

KARPAGAM ACADEMY OF HIGHER EDUCATION (Deemed to be University) (Established under Section 3 of UGC Act 1956) Eachanari post,Coimbatore-641 021.

Syllabus

~

17CTU302	PROGRAMMING	WITH VISUAL BASIC	Semester – III 4H – 4C
Instruction Hour	rs / week: L: 4 T: 0 P: 0	Marks: Int : 40 Ext : 60	Total: 100

SCOPE

Visual Basic .Net is emphasizing OOP more than ever before, scope has become even more important because much of the power of classes and objects is all about restricting scope and hiding implementation details to make things simpler.

OBJECTIVES

- This course introduces computer programming using the VISUAL BASIC programming language with object-oriented programming principles.
- Emphasis is on event-driven programming methods, including creating and manipulating objects, classes, and using object-oriented tools such as the class debugger.
- To identify the differences between the procedural languages and event driven languages. **UNIT I**

Beginning Visual Basic: Introduction to Visual Basic: Introduction Graphics User Interface (GUI), Programming Language (Procedural, Object oreinted, event driven), The Visual Basic environment IDE, Introduction to VB Controls: Textboxes, Frames, check boxes, options

environment IDE, Introduction to VB Controls: Textboxes, Frames, check boxes, options buttons, setting a border and style, the shape control, the line control, working with multiple controls and their properties, designing the user interface, keyboard access, tab controls. Default & controls property, coding for Controls, list box and combo box and their properties, filling the list box using property window/add item method, picture/image box and their properties.

UNIT II

Dealing with data: Operators-Variables-declaring variables- types of variables – data types – constants – arrays – declaring arrays – specifying arrays – Multidimensional arrays – dynamic arrays – arrays of arrays. Val function, Arithmetic operations, formatting data. Error functions and types. Introducing to Menu editor.

UNIT III

Writing Code: Control flow statements – If – Then – If-then-else – Nested control statements – Select case – Loop statements – Do-loop – For-Next – While Wend – Exit statement . Displaying message in Message box, testing whether input is valid or not. Collections – procedures – Subroutines – Functions – Calling procedures – Object Browser – Creating classes and Objects – I/O statements

UNIT IV

Working with forms and procedures: Introducing to forms and types of forms and setting form properties, creating, adding, removing forms in project, hide, show method, load, unload,

statement, Me keywords, Referring to objects on a different forms. Creating an application using controls: What is on the toolbar – Textbox control – Picture box – Image box – Label box – Frame – List box – Option button – Combo box – Command Button – check box – The Drive, Directory, File list controls – Teh Line & Shape control – Scroll Box – Data – Timer.

UNIT V

Multiple Document Interface & Menus: Why MDI Forms – Features of an MDI forms – Loading MDI forms & child forms – creating a simple MDI forms –Accessing MDI froms – creating MENUS – POP-UP MENUS.

Data access controls: JET database Engine – ADODC – DAO Data control – ODBC Data Source Administrator – DATA REPORT.

SUGGESTED READINGS

- 1. Noel Jerke (2008). Visual Basic 6.0: Teh Complete Reference. Tata McGraw Hill Publishing Company Ltd.
- 2. Mohammed Azam. Programming with VB 6.0. Vikas Publishing.
- 3. Peter Wrights (1999). Beginning VB 6.0 (4th ed.). New York:Springer-Verlag Incorporated.

REFERENCES

R1. LiewVoonKiong(2006). Visual Basic 6.0 Made Easy: A Complete tutorial for beginners. Booksurge,LLC

WEB SITES

W1. www.vbtutor.net/vbtutor.html

W2.www.freetutes.com/learn-vb6/

W3.www.visual-basic-tutorials.com/

W4.https://docs.microsoft.com/en-us/dotnet/visual-basic/programming- guide/language-features/objects-and-classes/

W5. www.tutorialpoint.com

W6.www.virtualsplat.com/tips/visual-basic-tutorials-gui-asp

W7. Zetcode.com/lang/visual basic/io/

W8. www.vb6.us/tutorials/Database

ESE Pattern		
Part – A (Online)	$20 \ge 1 = 20$	
Part – B	5 x 2 = 10	
Part – C (Either or)	5 x 6 = 30	
Total	60 marks	

Enable | Enlighten | Enrich (Deemed to be University) (Established Under Section 3 of UGC Act, 1956)

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University) (Established Under Section 3 of UGC Act 1956) Coimbatore – 641 021.

