



KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed to be University Established Under Section 3 of UGC Act 1956)

Coimbatore – 641 021.

(For the candidates admitted from 2016 onwards)

DEPARTMENT OF COMMERCE (CA)

Syllabus	Semester 5
16CCU501A	L T P C
SOFTWARE DEVELOPMENT WITH VISUAL BASIC	4 - - 4

SCOPE : Software Development with Visual Basic enables the students to understand the basis of VB application, Variables, Active Data Object, and the aspects of reporting. This helps to understand the phases of software development

OBJECTIVES:

- To enable the students to develop a front end tool for customer interaction in business.
- To make the student to develop an application using Visual Basic.

UNIT- I

Introduction to VB – steps in VB application – Project Explorer Window – Property Window – Form Layout – Code Window – Event driven programming – Working with Forms.

UNIT-II

Variables – Constants – Literals – Data Types – Operators – Sub routine and Functions Programme Flow Control – String function – Numeric function – Date function.

UNIT-III

Pointers – Label – Frame – Check Box – Compo Box – Scroll Bar – Timer – Shape and Line Control - Command Button – List Box - Image Box - Picture Box – text Box – SDI and MDI form – Data Grid - Flex Grid – Menus – Dialog Boxes.

UNIT-IV

DAO – Creating a Data base – Types of Record set – ActiveX Data Object (ADO).

UNIT- V

Data Report: Data Environment – Designer – Connection object – Command object – Data Report control – Sections of Report designer. Case Study: Automated system for student mark list – Automated system for Railway reservation.

Suggested Readings:**Text Book:**

1. Gary Cornell. (2005). *Visual Basic 7 from the Ground up* [3rd Edition]. New Delhi: Tata McGraw Hill

Reference Books:

1. Content Development Group(2012). *Visual Basic 7 programming*. New Delhi: Tata McGraw Hill.



KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University Established Under Section 3 of UGC Act 1956)

Coimbatore – 641 021.

LECTURE PLAN DEPARTMENT OF COMPUTER SCIENCE

STAFF NAME: R.BHUVANESWARI

SUB.CODE:16CSU501A

SUBJECT NAME: SOFTWARE DEVELOPMENT WITH VISUAL BASIC

SEMESTER: V

CLASS: III B.Com (CA)

S.No	Lecture Duration Period	Topics to be Covered	Support Material/Page Nos
UNIT-I			
1	1	Introduction to VB	W1, W2
2	1	Steps in VB Application	T1: 1 – 19
3	1	VB IDE Explanation	W2, W3, R1
4	1	Project Explorer Window, Properties Window	T1: 33 – 45
5	1	Form Layout, Code Window	T1: 63 - 83
6	1	Event Driven Programming	T1: 83 – 89
7	1	Working with forms	T1: 90 – 94
8	1	Sample VB Program	W3
9	1	Recapitulation and Discussion of Important Questions	
Total No of Hours Planned For Unit 1=9			
UNIT-II			
1	1	Variables, Constants	T1: 150 – 152
2	1	Literals, Data types Declaration of variables and data types Giving length and values for variables	T1: 155 – 168 R1:170 - 178 W1, W2
3	1	Operators, Types of operators	T1: 176 – 180

4	1	Sub routine and functions Function Definition Call by Value and Call by Reference	T1: 193 - 200 W2 W3
5	1	Program Flow Control Control Statements, Looping Statements Unconditional Branching Statements	T1: 219 -254 W1 W2
6	1	String Function	T1:255 – 270
7	1	Numeric Function	T1: 285 – 287
8	1	Date Function	T1: 288 – 290
9	1	Sample Program	W2
10	1	Recapitulation and Discussion of Important Questions	
Total No of Hours Planned For Unit II=10			
UNIT-III			
1	1	Pointers, label, frame, scroll bar	T1: 122 -124
2	1	Check box, Combo box, Timer control	T1: 107 – 110
3	1	Shape and line control	T1:112 – 116
4	1	Command button, list box	T1:
5	1	Image box, picture box, text box	T1
6	1	SDI and MDI Form	T1
7	1	Data Grid, Flex Grid	T1
8	1	Menu, Dialog Boxes	T1
9	1	Sample Program	T1
10	1	Recapitulation and Discussion of Important Questions	
Total No of Hours Planned For Unit III=10			
UNIT-IV			
1	1	DAO – Introduction	T1, T2

2	1	Creating a Data Base	T1
3	1	Working with Data Base	T1
4	1	Types of Record set	T1
5	1	ActiveX Data Object(ADO)	T2
6	1	ActiveX Data Object (ADO)	T1
7	1	Sample program	T1
8	1	Recapitulation And Discussion Of Important Questions	
Total No of Hours Planned For Unit IV=8			
		UNIT-V	
1	1	Data Report: Data environment, designer	T1
2	1	Connection object, Command object	T1
3	1	Data Report control	T1
4	1	Sections of report designer	T2
5	1	Automated system for student mark list	T1
6	1	Automated system for railway reservation	T1
7	1	Recapitulation and Discussion of important Questions	
8	1	Discussion of Previous ESE Question Papers.	
9	1	Discussion of Previous ESE Question Papers.	
10	1	Discussion of Previous ESE Question Papers.	
11	1	Discussion of Previous ESE Question Papers.	
	Total No of Hours Planned for unit V=11		
Total Planned Hours			48

TEXT BOOK

1. Donald Hearn and M. Pauline Baker. 2007. *Computer Graphics-C Version*, Pearson Education, New Delhi, 2nd Edition.
(Page Nos. : 24-54, 56-77, 80-92, 103-118, 204-215, 236-256, 427-443, 458-463, 490-495, 502-505)

REFERENCES

1. Amarendra N. Sinha. 2008. *Computer Graphics*, Tata McGraw Hill, New Delhi, 1st Edition.
2. Foley, Vandam, Feiner and Hughes. 1999. *Computer Graphics Principles and Practices*, Addison Wesley, Singapore, 2nd Edition.
3. Evangeline, D., Anita, S. 2016. *Computer Graphics And Multimedia Insights, Mathematical Models And Programming Paradigms*. PHI Learning Pvt. Ltd.,
4. Rae Earnshaw.2014.*Computer Graphics: Developments in Virtual Environments*, academic Press, New Delhi, 2nd Edition.

WEBSITES

W1: [www.openlearning.intercol.edu/Computer graphics](http://www.openlearning.intercol.edu/Computer%20graphics)

W2: <http://www.cgtutorials.com>

W3: <https://www.touchsystems.com/opticaltouch>

UNIT-I**SYLLABUS**

Intoduction to VB – Steps in VB Application – Project Explorer Window – Property Window – Form Layout Window – Code Window – Event Driven Programming – Working with forms.

Introduction of Visual Basic

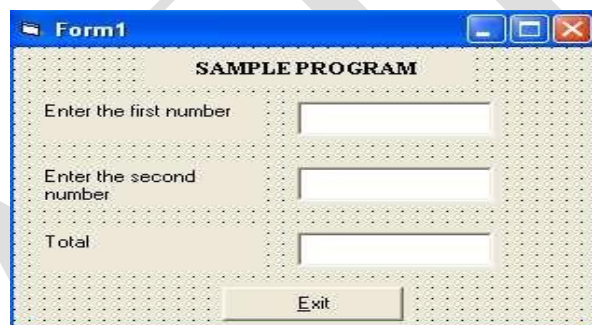
VISUAL BASIC is a high level programming language which evolved from the earlier DOS version called BASIC. BASIC means Beginners' All-purpose Symbolic Instruction Code. It is a very easy programming language to learn. Different software companies produced different versions of BASIC, such as Microsoft QBASIC, QUICKBASIC, GWBASIC, and IBM BASICA and so on. However, people prefer to use Microsoft Visual Basic today, as it is a well developed programming language and supporting resources are available everywhere. Now, there are many versions of VB exist in the market, the most popular one and still widely used by many VB programmers is none other than Visual Basic 6. We also have VB.net, VB2005, VB2008 and the latest VB2010. Both Vb2008 and VB2010 are fully object oriented programming (OOP) language. VISUAL BASIC is a VISUAL and events driven Programming Language. These are the main divergence from the old BASIC. In BASIC, programming is done in a text-only environment and the program is executed sequentially. In VB, programming is done in a graphical environment. In the old BASIC, you have to write program code for each graphical object you wish to display it on screen, including its position and its color. However, In VB, you just need to drag and drop any graphical object anywhere on the form, and you can change its color any time using the properties windows. On the other hand, because the user may click on certain object randomly, so each object has to be programmed independently to be able to response to those actions (events). Therefore, a VB Program is made up of many subprograms, each has its own program code, and each can be executed independently and at the same time each can be linked together in one way or another.

What is Visual Basic and need of visual basic?

Visual Basic is a tool that allows you to develop Windows (Graphic User Interface - GUI) applications. Basic denotes method of writing programs code functionality.

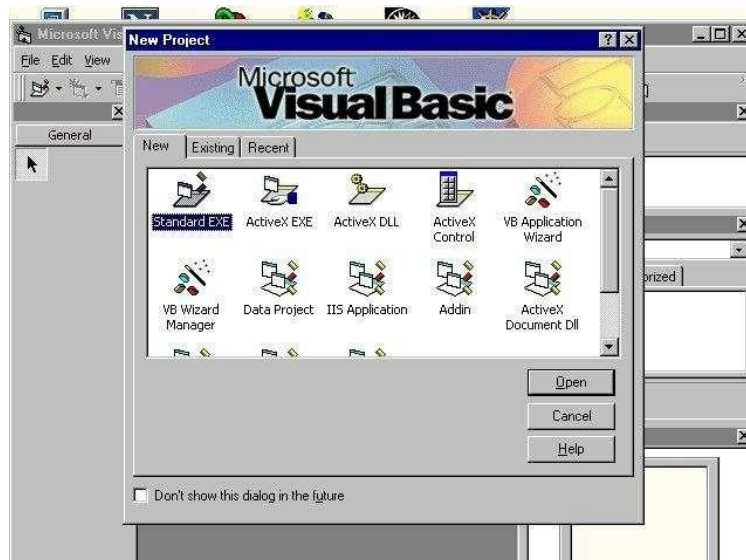
Visual Basic is event-driven, meaning code remains idle until called upon to respond to some event (button pressing, menu selection ...). Visual Basic is governed by an event processor. Nothing happens until an event is detected. Once an event is detected, the code corresponding to that event (event procedure) is executed. Program control is then returned to the event processor.

Getting Started

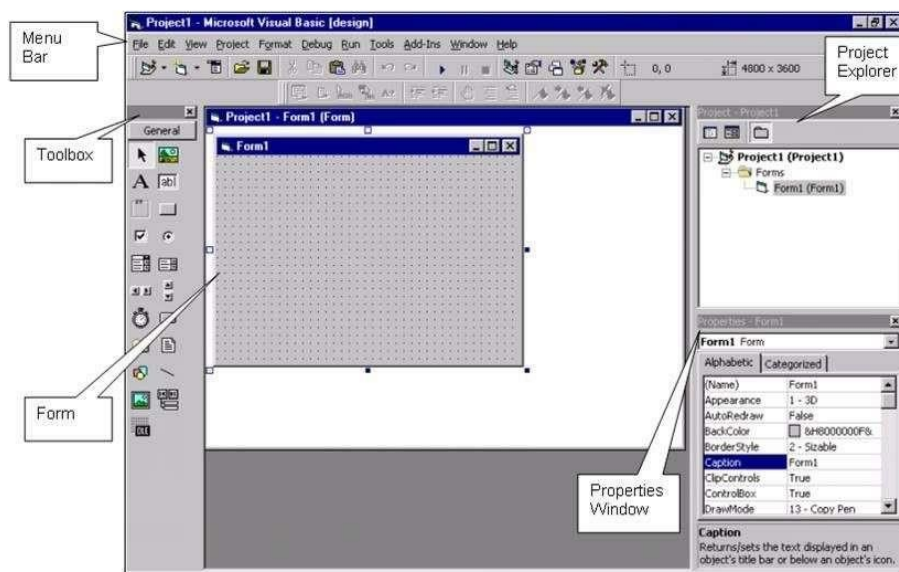


The diagram shows a typical Graphical User interface which has been created using Visual Basic. The programming created in Figure 1.1 requires an environment where we can create and modify the application. This is known as the Integrated Development Environment and is discussed in the next section below.

To launch Visual Basic, on the Taskbar, click Start \geq (All) Programs -> Microsoft Visual Studio 6.0 -> Microsoft Visual Basic 6.0. When Microsoft Visual Basic starts, the New Project dialog box comes up:



Double Click on the Standard EXE option (or) Select the default highlighted icon and click an open button. You will be then placed in Visual Basic's design environment as illustrated in Figure 1.3. To open the existing project, select File Menu open project.



Visual Basic is a high level programming paradigm. Its concepts are based upon Event driven programming. The environment to edit, delete and write code as well as develop windows based applications is known as the 'Integrated Development Environment' (IDE).

From the diagram it can be seen that the IDE is divided into separate areas or 'windows'. We have the Toolbox control which allows us to add objects on to Form window. We can change the properties using the properties windows for all the objects on the form. We can also edit/create the event handlers using the Code Window. When creating applications in Visual Basic it is quite common to use multiple forms, modules etc. The project explorer window is used to keep track of all the additional files used.

The main components of the Integrated Development Environment (IDE) are illustrated in the subsequent text.

Title Bar and Menu Bar

The title bar is the horizontal bar located at the top of the screen. It gives the name of the application and is common to all windows application. Everything below the title bar and menu bars in a window application is called the client area.

In Visual Basic, the title bar starts out displaying: Project1 – Microsoft Visual Basic [design]

The Menu bar gives you accesses too many features within the development environment.

1. On the left is the File Menu. It contains the commands such as you can create, open, print and save projects. All of this menu can also be accessed by right-clicking in the project explorer.
2. Next to File is the Edit menu. From here you can perform many of the editing tools that will help you write the code that activates the interface you design for your application, including the search-and-replace editing tools.
3. The View menu gives you fast access to the different parts of your program and to the different parts of the Visual Basic environment.
4. The Project menu is the heart of your project. You can add to and remove forms, code modules, user controls, property pages, as well as ActiveX designers from your projects.
5. The Format menu gives you a way to specify the look of controls that you will place on your forms.
6. The Debug menu contains the tools used to correct (debug) problems, or bugs in your code.
7. The Run menu gives you the tools needed to start and stop your program while in the development environment.

8. The Query and Diagram menu are mostly used in advanced database development. The commands in the Query menu simply the creation of SQL queries. The Diagram menu is used for building database application.

9. The Tools menu gives you access to ways of adding procedures and menus to your programs

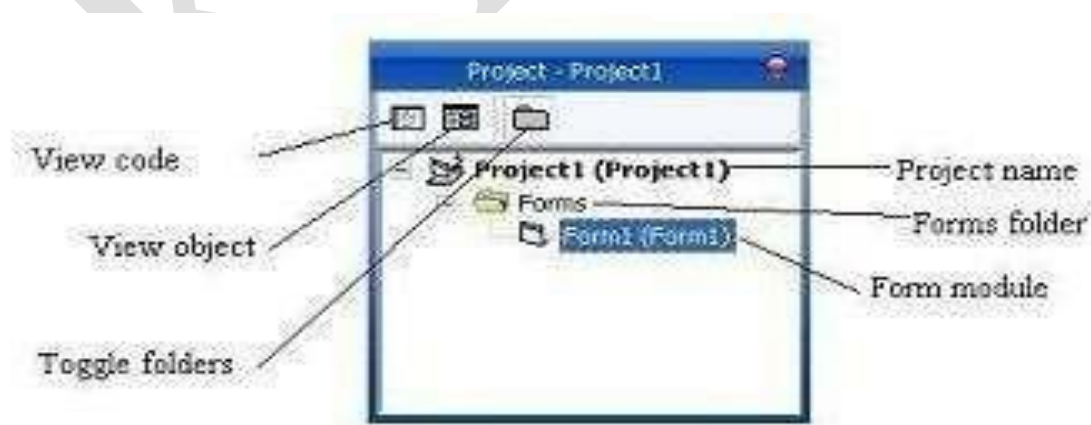
10. The Add-Ins menu contains additional utilities called Add-Ins. By default you should have an option for Visual Data Manager and another for the Add-In Manager. Visual Data Manager is a simple but useful tool that allows you to design and populate a database in many popular formats, including Microsoft Access. The Add-In Manager allows you to select other Add-In utilities to be added to the Add-Ins menu.

11. The Window menu lets you control how the windows that make up the visual Basic environment are arranged.

12. Finally, the Help menu is your second stop when you get in a jam.

Notice that all menus have one letter underlined. Pressing ALT and the underlined letter open that menu.

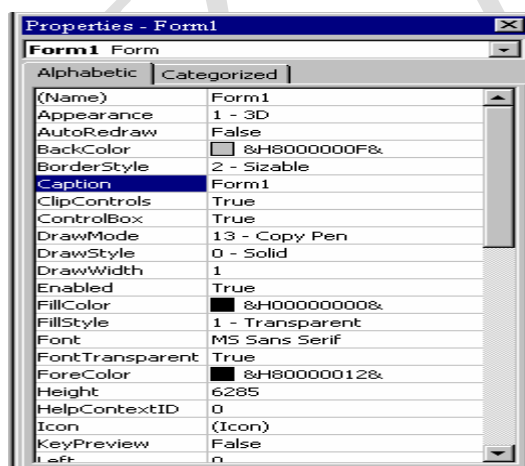
Project Explorer



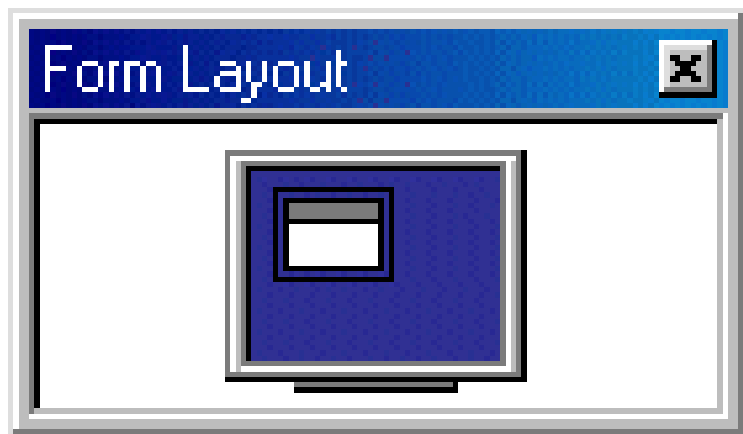
Docked on the right side of the screen, just under the tool bar is the Project Explorer Window. The Project Explorer Window as shown above serves as a quick reference to the various elements of a project namely for, classes and modules. The entire object that makes up the application is packed in a project. It looks like a tree-like structure. A simple project will typically contain one form, which is a window that is designed as part of a program's interface. The three tools in the top of the Project Explorer are described in the following table.

Properties Window

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



Form Layout Window

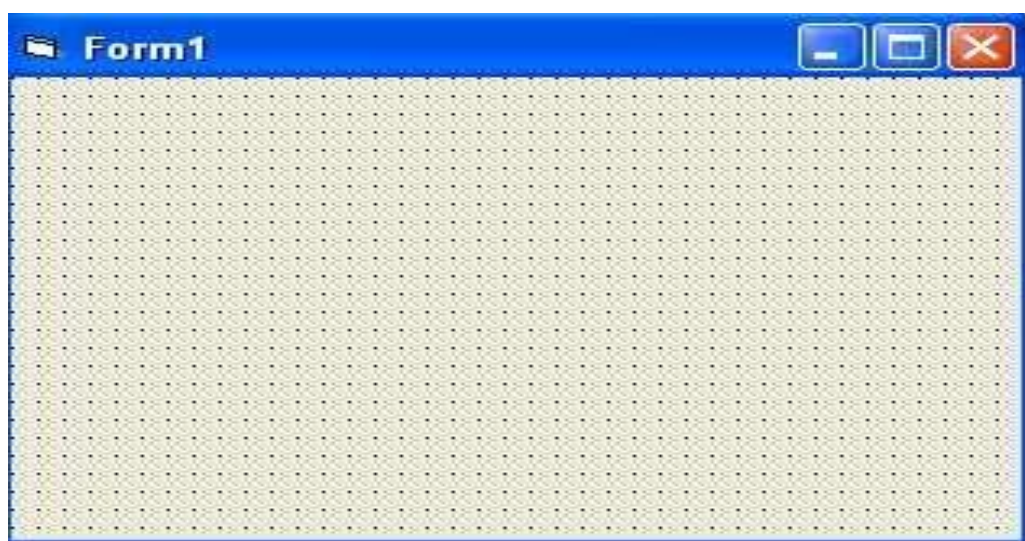


The Form Layout window allows you to position the location of the form at run time relative to the entire screen using a small graphical representation of the screen.

