PERIYAR INSTITUTE OF DISTANCE EDUCATION
(PRIDE)

PERIYAR UNIVERSITY

SALEM - 636 011.

B.Sc. COMPUTER SCIENCE
THIRD YEAR
PAPER – VII : PROGRAMMING LANGUAGE
(VISUAL BASIC)

Prepared By
R.VALARMATHI, M.Sc., M.Phil.,

B.Sc. COMPUTER SCIENCE
THIRD YEAR
PAPER – VII : PROGRAMMING LANGUAGE (VISUAL BASIC)

Unit: I

1.1 Welcome to VB
1.2 Creating an Application
1.3 2nd Look at IDE, Forms and controls
1.4 Variables in Visual Basic

Unit: II

2.1 Writing Code in VB
2.2 Working with Files

Unit: III

3.1 Menus
3.2 MDI Applications
3.3 Debugging Tips
3.4 The Common Dialog control
3.5 Introduction to Databases
3.6 Working with the Data Control

Unit-IV

4.1 DOA
4.2 Additional Controls Available in VB 6.0
4.3 ActiveX data Objects

Unit-V

5.1 Crystal and Data Reports
5.2 Distributing your application
5.3 ActiveX
5.4 ActiveX and Web pages
5.5 ActiveX Documents

INTRODUCTION

Dear students,

Visual Basic is considered “The fastest and easiest way to create applications for Microsoft Windows”. Visual Basic provides you with a complete set of tools to simplify rapid application development.

Visual Basic is an event driven programming language. The **“Visual”** part refers to the method used to create the graphical user interface. The **“Basic”** part refers to the BASIC language, a language used by more programmers than any other language in the history of computing.

This book is intended to the student how to program in Visual Basic. Totally this book contains five units. The **first unit** explores the basic concepts of Visual Basic.

The **second unit** explores Writing code in Visual Basic and how to working with files. The **third unit** explores Menus, MDI applications, Debugging Tips, The common Dialog Control, Why databases and How to working with the Data Control.

The **fourth unit** explores the DAO, Additional Controls Available in VB6.0 and ActiveX Data Objects. The **fifth unit** explores the Crystal and Data Reports, Distributing your application, ActiveX and Web Pages and ActiveX documents.

All the above said units of lesson of this book have been prepared by **R.VALARMATHI, M.Sc., M.Phil.,** to make your task much easier while going through it.

PRIDE would be happy if you could make use of this learning material to enrich your knowledge and skills to serve the society.

SYLLABUS

Unit: I:

Welcome to VB: What is Visual Basic - Features of Visual Basic - Visual Basic Editions - The Visual Basic Philosophy - Developing an Application. Creating an Application: Objectives- The Tool Box – Project Explorer - The Properties Window – The Form Window – What does Visual Basic 6 have for you to create Applications.2nd Look at IDE, Forms and controls: Objectives - The Form – The Working with a Control – Opening the Code Window. Variables in Visual Basic: Objectives – What is a Variable.

Unit: II:

Writing Code in VB: Objectives – The Code Window – The Anatomy of Procedure- Editor Features – The For .. Next Statement – The Decision Maker… If..Loop – The While loop – Selective Case… End Select. Working with Files: Objectives – Visual Basic File System Controls - Types of Files – Working with Files.

Unit: III :

Menus: Objectives – Building the User Interface. The first step – All about Menus. MDI Applications: Why MDI Forms – Features of an MDI Form- Loading MDI Forms and Child Forms – The Active Form property. Debugging Tips: Objectives – The Debugging Methods. The Common Dialog control: Working with the Common Dialog Control – The file open Dialog Box – Saving a file – Changing the color. Introduction to Databases: Why databases – What is a Database – Which Database. Working with the Data Control : The Data Control – The Bound Controls – Caution – Coding.

Unit-IV:

DOA: The Jet Database Engine – Functions of the Jet Database Engine – SQL – The DAO Object Model. Additional Controls Available in VB 6.0 – Objectives – SSTab Control. Active X data Objects – Objectives Why ADO – Establishing a Reference.

Unit-V:

Crystal And Data Reports: Crystal Reports – Data Report. Distributing your application: Objectives – Working with the Packaging and Deployment Wizard. Active X: Objectives – What is ActiveX – Why ActiveX. ActiveX and Web pages: Objectives – ActiveX and Internet. ActiveX Documents: The Application Form Document . Sample Application in VB Like Inventory Control.

TEXT BOOK:

1. “ Programming With Visual Basic 6.0”- Mohammed Azam.
   - Vikas Publishing House Pvt Ltd.

UNIT – I

Objectives:

The objectives of this unit are to introduce the basic concepts of Visual Basic and explaining how to create an Application, giving forms and controls, variables in Visual Basic.

Contents:

1.1 WELCOME TO VISUAL BASIC (VB)

It is considered "the fastest and easiest way to create applications for Microsoft windows(r)". Whether you are an experienced professional or brand new to windows programming. Visual Basic (VB) provides you with a complete set of tools to simplify rapid application development (RAD)

Windows based applications have a consistent user interface, that helps the user to simply 'point and click'. All windows application provides a picture (or an 'icon' as the jargon goes) or a button for a function. The user will point at that pictures with a mouse and click. The computer simply jumps up to perform (or sometimes declines) the task. Visual basic is an ideal medium for developing Windows based application.

What is Visual Basic?

Visual Basic is an event driven programming language. The "visual" part refers to the method used to create the graphical user interface (GUI).

The "Basic" part refers to the BASIC language, a language used by more programmers than any other language in the history of computing. Visual Basic has evolved from the original BASIC language and now contains several hundred statements, functions, and keywords, many of which relate directly to the windows GUI.

Note: the Visual Basic programming language is not unique to Visual Basic. The Visual Basic programming system. Applications Edition included in Microsoft Excel, Microsoft Access, and many other windows applications uses the same language. The Visual Basic programming system, Scripting Edition (VBScript) for Internet programming is a subset of the Visual Basic language. The investment you make in learning Visual Basic will carry over to these other areas.

Where Visual Basic can be used,

1. To create a small utility for yourself.
2. An Application for a department, work group, a large enterprise-wide system.

3. Distributed applications spanning the globe via the internet.

- Data access features allow you to create databases and front-end applications for most popular database formats, including Microsoft SQL Server and other enterprise-level databases.
- Active X (TM) technologies allows you to use the functionally provided by other applications, such as Microsoft word, Microsoft Excel spreadsheet, and other windows applications.
- Internet capabilities makes it easy to provide access to documents and applications across the internet from within your application. (Active X documents)
- Your finished application is a true. exe file that uses a run-time dynamic link library (DLL) that you can freely distribute. (Application Setup Wizard)

Visual Basic (VB) is available in three versions, for a specific set of development requirements.

- The Visual Basic **Learning Edition**, allows the programmers to easily create powerful applications for Microsoft Windows 95 and Windows NT (r). It includes all intrinsic controls, grid tab, and data-bound controls. Documentation provided with this edition includes Learn Visual Basic Now, a printed Programmer's Guide, online Help, plus Visual Basic books Online.
- The **professional Edition** provides computer professionals with a full-featured set of tools for developing solutions for others. It includes all the features of the Learning edition, plus additional Active X controls, including internet controls, and the crystal Report Writer. Documentation provided with the Professional edition includes the Programmer's Guide, online help, the component Tools Guide and the Crystal Reports for Visual Basic User's Manual.
- The **Enterprise Edition** allows professionals to create robust distributed applications in a team setting. It includes all the features all of the professional editions, Plus

The Automation manager,

Component manager,

Database management tools,

The Microsoft visual SourceSafe (TM) project-oriented version control system, and more.

The Visual Basic Philosophy

Windows became popular because of its easy and intuitive GUI or Graphical User Interface. Visual Basic helps create applications that will have the same GUI as windows. This makes it very easy for users to learn how to use the software. The Windows GUI is event driven. You have seen that unless the user initiates an action nothing happens. You have to click a button for something to happen. Visual Basic is an event driven programming language.

The Controls

A critical looks at any applications will tell you that it consists of a number of programs or procedures that perform various activities. For example, there is one program that accepts text data from the user. There is another program that verifies the numeric data or validates the data entered by the user.

Then there is a routine, that checks the options selected by the user. Another routine to update a file or database and so forth. The people behind visual basic decided to create special routines that would perform a specific task.

Words like routines, procedures, programs were discarded and they decided to call these special routines with a new name called controls.

The Properties

Just creating the controls was not enough. The controls are given some attributes that are clearly defined. A control is supposed to do only such and activity. For example a control like the Label Box should not allow the user to edit its contents. A control like the text box that can accept multiline text or allow word-wrap. All these attributes are called properties. So you have controls that perform certain activities. And each control has certain properties.

EVENTS

Now there are a number of things can happen to control. For example, a command button control can be clicked, as you click the start button in Windows.

These are all some of the events that can take place on an application. Visual Basic allows you to write code to respond to such activities. The wonderful thing about this is that you need to write code only for those events that you are interested in. For example, if your program has to respond when the user clicks a Command Button, you need to write code only for the click event.

Methods

The action taken when the event occurs is the method. You may want to exit the program when the user clicks on the command Button, or you may show a new picture when the user clicks on a Textbox! Controls in Visual Basic have many built in methods. As you write the code for the various controls you will come across the methods.

From a laymen’s point of view, a method is a piece of code that accomplishes a task. So in an event driven program, there are ‘controls’, which have ‘Properties’. When an ‘Event’ occurs to the ‘Controls’ some ‘Methods’ are invoked.

Developing an Application

1. Design the User interface
2. Write code to respond to User Input/Events

Design the User Interface

The user interface is built using the controls and setting the properties for the controls. For example the location of the TextBox, where the user will enter the customer ID.

Write Code to Respond To User Input/Events

The code invokes the methods associated with the controls. If the user clicks on the control that displays the next record form the database, or the user selects particular option, or wants to 'find' the details of a client, etc. All such events have to be acted upon. There are a number of built-in keyword, associated with the controls that accomplish the given task, As a programmer it is your job to orchestrate the activity of the controls through the methods.

1.2 CREATING AN APPLICATION

Creating an application in Visual Basic means working with projects. A project is the collection of files you use to build an application.

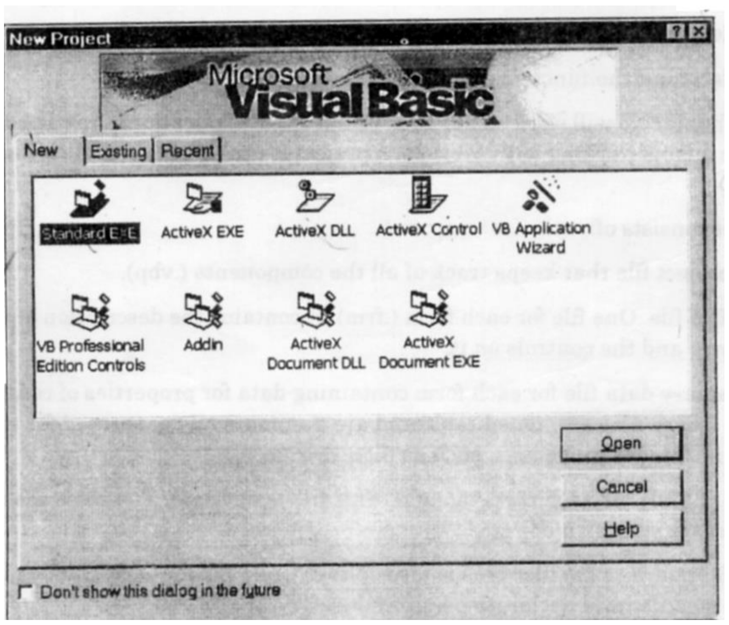
A project consists of:

- One project file that keeps track of all the components. (.vbp)
- The .frm file. One file for each form (.frm). It contains the description of the properties of the form and the controls on it.
- One binary data file for each form containing data for properties of controls on the form (.frx). These files are not editable and are automatically generated for any .frm file that contains binary properties, such as picture or icon.
- The (.cls) file for each class module. This file is optional. The class file is created when you create your own objects.
- The standard (.bas). This is also optional, one file for each standard module. It contains module level declarations, procedures.
- The Active X control (.ocx) file, becomes a part of the project file only if optional controls are added in your program.
- The resource (.res) file, contains bitmaps, text strings that are used in your program. You can have only one resource file.

The project file is simply a list of the files and objects associated with the project, as well as information on the environment options.

This information is updated every time you save the project. You can convert the project into an executable file. You can also create other type of executable files such as the .ocx, .dll files, etc. Invoke Visual Basic 6.0 (VB6) by either double clicking on the shortcut path or by going through the pull-up menu

from Start.After a while it will show a screen that looks like the figure below. This is New Projects Screen or a dialog box.

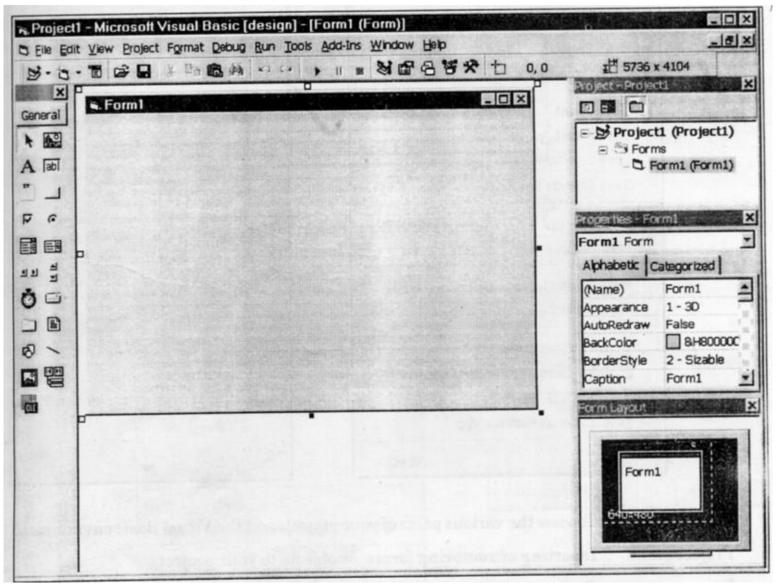


Here a number of icons are displayed along with the types of projects that each will start. The New Project Screen has three tabs. The current tab is 'New'. The other two tabs are 'Existing' and 'Recent'. Clicking the 'Existing' tab will display the existing projects on your system. Clicking the 'Recent' tab will display the projects on which you have recently worked.

Choose the standard. Exe icon by clicking on it, you will then see the screen that will look something like the following screen.

Type of Files

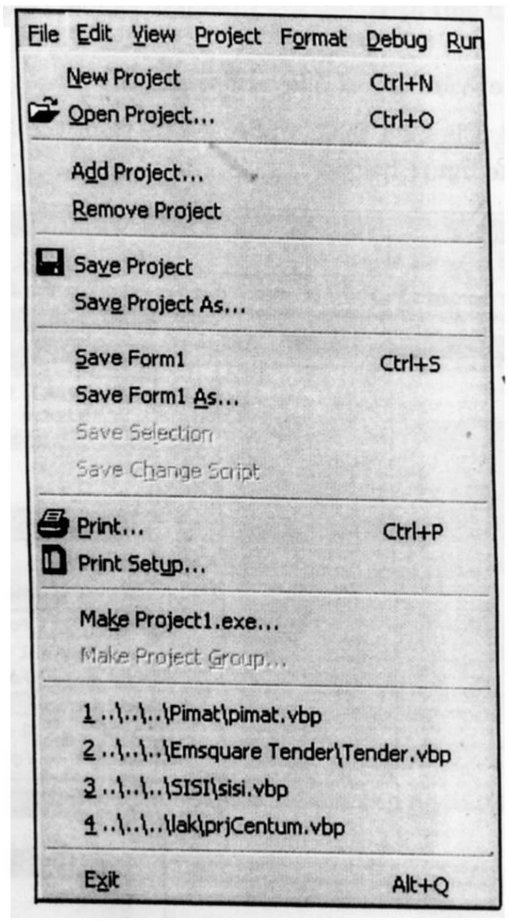
- ❖ **.vbp:** visual basic project file
- ❖ **.vpg:** Visual Basic Group file
- ❖ **.mak:** project file built with earlier versions of isual basic.

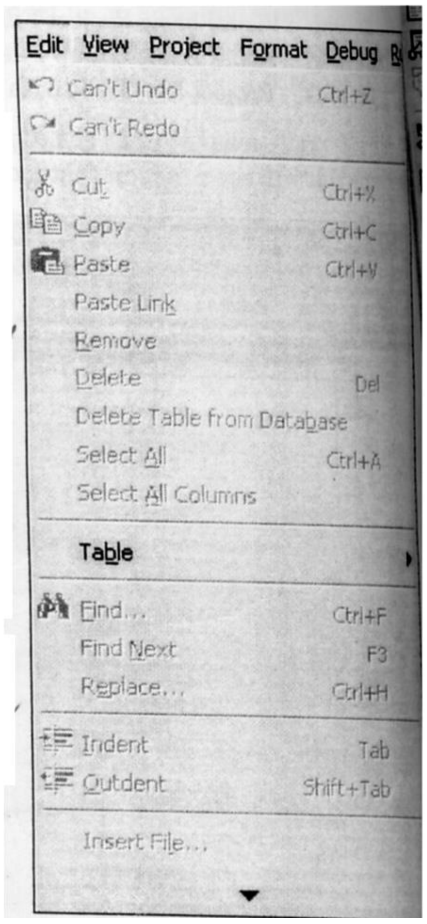


Below the title bar is the menu bar. The menu bar has the following menus

File menu: to open and save a new or existing project , to print and to make a project file.

Edit menu: for all editing requirement Cut, paste, Find Undo, etc.

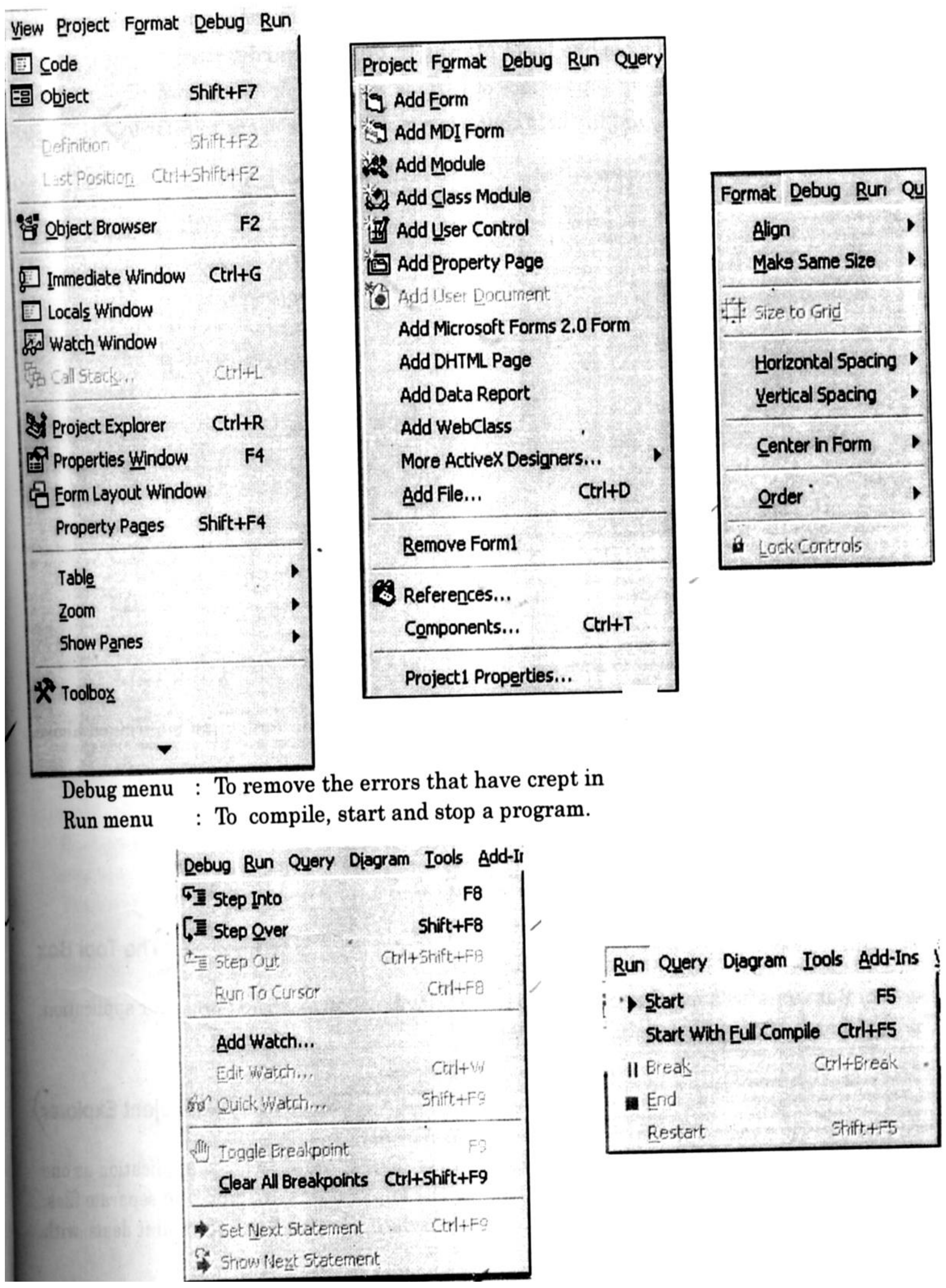




View menu : To view the various parts of your project, and Visual Basic environment

Project menu : Inserting or removing forms, or objects to your project.

Format menu : For spacing, placing and appearance of controls in the form.


Debug menu : To remove the errors that have crept in
Run menu : To compile, start and stop a program.