Department of Computer Science, Computer Applications & Information Technology

STAFF NAME: Dr.S.KOWSALYA SUBJECT NAME: PROGRAMMING WITH VISUAL BASIC SEMESTER: III

SUB.CODE:17CTU302

CLASS: II B.Sc. (CT)

UNIT I			
SL. No.	Lecture Duration (Hours)	Topics to be Covered	Support Materials
1.	1	Beginning Visual Basic: Introduction to Visual Basic, Advantages and Disadvantages of VB	SR1:3-9, W5
2.	1	Introduction Graphics User Interface (GUI), Programming Language (Procedural, Object oriented, event driven) -	SR1:23-56, W6, W1
3.	1	The Visual Basic environment IDE	SR1:23-56, W2, SR4:2-10
4.	1	Fextboxes, Frames, check boxes, options buttons, Setting a porder and style, the shape control, the line controlW1, W2, SR1:34 SR4:13-15	
5.	1	working with multiple controls and their properties	W1,W2,W6
6.	1	Designing the user interface, keyboard access, tab controls. W6, W1 Default & controls property	
7.	1	Coding for Controls, list box and combo box and theirW1,W6 properties, filling the list box using property window/add itemSR4:17-19 method.	
8.	1	Picture/image box and their properties.	W1,W6, SR4:16
9.	1	Recapitulation and Discussion of Important Questions	
Total No. of Hours Planned for Unit I : 9			
UNIT II			
SL. No.	Lecture Duration (Hours)	Topics to be Covered	Support Materials
1.	1	Dealing with data: Operators in VB SR4:30-32,W2	

Prepared by S.Kowsalya ,Department of CS,CA & IT, KAHE

2.	1	Variables-declaring variables- types of variables – Rules for Variables SR4:28-30,W2	
3.	1	Data types – Numeric Data types – Non numeric Data types - constants	SR4:26-28,W2,
4.	1	Arrays – declaring arrays – specifying arrays –	SR4:94-96,W2
5.	1	Multidimensional arrays	SR4:97-99,W2
6.	1	Dynamic arrays – Control Array	SR4:100-101,W2
7.	1	Val function, Arithmetic operations	SR4:66-68,82, W2
8.	1	Formatting data. Error functions and types, Introduction to Menu editor.	W2
9.	1	Recapitulation and Discussion of Important Questions	
		Total No. of Hours Planned for Unit II : 9	
		UNIT III	
SL. No.	Lecture Duration (Hours)	Topics to be Covered	Support Materials
1.	1	Writing Code: Control flow statements	SR4:34,W2,W6
2.	1	If – Then – If-then-else – Nested control statements	SR4:34-39,W2
3.	1	Select case	SR4:40-44,W2
4.	1	Loop statements – Do-loop – While wend – Exit statements	SR4:45-49,W2
5.		For-Next, Nested Loop, Displaying message in Message SR4:51-60,W2	
0.	1	For-Next, Nested Loop, Displaying message in Message box, testing whether input is valid or not. Collections	SR4:51-60,W2
6.	1	For-Next, Nested Loop, Displaying message in Message box, testing whether input is valid or not. Collections Inputbox function with example	SR4:51-60,W2 SR4:61:62,W2
6. 7.	1 1 1	For-Next, Nested Loop, Displaying message in Message box, testing whether input is valid or not. Collections Inputbox function with example Procedures – Subroutines- Functions, Calling procedures – Object Browser	SR4:51-60,W2 SR4:61:62,W2 W3
6. 7. 8.	1 1 1 1	For-Next, Nested Loop, Displaying message in Message box, testing whether input is valid or not. Collections Inputbox function with example Procedures – Subroutines- Functions, Calling procedures – Object Browser Creating classes and Objects, I/O statements	SR4:51-60,W2 SR4:61:62,W2 W3 W4

Prepared by S.Kowsalya ,Department of CS,CA & IT, KAHE

Total No. of Hours Planned for Unit III : 9				
		UNIT IV		
SL. No.	Lecture Duration (Hours)	Topics to be Covered	Support Materials	
1.	1	Working with forms and procedures: Introducing to forms and types	W2	
2.	1	Setting form properties, creating, adding, removing forms in project	W2	
3.	1	Hide, show method, load, unload, statement, Me keywords, Referring to objects on a different forms.	W2	
4.	1	Creating an application using controls: What is on the toolbar, Textbox control – Picture box	W2	
5.	1	Image box – Label box – Frame – List box – Option button	W2	
6.	1	Combo box – Command Button – check box, The Drive, Directory, File list controls	W2	
7.	1	Text Line & Shape control, Scroll Box – Date – Timer	W1,W2	
8.	1	Tree view and List view control	W2,W6	
9.	1	Recapitulation and Discussion of Important Questions		
Total No. of Hours Planned for Unit IV : 9				
UNIT V				
SL. No.	Lecture Duration (Hours)	Topics to be Covered	Support Materials	
1.	1	Multiple Document Interface & Menus: Why MDI Forms	W2	
2.	1	Features of an MDI forms – Loading MDI forms & child forms	W2	
3.	1	Creating a simple MDI forms –Accessing MDI forms	W2	