Tool Box

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

Form Designer



It is used to design the interface application. We add controls, graphics and pictures to a form to create the way we want. Each form in the application has its own form designer windows.

Object Browser

The Object Browser allows us to browse through the various properties, events and methods that are made available to us. It is accessed by selecting Object Browser from the View menu or pressing the key F2. The left column of the Object Browser lists the objects and classes that are available in the projects that are opened and the controls that have been referenced in them. It is possible for us to scroll through the list and select the object or class that we wish to inspect. After an object is picked up from the Classes list, we can see its members (properties, methods and events) in the right column. A property is represented by a small icon that has a hand holding a piece of paper. Methods are denoted by little green blocks, while events are denoted by yellow lightning bolt icon.

Tool Bars

The toolbars gives you quick access to commonly used menu commands. The Visual Basic IDE provides additional toolbars for specific purpose, such as editing, form designing and debugging. To view the additional toolbars, choose View > Toolbars

Visual Basic supports multiple toolbars which are split into four main areas such as,

Standard toolbar Edit toolbar Debug toolbar Form Editor

Standard Tool Bar

The Standard tool bar is displayed by default and offers quick access to frequently used functions.

Left

Right



Edit Tool Bar

The Edit tool bar's buttons are handy when you're debugging your program.

Left

Right



Debug Tool Bar

The debug tool bar has buttons for use when you're debugging your programs



Form Editor Tool Bar

The Form Editor Toolbar contains buttons that help you tweak the appearance of the controls on your forms.

left

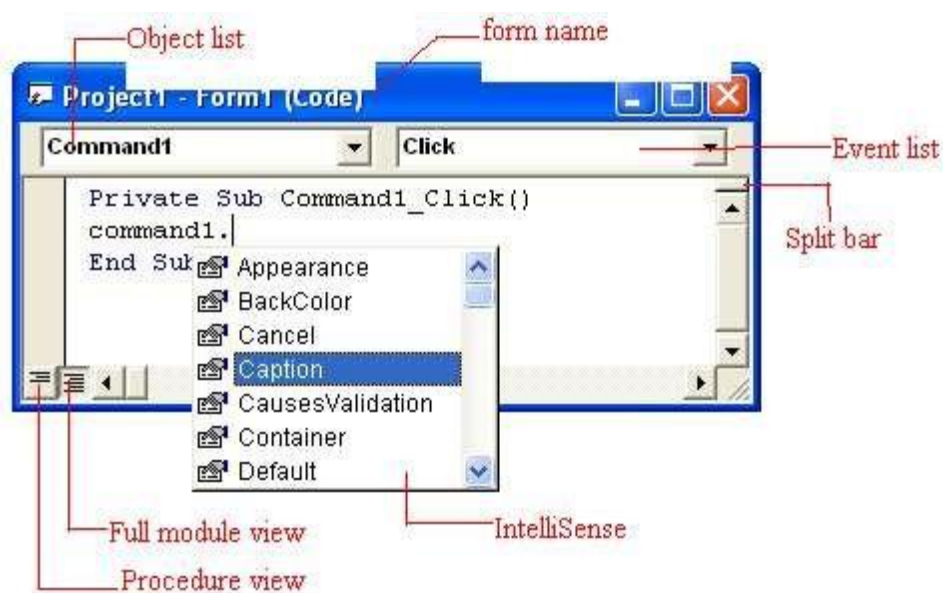
right



The Code Window

The Code Window is where you write Visual Basic Code for your application. Code consists of languages of statements, constants and declarations. Using the code window you quickly view and edit any of the code in your application. The Code window opens whenever you double-click a control or form or From Project Explorer Window select the name of a form or module or you can choose the View Code icon or View Code.

The Code Window has the following sections see figure



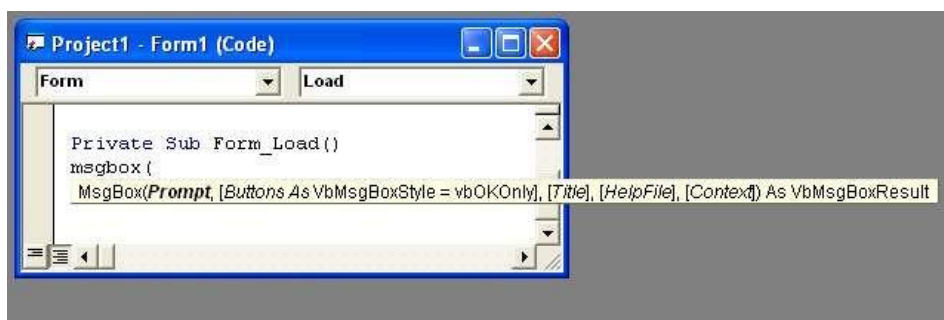
The Split Bar: The Split bar is located below the title bar at the top of the vertical scroll bar. Split bar is used when complicated codes are performed in your application. That is the window is splitted into two different parts of your code at once.

The Object List Box: The left drop-down list box in the code window, called the object box. Lists all the objects on the form.

The Event List Box: The right drop-down list box in the code window, called the Event List box. This Box gives the events recognized by the object you have selected in the object list box.

IntelliSense: IntelliSense saves you a lot of typing work. Its purpose is to display a pop up little boxes with helpful information about the object you are working with. It works with three components.

Quickinfo: You get the information about the syntax. Whenever you enter a keyword followed by a space or opening parenthesis, a tip appears and gives the syntax for that element. For Example, Quickinfo feature work for the MsgBox Statement, which pops up a simple box.



List Properties /Methods: This feature gives you a list of the properties and methods of an object right after you type the period. See figure

Available constant: If you had a label named Label1 on your form and you entered

Label1.Visible= True

You would see a pop up dialog box listing True or False (select any one from the list and press TAB to complete it)



KARPAGAM ACADEMY OF HIGHER EDUCATION
 (Deemed University Established Under Section 3 of UGC Act 1956)
 Coimbatore - 641021.
 (For the candidates admitted from 2017 onwards)
Department of Commerce (CA)

SUBJECT: SOFTWARE DEVELOPMENT USING VISUAL BASIC

SEMESTER : V

SUBJECT CODE: 16CCU501A

CLASS : III B.COM CA

UNIT:I

S.NO	QUESTIONS	OPTION 1	OPTION 2	OPTION 3	OPTION 4	ANSWER
1	visual basic was developed form the programming language	FORTTRAN	C	BASIC	C++	BASIC
2	visual bais cis developed in the year	1978	1980	1970	1972	1970
3	A complete installation of the most powerful version of visual basic 6.0, the enterprise edition require _____ of hard disk space	250MB	250GB	460MB	460GB	250MB
4	Visual Basic 6.0 requires _____ of RAM	18MB	16MB	18GB	16GB	16MB
5	IDE stands for _____	Integrated development	Integrated developed environment	Integration development environment	none	Integrated development environment
6	IDE is also commonly referred to as _____ environment	interface	development	design	integrated	design
7	IDE was designed as a _____	OLE	SDI	MDI	none	SDI

8	The windows associated with the project will stay with in a single container called as _____	child	parent	code	none	parent
9	The _____ in the menubar provide quick access to the commonly used commands	toolbars	toolbox	project window	form	toolbars
10	_____ helps to move and resize the controls and forms	label	shape	OLE	pointer	pointer
11	_____ carries out the specified action when the user choose it	option button	image control	shape control	command button	command button
12	_____ is the control used to display messages and enter text	tool box	text box	command button	option button	text box
13	_____ also acts as a coontainer for child forms and some controls	SDI	MDI	IDE	none	MDI
14	_____ is a window that contains an application code and has other objects place on it to create the use interface	properties	menu	form	border	form
15	_____ is an action that can be performed on objects	form	move	click	method	method
16	_____ control enables the user to connect to an existing database and display information from it.	object	properties	data	file	data
17	_____ serves as a window that can be customized and controls graphics and pictures can also be added to it.	object	properties	form	file	form
18	Each and every form in an application is considered an _____	object	properties	file	none	object
19	Characteristics of an object are called its _____	objects	properties	project	none	properties
20	_____ method is used to display the form object	load	visible	show	display	show
21	_____ method is used to load a form or control into memory but	load	visible	show	display	load

	doesnot display it					
22	_____ displays a text that the user cannot modify or interact with	label	text box	timer	line	label
23	_____ has the zorder event in it.	list box	Ms flex grid	grid	label	list box
24	_____ displays and operates on tabular data	grid	Msflexgrid	DB grid	none	Msflexgrid
25	_____ displays information at run time	status bar	option button	text box	rich text box	text box
26	_____ control is used to display icons, bitmaps, metafiles etc....,	shape control	image control	picture box	check box	image control
27	The code entered in the _____ event fires when there is a change in the contents of the text box.	change	click	mouse move	none	change
28	The _____ event fires when the textbox control is clicked.	change	click	mouse move	none	click
29	The _____ event fires when the mouse is moved over the textbox	change	click	mouse move	none	mouse move
30	A complete repaint of a form or control can be enforced by _____ method	refresh	set focus	change	repaint	refresh
31	_____ is used to link or embed object, display and manipulate data from other windows based application	.VBP	.FRM	OLE	none	OLE
32	_____ is the named attribute of a programming object	property	window	object	none	property
33	_____ is the named attribute of a programming object	window	object	property	none	property
34	_____ lets windows decide whrer the form should be shown	center screen	manual	center owner	windows default	windows default
35	_____ indicates whether the window is shown normally, maximized or minimized.	window state	windows default	both	none	windows default
36	_____ event occurs when the form is closed by user.	load	unload	show	none	unload
37	Writing a VB program involves _____ steps	2	5	3	4	2

38	_____ steps involves designing an application with various tools that come along with VB package	visual programming	code programming	both	none	visual programming
39	_____ steps involves writing programs using text editor.	visual programming	code programming	both	none	code programming
40	_____ is the name property for the form object	display prog	code programming	form display	none	code programming
41	_____ is the caption property for the command button if name is cmdexit	&exit	& clear	&display	none	&exit
42	_____ is the name property for the command button if caption property is &clear.	cmdexit	cmdclear	cmddisplay	none	cmdclear
43	_____ is the caption property for the command button if name is cmddisplay	&exit	& clear	&display	none	&display
44	_____ is the name property for the command button if caption is &display	cmdexit	cmdclear	cmddisplay	none	cmddisplay
45	VB project files are saved with an _____ extension	.VBP	.FRM	.OLE	.EXE	.VBP
46	VB form files are saved with an _____ extension	.VBP	.FRM	.OLE	.EXE	.FRM
47	_____ box displays the name of the selected object associated with the form	object	tool	text	command	object
48	Intersection of row and column is _____	cell	record	field	none	cell
49	the cols and rows properties are used to determine the number of columns and rows in a _____ control	flexgrid	ms flexgrid	grid	none	ms flexgrid
50	_____ event occurs whenever the size of form is ahanged	size	resize	height	width	resize
51	_____ kinds of rows/columns are created in the msflexgrid contorl.	4	3	2	1	2
52	Ms flexgrid control is an _____ control	OCX	XCO	COX	XOC	OCX
53	user can change the current cell at run time using _____	mouse	arrow keys	both	none	both
54	To display the wrapped text, we need to _____ the cell's column width	increase	decrease	normal	hide	increase

55	If a cell's text is too long to be displayed in the cell, and the word wrap property is set to _____	FALSE	TRUE	0	1	TRUE
56	To add the flexgrid, choose compliments from _____ menu	browser	property	project	none	project
57	After setting the properties, the ms flexgrid control is enlarged vertically and horizontally by _____ its handles.	moving	resizing	hiding	dragging	dragging
58	Ms flexgrid has the _____ properties	row	rows	both	none	both
59	Ms flexgrid has the _____ properties	col	cols	both	none	both
60	_____ property is set to change the height of a cell	row height	height row	height	none	row height

UNIT-II

SYLLABUS

Variables – Constants – Literals – Data Types – Operators – Sub routines and Functions
Program Flow Control – String Functions – Numeric Functions – Date Functions.

VARIABLES

Variables used to store information in the computer's memory while running the programs. There are three components to define a variable,

- The Variable's name
- The type of information being stored
- The actual information itself

A Variable must have a name for you to be able to assign values to it. There are rules you should follow while naming your variables. The name of a variable:

- Must begin with a letter
- Cannot have a period (remember that we use the period to set a property; in other words the period is an operator)
- Can have up to 255 characters.
- Must be unique inside of the procedure or the module it is used in (we will learn what a module is)

Once a variable has a name, you can use it as you see fit. For example, you can assign it a value and then use the variable in your program as if it represented that value.

Creating/ Declaring Variables

When a variable is created within a program, it is said to be "declared". When declaring a variable, you must give it a name and a type. Based on the type, the variable will automatically be loaded with some default data. Your first variable will be named "x" and will hold a number. Consider the follow code,

```
Private Sub  
Form_Load() Dim x  
As Integer  
End Sub
```

Analysis

The previous code consists of 3 lines. We will only take interest in the second line for now. This statement is made up of 4 words. "Dim" stands for "Dimension", and instructs your computer that you would like to create a new variable. "x" is the name of the variable. You must separate "Dim" and the name of your variable with a space. "As" is an instruction to your computer that you would now like to define the type of the new variable. "Integer" is the

name of a type that is built-in to Visual Basic. "As" must be separated from the variable name and the variable type by a space.

Output

When you launch this program with F5, nothing will appear to happen before the form loads. However, a variable has been created in memory called "x" with a type of "Integer".

Data Types

• Byte

The Byte data type is an 8-bit variable which can store value from **0 to 255**. This data type is very useful for storing binary data.

• Double

. The Double data type is a 64-bit floating point number used when high accuracy is needed. These variables can range from - 1.79769313486232e308 to - 4.94065645841247e-324 for negative values and from 4.94065645841247e-324 to 1.79769313486232e308 for positive values

• Integer

The Integer data type is a 16-bit number which can range from **-32768 to 32767**. Integers should be used when you are working with values that can not contain fractional numbers.

• Long

The Long data type is a 32-bit number which can range from **-2,147,483,648 to 2,147,483,647**. Long variables can only contain non fractional or whole integer values. I myself use Long variables over Integers for increased performance. Most Win32 functions use this data type for this reason.

Variables	Character
Integer	%
Long	&
Single	!
Double	#
Currency	@
String	\$
Byte	None
Boolean	None
Date	None
Object	None
Variant	None

• Single

The Single data type is a 32-bit number ranging from -3.402823e38 to -1.401298e-45 for negative values and from 1.401298e-45 to 3.402823e38 for positive values. When you need fractional numbers within this range, this is the data type to use.

• String

The String data type is usually used as a variable-length type of variable. A variable-length string can contain up to approximately 2 billion characters. Each character has a value ranging from 0 to 255 based on the ASCII character set. Strings are used when Text is involved.

Note: Variant variables can store any type of data.

Declaring a Variable

There are two ways to declare a variable in Visual Basic

1. Explicit Declarations – These statements do not assign values to the variables but merely tell Visual Basic what the variables should be called and what type of data they can contain.
2. Implicit Declarations – with this type of declarations, a special character is used at the end of the variable name when the variable is first assigned a value

Eg: nNumVal%=0
Sfirstname\$=" Gary
Cornel" VInpval = 5

Notice that the variable VInpval doesn't have a declarations character. Such a type is named by Variant type.

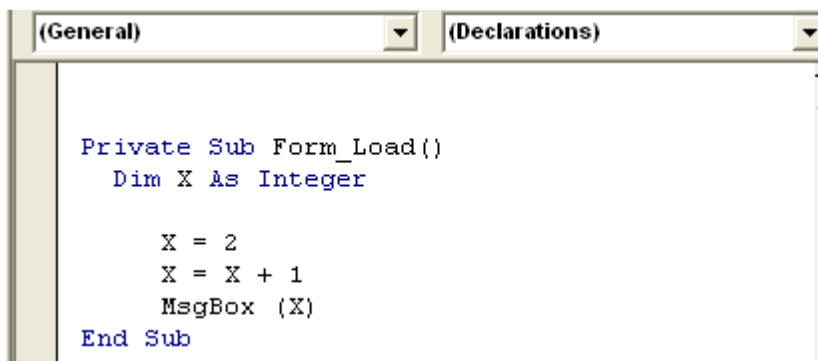
- Fixed –Length Strings – A fixed-length string remains the same size, regardless of the information assigned to it.

Syntax: Dim varname As string * strlength

Assigning a value less than 15 characters in length to sName would make the remaining character spaces

Eg:1 Assigning and Displaying the result

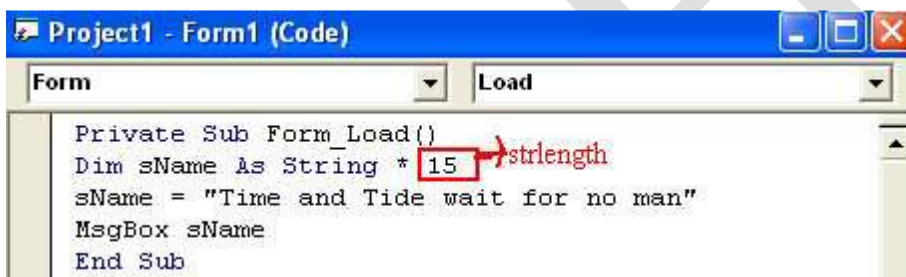
To give a variable some new data, you can use the assignment operator ("="). For example, declaring x as an Integer variable and loading it with the value 2 would be accomplished in this way:



```
(General) (Declarations)

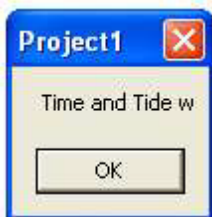
Private Sub Form_Load()
    Dim X As Integer

    X = 2
    X = X + 1
    MsgBox (X)
End Sub
```

**Eg:2 Assigning and Displaying the result**

```
Project1 - Form1 (Code)
Form Load

Private Sub Form_Load()
    Dim sName As String * 15
    sName = "Time and Tide wait for no man"
    MsgBox sName
End Sub
```



Constants

A constant is a meaningful name that takes the place of a number or string that does not change. Constants store values that, as the name implies, remain the same throughout the execution of an application. You can greatly improve the readability of your code and make it easier to maintain by using constants. Use them in code that contains values that reappear or that depends on certain numbers that are difficult to remember or have no obvious meaning.

Declaring a Constant

The Const keyword is used to declare a constant and set its value. By declaring a constant, you assign a meaningful name to a value. Once a constant is declared, it cannot be modified

or assigned a new value. The constant must have a valid symbolic name (the rules are the same as those for creating variable names) and an expression composed of numeric or string constants and operators (but no function calls). For example consider the following

Public Const DaysInYear = 365

Private Const WorkDays = 250

Operators

An operator is a code element that performs an operation on one or more code elements that hold values. Value elements include variables, constants, literals, properties, returns from Function and Operator procedures, and expressions.

An **expression** is a series of value elements combined with operators, which yields a new value. The operators act on the value elements by performing calculations, comparisons, or other operations.

Types of Operators

Visual Basic provides the following types of operators:

- **Arithmetic Operators** perform familiar calculations on numeric values, including shifting their bit patterns.
- **Comparison Operators** compare two expressions and return a Boolean value representing the result of the comparison.
- **Concatenation Operators** join multiple strings into a single string.
- **Logical and Bitwise Operators** in Visual Basic combine Boolean or numeric values and return a result of the same data type as the values.