Debug menu: To remove, the errors that have crept in

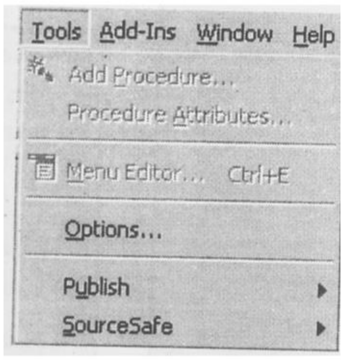
Rum menu: to compile, start and stop a program.

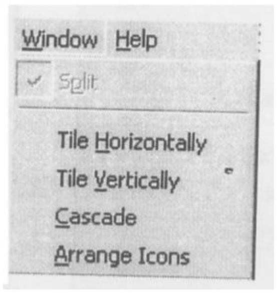
Tool menu: To add procedure and to customize the environment for your project.

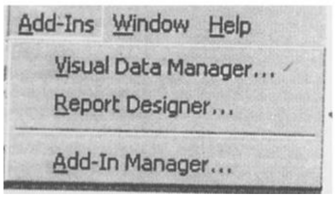
Add-Ins menu: To add tools like **Data Manager**: other wizard, etc.

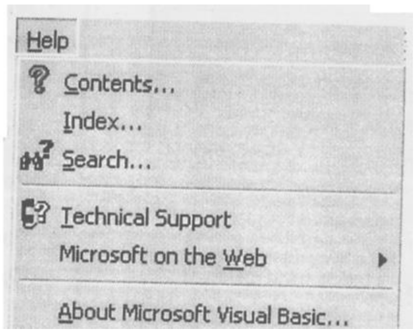
Window menu: Arranging appearance of various windows on the desktop.

Help menu : For the on-line help that every programmer needs to refer to.









In order to build your application you need the aid of all the tools/aids mentioned above.

The most visible are the

1. The tool box
2. The project explorer window
3. The properties window and the form itself.

Project Explorer

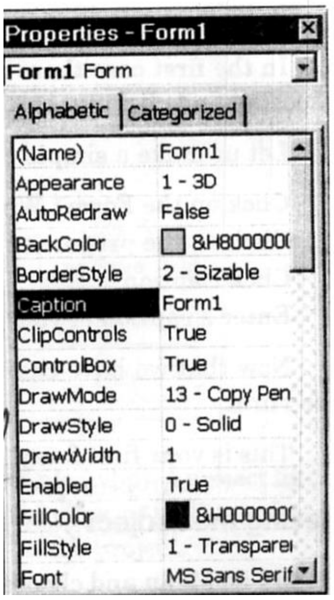
The project explorer is located on the right side (usually). It organizes the application as one project. All the code and controls that are used in the applications are stored in separate files.

It is called a 'project explorer' because it has an interface like the explorer and it deals with the project. The project Explorer has three icons on its tool bar. Each icon represents a function.

1. To view the code
2. To view the controls
3. To show or hide the forms.

Below the 'project Explorer Window' you will see the properties window. This window lists all the properties for an object or control used in visual Basic.

For example you can change the caption, the height, the width, etc., Each object has a number of properties that can be changed as the need dictates.

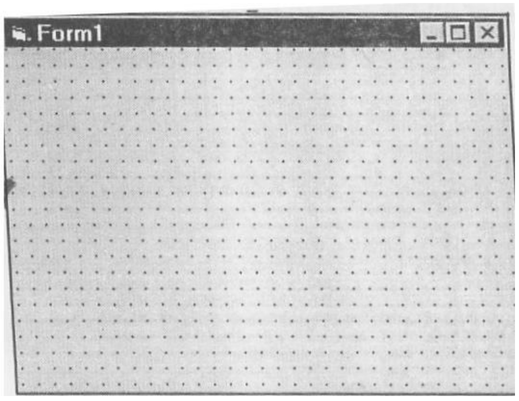


(Name)	Form1
Appearance	1 - 3D
AutoRedraw	False
BackColor	▢ &H8000000(
BorderStyle	2 - Sizable
Caption	Form1
ClipControls	True
ControlBox	True
DrawMode	13 - Copy Pen
DrawStyle	0 - Solid
DrawWidth	1
Enabled	True
FillColor	■ &H0000000(
FillStyle	1 - Transparei
Font	MS Sans Serif

The Form Window

The controls that we will learn about, their properties, the Project Explorer, etc., are all concerned with this form.

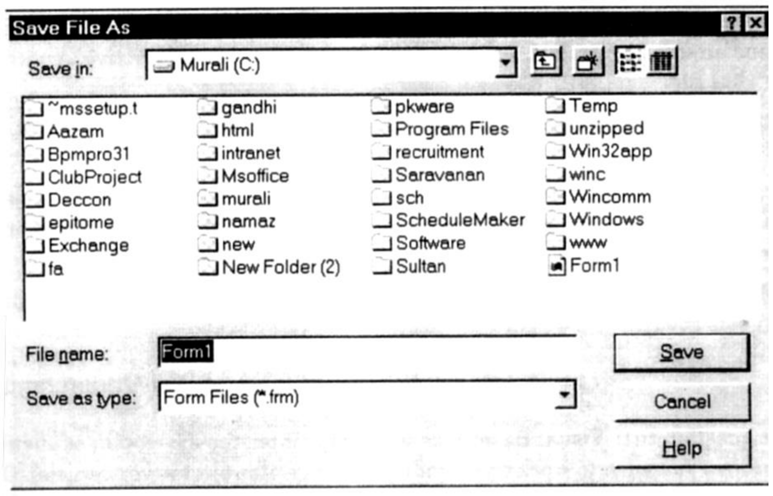
The form has a title bar. It has the Minimize, Maximize and the Close buttons.



Saving the Project

Go to file menu and choose the Save project option.

Then you will be asked to give a name for the project. Give an appropriate name.

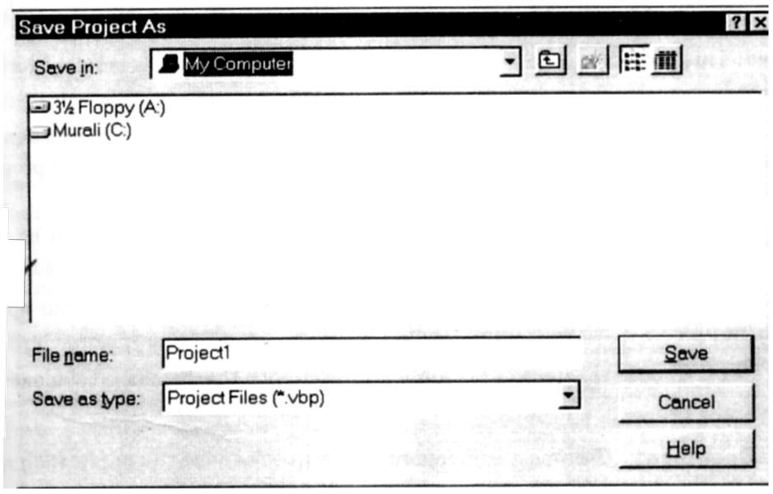

Save File As

Save in: Murali (C:)

~mssetup.t	gandhi	pkware	Temp
Aazam	html	Program Files	unzipped
Bpmpro31	intranet	recruitment	Win32app
ClubProject	Msoffice	Saravanan	winc
Deccon	murali	sch	Wincomm
epitome	namaz	ScheduleMaker	Windows
Exchange	new	Software	www
fa	New Folder (2)	Sultan	Form1

File name: Form1

Save as type: Form Files (*.frm)


Save Project As

Save in: My Computer

3½ Floppy (A:)
Murali (C:)

File name: Project1

Save as type: Project Files (*.vbp)

Understanding projects

Following are the options available with visual Basic:

Standard: This project type must be chosen if you wish to develop a small or large standalones application.

ActiveX EXE: Choose this option if you wish to create an executable component. An ActiveX executable component can be executed from other application. This will be program that can provide functionality to a number of other applications.

Active Control: This helps to create a custom ActiveX control that can be used in other application. These are like the third party controls that you buy from other software vendors.

ActiveX DLL: Like the ActiveX EXE it provides added functionality to your application will work ‘in-process’ with your application.

Data project: Choose this option to create a project with the database components.

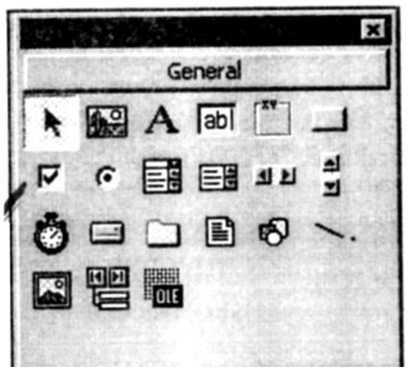
IIS: This helps to create an Internet application.

ActiveX Document: creates a component that can take over the application at runtime. It creates an internet application that can be executed from a browser.

DHTML Application: creates an application that can be executed from a web browser only.

What Does Visual Basic 6 Have For You To Create Application?

Depending upon the edition of VB6, you have, the control you see on your Tool Box will be a little different.



Customizing This Toolbar

The Tool Box can be customized, by following the steps given below:

- ❖ Place the Mouse on the Tool Box and click the right mouse button.
- ❖ From the pop-up menu choose components.
- ❖ A window pops up displaying the controls that are available.
- ❖ Click on the controls that you want to add. Push the OK button and you have them on your ToolBox.

Text Box Control

In order to display or accept user input in the form of text like name, customer ID, etc., Visual Basic 6 provides you with this control. It is used to display text and allows the user to edit the data in the box. A text Box control is also called an edit field or edit control.

The Picture Box

A picture Box control can display a graphic from a bitmap, icon, or metafile, as well as enhanced metafile, JPEG, or, GIF files.

Label Box

It allows you to display text that you don’t want the user to change, such as a caption under a graphic.

Option Button

It allows to display multiple choices from which the user can choose only one option.

Frame

The frame controls allows you to create a graphical or functional grouping controls. To group controls, draw the Frame first, and then draw controls inside the frame.

List Box

List box control to display a list of item. The list Box has a small limitation. It can display only a set of items that are available.You cannot add a new item to you repository in future.

Combo Box

The user can either choose an item from the list or enter a value in the text box. The new value entered can be added to the existing data. The List box and the Combo Box display data usually from a database.

Data

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

Hscrollbar (Horizontal Scroll Bar)

Provides a graphical tool for quickly navigating through a long list of items or a large amount of information, for indicating the current position on a scale, or as an input device or indicator of speed or quantity.

VScroll Bar(vertical scroll bar)

Provides a graphical tool for quickly navigating through a long list of items or a large amount of information, for indicating the current position on a scale, or as an input device or indicator of speed or quantity.

Command Button

Creates a button that the user can choose to carry out a command. The user will click on this button and the computer will perform the task associated with the button.

Check Box

Creates a box that the user can easily choose to indicate if something is true or false, or to display multiple choices when the user can choose more than one.

The Drive, Directory and File List Controls

These controls are used to display available Drives, Directories and Files. The user can select a valid drive on his system. The user can be see a hierarchical structure of directories and files.

The Line And Shape Controls

These controls are used to draw lines, squares, circles, etc.

The Image Control

This is very similar to the picture Box control. Images displayed in an Image control can only be decorative and use fewer resources than a Picture Box.

OLE (Object Linking and Embedding)

This controls allows you to link your program to another object or program.

Other Tools for Software Development

The Visual Basic Integrated Development Environment (IDE) consists of the following elements:

Menu Bar

Displays the commands you use to work with Visual Basic. Besides the standard File, Edit, View, Window, and help menus, menus are provided to access functions specific to programming such as project, Format, or Debug.

Content Menus

Contain shortcuts to frequently performed actions. To open a context menu, click the right mouse button on the object you are using. The specific list of shortcuts available from context menus depends on the part of the environment where you click the right mouse button.

Tool Bars

Provide quick access to commonly used commands in the programming environment click a button on the toolbar once to carry out the action represented by that button.

Tool Box

Provides a set of tools that you use at design time to place controls on a form.

Project Explorer Window

Lists the forms and modules in your current project. A project is the collection of files you use to build an application.

Properties Window

Lists the property settings for the selected form or control. A property is a characteristic of an object, such as size, caption, or color.

Object Browser

You can use the object browser to explore objects in Visual Basic and other applications, see that methods and properties are available for those objects, and paste code procedures into your application.

From Designer

Serves as a window that you customize to design the interface of your application. You add controls, graphics, and pictures to a form to create the look you want. Each form in your application has its own form designer window.

Serves as an editor for entering application code. A separate code editor window is created for each form or code module in your application. The programmer can change the font size of the code.

Form Layout Window

The form layout window allows you to position the forms in your application using a small graphical representation of the screen.

1.3. 2ND LOOK AT IDE, FORMS AND CONTROLS :

The Form

A form is actually a control and has its own properties, events, and methods with which you can control its appearance and behavior. You can set a form's properties at design time in the Properties window or at run time by writing code.

Setting Form Properties

A form in Visual Basic has as many as 50 properties. Each of these properties will affect the appearance and behavior of the form in subtle or very obvious ways.

Working With The Properties Window

You assign properties to a control during the Design Time by invoking the properties window. The properties window allows to change the current settings of the controls.

In order to access the properties of a control,

Click the right mouse button.

Choose the properties option from the pop-up menu:

OR

Select the control . Press F4. This will display the properties window, with the current setting for the Form.

Name

It is the name by which the form is referred to in your code. Visual Basic by default names the forms as Form1, Form2,Form3, etc.

Caption

The caption can be changed at runtime. The caption must be meaningful and informative to the user.

Picture

The name of the file that contains the picture to be displayed on the form. This can be set at design time and also changed at runtime.

Background Color

This property determines the background color of the form. In order to change the settings of this property, double click on the value on the right. A color palette will pop up. Select the color of your choice.

The Control Box

The value for this is True or False. If it is set to True, the control Box is visible on the top left-hand corner of the form. If it is set to false, it is not visible.

Min Button and Max Button

The value for each of these properties is true or false. Set this to false if you do not want the user to minimize or maximize the form during runtime.

Movable

The default is true. if set to false, this form cannot be move by the user during the runtime. This property is set to false when you do not want the user to move the form around and thereby miss out some important information.

Border Style

This property determines the type of window that the user will see during runtime. You can allow the user to resize the window, or allow him to move the window around.

Font Properties

The font properties for a Form include the following:

Font name: Name of the font

Font bold: If set to True, the text will be displayed in bold.

Font Size : You can set the size of the text in points.

Position Properties

Properties like Left, Top, Height and Width can be set at design time as well as runtime to locate the form at a place of your choice.

Startup Position

This property is set at design time will specify the position of the form at runtime.

Form Methods

Forms have methods that are used in code to control their behavior. Visual Basic supports many methods to change the appearance and behavior of the form at runtime.

Move

The move method for example, allows the programmer to position the form at a desired location on the screen.

The syntax of the move methods is as follows:

Form name. Move Left, [Top], [Width], [Height]

Each of the liens of code given below will result in the form changing its size and position.

Frminvoice. Move. 0.0, 3500, 4000

Frm invoice: Move 4000, 4000, 4000, 4000.

Graphic Methods

There are other properties for drawing lines, circle, clearing the form for any drawing object. There are methods like getting the color a point at a particular location.

Circle : to draw a circle (s)

Line: to draw a line (s)

Pset: to draw a point with given color at a given location.

Point: returns the color of screen at the given location.

Show Method

Before going to the Show Method we need to cover a little ground about the lifecycle of a form. A form comes to life when the user click starts an application.

Initialize

As the name suggests, all the variables associated with this form are initialized.

Load

During the load event, the form with all its properties and variables is loaded in memory. This load event occurs whenever the 'show' method is executed or a form property is referenced.

Activate

This event occurs when the form gets user input. This event also occurs when the Show method or set Focus method of the form is called.

Deactivate

This event occurs when an another form gets the focus.

Unload Event

When the user closes the Form, the Form is unloaded from the memory.

Terminate

The final event in the lifecycle of a Form. All the memory that was held for the Form variables is released.

Show Method

The show method displays the form to the user. It is like setting the 'visible' property to True.

Show Style

There are two options.

1. The form can be opened as Model Form. In this case this form will have the focus as long as it is displayed.
2. The default option is a modeless form. The user can interact with other forms even while this Form is displayed.

Hide Method

You can unload any form that is not immediately required by the user. As and when the user wants a particular form, the same can be loaded. But such an approach has its drawbacks. Loading and unloading forms every now and then can be time consuming. In order to be able to provide the forms quickly to the user, it is good idea not to unload the form but to hide it. The 'Hide' method comes in handy.

The syntax is

Form invoice. Hide

This will not unload the form, it will merely make the form invisible to the user, it is the opposite of the show method.

How do you put of create the control on the Form?

There are two methods of creating the control on the Form: The first method has following steps:

1. Let us say you want to put a command button on the form.
2. Click with the left mouse button on the command button.
3. Next move the mouse pointer to the location on the toolbar where you want the command button on the form
4. Notice that the pointer has changed to crosshair.
5. Hold down the left mouse button and drag in any direction.
6. Release the left mouse button.

or

2. Double-click the desired controls on the Toolbox. That control will appear at the center of the form. Now you can drag the control and place it at a location of your choice.

Working with a Control

After you have placed the control on the Form, you may want to change the location or the dimension.

Windows application have a uniform size for their buttons.To re-size the control you are given a total of eight sizing handles.The four handles on the four corners are to increase or decrease the length and breadth of the control proportionately.

The two sizing handles on the horizontal edges are used to increase or decrease the height of the control.

The two sizing handless on the vertical edges are used to increase or decrease the width of the control.

To move a control to another location, click on the control, hold down the left mouse button and drag the control to a location of your choice.

The Code Window

After having placed the controls in the right place with the right temperament, you need to tell the control how it must respond against a given event. For example, how should the command Button control respond when the user clicks on it.

Opening the Code Window

Open the code Window by pressing the right mouse button on the control. Choose the “view code” option.

OR press **F7** after clicking on the control.

OR click on **View** in the menu bar. Then click **view code.**

Anatomy of the Code Window

The title bar will contain the name for the project and Form Name. Next there are two Combo Boxes. One holds the text **Form** while the other holds the text **Load**. This means that **Form** is the name of the object and **Load** is the event that you want to write the code for.The code for an event must be entered in the code window.

The lines given below provide a framework within which you can enter the code.

```
Private sub command_click( )
End sub
```

The **private** means that the variables declared and the code used here can be used only by this function. Then we have the word **sub.** This is short for sub-routine or function . Command_click() is the name of the function which is self-explanatory. **End sub** means end of this sub-routine.

The First Example

This is the first example and it is very simple one using only those skills or topics that we have discussed so far. We need to create a single user-screen, which will help a child learn the three basic colors. This means that three different colors need to be displayed. When the user clicks on a particular color, he must see a message telling him the name of the color that has been selected.

We need three buttons with three different colors.

We need a place where the message can be displayed.

We need a button that will tell the computer that we are through with this exercise.

Step 1: Creation and placement of controls

Create four command button controls

Create one Text Box.

Design the form so that you arrive at a screen shot like the figure above you are however free to place the controls the way you like.

Step2:Naming the controls

Click on Command Button control and press **F4** to access the properties window.

Change the option appropriately for each of the Buttons. You may even leave the caption blank.

On one button, the caption must be **"&Quit**". This will be the button that you will click to exit from this program.

Important

Always remember to provide a provision for the user to exit from a Form after clicking a button like **"Exit"** or **"Quit"**. This will allow you to perform necessary tasks like saving data, updating a database. etc. If the user assumes control, you may as well not develop applications.

Step3: Changing the Color for the Command Buttons

Click on the style property and change it to **'Graphical'** and **'standard'.**

Click on the Back color property and change it to the color that you want the button to display.

Observe that the color of the Command Button control has changed to the color you have selected.

Step4: The Textbox control

Click on the TextBox control and press **F4**.

In the properties window, click on Text. The default value for this property is "Text1" since this is Textbox1. Delete the value .

Now you have Four Buttons.

Three of the buttons have the relevant colors .

One button s the **"Quit"** button.

Ans one is blank TextBox.

Now entering the code:

Click on one of the buttons. Press **F7** or double-click on the button. The Code Window will pop up as shown in the screen shot.

Click on the Event Window. Select click. Usually the click event procedure will be displayed as the default procedure

Enter the following line

Text 1. text = "This is Green"

For the two Command Buttons, add the appropriate code just as explained above.

In the command Button with the caption "Quit" add the following line

Unload Me

Press **F5.** Save the changes**.**

You have successfully created and executed the first sample program using Visual Basic 6.

Explanation: The line **Unload Me** in the code window of the quit buttons unloads the form.

The line Text1. text= “This is Green” assigns the value **This is Green** to the Textbox control when you click on the button.

1.4. Variables in Visual Basic

What is a Variable?

The various values used during computation are stored in what are called ‘variables’.

Declaring Variables

Variables in VB are declared using the DIM statement and the following syntax:

Dim variable name [**As** type]

For example:

Dim Total Bill Amount [**As** integer]

A variable has to have a name. There are some naming conventions or rules of nomenclature.

The name of a variable:

Must begin with an alphabet;

Must not have an embedded period or a special character;

Must not exceed 255 characters.

Must be unique within the same scope.

Data Types

Variables have a name and data type. The data type of a variable determine how the bits/ bytes representing those values are stored in the computer’s memory.

The following table will give you an idea about variables and their purposes.

Integer	: A numeric variable, holds numeric values -32,768 to 32, 767.
Long (Long Integer)	: A Numeric Variable-holder a wider of integers than integer.-2,147,483,648 to 2, 147, 483, 647.
Single	: A numeric variable which holds numbers with decimal places. -3.402823E38 to –1.401298E-45 for negative values. -1.401298E-45 to3.402823E38 For positive values.
Double	: a numeric variable with a wider range than single. 1.79769313486232E308 to - 4.94065645841247E-324 for negative values; 4.94065645841247E-324 –to- 1.79769313486232E308.
Currency	: For holding monetary values. -922,337,203,685,477. 5808 to 922,337,203,685, 477,5807.

String : For holding text or string values.
0 to approximately 2 billion for variable length.
1 to approximately 65,400 for fixed length.

Byte : A numeric variable, holding less than the value 255, 0 to 255.