4.	1	creating MENUS – POP-UP MENUS	W2	
5.	1	Data access controls: JET database Engine - ADODC	SR4:232-236, W1,W8	
6.	1	DAO Data control	W1,W8	
7.	1	ODBC Data Source Administrator	W1,W8	
8.	1	DATA REPORT	W1,W8	
9.	1	Recapitulation and Discussion of Important Questions		
10.	1 Discussion of Previous ESE Question Papers			
11.	1	Discussion of Previous ESE Question Papers		
12.	1	Discussion of Previous ESE Question Papers		
	Total No. of Hours Planned for Unit V : 12			
	Total No. of Hours: 48			

Suggested Readings

- SR1: Noel Jerke (2008). Visual Basic 6.0: The Complete Reference. Tata McGraw Hill Publishing Company Ltd.
- SR2: Mohammed Azam. Programming with VB 6.0. Vikas Publishing.
- SR3: Peter Wrights (1999). Beginning VB 6.0 (4th ed.). New York:Springer-Verlag Incorporated.
- SR4: LiewVoonKiong(2006). Visual Basic 6.0 Made Easy: A Complete tutorial for beginners. Booksurge,LLC

Web Sites

- W1. www.vbtutor.net/vbtutor.html
- W2.www.freetutes.com/learn-vb6/
- W3.www.visual-basic-tutorials.com/
- W4.https://docs.microsoft.com/en-us/dotnet/visual-basic/programming-guide/language-features/objects-and-classes/
- W6. www.virtual splat. com/tips/visual-basic-tutorials-gui-asp
- W7. Zetcode.com/lang/visual basic/io/
- W8. www.vb6.us/tutorials/Database



KARPAGAM ACADEMY OF HIGHER EDUCATION CT) COURSE NAME : PROGRAMMING WITH VISUAL BASIC

CLASS : II B.Sc. (CT) CO COURSE CODE: 17CTU302 SEMESTER III

NAME : PROGRAMMING WITH VISUAL BASIC UNIT: I BATCH (2017-2020)

<u>UNIT I</u>

SYLLABUS

Beginning Visual Basic: Introduction to Visual Basic: Introduction Graphics User Interface (GUI), Programming Language (Procedural, Object oreinted, event driven), The Visual Basic environment IDE, Introduction to VB Controls: Textboxes, Frames, check boxes, options buttons, setting a border and style, the shape control, the line control, working with multiple controls and their properties, designing the user interface, keyboard access, tab controls. Default & controls property, coding for Controls, list box and combo box and their properties, filling the list box using property window/add item method, picture/image box and their properties.

BEGINNING VISUAL BASIC

INTRODUCTION TO 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 graphical environment. In the old BASIC, you have to write program code for each graphical



CLASS : II B.Sc. (CT)COURSE NAME : PROGRAMMING WITH VISUAL BASICCOURSE CODE: 17CTU302UNIT: ISEMESTER IIIBATCH (2017-2020)

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.

ADVANTAGES AND DISADVANTAGES OF VB

Advantages of Visual Basic

a) It is not just a language to program in but a whole graphical development environment. This aids your programming skills allowing you to concentrate on developing novel ideas instead of going over old ground.

b) It is quick to develop new programs. A newcomer will have a window proudly opening and greeting you with "Hello World!" - always the programmers first program - in less than 5 minutes.

c) OLE programming is simple. This allows you to embed objects such as Word documents and Excel spreadsheets with a minimum of fuss.

d) It can be used as a front end to SQL (or other databases) allowing the user to enhance the way they access their data.

e) It is widely used for in-house application program development and for prototyping.

f) It can also be used to create ActiveX and COM components for useonline or in desktop applications.

g) It is very simple to learn. As the name Basic suggests it uses easy to understand and remember terminology.