The value elements that are combined with an operator are called *operands* of that operator. Operators combined with value elements form expressions, except for the assignment operator, which forms a *statement*.

Arithmetic Operator

Operator	Meaning	Example
+	Addition	A=3+5 => A=8
-	Subtraction	B=6-4 => B=2

KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: III BCom CA COURSE NAME: SOFTWARE DEVELOPMENT WITH VISUAL BASIC
COURSE CODE: 16CCU501A UNIT: II BATCH-2016-19

*	Multiplication	$C=5*7 \Rightarrow C=35$
/	Division	$D=5/2 \Rightarrow D=2.5$
^	Exponent	$E=2^2 \Rightarrow E=4$
%	Modulus	$F=5\%4 \Rightarrow F=1$
	Integer Division	$G=5 \ 2 \Rightarrow G=2$

Comparison Operator

Operator	Meaning	Example
>	Lessthan	$8 > 7 \Rightarrow \text{False}$
<	Greaterthan	$8 < 7 \Rightarrow \text{True}$
<=	Lessthan or Equals to	$8 <= 7 \Rightarrow \text{False}$
>=	Greaterthan or Equals to	$8 >= 8 \Rightarrow \text{True}$
= =	Equality	$9 = = 8 \Rightarrow \text{False}$
< >	Inequality	$9 < > 8 \Rightarrow \text{True}$

Concatenation Operator

Concatenation operators join multiple strings into a single string. There are two concatenation operators, '+' and '&'. Both carry out the basic concatenation operation, as the following example shows.

```
Dim x As String = "Mic" & "ro" & "soft"
```

```
Dim y As String = "Mic" + "ro" + "soft"
```

' The preceding statements set both x and y to "Microsoft".

Sub Routine and Functions

Subroutines is a sub or miniature programs. A subroutine has a name attributed with it, much like a variable does. Unlike a variable, a subroutine doesn't hold data. Instead, it holds code. When the code in a subroutine is executed, the subroutine is said to be "called". Therefore, one subroutine can "call" another subroutine. Some subroutines are called automatically when certain actions are performed. For example, the Form_Load subroutine is automatically called when a form loads. This is where you have been inserting your test code before.

Creating a subroutine involves two lines of code. A subroutine begins with the word "Sub", followed by a space, then a name identifying the subroutine. Two parentheses follow, which are used for a parameter list.

Syntax:**Sub Test()****End Sub****Example****Sub TestSub()**

MsgBox "code in TestSub()"

End Sub**Private Sub Form_Load()**

MsgBox "Code in Form_Load()"

TestSub

MsgBox "Back in Form_Load()"

End Sub**Passing Parameter to the sub routines**

Parameters, also called Arguments, are variables that can be "passed into" a subroutine. A subroutine with parameters example

Private Sub DisplayAdd(x As Integer, y As Integer)

MsgBox x + y

End Sub**Private Sub Form_Load()**

DisplayAdd 5, 2

End Sub

A new subroutine has been declared called DisplayAdd. This declaration is different than the declarations that you have seen so far, however, as code has been added between the parenthesis. From your knowledge of variables, this syntax should look somewhat similar to you. `x As Integer` and `y As Integer` appear to be variable declarations without the "Dim" keyword. These declarations are separated by a comma. These variables are the Parameters for the DisplayAdd subroutine. Code within the subroutine can access x and y as usual, as if they were normal variables. However, when the subroutine is called, the calling subroutine will also provide values for these parameters. Therefore, the subroutine has now become dynamic. The code can act on input without caring where the input came from. When the Form_Load subroutine calls DisplayAdd with the parameters 5 and 2, the code within DisplayAdd is executed. The first line adds x and y together and displays the result. x and y have already been filled with the values 5 and 2 from the Form_Load subroutine.

Call by Value and Call by Reference

Parameters can be sent to a subroutine By Reference (ByRef) or By Value (ByVal). ByRef is the default, and means that changes to the variable in the subroutine will result in changes to the source variable outside of the subroutine. ByVal literally copies the values of the variables from the calling subroutine into the called subroutine. By doing this, the variables can be changed, but their values will not change outside of the called subroutine. ByVal can also be a lot slower with large variable types, however, since memory has to be copied from one location to another. If you don't have any reason to do so, there is no need to pass variables ByVal. You can explicitly state the way that a variable is passed to a subroutine by using these keywords before the variable name. Using the ByRef keyword, one could write a Swap function, which switches the values of 2 variables.

```
Private Sub Swap(ByRef x As Integer, ByRef y As Integer)
```

```
    Dim temp As Integer
```

```
    temp = x
```

```
    x = y
```

```
    y = temp
```

```
End Sub
```

```
Private Sub DisplayVals(ByRef a As Integer, ByVal b As Integer)
```

```
    'Don't worry about understanding the next line yet, it will be explained later
```

```
    MsgBox "a = " & CStr(a) & vbCrLf & "b = " & CStr(b)
```

```
End Sub
```

```
Private Sub Form_Load()
```

```
    Dim a As Integer
```

```
    Dim b As Integer
```

```
    a = 10
```

```
    b = 12
```

```
    'Display values, swap, and display again
```

```
    DisplayVals a, b
```

```
    'The next line is functionally identical to "Swap a, b"
```

```
    Call Swap(a, b)
```

```
    DisplayVals a, b
```

```
End Sub
```

Notice that Call was used instead of simply stating the subroutine name. When using the Call method however, you must use parenthesis when calling the subroutine. Note that this program would also have worked without typing "ByRef" anywhere, since it is the default. The ByRef and ByVal keywords are rarely used in simple programs, however, but it's a nice trick for your toolkit.

Functions

Subroutines have a close cousin called Functions. Functions are basically the exact same as subroutines, except that they return a value. That means that the function itself has a type, and the function will return a value to the calling subroutine based on the code that it contains. An example of this would be a function that adds two numbers, shown below. A function is declared the exact same way as a subroutine, except using the "Function" keyword instead of "Sub". To return a value, assign a value of the proper type to the function's name, as if it were a variable.


```
Private Function Add(ByVal x As Integer, ByVal y As Integer) As Integer
```

```
    Dim Res as integer
```

```
    Res = x + y
```

```
    Add = Res
```

```
End Function
```

```
Private Sub Form_Load()
```

```
    Dim a As Integer
```

```
    Dim b As Integer
```

```
    Dim c As Integer
```

```
    a = 32
```

```
    b = 64
```

```
    c = Add(a, b)
```

```
    MsgBox ("Sum is : " & c)
```

```
End Sub
```

Control Statements

Control Statements are used to control the flow of program's execution. Visual Basic supports control structures such as if... Then, if...Then ...Else, Select...Case, and Loop structures such as Do While...Loop, While...Wend, For...Next etc method.

1. If...Then selection structure

The If...Then selection structure performs an indicated action only when the condition is True; otherwise the action is skipped.

Syntax

```
If <condition> Then
```

```
statement
```

```
End If
```

Example

```
If average>75 Then
```

```
txtGrade.Text = "A"
```

```
End If
```

The above example checks the condition and if it's true then assigns the grade A to the text box txtGrade.

2. If...Then...Else selection structure

The If...Then...Else selection structure allows the programmer to specify that a different action is to be performed when the condition is True than when the condition is False.

Syntax

If <condition > Then

statement 1

Else

statement 2

End If

If the condition is true then statement 1 will be executed if its false statement 2 will be executed.

Example

If average>50 Then

txtGrade.Text = "Pass"

Else

txtGrade.Text = "Fail"

End If

In the above example if the average is above 50 text box txtGrade will be assigned Pass or if the condition is false then text box txtGrade will be assigned Fail.

3. Nested If...Then...Else selection structure

Nested If...Then...Else selection structures test for multiple cases by placing If... Then.. .Else selection structures inside If...Then...Else structures.

Syntax of the Nested If...Then...Else selection structure

If < condition 1 > Then

statements

Else If < condition 2 > Then

statements

Else If < condition 3 > Then

statements

Else

Statements

End If

Example

Assume you have to find the grade using nested if and display in a text box

If average > 75 Then

txtGrade.Text = "A"

Else If average > 65 Then

txtGrade.Text = "B"

Else If average > 55 Then

txtGrade.text = "C"

Else If average > 45 Then

txtGrade.Text = "S"

Else

txtGrade.Text = "F"

End If

In the above example the average is check and according the grades A, B, C, S, F are assigned to the text box named txtGrade.

4. Select...Case selection structure

Select...Case structure is an alternative to If...Then...ElseIf for selectively executing a single block of statements from among multiple block of statements. Select...case is more convenient to use than the If...Else...End If. The following program block illustrate the working of Select...Case.

Syntax

Select Case Index

Case 0

Statements

Case 1

Statements

End Select

According to the index value any one of the case statement will be selected and executed.

Example

Assume you have to find the grade using select...case and display in the text box

Dim average as Integer

average = txtAverage.Text

Select Case average

Case 100 To 75

txtGrade.Text ="A"

Case 74 To 65

txtGrade.Text ="B"

Case 64 To 55

txtGrade.Text ="C"

Case 54 To 45

txtGrade.Text ="S"

Case 44 To 0

txtGrade.Text ="F"

Case Else

MsgBox "Invalid average marks"

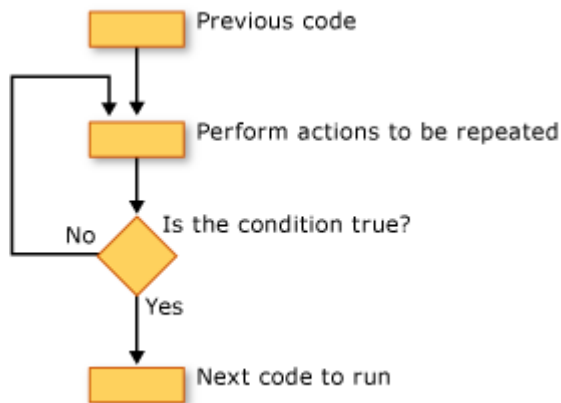
End Select

Here average is the case index according to the average value any one of the case statement is selected and executed.

Looping Statements

Visual Basic loop structures allow you to run one or more lines of code repetitively. You can repeat the statements in a loop structure until a condition is True, until a condition is False, a specified number of times, or once for each element in a collection.

The following illustration shows a loop structure that runs a set of statements until a condition becomes true.



Running a set of statements until a condition becomes true

1. While loop

The While ... End While construction runs a set of statements as long as the condition specified in the While statement is True. Runs a series of statements as long as a given condition is true.

Syntax

While condition

[statements]

[Continue While]

[statements]

[Exit While]

[statements]

End While

Term	Definition
condition	Required. Boolean expression. If condition is Nothing, Visual Basic treats it as False.
statements	Optional. One or more statements following While, which run every time condition is True.
Continue	Optional. Transfers control to the next iteration of the While block.

Term	Definition
While	
Exit While	Optional. Transfers control out of the While block.
End While	Required. Terminates the definition of the While block.

Example:

Dim value, i As Integer

i=1

value = 0

While i<=10

value = value + i

i = i + 1

End While

The above example counts the numbers from 1 to 10 and stores in variable named value.

2. Do While Loop

The **Do...Loop** While statement first executes the statements and then test the condition after each execution.

Syntax

Do

[statements]

[Continue Do]

[statements]

[Exit Do]

[statements]

Loop While condition

KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: III BCom CA COURSE NAME: SOFTWARE DEVELOPMENT WITH VISUAL BASIC

COURSE CODE: 16CCU501A

UNIT: II

BATCH-2016-19

Term	Definition
Do	Required. Starts the definition of the Do loop.
While	Required unless Until is used. Repeat the loop until condition is False.
condition	Optional. Boolean expression. If condition is Nothing, Visual Basic treats it as False.
statements	Optional. One or more statements that are repeated while, or until, condition is True.
Continue Do	Optional. Transfers control to the next iteration of the Do loop.
Exit Do	Optional. Transfers control out of the Do loop.
Loop	Required. Terminates the definition of the Do loop.

Example

Dim number As Long

number = 0

Do

number = number + 1

Loop While number < 201

The program executes the statements between Do and Loop While structure in any case. Then it determines whether the counter is less than 501. If so, the program again executes the statements between Do and Loop While else exits the Loop.

3. Do Until Loop

Unlike the **Do While...Loop**, the **Do Until... Loop** structure tests a condition for falsity. Statements in the body of a **Do Until...Loop** are executed repeatedly as long as the loop-continuation test evaluates to False.

Syntax**Do Until condition**

[statements]

[Continue Do]

[statements]

[Exit Do]

[statements]

Loop

Term	Definition
Do	Required. Starts the definition of the Do loop.
Until	Required unless Until is used. Repeat the loop until condition is False.
condition	Optional. Boolean expression. If condition is Nothing, Visual Basic treats it as False.
statements	Optional. One or more statements that are repeated while, or until, condition is True.
Continue Do	Optional. Transfers control to the next iteration of the Do loop.
Exit Do	Optional. Transfers control out of the Do loop.
Loop	Required. Terminates the definition of the Do loop.

Example

```
Dim number As Long
number=0
Do Until number > 1000
number = number + 1
Print number
Loop
```

Numbers between 1 to 1000 will be displayed on the form as soon as you click on the command button.

4. For Loop

The For....Next construction performs the loop a set number of times. It uses a loop control variable, also called a counter, to keep track of the repetitions. You specify the starting and ending values for this counter, and you can optionally specify the amount by which it increases from one repetition to the next.

For counter [As datatype] = start To end [Step step]

[statements]

[Continue For]

[statements]

[Exit For]

[statements]

Next

Part	Description
counter	Required in the For statement. Numeric variable. The control variable for the loop. For more information, see Counter Argument later in this topic.
datatype	Optional. Data type of counter. For more information, see Counter Argument later in this topic.
start	Required. Numeric expression. The initial value of counter.
end	Required. Numeric expression. The final value of counter.
step	Optional. Numeric expression. The amount by which counter is incremented each time through the loop.

Part	Description
statements	Optional. One or more statements between For and Next that run the specified number of times.
Continue For	Optional. Transfers control to the next loop iteration.
Exit For	Optional. Transfers control out of the For loop.
Next	Required. Terminates the definition of the For loop.

Example**Dim x As Integer****For x = 1 To 50****Print x****Next**

The above for loop prints the numbers from 1 to 10 till the condition returns false.

Other Statement in VB**Exit Statement**

Exit Statement just exits a procedure or block and transfers control immediately to the statement following the procedure call or the block definition.

Syntax**Exit { Do | For | Function | Select | Sub | While }****Statements****Exit Do**

Immediately exits the Do loop in which it appears. Execution continues with the statement following the Loop statement. Exit Do can be used only inside a Do loop. When used within nested Do loops, Exit Do exits the innermost loop and transfers control to the next higher level of nesting.

Exit For

Immediately exits the For loop in which it appears. Execution continues with the statement following the Next statement. Exit For can be used only inside a For...Next loop.

Exit Function

Immediately exits the Function procedure in which it appears. Execution continues with the statement following the statement that called the Function procedure. Exit Function can be used only inside a Function procedure.

Exit Select

Immediately exits the Select Case block in which it appears. Execution continues with the statement following the End Select statement. Exit Select can be used only inside a Select Case statement.

Exit Sub

Immediately exits the Sub procedure in which it appears. Execution continues with the statement following the statement that called the Sub procedure. Exit Sub can be used only inside a Subprocedure.

In a Sub procedure, the Exit Sub statement is equivalent to the Return statement.

Exit While

Immediately exits the While loop in which it appears. Execution continues with the statement following the End While statement. Exit While can be used only inside a While loop. When used within nested While loops, Exit While transfers control to the loop that is one nested level above the loop where Exit While occurs.

Example

```
Dim index As Integer = 0
```

```
Do While index <= 100
```

```
    If index > 10 Then
```

```
        Exit Do
```

```
    End If
```

```
    Debug.Write(index.ToString & " ")
```

```
    index += 1
```

```
Loop
```

```
Debug.WriteLine("")
```

```
' Output: 0 1 2 3 4 5 6 7 8 9 10
```

The above example, the loop condition stops the loop when the index variable is greater than 100. The If statement in the loop, however, causes the Exit Do statement to stop the loop when the index variable is greater than 10.

Goto Statement

GoTo statement branches unconditionally the flow of the control to a specified line in a procedure.

Syntax**GoTo line**

Here, line requires any line label.

The GoTo statement can branch only to lines in the procedure in which it appears. The line must have a line label that GoTo can refer to.

Example

Sub gotoStatementDemo()

Dim number As Integer = 1

Dim sampleString As String

' Evaluate number and branch to appropriate label.

If number = 1 Then GoTo Line1 Else GoTo Line2

Line1:

sampleString = "Number equals 1"

GoTo LastLine

Line2:

sampleString = "Number equals 2"

LastLine:

Debug.WriteLine(sampleString)

End Sub

The above program shows the unconditional branching of the program flow using the goto statement.

End Statement

End statement terminates the procedure unconditionally.

Syntax**End****Example**

Sub sample()

MsgBox("Sample Program")

End

End Sub

The above program terminates once the message box is displayed to the user.

String Functions:

VB has numerous built-in string functions for processing strings. Most VB string-handling functions return a string, although some return a number (such as the Len function, which returns the length of a string and functions like Instr and InstrRev, which return a character position within the string). The functions that return strings can be coded with or without the dollar sign (\$) at the end, although it is more efficient to use the version with the dollar sign. Visual Basic string functions return a string, and usually accept one or more strings as arguments. The table below lists some of the common string functions.

Function	Description
<u>InStr</u>	Returns the position of the first occurrence of one string within another. The search begins at the first character of the string
<u>InStrRev</u>	Returns the position of the first occurrence of one string within another. The search begins at the last character of the string
<u>LCase</u>	Converts a specified string to lowercase
<u>Left</u>	Returns a specified number of characters from the left side of a string
<u>Len</u>	Returns the number of characters in a string
<u>LTrim</u>	Removes spaces on the left side of a string
<u>RTrim</u>	Removes spaces on the right side of a string
<u>Trim</u>	Removes spaces on both the left and the right side of a string
<u>Mid</u>	Returns a specified number of characters from a string
<u>Replace</u>	Replaces a specified part of a string with another string a specified number of times
<u>Right</u>	Returns a specified number of characters from the right side of a string
<u>Space</u>	Returns a string that consists of a specified number of spaces
<u>StrComp</u>	Compares two strings and returns a value that represents the result of the comparison
<u>String</u>	Returns a string that contains a repeating character of a specified length
<u>StrReverse</u>	Reverses a string
<u>UCase</u>	Converts a specified string to uppercase

Date and Time Function

KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: III BCom CA COURSE NAME: SOFTWARE DEVELOPMENT WITH VISUAL BASIC

COURSE CODE: 16CCU501A

UNIT: II

BATCH-2016-19

Visual Basic includes the date and time functions described in the table below. Note that the functions `Date()`, `Now()`, and `Time()` do not accept arguments, so parentheses are not actually required.

Function	Description
<u>CDate</u>	Converts a valid date and time expression to the variant of subtype Date
<u>Date</u>	Returns the current system date
<u>DateAdd</u>	Returns a date to which a specified time interval has been added
<u>DateDiff</u>	Returns the number of intervals between two dates
<u>DatePart</u>	Returns the specified part of a given date
<u>DateSerial</u>	Returns the date for a specified year, month, and day
<u>DateValue</u>	Returns a date
<u>Day</u>	Returns a number that represents the day of the month (between 1 and 31, inclusive)
<u>FormatDateTime</u>	Returns an expression formatted as a date or time
<u>Hour</u>	Returns a number that represents the hour of the day (between 0 and 23, inclusive)
<u>IsDate</u>	Returns a Boolean value that indicates if the evaluated expression can be converted to a date
<u>Minute</u>	Returns a number that represents the minute of the hour (between 0 and 59, inclusive)
<u>Month</u>	Returns a number that represents the month of the year (between 1 and 12, inclusive)
<u>MonthName</u>	Returns the name of a specified month
<u>Now</u>	Returns the current system date and time
<u>Second</u>	Returns a number that represents the second of the minute (between 0 and 59, inclusive)
<u>Time</u>	Returns the current system time
<u>Timer</u>	Returns the number of seconds since 12:00 AM
<u>TimeSerial</u>	Returns the time for a specific hour, minute, and second
<u>TimeValue</u>	Returns a time
<u>Weekday</u>	Returns a number that represents the day of the week (between 1 and 7, inclusive)
<u>WeekdayName</u>	Returns the weekday name of a specified day of the week
<u>Year</u>	Returns a number that represents the year

Numeric Function

KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: III BCom CA COURSE NAME: SOFTWARE DEVELOPMENT WITH VISUAL BASIC

COURSE CODE: 16CCU501A

UNIT: II

BATCH-2016-19

The table below lists some of the common mathematical functions which are used for numeric value evaluations.

Function	Description
<u>Abs</u>	Returns the absolute value of a specified number
<u>Atn</u>	Returns the arctangent of a specified number
<u>Cos</u>	Returns the cosine of a specified number (angle)
<u>Exp</u>	Returns e raised to a power
<u>Hex</u>	Returns the hexadecimal value of a specified number
<u>Int</u>	Returns the integer part of a specified number
<u>Fix</u>	Returns the integer part of a specified number
<u>Log</u>	Returns the natural logarithm of a specified number
<u>Oct</u>	Returns the octal value of a specified number
<u>Rnd</u>	Returns a random number less than 1 but greater or equal to 0
<u>Sgn</u>	Returns an integer that indicates the sign of a specified number
<u>Sin</u>	Returns the sine of a specified number (angle)
<u>Sqr</u>	Returns the square root of a specified number
<u>Tan</u>	Returns the tangent of a specified number (angle)



KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed University Established Under Section 3 of UGC Act 1956)

Coimbatore - 641021.

(For the candidates admitted from 2017 onwards)

Department of Commerce (CA)

SUBJECT: SOFTWARE DEVELOPMENT USING VISUAL BASIC

SEMESTER : V

SUBJECT CODE: 16CCU501A

CLASS : III B.COM CA

UNIT:II

S. NO	QUESTIONS	OPTION 1	OPTION 2	OPTION 3	OPTION 4	ANSWER
1	VB uses _____ blocks such as variables, data types etc...,	constructing	building	fundamental	specified	building
2	Code in VB is stored in the form of _____	modules	subroutines	variables	standards	modules
3	There are _____ kinds of modules	5	2	3	4	3
4	A simple application may contain a single form and the code resides in the _____ itself	form	module	form module	none	form module
5	_____ is the foundation of object oriented programming in VB	standard	class	form	none	class
6	_____ is the extension for class module	.frm	.CLS	.std	none	.CLS
7	_____ may include constant,type,variable and DLL procedure declarations.	declaration	procedure	variable	statement	declaration
8	By default VB variables are of the _____ data type.	const	static	variant	none	variant
9	_____ is the value range of byte	0 to 255	1 to 255	0 to 266	1 to 266	0 to 255
10	_____ is the value range of integer	-32767 to 32768	-32768 to 32767	32767 to - 32768	32768 to - 32767	-32768 to 32767
11	_____ are used for storing values temporarily.	character	constant	variable	module	variable
12	From the following which character is allowed while	%	@	_	\$	_

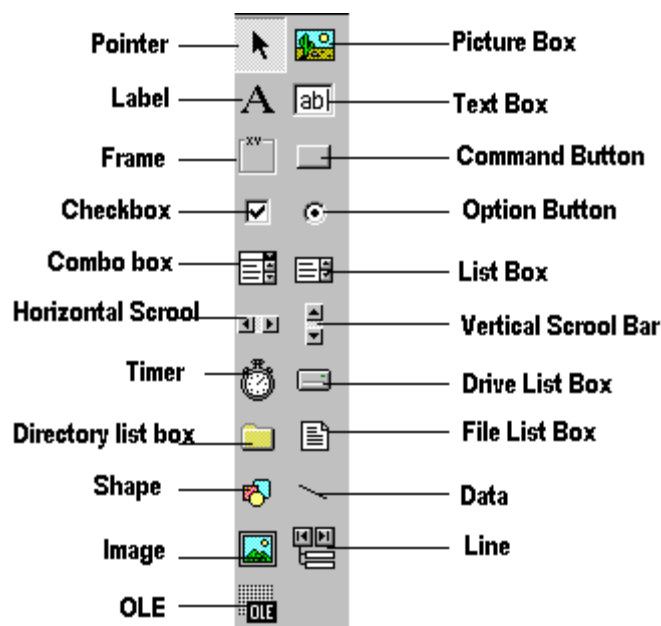
	declaring variables					
13	There are _____ ways for declaring variables	1	3	2	4	3
14	_____ statement checks in the module for usege for any undeclared variables.	option explicit	option implicit	int count	none	option explicit
15	_____ variable is one that is declared inside a procedure	global	local	static	scope	Local
16	Variables that are declared with keyword _____ exist only as long as the procedure is being executed	public	private	dim	double	Dim
17	To make all variables in a procedure static, _____ keyword is placed at the beginning of the procedure.	public	private	dim	static	Static
18	The module-level variable is available to all the procedures in the module. They are declared using _____ keywords	public	private	a and b	a or b	a or b
19	VB programs can be broken into smaller logical components called as _____	procedures	programs	sub	event	Procedures
20	Event procedures acquire the declaration _____ by default	public	private	static	global	Private
21	_____ window is opened for the module to which the procedures is to be added.	code	add	type	sub	Code
22	Functionns are especially useful for taking one or more pieces of data called as _____	modules	arguments	procedures	programs	Arguments
23	_____ statements are used to control the flow of program execution	control	property	while	do-while	Control
24	_____ block is used to define several blocks of statements, in order to execute one block	if then Else	if	then	if ... then	if then Else
25	_____ block is used for conditional execuion of one or ore statements	if then ... else	if	then	if ... then	if ... then
26	_____ is an alternative to If...Then....Else.	select...case	case...select	select	none	select...case
27	Select...case structure evaluates an expression _____ at the top of the structure	twice	once	thrice	none	Once
28	The _____ statement first executex the statemetn and then test the condition after each execution	do....while	while....do	select....case	while	do....while
29	_____ structure executes the statements until the condition is satisfied	do...loop	do..loop until	do while...loop	none	do..loop until
30	do...loop until is _____ loop	finite	infinite	long	small	Infinite

31	_____ function retrives only date	for...next	next...for	exit for	exit do	for...next
32	A_____ loop can be terminated by an exit for statement	for...next	next...for	exit for	for exit	for...next
33	do....while loop is terminated using _____ statement	exit for	for exit	exit do	do exit	exit do
34	A sequence of variables by the same name can be referred using _____	arrays	modules	sub-routines	none	Arrays
35	The individual element of an array are identified using _____	modules	index	base	none	Index
36	Fixed size of an array is always _____	remains same	changes	. increased	decreased	remains same
37	Size of dynamic array is changed at _____ time	execution	run	break	stop	Run
38	Variables of different datatypes when combined as a single variable to hold several related information is called a _____ datatype	user defined	user like	user string	none	user defined
39	_____ statement is used to create a constant.	constant	const	consta	constan	Const
40	The _____ function retrieves the date and time	Now	Date	Time	none	Now
41	_____ function retrives only time	Now	Date	Time	none	Time
42	_____ function retrives only date	Now	Date	Time	none	Date
43	_____ function returns the intervals between 2 dates in terms of years, months or dates.	format	diffdate	datediff	relational	datediff
44	_____ function accepts a numeric value and converts it to a string in format	specified	relational	logical	comparison	specified
45	_____ returns the variant (string) containing a specified number of characters from a string	left()	right()	mid()	instr()	mid()
46	The _____ function returns the length of the string	strcmp()	strrev()	len()	format()	len()
47	_____ is a group of controls that share the same name and type	control arrays	arrays	format arrays	none	control arrays
48	There are _____ ways to create a control array at design time	3	2	1	5	3
49	_____ is the value of the index in the array	TRUE	FALSE	0	index%	index%
50	Control arrays are added at run time using _____ statement.	load	unload	format	unformat	load
51	_____ statement removes a control from an array	load	unload	format	unformat	unload
52	_____ is a named sequence of statements executed as a unit	property	procedure	project	browser	procedure
53	A procedure name is always defined at _____ level	routine	subroutine	procedure	project	procedure

54	_____ is the list box at the upper-right corner of the code window and the debug window that displays the procedures recognized for the object displayed in the object box.	project file	property page	procedure box	none	procedure box
55	VB offers different types of _____ to execute small sections of coding in applications	project	property	procedure	none	procedure
56	Procedures used in one program can act as a _____ for other programs with slight modifications	building blocks	buildings	construction	module	building blocks
57	A _____ procedure is a procedure block that contains the control's actual name, an under score (_) and a event name.	sub	event	general	function	event
58	A function can also be written by selecting _____ procedure dialog box from the tools menu and by choosing the required scope and type	add	sub	mul	div	add
59	Boolean has _____ function	cbool	cbol	cboolean	cboolea	cbool
60	Decimal has _____ function	cdecimal	cdec	cdemal	cdec	cdec

UNIT-III**SYLLABUS**

Pointers – Label – Frame – Check Box – Compo Box – Scroll Bar – Timer – Shape and Line Control – Command Button – List Box – Image Box – Picture Box – SDI and MDI Form – Data Grid – Flex Grid – Menus – Dialog Box

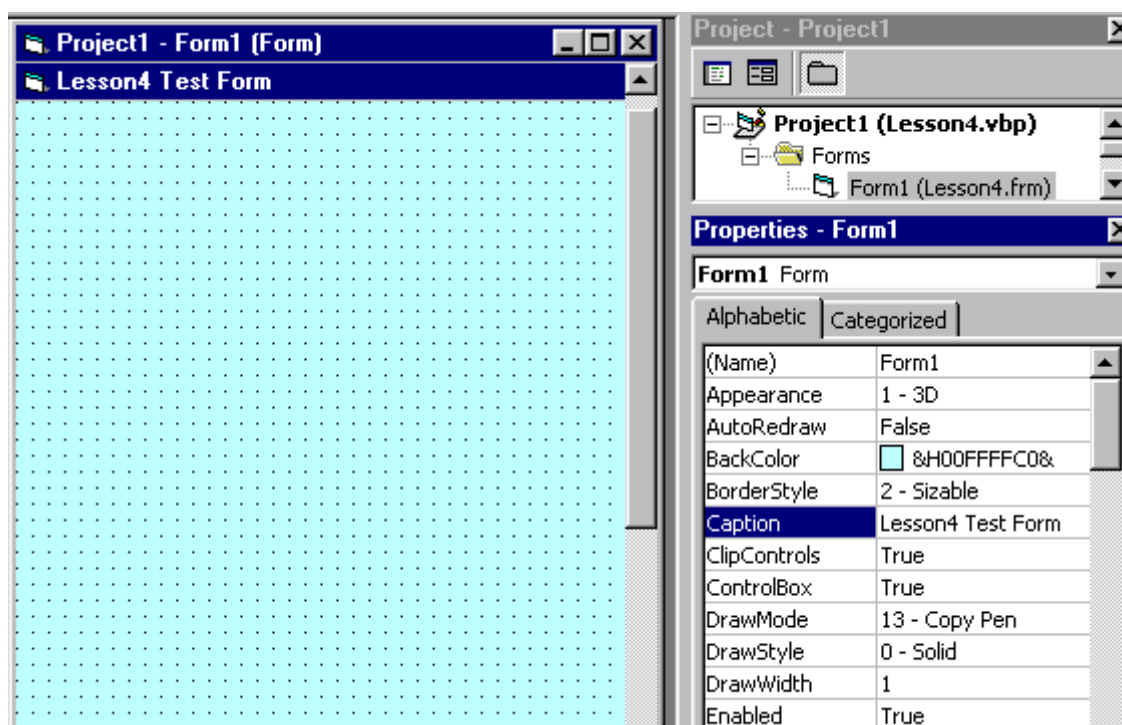
Toolbox

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

The Form

The Form is the first object you see when you Open the application. It's the window into which all the controls will appear, where you will input data and see results.

There's not too much you can do with the form at this time. Basically, you adjust the **BackColor** and the **StartPosition**(where it will open on the screen when you Run it) and then you start putting controls on it.

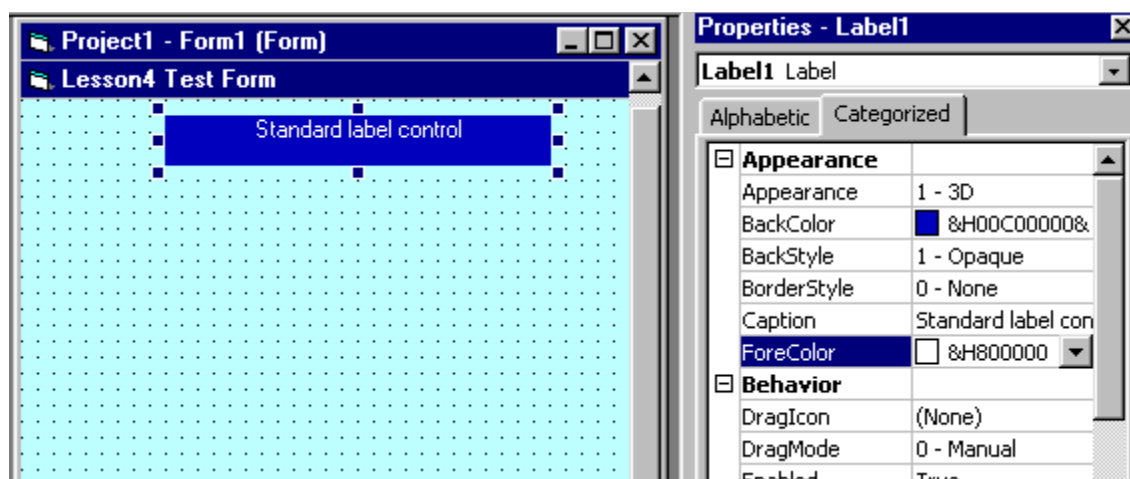


Form Design and its properties

The Label

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

- Caption – the text that is displayed in the label
- BackColor and ForeColor – colors of the background and the text
- BackStyle – Opaque or Transparent – whether the background is visible or not
- Font – font and size of text
- Alignment – text centered, left or right
- Multiline- True or False – if True, you can have several lines of text, delimited by <CR> in the label – by default, it is set to False



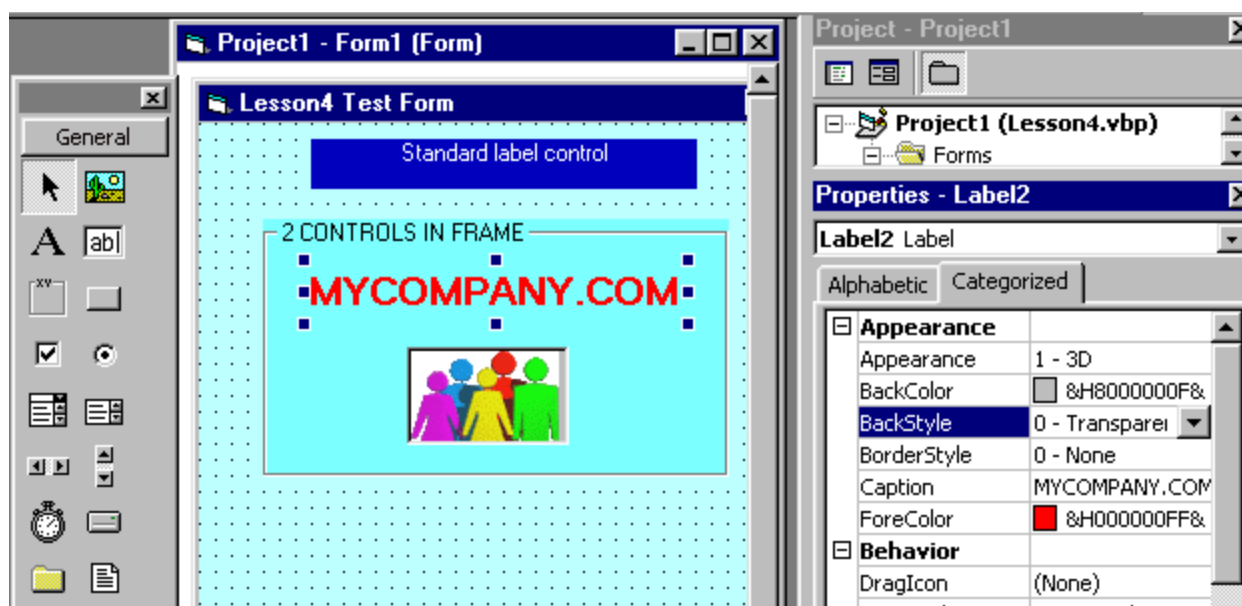
Label and its properties

Frame & PictureBox

When you want to group several controls together - name and address, for example - you use a **Frame**. The frame backcolor can be the same as the form's and only the frame borders will be obvious, or it can be a different color and stand out.

You create the frame before the controls. When you create controls in a frame, they are tied to the frame and move with it. The frame caption is the text that appears at the top of the frame - you use it to define the group.

The **PictureBox** is like a Label with a picture in it instead of text. The **Picture property** determines the name of the file, .BMP or .GIF, that will be displayed. It can be used for a company logo, etc.



Picture Box and Frame with its properties

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

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

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

KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: III BCom CA COURSE NAME: SOFTWARE DEVELOPMENT WITH VISUAL BASIC
COURSE CODE: 16CCU501A UNIT: III BATCH-2016-19

MousePointer	Integer. Specifies the shape of the mouse pointer when over the area of the ComboBox
NewIndex	Integer. Index of the last item added to the ComboBox. If the ComboBox does not contain any items , NewIndex is -1
Sorted	Boolean. Specifies whether the ComboBox's items are sorted or not.
Style	Integer. Specifies the style of the ComboBox's appearance
TabStop	Boolean. Specifies whether ComboBox receives the focus or not.
Text	String. Specifies the selected item in the ComboBox
ToolTipIndex	String. Specifies what text is displayed as the ComboBox's tool tip
Visible	Boolean. Specifies whether ComboBox is visible or not at run time
Methods	
AddItem	Add an item to the ComboBox
Clear	Removes all items from the ComboBox
RemoveItem	Removes the specified item from the ComboBox
SetFocus	Transfers focus to the ComboBox
Event Procedures	
Change	Called when text in ComboBox is changed
DropDown	Called when the ComboBox drop-down list is displayed
GotFocus	Called when ComboBox receives the focus
LostFocus	Called when ComboBox loses it focus

Adding items to a List

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

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



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

Object.AddItem item, Index

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

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

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

```
Private Sub Form_Load()
```

```
    Combo1.AddItem 1  
    Combo1.AddItem 2  
    Combo1.AddItem 3  
    Combo1.AddItem 4  
    Combo1.AddItem 5  
    Combo1.AddItem 6
```

```
End Sub
```

Removing Items from a List

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

Object.RemoveItem index

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

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

Sorting the List

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

Using the ComboBox

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



- Dropdown Combo (style 0)
- Simple Combo (style 1)
- Dropdown List (style 2)

The Simple Combo box displays an edit area with an attached list box always visible immediately below the edit area. A simple combo box displays the contents of its list all the time. The user can select an item from the list or type an item in the edit box portion of the combo box. A scroll bar is displayed beside the list if there are too many items to be displayed in the list box area.

The Dropdown Combo box first appears as only an edit area with a down arrow button at the right. The list portion stays hidden until the user clicks the down-arrow button to drop down the list portion. The user can either select a value from the list or type a value in the edit area.

The Dropdown list combo box turns the combo box into a Dropdown list box. At run time , the control looks like the Dropdown combo box. The user could click the down arrow to view the list. The difference between Dropdown combo & Dropdown list combo is that the edit area in the Dropdown list combo is disabled. The user can only select an item and cannot type anything in the edit area. Anyway this area displays the selected item.

Example

This example is to Add , Remove, Clear the list of items and finally close the application.

- Open a new Standard EXE project is opened an named the Form as Listbox.frm and save the project as Listbox.vbp
- Design the application as shown below.

Object	Property	Settings
Form	Caption	ListBox
	Name	frmListBox
TextBox	Text	(empty)
	Name	txtName
Label	Caption	Enter a name
	Name	lblName
ListBox	Name	lstName
Label	Caption	Amount Entered
	Name	lblAmount
Label	Caption	(empty)
	Name	lblDisplay
	Border Style	1 Fixed Single
CommandButton	Caption	Add
	Name	cmdAdd
CommandButton	Caption	Remove
	Name	cmdRemove
CommandButton	Caption	Clear
	Name	cmdClear

CommandButton	Caption	Exit
	Name	cmdExit



The following event procedures are entered for the TextBox and CommandButton controls.

```
Private Sub txtName_Change()  
If (Len(txtName.Text) > 0) Then 'Enabling the Add button  
'if atleast one character  
'is entered  
cmdAdd.Enabled = True  
End If  
End Sub
```

```
Private Sub cmdAdd_Click()  
lstName.AddItem txtName.Text 'Add the entered the characters to the list box
```

```
txtName.Text = "" 'Clearing the text box
```

```
txtName.SetFocus 'Get the focus back to the  
'text box
```

```
lblDisplay.Caption = lstName.ListCount 'Display the number of items in the list box
```

```
cmdAdd.Enabled = False ' Disabling the Add button
```

```
End Sub
```

The click event of the Add button adds the text to the list box that was typed in the Text box. Then the text box is cleared and the focus is got to the text box. The number of entered values will is increased according to the number of items added to the listbox.

```
Private Sub cmdClear_Click()  
lstName.Clear  
lblDisplay.Caption = lstName.ListCount  
End Sub
```

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

```
Private Sub cmdRemove_Click()  
Dim remove As Integer
```

```
remove = lstName.ListIndex 'Getting the index
```

```
If remove >= 0 Then 'make sure an item is selected  
'in the list box
```

```
lstName.RemoveItem remove 'Remove item from the list box
```

```
lblDisplay.Caption = lstName.ListCount 'Display the number of items  
'in the listbox
```

```
End If
```

```
End Sub
```

Remove button removes the selected item from the list as soon as you pressed the Remove button. The number of items is decreased in the listbox and the value is displayed in the label.

The code for the clear button clears the listbox when you press it. And the number of items shown in the label becomes 0.

When compared to TextBox controls, these controls are really simple. Not only do they expose relatively few properties, they also support a limited number of events, and you don't usually write much code to manage them.

Command Button Controls

Using Command Button controls is trivial. In most cases, you just draw the control on the form's surface, set its Caption property to a suitable string (adding an & character to associate a hot key with the control if you so choose), and you're finished, at least with user-interface issues. To make the button functional, you write code in its Click event procedure, as in this fragment:

```
Private Sub Command1_Click()  
' Save data, then unload the current form.  
Call SaveDataToDisk  
Unload Me  
End Sub
```

You can use two other properties at design time to modify the behavior of a `CommandButton` control. You can set the `Default` property to `True` if it's the default push button for the form (the button that receives a click when the user presses the `Enter` key—usually the `OK` or `Save` button). Similarly, you can set the `Cancel` property to `True` if you want to associate the button with the `Escape` key.

The only relevant `CommandButton`'s run-time property is `Value`, which sets or returns the state of the control (`True` if pressed, `False` otherwise). `Value` is also the default property for this type of control. In most cases, you don't need to query this property because if you're inside a button's `Click` event you can be sure that the button is being activated. The `Value` property is useful only for programmatically clicking a button:

This fires the button's `Click` event.

```
Command1.Value = True
```

The `CommandButton` control supports the usual set of keyboard and mouse events (`KeyDown`, `KeyPress`, `KeyUp`, `MouseDown`, `MouseMove`, `MouseUp`, but not the `DbClick` event) and also the `GotFocus` and `LostFocus` events, but you'll rarely have to write code in the corresponding event procedures.

Properties of a `CommandButton` control

- To display text on a `CommandButton` control, set its *Caption* property.
- An event can be activated by clicking on the `CommandButton`.
- To set the background colour of the `CommandButton`, select a colour in the `BackColor` property.
- To set the text colour set the `ForeColor` property.
- Font for the `CommandButton` control can be selected using the `Font` property.
- To enable or disable the buttons set the `Enabled` property to `True` or `False`
- To make visible or invisible the buttons at run time, set the `Visible` property to `True` or `False`.
- Tooltips can be added to a button by setting a text to the `Tooltip` property of the `CommandButton`.

- A button click event is handled whenever a command button is clicked. To add a click event handler, double click the button at design time, which adds a subroutine like the one given below.

```
Private Sub Command1_Click( )
```

```
.....
```

```
End Sub
```

Option Button Control

Option Button controls are also known as radio buttons because of their shape. You always use Option Button controls in a group of two or more because their purpose is to offer a number of mutually exclusive choices. Anytime you click on a button in the group, it switches to a selected state and all the other controls in the group become unselected.

Preliminary operations for an Option Button control are similar to those already described for Check Box controls. You set an Option Button control's Caption property to a meaningful string, and if you want you can change its Alignment property to make the control right aligned. If the control is the one in its group that's in the selected state, you also set its Valueproperty to True. (The Option Button's Value property is a Boolean value because only two states are possible.) Value is the default property for this control.

At run time, you typically query the control's Value property to learn which button in its group has been selected. Let's say you have three OptionButton controls, named optWeekly, optMonthly, and optYearly. You can test which one has been selected by the user as follows:

```
If optWeekly.Value Then
```

```
    ' User prefers weekly frequency.
```

```
ElseIf optMonthly.Value Then
```

```
    ' User prefers monthly frequency.
```

```
ElseIf optYearly.Value Then
```

```
    ' User prefers yearly frequency.
```

```
End If
```

Strictly speaking, you can avoid the test for the last `OptionButton` control in its group because all choices are supposed to be mutually exclusive. But the approach I just showed you increases the code's readability.

A group of `Option Button` controls is often hosted in a `Frame` control. This is necessary when there are other groups of `Option Button` controls on the form. As far as Visual Basic is concerned, all the `Option Button` controls on a form's surface belong to the same group of mutually exclusive selections, even if the controls are placed at the opposite corners of the window. The only way to tell Visual Basic which controls belong to which group is by gathering them inside a `Frame` control. Actually, you can group your controls within any control that can work as a container—`Picture Box`, for example—but `Frame` controls are often the most reasonable choice.

Example

Open a new Standard EXE project and the save the Form as `Option.frm` and save the project as `Option.vbp`.

Design the Form as per the following specifications table.

Object	Property	Settings
Label	Caption	Enter a Number
	Name	Label1
TextBox	Text	(empty)
	Name	Text1
CommandButton	Caption	&Close
	Name	Command1
OptionButton	Caption	&Octal
	Name	optOct

OptionButton	Caption	&Hexadecimal
	Name	optHex
OptionButton	Caption	&Decimal
	Name	optDec

The application responds to the following events

- The change event of the TextBox reads the value and stores it in a form-level numeric variable.
- The click event of optOct button returns currentval in octal.
- The click event of the optHex button currentval in hexadecimal
- The click event of the optDec button returns the decimal equivalent of the value held currentval.

The following code is entered in the general declarations section of the Form.

Dim currentval as variant

The variable is initialized to 0 by default. The change event procedure checks to ascertain the number system (Octal, Hexadecimal) that is in effect and then reads in the number.

```
Private Sub Text1_Change()  
If optOct.Value = True Then  
currentval = Val("&O" & LTrim(Text1.Text) & "&")  
Elseif optDec.value = True Then  
currentval = Val(LTrim(Text1.Text) & "&")  
Else  
currentval = Val("&H" & LTrim(Text1.Text) & "&")  
End if  
End Sub
```

The Val function is used to translate string to a number and can recognize Octal and Hexadecimal strings. The LTrim function trims the leading blanks in the text. The following code is entered in the click events of the OptionButton controls.

```
Private Sub optOct_Click()  
Text1.Text = Oct(currentval)  
End Sub
```

```
Private Sub optHex_Click()  
Text1.Text = Hex(currentval)  
End Sub
```

```
Private Sub optDec_Click()  
Text1.Text = Format(currentval)  
End Sub
```

The following code is entered in the click event of the Close button.

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

The Application is run by pressing F5 or clicking on the Run icon in the tool bar. By pressing the Exit button the program is terminated.

The following example illustrates the use of CheckBox control

* Open a new Project and save the Form as CheckBox.frm and save the Project as CheckBox.vbp

* Design the Form as shown below

Object	Property	Setting
Form	Caption	CheckBox

KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: III BCom CA COURSE NAME: SOFTWARE DEVELOPMENT WITH VISUAL BASIC

COURSE CODE: 16CCU501A

UNIT: III

BATCH-2016-19

	Name	frmCheckBox
CheckBox	Caption	Bold
	Name	chkBold
CheckBox	Caption	Italic
	Name	chkItalic
CheckBox	Caption	Underline
	Name	chkUnderline
OptionButton	Caption	Red
	Name	optRed
OptionButton	Caption	Blue
	Name	optBlue
OptionButton	Caption	Green
	Name	optGreen
TextBox	Name	txtDisplay
	Text	(empty)
CommandButton	Caption	Exit
	Name	cmdExit



Following code is typed in the Click() events of the CheckBoxes

```
Private Sub chkBold_Click()
```

```
If chkBold.Value = 1 Then
```

```
txtDisplay.FontBold = True
```

```
Else
```

```
txtDisplay.FontBold = False
```

```
End If
```

```
End Sub
```

```
Private Sub chkItalic_Click()
```

```
If chkItalic.Value = 1 Then
```

```
txtDisplay.FontItalic = True
```

```
Else
```

```
txtDisplay.FontItalic = False
```

```
End If
```

```
End Sub
```

```
Private Sub chkUnderline_Click()
```

```
If chkUnderline.Value = 1 Then
```

```
txtDisplay.FontUnderline = True
```

```
Else
```

```
txtDisplay.FontUnderline = False
```

```
End If
```

```
End Sub
```

Following code is typed in the Click() events of the OptionButtons

```
Private Sub optBlue_Click()
```

```
txtDisplay.ForeColor = vbBlue
```

```
End Sub
```

```
Private Sub optRed_Click()
```

```
txtDisplay.ForeColor = vbRed
```

```
End Sub
```

```
Private Sub optGreen_Click()  
    txtDisplay.ForeColor = vbGreen  
End Sub
```

To terminate the program following code is typed in the Click() event of the Exit button

```
Private Sub cmdExit_Click()  
    End  
End Sub
```

Run the program by pressing F5. Check the program by clicking on OptionButtons and CheckBoxes.

The PictureBox Control

PictureBox controls are among the most powerful and complex items in the Visual Basic Toolbox window. In a sense, these controls are more similar to forms than to other controls. For example, PictureBox controls support all the properties related to graphic output, including AutoRedraw, ClipControls, HasDC, FontTransparent, CurrentX, CurrentY, and all the Drawxxxx, Fillxxxx, and Scalexxxx properties. PictureBox controls also support all graphic methods, such as Cls, PSet, Point, Line, and Circle and conversion methods, such as ScaleX, ScaleY, TextWidth, and TextHeight. In other words, all the techniques that I described for forms can also be used for PictureBox controls (and therefore won't be covered again in this section).

Loading images

Once you place a PictureBox on a form, you might want to load an image in it, which you do by setting the Picture property in the Properties window. You can load images in many different graphic formats, including bitmaps (BMP), device independent bitmaps (DIB), metafiles (WMF), enhanced metafiles (EMF), GIF and JPEG compressed files, and icons (ICO and CUR). You can decide whether a control should display a border, resetting the BorderStyle to 0-None if necessary. Another property that comes handy in this phase is AutoSize: Set it to True and let the control automatically resize itself to fit the assigned image.

You might want to set the Align property of a PictureBox control to something other than the 0-None value. By doing that, you attach the control to one of the four form borders and have Visual Basic automatically move and resize the PictureBox control when the form is resized. PictureBox controls expose a Resize event, so you can trap it if you need to move and resize its child controls too.

You can do more interesting things at run time. To begin with, you can programmatically load any image in the control using the LoadPicture function:

```
Picture1.Picture = LoadPicture("c:\windows\setup.bmp")
```

and you can clear the current image using either one of the following statements:

' These are equivalent.

```
Picture1.Picture = LoadPicture("")
```

```
Set Picture1.Picture = Nothing
```

The LoadPicture function has been extended in Visual Basic 6 to support icon files containing multiple icons. The new syntax is the following:

```
LoadPicture(filename, [size], [colordepth], [x], [y])
```

where values in square brackets are optional. If filename is an icon file, you can select a particular icon using the size or colordepth arguments. Valid sizes are 0-vbLPSmall, 1-vbLPLarge (system icons whose sizes depend on the video driver), 2-vbLPSmallShell, 3-vbLPLargeShell (shell icons whose dimensions are affected by the Caption Button property as set in the Appearance tab in the screen's Properties dialog box), and 4-vbLPCustom (size is determined by x and y). Valid color depths are 0-vbLPDefault (the icon in the file that best matches current screen settings), 1-vbLPMonochrome, 2-vbLPVGAColor (16 colors), and 3-vbLPColor (256 colors).

You can copy an image from one PictureBox control to another by assigning the target control's Picture property:

```
Picture2.Picture = Picture1.Picture
```

The PaintPicture method

PictureBox controls are equipped with a very powerful method that enables the programmer to perform a wide variety of graphic effects, including zooming, scrolling, panning, tiling, flipping, and many fading effects: This is the PaintPicture method. (This method is also exposed by form objects, but it's most often used with PictureBox controls.) In a nutshell, this method performs a pixel-by-pixel copy from a source control to a destination control. The complete syntax of this method is complex and rather confusing:

```
DestPictureBox.PaintPicture SrcPictureBox.Picture, destX, destY, [destWidth], _  
[destHeight], [srcX], [srcY2], [srcWidth], [srcHeight], [Opcode])
```

The only required arguments are the source PictureBox control's Picture property and the coordinates inside the destination control where the image must be copied. The destX / destY arguments are expressed in the ScaleMode of the destination control; by varying them, you can make the image appear exactly where you want. For example, if the source PictureBox control contains a bitmap 3000 twips wide and 2000 twips tall, you can center this image on the destination control with this command:

```
picDest.PaintPicture picSource.Picture, (picDest.ScaleWidth - 3000) / 2, _  
(picDest.ScaleHeight - 2000) / 2
```

In general, Visual Basic doesn't provide a way to determine the size of a bitmap loaded into a PictureBox control. But you can derive this information if you set the control's AutoSize property to True and then read the control's ScaleWidth and ScaleHeight properties. If you don't want to resize a visible control just to learn the dimensions of a bitmap, you can load it into an invisible control, or you can use this trick, based on the fact that the Picture property returns an StdPicture object, which in turn exposes the Height and Width properties:

```
' StdPicture's Width and Height properties are expressed in  
' Himetric units.
```

```
With Picture1
```

```
width = CInt(.ScaleX(.Picture.Width, vbHimetric, vbPixels))
```

```
height = CInt(.ScaleY(.Picture.Height, vbHimetric, _
```

vbPixels))

End With

By the way, in all subsequent code examples I assume that the source PictureBox control's ScaleWidth and ScaleHeight properties match the actual bitmap's size. By default, the PaintPicture method copies the entire source bitmap. But you can copy just a portion of it, passing a value for srcWidth and srcHeight:

' Copy the upper left portion of the source image.

picDest.PaintPicture picSource.Picture, 0, 0, , , , _

picSource.ScaleWidth / 2, picSource.ScaleHeight / 2

If you're copying just a portion of the source image, you probably want to pass a specific value for the srcX and srcY values as well, which correspond to the coordinates of the top-left corner of the area that will be copied from the source control:

' Copy the bottom-right portion of the source image

' in the corresponding corner in the destination.

wi = picSource.ScaleWidth / 2

he = picSource.ScaleHeight / 2

picDest.PaintPicture picSource.Picture, wi, he, , , wi, he, wi, he

You can use this method to tile a target PictureBox control (or form) with multiple copies of an image stored in another control:

' Start with the leftmost column.

x = 0

Do While x < picDest.ScaleWidth

y = 0

' For each column, start at the top and work downward.

Do While y < picDest.ScaleHeight

picDest.PaintPicture picSource.Picture, x, y, , , 0, 0

' Next row

y = y + picSource.ScaleHeight

Loop

' Next column

x = x + picSource.ScaleWidth

Loop

Another great feature of the PaintPicture method lets you resize the image while you transfer it, and you can even specify different zoom-in and zoom-out factors for the x- and y-axes independently. You just have to pass a value to the destWidth and destHeight arguments: If these values are greater than the source image's corresponding dimensions, you achieve a zoom-in effect, and if they are less you get a zoom-out effect. For example, see how you can double the size of the original image:

```
picDest.PaintPicture picSource.Picture, 0, 0, _  
picSource.ScaleWidth * 2, picSource.ScaleHeight * 2
```

As a special case of the syntax of the PaintPicture method, the source image can even be flipped along its x-axis, y-axis, or both by passing negative values for these arguments:

' Flip horizontally.

```
picDest.PaintPicture picSource.Picture, _  
picSource.ScaleWidth, 0, -picSource.ScaleWidth
```

' Flip vertically.

```
picDest.PaintPicture picSource.Picture, 0, _  
picSource.ScaleHeight, , -picSource.ScaleHeight
```

' Flip the image on both axes.

```
picDest.PaintPicture picSource.Picture, picSource.ScaleWidth, _  
picSource.ScaleHeight, -picSource.ScaleWidth, -picSource.ScaleHeight
```

As you might expect, you can combine all these effects together, magnifying, reducing, or flipping just a portion of the source image, and have the result appear in any point of the destination PictureBox control (or form). You should find no problem in reusing all those routines in your own applications.

As if all these capabilities weren't enough, we haven't covered the last argument of the PaintPicture method yet. The opcode argument lets you specify which kind of Boolean operation must be performed on pixel bits as they're transferred from the source image to the

destination. The values you can pass to this argument are the same that you assign to the DrawMode property. The default value is 13-vbCopyPen, which simply copies the source pixels in the destination control. By playing with the other settings, you can achieve many interesting graphical effects, including simple animations.

The Image Control

Image controls are far less complex than PictureBox controls. They don't support graphical methods or the AutoRedraw and the ClipControls properties, and they can't work as containers, just to hint at their biggest limitations. Nevertheless, you should always strive to use Image controls instead of PictureBox controls because they load faster and consume less memory and system resources. Remember that Image controls are windowless objects that are actually managed by Visual Basic without creating a Windows object. Image controls can load bitmaps and JPEG and GIF images.

When you're working with an Image control, you typically load a bitmap into its Picture property either at design time or at run time using the LoadPicture function. Image controls don't expose the AutoSize property because by default they resize to display the contained image (as it happens with PictureBox controls set at AutoSize = True). On the other hand, Image controls support a Stretch property that, if True, resizes the image (distorting it if necessary) to fit the control. In a sense, the Stretch property somewhat remedies the lack of the PaintPicture method for this control. In fact, you can zoom in to or reduce an image by loading it in an Image control and then setting its Stretch property to True to change its width and height:

```
' Load a bitmap.  
Image1.Stretch = False  
Image1.Picture = LoadPicture("c:\windows\setup.bmp")  
' Reduce it by a factor of two.  
Image1.Stretch = True  
Image1.Move 0, 0, Image1.Width / 2, Image1.Width / 2
```

Image controls support all the usual mouse events. For this reason, many Visual Basic developers have used Image controls to simulate graphical buttons and toolbars. Now that

Visual Basic natively supports these controls, you'd probably better use Image controls only for what they were originally intended.

The Timer Control

A Timer control is invisible at run time, and its purpose is to send a periodic pulse to the current application. You can trap this pulse by writing code in the Timer's Timer event procedure and take advantage of it to execute a task in the background or to monitor a user's actions. This control exposes only two meaningful properties: Interval and Enabled. Interval stands for the number of milliseconds between subsequent pulses (Timer events), while Enabled lets you activate or deactivate events. When you place the Timer control on a form, its Interval is 0, which means no events. Therefore, remember to set this property to a suitable value in the Properties window or in the Form_Load event procedure:

```
Private Sub Form_Load()  
Timer1.Interval = 500 ' Fire two Timer events per second.  
End Sub
```

Timer controls let you write interesting programs with just a few lines of code. The typical (and abused) example is a digital clock. Just to make things a bit more compelling, I added flashing colons:

```
Private Sub Timer1_Timer()  
Dim strTime As String  
strTime = Time$  
If Mid$(lblClock.Caption, 3, 1) = ":" Then  
Mid$(strTime, 3, 1) = " "  
Mid$(strTime, 6, 1) = " "  
End If  
lblClock.Caption = strTime  
End Sub
```

You must be careful not to write a lot of code in the Timer event procedure because this code will be executed at every pulse and therefore can easily degrade your application's performance. Just as important, never execute a DoEvents statement inside a Timer event

procedure because you might cause the procedure to be reentered, especially if the Interval property is set to a small value and there's a lot of code inside the procedure.

Timer controls are often useful for updating status information on a regular basis. For example, you might want to display on a status bar a short description of the control that currently has the input focus. You can achieve that by writing some code in the GotFocus event for all the controls on the form, but when you have dozens of controls this will require a lot of code (and time). Instead, at design time load a short description for each control in its Tag property, and then place a Timer control on the form with an Interval setting of 500. This isn't a time-critical task, so you can use an even larger value. Finally add two lines of code to the control's Timer event:

```
Private Sub Timer1_Timer()  
On Error Resume Next  
lblStatusBar.Caption = ActiveControl.Tag  
End Sub
```

The Line Control

The Line control is a decorative control whose only purpose is let you draw one or more straight lines at design time, instead of displaying them using a Line graphical method at run time. This control exposes a few properties whose meaning should sound familiar to you by now: BorderColor (the color of the line), BorderStyle (the same as a form's DrawStyle property), BorderWidth (the same as a form's DrawWidth property), and DrawMode. While the Line control is handy, remember that using a Line method at run time is usually better in terms of performance.

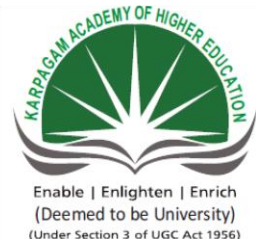
The Shape Control

In a sense, the Shape control is an extension of the Line control. It can display six basic shapes: Rectangle, Square, Oval, Circle, Rounded Rectangle, and Rounded Square. It supports all the Line control's properties and a few more: BorderStyle (0-Transparent, 1-Solid), FillColor, and FillStyle (the same as a form's properties with the same names). The same performance considerations I pointed out for the Line control apply to the Shape control.

The OLE Control

When OLE first made its appearance, the concept of Object Linking and Embedding seemed to most developers nothing short of magic. The ability to embed a Microsoft Word Document or a Microsoft Excel worksheet within another Windows application seemed an exciting one, and Microsoft promptly released the OLE control—then called the OLE Container control—to help Visual Basic support this capability.

In the long run, however, the Embedding term in OLE has lost much of its appeal and importance, and nowadays programmers are more concerned and thrilled about Automation, a subset of OLE that lets them control other Windows applications from the outside, manipulating their object hierarchies through OLE. For this reason, I won't describe the OLE control: It's a rather complex object, and a thorough description of its many properties, methods, and events (and quirks) would take too much space.



KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed University Established Under Section 3 of UGC Act 1956)

Coimbatore - 641021.

(For the candidates admitted from 2017 onwards)

Department of Commerce (CA)

SUBJECT: SOFTWARE DEVELOPMENT USING VISUAL BASIC

SEMESTER : V

SUBJECT CODE: 16CCU501A

CLASS : III B.COM CA

UNIT:III

S. NO	QUESTIONS	OPTION 1	OPTION 2	OPTION 3	OPTION 4	ANSWER
1	_____ contains the text box and list box	combo box	shape control	image control	timer control	combo box
2	If the number of items exceed the value that can be displayed, _____ bars will automatically appear on the control	icon	option button	command button	scroll bars	scroll bars
3	The _____ method is used to add items to a list at run time.	item	index	remove item	add item	add item
4	The _____ method is used to remove an item from the list	item	index	remove item	add item	remove item
5	The _____ property sets the index number of the currently selected item	index number	list index	list count	none	list index
6	The _____ property returns the index of the last item added to the list	index number	list couont	new index	old index	new index
7	The sorted property is set to _____ to enable a list to appear in alphanumeric order	0	1	TRUE	FALSE	TRUE
8	The sorted property is set to _____ to display the items in the order in which they are added to the	0	1	TRUE	FALSE	FALSE

	list					
9	_____ is the style of dropdown combo box	style 0	style 2	style 3	style 1	style 0
10	_____ is the style of simple combo box	style 0	style 1	style 3	style 2	style 1
11	_____ is the style of dropdown list box	style 0	style 1	style 2	style 3	style 2
12	_____ box saves the space on a form	list box	tool box	combo box	none	combo box
13	The _____ property represents the minimum value in the scroll bar	min	max	minimum	maximum	min
14	The _____ property represents the maximum value in the scroll bar	min	max	minimum	maximum	max
15	_____ provides easy navigation through a list of items or a large amount of information	scroll bar	command button	tool bar	tool box	scroll bar
16	The style property in _____ tab allows us to set the appearance of the control	line style	control	general	special	general
17	The _____ property is used to set the style of the lines displayed between nodes	line style	special style	general style	node	line style
18	In a tree view control _____ is always better to first se the name property with a prefix	twv	tvw	wvt	vwt	tvw
19	In tree view control 0 indicates _____ line	tree	root	style	none	tree
20	In tree view control 1 indicates _____ line	tree	root	style	none	root
21	The _____ property is a Boloean property that allows us to enable of disable the automatic label editing feature of the control	label edit	line edit	enable	disable	label edit
22	The _____ property is to the name of an existing image list control if pictures are to be displayed in tree view control	line edit	image list	image edit	none	image list
23	_____ configures the control for either manual / automatic dragging	OLE drag mode	OLE drop mode	label edit	image list	OLE drag mode
24	The _____ property allows us to set or retriive the delimiter character used for the path returned by a node's full path property	path seperator	seperator	indentation	none	path seperator
25	_____ uses to display the results of query on a DataBase.	tree view control	view control	tree control	none	tree view control
26	_____ is the extension for the Bitmap files	.Btp	.BMP	.bmp	.dbs	.bmp

27	A _____ bar control is a frame that can consist of several panels, which informs the user about the status of an application.	control bar	status bar	image bar	picture bar	status bar
28	An _____ is an object that we place on a form to enable or enhance a user's interaction with an application.	Active X control	Active Y control	Active Z control	none	Active X control
29	An _____ is an object that we place on a form to enable or enhance a user's interaction with an application.	tabstrip	strip	tab	none	tabstrip
30	_____ control serves as a visual and functional container for controls	form	frame	Ole	ADO	frame
31	_____ displays a true/false or yes/no option	box	command button	text box	check box	check box
32	_____ control is used link or embed on object, display and manipulate data from other windows based applications	Ole	ADO	DAO	none	Ole
33	_____ control adds a shape to a form	image	picture	shape	none	shape
34	_____ Box allows the user to select directories and paths which are displayed	Dir list	list	tool	Dir tool	Dir list
35	The _____ button control which is a part of an option group allows the user to select one option even if it displays multiple choices	command	option	check	image	option
36	_____ box displays the valid disk drives and allows the user to select one of them	Dir list	Drive list	list Drive	none	Drive list
37	_____ box displays a set of files from which a user can select the desired one	Drive list box	file list box	picture box	Dir list box	file list box
38	_____ control draws a straight line to the form.	line	circle	oval	rectangle	line
39	_____ control enable the user to connect to an existing DataBase and display information from it.	data base	data	file	list	data
40	_____ used in groups to display multiple choices from which the user can select one or more.	option button	check box	combo box	label box	check box
41	An _____ control is the rectangular portion into which picture files can be loaded.	line	shape	image	none	image
42	A bitmap also called _____ graphics defines an	paint type	circle type	oval type	drawing type	paint type

	image as a pattern of dots					
43	VB provides _____ controls for creating graphical applications.	1	2	3	4	3
44	_____ method sets the colour of the individual pixels	color	shape	control	none	color
45	_____ method sets the colour of an individual pixel	set color	set pixel	set method	none	set method
46	OLE stands for _____	object linker and embedd	object linking and embedded	object linking and embedding	object linker and embedded	object linking and embedding
47	DDE stands for _____	dynamic data exchange	dynemer data exchange	dynamic datum exchange	dynamic data exchanger	dynamic data exchange
48	_____ actually transfers control to the original application	ADO	DAO	OLE	DDE	OLE
49	The _____ function can be used for creating the object in code	create object	get object	both	none	both
50	The form we work with during design time is a _____	class	module	instance	sub routine	class
51	The OLE control can have only _____ object at a time.	one	two	three	four	one
52	Each time an OLE control is drawn on a form _____ object dialog box appears	delete	add	insert	none	insert
53	Paste special dialog box can be used to create an object during _____ time.	design	run	execution	view	design
54	The _____ method cab be used to specify an empty embedded object at runtime.	create embed	empty embed	OLE container	none	create embed
55	The _____ method is used to display the insert object dialog box	obj dlg insert	insert obj dlg	dlg obj insert	none	insert obj dlg
56	_____ property determines whether the OLE drop operations are allowed or not	OLE drop allowed property	OLE drop not allowed property	OLE drop mode property	VB OLE drag automatic property	OLE drop allowed property
57	The _____ parameter describes the format of the data stored in the data parameter	object	data format	create	insert	data format
58	_____ configures the Tree View control to enable / disable Ole drop operations	OLE drag mode	OLE drop mode	label edit	image list	OLE drop mode

59	The _____ property determines the horizontal distance between nodes in the view	path seperator	seperator	indentation	none	indentation
60	_____ function specifies the file name and assigns the picture to the picture property.	load picture	add picture	remove picture	none	load picture

UNIT-IV**SYLLABUS**

DAO – Creating a Database – Types of Record Set – ActiveX Data Object (ADO)
--

DAO

DAO (Data Access Objects) is an application program interface (API) available with Microsoft's Visual Basic that lets a programmer request access to a Microsoft Access database. DAO was Microsoft's first object-oriented interface with databases. DAO objects encapsulate Access's Jet functions. Through Jet functions, it can also access other Structured Query Language (SQL) databases.

Data Access Objects (DAO) provides data access to native Microsoft Jet engine databases (.mdb files), selected ISAM databases, and any ODBC data source. Historically, DAO is a popular solution when it comes to using Microsoft® Access (.mdb) and ISAM data sources such as Btrieve, FoxPro, Paradox, and dBase.

The general characteristics of DAO are:

- Difficulty in coding.
- Flexibility, with facilities to access many different data sources.
- Adequate-to-slow performance.
- Dynamic Data Language (DDL) functionality.
- Support for complex cursors.

Compared to the newer ActiveX Data Objects (ADO) or Remote Data Objects (RDO) technologies, Data Access Objects (DAO) is a slower, less capable data access alternative. DAO (and its companion, the Microsoft Jet database engine) was originally designed to handle remote ISAM data access. DAO is tied to the Microsoft Jet engine because it uses the Microsoft Jet engine query and result set processors.

Connecting Data Control to Database

To connect the data control to this database, double-click the DatabaseName property in the Properties window and then click on the button with three dots on the right to open a file selection dialog. From the dialog, search the folders of your hard drive to locate the database

file NWIND.MDB. It is usually placed under Microsoft Visual Studio\VB98\ folder, Select the aforementioned file and now your data control is connected to this database file.

The next step is to double-click on the RecordSource property to select the customers table from the database file NWIND.MDB, You can also change the caption of the data control to anything, we use Click to browse Customers. After that, we will place a label and change its caption to Customer Name. In addition, insert another label and name it as cus_name and leave the label empty as customers' names will appear here when we click the arrows on the data control. We need to bind this label to the data control for the application to work. To do this, open the label's DataSource and select data_navigator that will appear automatically. One more thing that we need to do is to bind the label to the correct field so that data in this field will appear on this label. To do this, open the DataField property and select ContactName.

Example program for database connectivity, Employee Payslip

```
Private Sub Command1_Click()  
Adodc1.Recordset.MoveNext  
MsgBox "This is the next record"  
End Sub  
  
Private Sub Command2_Click()  
Adodc1.Recordset.MovePrevious  
MsgBox "This is the previous record"  
End Sub  
  
Private Sub Command3_Click()  
Adodc1.Recordset.AddNew  
Text1.Text = ""  
Text2.Text = ""  
Text3.Text = ""  
Text4.Text = ""  
Text5.Text = ""  
Text6.Text = ""
```

Text7.Text = “ “

MsgBox “The record is added successfully”

End Sub

Private Sub Command4_Click()

Adodc1.Recordset.Delete

MsgBox “The record is deleted”

End Sub

Private Sub Command5_Click()

Adodc1.Recordset.Update

MsgBox “The record is updated”

End Sub

Private Sub Command6_Click()

End

End Sub

Private Sub Command7_Click()

Text6.Text = Val(Text5.Text) * 10 / 100

Text7.Text = Val(Text5.Text) * 12 / 100

Text8.Text = Val(Text5.Text) * 5 / 100

Text9.Text = Val(Text5.Text) * 5 / 100

Text10.Text = Val(Text5.Text) * 2.5 / 100

Text11.Text = Val(Text5.Text) + Val(Text6.Text) + Val(Text7.Text) + Val(Text8.Text) +
Val(Text9.Text) + Val(Text10.Text)

Text12.Text = Val(Text11.Text) - Val(Text10.Text) - Val(Text9.Text) - Val(Text7.Text)

End Sub

EMPLOYEE PAYROLL			
EMP NO.	0001	PF	1000
EMP NAME	KARUNAPPA	DA	150
DESIGNATION	Manager	EI	150
DEPARTMENT	Accounting	DEDUCTION	375
BASIC PAY	15000	GROSS SALARY	20775
HRA	1500	NET SALARY	17250

FIRST LAST NEXT PREVIOUS CALCULATE

ADD DELETE UPDATE EXIT

ActiveX Data Object



- The ADO (ActiveX Data Object) data control is the primary interface between a Visual Basic application and a database. It can be used without writing any code at all! Or, it can be a central part of a complex database management system. This icon may not appear in your Visual Basic toolbox. If it doesn't, select Project from the main menu, then click Components. The Components window will appear. Select Microsoft ADO Data Control, then click OK. The control will be added to your toolbox.

- As mentioned in Review and Preview, previous versions of Visual Basic used another data control. That control is still included with Visual Basic 6.0 (for backward compatibility) and has as its icon:



Make sure you are not using this data control for the work in this class. This control is suitable for small databases. You might like to study it on your own.

- The data control (or tool) can access databases created by several other programs besides Visual Basic (or Microsoft Access). Some other formats supported include Btrieve, dBase, FoxPro, and Paradox databases.

- The data control can be used to perform the following tasks:

1. Connect to a database.

2. Open a specified database table.

3. Create a virtual table based on a database query.

4. Pass database fields to other Visual Basic tools, for display or editing. Such tools are bound tools (controls), or data aware.

5. Add new records or update a database.

6. Trap any errors that may occur while accessing data.

7. Close the database.

- Data Control Properties:

Align Determines where data control is displayed.

Caption Phrase displayed on the data control.

ConnectionString Contains the information used to establish a connection to a database.

LockType Indicates the type of locks placed on records during editing (default setting makes databases read-only).

Recordset A set of records defined by a data control's ConnectionString and RecordSource properties. Run-time only.

RecordSource Determines the table (or virtual table) the data control is attached to.

- As a rule, you need one data control for every database table, or virtual table, you need access to. One row of a table is accessible to each data control at any one time. This is referred to as the current record.

- When a data control is placed on a form, it appears with the assigned caption and four arrow buttons:



The arrows are used to navigate through the table rows (records). As indicated, the buttons can be used to move to the beginning of the table, the end of the table, or from record to record.

Example program for Ado connectivity,

```
Private Sub Command1_Click()  
Adodc1.Recordset.MoveNext  
MsgBox "This is the next record"  
End Sub  
  
Private Sub Command2_Click()  
Adodc1.Recordset.MovePrevious  
MsgBox "This is the previous record"  
End Sub  
  
Private Sub Command3_Click()  
Adodc1.Recordset.AddNew  
Text1.Text = ""  
Text2.Text = ""  
Text3.Text = ""
```


KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: III BCom CA COURSE NAME: SOFTWARE DEVELOPMENT WITH VISUAL BASIC
COURSE CODE: 16CCU501A UNIT: III BATCH-2016-19

Text4.Text = “ “

Text5.Text = “ “

Text6.Text = “ “

Text7.Text = “ “

MsgBox “The record is added successfully”

End Sub

Private Sub Command4_Click()

Adodc1.Recordset.Delete

MsgBox “The record is deleted”

End Sub

Private Sub Command5_Click()

Adodc1.Recordset.Update

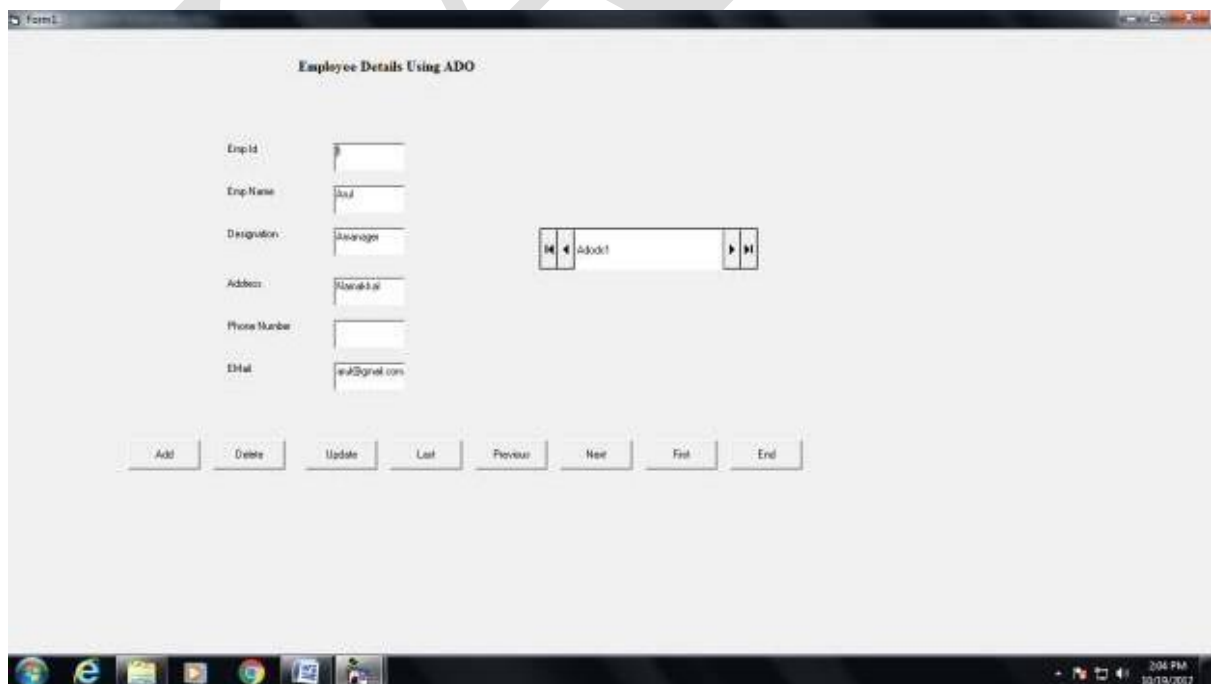
MsgBox “The record is updated”

End Sub

Private Sub Command6_Click()

End

End Sub

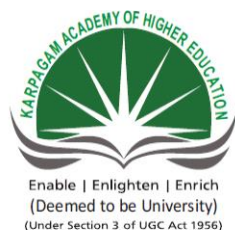


Form Design

KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: III BCom CA COURSE NAME: SOFTWARE DEVELOPMENT WITH VISUAL BASIC
COURSE CODE: 16CCU501A UNIT: III BATCH-2016-19

KAHE



KARPAGAM ACADEMY OF HIGHER EDUCATION
 (Deemed University Established Under Section 3 of UGC Act 1956)
 Coimbatore - 641021.
 (For the candidates admitted from 2017 onwards)
Department of Commerce (CA)

SUBJECT: SOFTWARE DEVELOPMENT USING VISUAL BASIC

SEMESTER : V

SUBJECT CODE: 16CCU501A

CLASS : III B.COM CA

UNIT:V

S.NO	QUESTIONS	OPTION 1	OPTION 2	OPTION 3	
1	Visual Basic can be used to compile class-based projects such as _____ components.	Active X	Active Y	Active Z	N
2	An Active X DLL is an _____ process server.	in	out	up	d
3	Active X components are _____, that can be used with other applications.	file make	object servers	compilation	n
4	_____ is imperative that we trap run time errors in our project	error checking	error finding	error catching	n
5	Active server pages are basically _____ pages that contain VB script code, which is executed on the server	C	C++	VB	F
6	_____ communicates with data sources through the JET database engine	Data Access Objects	data control	cointainer	r
7	_____ binds data-aware controls to microsoft access and other ODBC data sources.	DAO	ADO	Data control	D
8	_____ provides a framework for using code to create and manipulate components of a remote ODBC DataBase system	RDO	RDC	ODBC	V
9	_____ binds the control of a ODBC remote DataBase .	RDO	RDC	ODBC	V
10	_____ is an API call interface to the open DataBase connectivity libraries and drivers to provide data access to Oracle	RDO	RDC	ODBC	V

11	_____ is an implementation of the DataBase library API specifically designed to provide access to an SQL server through a VB application.	RDO	RDC	ODBC	V
12	_____ is a programming model that eliminates the need to choose from among DAO and RDO	ADO	RDC	ODBC	D
13	A data access object is a collection of _____ classes that model the structure of a relational database system.	module	procedure	object	p
14	The _____ object corresponds to a stored table definition.	DataBase	TableDef	QueryDef	R
15	The _____ is a stored query definition , which is a pre compiled SQL statement.	DataBase	TableDef	QueryDef	R
16	The record set object corresponds to a ___ view into a database table or the results of a query.	cursor	pointer	manual	m
17	A cursor view is one that stores rows of data in buffer and points to one row of data at a time called _____ record	previous	next	side	c
18	The _____ object represents a parameter associated with a query def object created form a parameter query.	index	user object	parameter	p
19	A _____ object includes information about one instance of a type of object.	group	relation	property	d
20	A _____ object represents a built-in characteristic or a user defined characteristic of a DAO.	property	group	document	r
21	A _____ object holds information describing the objects that are grouped into that container.	container	holder	DataBase	D
22	There are _____ types of DAO libraries supported by Visual Base 6.0.	1	2	3	
23	To open an existing database , the _____ method of the workspace object is used.	open database	create database	database	n
24	A _____ is an object that contains a set of records from database.	set	records	record set	n
25	The _____ method moves to the first row in the record set.	move first	move next	move previous	n
26	The _____ method moves to the last row in the record set.	move first	move next	move previous	n
27	The _____ method moves to the previous row in the record set.	move first	move next	move previous	n
28	The _____ method moves to the last row in the record set.	move first	move next	move previous	n
29	The _____ property is True when the user moves beyond the last record in the record set.	EOF	BOF	Both	n
30	The _____ property is True when the user has moved to a position before the first record in the record set.	EOF	BOF	Both	n

31	Modifying and Deleting records are used to _____ a record in a record set.	manipulate	edit	change	n
32	The _____ method can be used to locate a record in a table type record set.	open	close	store	s
33	The _____ method is used to perform action queries.	open	close	store	e
34	New fields can be added or existing fields can be deleted using the _____ methods respectively on the Tabledef object.	append	delete	a or b	a
35	The _____ command is used to add rows to a table.	add	insert	sub	d
36	_____ is a low-level object-based programming interface designed to access a wide variety of data sources and also not restricted to ISAM.	ADO	DAO	OLEDDB	R
37	_____ objects is the object-based interface to OLEDB.	Active X data	Active Y data	Active Z data	n
38	_____ window shows all the current connections we have built to databases, as well as any data environment connections that have been added by user.	Data view window	Data report designer	Query designer	D d
39	_____ supports page and report headers , detail lines and many other common features, including a variety of graphics and font features.	Data view window	Data report designer	Query designer	D d
40	The _____ designer enables us to design queries save them in our DataBase.	Dataview window	Data report	Query	D
41	With the help of _____ Designer, we can link it to all our databases, tables and Queries with a single object.	Dataview window	Data report	Query	D
42	The _____ can actually write real ADO code that is designed to browse records in the same way as the old data contsl could.	Data form wizard	Data view window	Query	D
43	RDO is _____	Remote Data Objects	Remotes Data Objects	Remote Datum Objects	R C
44	_____ method is used to create a new record in the record set.	add new	create new	new add	n
45	_____ method can be used to delete an existing record in dynaset of table type record set.	delete method	delete existing	delete	n
46	The _____ method is used to retrive data from the tables.	execute	run	stop	o
47	_____ method is usde to open a table Dynaset of Snapshot record set from the tabledef object	open record set	create record set	open table	c
48	_____ method updates and refreshes any attached table links for the tabledef object.	new link	fresh link	refresh link	n
49	_____ method is used to create and store a user-defined property.	create index	create property	create field	c

50	_____ method is used to add an index to the table def object.	add index	new index	create index	f
51	_____ method is used to ass a new field to an existing tabledef object.	add field	new field	create field	f
52	_____ queries are SQL statements that perform specific actions on the database.	SQL	action	calculation	M
53	A _____ is a collection of users with similar access rights.	set	gang	group	o
54	A _____ object represents a relationship between fields in tables or queries.	property	document	relation	M
55	The_____ object is used to define and enforce database security.				
56	The_____object corresponds to a column of data type and set of properties.	field	Table	Query	r
57	DataBase is the variable that represents the _____ object.	database	visual basic	HTML	D
58	_____ type of record set is identical to the snapshot record set except that we can only scorll forward through its record.	Backward only type record set	Reverse only type record set	Forward only type record set	n
59	The_____ type record set can refer to any table, attached table or query.	snap shot	backward	forward	d
60	A _____ type record set cannot be update and does not reflect any changes made by the users.	dynaset	forward	backward	s



KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed University Established Under Section 3 of UGC Act 1956)

Coimbatore - 641021.

(For the candidates admitted from 2017 onwards)

Department of Commerce (CA)

SUBJECT: SOFTWARE DEVELOPMENT USING VISUAL BASIC

SEMESTER : V

SUBJECT CODE: 16CCU501A

CLASS : III B.COM CA

UNIT:IV

S. NO	QUESTIONS	OPTION 1	OPTION 2	OPTION 3	OPTION 4	ANSWER
1	_____ is used for opening many windows at the same time	MDI	IDM	DIM	none	MDI
2	All the document windows are contained in a _____	child	parent	both	none	parent
3	VB application can have only _____ MDI form.	1	2	3	4	1
4	Each time the user clicks new from the file menu, a new _____ window is created and displayed.	parent	form	standard	child	child
5	'The text box in the child form can be adjusted to the same size of the child form' This can be incorporated into the _____ event.	size	resize	form	reform	resize
6	The resize event is fired whenever the size of the form is _____	fixed	not changeable	changed	none	changed
7	The _____ is reserved word used in the form _ Resize() proccedure is a variable containing the names of the form where the code is currently executed	my	mine	myself	me	me
8	The ME keyword in VB behaves like an _____ declared variable.	implicity	explicity	static	constant	implicity

9	_____ is an area containing control to provide quick access to the most commonly used oprations	status bar	tool bar	grid	line	tool bar
10	Tool Bar is also called as _____	ribbon bar	control bar	both	none	both
11	In order to add a tool bar item, the MDI form is selected and the picture control in the tool box is _____ clicked.	left	right	single	double	double
12	Status bar appears at the _____ of an MDI form.	top	bottom	middle	side	bottom
13	A _____ can be used to add new commands to the existing menus, create new menus and menu bars, change or delete existing menus and menu bars.	Tool editor	Menu interface	Menu editor	none	Menu editor
14	A menu editor can be added oly after _____ a project.	creating	opening	closing	maximizing	opening
15	The _____ includes all the menu controls of the current form.	Tool editor	Menu interface	Menu editor	none	Menu editor
16	A _____ bar is a line which seperates the menu item.	join	include	divide	seperator	seperator
17	Pop-up menus are also called _____ menus, because the items displayed on the pop-up menu depend on where the pointer is located when the right mouse button is clicked.	menu controls	check marks	context menus	none	context menus
18	When the menu item is _____ it is dimmed and cannot be selected, but viewed.	able	disable	view	visible	disable
19	To place a check mark in a menu item, the checked property is set to _____	TRUE	FALSE	0	1	TRUE
20	The most common way to build a new class for new objects in VB is to use a _____	module	form	class	class module	class module
21	_____ number of class modules can be created in a project.	less	1	2	any	any
22	Each class mocule can respond to only _____ events.	1	2	3	4	2
23	The _____ event for a class module is triggered when the class created via the New keyword is set to nothing	initiate	terminate	both	none	terminate

24	The_____ Event procedure contains code that needs to be executed at the moment the object is created.	Initialize	Terminate	stop	start	Initialize
25	The _____ event contains any code that is needed to execute in order to clean up after the object when it is being destroyed	Initialize	Terminate	stop	start	Terminate
26	Click the _____ button to destroy the existing thing object and create a new one.	destroy old thing	create new thing	both	none	create new thing
27	Press the key _____ bigger print to run the project.	F4	F5	F6	F9	F5
28	An _____ is a combination of code and data that can be treated as a unit	class	module	object	none	object
29	The _____ keyword is used only when we want VB to create a new instance of the original object.	old	new	fresh	begin	new
30	_____ is the key concepts in working with objects.	polymorphism	inheritance	encapsulation	variables	encapsulation
31	_____ combines data and characteristics into a single package.	encapsulation	inheritance	polymorphism	none	encapsulation
32	_____, the ability to make classes descend from other classes is not supported by VB.	encapsulation	inheritance	polymorphism	none	inheritance
33	In an Object Oriented Programming Language, the ability to redefine a routine in a derived class is called _____	encapsulation	inheritance	polymorphism	none	polymorphism
34	_____ is helpful when creating new objects from old ones because it makes a programmers job simpler	polymorphism	encapsulation	inheritance	variables	polymorphism
35	In VB, the setting or sttributes of an object are referred to as _____	properties	objects	methods	contrast	properties
36	The various procedures that can operate on the object are called its _____	objects	methods	properties	contrast	methods
37	Properties that we can set and those that we manipulate at run time are called _____	set manipulate	run time design	read write properties	none	read write properties
38	Properties that we can only read at run time is _____	read only property	only read property	read write property	none	read only property
39	_____ can affect the values of properties	objects	classes	. modules	methods	methods

40	The role of _____ is to free the space allocated for the constructor	destructor	collection	object	procedure	destructor
41	If multiple instances of the form are created using the DIM statement with the new keyword, a _____ is created	collection object	forms collection	constructors	destructors	collection object
42	A _____ is an ordered set of items that can be considered as a unit.	collection object	collections	constructors	destructors	collection object
43	_____ method add items to the collection	remove	add	+	addition	+
44	_____ method removes items from the collection	-	sub	remove	removal	remove
45	_____ method returns the number of items present in the collection	return	count	number	none	count
46	_____ method returns an item referenced by index property.	return	count	Item	ideal	Item
47	' for each...next ' statement uses an object called an _____	collection	control	enumerator	module	enumerator
48	_____ keeps track of the place where we are present in the collection and return the next item when it is needed.	statement	enumerator	constructor	destructor	enumerator
49	_____ is an array that contains all the controls on a form in VB	Control Collection	class collection	module collection	none	Control Collection
50	The _____ property of the control collection returns the number count behave set.	number	count	behave	set	count
51	The _____ statement is used to associate a particular control with an object variable	set	reset	class	object	set
52	There are _____ types of scroll bars are available.	1	2	3	4	2
53	The _____ control is a visual element that contains several predefined shapes	shape	line	rectangle	square	shape
54	The _____ control is a straight line segment that is drawn at design time.	square	rectangle	shape	line	line
55	The Fill color and Fill style properties in shape control can be _____	not changed	changed	enabled	disabled	changed
56	The _____ is a commonly used control , which enables the user to select a value by positioning it at the desired location.	vertical scroll bar	Horizontal scroll bar	scroll bar	status bar	scroll bar
57	_____ control executes timer events at	time	timer	watch	none	timer

	specifies intervals of time					
58	The _____ control which is a part of an option group allows the user to select one option even if it displays multiple choices.	check box	command button	option button	timer	option button
59	_____ allows the user to select an item from the dropdown list box to type a selection in the text box.	combo box	command button	check box	none	combo box
60	To create a child form, set its MDI child property to _____	TRUE	FALSE	0	1	TRUE

UNIT-V

SYLLABUS

Data Report: Data Environment – Designer – Connection Object – Command Object – Data Report Control – Sections of Report Design. Case Study: Automated System for student mark list – Automated System for Railway reservation.

Data Environment

In this chapter, we will understand how to use Data Environment, which was introduced in Visual Basic 6.0, how to use DataCombo, which is a data-bound ActiveX control, and how to use DataGrid, which is also a data-bound ActiveX control.

Let us understand each of them first. Then we will develop a data-entry screen using all these three.

Data Environment

Data Environment object is created and managed using Data Environment designer. A Data Environment object can accomplish any of the following tasks.

- Can contain connection and command objects
- Can establish relationship between command object to represent hierarchies
- Can group data and get aggregates
- Allows fields to be dragged and dropped on to a Data Report, for report, and a Form for data entry screen
- Can work as source for Databound control such as DataCombo controls and Hierarchical Flexgrid control.

In order to work with Data Environment object, you have to add Data Environment Designer to your project. Data Environment Designer allows programmer to accomplish any of the tasks mentioned above. Data Environment Designer is invoked when you double click on Data Environment object in Project Explorer.

To add Data Environment object to the project:

1. Select Project-> Add Data Environment. If Add Data Environment option is not found then select More ActiveX Designers and then select Data Environment.

Note: Visual Basic adds Data Environment Designer to your project and also creates a connection object.

Data Environment Designer

Working with Connection Object

Whenever you create a Data Environment object a Connection object is added to Data Environment object. But that Connection object is not connected to any database. So first we have to establish a connection with the Connection object. Now, let us see how to connect to database through Connection object.

To connect to Biblio.mdb using connection object:

1. Select Connection object (Connection1) and click on right button.
 2. From popup menu select Properties option
- Data Environment designer displays Data Link properties dialog.
3. Select the required OLEDB provider (Microsoft Jet 3.51 OLE DB Provider)
 4. Select Connection tab and select the name of the database (BIBLIO.MDB)
 5. Click on Ok

To rename connection object:

1. Invoke popup menu by click on Connection object with right button.
2. Select Rename option.
3. Enter new name and press enter key to confirm it.

Creating Command Object

After connection is established using Connection object, now we have to create a command object to access the data of the database. A connection object can contain any number of command objects. In fact, a command object can contain another command object, which is called as child command object.

To create command object to get data from Authors table:

1. Invoke popup menu of Connection object and select Add Command option
Data Environment Designer creates a command object with name Command1.
2. Invoke Popup menu of command object by clicking on right button and select Properties option.
Properties dialog is displayed
3. Change the Name of the command object to Authors
4. Select Table as Database Object
5. Invoke the list of tables by clicking on down arrow of Object Name drop down and select Authors as the name of the table.
6. For the time being, ignore remaining tabs and click on OK.

Properties of Command object.

A command object is created with the default name (command1). If you expand the command object by clicking on + sign, you get the columns of AUTHORS table.

Data Environment designer after a command object is created.

To create master-detail relationship:

One of the interesting features of Data Environment object is, it allows you to create hierarchy of objects.

To understand how to create hierarchy of command objects, let us create two command objects. One for Publishers table and another for Titles table.

Publishers command object becomes parent command and Titles command object becomes child command object. These hierarchies can be used straightaway in Hierarchical FlexGrid control and Data Report object.

To create command object for Publishers:

1. Select connection object, Biblio, and click on right button to invoke popup menu and select Add Command option of popup menu.
2. After a command object is created invoke its properties by selecting Properties option from Popup menu.
3. Change Name of the command object to Publishers
4. Choose Table as the Database Object and select Publishers as the name of the object.
5. Click on Ok.

To create child command object for Titles:

1. Select Publishers command object and invoke popup menu.
2. Select Add Child Command option.
3. A new command object is created and placed along with fields of Publishers command object.
4. Select Child command object and invoke its properties
5. Change name to Titles.
6. Choose Table as Database Object and Titles as the name of the database object.
7. Click on Relation tab to set relationship between Publishers and Titles command object.
8. Make sure Parent Fields and Child Fields are set to PubID and click on Add button to establish relationship between Publishers and Titles based on PubId field.
9. Click on Ok.

After child object is created, Data Environment Designer has three command objects – Authors, Publishers, and Titles. Where Authors and Publishers are at the same level (Parent objects), Titles is a child command object of Publishers command object. When you expand all command objects, Data Environment Designer

Data Environment Designer after three command objects are added.

Data Combo and Data List controls

DataCombo and DataList controls are used to present a list of values to users from which user can select one. The list of values to be displayed to user will come from one recordset and the selected value goes into another recordset.

Note: Whenever I refer to DataCombo, it also refers to DataList, unless the difference is explicitly specified.

As DataCombo is an ActiveX control, it is to be loaded into its project explicitly. And DataCombo is based on OLEDB interface. So it can be used with either Data Environment or ActiveX data control, because they are also based on OLEDB (use ADO).

To load Data List controls:

1. Select Project->Components
2. Visual Basic displays a list of available ActiveX controls.
3. Turn on check box on the left of Microsoft DataList Controls 6.0 (OLEDB)
4. Click on OK.

Two ActiveX controls are loaded into project – DataList and DataCombo.

Note: You also have Microsoft Data Bound List Controls 6.0 ActiveX control. It contains DBList and DBCombo, which work with Data Control and not with OLEDB. If you are developing a project with only DAOs or RDOs, then you can use these controls.

The following are the important properties that are specific to Data List controls (Data Combo and Data List).

Property Meaning

BoundColumn Contains the name of the source field in the Recordset object that is used to supply a data value to another Recordset to which control is bound.

ListField Returns or sets the name of the field in the Recordset object, specified by the RowSource property, used to fill the DataCombo or DataList control's list portion.

RowSource Sets a value that specifies the source from which the control's list is filled. The source is either an ActiveX Data control or Data Environment.

RowMember Returns or sets the data member to be used to display list text. Used when RowSource is a Data Environment.

BoundText Returns or sets the value of the field specified by the BoundColumn property.

DataSource Contains an object to which control is bound. It is generally ADODC or Data Environment.

DataMember Specifies the data member of Data Environment to which control is bound.

DataField Specifies the name of the field to which control is bound.

DataChanged Determines whether data in the control has been changed by any process.

Style Determines the behavior and appearance of the control. Applicable only to DataCombo.

MatchedWithList Returns True if the current content of the BoundText property matches one of the records in the list portion of the control.

SelectedItem Returns a value containing a bookmark for the selected record in a DataCombo or DataList control.

VisibleItems Returns an array of bookmarks, one for each visible item in the DataCombo or DataList control's list.

VisibleCount Returns a value indicating the number of visible items in the list portion of the DataCombo or DataList control.

MatchEntry Returns or sets a value indicating how the DataCombo or DataList control performs searches based on user input.

Table 20.1: Properties of DataList control.

Note: See Sample Data Entry screen later in this chapter to understand how to use various properties of the control.

DataGrid Control

Allows you to display and manipulate a set of records and columns taken from the specified recordset. You can specify whether you want to allow user to modify displayed data, add new records, and delete existing records.

The following are a few features supported by DataGrid control.

It supports the following features:

- Each cell contains a value. The value of the cell can be edited interactively by user or programmatically.
- Provides Columns collection using which you can access columns of the grid. When you select a column ColIndex property is set to enable you to access the columns selected by the user.
- Data in the current row can be accessed through Bookmark property, which provides access to underlying recordset object's record.
- Each column can have its own display attributes
- The DataGrid control functions similarly to the DBGrid (which is based on DAO & RDO) control except that it doesn't support an unbound mode.

The following are the important methods, properties and events of DataGrid control.

Methods of DataGrid control

The following are the methods that are specific to DataGrid.

Method Meaning

CaptureImage Returns a captured image of the grid's display in its current state.

ClearFields Clears all fields and restores grid to two columns and two rows.

ClearselCols Deselects all the columns that are currently selected.

HoldFields Sets the current column/field layout as the customized layout.

ColContaining Returns the ColIndex value of the DataGrid control column containing the specified coordinate (X) value.

GetBookmark Takes a relative row and returns the bookmark of the relative row. If

GetBookmark 1 is given then the bookmark of next row of the current row is given.

Rebind Causes the DataGrid control to reset the columns, headings and other properties based on the current Data control properties.

RowBookmark Returns a value containing a bookmark for a visible row in the DataGrid control.

RowContaining Returns a value containing a bookmark for a visible row in the DataGrid

control.

RowTop Returns a value containing the Y coordinate of the top of a specified row of a DataGrid control

Scroll Scrolls the DataGrid control horizontally and vertically in a single operation. You have to specify the number of rows and number of columns to scroll.

Table 20.2: Methods of DataGrid control.

To put an image of the DataGrid in an image box:

```
Image1.picture = DataGrid1.captureImage
```

To get the row and columns of the current mouse pointer:

X and Y are coordinates of Mouse pointer

```
RowValue = DataGrid1.RowContaining(Y)
```

```
colValue = DataGrid1.ColContaining(X)
```

Properties of DataGrid control

The following are important properties of DataGrid.

Property Meaning

AllowArrows Sets or returns a value that determines whether the control uses the arrow keys for grid navigation.

AproxCount Returns the approximate number of rows in the grid.

CurrentCellModified Sets or returns modification status of the current cell. If it is true, it means current cell, identified by BookMark and Col properties, is modified.

CurrentCellVisible Returns true if current cell is visible.

EditActive Determines the current editing status of the current cell in the grid. Setting it to true will initiate editing of current cell. Setting it to false ends editing. Changing its value will automatically fire relevant events.

TabAction Sets or returns a value that defines the behavior of the tab key. Valid values are: 0 - tab moves out of grid, 1- tab moves between cells in the same row, but once row changes then control moves out of grid, 2 – tab moves amount cells.

WrapCellPointer Determines whether moving right on the last column will move control to first column of the next row (True) or not (False).

AllowAddnew,

AllowDelete,

AllowUpdate Determines whether the corresponding action will be allowed.

Columns Returns a collection of column objects.

DataChanged Returns true, if data in the grid is changed.

Firstrow Contains the bookmark of the first visible row in the grid.

Leftcol Contains the number of the column that is to be displayed as the left-most column of the grid.

SelBookMarks Returns a collection of bookmarks for all the selected rows.

VisibleCols Returns the number of visible columns in the grid.

VisibleRows Returns the number of visible rows.

RowHeight Sets the height of all rows in datagrid.

Table 20.3: Properties of DataGrid control.

Events of DataGrid control

The following are events of DataGrid control.

Event When is it fired?

AfterColEdit After the editing of a cell is over. This event occurs after BeforeColUpdate and AfterColUpdate events.

BeforeColEdit Just before the user enters a character to enter into edit mode. Set Cancel property to True, to prevent user from editing the cell.

Coedit When user has entered into edit mode. Occurs immediately after BeforeColEdit event.

Error When a data access error has taken place.

OnAddNew When an action invokes an AddNew operation. The AddNew operation is invoked by either entering a character in a new record or by programmatically changing the data in new record.

SelChange When user selects a different range of rows or columns.

AfterColUpdate After data is moved from a cell in the DataGrid control to the control's copy buffer.

BeforeColUpdate After editing is completed in a cell, but before data is moved from the cell to the DataGrid control's copy buffer.

AfterDelete, AfterUpdate,

AfterInsert After the corresponding operation has taken place.

BeforeDelete, BeforeUpdate,

BeforeInsert Before the corresponding operation takes place.

Table 20.4: Events of DataGrid control.

The following are a few examples using events, methods and properties of datagrid control.

To take confirmation before a record is deleted from grid:

```
Private Sub DataGrid1_BeforeDelete (Cancel As Integer)
```

```
Dim mResult As Integer
```

```
mResult = MsgBox("Are you sure that you want to delete " &
```

```
DataGrid1.Selbookmarks.count & " record?", _
```

```
vbYesNo + vbQuestion, "Delete Confirmation")
```

```
‘ cancel deletion, if user click on NO button
```

```
If mResult = vbNo Then Cancel = True
```

```
End Sub
```

Listing 20.1: Code to take confirmation before record is deleted.

To prevent from entering a date that is less than today's date:

```
Private Sub DataGrid1_BeforeColUpdate (ColIndex As Long, OldValue As
```

```
Variant, Cancel As Integer)
```

```
If ColIndex = 1 Then
```

```
If DataGrid1.Columns(1).Value < Now Then Cancel = True MsgBox "You must enter a date  
that is later than today." End If End If End Sub Listing 20.2: Code to check whether entered
```

```
date is less than current date. Creating a Data Entry screen Let us create a sample data entry  
screen to take the details of a sales transaction. We will use a Data Environment to access the
```

data. DataCombo is used to provide list of products to user so that user can select one from the list of products. The data entry screen also provides the complete details of all the products upon user's request. For this we use a DataGrid. The following are the various major steps involved in creating this application. • Creating Data Environment • Designing Form • Writing Code

Creating Data Environment

The following are the steps required to create new application , add Data Environment and create required objects in Data Environment .

1. Start a new project using File -> New Project and select Standard Exe as the project type.

2. Add a Data Environment object by selecting Project -> Add Data Environment.
3. Change the Name of Data Environment to DenvProducts.
4. Click right mouse button on Connection object to invoke popup menu and select properties option to invoke Data Link Properties.
5. Select Microsoft Jet 3.51 OLE DB Provider as OLE DB provider.
6. Select Connection tab and choose Products.mdb as the database.
7. Click on OK to close Data Link Properties.
8. Invoke popup menu of Connection object and select Add Command to add a command object.
9. A new Command object will be created with the name Command1.
10. Select Command object and invoke its properties.
11. Change the following properties of the command1 object.

Name Products

Database Object Table

Object Name Products

12. Invoke popup menu of Connection object again and select Add Command option.
13. A new command object will be created again.
14. Change the following properties of the newly created command object.

Name Sales

Database Object Table

Object Name Sales

After both the command objects are created, the Data Environment designer should look like

this.

Data Environment designer after two command objects are created.

Designing form

After we have created required objects in Data Environment, let us now concentrate on placing the required controls on the form and changing their properties. In this application we have to have two forms. One is automatically added to the project as it is a Standard Exe project. So we have to create only one form. We will concentrate on second form a bit later. First let us place required controls on the first form.

Before we proceed let us load two ActiveX controls that will be used in this project.

To Load DataList control into project:

1. Select Project -> Components.
2. Visual Basic displays Components dialog with list of available ActiveX controls.
3. Check Microsoft DataList Controls 6.0 (OLEDB)
4. Click on Ok

Note: DataList consists of DataList control and DataCombo control.

To Load DataGrid control into project:

1. Select Project -> Components
- Visual Basic displays Components dialog with list of available ActiveX controls.
2. Check Microsoft DataGrid Controls 6.0 (OLEDB)
 3. Click on Ok

Designing first form (Data Entry)

Designing data entry screen involves the following steps.

1. Open the Form and the Data Environment designer and place them side by side.
2. Drag Fields (QTY SOLD, RPU and DISCOUNT) from Data Environment designer to Form.
3. When you drag and drop a field from Data Environment Designer to Form, a textbox for

field and a label control for the name of the field will be created on the form at the position where you dropped the field.

If you look at the properties of textbox that is created when you drag and drop a field of Data Environment on the form, you find the following properties already set by Data Environment. These properties make the textbox bound to Data Environment.

The properties for qtySold are:

DataSource DenvProducts

Datamember Sales

DataField QtySold

Using DataCombo control

1. Place DataCombo control on the from .
2. And change the following properties.

Property Value

Name DcmbProdno

BoundColumn Prodno

ListField Proddesc

DataField Prodno

DataMember Sales

DataSource Denvproducts

RowSource Denvproducts

RowMember Products

Style 2-dbcDropdownlist

DataCombo box is used to get the list of products from Products command of Data Environment. It displays product description (listField is proddesc) but when user selects an

item it takes the corresponding product number (boundcolumn is prodno).

The data combo box is bound to prodno field (datafield is prodno) of Sales command object (datamember is Sales) of the Data Environment (datasource is Denvproducts). As the result whatever product number is selected by the user that is copied into prodno field of Sales table.

That is all about DataCombo control.

1. Add a line at the bottom of the form using Line tool.
2. Add four command buttons and arrange them.
3. Change the properties of the form and command buttons as follows.

Object Property Value

Form1 Name FrmSales

Caption Sales Entry Form

BorderStyle 3-Fixed Dialog

Command1 Name CmdSave

Caption &Save

Default True

Command2 Name CmdCancel

Caption &Cancel

Cancel True

Command3 Name CmdList

Caption &Products List

Command4 Name CmdQuit

Caption &Quit

That's all we have so far as the data entry form is concerned. We will write code later. It is time to design our second form in which we display the details of products using datagrid.

Designing second form (List form)

1. First, make sure you have second form added to your project. If not, add a form using Project->Add Form.
2. Place DataGrid control and a command button on the form.
3. Change the properties as follows.

Object Property Value

Form2 Name FrmProdList

Caption Products List

BorderStyle 3-Fixed Dialog

Command1 Name CmdClose

Caption &Close

DataGrid1 DataSource DenvProducts

DataMember Products

AllowUpdate false

After properties of DataGrid are changed, change the size of data grid so that all columns of the grid can be displayed to user.

To resize grid to the size of columns:

1. Invoke Popup menu for DataGrid and select Retrieve Fields option.
2. DataGrid takes columns information from data source (Data Environment) and resizes the columns of the grid to the required size.
3. Now resize the grid in such a way that user can view all the columns. where DataGrid is resized by leaving some area on the right side for scrollbar.

Writing Code

Before we write code let us understand what are the important events for us, and what we do when those events occur.

Event Required Action

Selecting Save button Saves the current record to SALES table and creates a new blank

record. It also decreases QOH in Products table by qtySold.

Selecting Cancel button Erases the data already entered and keeps the form ready for new record.

Selecting ProductList button Displays frmProdList form, which displays the details of the products.

Selecting Quit button Terminates the application.(RPU)

Selecting a product form DataCombo control Immediately displays the rate per unit of the selected product.

Moving out of qtySold item Checks whether the entered amount of qty is really existing in the Products table.

Clicking on a row in DataGrid Places the selected product into dcmbProdno of the frmSales form so that when you come back you do not have to repeat the selection of the same product.

Selecting Close button of frmProdList Closes the form and returns to qtySold field of frmSales.

Table 20.5: Events to which we need to respond in sample application.

Code for frmSales form

Here is the code to take required actions for various events in frmSales.

```
Private Sub cmdCancel_Click()
```

```
With DenvProducts.rsSales
```

```
.CancelUpdate
```

```
.AddNew
```

```
ClearFields
```

```
End With
```

```
End Sub
```

```
Private Sub cmdList_Click()
```

```
frmprodlist.Show vbModal
```

```
' goto txtqtysold text box
```

```
txtqtysold.SetFocus
```

KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: III BCom CA COURSE NAME: SOFTWARE DEVELOPMENT WITH VISUAL BASIC
COURSE CODE: 16CCU501A UNIT: III BATCH-2016-19

End Sub

Private Sub cmdSave_Click()

With DenvProducts.rsSales

' save current record

.Update

UpdateProducts

.AddNew

ClearFields

End With

End Sub

Private Sub Cmdquit_Click()

Unload Me

End Sub

Private Sub dcmbProdno_Change()

GetRate

End Sub

Private Sub dcmbProdno_Click(Area As Integer)

GetRate

End Sub

Private Sub Form_Activate()

With DenvProducts.rsSales

If .EditMode = adEditNone Then

.AddNew

End If

End With

End Sub

Public Sub ClearFields()

txtqtysold.Text = ""

txtrpu.Text = ""

txtdiscount.Text = ""

dcmbProdno.SetFocus

End Sub

Private Sub txtqtysold_Validate(Cancel As Boolean)

' check whether quantity is sufficient

If txtqtysold.Text = "" Then

Exit Sub

End If

If Not check_qty Then

Cancel = True

End If

End Sub

Public Function check_qty() As Boolean

' get quantity on hand for the selected product

With DenvProducts.rsProducts

.Bookmark = dcmbProdno.SelectedItem

qoh = .Fields("qoh").Value

If qoh < CInt(txtqtysold.Text) Then

MsgBox "Insufficient Quantity on Hand"

check_qty = False

Else

```
check_qty = True
```

```
End If
```

```
End With
```

```
End Function
```

```
Public Sub GetRate()
```

```
If Not dcmbProdno.MatchedWithList Then
```

```
Exit Sub
```

```
End If
```

```
' get rate and put that into txtRpu
```

```
' move to the required record
```

```
With DenvProducts.rsProducts
```

```
.Bookmark = dcmbProdno.SelectedItem
```

```
txtrpu.Text = .Fields("rpu").Value
```

```
End With
```

```
End Sub
```

```
Public Sub UpdateProducts()
```

```
' Decrease QOH of PRODUCTS table by QTYSOLD
```

```
With DenvProducts.rsProducts
```

```
.Bookmark = dcmbProdno.SelectedItem
```

```
.Fields("qoh") = .Fields("qoh") - CInt(txtqtysold.Text)
```

```
.Update
```

```
End With
```

```
End Sub
```

Listing 20.3: Code for frmSalesEntry form

Code for frmProdList form

Here is the code for frmProdList form. The code takes care of placing selected product number into dcmbprodno of frmSales. Close button closes the form and control goes back to frmSales.

Private Sub cmdClose_Click()

Unload Me

End Sub

Private Sub DataGrid1_Click()

' pass the selected product number to data entry screen

frmSales.dcmdbProdno.BindText = DataGrid1.Columns(0).Value

End Sub

Reg No.....
16CCU501A

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University)

Established Under Section 3 of UGC Act 1956

COIMBATORE- 641021

(For the candidates admitted from 2016 onwards)

First Internal Examination – July 2018

B.COM CA - FIFTH SEMESTER

SOFTWARE DEVELOPMENT WITH VISUAL BASIC

Time : 2 HOURS

Maximum: 50 Marks

PART - A (20*1= 20 Marks)

Answer All the Questions

1. Visual Basic was developed from _____ programming language.
a) FORTRAN b) C c) BASIC d) C++
2. GUI stands for _____.
a) Graphical User Interface b) Graphical User Intraface
c) Graphical Used Intraface d) Graphical Used Interface
3. The _____ in the menu bar provides quick access to the commonly used commands.
a) toolbars b) toolbox c) form d) project window
4. The _____ event fires when the command button is clicked.
a) click b) change c) mouse move d) none
5. _____ helps to move and resize the controls and forms
a) label b) shape c) pointer d) OLE
6. The _____ event fires when mouse is moved over the textbox.
a) click b) change c) mouse move d) none
7. _____ property is set to change the height of a cell.
a) row height b) height row c) height d) none
8. VB is a technology developed by _____ company.
a) Microsoft b) Sun Microsystems c) IBM d) Apple computers
9. Data Members of a class are by default _____.
a) public b) static c) private d) none
10. _____ menu contains commands for opening and saving projects.
a) File b) View c) Edit d) Project
11. _____ window gives an overview of the solution we are working with and lists all the forms in the projects
a) Project Explorer b) Solution Explorer c) Server Explorer d) none
12. Objects that can be placed on a form are called _____.
a) Pictures b) Controls c) Label d) Buttons
13. To declare a variable, use the _____ statement followed by the variable's name, the As keyword and its type
a) Dim b) String c) Integer d) Single
14. Forms and controls in VB are called as _____.
a) events b) objects c) projects d) properties

15. VB is a _____ programming language.
a) object based b) procedure based c) event based d) application based
16. VB starts with a _____.
a) single blank b) standard exe c) add d) none
17. Variable declarations are of _____ types.
a) two b) three c) four d) one
18. VB projects are saved with _____ extensions.
a) *.vbp b) *.vbg c) *.vbf d) *.vbe
19. _____ property is used to change the content of a label.
a) Caption b) Name c) Font d) Int
20. _____ control is used to enter the text at run time.
a) Textbox b) Label c) Command Button d) Frames

PART - B (3*2= 6 Marks)

Answer all the Questions

21. What are the various windows available in VB Application?
22. What is a variable?
23. What is an event?

PART - C (3*8= 24 Marks)

Answer all the Questions

24. a) Write in detail about the standard toolbar
(OR)
b) Explain various datatypes in VB
25. a) Explain Integrated Development Environment
(OR)
b) Describe the properties of command button in VB Application
26. a) Explain various operators in VB
(OR)
b) Describe the steps in developing a VB Program for add two values along with form design

Reg No. 16CCU501A

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University)
Established Under Section 3 of UGC Act 1956
COIMBATORE- 641021

(For the candidates admitted from 2016 onwards)
Second Internal Examination – August 2018

B.COM CA - FIFTH SEMESTER
SOFTWARE DEVELOPMENT WITH VISUAL BASIC
Maximum: 50 Marks

Time : 2 HOURS

Date :

PART – A (20*1= 20 Marks)

Multiple choice Questions

1. _____ block is used for conditional execution of one or more statements.
a. if....then....else b. while c. do while d. none
2. The _____ function retrieves only the time.
a. now b. date c. time d. none
3. _____ block is used to define several blocks of statements, in order to execute one block.
a. if....then....else b. while c. select...case d. none
4. The _____ function retrieves only the date.
a. now b. date c. time d. none
5. Code in VB is stored in the form _____.
a. modules b. sub routines c. function d. none
6. The _____ function is used to find the length of the string.
a. strlen b. len c. length d. none
7. The _____ statement first executes the statement and then tests the condition after each execution.
a. do....while b. while c. select...case d. for
8. The _____ function returns the intervals between 2 dates in terms of years, months or dates.
a. format b. diffdate c. datediff d. diff
9. The _____ window is opened for the module to which the procedures are to be added.
a. form b. project c. code d. project explorer
10. _____ is an alternative to If...Then...Else statement.
a. while b. do while c. select...case d. for
11. The _____ function retrieves the date and time.
a. now b. date c. time d. none
12. The _____ contains the text box and list box
a. combo box b. shape box c. list box d. timer control
13. The _____ property returns the index of the last item added to the list.

- a. index number b. new index c. list index d. none
14. The _____ control adds a shape to a form.
a. image b. picture c. shape d. none
15. _____ is the style of simple combo box.
a. Style 0 b. Style 1 c. Style 2 d. Style 3
16. The _____ method is used to add items to a list at run time.
a. item b. index c. remove item d. add item
17. The resize event is fired whenever the size of the form is _____.
a. fixed b. not changed c. changed d. none
18. _____ is the style of dropdown list box.
a. Style 0 b. Style 1 c. Style 2 d. Style 3
19. The _____ method is used to remove an item from the list
a. item b. index c. remove item d. add item
20. The _____ property sets the index number of the currently selected item.
a. index number b. new index c. list index d. none

PART – B (3*2= 6 Marks)
Answer all the Questions

21. Describe properties of text box control.
22. Differentiate between Check Box and Combo Box.
23. What is the use of goto statement?

PART – C (3*8= 24 Marks)
Answer all the Questions

24. a) Explain Looping Statements in detail with example program.
(OR)
b) Explain sub routine and function program with example program.
25. a) Explain Date and Time functions and Numeric functions in VB.
(OR)
b) Explain Control Statements in detail with example program.
26. a) Explain different types of controls in vb.
(OR)
b) How the message box control work in VB environment.