Boolean : For holding True or False values.

Date : For holding date values inclusive of and between January 1,100 to December31, 9999.

Object : For holding references to objects in Visual Basic and other applications. Any object reference.

User-defined (using Type) : Number required by elements. The range of each element is the same as the range of its data type.

Variant : A general-purpose variable that can hold most other types of variables values (with number). Any numeric value up to the range of Double. With character values, it has the same range as for variable-length string.

Let us look at the various data types one by one.

Integer : This data type is used to store whole numbers, and cannot be used in calculations where decimals or fractions are involved. They can store reasonably large numbers. The Integer data type occupies only two bytes of memory and is quite fast when used in calculations.

Long : This is the big brother of the Integer data type. It can hold much larger values, as you can see in the table. It occupies twice as much space as the Integer. It must be used only where the calculations involve large numbers and is much slower than the integer.

Single : This is the equivalent of the Floating-Point number. It can store fractions and provide precision to a fairly high level. It occupies 4 bytes of memory space and should be used where very high precision is not a must. For example if you want the value of say, 400000000 raised to power 50, then you may not have the exact figure.

Double : This solves the problem of precision that the Single data type lacks.

If occupies 8 bytes of memory space and should be used in applications where the requirement of precision is very high. This is not advisable for regular commercial applications as it can be fairly slow compared to the Integer data type and should be used when you want accuracy in calculations involving figures beyond the fourth decimal point. Foreign Exchange dealers will tell you the importance of this accuracy.

Currency : This data type is used for holding values related to items rates, payroll details and other financial functions. However, this data type should not be used if you need extreme accuracy beyond the fourth decimal point as for example, if you are working on Foreign Exchange details or interest rates for very large values.

Byte : This data type can hold values from 0 to 255. it cannot hold negative number or numbers larger than 255. Assigning negative values beyond 255 will result in a runtime overflow error. The Byte data type occupies only one byte of memory.

Boolean : This data type accepts only True or False values. Since the default value for all numeric data types is zero, the default value for a Boolean data type is also zero. Zero value is interpreted as False and non-Zero value is interpreted as True. The VB keywords **True** and **False** can be used to assign values to the Boolean data type.

Date : This variable holds date and time data. It can hold time from January 1 100 to December 31. 9999, and time from 00.00.00 (midnight) to 23.59.59 (one second before midnight) in one second increments. It occupies 8 bytes of memory. The data is displayed as per the settings in your computer. You can store it in British format or American format, or any other format that is available or the Regional Settings on your control panel.

When other numeric data types are converted to Data, values to the left of the decimal represent date information, while values to the right of the decimal represent time. Midnight is 0, and midday is 0.5. Negative whole numbers represent dates before December 30, 1899.

The string data type: Probably the most commonly used data type is the string. Every application has details like name, Address, Zip code, Phone number etc.

Object data type

In Visual Basic, forms, controls, procedures and recordsets, are all considered as Objects.

A variable declared as on Object is one that can subsequently be assigned (using the set statement) to refer to any actual object recognized by the application.

```
Dim objDb  As database
Set objDb =  openDatabase  (“c:\SISI\EIS.mdb”)
```

A variable declared as an object occupies 4 bytes of storage)

The variant data type:

A variant data type is a variable that can change its type freely. It can accept text, numeric data or byte data easily without any hiccups. If you don’t supply a data type, the variable is given the Variant data type by default. Visual Basic automatically performs any necessary conversion.

```
Dim VarValue                ‘Variant by default.
VarValue                              ‘ VarValue contains “100” (a string).
VarValue = VarValue-70      ‘ VarValue now contains the numeric
                                      value 30  The ‘ conversion is done
                                      automatically
```

The variant data type so special is that it can contain values that the other variables cannot handle. These values are

The Null value

The Empty value

The Error value

The Null Value

Null is commonly used in database applications to indicate unknown or missing data.

The Error Value

In a Variant, **Error** is a special value used to indicate that an error condition has occurred in a procedure.

The Empty Value

A Variant variable has the **Empty** value before it is assigned a value. The **Empty** value is a special value different from 0, a zero-length string (“”), or the Null value. You can test for the **Empty** value with the **IsEmpty** function:

If IsEmpty (X) Then…………

A variant can be assigned the **Empty** value using the **Empty** keyword.

When a Variant contains the **Empty** value, you can use it in expressions, where it is treated as either 0 or a zero-length string, depending on the expression.

The Scope of a Variable

The scope of a variable is the range from which the variable can be referenced- a procedure, a form, and so on. The value of a variable in one procedure cannot be accessed from another procedure. The value of a variable is local to that procedure.

Variable declared using the dim keyword exists only as long as the procedure is executing. There is one variation though, in the form of the **Static** variable. Values in local variables declared with **Static** exist the entire time your application is running. Variables are declared as static using the **‘Static’** keyword.

Static intCounter As Integer

Variable declared using the **Static** keyword exist as long as the application is running and are usually used to update counters.

Module Level Variables

Variables declared as module-level variables will be available to all procedures within that module. However they will not be available to procedures in other modules. A module-level variable is declared using the **Private** keyword in the declaration section of the module.

Private intCount as integer.

The declaration must be made in the declaration section of the module.

In order to make the variable available to all other modules, use the **Public** keyword. The declaration must be made in the declaration section of the module. Public variables cannot be declared in procedure. They can be declared only on the declaration section of a module

Public intTemp As Integer

The following table will give you an idea about the scope of a variable vis-à-vis the method of declaration.

Scope	Private	Public
Procedure-level	Variable are private to the procedure in which they appear.	Not applicable. You cannot declare public variables within a procedure
Module level	Variables are private to the module in which they appear	Variables are available to all modules.

Constants

Constants store values like variables, but as the name implies, those values remain constant throughout the execution of an application. There are a number of built-in constants in Visual Basic.

Creating Your Own Constants

The syntax for declaring a constant is:

[Public / Private] Const constant name [As type]= expression

Constant name should be a valid name

As type is the data type.

expression is the numeric or string value that has to be assigned to the constant.

The naming rules are the same as those for creating variable names.

The following piece of code will help calculate the circumference of Circle A.

```
Const conPi = 3. 1415926
Dim IntRad     = Val (Textl. Text)
CircumCircle A = 2 * ConPi * IntRad
```

Scope of a Constant

By declaring a Constant in the declarations section of form, standard, or class module, rather than within a procedure, the Constant will be available to all the procedures in the module.

By declaring a Constant using the **Public** keyword, it is available throughout the application .

Declaring a constant in a procedure will be available to that procedure only.

Circular References

Constants can be defined with reference to other constants. Consider the following;

Public Const conA = ConB* 1.414

Public Const conB = ConA * 2

Since both the constants are available throughout the application. Visual Basic will generate an error. This method of defining constants where each is defined in terms of the other is called *Circular Reference***.**

Converting Data Types

Visual Basic provides functions to convert values into data types. The following table lists these functions.

Conversion function	**Converts an expression to**
Cbool	Boolean
Cbyte	Byte
Ccur	Currency
Cdate	Date
CDbl	Double
Cint	Integer
CLng	Long
CSng	Single
CStr	String
Cvar	Variant
CVErr	Error

Arrays, How do you define them?

An array is a set of similar items. All items in an array have the same and are identified by an index. Arrays allow you to refer to a series of variable by the same name and to use a number (an index) to tell them apart.

Syntax

Dim Varname [([subscripts])] as [New] type [,varname….]

Dim nums (10) as integer.

Dim x (10 to 20) as integer.

Declaring Fixed- Size Arrays

The scope of the array will depend upon the method of declaration.

1. To create a local array, use the Private statement in a procedure to declare the array.

DIM Counters (10) As integer

2. To create a module-level array, use the Private statement in the Declarations section of a module to declare the array

Private counters (10) As integer

3. To create a public array, use the Public statement in the Declarations section of a Form.

Public counters (10) As Integer

Multi-dimensional arrays

Arrays can have more than one dimension. A table of data will be represented by multidimensional array. You would declare the array as follows

Dim Saleval (11,2) As integer.

Dynamic Arrays

Dynamic Array are used when you do not know the number of elements for an array. A dynamic array can be resized at any time and this helps you to manage memory efficiently. Can use a large array for a short time and then free memory to thesystemwhen you are no longer using the array. The size of the array after having declared a smaller array.

The **ReDim** is used in conjunction with the **Dim** statement while declaring these arrays.

Declaring a Dynamic Array

1.You declare the array as dynamic by giving it an empty dimension list.

Dim DynSaleval( )

2.Use the Redim statemen to allocate the actual number of elements.

ReDim DynSaleval (11,4)

The Preserve Keyword

Whenever the ReDim statement is used the previous array and its contents are destroyed. Visual Basic resets the values to the Empty value (for variant arrays), to zero (for numeric arrays). To a zero-length string (for string arrays), or to Nothing (for arrays of objects). This is useful when you want to prepare the array for new data, or when you want to shrink the size of the array to take up minimal memory. However, if you need to expand or increase the size of he array **ReDim** by itself is not good news. In order to allow the array to ‘grow’ the **Preserve** keyword is used. The statement

ReDim Preserve Saleval (12,9)

Will not destroy the data that has already been entered. It will only add the value for the second dimension.

In the case of a single dimension dynamic array, you can enlarge an array by one element without losing the values of the exiting elements using the **UBound** function to refer to the upper bound. The **Ubound** function can be used to get the ubound of the array.

Review Questions:

1. List out the features of VB?
2. What is error value?
3. Explain the uses of project Explorer?
4. Discuss about scope of variable?
5. How will you develop an application in VB?
6. Explain in detail about VB controls.
7. What are the types of toolbars available in VB?
8. What are the advantages of VB programming?
9. What is the usage of various windows available in VB IDE?
10. With suitable example, explain the creation of an application in VB.
11. Explain the different data types used in VB with suitable examples.
12. Which controls does the value as “Value” possess?
13. State two examples for custom controls.
14. What is a variable?
15. How are variables declared and assigned values?
16. How do you put or create the control on the Form?

NOTES

……………………………………………………………………………………..
……………………………………………………………………………………..
……………………………………………………………………………………..
……………………………………………………………………………………..
……………………………………………………………………………………..
……………………………………………………………………………………..
……………………………………………………………………………………..
……………………………………………………………………………………..
……………………………………………………………………………………..
……………………………………………………………………………………..
……………………………………………………………………………………..
……………………………………………………………………………………..
……………………………………………………………………………………..
……………………………………………………………………………………..
……………………………………………………………………………………..
……………………………………………………………………………………..
……………………………………………………………………………………..
……………………………………………………………………………………..
……………………………………………………………………………………..
……………………………………………………………………………………..
……………………………………………………………………………………..
……………………………………………………………………………………..
……………………………………………………………………………………..
……………………………………………………………………………………..
……………………………………………………………………………………..
……………………………………………………………………………………..
……………………………………………………………………………………..
……………………………………………………………………………………..
……………………………………………………………………………………..
……………………………………………………………………………………..
……………………………………………………………………………………..
……………………………………………………………………………………..
……………………………………………………………………………………..

UNIT-II

Objectives:

This objective of this unit are to understand the code window, Use the various functions in Visual Basic, Understand the various control structures used in Visual Basic, Understand which control structure can be used for a given situation, Write programs using control structures. Next this unit gives the controls provided by Visual Basic for file management, the various types of files, File management functions, opening and closing files, Reading and writing data to files.

Contents:

2.1 Writing Code in visual Basic:-

The code window

VBA is the name of the programming language used in Visual Basic. It is the language used for the other desktop products from the Microsoft stable, and has over a period of time added many features to make the job of programmers (composers) as easy as possible.

Visual Basic provides you the code window to compose (view, write, edit) all your VBA code. You can have as many code windows, as there are modules and cut and paste code between them. The code window has probably all the features (including a certain amount of intuition) that a programmer will hope for.

Opening the code Window

The right mouse button on the control. Choose the “View Code” option.

OR press F7 after clicking on the control.

OR Click on View in the menu bar and click view code.

OR right click the name of the form from the Project Explored and select view code.

The code window will open on the default event of that control. For example if the Form has the focus, the code window will open on the Form’s Load event.

Part of the Code Window

Project Name

The name of the project to which this procedure belongs is displayed in the title bar. The project is the outermost container of all procedures for an application.

Module Name

Next in the hierarchy after the Project is the Module. A project can contain more than one module. (A form is also a module). Therefore, while editing the code it would be very helpful to know the name of the module whose code is being edited.

Object Box

This displays all the object associated with that form and the form itself.

Procedures / Events Box

This will list all the events recognized by Visual Basic for the control or form displayed in the object box.

Split Bar

Use the Split bar to view different part of the code. Each part can be independently scrolled at the same time (remember the Norton Editor?). Click on ‘Windows’ on the menu bar. From the drop down menu select ‘Split’. This will split the code window into two horizontal panes. This way you can scroll up or down the code for two different procedures, compare, cut and paste the code from one pane to another. The object box will display the names of the procedure that has the focus, preventing confusion about the procedure to which a piece of code may belong.

Double click on the bar to close the pane or move the split bar to the bottom of the code window.

Margin Indicator Bar

The gray area on the left side of the code window is the margin indicator. You will find this now empty area come alive when you debug your program.

Procedure View Icon

Displays the selected procedure. Only one procedure at a time is displayed in the code window.

Full Module View Icon

Displays the entire code in the module.

The Procedure Separator

It is a horizontal gray line that separates two procedures. It can be turned on or off.

The Anatomy of a Procedure

The procedure is the piece of code that gets executed when the control that it is associated with senses an event. The event as mouse click, mouse move, or a method invoked by the code.

A procedure consists of all or most of the following.

1. **Name** Every procedure must have a name. The name of a procedure is usually tied to the control. You can have a procedure called. cmdExit_Click( ). This means that this procedure will be executed

against the ‘Click’ event of the ‘cmdExit’button. The ‘Sub’ before the name means that the procedure is a Subroutine.

2. **Declaration area** Strictly speaking, there is no clear demarcated area where one should make all the declarations for variables, constants, etc. However it is considered a good idea to declare all the variable at the beginning of the procedure. This makes it easy to located variables and verify the same during the process of debugging.
3. **Statements** The procedure deals with basic execution. A procedure will contain VBA statements (or code if you like) that will perform the intended task. A procedure can have as many statements as you care to add, but keeping your procedures short and simple will render them easy to debug.
4. **Call to other procedures and / or functions** One way of reducing the number of lines in a procedure is to break up the entire activity into smaller functions or procedures that can be called as and when required. Therefore, if a particular task has to be performed by more than one event, that task can be written as a procedure to a function and called as and when required. The principle used here is “Write once-use many times”. A call to another procedure or function is not a must for every procedure.
5. **Terminator** All good things come to an end. Every procedure (subroutine in this case) is terminated by the “End Sub” statement.

Subroutine or Function

Subroutines and **Functions** are both procedures that are executed as a unit of code. The difference is that **Function** will return a value to the calling program, and can therefore be used as a variable in order to return a value. On the other hand, a procedure will not return any value to the calling program.

The function is declared with the **‘Function’** key word and terminated with the **‘End Function’** statement.

Example

```
Function sum (intn1 an integer, intn2 as integer) as integer
        sum = intn1 + intn2.
End function
```

The functions ‘sum’ can be called anywhere in the program. It can be called by simply passing the variables to it as arguments.

```
Dim intnum1 as integer, intnum2 as integer
Dim intans as Integer
intans = sum (intnum1, intnum2)
```

Automatic Word Completion

Visual Basic provides you with the Auto Word Completion option. To enter a ginger strainer like ‘Mouse Button Constants’, enter the alphabet ‘m’ and press Ctrl+Spacebar. A drop-down list box will appear with a scrollable list of keywords starting with ‘m’. If you type in enough of the word to narrow the selection to the word you want, press Tab or Spacebar. The rest of the word is completed by Visual Basic.

Auto List Members

This feature is similar to the Auto Word completion. In this case, as soon as you enter the name of a Visual Basic object that has been created on the Form, followed by the period, a drop-down list box appears with all the properties and methods of that object.

In order to assign a caption to a Label Box, enter ‘lblWarning’. A drop down listbox appears with all the properties and methods associated with the Lable Box object. From the list, select the property that you want (caption in this case), and press spacebar. The word is completed by Visual Basic.

Color Cueing

This is an important feature that makes the programmers’ job easy. As a standard, statements with errors appear in red, comments in green, Visual Basic keywords in blue and Identifiers like SQL in black.

Line Continuation Character

This character is the underscore “-“ and comes in handy when you have long statements to make. The Statement that you may have seen so far are very simple and short. However, while working on SQL statements or Windows API declarations, you will find they are very long and will not fit on one line.

The line continuation character makes the reading of statements easy too.

```
Set rst = dbs. Open Recordset (“SELECT Productcode, “-
        & “ProductName FROM Product;”)
```

Commenting and Uncommenting Statements

Debugging code is usually done procedure by procedure. While debugging your code, you may need to stop a particular procedure from executing while another r procedure executes, as the helps to isolate a problem. One way to accomplish this is to delete the offending procedure. The other option is to comment the entire block. Visual Basic provides the ‘Comment Block’ button on the View |Toolbar| Edit. Select the block and click on the ‘Comment block’ button, whereby the entire block will turn green.

You can uncomment the block by clicking on the ‘Uncomment block’ button.

Visual Basic provides you the necessary control structures in order to develop your applications. In a program you may want to

perform an action if a condition is true.

perform an action repeatedly a certain number of times.

perform an action till a certain condition becomes true.

perform an action as long as a condition remains true.

perform different actions for different values of a variable.

The Control Structures in Visual Basic are

1. If condition Then [statements] [Else statements]

   which conditionally execute a group of statements, depending on the value of an expression.
2. For counter = start to end [Step step] [Statements] … Next which repeats a group of statement a specified number of times.
3. Do[{while| Until} condition] … Loop

   Which repeats a block of statements while a condition is True or until a condition becomes True.
4. While condition [statements] Wend

   Which executes a series of statements as long as a given condition is True.
5. Select Case test expression [Case expression list-n [statements –n]]… End select.

   Which executes one of several groups of statements, depending on the value of an expression.

The For …Next Statement

This structure is used when you want to execute a statement or a block of statements a certain number of times.

Example-1

```
For intI  = 1 to 10
  intTotal = intTotal + intI
  Next intI
```

This loop will compute the sum of the numbers from 1 to 10.

The statements intTotal = intTotal + intI is performed 10 times.

Example-2

```
For intI = 5 to 100 Step 5
            debug . print intoI
      Next intI
```

will print all the numbers from 5 to 100 that are divisible by 5.

The Decision Maker … If

The If conditions …. statement is used when the program has to perform an instruction or a block of instructions depending upon the value of an expression. If the expression returns True then a set of statement(s) is executed. The program may or may not execute any statement(s) if the expression. If the expression returns True, then a set of statement(s) is executed. The program may or may not execute any statement(s) if the expression returns False.

Syntax

```
If condition Then the condition  must return a True or False
        [statements]
[ElseIf condition –n Then the condition must return a True or False
        [elseifstatements]…
[Else
        [elsestatements]]
End If
```

Example 1

```
If string1 = string2 then MsgBox Both the strings are equal.
```

Example 2

```
If CandiateAge <= 18 then
        Juniors = juniors +1
else if candidateAge <= 50 then
        Seniors = Seniors +1
else
        Oldies = Oldies +1
End if
```

When the If … statement returns a true value, the statements following the THEN keyword will be executed. In the first example, when the If… does not return true value, no action is taken.

How does it work

1. The first IF statement evaluates the expression.
2. If an expression returns a True value the statement after THEN keyword executed.
3. If the expression returns a False value, the rest of the ElseIf conditions (if any) evaluated by turn.
4. If any expression returns a True value, the statements following the related THEN keyword are executed.
5. If there are no Else If or Else conditions program execution continues with the statements following the End If statement.

The Loop

```
Do [{While | Until} condtion]
[statements]
[Exit Do]
        Loop
```

Example code

```
intI =1
Do While intI < 10
```

```
        intsum = intsum + intI
        intI = intI +1
    loop
```

The While Loop

While condition [statements] Wend

This structure executes a series of statements as long as a given condition is true.

Syntax

```
    While condition
        [statements]
    wend
```

Example Code

```
    Dim intI as Integer
        intI =1
    While intI < 10
        intsum = intsum +1
        intI = intI +1
    wend
```

Select Case … End Select

This method is used when the program has to execute one of several groups of statements, depending on the value of an expression.

Syntax

```
    Select Case testexprssion
        [Case expressionlist-n
            [statements-n]] …
        [Case Else
            [elsestatements]]
    End Select
```

Example Code

```
Select Case CandidateAge
    Case <= 18
        MsgBox you are in the Juniors
    Case <= 50
        MsgBox you are in the Seniors
    Case Else
        MsgBox You are in the Oldies
   End Select
```

2.2. Working with Files

All the data that you create, manipulate etc is stored as a file. Even databases are stored as files, such as Biblio.mdb, Nwind.mdb etc.

You will have to transfer data from a file to your database, or write data from your database to file. You may have to transfer data from the Microsoft platform to another platform or vice-versa.

File handling are in two views. First, we will look at the controls that Visual Basic has provided to help us navigate through the drives and directories of our computer. Having found the file that we want, we will look at opening, reading, writing and closing that file. We will also look at the types of files that are used and how we can perform various functions on these files.

Visual Basic File System Controls

Moving from one drive to another	:	The Drive ListBox
Moving from one directory to another	:	DirListBox
List file(s) in a directory	:	FileListBox

The DirveListBox Control

This is a drop-down list box that will display the list of drives on your computer. It gets all information from the Operating System and allows the user to select a drive of his choice. Selecting a particular drive records a change in the **Drive** property of the **DriveListBox**.

The DirListBox Control]

A drop-down list box that displays a hierarchical list of directories in the current drive.

The FileListBox

Displays all files in the current Directory or Folder. Allows users to set up search criteria for files.

Visual Basic has built in functions such as

ChDrive	:	Changes the current logged drive
ChDir	:	Changes the default directory
MkDir	:	Creates a new directory
RmDir	:	Deletes a directory
Name	:	Renames a file
Kill	:	Deletes a file
File Copy	:	Copies source file to destination
File Date Time	:	Returns the date and time when the file was modified
GetAttr	:	Returns the attributes of a file as an Integer value
SetAttr	:	Sets the attributes of a file.

Ch Drive : Change the current logged drive.

Syntax

ChDrive drive

- drive is a string, which specifies an existing drive
- if the drive does not exist, the drive will not change.

Example

ChDrive "A"

This will change the drive to "A"

ChDir : Changes the current directory or folder.

Syntex

ChDir Path

Example

ChDir "C:\AZAM\EMSQUARE"

This will change the current directory to 'C:\AZAM\EMSQUARE'

MKDir : Creates a new directory or folder.

Syntax

MkDir path

- path is a string that identifies the directory to be created.
- path may include the drive name. If drive is not specified, then new directory is created in the current drive.

Example

MkDir"c:\EMSQUARE\Expermnt"

This will create a directory 'expermnt' under 'c:\EMSQUARE'

RmDir : Deletes a directory

Removes an existing directory or folder.

Syntax

RmDir path

- path is a string that identifies the directory or folder to be removed.
- path may include the drive.

Example

RmDir "C:\Temp\Temp1"

Name : Renames a disk file, directory, or folder.

Syntax

Name oldname as newname

- oldname is the name of the file that has to be renamed or moved
- newname is the string that identifies the new name that the file should have.

- If the new name contains the name of a different directory then the file is moved to the new directory.
- new name cannot already exist.
- The file being renamed should not be open.

Example

Name OldFile As NewFile *OldFile is renamed as NewFile

Kill : Deletes a file or files

Syntax

Kill pathname

- pathname is the name of the file(s) to be deleted
- Kill accepts wildcard characters like "*" and "?" to delete a group of files.
- Kill will not delete an open file.

Example

Kill "testdoc" 'to delete the file 'testdoc'.

Kill "*.Doc" to delete all the '*.Doc" files in the current directory.

FileCopy : copies the source file to the specified destination.

FileCopy source, destination

- Source is a string that identifies the name of the file to be copied.
- destination is a string that identifies the target file name.
- FileCopy should not be used to copy an open file.

Example

Filecopy Maxfile Highfile.

FileDate Time : Returns a Variant (Date) that indicates the date and time when a file was created or last modified.

Syntax

FileDateTime (pathname)

pathname is the name of the file.

Example

Dated = File Date Time ("Maxfile")

*This will return the date and time when the file 'Maxfile' was last modified.

GetAttr : Returns the attributes of a file as a Integer value

Syntax

GetAttr (pathname)

*pathname is string that identifies the file.

Example

```
Dim Fileattrib As Integer
Fileatrib = GetAttr(Maxfile)
```

SetAttr: Sets the attributes of a file.

Syntax

SetAttr pathname, attribuites

- pathname is a string that identifies a file.
- attributes, is a numeric expression or constant that specifies file attribute.

Example

```
SetAttr Highfile, vbHidden + vbSystem
```

The method of access of a file is the basis of classification. From the programming point of view files are classified as

- Sequential access files
- Random access files
- Binary access files

Working with Files

What is a Record?

A file can have records of the same size or of different sizes. You can therefore have a fixed-length record or a variable-length record. In the illustration below, we have the same file displayed as a 'variable-length' record file.

```
RR234, Raw Rice IR50,17.00,KG,1400
RR317,Raw Rice Poni,20.00,KG,2050
WTPP12,Wheat Punj1,12.00,KG,130
SUO16,SunFOil V1,45.00,KG,700
```

In this case, each record is of different size and is separated by a comma. Do not be under the impression that a file will contain headings that will be self-explanatory. The programming involved in reading data from these two types of files is different.

While working with any type of file you will need to

Open a file

Read the file

Close the file

Determine the end of file

Determine the length of file

Determine the beginning of file.

Opening a Sequential File

The file can be read character by character, or by lines, or as a block of characters.

Complete Syntax

Open pathname for mode [Access] [look] As [#] file number [Len=reclength]

For the sake of this function we will look at the syntax for opening a file for reading follows

Syntax

Open pathname form Input As [#] file number

- ❖ Open is the command to open the file.
- ❖ Pathname is the full pathname of the file to be opened.
- ❖ Input specifies that the file must be opened for reading purposes.
- ❖ file number is the number assigned to the file for purposes of identification
- ❖ the file number must be unique. The range is from 1 to 511.

Example:

Open c:\EMSQUARE\Item Master.txt for Input As #1

Closing a File

A file must be closed for the operating system to write the data from the associated buffer to the file. The Close command ensures that this done.

Syntax

Close [file number List]

- ❖ Close is the command.
- ❖ file number List is the list of the file number that represent the files that have to be closed.
- ❖ The file number will be returned to a list that is displayed by the function 'Free File'.
- ❖ If file has been opened for writing, then the output buffer is written to buffer allotted for hat file.

Reading a File

The **Open** command creates a buffer in the memory which helps in I/O operations. The buffer is block of memory that is reserved for I/O operation on the file. The **Read** operation transfers the contents of the file from the disk to the buffer. The '**Input**' function provided by Visual Basic reads a sequential file's contents into the buffer. The **Input** function has three options or methods of work.

1. Input

Syntax

Input (number, [#] file number)

- ❖ Input is the command name.
- ❖ number is the number of characters to be read.
- ❖ filenumber is the number of the open file that has to be read.

Example

string 1= Input (25, # 1)

2. Input #

Syntax

Input # file number, varlist

Where * file number is the file number of the file to be read.

* varlits is the list of variable that will hold the data.

Example:

Input # 1, Serial Num, Name, Designation

3. Line Input #

This version of the Input command reads an entire line into a variable. It will continue to read the contents of the file till it encounters a Carriage Return-Linefeed sequence.

Syntax

Line Input # file number, databuffer

- ❖ Line means read the entire line.
- ❖ file number is the file number of the file to be read.
- ❖ databuffer is the variable that will hold the line that has been read.

Example:

Line Input # 1, linebuffer

Review Questions:

1. What are the types of file system controls?
2. Write the difference between file and record.
3. Explain types of files.
4. Explain the uses of picture box with its properties.
5. Explain For..Next statement, decision maker _If, While, Select case?
6. Design your form and write code for simple arithmetic calculator.
7. How will you give name to the text box?
8. Write a note on the For..Next loop.
9. Explain with example, about the Exit for statement.
10. Explain the steps followed while adding a text box.
11. What is a record? Explain with example.
12. Write a program to append ten strings into a file.

NOTES

UNIT-III

Objectives:

The objective of this unit are to Appreciate interface-related issues, List the items in the menu editor, Create menus for your applications, Change the structure or order of menu items, Add code to menu items. Next it gives you to understand the meaning of MDI forms, Understand the features of MDI forms, Work with multiple MDI child forms, Create a small application using MDI forms.

This unit also discusses When and how debugging is to be done, Setting the Error Trap, Writing an Error-Handling Routine, The Immediate, Watch and Debug windows, The common Dialog Control, Why databases are used to store data, Database terminology, Using the Visual Data Manager, Creating tables, Modifying tables, Working with the Data Control.

Contents:

3.1. MENUS

Building the User Interface. The First Step:

Design the form that you want the user to see on paper. I know that computers are meant to reduce paper work but nevertheless go ahead. The same applies to coding as well.

Overcrowding

Do not clutter up the form. Let there be enough of white or blank space. When a user sees too many buttons, textboxes, label boxes, etc, he is going to feel that the program is too confusing or 'high-funda'.

Important Information Must be Given Prominence

Not all information about your customer or the items that you sell is equally important. The customer name and address are important while generating an invoice, but details about his spouse or pets may not be relevant, unless you have promotional scheme of some sort. In such situations, the extra information can be provided to the user if he pushes a button such as "More Info".

Consistency

Visual Basic offers a host of controls. However, do not make the mistake of using all of them. A number of controls can be used to do the same job. The List Box, ComboBox and TextBox can be used to display a list of names. Do not use different controls at different places, unless you are compelled by the application. For a particular task use the same control throughout.

The Fonts

The fonts are another area where the designer might run away with his imagination. Stick to the standard fonts like Times New Roman, System, Arial, Courier, etc or supply the fonts along with the software.

Consistency Across Forms and the Application

The forms too must look related. The color of forms and placement of controls on individual forms must be uniform. You cannot have the save button on the right hand bottom corner of one form and in the middle on another form.

Affordances

Affordances are visual clues to the purpose of an object. The mouse that you use is shaped in such a way that you know it must be held under the palm.

Simplicity

Keep it simple son! (KISS). Yes. Make sure that the interface you present to the user is simple and easy to use.

Usability

The user must be able to use your application with ease. He is not concerned with the technical brilliance or the brevity of your code. He may also not care about the version of Visual Basic used to develop the application. He would not like to unlearn all that he has learnt so far.

Images

A picture is worth a thousand words. Use context sensitive help. Use the status bars to display a message or use balloon help. Check out how Microsoft products offer this type of help. It is not a bad idea to use the icons that they do.

Colors

Use simple colors. The user is going to be looking at the form that you have designed. If it has a riot of colors and your name is not Rembrandt, it will not be well received. While programming for an international audience, make sure that the colors used are soft and mild.

Interacting With the User

In order to stop the user from pressing the panic button, or condemning your application, you should build in messages that will convey the problem at hand to the user and accept his choice under the circumstances, before you take action.

All about Menus

Menus contain a number of options, logically organized and easily accessible by the user. Toolbars supplement the menus by providing quick access to a number of frequently used options.

Visual Basic provides a very simple method to create menus for your application. You create Menus using the Menu Editor.

In the figure below you have the Standard Menu Bar.

File Edit View Format Window Help

The Menu System

The menu bar consists of a number of options. Window-based applications follow the standard of a File menu on the left, then optional menus such as Edit and Tools, followed by Help on the right.

When the user clicks a menu option, a list of options is displayed. Clicking on any item on the list will generate a click event. You can write a program to respond to that event. An option on the list can have its own submenu. A menu item can have four child menus.A menu is tied to a form. You cannot have a menu without a form.

A menu item can

1. Have a check mark to provide an on and off settings.
2. Have an arrow on the right side to indicate a child menu.
3. Be disabled or enabled depending upon the situation.
4. Have keyboard equivalents for easy access.
5. Have separator line to logically group menu items.
6. Have an ellipsis to indicate a dialog box.

Menu Conventions

You must first decide on the Main Options and sub-options for each of the main options. The placement of submenus is very important. Users expect to find Copy, Cut and Paste beneath the Edit menu; moving them to the File menu would be confusing at best. Don't deviate from the established guidelines unless you have a good reason to do so. Each of the Menu Items that you create will be treated like controls, and can be coded to trigger related activities.

The Menu Editor

Start a new project. With the form in focus, click on Tools | Menu Editor. This will display the Menu Editor.

The Menu Editor has the following Items:

1. The Caption Box: What you type here is what the user sees.
2. The Name Box: The name entered here is the control name for the menu item. You enter the code for menu item under this control name.
3. Shortcut Box: You enter the shortcut keys for the menu items here. Click on the down arrow, a drop-down list box will be displayed with all the shortcut keys. Click on the one you want to use.
4. The Checked Check Box: Click on this if you want to display a tick mark to show that a menu item has been selected. The default is off. This can be turned On or OFF at run time.

5. Enabled Check Box: Click on this to make a menu item enabled. If a menu item is enabled, it will respond to the mouse click. This can be turned On or OFF at run time.
6. The Visible Check Box : If it is set to Off then the menu item and its sub-menus will not be visible. This can be turned On or Off at run time.
7. The Tex Window: This gives a preview of what you have entered. You can also view the hierarchy of menus and sub-menus by looking at the indentations.
8. The Left Arrow button brings the menu item one level up.
9. Clicking on the Right Arrow moves the menu item one level deeper.
10. The up Arrow interchanges the current line with the line above.
11. The Down Arrow interchanges the current line with the line below.
12. Index : Allows you to assign a numeric value that determines the control's position within a control array. This position isn't related to the screen position.
13. Help contextID: Allows you to assign a unique numeric value for the context ID. This value is used to find the appropriate Help topic in the Help file identified by the Help File property.
14. Negotiate Position : Allows you to select the menu's Negotiate Position property. This property determines whether and how the menu appears in a container form .
15. Next: Moves selection to the next line.
16. Insert : Inserts a line in the list box above the currently selected line.
17. Delete : Deletes the currently selected line.
18. OK : Closes the Menu Editor and applies all changes to the last form you selected.
19. Cancel : Closes the Menu Editor and cancels all changes.

Adding the ToolBar

The ToolBar must hold relevant icons that represent frequently accessed functions. For example, the icon with Scissors represents the action 'Cut', where you cut a block of selected text. The icon with a floppy represents the action 'Save'.

ToolBar Conventions

A ToolBar is a supplement to the Menu Bar. It offers mouse shortcuts for frequently used functions.

It can be used without a menu bar only if the options on the menu are few. Trying to provide buttons for all functions of Microsoft Word would be an unnecessary exercise.

Let us now create a ToolBar

- ❖ You need the ToolBar control. If you do not have it on your ToolBox, do the following.
- ❖ Place the mouse on the ToolBox and click the right mouse button.
- ❖ From the pop-up menu choose components.
- ❖ From the pop-up menu choose components.
- ❖ A window pops up displaying the controls that are available.
- ❖ Click on Microsoft Windows common controls
- ❖ Click on Wang image edit control.

Push the OK button and you will see some new controls on your ToolBox.

3-2 Multiple Document Interface Applications

The MDI was designed for applications like Microsoft Word and Microsoft Excel, which need to show more than one document (multiple documents) at the same time.

An MDI application allows the user to display multiple documents at the same time, with each document displayed in its own window. Documents or child windows are contained in a parent window.

Why MDI Forms?

1. An MDI Form acts like a container for the other forms in the application. Therefore the user can start an application, minimize it and close it with a single point control. One need not close all the constituent forms.
2. Most of the control buttons and code for the various forms can be shared. After all functions like adding, displaying, sorting records, etc. are commonly used by most of the forms.
3. Reduce the number of controls. Common controls can be available with the MDI form and shared by the other forms.
4. All activities like day-end operations and updating of records can be handled from the MDI form.

An MDI Application consists of

1. One MDI Parent form
2. One or more MDI Child Form(s)
3. Optionally independent forms and modules.

Creating a simple MDI application

Start a new Standard. Exe project

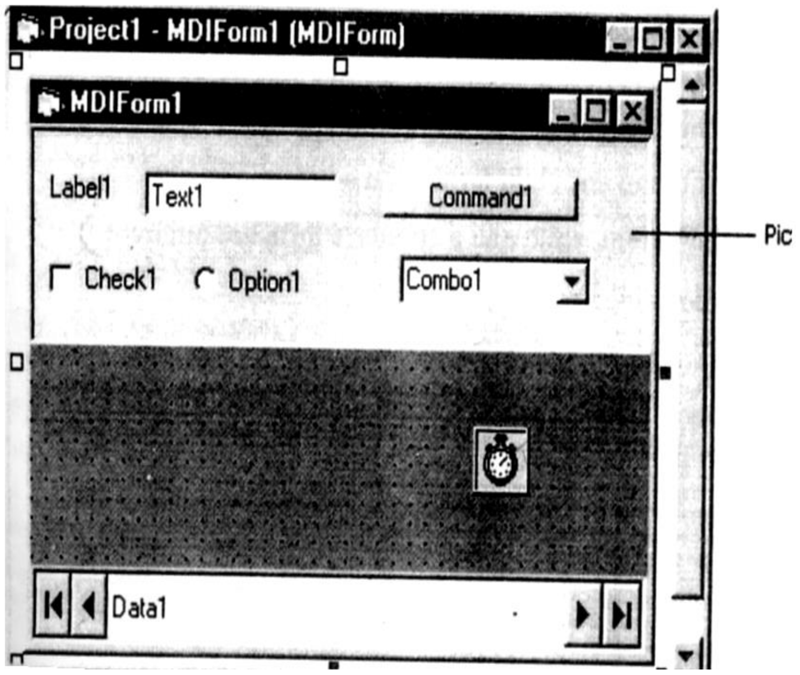
Click on Projects

From the drop-down menu, Click on ‘Add MDI Form’

A form with a darker color will be displayed. This is the MDI Form.

1. An application can have only one MDI form. If a project already has an MDI form, the Add MDI Form command on the Project menu will be unavailable.

2. It can contain only those controls that have an ‘Align’ property or those controls that are not visible during runtime. This is because the internal area of an MDI form is defined by the area not covered by these controls. These are Toolbar, Picture Box, and Data control. The Align property here is related to the alignment of the control on the form.



3. To place other controls on an MDI Form, you can draw a picture box on the form, and then draw other controls inside the picture box. You can use the Print method to display text in a picture box on an MDI For, but not to display text on the MDI Form itself.
4. An MDI form object can’t be modal.
5. The menu for the Child form will be displayed as the menu on the MDI Form when the Child Form has the focus.
6. You can access the collection of controls on an MDI Form using the controls collection. For example, in order to hide all the controls on an MDI Form you can use code similar to the following.

```
For Each Control in MDIForm1. Controls
    Control. Visible =  False
    Next control.
```

7. The default Auto Show Children property displays the child forms automatically when they are loaded.
8. The default Scrollbars property will display the scroll bars when the child forms extend beyond the boundaries of the parent form.
9. The count property of the MDI form tells you the number of controls in the controls collection.

To create an MDI child Form, select Form 1 (or add anew form) and set its MDI child Property to true.

When you double-click on a Word document on your desktop, the Word program is automatically invoked. The document is the child form, and the Word program, is the MDI parent form. When you load a child form, its parent form (the MDI form) is automatically loaded and displayed. When you load the MDI form, however its children are not automatically loaded.

You can set the AutoShowChildren property as False, in order to load MDI child window as hidden, and leave them hidden until you display them using the show method.

MDI application may open, save and close several child forms in one session. The functions and the methods we can use to specify the active child form or control, load and unload MDI and child forms, and maintain state information for a child form.

This property returns the form that is the active window in the application. If a control in the child form has the focus, it returns the name of that form.

Syntax

Object. ActiveForm

Note : An active dialog box is not the same as an active form.

In the click event of the MDI Form add the following

ActiveForm.Text1.Text = "This is to test if we have got it right".

At the bottom of the pop-up menu, a list of open documents will be displayed with a tick mark against the current document.

Add a menu item Windows to your MDI Form.

Check the box 'Window List'.

The size of the MDI Child Form (with a sizeable border) is determined by Windows, based on the size of the Parent. Therefore the size and position of the child Form at runtime will be different from its size at design time.

Open the menu editor again. To the menu item Window, add a sub-option. Call it Arrange. In the click event of the Arrange menu item, add the following line.

```
Private Sub arrangemethod_Click( )
MDIForm1. Arrange vbCascade
End Sub
```

Now run the application. You will be able to cascade the windows. The MDI Forms' 'Arrange' method has four options. They are

vbCascade	: Arranges open forms such that the title bars of forms behind the current form are visible.
vbVertical	: Open forms are displayed vertically.
vbHorizontal	: Open forms are displayed horizontally

vbArrangeIcons	: Icons of minimized forms are neatly lined up at the bottom of the MDI form.

3.3 Debugging Tips

Debugging is an integral part of software development.

1. How to avoid bugs
2. How to trap and weed out bugs that have crept in anyway
3. How to handle the bugs that could not be trapped

The usage of the term ‘bug’ has many versions. Whenever a problem occurred in a system, it was called a bug. The process of finding out and removing the error is called debugging. Therefore the programmer (that means you) has to carry out the debugging process before distributing the software to the end-user.

1. When your application doesn’t produce correct results.
2. When your application halts unexpectedly.
3. When your application goes into an infinite loop.

The errors that can occur in your program are classified into three categories. They are

1. Errors of Syntax
2. Logical Errors
3. Runtime Errors.

“DIM intI Integer”

The moment you press the Enter button, a message box will pop-up and tell you that you made a mistake.

Click on the Tool and choose options from the menu. Under the ‘Editor’ tab you will find the following items checked. They are code settings and wind settings.

Code Settings

1. Auto Syntax check
2. Require variable declaration
3. Auto list members
4. Auto quick info
5. Auto data tips.

Window Settings

1. Drag-and-Drop Text Editing
2. Default to Full module View
3. Procedure Separator.

```
DIM intMaxmarks as Integer
Maxmarks = marks + Maxmarks
Debug . print  Maxmrks
```

The Auto Syntax Check option will not trap this error. However, the option ‘Require variable Declaration’ will ensure that wrongly spelt variables will not spoil the show.

When we entered the statement “DIM intI as”, a drop-down list box appears with all the possible options we need. This will reduce the time taken to type and also minimize the chances of spelling errors.

This option will provide you with the complete syntax for the function that you intend to write. It is context sensitive and it is not necessary to remember the syntax of every function by rote.

We will look at this function when we take up debugging. With such options at your disposal, you can be sure that syntax errors can be minimized.

1. Plan your program. Make sure you know where your program should begin, the path it should take and where it should end. Draw Data-flow diagrams, flow-charts, etc., such that your logic is clear. Then begin to write your code.
2. Use a standard naming convention. You can have your own naming convention, but it must be maintained throughout the application.
3. The names used for the controls, variables, etc must be meaningful. Remember that there is no constraint on the Visual Basic programmer today as there was for programmers some years ago.
4. Declare the variables at the beginning, although you can declare them anywhere in the procedure.
5. Comment your code. Wherever necessary, add your comments. Start with a narration about the purpose of the procedure and to what it is related.

A run-time error results when a statement attempts an invalid operation. Logical errors occur when there is a flaw in the program logic.

Logical errors have to be trapped by going over the program very carefully and supplying data of any type. Runtime errors are caused by hardware faults, change is in the settings, etc.

The tools and methods include setting watches, setting breakpoints, stepping through the code using the locals window, the immediate window, the Stack, etc.

Traditionally the programmer fraternity uses the message box to display the state of values that have been assigned to a variable.

Look at the code below

```
Private Sub Command1_Click( )
    Dim intI As Integer
    For intI = 1 To 20 step 4
    MsgBox intI
    ‘Other statements…..
    Next intI
```

```
        End Sub.
```

A routine like this will keep the programmer informed about the value of the variable intI.

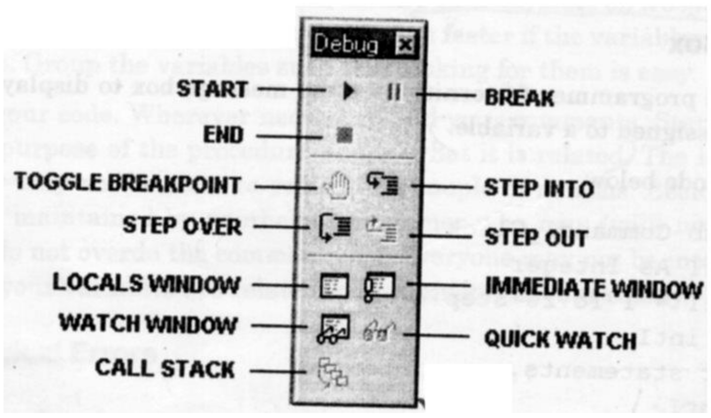
The other method is to use the Debug object. The debug object will send the output to the immediate window.

```
Private Sub Command1_Click( )
        Dim intI As Integer
        For intI = 1 To 20 Step 4
                Debug. print intI
        Next intI
End Sub
```

The values assigned to the variable ‘intI’ will be displayed in the ‘Immediate Window’.

The Debug Toolbar

In order to begin debugging you need the ‘Debug Toolbar’. The options on the Debug toolbar are available as sub-options of the Debug Menu item. However it is more convenient to have the Debug toolbar on top. To bring up the Debug toolbar, right-click the toolbar and select Debug from the pop-up menu.



The Start Button:

You start the program by clicking on this button. It is like pressing the F5 button and choosing the Run option.

The Break Button:

It pauses the program. It is like pressing the Ctrl +Break key.

The End Button:

Halts the program. The next sets of buttons come into play only if we set the Break point.

The Breakpoint:

Debugging is the art of weeding out a bug. A time-tested method is to first isolate the problem. You can specify the point where the program should halt by setting ‘Breakpoint’. Click on the gray margin area on the left side of your code window in line with the statement where you want to pause the execution. A brown dot will appear in the margin. Now run the program. The program will pause before it reaches the statement that has been marked as the Breakpoint.

Toggle Breakpoints:

You can have more than one breakpoint, Click on this button to shift the breakpoints.

Step Into:

Now that the program has paused where you want it to, you can step through the code line by line. Click on the Step Into icon to execute one line of code. You can watch what happens to the variables in the procedure.

Step Over:

The procedure you are watching may branch to another procedure. If you are confident that the called procedure is perfect, you need not cover that ground again. Click on the Step .over icon. The called procedure will be executed, and control will be returned to the next line in the current procedure.

Step Out:

If you have seen all that you need to see in the current procedure, click on Step Out icon.

The next set of buttons deals with the following Windows.

- ❖ Locals Window
- ❖ Immediate Window
- ❖ Watch Window
- ❖ Quick Watch
- ❖ Call Stack

The Locals Window:

Click on the Locals Window icon. A Window like the one in the figure will be displayed. This Window will display all the variables that are declared in the current procedure.

The Quick Watch Window:

Right click on a variable whose value you want to watch. From the pop-up menu select ‘Add Watch’. The following dialog box will pop-up. Enter the name of the variable for confirmation. Now run the application after setting a breakpoint. Observe the value of the variable change as you step through the code. You will be able to note when a variable assumes a certain value and take appropriate action.

The Immediate Window:

This is the window to which the ‘Debug-print’ sends the output. In this window you can directly execute a statement as if you are directly communicating with the interpreter.

The Watch Window:

In this window all the variables used in the application till the breakpoint are displayed. The Watch window also displays the name of the procedure where the variables is used and the value that variables assume.

What is an Error Handler?

It is a routine that traps errors and directs the user or the program to perform a certain task to overcome the error.

The err object

The VBA engine is always on the lookout for errors that may occur in your application (as if the Project Manager is not enough). Whenever an error occurs, VBA activates the err object. The err object has the following properties and methods.

The Clear Method : The err.clear clears the properties of the error object.

The Raise Method: Helps raise an user-defined error.

- ❖ Description property
- ❖ Help Context property
- ❖ Help File property
- ❖ Number property
- ❖ Last DLLError property.

3.4 The Common Dialog Control

The Windows Common Dialog Control allows you to deploy the dialog boxes that it provides with all its applications. The Common Dialog control lets you display the following dialog boxes.

- ❖ Open a File
- ❖ Save a File
- ❖ Set a Color
- ❖ Set a Font
- ❖ Print a Document

The **Common Dialog** control can display the following dialogs, using the specified methods.

Method	**Dialog Displayed**
ShowOpen	Show Open Dialog Box
ShowSave	Show Save As Dialog Box
ShowColor	Show Color Dialog Box
ShowFont	Show Font Dialog Box
ShowPrinter	Show Print or Print Options Dialog Box
ShowHelp	Invokes the Windows Help Engine.

Working with the Common Dialog Control

Add the Common Dialog Control to your toolbox by Right clicking on the toolbox and selecting ‘Components’ from the pop-up menu. Next select ‘Microsoft Common Dialog Control 6.0 from the list of components.

Draw it on your form. During runtime it is not visible. Right click on the common dialog control and select Properties from the pop-up menu. You can set the properties for each of the methods.

Small Program

Start a new project. Give it an appropriate name.

Add the Microsoft Common Dialog Control to your toolbox.

Draw the following controls on the form

One textbox control

Five Command Button controls (Open, Save, Color, Print and Exit.).

Write some simple code to allow the user to open a file and display its contents in the textbox. We therefore need to use the Show Open method. Add the following line in the code window of the open command Button commonDialog1. ShowOpen.

There are two ways you can give a title to the dialog box.

By setting the title at design time

By setting the title at runtime through code.

The user to pen a file of his choice and display its contents. However, the user may also want to create a new file, add some text and save it in a directory of his choice.

The ShowSave method to allow the user to enter the name of the file in which the text entered in the textbox has to be saved.

```
Dim intfreefilenum As integer
Dim intFileLength As Integer
Dim strdatabuffer As String
On Error GoTo cancel
CommonDialog1. DialogTitle = “Enter the name of the file to save”
CommonDialog1.ShowSave
```

Changing the color

The Common Dialog control also provides the means to change the color of the text or other controls on the form. In order to display the Color Dialog Box you must use the ShowColor method. Double Clikc on the ‘color’ button on your form to bring up the code window. Add the following line.

```
CommonDialog1. ShowColor
```

Printing a document

The printer dialog box is a great help when you are building applications for clients or for the other user departments. To the Print button's code add the following line.

CommonDialog1. ShowPrinter

Rich TextBox Control

The Rich TextbBox control allows the user to enter and edit text with more advanced formatting features than the conventional TextBox control.

The Rich TextBox control provides a number of properties you can use to apply formatting to a selected portion of text within the control. Select the portion of the text to be formatted and apply the necessary changes. You can make text bold or italic, change the color, and create superscripts and subscripts. You can also adjust paragraph formatting by setting both left and right indents, as well as hanging indents.

3.5 Introduction to Databases

A database is a collection of records that can be manipulated with ease. The conceptual view will map into a physical view of the database. The physical view of the database will map into the way the data is actually stored. This method of separating the physical storage from the user view will permit data independence. The views of all the users will be used to develop the conceptual view of the data and any change in the user's view will be accommodated by the conceptual view.

What is a database?

A database can be defined as a collection of record stored in tables. It has a set of rules and tools to manage these records. There are different types of databases, each with its own format.

The most commonly known database types are

- Relational Model
- Network Model
- Hierarchical Model

Tables

A table is a basic repository in which data is stored, and has a specific structure for storing data. It is made up of one or more than one column. The data is stored in the form of Rows and Columns.

Rows

A record is one row in a table. The row will span across all the columns of the table, and each row has one full set of information about one 'subject'.

Columns

Each row has four columns or attributes, and every time an attribute of a record is added a column will be added. An attribute is referred to as a Field, and the Column type will be decided based on the type of data to be stored.

Why Relational Databases?

Data redundancy can be avoided if you create tables in the following form.

Supplier Table

Supplier Code

Supplier Name

Address 1 (Door number & Street)

Address 2 (Road, Area)

Address 3 (City, State and Zip)

Item Table

Item Code

Item Name

Description

Price

Supplier Item table

Supplier Code

Item Code

The Supplier Table has a relationship with the Supplier_Item Table through the Suppler Code. Likewise, the item Table has a relationship with the Supplier_Item Table through the Item Code.

The Primary Key

A primary key is a unique filed or a combination of more that one field that identifies a record.

In the above example, the Supplier Code is a primary key, since no two records have the same serial number. The Supplier Code can be the unique fields that identify the supplier in the supplier table.

Index

An index is a list that contain the key field of the record and (a pointer to) its physical location in the database. Therefore the database engine can located a record quickly.

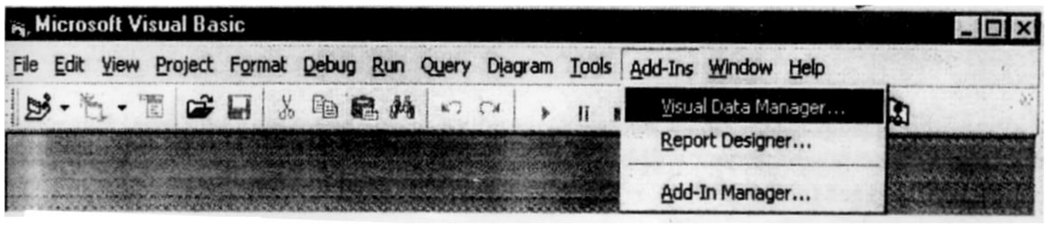
Creating a Table

You can create your own tables,

- ❖ Using DAO (data Access Objects)
- ❖ Using Microsoft Access
- ❖ Using Visual Data Manager that comes with Visual Basic.

Working with Visual Data Manger

The Visual Data Manager option is available under the Add Ins Menu.


Microsoft Visual Basic
File Edit View Project Format Debug Run Query Diagram Tools Add-Ins Window Help
Visual Data Manager...
Report Designer...
Add-In Manager...

Click on the Visual Data Manager. Visual basic responds by showing you the VisData window.

The menu items available are File, Utility, Window and Help

There are NINE buttons too. Drag your mouse over the buttons and take a look at the balloon help.

From the File Menu select the option New.

From the pop-up menu select Access. Depending upon the version of Visual basic that is installed on your machine, you will be asked to choose the version of MDB. Select the larger number (The bigger the better?)

Choose Access 2.0 if your application is going to work on Window 3. X Access 7.0 is for the 32-bit platform only. Visual Basic will now preset you with a Save Dialog Box. Enter the name of the file for your database.

Now Visual Basic will display two windows, the Database window and the SQL statement window. The Database windows will have an item Properties click on the box with + to see the properties that have been set or available for this database.

Creating a table

Right click the mouse in the database window. From the pop-up menu select new Tables. This will result in another window being displayed with a number of text boxes, checkboxes and a rabbit. If you do not see a rabbit, it means you are following instructions. Enter the name of the Table that you want to create.

Click on the AddFields button

1. In the textbox for Name, enter EmpNo.
2. Select the Type. From the drop-down list box, select Integer.
3. Notice that length is fixed as 2. Some of the other options are made unavailable.
4. Click on Required. This means that the EmpNo cannot hold a Null value
5. Now click on OK.

The AddField Dialog Box is now waiting for the next field to be entered. Add the next two fields too.

Modifying tables

Now you have created a table with some fields. There is a saying that the only thing that is constant is change!

Modifying the Name of a field

You can modify the name of a field by editing it in the Table structure dialog box. The other properties that can be modified are

Ordinal Position

Validation Rule

Default Value

Allow Zero Length

Required

Copying a table

You can copy an entire table with the data and structure to another database. In the database window, right-click on the table that you want to copy. From the pop-up menu choose ‘Copy Structure’. Enter the destination database where you want to copy the table. You can copy the data as well by checking the ‘Copy Data’ check box.

3.6 Working with the Data Control

A database can be created easily with the help of the Visual Data Manager. The tables can be populated by simply entering the data in the respective fields. Next we need to create an application to display the data from the database, and allow the users to add new records or modify existing data.

To created such an application we have to

1. Establish a connection with the database.
2. Extract the fields from the relevant tables.
3. Display them on the form
4. Accept changes made to the data on the form and update the database.

The Data control

1. Establishes a connection to a database.
2. Returns a set of records from the database.
3. Enables you to move from record to record
4. Enables you to display and manipulate data from the records in bound controls.

How does the Data control Work?

In order for the Data control to deliver the goods, two properties need to be set. These properties can be set at runtime or design time. The properties are

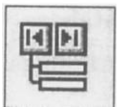
Database Name: This specifies the name of the database that must be opened.

Record Source: This specifies the name of the table(s) of the database from which the data has to be extracted.

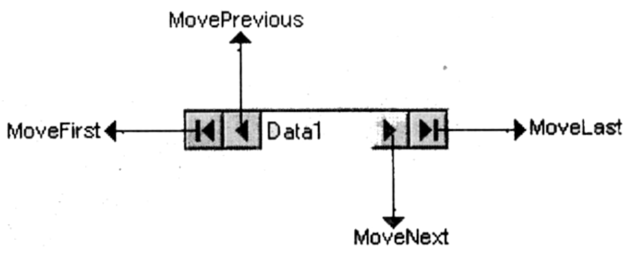
When you run the program after setting these properties. Visual Basic connects to the database specified, and returns a set of records from the table(s) in the form of a Recordset. A Recordset is an object that points to the data in the database.

Using the Data control

The Data control is part of the standard toolbox. It looks like this.



Draw the Data control on the form as you would draw a textbox control. It looks like this.



It has buttons for moving from record to record. You have buttons to move to the next or last record and to move to the previous and first record. You can also set the caption property for the Data control.

Setting the properties for the Data Control

We will take a look at the important properties of the Data control.

BOFAction : Action to be taken when the used reaches the beginning of File.

EOFAction : Here we tell the Data control what to do when we reach the end of File.

Caption : The text to be displayed on the Data control's title bar.

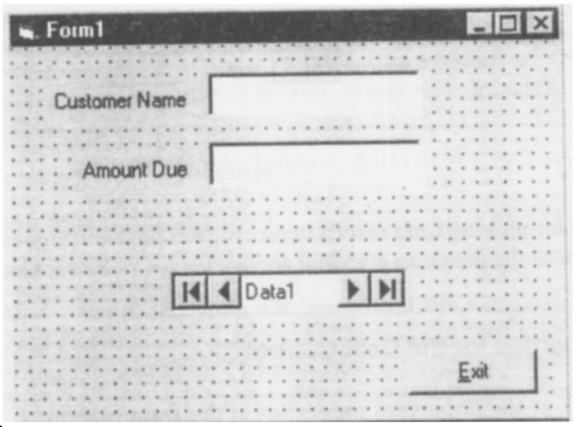
Connect : This tells the Data control the type of database that will be accessed. It could be FoxPro, dBase, Paradox, Access, etc. The default is Access.

DatabaseName : Specifies the name of the database that the Data control will access. Give the name of the file that contains the database. Hard coding the name here is not a wise thing to do. It can be set at runtime.

Recordset Type: This property specifies the type of Recordset object the control will use to access the database.

There are three options: 1. Table 2. Dynaset. 3. Snapshot.

RecordSoruce : The RecordSoruce property specifies the source of the records accessible through bound controls on your form.



Setting the Properties

Click on the Data control and bring up the Properties window

1. Click on Name: Let us call it Customerdata.

   We will be referring to the Data control with this name.
2. Click on BOFAction. Set it on MoveFirst.

   When the Beginning of File is reached, the Data control must be told to point to the first record.
3. Click on Caption. Set it to Customers. This will be the title of the Data control. The default is Data 1.
4. Click on the connect property. Select Access. Access is the default.

   This tells visual basic the type of database that will be used. Visual Basic can work with a number of other databases.
5. Click on database Name Property

   We have to mention the name of the MDB file that must be opened by the Data Control.
6. Click on EOFAction. Set it to MoveLast.

   This tells the Data control to point to the last record when the user reaches the End of File.
7. Click on Recordset Type Property. Select 0-Table type.

   A Recordset Type property determines the type of object that will create by the Data control. The options are 1. Table Type where an editable Recordset is created based on data from a single table. 2. Dynaset, where an editable Recordset is created based on data from one or more tables.

   3. Snapshot, where a non-editable Recordset is created based on data from one or more table.
8. Click on RecofdSource Property. Set it to Customer_Data.

   In this property we tell the Data control the table or query that must be used while creating the Recordset.

Some of the controls can be ‘bound’ to the Data control. Each bound control is bound to one field of the Recordset. They display the fields from the current record of the Recordset.

The following controls are all data-aware and can be bound to a single filed of a Recordset managed by the Data control.

Label

TextBox

CheckBox

Image

OLE

ListBox

Picture

ComboBox.

The following controls are all capable of managing sets of records when bound to a Data control. All of these controls permit several records to be displayed or manipulated at once.

DBList

DBCombo

DBGrid

Binding the Bound controls

We can make a control Data-aware by setting its properties. In order to make the TextBox, Data-aware, we need to bind it to the Data control. This is how we go about it.

Select Text 1. Bring up the Properties window

1. Click on DataSource Property. Set it to Customerdata. This is the name of the Data control. From now on this text box will display a field from the Recordset returned by the Data Control.
2. Click on DataField. A list of fields available will drop down. Select Customer Name.

Enhancing our Program

Let us see how we can add extra functionality to our program.

Add the following to your form.

1. Two LabelBox controls
2. Two TextBox controls
3. Five CommandButtons
4. Some Red Paint.

Setting the properties

For the Data control set the following properties

Name	Customerdata
BOFAction	0- Move First

Caption	Customer Data
Connect	Access
DatabaseName	C:\azam\VB-exercises\invoice.mdb
EOFAction	0-Move Last
Recordset	Type 1- Dynaset
RecordSource	Customer_Data

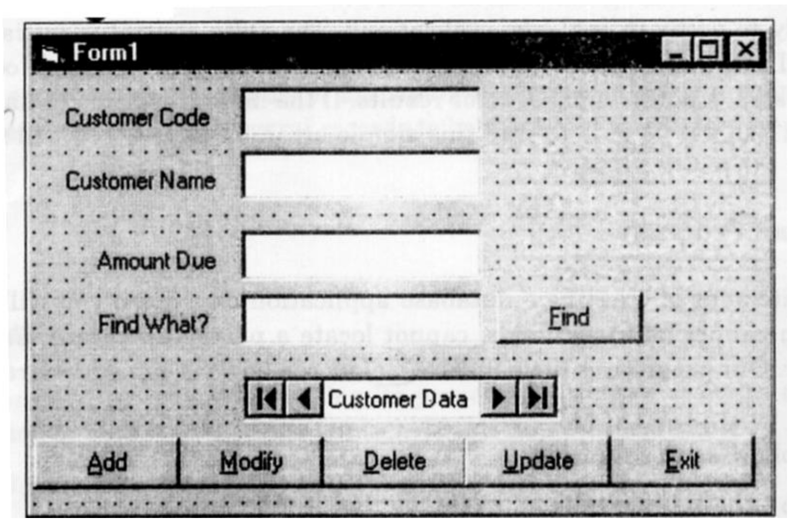
For the TextBox Controls set the following properties

Name	C_Code/C_Name/Amt_Due
DataSource	Customerdata
DataField	Customer_Code/Customer_Name/Amount_Due

For the fourth textbox, change the name to FindTxt.

For the CopmmandButtons give the following captions,

Add, Modify, Delete, Update and Find.



Coding

Adding a New Record

In order to add a new record we must invoke the 'AddNew' method which will create a blank record in the memory. This blank record becomes the current record. After you enter the data, you can add the new record to the database by either of the following methods.

1. Moving to another record
2. Invoking the 'Update' method

Add the following code to the Add button

Private Sub CmdAdd_Click)

Customerdata. Recordset. AddNew.

Deleting a Record

In order to delete a record, we have to invoke the 'delete' method. The current record that is displayed will get deleted. Once this record is deleted, we must move the record pointer to the next record. Your code will look like this,

```
Private Sub CmdDelete_Click)
Customerdata. Recordset. Delete.
```

Updating the Database

The Data control itself handles the task of updating the database with the changes made to the current record, when you move the record pointer to another record.

UpdateRecord and UpdateControls Method

These methods are special to the Data control. The Update Record method allows the users to save the changes to the current record and continue editing.

The Update Controls method does the reverse of the above. This method will be of use when the user wants to roll back the editing changes made to the current record before updating.

Finding a Record

The Find method works only when the Recordset type is a Dynaset or snapshot. If the Recordset type is set to Table type, then you have to use the Seek method.

```
Private Sub CmdFind_Click)
Customerdata. Recordset.FindFirst "Customer_Name = "'& Trim
(Text3.Text) & """ End Sub.
```

The DBGrid Control

The DBGrid control displays fields from a Recordset object as a series of rows and columns, and enables manipulation of this data.

Let us see the DBGrid in action

Start a new project

On the form add the following control

One Data Control

One DBGrid Control

If you do not have the DBGrid control, as part of your ToolBox (which is usual), Right click on the ToolBox. Select Components from the pop-up menu. From the list displayed thereafter, click on DBGrid control. Click on OK and you have it on your ToolBox.

Set the properties for the controls as follows

Data Control

DataSource: C:\VB\Biblio.mdb

RecordSource: Publishers

DBGrid Control

DataSource: Data 1

Now run the program.

Review Questions:

1. What is database?
2. Write any five additional controls in VB.
3. Explain VB built_in functions.
4. Explain the features of MDI.
5. Explain debugging methods?
6. Write in detail about Bound controls?
7. What are the two important properties for menu controls?
8. How will you add a picture box control to the form?
9. How will you add one module to the form?
10. How will you add a menu in an application? Explain.
11. With suitable example, explain:i. Simple Combo box. ii. Drop-down list combo box.
12. Explain in detail about common dialog control.

NOTES

UNIT-IV

Objectives:

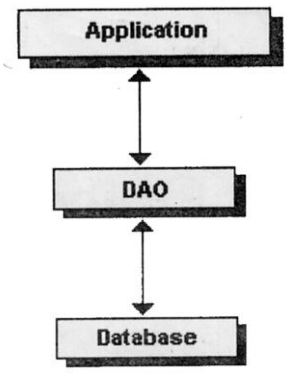
The objective of this unit discusses: The Jet database engine, Function of a database, An introduction to SQL, Working with Data Access Objects. Additional Controls Available in Visual Basic 6.0, Why ADO, What is ADO, The structure and component of ADO, Working with ADO.

Contents:

4.1. Data Access Objects

The Jet Database Engine:

Jet Database engine was the genie doing all the hard work behind Microsoft Access. Jet 1.0 could handle the functions that were expected of a database engine. Apart from the standard DDL, DML, maintenance and security features, it could gather data from different sources at the same time, and introduced the concept of Dynaset. A database engine provides the link between the data and the application. The Jet Database engine could be used with all the Microsoft products, from MS excel to Visual C++. It worked as the data manger in a data access application built using other products like Visual Basic or Visual C++. DAO 1.1 allowed users to use the DDL to control the Jet engine. The next version of Jet implemented a better hierarchical model and allowed the user to control all its capabilities through the DAO.



DAO, Hierarchical Model

From then on, there was no looking back for Jet. It is now a 32-bit implementation and provides.

- Referential integrity
- Cascading updates and deletes
- Replication and synchronization.

- Replication over the Internet.
- The Jet database engine can read and write data stored in other database formats.
- It can read data stored in.
    - Spreadsheets
    - Text files.
    - HTML tables and lists.
    - ODBC database.

Functions of the Jet Database Engine

Microsoft Jet provides the seven basic function of a DBMS:

- Data definition and integrity. It allows the users to create and modify structures for holding data, such as tables and fields. It also ensures that rules are applied to data operations to prevent data corruption from invalid entries or operations.
- Data manipulation: It allows the users to add new data, modify or delete existing data from the system.
- Data storage: It holds data as per the structures and stores it in the file system.
- Data retrieval: It allows the users to retrieve and view the data from the system.
- Database maintenance: compact, repair, and convert data and database objects.
- Security: control users' access to database objects and data, thereby protecting objects and data against unauthorized use.
- Data sharing: It allows more than one user to access the data in a multi-user environment.

Data Definition (DDL) and Data Manipulation (DML) Language

DDL deals with the structure of your data. Data Definition deals with creating tables, adding fields, creating indexes etc. The activity carried out by the Visual Data Manager comes under this. Once the structures are defined they are stored in a single file. The data definition and well as the data is stored in one file. DAO methods are used to create these structures.

DML deals with the content of your database. The Data Manipulation Language deals with allowing the user access to the data in the database. You can create views, or select some of the data through queries. You can access the data using SQL statements, or by using DAO.

The general sentiment is that DAO is most suitable for DDL functions and SQL is suitable, for DML functions.

Storage

The Jet Database is stored as a single file with the. MDB (Microsoft access Database) extension. The .mdb files are stored in the ISAM format. The records are stored in a variable – length format, on pages that are 2k in length. One record cannot span two pages.

Retrieval

Data can be retrieved form the database, either through SQL statements or through DAO. Data retrieval is normally through 'Queries', that are instructions to the database to present the data in a certain format.

Relationships

Jet is an RDMBS, and hence the tables are built in such a way that one table will have a relationship with one or more tables in the database. The data is spread across more than one table in order to reduce the redundancy of data and allow easy access to data.

Multi-user Environment

Jet allows more than one user to access the database. It is a remote multi-user implementation. In this case the data resides on a single server, and the database engine resides on each of the workstations. It cannot handle more than a few dozen users, unlike the other databases like Oracle or SQL server.

SQL:-

What is SQL?

SQL is a database programming language. The earliest version was called Sequel, and we still like to use the same. SQL is an industry-standard database interface, and the knowledge of SQL commands allows you to access and manipulate a wide variety of database products from many different vendors.

The SELECT statement

The SELECT statement fetches the data from the database as a set of records.

The syntax for SELECT must include

The criteria for selection

The source of data from where the data has to be selected.

Example

SELECT * from Titles.

Syntax

Select [predicate] { * | table. * | [table.] field1 [As alias1] [,table.] field2 [As alias 2] [, ….]] }

FROM table expression [, …] {IN external database }

[WHERE…]

[GROUP BY…]

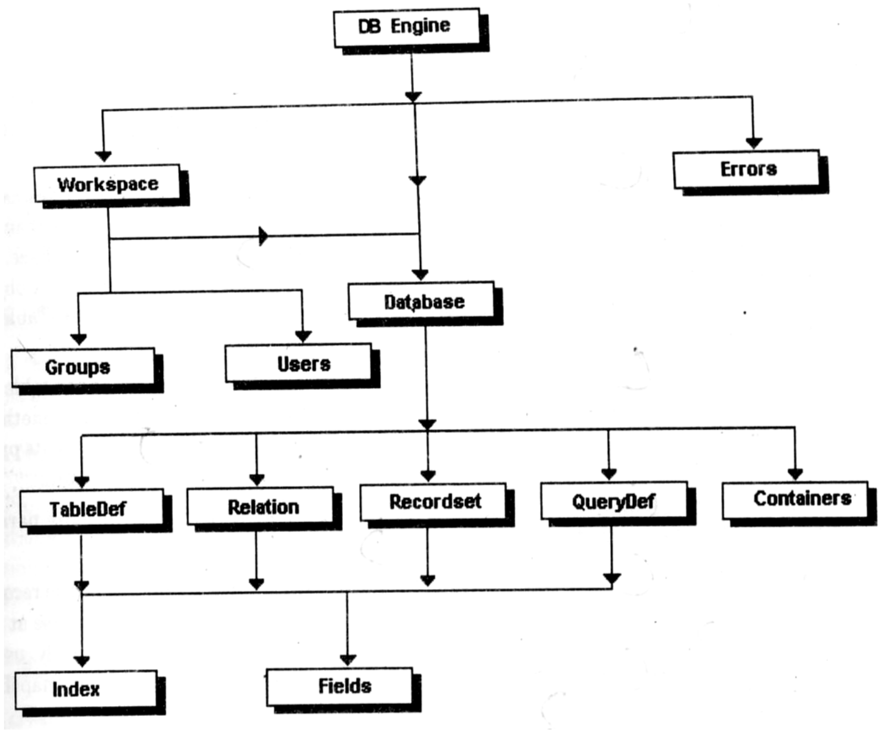
[HAVING…]

[ORDER BY…]

[WITH OWNERACCESS OPTION]

The DAO object Model

The model includes much more than fields and tables, and is illustrated below.



Understanding the DAO Object Model

The DAO object model models the structure of a relational database system. This includes creating databases, defining tables, fields, and indexes, establishing relations between tables, and navigating and querying the database.

There are 15 objects in the DAO object model, as listed here.

1. DBEngine object

This DAO object holds all other objects and maintains engine options. The DBEngine object contains and controls all other objects in the hierarchy of DAO objects.

2. Workspace Object

Defines and manages the current user session. This object contains information on open databases, and provides mechanisms for simultaneous transactions for a user. In a multi-user environment, this makes a lot of sense. Each use has his own workspace. A workspace is a non-persistent object that defines how your application interacts with data-either by using the Microsoft Jet database engine, or IDBCDirect to access external data.

3. Database object

Represents a database with at least one open connection. This can be a Microsoft Jet database or an external data source. You use the Database object and its methods and properties to manipulate an open database.

4. TableDef object

Contains both Field and Index objects to describe database tables. A Table Def is the stored definition of a ‘base’ table or a ‘linked’ table. Through its methods like Open Recordset, you can link various types of Recordset objects.

5. QueryDef Object

Represents a stored SQL query statement with zero or more parameters, maintained in a Microsoft Jet database.

6. Recordset object

A Recordset object represents a set of records in a table, or the records returned by a query. You use Recordset object to manipulate data in a database at a record level. When you use DAO objects, you manipulate data almost entirely, using Recordset objects. DAO has five types of Recordset objects: table Dynaset, snapshot, forward-only, and dynamic.

7. Container object

Represents a particular set of objects in a database for which you can assign permissions in a secure workgroup. In addition to the container objects provided by DAO, an application may define its own container objects such as saved forms, modules, reports, or script macros.

8. Relation object

Represents a relationship between fileds in tables and queries. You can use the Relation object to create, delete, or change the type of relationship, and determine which tables supply the fields that participate, whether to enforce referential integrity, and whether to perform cascading updates and deletes.

9. Filed object

Represents a filed in table, query, index, relation, or Recordset. A filed object represents a column of data with a common data-type and a common set of properties. A filed object contains data, and you can use it to read data from a record or write data to a record.

10. Index object

Represents an index on a table in the database.

11. Parameter object

Represents a value associated with a Query Def object. Query parameters can be input, output, or both.

12. Document object

Contains information about individual objects in the database (such as tables, queries, or relationships).

13. User object

Represents a user account with particular access permissions.

14. Group object

Represents a group of user accounts that have common access permissions in a particular workspace.

15. Error object

Contains information about an error that occurred during a DAO operation. Any operation involving DAO can generate one or more errors. For example, a call to an ODBC server might result in an error from the database server, an error from. ODBC, and a DAO error. As each such error occurs, an Error object is placed in the Errors collection of the DBE Engine object. A single event can therefore result in several Error objects appearing in the Errors collection.

4.2 Additional Controls Available in Visual Basic 6.0

SSTab Control :

The SSTab Control provides an easy way of presenting several dialogs or screens of information on a single form, using the same interface seen in many commercial Microsoft Windows applications. Only one tab is active in the control at a time, displaying the controls it contains to the user, while hiding the controls in the other tabs.

The Tabbed Dialog control provides a group of tabs, each of which acts as a container for other controls.

For each of the ‘tabs’ you can set properties, add other controls and write code necessary. The properties are set using the property Pages.

Working with SSTab Control

Start a new project.

The SSTab control is not part of the standard controls on the Toolbox. To add the SSTab controls, carry out the following instructions.

Right click on the toolbox.

Select Components form the pop-up menu.

From the components select Microsoft Tabbed Dialog Control 6.0.

You will see this control on you Toolbox.

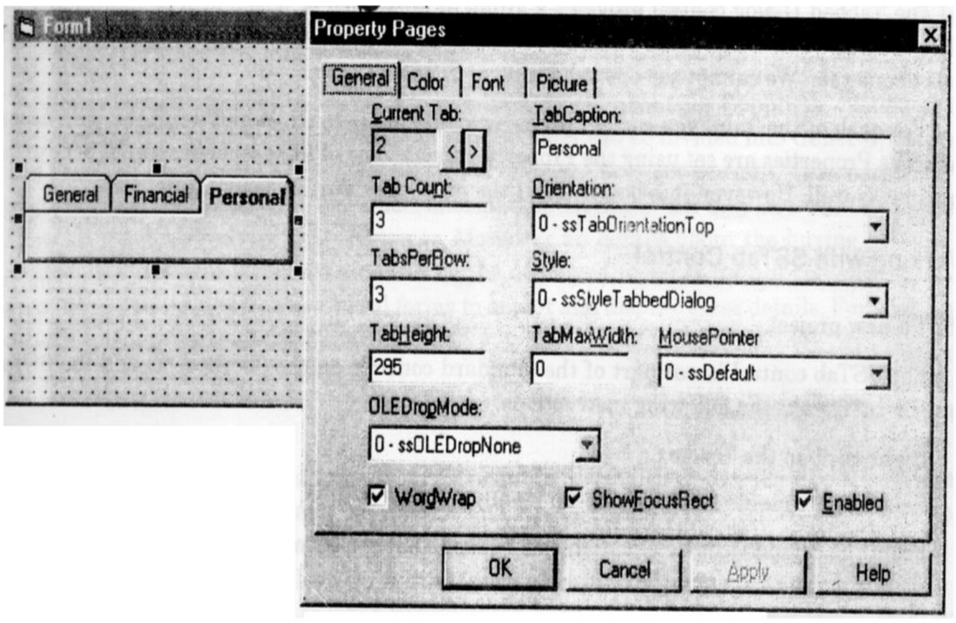
Draw the control on your Form. It will appear like this.

New we need to set the Properties for each of the Tabs. We need to,

Add controls to each of the tabs.

Change the caption for each of the tabs.

Write code for the controls where necessary.



Current Tab: 2
TabCaption: Personal
Tab Count: 3
Orientation: 0 - ssTabOrientationTop
TabsPerRow: 3
Style: 0 - ssStyleTabbedDialog
TabHeight: 295
TabMaxWidth: 0
MousePointer: 0 - ssDefault
OLEDropMode: 0 - ssOLEDropNone
☑ WordWrap ☑ ShowFocusRect ☑ Enabled

Give the captions for the tabs as follows

Tab0: General

Tab1: Financial

Tab2: Personal

At run time, the user can navigate through the tabs by either clicking on them, by pressing CTRL + TAB, or by using mnemonics defined in the caption of each tab.

Adding Controls to tabs

Add controls to a particular tab's 'client area', just as you add controls to a from.

The Image List control

The Image List control acts like a repository of images for the other controls.

An Image List control contains a collection of images that can be used by other windows common controls-specifically the List View, Tree View, Tab Strip, and Toolbar controls. The List Image object has the standard collection object properties: Key and Index. It also has standard methods, such as Add, Remove, and Clear. However, once the ImageList has been associated with another control you cannot delete or insert images in the ListImages collection.

Adding images to the Image List

The ImageList control contains the ListImages collection of ListImage objects, each of which can be referred to by its Index of Key property value. To add an image to a control at design time, use the ImageList control's Property pages dialog box.

To add ListImage objects at design time.

Right-click the ImageList control and click Properties to bring up the Property pages.

Click Insert Picture to display the Select Picture dialog box.

Use the dialog box to find either bitmap or icon files, and click Open.

Click on the Key box and enter a string that will uniquely identify that image. This string can be used to refer to the image that has been added to the ImageLists collection.

Optional, Assign a Tag property setting by clicking in the Tag box and typing a string. The Tag property doesn't have to be unique.

Add as many images as you need for + he project.

TabStrip Control

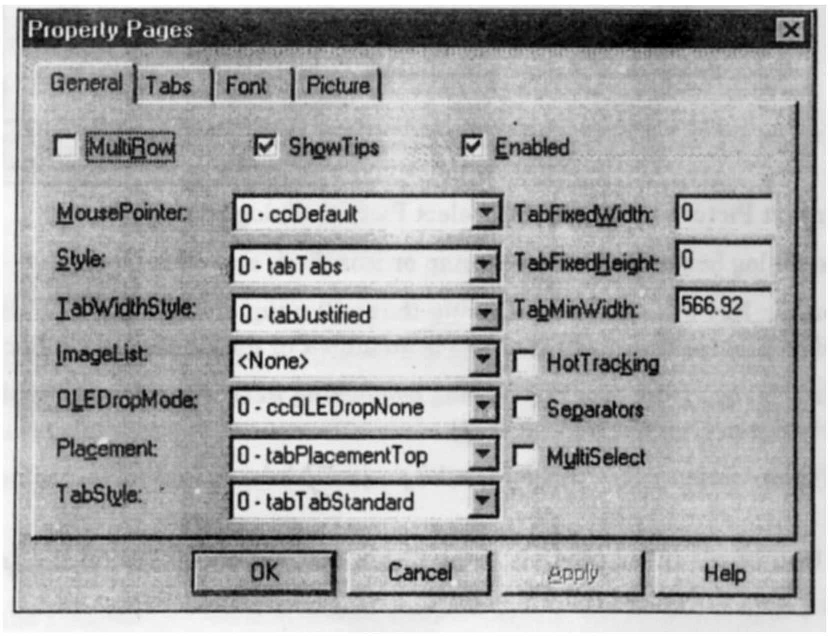
The function of a TabStrip control is very similar to that of the SSTab.

The control consists of one or more Tab objects in a Tabs collection.

Creating Tabs at Design Time of Run Time

To create Tab objects at design time.

1. Right-click the TabStrip control and click Properties to display the Property Pages dialog box.
2. Click Tabs to display the Tabs page and make the changes.



☐ MultiRow ☑ ShowTips ☑ Enabled

MousePointer:	0 - ccDefault	TabFixedWidth:	0
Style:	0 - tabTabs	TabFixedHeight:	0
TabWidthStyle:	0 - tabJustified	TabMinWidth:	566.92
ImageList:	<None>	☐ HotTracking	
OLEDropMode:	0 - ccOLEDropNone	☐ Separators	
Placement:	0 - tabPlacementTop	☐ MultiSelect	
TabStyle:	0 - tabTabStandard		

MSFlexGrld contol

The MSFlexGrid control displays and operates on data in a table from. The Flex Grid is designed to only display the data and not allow the user to enter data in it. However, the user can sort, merge, and format tables containing strings and pictures.

Tool Bar Control

The Tool Bar control is another control used to enhance the user interface. This control allows us to create a toolbar for our application. A toolbar contains buttons that provide a graphic interface for the user to access an application's most frequently used functions and commands. These are actually shortcuts to items in an application's menu.

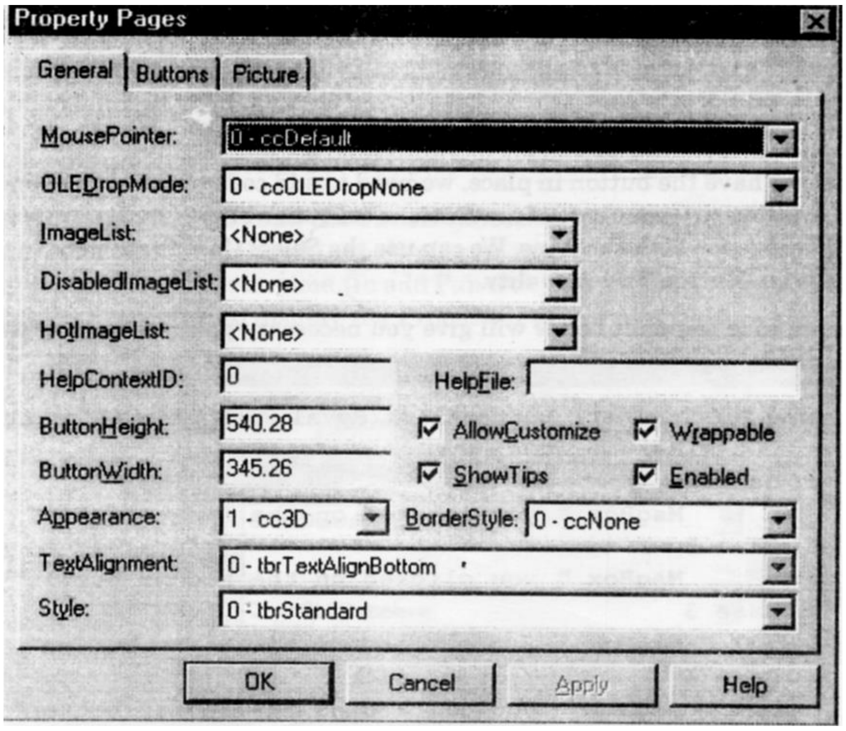
The ToolBar control is part of the Microsoft Windows Common Controls 6.0 library.

Creating a ToolBar

A ToolBar normally contains buttons with or without images/icons that represent a particular activity. A button with the image of a pair of scissors will represent an activity like cutting.

The ToolBar control consists of one of one or more Button objects n a Buttons collection. At both design time and run time, you can create Button objects. Each button can have an image, a caption, a ToolTip, or three.

1. Right-click on the Toolbar control and click Properties to display the ToolBar Property Pages.
2. Click the Buttons tab to display the dialog box shown.
3. Click Insert Button to insert a new Button object.
4. Set appropriate properties, such as Key, Caption, Image, and ToolTipText.
5. Set the Button object's Style properly by clicking the style, box and selecting a style.



MousePointer: 0 - ccDefault
OLEDropMode: 0 - ccOLEDropNone
ImageList: <None>
DisabledImageList: <None>
HotImageList: <None>
HelpContextID: 0
HelpFile:
ButtonHeight: 540.28
ButtonWidth: 345.26
☑ AllowCustomize ☑ Wrappable
☑ ShowTips ☑ Enabled
Appearance: 1 - cc3D
BorderStyle: 0 - ccNone
TextAlignment: 0 - tbrTextAlignBottom
Style: 0 - tbrStandard

The Status Bar Control

At the bottom of the screen a more explanatory line on that button is displayed, and at the bottom of the screen you will also observe that it displays the status of the Caps Lock, Scroll Lock, NUM lock, etc. This line is called the Status Bar. It also displays other information like the line and column number where your cursor is positioned, and the page number.

The Status Bar control can hold up to a maximum of sixteen panels or frames. Each of these panels can give different types of information like the time, micro help for a button, the status of a key, etc.

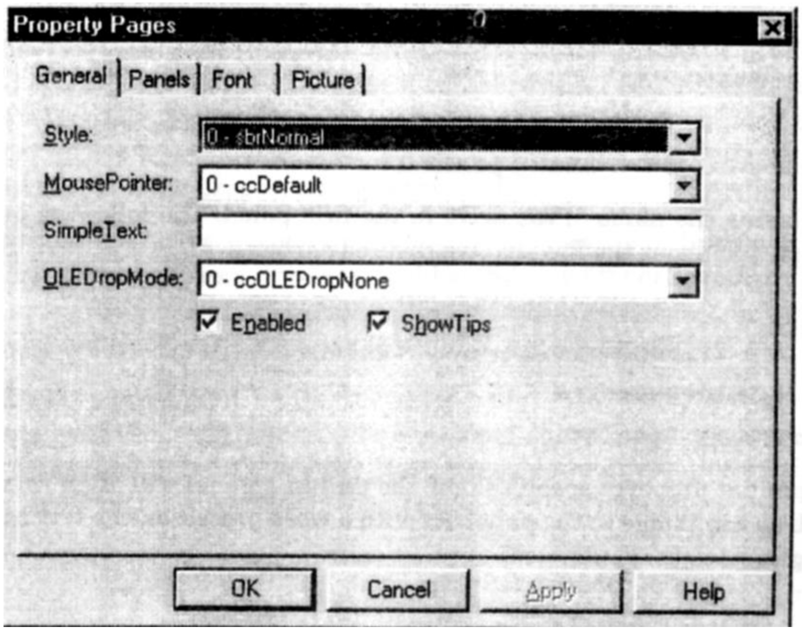
Working with the StatusBar control:

The Status Bar is a member of the Microsoft Windows Common Controls group. On your Tool Box, the Status Bar icon looks like this.

Draw the Status Bar on your form. It will automatically get aligned to the bottom of the form. By default it will show single panel.

The panel object and the panels collection

The Status Bar control is built around the Panels collection. Each object can display an image and text. To add panel objects at design time, right-click on the control and click on properties to display the property pages dialog box. By setting the Style property you can specify the status that should by displayed in the associated panel. The Style properties available are Text, Caps, Num. Ins, Scrl, Time, data and kana.



Style: 0 - sbrNormal
MousePointer: 0 - ccDefault
SimpleText:
OLEDropMode: 0 - ccOLEDropNone
[✓] Enabled [✓] ShowTips

At run time, you can dynamically change the text, images, or widths of any Panel object, using the Text, Picture and Width properties. Panel created with the Add method, as shown in the code below:

The Status Bar control is named “Status Bar1”

Dim Panl As Panel

Set Panl = StatusBarl. Panels.Add( )

Tree View Control

The Tree View control is very interesting control to work with. With this control you can display the data as a hierarchy. The windows explorer is built around a TreeView control. It displays the drives, directories, subdirectories and files in the form of a hierarchy. You can use the same control to display the data from a database.

Each item on the TreeView is a Node (like a directory) object. A Node object can have Child Nodes. A Child Node can be a simple node or it can have other Child Nodes. The TreeView control provides methods to expand and collapse Node objects.

You can navigate through a tree in code by retrieving a reference to Node objects using Root, Parent, Child, First Sibling, Next, Previous, and Last Sibling properties. A Tree View control can use only one Image List at a time.

Creating a Tree View control

The Tree View control looks like this on your toolbox.

A Tree View control is made up of nodes. Each node is placed 'relative' to another node.

The new node will also have a relationship with the 'relative' node. To add a node, use the Add Method. The Syntax is as follows.

Object. **Add** (relative, relationship, key, text, image, selected image), where

Object	:	The name of the Tree View control.
Relative	:	The index number or key of a pre-existing Node object. The relationship between the new node and this pre-existing node is found in the next argument.
Relationship	:	Optional. Specifies the relative placement of the new Node object and the pre-existing node object referred to in the relative position.
Key	:	A Unique string that can be used to retrieve the Node with the Item method.
Text	:	The name of the node object that will be displayed in the Tree View control.
Image	:	Optional. The index of an image in an associated Image List control that is shown when the Node is selected.

tvwCustTree.Nodes.Add "state", tvwchild "city"

Will add a node called the city which will be a Child Node to the State Node. The constants which can be used and the effects of using these constants are given below.

Constant	**Value**	**Description**
tvwLast	1	The Node is placed after all other nodes at the same level of the node named in relative.
tvwNext	2	The Node is placed after the node named in relative
tvwPrevious	3	The Node is placed before the node named in relative.
tvwChild	4	The Node becomes a child node of the node named relative.

Example

tvwCusTree. Nodes.Add “State”, tvwNext

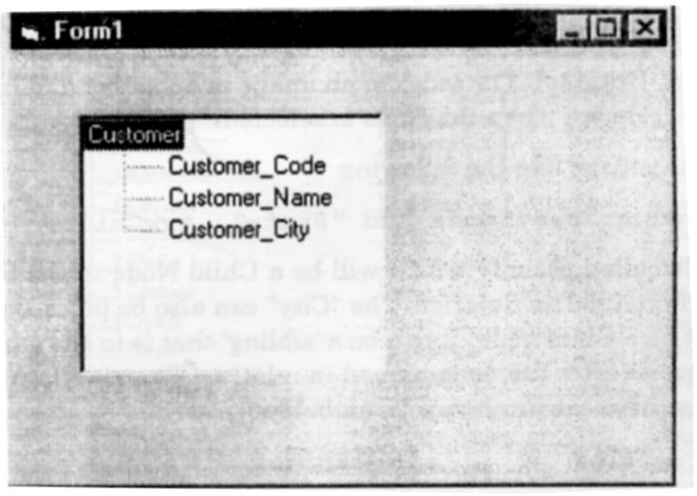
The above statement will add a new node next to the “State” node.

Working with the Tree View control

Draw a Tree View control on your form. Add the following code to the form load event.

```
Private sub Form_Load ( )
Dim NewNode As Node
    Set NewNode = TreeView1. Nodes. Add (, , “c”, “Customer”)
    Set NewNode = TreeView1. Nodes. Add (“c” tvwChild, “child1”, “Customer_code”)
    Set NewNode = TreeView1. Nodes. Add (“c” tvwChild, “child2”, “Customer_code”)
    Set NewNode = TreeView1. Nodes. Add (“c” tvwChild, “child3”, “Customer_City”)
End sub
```

Run the program, you will see a figure like the one below. When you click on the item Customers in the Tree View control, the node will display the child nodes.



Displaying Data from a Database

Use the Tree View control to display the data from the Supplier and the Supplier _ Item tables.

```
Dim db As Database
Dim rsSupplier As Recordset   'for the supplier table
Dim rsItems As Recordset      'for the Supplier _ product table
Dim New Node As Node
In the form load event, add the following code.
Private sub Form _ Load ( )
    'Open the database
    Set db = Open Database ("C:\azam\vbexercise\Invoice.MDB")
    ' Create recordset form the two tables.
    Set rsItems = db. OpenRecordset ("Supplier _ Product", db
        OpenDynaset)
    Set rsSupplier = db. OpenRecordset ("Supplier",
        dbOpenDynaset)
    'Initialize the Tree View control, Add a new node and set its text
property.
    TreeView1. Sorted = True
    Set NewNode = Tree View. Nodes. Add( )
    NewNode. Text = "Suppliers"
End sub
```

Populating the Tree View control

We need to list the suppliers from the Supplier table. For each supplier one node must be added. These nodes must be child nodes to the first node that we created in the form load event. Therefore, the syntax must be as follows.

Set New Node = Tree View 1. Nodes. Add(1,tvwChild), where

'1' specifies the first node (relative) and the constant tvwChild indicates that the new node must be a child node to the first node.

Your code will look like this.

```
    'Move to the first record of the Recordset.
rsSupplier. Move First
    Dim intIndex As Integer 'Variable for index. We will use this index
later.
    'Add a node for every record in the Recordset. For each node the
        Supplier Name will be the text.
Do Until rsSupplier.EOF
Set NewNode = TreeView1. Nodes. Add (1,tvwChild, , rsSupplier. Fields (1))
    ' The above can be written in two lines for readability
```

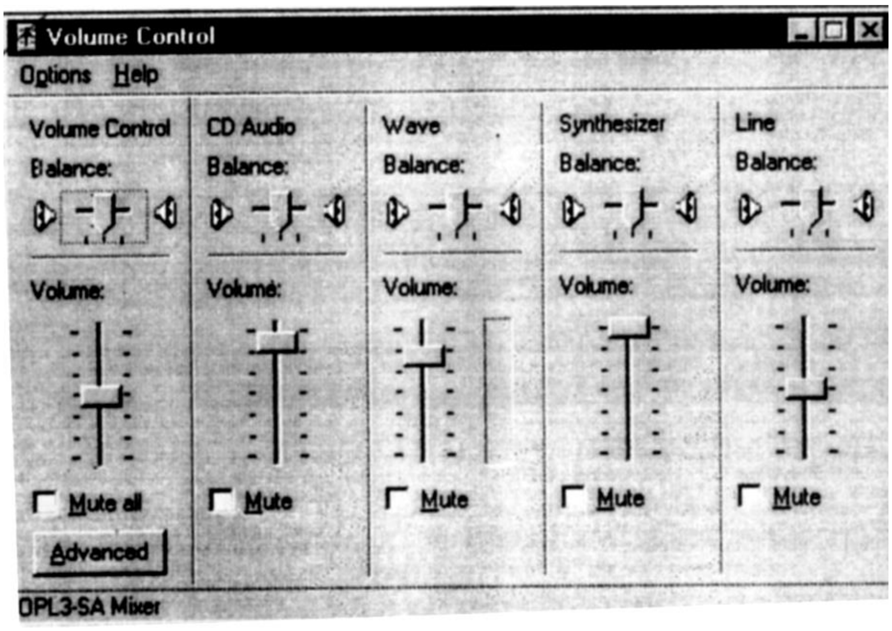
‘ Set NewNode = TreeView1. Nodes. Add(1, tvwChild)

‘ NewNode. Text = rsSupplier. Fields (1)

Slider Control

A Slider control is a window containing a slider and optional tick marks. You can move the slider by dragging it, clicking the mouse to either side of the slider, or using the keyboard.

To see a number of Slider controls in action, right click your mouse on the Speaker icon on your taskbar. Select ‘Open Volume Controls’. You will see a figure like this.



The Slider control can be used to,

1. Select a particular value.
2. Select a range of numbers to be passed into an array.
3. Resize a form, field, or other graphic object.

Tick Style and Tick Frequency Properties

The Slider control consists of two parts: the thumb and the ticks. The appearance of the control depends on the Tick Style property.

In addition to the placement of the ticks, you can also program how many ticks appear on the control by setting the Tick Frequency property. The Min and Max properties determine the upper and lower limits of a Slider control, at design time, right-click on the control, click Properties to display the Property pages dialog box, and set the Min, Max Values. At run time, you can reset the Min and Max settings to accommodate different ranges.

SmallChange and LargeChange Properties

The SmallChange and LargeChange properties determine how the Slider control will increase or decrease when the user clicks it. The smallChange property specifies how many ticks the thumb will move when the user presses the left or right arrow keys. The LargeChange property specifies how many ticks the thumb will move when the user clicks the control or when the user presses the PAGEUP or PAGEDOWN keys.

MaskEdit Box Control

The MaskedEdit control is used to prompt users for data input using a mask pattern. You can also use it to prompt for dates, currency and time, or to convert input data to all upper- or lowercase letters. When you define an input mask using the Mask property, each character position in the Masked Edit control will map to a placeholder of a specified type, or to a literal character. The MaskedEdit control is a bound control and can be used with a data control to display or update filed value in a data set.

Possible Uses To prompt for date/time, number, or currency information.

The Mask Property

The Mask property determines the type of information that is input into the MaskedEdit control.

Mask character	**Description**
#	Digit placeholder.
.	Decimal placeholder.
,	Thousands separator.
:	Time separator.
/	Date separator.
\	Treat the next character in the mask string as a literal.
&	Character placeholder.
>	Convert all the characters that follow to uppercase.
<	Convert all the character that follow to lowercase.
A	Alphanumeric character placeholder.
a	Alphanumeric character placeholder.
9	Digit placeholder.
C	Character or space placeholder
?	Letter place holder.

MaskEdBox1.Mask = (###)-###-####

The text property of the example above returns the string "(555)-555-5555"- the phone number that was entered.

Defining the Input Character

All mask characters are underlined.

You can also change the underline input character to a different character by using MaskEdBox1.PromptChar = "*".

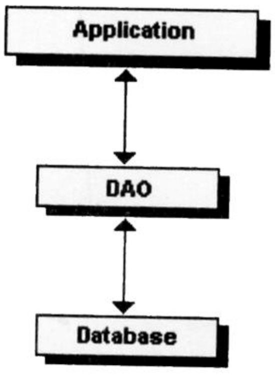
Data type	**Value**	**Description**
Number	(Default)	Empty string General Numeric format. Displays as entered.
Number	$#,##0.00;	($#,##0.00) Currency format. Uses thousands separator; displays negative numbers enclosed in parentheses.
Number	0	Fixed number format. Displays at least one digit.
Number	#,##0	Commas format. Uses commas as thousands separator.
Number	0%	Percent format. Multiplies value by 100 and appends a percent sign.
Number	0.00E+00	Scientific format. Uses standard scientific notation.
Date/Time	dddddd	Long Date format. Same as the Long Date setting in the international section of the Microsoft Windows Control panel.Example: Tuesday, May 26,1992.
Date/Time	dd-mmm-yy	Medium Date format. Example: 26-may-92.
Date/Time	ddddd	Short Date format. Same as the Date settings in the international section of the Microsoft Windows Control Panel. Example: 5/26/92.
Date/Time	ttttt	Long Time format. Same as the Time setting in the International section of the Microsoft Windows Control Panel. Example: 05:36:17 A.M.
Date/Time	hh:mm A.M/P.M	Medium Time format. Example: 05:36 A.M.
Date/Time	hh:mm	Short Time format. Example: 05:36.

The ValidationError Event

The ValidationError event occurs when the MaskedEdit control receives invalid input, as determined by the input mask. Unless you write an event handler to respond to the ValidationError event, theMaskedEdit control will simply remain at the current insertion point and nothing will happen.

4.3ActiveX Data Objects

ADO is meant to replace DAO, RDO and ODBC,



What we need is a simple, consistent application programming interface (API) that enables applications to gain access to and modify a wide variety of data sources. A data source may be a database, a text file, a spreadsheet, a graphics application, a cluster of heterogeneous databases, or something yet to be invented.

OLE DB

OLE DB, a set of Component Object Model (COM) interfaces that provide uniform access to data stored in diverse information sources. It is defined as a general- purpose set of interfaces designed to let developers build data access tools as components using the Component Object Model (COM). OLE DB enables applications to have uniform access to data stored in DBMS and non- DBMS information containers, while continuing to take advantage of the benefits of database technology without having to transfer data from its place of origin to a DBMS.

OLE DB has what it calls ‘providers’ which let you access the different data sources. For different data sources you have different data providers.

1. A Cursor Service. A cursor is defined as a temporary, read-only table that saves the results of a query with an assigned name. The cursor is available for browsing, reporting, or other uses until it is closed.
2. A service to perform batch updates.
3. A shape service to build the data in the form of a hierarchy.
4. A remote data service provider for managing data in multi-tier environments over connected or disconnected networks.

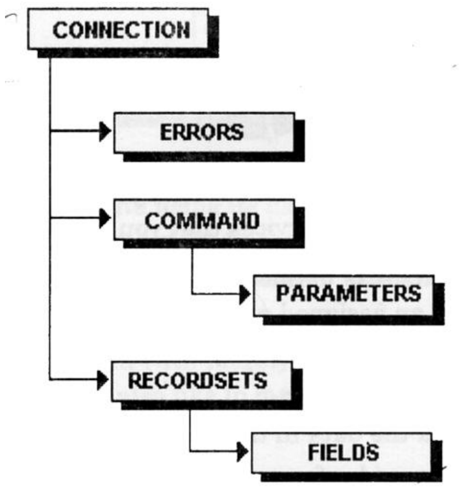
The ADO acts like the intermediary between the application and the OLE DB.

ADO

“ADO’s primary benefits are ease of use high speed, low memory overheads, and a small disk footprint. ADO supports key features for building client/server and Web-based applications.

The ADO features an object model like the DAO and the RDO but it is much flatter.

The ADO programming model illustrated below allows you do all this more.



The goal of ADO is to gain access, to edit, and update data sources. It provides classes and objects to perform each of the following activities:

Connection Make a connection to a data source.

Command Create on object to represent an SQL command.

Parameter Specify columns, tables, and values in the SQL command as variable parameters.

Recordset Execute the command and store the result, if the command is row returning, in a cache. Allow the user to sort, view or edit the data. If necessary, update the data source.

Connection You can access a data source using the Connection object. Unless a connection is made, data cannot be exchanged between the data source and the application. The connection object specifies the name of the data source, the provider that will be used to access the data, and other parameters.

Command Once a connection has been established with the data source, the data has to be extracted. This is done using the Command object. The command adds, deletes and updates data in the data source, or retrieves data in the form of rows in a table.

Parameter The command to retrieve data can be qualified using parameters. Parameters are arguments to a command that alter the result of the execution of the command. Parameters are especially useful for executing commands that behave like functions.

Recordset Although ADO allows you to access any type of data, our discussion here is limited to data from a database. The command object when executed will return a set of rows from one or more tables. This set of rows is called a Recordset.

The Recordset is the primary means of examining and modifying data in the rows.

- Specify which rows are available for examination
- Traverse the rows
- Specify the order in which the rows may be traversed
- Add, change, or delete rows
- Update the data source with changed rows
- Manage the overall state of the Recordset.

Field A row of a Reckordset consists of one or more fields. If you envision the Recordset as a two dimensional grid, the fields line up to form columns. Each field (column) has among its attributes a name, a data type, and a value.

Error Errors can occur at any time in your application, due to the data source being corrupted or renamed by somebody, or the Password being changed.

Property There are two types of properties: built-in and dynamic. Built- in properties are parts of the ADO object, and are always available. Dynamic properties are added to the ADO object's Properties collection by the underlying data provider.

Collection The objects in the collection can be retrieved with a collection method, either by name, as a text string, or by ordinal as an integer number.

The command object has the Parameters collection, which contains all Parameter objects that apply to that Command object.

The Recordset object has the Fields collection, which contains all Field objects that define the columns of that Recordset object.

The connection, Command, Recordset, and Field objects all have a Properties collection, which contain all the Property objects that apply to their respective containing objects.

Events ADO 2.0 introduces the concept of events to the programming model. Events are notifications that certain operations are about to occur, or have already occurred.

Connection Events: Events are issued when transactions on a connection begin, are committed, or rolled back, when Commands execute, and when Connections start or end.

Recordset Events: Events are issued to report the progress of data retrieval in the following cases. When you navigate through the rows of a Recordset object, when you change a field in a row of a Recordset, change a row in a Recordset, or make any change in the entire Recordset.

Establishing a Reference

To use ADO in your project, you have to make a reference to it. This is done just as we made the reference to DAO. Click on Projects, and from the menu Select References. From the list displayed in the reference dialog box, select Microsoft Active X Data Objects 2.0 Library and the Microsoft Active X Data Objects Recordset 2.0 Library. Now you can use ADO in your project.

The ODBC Data Source Administrator

ODBC is defined as “a standard protocol for database servers. ODBC has drivers for various databases that enable applications to connect to the databases and access their data.” The condition is that these databases must have SQL as the standard for data access. From the Control Panel, double click on the ODBC icon. This will bring up the ODBC Data Source Administrator dialog box. Click on the Add button to add a data source. Another dialog box will be displayed, asking you to select the driver.

Select Microsoft Access driver (*.mdb) since we are going to work on our Invoice. mdb. Click on Finish. The next dialog box displayed will ask you to specify the name of the database.

Click on Select to choose the name of the .mdb file. After selecting the .mdb file, enter the name of the data source. You will be using this name as the DSN (Data Source Name). Click OK and exit from the ODBC administrator.

Using the Data Source Name in Our Project

In the General declaration, add the following lines of code

```
Dim adocon As New ADODB. Connection
Dim rs As Recordset
Dim strconnect As string
```

The first line declares and sets ‘adocon’ as an ADODB connection object. You cam declare the above as follows

```
Dim adocon as ADODB. Connection
```

In the Form-Load event you can say

```
Set adocon As New ADODB. Connection
```

We have declared the connection object. Now to set the connection to a data source.

Creating the command We can create a command object and assign the command string as the commandText as follows.

```
Dim comd As New ADODB. Command
Comd. CommandText = “SELECT * from customer”
```

The command object must be linked to the connection object using the following line.

```
comd.ActiveConnection = adocon
```

Executing the command The command can be executed by either using the command object, or by using the Recordset object.

Add the following lines to your code

```
set rs = comd.Execute
MsgBox rs.Fields (1)
```

Manipulating the records in the recordset

Add a command button to your form. Add the following lines of code to its click-event.

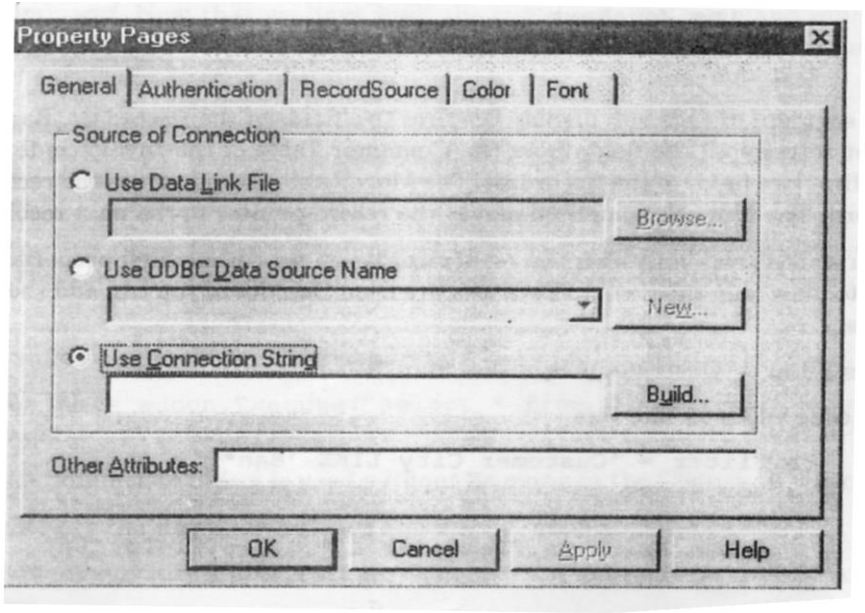
```
        Private Sub Command1-Click( )
            rs. MoveFirst
          Do While Not rs. EFO
             Debug. Print rs. Fields (0) & " "; rs. Fields (1)
                rs. MoveNext
        Loop
End  Sub
```

This segment of code will display the first two fields of the Recordset. Remember that this Recordset returns all the fields from the Customer Table of the invoice.mdb. We are viewing only the first two fields of the Recordset. The Move First method moves the record pointer to the first record. The MoveNext method moves the record pointer to the next record.

USING THE ADO DATA CONTROL

We can display the data from a Recordset (data source) using ADO code, or with the help of the ADO Data Control. In order to use the ADO Data, we need to add the control to the form. The ADO Data works just like the Data control that we worked on earlier. However the Data control cannot work with ADO, so we need add the ADO Data control.

Right click on the ToolBox, and from the pop-up menu select Components. From this dialog box click on Microsoft ADO Data Control 6.0 (OLEDB). The ADO Data control gets added to your Tool Box. Draw the ADO data control on your form and set the properties. Right Click on the ADO Data control and select ADODC Properties from the menu. The Property Pages dialog box will look like this.



The Property Pages of ADODC contain four tabs. They allow you to set the various properties of the ADODC. They are

General In this tab you specify how the ADODC should connect to a data source. There are three options.

Use data links file You will need this option if you are going to link a textbox or a grid or some such control to an application like. Excel or Word via DDE.

Use ODBC data source name You can mention the name of the DSN that we created using the ODBC Data Source Administrator. The DSNs already created will be displayed in a drop-down ListBox. You can select the one you need to work with, or you can build a new DSN.

Use connection string You can build the connection string here by clicking on the 'Build' button. This will bring up a Wizard and guide you along.

Authentication This lets you enter Authentication information like the User Name and Password.

RecordSource Here you can specify the method of creating the Recordset. That is, you can indicate the Command Type (adCmdUnknown or adCmdText or adCmdTable, or adCmdStoredProc)

Updating the data in the Data Source

There are two approaches that ADO uses to add or modify the data in the database

1. Changes made to the data or the rows are made in the 'copy buffer' and not directly to the Recordset. If the changes are acceptable then they are applied to the Recordset.
2. Changes are made directly to the data source either immediately or in a batch mode. These modes are governed by the **CursorLocation** and **LockType** properties. Changes will made to the data source in the immediate mode as soon as soon as you confirm an update.

Review Questions:

1. Define simplicity?
2. Define SQL and its name.
3. Define DDL and DML.
4. Explain functions of jet database engine?
5. Explain DAO object model?
6. How to create Database?
7. How to create Table? Explain it with example.
8. Write VB code for Quiz program.
9. Explain SSTab control.
10. How to edit and update the record?
11. What is the difference between TabStrip and SSTab?
12. How to create a TreeView control?
13. What is ADO? Write the structure and component of ADO.
14. How to work with ADO? Explain.

NOTES

UNIT-V

Objectives:

The objective of this unit is to introduce Crystal and Data Reports, Distributing Your Application, What is ActiveX, DDE and OLE, the technologies before ActiveX, What is an ActiveX Control, Creating an ActiveX Control, Writing the code, adding properties, Registering the ActiveX Control.

This unit also discusses ActiveX and Internet, Microsoft ActiveX Control Pad, Creating a Web Page, What is HTML, Placing an ActiveX Control on the Page, ActiveX Documents.

Contents:

5.1Crystal and Data Reports

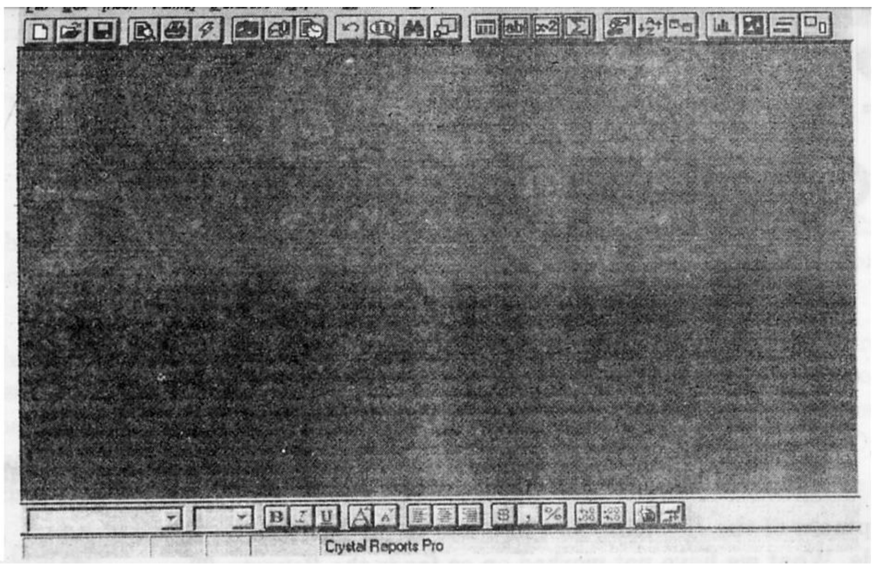
Crystal Reports Crystal Reports is a third party product developed by Seagate of Singapore. It has been bundled with various data access tools. It can work with Visual Basic on your PC.

Prerequisites for working with Crystal reports

HardWare A printer must be installed. It need not be physically connected though. This is because Crystal Reports builds the reports based on the properties of the printer.

Application You can access Crystal Reports only through the VB IDE. If Crystal report has not been installed then follow the steps given below. Prepare a pencil copy of the report structure that you want to create.

Installation So, Crystal Reports has been installed and you have a rough ‘copy’ of the report that you want.



Creating a Report through a Wizard

You are presented with the Create New Report Wizard that will allow you to create new reports. You can choose the report style. For this exercise, click on Standard.

The wizard in the dialog box will ask you to select the database(s) that you will be using to generate the report. Let us select Nwind.mdb for a change. (You will be generating your own reports using the Invoice.mdb). Upon selecting the Nwind.mdb, all the tables and stored queries/view will get added to the ListBox. After you have added all the databases that you want to work on click on 'Done'.

If you think there are too many tables and views and you do not need all of them, then you can delete some of them. Click on the button 'Back'.

You can select the items that you do not need and click on remove to remove them one by one. When you are sure you have only those tables that you need click Next to continue. You will see the figure with the selected tables and their relationship. Click Next to continue.

You can add the fields that you wish to include in your report. The fields that you select here will appear on the report. However the selection criteria for the selected records need not depend on these fields alone. When you have selected the fields and added them one by one in the 'Report fields' ListBox. Click on Next to continue.

You can choose the fields on which the report is to be sorted out. For example, you can sort all the details based on the City, or the Product that a customer uses, or the Turnover of the company, etc. Select the fields on which the criteria are to be built and then select the sort order. For example, you can sort the details in the ascending order or descending order. When you are through with this click Next to continue.

Select the fields on which you have to perform calculations like group total, sub-total, etc. For example if you want to know the number of customers in a particular city, then select Customer-City and add it to the “Total Fields’ ListBox. Here you can also choose if you want to total the number of customers for a city or if you want to the add the figures for a particular column. For our example choose Count. Then Click Next to continue.

Choose the fields based on which the records must be selected from the database. In the Report Fields ListBox you are presented with the fields that you have selected for the report. If none of these fields meet your requirements to determine the selection criteria, you can scroll down further and select from the fields that have not been included in the report. Build your selection criteria and click Next to continue.

Select the layout of the report. Select the report layout style that you think suits you best. The selection of the style will depend upon the type of data that you are likely to have on the report. Next you preview the report.

Data Report

Data Report is the new offering from Microsoft perhaps with a view to replacing Crystal Reports in the long run. Data Report as it stands today is ment for programmers. A general user of computers will not be able to get around it. You need to follow the following steps to generate a report using the data Repost.

1. Create a data source using ADO.
2. Add the data Report object to your project.
3. Place text boxes representing the various filed that you want on the Data Report object.
4. Link the Textboxes to the various fields of the data source.
5. Display the Report using the Show method.

Getting acquainted with Data Report Designer

The Data Report Designer is not part of your toolbox. To add it to your toolbox, right click on the toolbox and select Components from the pop-up menu. On the Component dialog box, click on the ‘Designer’ tab and select ‘Data Report’. Close this dialog box.

Click on ‘Project’ In this menu you will see a new item ‘Add Data Report’. Select this item to add a Data Report Designer to your IDE. The Data Report Designer is a separate from by itself. Open the Project Explorer and you will see another item called Data Report with the Form-1.

Parts of the Data Report

The Data Report consists of three main components.

- Data Report object
- Section object
- Data Report Controls

This is the visual Designer component of the Data Report object. The designer component can be controlled programmatically using the Data Report object. The Data Report controls, which are special control that you can create on the

Data Report designer. These tools are placed under a separate tab on your toolbox.

The default Data Report designer contains these Sections

- **Report Header:** You give the tile of the report in this section. If the first page of the report should contain only the tile, then set its Force Page Break property to rpt Page Break After.
- **Page Header:** You give the page heading here.
- **Group Header/Footer:** you give the heading for every group here. For example your report can contain details of customer for each city here. So the Group Header can be the name of city. A group header must also have a Group Footer.
- **Details:** This section contains the actual data. The records are displayed in this section.
- **Page Footer:** You give the page footer here. This can be the page number or any relevant text like the data of report, etc.
- **Report Footer**: You give the summary for the report in this section. This can contain the address, the bibliography, contact address, etc. the report Footer appears between the last Page Header and Page Footer.

Data Report Controls

Following are the new set of controls that are placed under the Data report tab on your toolbox.

- TextBox Control (RptTextBox)- To display text or other formatted data.
- Label Control (RptLable) - To display the labels on the report to identify fields or sections.
- Image Control (RptImage) - To display picture on the report. This control cannot be tied to a data field.
- Line Control (RptLine) – To draw lines on the report.
- Shape Control (Rptshape) – To draw rectangles,circles,etc on the report.
- Function Control (RptFunction) – This is a special text box that calculates value as the report is generated.

Extracting the Data

First create a data source using ADO. Let us create a list that consists of Customer Name, Customer-City and Order-Value. We need to work with two tables.

```
Dim adocon As ADODB. Connection
Dim adors As ADODB. Recordset
```

In the Form load event add code to create the connection and then to create a Recordset. The following lines of code will do the trick

```
        Set adocon = New ADODB. Connection
```

Adocon. Open “DSN = Ivoice”

Working with the Data Report

In the details section of the Data Report designer, add three of the RptTextBox controls. Notice that it is just like adding ordinary textbox controls. Also observe that these textboxes contain a caption called ‘Unbound’. This means that these RptTextBox controls are not bound to any data source or data field.

Displaying the Report

We are now ready to display the data. We have created the Recordset. We have assigned the fields in the data Report Designer. We need to link the Record source to the Data Report. Then we must call the show method of the Data Report. The following lines have to be added to the “Display” command button.

Set DataReport1. Data Source = adors

DataReport1. Show

Run the program now.

Creating Multiple Reports

Displaying more than one report using only one Data Report involves a little work. Details like Caption, Page Headers, Footers, etc for each of the reports must be determined. The heading for the data must also be determined. The data and the source of the data must also be worked out. Depending upon the number of reports that you may need to display on a form, you have to work out if it is feasible to create a Recordset or a number of Record sets for all the reports. Creating a Recordset every time the user asks for a report may not be a good idea especially in a multi-user environment.

5.2 Distributing your application

Working with the Packaging and Deployment Wizard

The wizard compresses your application files and places them in cabinet (.cab) files. You can choose to have a single .cab file or multiple files to be copied on floppies.

There are two ways which you can start the wizard. The wizard can be invoked from within the VB IDE. It is available under the Add_ Ins menu option. The other recommended method is to invoke it from the Visual Studio Program group.

Click on the Start button and select the Programs option. Then select Microsoft Visual Studio 6.0, and Microsoft Visual Studio 6.0 Tools. Under this option you will find the Packaging and Deployment Wizard.

From this Dialog box you can

1 Select the project that you want to package.

A drop down list box will display the past projects. You can the name of the project that you want to package, or you can click on the Browse button to select the project.

After selecting the project you can choose to,

Package the project.

Deploy the packaged project either on floppies, the local drive, or on the Internet. Manage the deployment scripts.

Click on the Package button to begin packaging your application. You can choose the packaging script in dialog box. If you have already packaged the current project, its previous setting is stored in a script.

Choose the type pf the package that you want to create. You can create a Standard Setup program that the user can install using the Setup.exe programs. Creating a standard setup program, choose the option to create the Standard Exe program. Click Next.

Select the directory where the setup program must be saved. Choose the directory where you want to save the new package. Click Next.

Choose the drive that you wish to include. If you have used only DAO, then you will need only Jet drivers. Select the driver and add it to the ListBox with the heading ‘Included drivers’. Click Next to continue.

AlistBox will show you all the files that he wizard will add for your application to work. These files are the standard drivers, DLLs. For every control that you have added to your toolbox, you will see that the wizard adds the OCXs, DLLs, dependency files, etc. You have to specify whether you want the entire setup program in one large file, or in smaller units. If you plan to distribute your application on CD-ROMS, then you can choose the option to create a single .cab file.

Enter the tile that will appear during installation. If you so desire, your application can appear on the Start Menu itself. Click Next to continue.

This dialog box displays the name and the current location of each file in the package, the files that will be installed and the folders where they will be copied. You can specify the location to which your application files will be installed on the user’s machine.

Select the file whose location information you wish to change, and then click on the macro under the heading ‘Install Location’. A dropdown ListBox will display the other location (macros) that are available where you can copy the file.

You can specify the files that can be shared by other applications. Click Next to continue.

Now you have to give the name under which the Setup script has to be saved. Accept the name suggested by the Wizard, and click Finish.

5.3 ActiveX

What is ActiveX?

Why ActiveX?

The concept of ActiveX was developed for very simple reasons.

To be able to make changes in the imported data such that the data in the parent application was also updated.

To be able to place different types of data or ‘objects’ in one document.

The technology was DDE or Dynamic Data Exchange.

This technology allowed application to exchange data.

It also allowed one application to send commands to the other application.

Next came OLE, an acronym for Object Linking and Embedding.

Under this technology, one document could display an object from different applications.

The advantage of this technology was that no conversion of data was done from one application to another.

Programmers who work in the C language are very found of using the term ‘function’.

OLE 2. The Next Step

OLE 1.0 did not succeed because of various limitations.

The need was for a document to load and save an object that it did not know about.

The application should provide an interface to edit objects that it contains.

To support drag-and-drop of objects that it does not know about.

To execute commands on objects belonging to unknown applications.

Microsoft came up with some specifications on Objects and how they should be handled. The specifications are as under, and are COM specifications. COM stands for Component Object Model, the new buzzword in computers.

1. A common method for applications to exchange objects.
2. A method to identify an object and relate an object with applications that can manipulate it.
3. A standard set of error codes and an error-reporting and responding system.
4. A system to invoke an object. Check if it is in use, and delete it if it is not in use. This ensures that memory resources are not blocked.
5. A set of rules for applications to access and manipulate objects.

An application that accepts or requests an object is called as the client or container. The application that provides an object is called the Server.

Rule 1: Word must know how to accept the spreadsheet from Excel.

Rule 2: If the user double-click on the spreadsheet, Word (or the system) must be able to identify the object and invoke Excel.

Rule 3: In case of an error like Excel being corrupted or “This Application has performed….’ Error, both the applications must be notified. A copy of the FIR is to be sent to the MLA!

Rule 4: Once the modifications have been completed on the object, the instance of Excel must be closed. This will ensure that resources are not blocked.

Rule 5: This ensure that the spreadsheet is presented as a spreadsheet, and that the rules for presenting spreadsheet data are followed.

Let us take a look at some of the properties of ActiveX controls.

Any control or object has

Properties	: Like BackColour, Font, Resize, Paint, etc.
Events	: Click, MouseMove, KeyPress, LostFocus, etc.
Methods	: The code associated with the control.

5.4 ActiveX and Web Pages

ActiveX and Internet

ActiveX Documents go well with the Microsoft’s strategy for the internet.

The internet is a collection of computers that are wired together and talk in TCP/IP. The World Wide Web is a collection of documents that are linked together. The documents are viewed by the users in the form of Pages. The Pages will have topics that are linked to other topics and so forth. These pages are read by what is called as Internet Browsers. The browsers understand a language called HTML’ or HyperText Markup Language.

All pages that have to be placed on the web have to be created using this language. ActiveX Controls can be added to Web Pages to improve the functionality of the Page. An ActiveX control can be loaded on the page to play a movie clip .avi file or to present a form. Microsoft has provided a new tool to simplify the addition of ActiveX controls to a Page. This tool is called Microsoft ActiveX Control Pad.

An HTML page begins with a tag <HTML>. This tag tells the browser that the file is an HTML file, and hence must be treated with respect.

An HTML page can contain text with different styles. Each style for the sake of (our) convenience can be called a Para. Each Para is preceded by a tag and ends with a tag. A tag is a bit of text that tells the browser how the following text has to be read and displayed.

The ending tag for a Para will tell the browser that the Para has come to an end the next definition if any will begin with the next tag. The HEAD section includes the tile pf the page and the name of the creator of this page, etc. The BODY section is where most of the action is. You can add text here or pictures, or other controls.

Some definitions are here for you:

HTTP: Hypertext Transfer Protocol. This protocol is used to transfer pages from the server to the client.

Server: The computer that has the pages that you requested.

Client: Your computer that has made the requested.

URL: Uniform Resource Locator. This contains the information needed to locate the page that you want. An URL has

- The Protocol
- The Server name
- The Pathname of the file in the server's directory
- Page Name, the name of the file that has the page
- Page Number#, the page number which hold the text that you want.

5.5 ActiveX Documents

An ActiveX document is Visual Basic Program (with the various controls) that can be read and displayed by a browser.

Now how do we go about creating an ActiveX Document?

Start Visual Basic.

Select ActiveX Document Exe from the opening menu.

You will see a form called the UserDocument1. It will look similar to the regular Visual Basic form.

From the Toolbox paste the controls to arrive the following picture.

You are welcome to make it look different.

Add the following code to User Document_ Initialize ( )

```
Private Sub UserDocument _ Initialize( )
Dim db As Database
Dim db As Recordset
'open the database
 Set db = OpenDatabase (App.Path & "\ Car Finance.mdb")
Set datal = db. OpenRecordset ('Schemes")
'the following code will populate the grid
Gridl.Rows = 11
Gridl.Cols = 7
Gridl . Colwidth (0) = Gridl . Width * 0.14
Gridl . ColWidth (1) = Gridl . Width * 0.163
Gridl . ColWidth (2) = Gridl . Width * 0.16
Gridl . ColWidth (3) = Gridl . Width * 0.16
Gridl . ColWidth (4) = Gridl . Width * 0.17
Gridl . ColWidth (5) = Gridl . Width * 0.07
Gridl . ColWidth (6) = Gridl . Width * 0.1
If datal . RecordCount > 0 Then datal . MoveFirst
i = 1
Do While Not datal . EOF
Grid1.Row = i
Grid1.TextMatrix(i,0) = data1!carname
Grid1.TextMatrix(i,1) = data1!company
Grid1.TextMatrix(i,2) = Format(data1!sellingprice,
"###, ###, ##0.00") & " Rs."
Grid1.TextMatrix(i,3) = Format(data1!inipayment,
"###, ###, ###0.00") & " Rs."
Grid1.TextMatrix(i,4) = Format(data1!ms,
"###, ###, ##0.00") & " Rs."
Grid1.TextMatrix(i,5) = data1!months
Grid1.TextMatrix(i,6) = data1!interest & "%P.A."
Data1.MoveNext
i = i+1
Loop
If data1.RecordCount > 0 Then data1. MoveFirst
Combo1. Clear
Do While Not data1.EOF
    If Not IsNull(data1.carname) Then Combo1.AddItem data1!carname
    data1.MoveNext
Loop
Data1.Close
Db. Close
SSFrame1.visible = False
End Sub
```

In the Click Event of the Application Form Command Button add the following code:

```
Private Sub SSCommand3_Click()
SSFrame1.visible = False
End Sub
```

The Application Form Document

Create the document using SSFrame and Paste it in the same document.

In the Click Event accept Command Button Add the Following Code

```
Private Sub SSCommand2_Click( )
Dim db As Database
Dim datal As Recordset
Set db = OpenDatabase (App.path & "\CarFinance.mdb")
```

```
    Set datal = db . OpenRecordset (" ApplicationForm")
    'Storing the information in to our database
    datal . Index = "AppIndex"
    datal . Seek "=", Trim (Textl (1) . Text)
    If Not datal . NoMatch Then
    Datal. Edit
    Else
    datal . Edit
    Else
        Datal . AddNew End If
    datal ! ClientName = Text1 (1) . Text
    datal ! address = Text1(2) . Text
    datal ! phone = Text1(3) . Text
    datal ! schemewanted = Text (4) . Text
    datal ! schemewanted = Trim ( Combol . Text)
    datal . Update
    datal . Close
    db . Close
    End Sub.
```

In the Home Command Button, add the following code:

```
     Private Sub SSCommand3_ Click ( )
     SSFramel . Visible = False
     End Sub
```

Review Questions:

1. Write the types of reports.
2. What is ActiveX?
3. Explain the concept of iterating ActiveX controls?
4. How to create an ActiveX control?
5. Write VB code to display student information system.
6. Explain DDE events and OLE.
7. Explain about distribution of your application.
8. How to create a Web Page?
9. What is HTML? Explain with example.
10. What is a tag? Explain in detail.
11. How to place an ActiveX Control on the page?
12. What is an ActiveX document? Explain in detail.
13. Create the document using SSFrame and Paste it in the same document.

NOTES