CLASS : II B.Sc. (CT)COURSE NAME : PROGRAMMING WITH VISUAL BASICCOURSE CODE: 17CTU302UNIT: ISEMESTER IIIBATCH (2017-2020)

h) Because of its popularity there are many resources available to the user - websites are numerous and books are plentiful for the programmer needing help.

i) Strict programming structures can be "turned off" to allow you to quickly develop a program - this could also be seen as a disadvantage when a bug arises!

j) Perhaps its strongest advantage is its simplicity. There is hardly any learning curve for programmers to begin learning the language or coming from another language.

k) VB is a "component integration language" which utilises Microsoft's Component Object Model ("COM") that allows parts to be bolted onto programs easily. These COM programs can be written in any language.

Disadvantages of Visual Basic

a) It is not suited to complex modern programming techniques. Because of its age little is being down to further the VB environment and it has been largely superceded by VB.net and other languages (even by Microsoft!).

b) Programs that are written in it tend not to be the quickest, this is largely due to the additional code that is often included that is not really necessary for your program to run.

c) VB is an interpreted language which again slows the execution of your program down.

d) VB programs require large libraries to be present on your PC to enable them to work. If you do not have them the programmer either has to supply them or you have to download them.

e) Because of its over-simplified approach VB can produce programmers that are sloppy in their work leading to workarounds having to be employed.

f) Because of its age VB does not allow many modern techniques such asObject Orientated Programming.



CLASS : II B.Sc. (CT)COURSE NAME : PROGRAMMING WITH VISUAL BASICCOURSE CODE: 17CTU302UNIT: ISEMESTER IIIBATCH (2017-2020)

g) As you can control the checking and warning systems in VB it often enables the programmer to write code that is very difficult to troubleshoot when a bug arises.

h) VB consists of features and syntax borrowed from other languages (often ones that are now no longer used). Watch a programmers face when you talk to them about "GoSub" and "On Error" commands!

i) OOP - Object Orientated Programming - is missing from VB, this is one of the most common techniques in all new languages allowing code to be easily reused (although this is available in VB.net)

j) There is no threading support (although this is also available inVB.net).

k) The programs a programmer produces in VB are not portable and cannot be used on non-Windows systems.

1) Mathematical performance is poor which slows down the speed of your program.

m) Service Packs! Everyone knows about Microsoft's love of service packs to fix the many bugs that have accumulated over time.

INTRODUCTION GRAPHICS USER INTERFACE (GUI)

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.



CLASS : II B.Sc. (CT) COU COURSE CODE: 17CTU302 SEMESTER III

COURSE NAME : PROGRAMMING WITH VISUAL BASIC 2 UNIT: I BATCH (2017-2020)

Getting Started



Figure 1.1

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. It Contains Properties, Event Procedures, Forms and Standard Controls

Properties:

The properties describe the appearance of the GUI component. When adding a component, the Name property should be set immediately, according to the three-letter mnemonic naming conventions. The properties are displayed in the Properties Window in Name/Value pairs in alphabetical order.

Event Procedures:

An event procedure is a piece of code that responds to events that can occur for that object. Most of the events are generated by the user, enabling them to dictate the order of execution.

Forms:

The Form is the main stage of your application. By default, the Standard Exe option starts with a form called "Form1". The Name property of the Form should be named with a three-letter mnemonic prefix of "frm". Each Form will be a Window in your application. Controls are added to the form by either double-clicking them in the toolbox, or by selecting the control and drawing a bounding rectangle on the form. Your application may use more than one form.

To add a new Form to the project, either select "Add Form" from the "Project" menu or right-click the Forms folder in the Project Explorer and select, "Add", and then "Form".



CLASS : II B.Sc. (CT)COURSE NAME : PROGRAMMING WITH VISUAL BASICCOURSE CODE: 17CTU302UNIT: ISEMESTER IIIBATCH (2017-2020)

To load a new form, use the Show method. The parameter, vbModal, is optional. If used, vbModal means that the form has focus until closed within the application.

The Load command can be used to load a form without showing it. This technique is useful if you want to preload a form, and then use either the "Show" or "Visible" method to make it visible as and when required.

Standard Controls:

Controls are added to the Form from the Toolbox. Each control has a set of properties, and a set of event procedures associated with it. The following lists the control, reading left to right, top to bottom as they appear in the standard Toolbox.

- ➢ Pointer.
- PictureBox Control.
- ➢ Label Control.
- TextBox Control.
- ➢ Frame Control.
- CommandButton Control.
- CheckBox Control.
- OptionButton Control.
- ComboBox Control.
- ListBox Control.
- Horizontal and Vertical Scroll bars.
- ➤ Timer Control.
- DriveListBox, DirListBox, and FileListBox.
- Shape Control.
- ