *50

If string more than 50 characters , the extra characters is truncated

When the string has fewer than 50, the extra space is padded with spaces

4) Date Variables: VB stores data and time values internally as double prescision number.

integer part as Date

fractional part as Time

Eg: Dim paydate As Date

Paydate = “2/11/2019” (dd/mm/yyyy)

Paydate = “12:15:00 PM”

Paydate = “02/1/2019 12 :15:15 PM”

Formate of date is (dd/mm/yyyy) is automatically changes to MM/DD/YYYY

When data is enclosed with #sign

Paydate = #02/1/2019 12 :15:15 PM# (MM/DD/YYYY)

5) Boolean Variables: store only true / false values

1 for true , 0 for false

occupy 2 bytes

Eg: Dim found As Boolean

Found = true

Its combined with logical operators AND, OR, NOT XOR

They are used in testing conditions

6) Object variable: are stored in 4 bytes

cab access the actual object

eg: Dim b1 As CommandButton, b2 As CommandButton

or

Dim b1 As object, b2 As object

set b1 = command1

set b2 = command2

there properties cab be manipulated as:

b1.caption = “hello”

b2.caption = “world”

b1.fontbold = true

7) Varient variables:

- When VB encounters an undeclared variable, It immediately creates a new variable type **Variant**
- **Variant :** can store all type of data values & its extremely flexiable
- Eg: Dim myval
- **Myval can be variant**

Operators

Operator	Symbol	usage
Exponentiation	^	Dim m , n , st M=10 : n=2 Print m^n Output is 100
Addition	+	Print m+n Output 12
Subtraction	-	Print m – n Output 8
Multiplication	*	Output 20
Division	/	Output 5
Modulus	Mod	Print 10 mod 3 Output is 1
String Concatenation	&	M = “new” n =“ Horizon” St = m & n Print st Output is new horizon

Relational operators

Operator	Symbol	Usage
Equal	=	X = 2
Not equal	< >	X <> 10
Less than	<	x< 10
Greater than	>	Y >5
Less than or equal to	<=	Y<=10
Greater than equal to	>=	Y >= 5

Logical Operators

Operator	Symbol	
Logical NOT	NOT	1 ---- 0 0----- 1
Logical AND	AND	1 1 --- 1 1 0 ----- 0 0 1 ----- 0 0 0 ------ 0
Logical OR	OR	1 1 --- 1 1 0 ----- 1 0 1 ----- 1 0 0 ------ 0

REM statement

- REM is a keyword
- Its used to include remarks explaining the program code
- Any statement with REM at its beginning is ignored by the complier
- Also it can be used at the end of the statement by using :
- Comments in VB using apostrophe (‘) symbol

- Syntax:

  Rem this is a comment statement

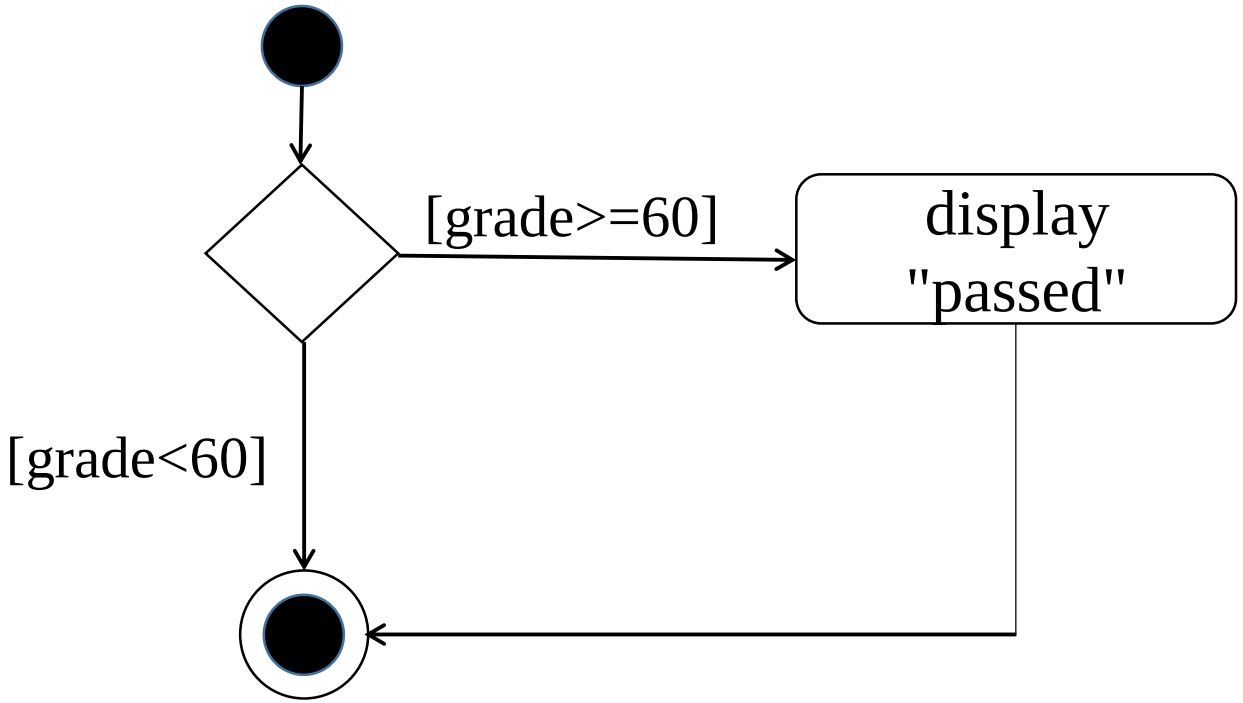
  ‘ comments can be defined here

```
Dim n1, n2 AS Integer ‘ variable n1 and n2
n1 * n2 : product of the number
```

In the Last Class: Selection Structure

- If … Then
- If … Then … Else
- Select … Case

If … Then




```
If grade >= 60
Then
Textbox1 = “A”
End if
sts1
```

If … Then … Else

If condition 1 Then

Sts block – S1

ElseIf conditions 2 Then

Sts block – S2

EndIf

S3

```
Dim n As integer
N = val(textnum)
Select Case n
Case 1
   Lblname = “Sunday”
Case 2, 3, 4, 5
    Lblname = “Working Days”
Case 6,7
    Lblname = “week end”
Case else
    Lblname = “wrong Entry”
End select
End sub
```

Select … Case

```
Select Case expression
Case Val1
S1
Case Val2
S2
…….
Case Valn
sn
Case Else
     Sd
End Select
```

```
Select Case grade
Case 100

Case 75 To 100
Text10.Text = "A"

Case 60 To 74
Text10.Text = "B"

Case 50 To 69
Text10.Text = "C"

 Case 40 To 49
 Text10.Text = "D"

Case Else
       Text10.Text = "F"

End Select
```

```
Private Sub Command1_Click()
Dim s1, s2, s3, s4, s5, tot_mar As Integer
Dim per As Single
Dim grade As String

s1 = Val(Text3.Text)
s2 = Val(Text4.Text)
s3 = Val(Text5.Text)
s4 = Val(Text6.Text)
s5 = Val(Text7.Text)
```

```
tot_mar = s1 + s2 + s3 + s4 + s5
Text8.Text = tot_mar
per = tot_mar / 5
Text9.Text = per
```

```
If per >= 75 Then
Text10.Text = "A"
ElseIf per >= 60 And per < 75 Then
Text10.Text = "B"
ElseIf per >= 50 And per < 60 Then
Text10.Text = "C"
ElseIf per >= 40 And per < 50 Then
Text10.Text = "D"
Else
```

5. Looping Structures / Control Structures

- Loops are required whenever a set of statements must be executed a number of times.
- Control structures can be classified into 2 categories:

1. Entry – Controlled Loops
2. Exit – Controlled Loops

Entry Controlled Loops checks the control condition before entering the loop. The body of the loop is executed only when the condition is true.

Otherwise, the control is transferred to next statement outside the loop.

The loop is executed repeatedly until the control condition becomes false.

Exit – Controlled Loops test the control condition at the end of the body of the loop. The body of the loop is executed at least once, since the condition is tested only at the end of the loop.

The looping statement will be executed repeatedly until the control condition is false.

Entry Controlled	Exit - Controlled
Do While Condition Statements Loop	Do Statements Loop While Condition
Do Until Condition Statement Loop	Do Statement Loop Until Condition
While Condition Statement Wend	
For Index = Start To End Statements Next	
For Each element in group Statement Next	

Do While Condition
    Statements
Loop

- The condition is checked first
- If true the sts are executed and control is transferred back to the Do-while sts.
- When the condition is False, sts in the body are skipped and control is transferred to the next sts outside the loop
- Its entry controlled loop

```
Dim M As Integer, N As Integer
M=12 : N=16

Do while M <> N
  if M > N then
  M = M – N
Else
   N = N – M
End If
Loop
```

Do
    Statements
Loop While Condition

It is exit controlled loop.

The sts are executed first and only then the condition is tested

Therefore the statements are executed at least once.

The loop is executed as long as the condition is fasle.

```
Dim M As Integer, N As Integer
M=12 : N=16

Do
  if M > N then
  M = M – N
Else
   N = N – M
End If
Loop while M <> N
```

```
Do until Condition
    Statement
Loop
```

- Its entry Controlled loop
- Which executes the statement as long as the condition is False.
- If true the sts are executed and control is transferred back to the Do-while sts.
- When the condition is False, sts in the body are skipped and control is transferred to the next sts outside the loop

```
Dim M As Integer, N As Integer
M=12 : N=16

Do until M <> N
  if M > N then
  M = M – N
Else
   N = N – M
End If
Loop
```

Do
Statement
Loop until Condition

- Its Exit controlled structure, where the condition is tested at the end of the loop.
- Therefore the statement are executed at least once.
- Further the loop is executed as long as the condition is False.

```
Dim M As Integer, N As Integer
M=12 : N=16

Do
if M > N then
   M = M – N
Else
    N = N – M
End If
Loop until M <> N
```

While Condition
Statement
Wend

Its Entry controlled loop

```
Private Sub Command1_Click()
Dim M As Integer, N As Integer, GCD As Integer
M = Val(Text1.Text)
N = Val(Text2.Text)

While M <> N

  If M > N Then
  M = M–N

Else

   N = N–M

End If
Wend
 GCD = N
 Text3.Text = GCD

End Sub
```

```
For Index = Start To end
statement
Next
```

For – next is an entry controlled loop which executes the statements for a specified number of times.

* sets the index variable to start

*check whether index is greater then end. If true it exits the loop

* otherwise executes the sts in the body of the loop

* Increments the index value by the step value by 1 (default)

Eg :

```
Dim I As Integer
For I = 1 to 10
print 2^I
Next
```

Output: 2, 4, 8, 16,…….

For Each element In group
sts
Next

- For Each – Next loop is similar to a for-next loop.
- It executes the set of statements for each elements of a collection of items.

```
Private Sub Command4_Click()
Dim ctl As Control
Dim cmdkout, Lblkount As Integer
Dim txtout As Integer
```

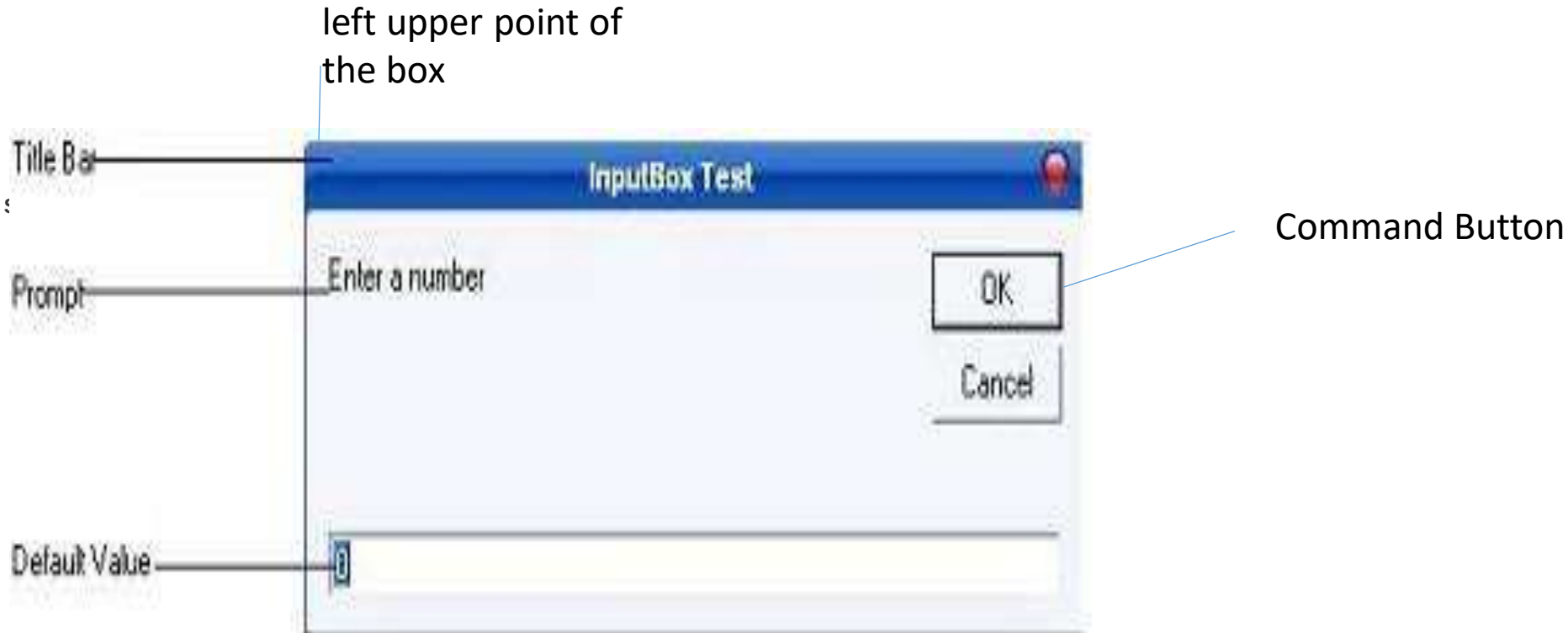
```
For Each ctl In Me
If TypeOf ctl Is TextBox Then
txtout = txtout + 1
ElseIf TypeOf ctl Is Label Then
Lblkount = Lblkount + 1
ElseIf TypeOf ctl Is CommandButton Then
cmdkout = cmdkout + 1
End If
Next
```

```
 MsgBox "number of textbox = " & txtout & vbCrLf & "Number of label boxes =" & Lblkount & vbCrLf & "Number of command =" & cmdkout & vbCrLf



End Sub
```

6. Explain InputBox() and MsgBox() functions in Visual Basic

INPUT BOX

- Displays a prompt in a dialog box
- Waits for the user to input text or click a button
- and returns a String containing the contents of the text box.



The general Form of InputBox Function

Variable Name = InputBox(prompt [,Title] [,Default] [x pos] [,y pos])

Prompt : It’s a string displaying a message in a label in the dialog box

Title : Optional, it’s a string displayed in the title bar of the dialog box

Default : Optional, it’s the default value of input, in case the user does not give any input. It will be a empty textbox

X pos and Y pos

Both optional. Numeric expressions that specify custom positioning of the box on screen (by default, the box is displayed in the center of the screen, which is usually desired). The left upper point of the box is the (X, Y position) selected by the user

MSGBOX

- Displays a message in a dialog box and wait for the user to click a button
- And returns an integer indicating which button the user clicked.

Syntax

- **MsgBox (** Prompt [,icons+buttons] [,title] **)**

- **Prompt :** String expressions displayed as the message in the dialog box.
- **Icons + Buttons :** Numeric expression that is the sum of values specifying the number and type of buttons and icon to display.
- **Title :** String expression displayed in the title bar of the dialog box. If you omit title, the application name is placed in the title bar.

Eg:

```
MsgBox "Hello World", vbOKOnly , "hi"
```

Button

Constant	Value	Description
vbOkOnly	0	Display OK button only
vbOkCancel	1	Display OK and Cancel buttons
vbAbortRetryIgnore	2	Display Abort, Retry and Ignore buttons
vbYesNoCancel	3	Display Yes, No and Cancel buttons
vbYesNo	4	Display Yes and No buttons
vbRetryCancel	5	Display Retr

ICON

Constant	Value	Description
vbCritical	16	Display Critical message icon
vbQuestion	32	Display Warning Query icon
vbExclamation	48	Display Warning message icon
vbInformation	64	Display information icon

Functions

- A function is to return a specific item or information to the calling function.
- General Form:

Function functionName(arg1 As type, arg2 As type,…..)As type

Statements

End Function

- FunctionName ---- is the name of the function
- Arg1 As type --- defines the arguments with their datatypes
- As type at the end of the sts --- defines the type of value that the function will return.

- A function procedure can be created using Add Procedure dialogue

- Go to Tools →Add procedure→ select type Function → select Scope → name the function

Eg:

```
Dim sqft


Private Function hi(x1 As Integer, x2 As Integer) As Single
 hi = x1 * x2
End Function


Private Sub Command5_Click()
Sqft = hi(3, 3)
Label1 = Sqft
End Sub
```

General Procedures

- A large program is divided into smaller program called Procedures.
- Procedures are written to define tasks that may have to be used at many points in a program.

Advantages:

1) It makes program development easier.
2) Software reusability
3) Avoids repeating the same code in a program
4) Easy to debug a program with procedure
5) Using a procedure is a good programming practice

- General procedures can be divided into:
  1) sub procedures
  2) function Procedures

To create a general Procedure in a form file the steps are:

Go to Tools →Add procedure→ select type Function → select Scope → name the function

Eg: Dim N1 As long, N2 As long, N3 As long

Private sub Big (x As Long, y As Long, z As Long)

End sub

Call Big(N1, N2, N3) ‘ Calls the procedure Big passing the value of N1, N2, N3 as argument

General procedure can be invoked in one of the three ways:

ProcedureName (Argument list)
ProcedureName Argument list
Call ProcedureName (Argument list)

General Procedure can be created and saved in two types of files:

* frm (form) files
* Bas (Module) files

- When its saved as Bas file, it can be used by any event or general procedure .
- To create a general procedure in a new Bas file the steps are:

1) select project → add Module.

2) click on the module icon and click on open

3) choose Tools →Add procedure→ select type Function → select Scope → name the function

Forms, Controls and Events

Forms

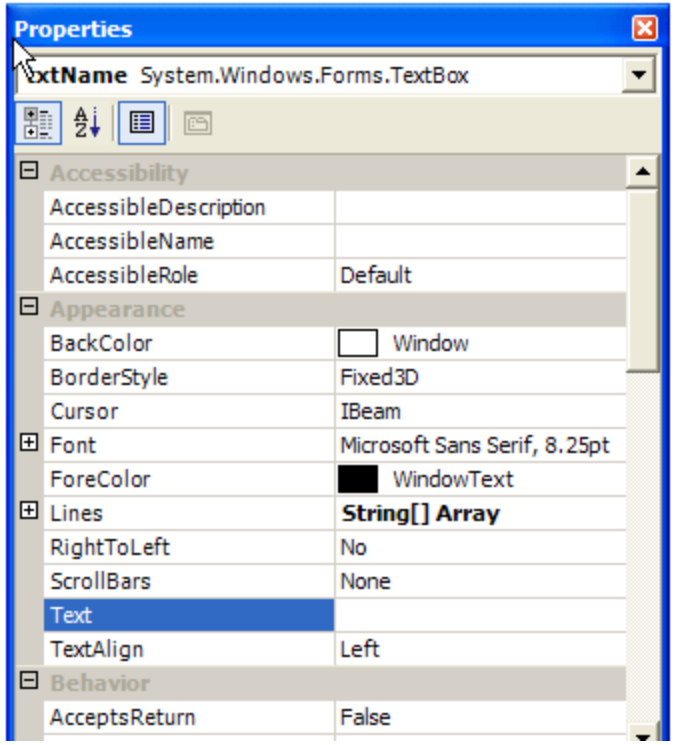
- A form is a container for controls
- A form is used to design a GUI-based window in a Windows application
- A form displays information and receives input from the user.
- Always orient a form at a task as defined by the user

Form Properties

- Properties are the attributes of an object
- Methods are its responses of the object
- A property of an object can be set at design time in the properties window or at run time through coding

Setting Properties

- Design Time → Set in Properties Window
- Run Time → Set / Change in Code



Form Properties

- Name: defined name of an object
- Appearance: the value is 0 – Flat and 1- 3D appearance
- BackColor: sets the background color of the object
- Caption: sets the message display in the title bar
- Enables : Set value to true or false
- Font : sets the font style, size etc.
- Visible: sets the value true or false depending on whether the objects must be visible or hidden
- Width, Height : sets the dimensions of the object

Border style

Different styles are available for a form border.

0 – none	VbBSNone
1- fixedsingle	vbFixedSingle (form has a border, but cannot be resized. Its fixed
2 – sizable	vbsizable consists of a border and title bar
3- fixed Dialog	vbFixedDialog consists of a fixed dialog box

4 – Fixed ToolWindow vbFixedToolWindow. Consists of only a close button and therefore cannot be re-sized

5-Sizable ToolWindow vbSizableToolWindow. Consists of a close button but can also be resized.

Intrinsic Control

- The default controls which are automatically displayed in a Tool Box when a form is loaded are known as Intrinsic Controls.
- Some common Intrinsic Controls are:

1) Label
2) CommandButton
3) TextBox
4) Frame
5)Option Button
6) Check Box
7) Picture Box
8) Image Box
9) Timer
10) Combo Box
11) List Box
12) Horizontal & Vertical Scroll bars
13) Shape and line
14) OLE

Label :

Label control allows the user to display text on a from. Text printed on label cannot be edited during runtime.

Ex: Label1.Caption = “Name”

Label Properties:

Autosize : Can be set to true or false. When true it allows the label to automatically resize itself to display its entire contents.

BackStyle: Can be set to

0 – Transparent or 1 – opaque

backStyle of labels set transparent, so that the text appears to be placed directly on the Form.

Command Button

its frequently used control.

Its used to invoke response from the user.

Command Button Properties:

1) Name – Name of the command Button is used to identify it in the VB code

   no two command Button in the same form can have the same name.

2) Style – it indicates the types of display of the command button.

   0 – standard it's a default value

   1 – Graphical Displays the command button with graphics

3) BackColor : Command button color can be changed by using Backcolor, this will be effected only when the style property is set to graphical.

4) Picture: a picture can be add in the Command button and it will be displayed only when the style property is set to graphical.

5)Enable : Command button can be enable or disabled , true or false . When False the command button is disabled and cant be used until set to true

6)Caption : it places the text entered on the command Button.

7) TabIndex : Tab Index value starting from 0 sets the order in which focus is received by the control on a form. When one of the values of Tab Index is altered the rest of the value are automatically re-ordered.

Most frequently used Event procedure for command Button are:

- Click
- KeyDown
- Keypress
- keyUp
- MouseDown
- MouseMove
- MouseUp
- GotFocus
- LostFocus

TextBox Control

- Its used to enter the text in a windows user interface

Basic Text Properties are:

1) Enable: indicates whether the user can interact with the control or not

2) Locked: indicates whether the user can type in the textbox or not

3) MaxLength: indicates the max number of characters input in the textbox.

4) MultiLine: specifies whether the TextBox will hold a single line or Multiple lines.by default it allows single line of text

5) PasswordChar: specifies the masking character for text display in the textbox. Passwordchar is set to “*”

6)ScrollBars: Indicates either Horizontal Scrollbars or both for the textbox, this property is used with multiline property.

7) Text: specifies the text in the textbox

8) Alignment: Alignment of the text in a textbox can be left-justified, centered, or right-justified

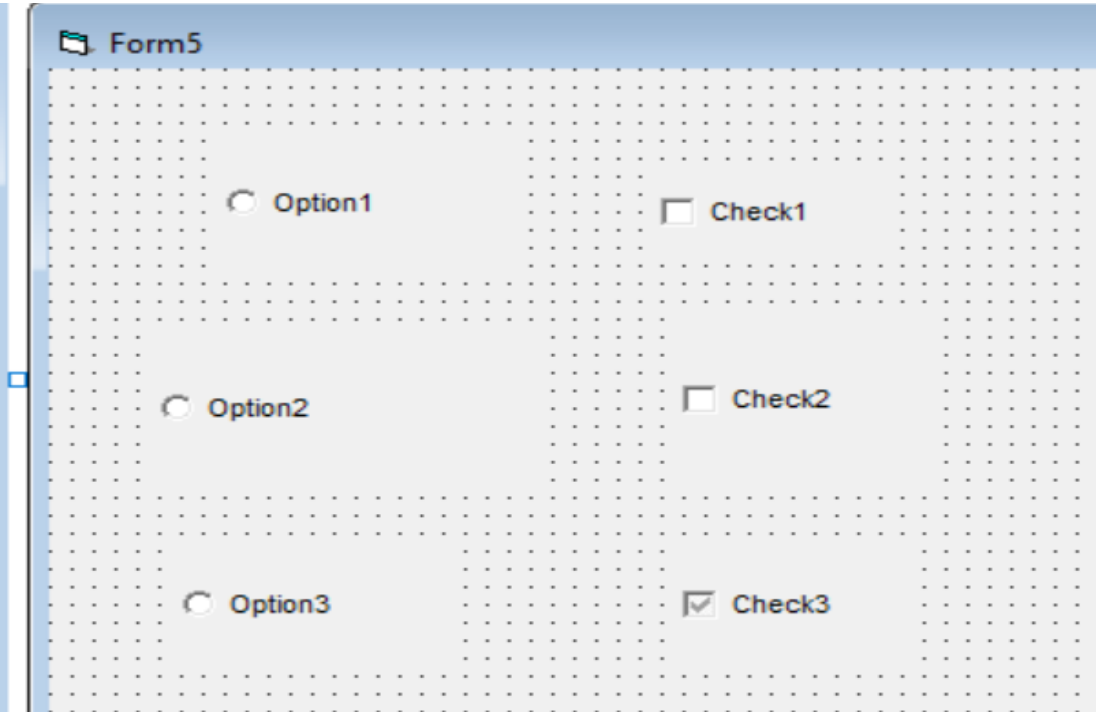
TextBox Events are:

- SetFocus
- Change
- Click
- Gotfocus
- Lostfocus
- Keypress

Option Button

control allows the user to select only one option from a group of options on a form.

If the user wants to select 2 options within a form, the option must be grouped together within a frame control



- Option Button Properties

1) Name: set the name of the option button
2) Caption: determines the purpose of the option button
3) Value: true or false.

set true at design time – pre selection at run time.

CheckBox Control:

A check box control is commonly used to express optional features.
If checked, the feature is used.
If unchecked, the feature is not used

- Multiple optional can be selected

- Checkbox have 3 possible value property:

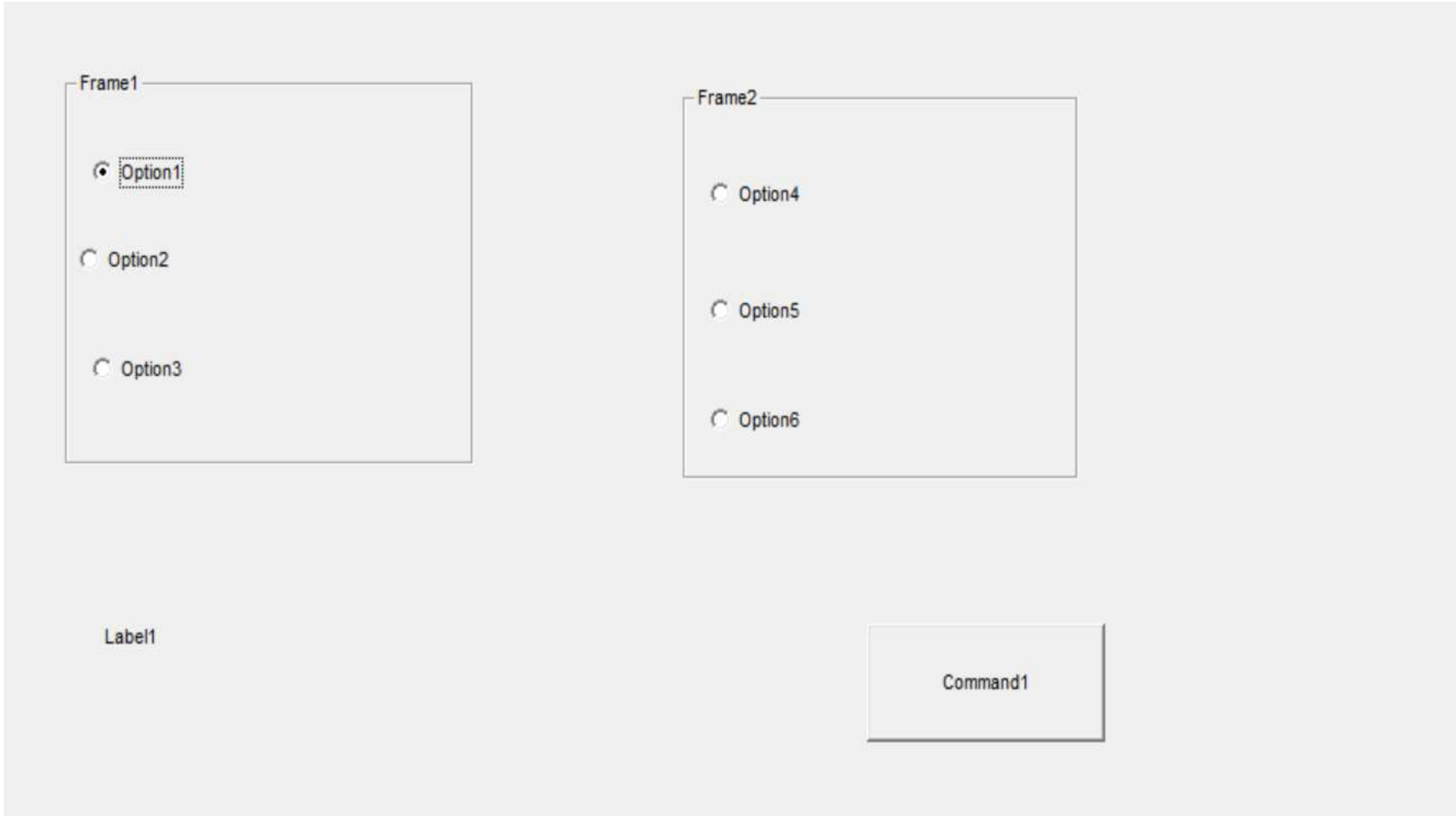
0 – Unchecked

1 – Checked

2 – Grayed

A Grayed checkbox is neither on or off, but the setting can be changed by the user.

Checked Box	Optional Button
A single control is valid	Always in groups; single control is of no use
Value property has 3 settings 0 – Unchecked 1 – Checked 2 – Grayed	Value property has 2 setting 1- on 0 - off
More than one option can be checked	In a group only one option can be selected


Frame1
Option1
Option2
Option3
Frame2
Option4
Option5
Option6
Label1
Command1

```
Private Sub Command1_Click()
If Option1.Value = True Then
Label1.Caption = "apple"

ElseIf Option2.Value = True Then
Label1.Caption = "orange"

ElseIf Option3.Value = True Then
Label1.Caption = "lemon"

End If
If Option4.Value = True Then
Label1.Caption = Label1.Caption + "
and ice cream"

ElseIf Option5.Value = True Then
Label1.Caption = Label1.Caption + "
and sweets"

ElseIf Option6.Value = True Then
Label1.Caption = Label1.Caption + "
and coffee"
End If


End Sub
```

```
Private Sub Command2_Click()
If Check1.Value = vbChecked Then
Label2.Caption = Label2.Caption + "happy"
End If

If Check2.Value = vbChecked Then
Label2.Caption = Label2.Caption + "sad"
End If

If Check3.Value = vbChecked Then
Label2.Caption = Label2.Caption + "happy" + "sad"

End If

If Check1 = 0 And Check2 = 0 And
Check3 = 0 Then
Label2.Caption = Label2.Caption
+ " None is selected"
End If

End Sub
```

VB provides 2 ways of displaying pictures on the screen are:

- Picture Box
- Image Box

Picture box control

- Are used to display graphics pictures during the design time from the property window picture.
- Picture box occupies more space.

Properties are:

- Name : identifies the name of the picture box
- Picture: load the picture during the design time.. Location of the image
- Autosize: when its true, the picturebox is automatically resized to display the entire content else

  when false, only the part of the entire picture will be displayed.

- Picture box uses PaintPicture
- Paint Picture: used to draw the content of a picturebox control into another picturebox

- Syntax :

Picturebox.paintpicture picturename,x1,y1,w1,h1,x2,y2,w2,h2

Image Box Control

- The image box control is similar to picture box control and is used to display picture
- This control takes less memory in comparison to picture box

Properties:

- Name
- Stretch : when set to true the picture is the image box is enlarged or shrunk according to the current coordinates. else its in original size

Type	file extension	description
GIF image	.gif	Graphic Interchange format image type
Bitmap image	.bmp	Device dependent bitmap
Icon image	.ico	icon
Cursor image	.cur	Cursor. A special bit map type used to represent the mouse pointer
Windows meta file	.wmf	Graphical images
JPGE	.jpg or jpeg	Joint photographic experts group

Image Box	Picture Box
Occupies less memory	Occupies more memory
Cannot use paint picture method	Paint picture method can be used
Cannot act as a container for other objects	Can act as a container for other controls
When stretch is set to false, the image control gets resized to the picture	When auto size to false oly part of the picture fitting into the current coordinate of the picture box is display

Image Box	Picture Box
When stretch is set to true, the image fits into the original size	When auto size to true the picture box itself gets enlarged or compressed depending of the size
Resizing the image control, resizes the picture it contains	Resizing picture box does not resize its picture even when auto size is set to true
Does not display border around its image	Display a border around its picture.

Multiple Document Interface (MDI)

- The Multiple Document Interface (MDI) was designed to simplify the exchange of information among documents, all under the same roof.

- With the main application, you can maintain multiple open windows, but not multiple copies of the application. Data exchange is easier when you can view and compare many documents simultaneously.

MDI Applications

- You can use three different types of forms in an MDI application
  - An **MDI parent form** acts as the container for the MDI child forms in the solution
  - An **MDI child form** always appears inside the visible region of its MDI parent form
    - A project may have one or more MDI child forms
  - **Standard forms** can be displayed anywhere on the screen and are not contained by the MDI parent form (SDI Single document interface)
    - Standard forms are typically displayed as dialog boxes

- An MDI application must have at least two Form, the parent Form and one or more child Forms. Each of these Forms has certain properties. There can be many child forms contained within the parent Form, but there can be only one parent Form.

- The parent Form may not contain any controls. While the parent Form is open in design mode, the icons on the ToolBox are not displayed, but you can't place any controls on the Form. The parent Form can, and usually has its own menu.

To create an MDI application, follow these steps:

- Start a new project and then choose Project → Add MDI Form to add the parent Form.
- Set the Form's caption to MDI Window
- Choose Project → Add Form to add a SDI Form.
- Make this Form as child of MDI Form by setting the MDI Child property of the SDI Form to True. Set the caption property to MDI Child window.

MDI Form cannot contain objects other than child Forms, but MDI Forms can have their own menus

The child Form can have any number of commands in its menu, according to the application

At design time double click on MDI Open and add the following code in the click event of the open menu.

Form1.Show

And so double click on MDI Exit and add the following code in the click event

End

Combo Box control

- Combo box control which is a combination of a textbox with a short dropt-down list

Creating a Combo BoxDraw a combo box on your form

- Select its style from the property window
- Add items to the List property at design or run time

- There are three types of combo box are:

  1) Drop-down combo box – style 0

  2) Simple Combo Box – style 1

  3) Drop – down box – style 2

0 - The drop-down combo box Presents the users with a text box combined with a dropdown list. The users can either select an item from the list portion or type an item in the text box portion.

1- The simple combo box Displays a text box and a list that doesnt drop down. the users can either select an item from the list portion or type an item in the text box portion.

2 -The drop-down list Displays a drop-down list box from which the users can make a choice. The users cannot enter items that are not in the list

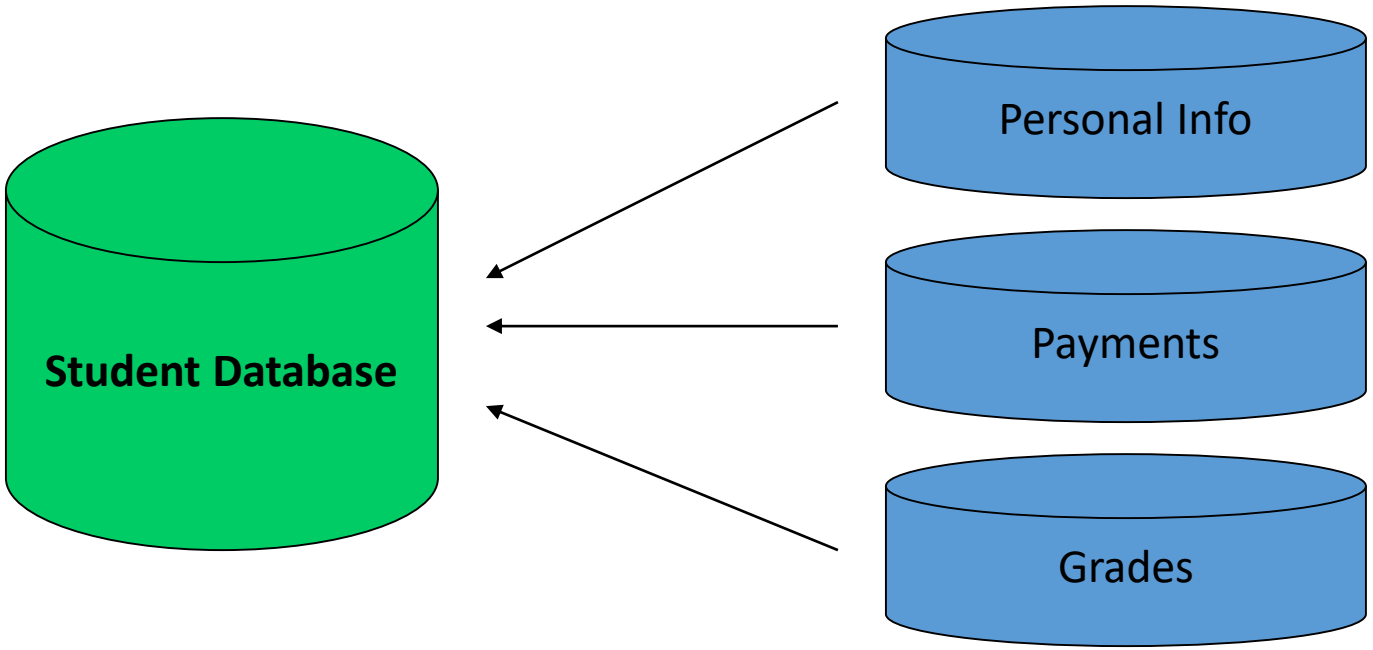
Combo Box features

- **Use the AddItem, RemoveItem, and Clear methods to modify the contents of the list.**
- **Sorted or an unsorted list.**
- **Support the ItemData array and NewIndex property.**
- **No multiple choices**
- **Combo boxes Allows the users to enter choices that are not on the list.**
- **Drop-down list is useful for presenting several choices in a small amount of space.**

Basics of Database Programming with VB6

What is a database?

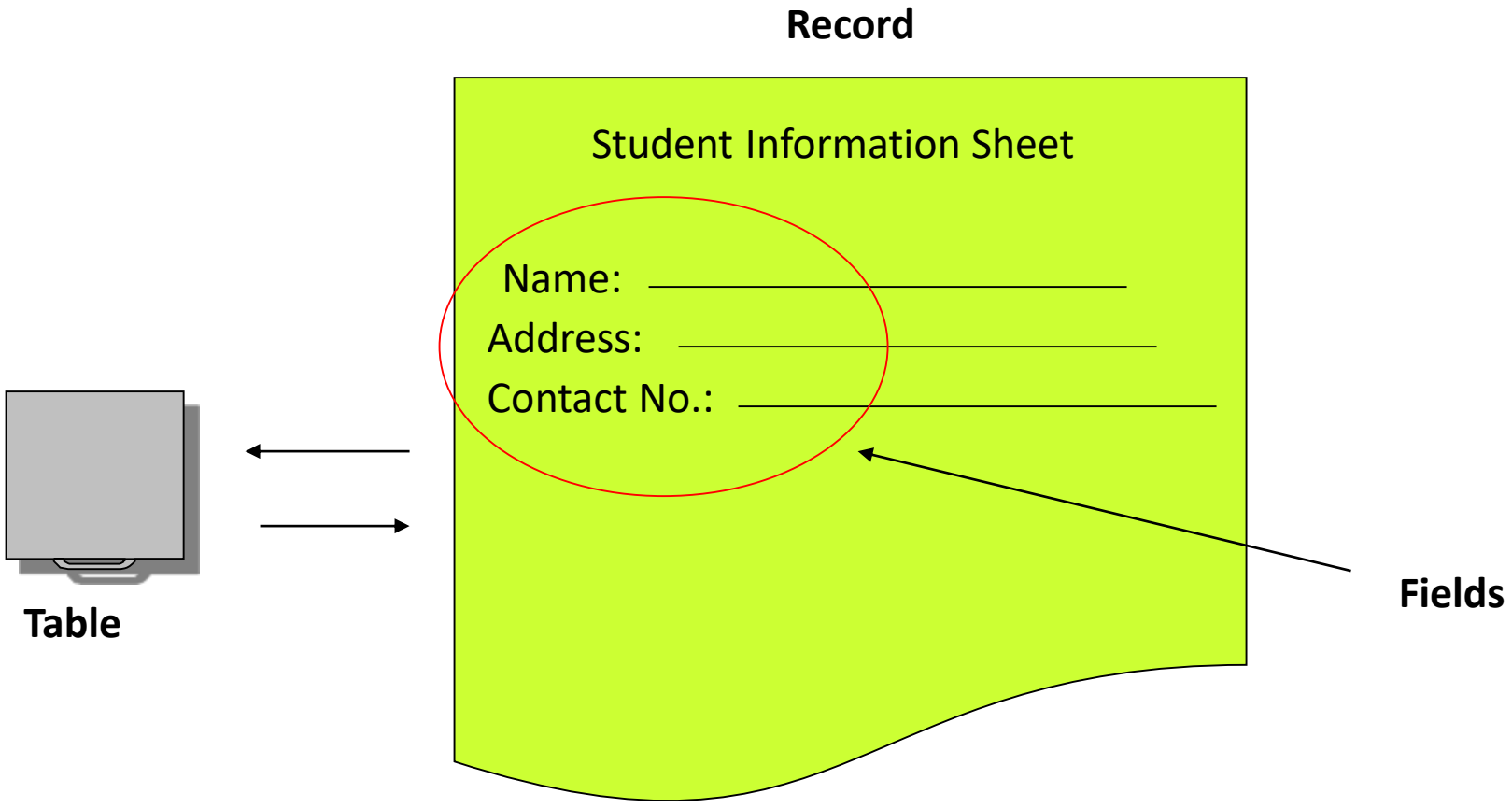
Database – a collection of related data or information, that is stored, modified and transmitted.



Structure of a Database

- **Tables** – collection of related information
- **Records** –single piece of information
- **Fields** – smallest unit of information that is useful to users.

Structure of a Database



Primary Key

each record has a unique identifier that indicates the record and relates it to other records in the other tables. This unique identifier is known as the Primary Key Eg: student_regnum

Foreign key

it’s a field in a table that relates its value to the value of a primary key in another table.

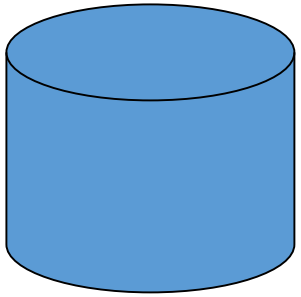
Eg: cust_no in Invoice Table is the foreign key. The same value of the cust_no in the customer table is the primary key

Visual Basic and Database

- **Front End** – the user interface, which the program uses to interact with the user.
- **Back End** – the database, where all data coming from the user are saved, to be retrieved later.

Visual Basic and Database

Database Driver ← ADO Object



(Database Application)

The user interacts with the program by user interface or screens.

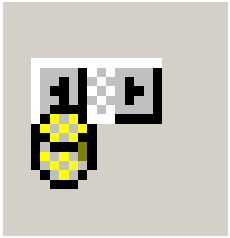
(Visual Basic)

What are database drivers?

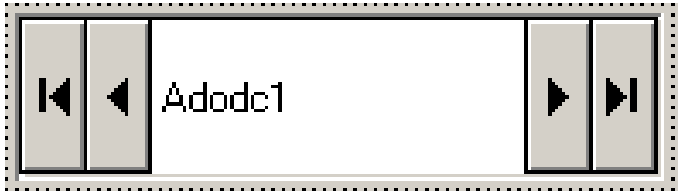
- It allows different programming languages to communicate or get information from different data sources.

Using the ADODC Object

- **ADODC (ActiveX Data Object)** – an object used to connect a Visual Basic program to a database.



ADODC Object on the toolbox



ADODC Object when placed on the form

ActiveX Data Objects (ADO)

- The ActiveX data Objects is a set of high level automation interface via the OLE DB (Object linking and Embedding)

- ADO data control allows the user to create a connection to database quickly by using activeX dataobjects

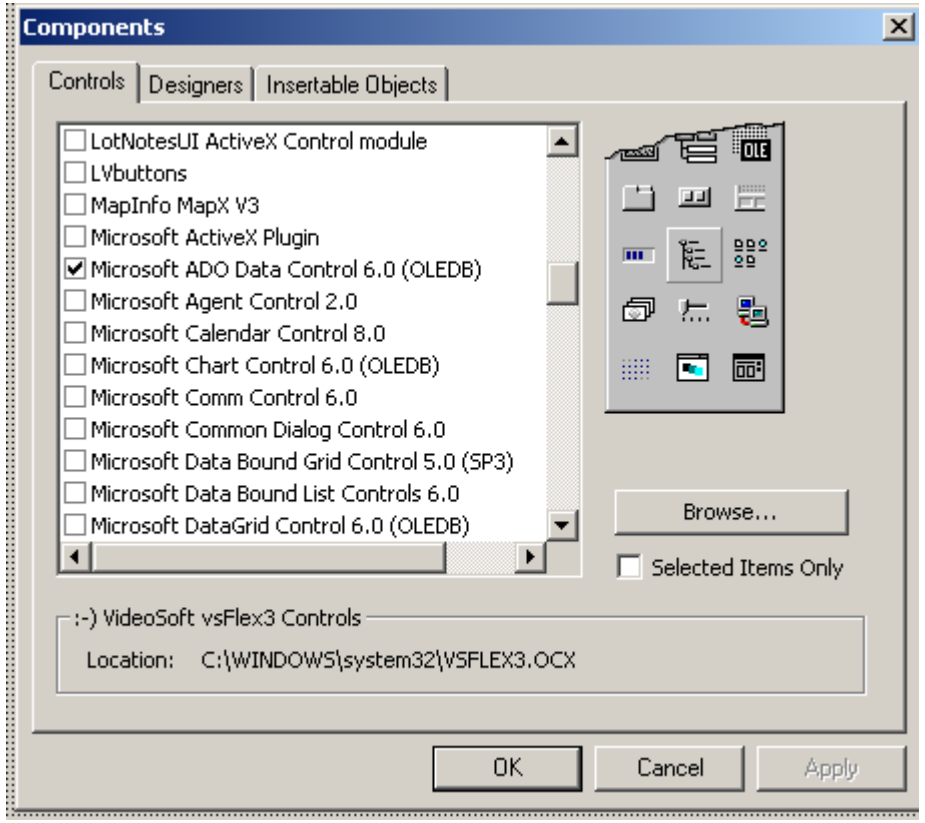
Advantages of ADO are:

- Ease of use
- Can be used with languages such as VB, Java, C++,VBScript and javascript
- Creates less number of objects
- Low memory overhead
- High speed performance
- Supports stored procedures
- ADO allows C++ programmers to access OLE DB interface
- Can access data from any OLE DB sources.

Inserting the ADODC Object

1. Go to **Project** menu, then choose **Components**. (or right click on the toolbox)
2. When the components dialog appears, choose **Microsoft ADO Data Control 6.0**.

Inserting the ADODC Object



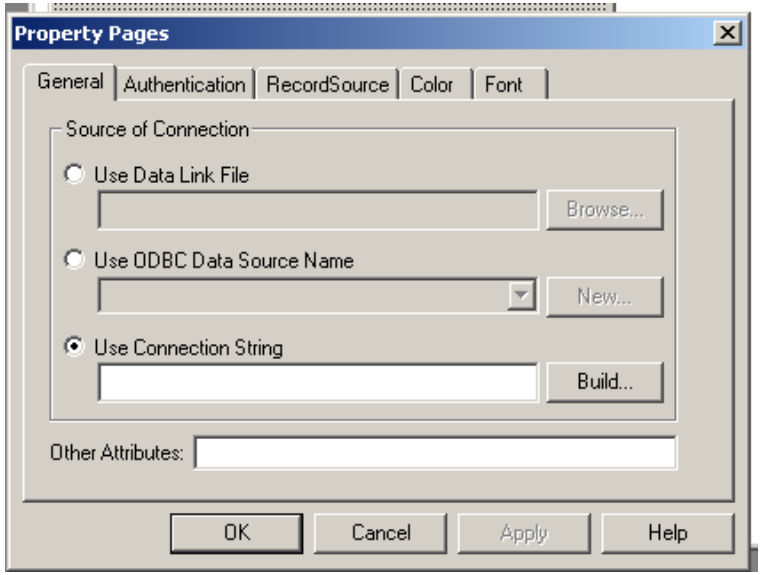
Components Dialog Window

Connecting the ADODC to a database

- **Straight Connection**
  - Setting of the Database Provider and directly specifying the path of the database (location).
- **Using ODBC (Open Database Connectivity)**
  - Creating a Data Source Name (DSN) using the ODBC Administrator of Windows

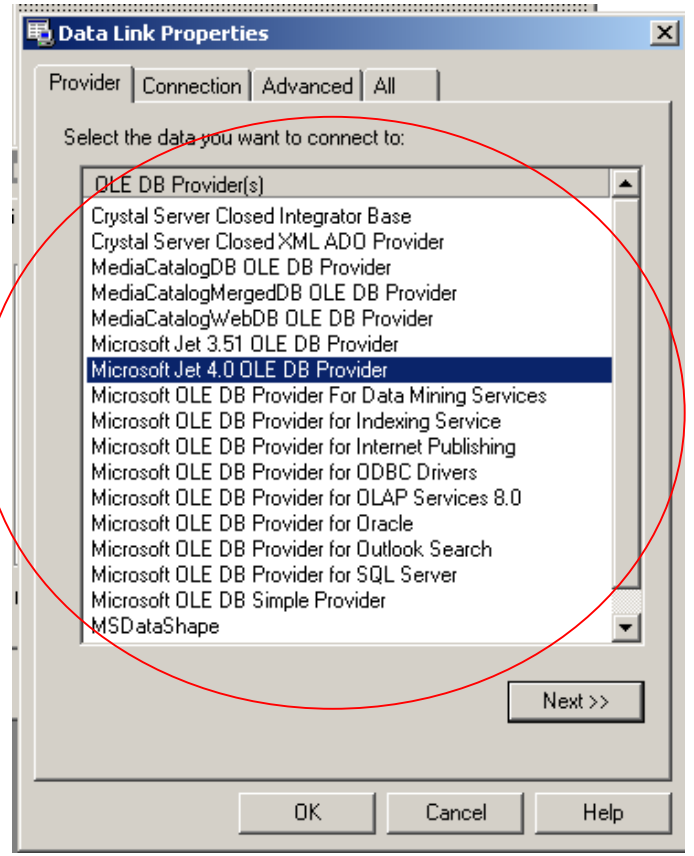
Steps in Connecting ADODC

- Right Click on the ADO Object
- Connect the ADO by using the Connection String or ODBC Data Sources

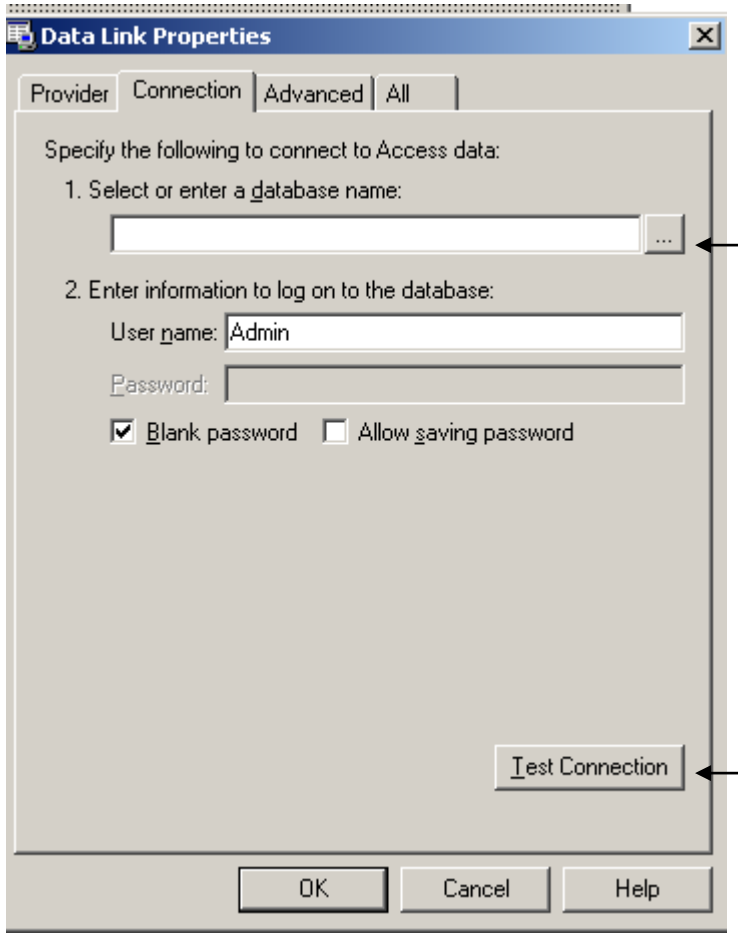


Steps in Connecting ADODC

Choose the correct database provider.



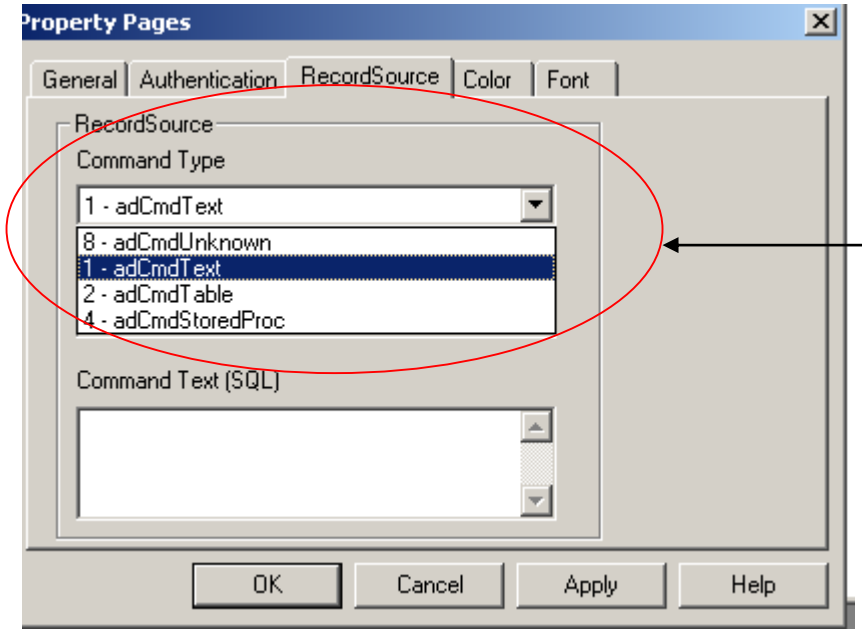
Steps in Connecting ADODC



Specify the correct path of the database

Test the connection if the ADODC where able to communicate with the data source.

Steps in Connecting ADODC



Specify how the ADODC will connect to the table.

The ADO Object Model

- The ADO object model is made up of 7 objects.

1) Connection
2) Command
3) Record Set

<u>The other dependent objects are:</u>

4)Parameter

5)Field

6)Error and

7)Property

Connection

- The connection object is the top-level object which represents a unique connection with a data source.
- The connection of the object to access the required records from database or to update the database.

PROPERTIES:

1) ConnectionString : Contains information used to establish a connection to a data source. It sets or returns a string value

2) Mode: specifies the available permissions for modifying data in a connection

- adModeRead : Indicates read only permission.
- adModewrite : Indicates write only permission.
- adModeReadWrite : Indicates both read and write permissions.

3) ConnectionTimeout: specifies the time in seconds, the server will wait before disconnecting an inactive connection

4) Provider: specifies the name of the provider for a connection object.

Command

- The command object defines a specific command such as an SQL sts that can be processed by a data source.
- Its used to query a database and return records in a RecordSet object.
- To setup a command object:

1) The EXECUTE method its used to execute the SQL sts.
2) RecordSet – associate a command object with it and retrieve the required RecordSet.
3) Specify the command eg: CommandText property

Indicate the type of the command with the command Type property before execution:

adcomdtext : Evaluates source as a textual definition. Its value 1

adCmdTable: Evaluates source as a Table name and its value is 2

adCmdUnknown: Indicates that source property is unknown. Its value is 8

3) Record Set

- A Record Set object holds the results i.e, the set of records, returned of an executed command.
- Its generally used to select and modify data in a row

This objects are:

specify the rows available for selection

Modify, add or delete records

update data source

Manage the state of the Recordset

Specify the order in which records may be traversed

- **RecordSet Properties are:**

  - *RecordCount* – returns the number of records on the table
  - *EOF* – End of File, returns True if the record pointer reaches the end of the table.
  - *BOF* – Beginning of File, returns True if the record pointer reaches the beginning of the table
  - Sort - Specify the order in which records may be traversed

    RS. Sort = “TotalMark DESC”
  - Source – Returns the source(table name) for data in a RecordSet
  - Filter – Specifies the row which are accessible for traversing

    RS. Filter = “Age > 16”

- EditMode: indicate the editing status of the current record. EDITMODE can have one of the following values.

adEditNone: no operations in progress

adEditInProcess: the current record is modified but not saved as yet

adEditAdd: adding new record but not yet saved

adEditDelete: the current record has been deleted

Methods:

- AddNew
- Update
- CancelUpdate
- Delete
- Close
- Find
- Move First
- Move Last
- Move Previous
- MoveNext

Error

- When an errors are generated during operations with ADO objects. Each error as it occurs, its placed in the ERRORS collection of the Connection Object.

- Each error object represents a specific provider Error, not an ADO error.

Synatx

```
Private Sub CmdAdd_click()
On Error go to Err_handle
…….
…….
Exit Sub
Err_handle:
…….
…….
End Sub
```

```
Private Sub Command1_Click()
On Error GoTo errmsg
Adodc1.Recordset.AddNew
MsgBox "successfull added"
Exit Sub

errmsg:
MsgBox "Error in saving Data"
End Sub
```

Parameter

- Data retrieval command can be used repeatedly by varying the parameter values each time, to retrieve specific information.

- Some common properties of a parameter objects are:

1) Name: specifies the name of a parameter
2) Value: Indicates the value of a parameter

Field

- A row of a RecordSet object contains one or more fields.
- Each field corresponds to a column in the RecordSet.
- Each column has attributes such as name, a DataType and a value

The Open Method (ADO Connection)

- Syntax:

  Connection.open ConnectionString, UserID, Password, OpenOptions

A string containing information required to establish a connection to a datasource

The user name at the time of the connection is establishing

The Password at the time of the connection is establishing

Optional
If the value is set to adConnectAsync, this connection will be opened asynchronously

Property

- Every ADO object has a set of unique properties that describes the behaviour of the object
- ADO objects consist of two type of properties:

1) built-in : Properties are part of the ADO object and are always available.
2) Dynamic: properties are normally added to the objects properties collection by the provider and exit only while that provider is being used.

Eg: MyObject.Properties (0)

Using the SQL Statements

- **SQL (Structured Query Language)** – is composed of series of statements and clauses, which, when combined, perform different actions.
- **Select Queries** – returns a specific set of records, based on criteria
- **Action Queries** – performs actions such as deleting, adding or updating records.

Basic Structure of Select

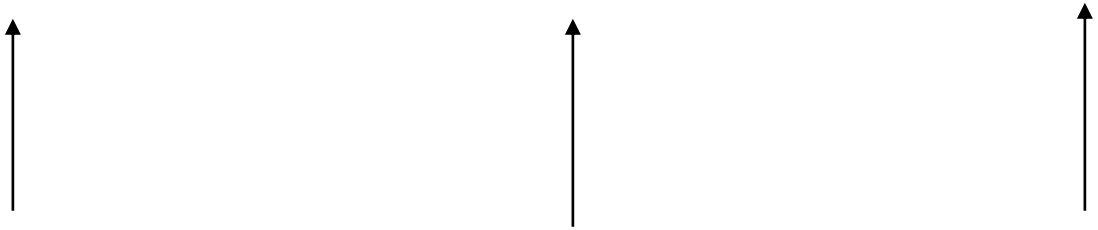
- **SELECT <fields> FROM <tablename> [<where> <condition> <order by>]**

  e.g.

  Select * from StudInfo

Structure of an ADODC commands

<adodc name>.RECORDSET.<method>



The Name of the adodc object

Refers to the table object

Methods that can be done to a table

Types of ADODC methods

- **Record Operations**
  - *Addnew* – used to add records to the table.
  - *Delete* – used to delete records from the table.
  - *Update* – used to save records to the table.
  - *CancelUpdate* – cancels any record-related operations.

Types of ADODC commands

- **Record Navigation**
  - *Find <parameters>* - used to find or search records, based on key fields.
  - *Movefirst* – go to the first record.
  - *Movelast* – go to the last record.
  - *Movenext* – go to the next record.
  - *Moveprevious* – go to the previous record.
  - *Move(record no.)* – go to a specified record no.

Types of ADODC commands

- **Record Counters**
  - *RecordCount* – returns the number of records on the table
  - *EOF* – End of File, returns True if the record pointer reaches the end of the table.
  - *BOF* – Beginning of File, returns True if the record pointer reaches the beginning of the table

Finding Records

- FIND “[*key field*] like ‘*comparing value*’

  e.g.

  adoSTUD.RECORDSET.FIND “[LName] like ‘Locsin’”

  In finding records, always REFRESH the table first. to complete the code:

  adoSTUD.REFRESH
  adoSTUD.RECORDSET.FIND “[LName] like ‘Locsin’”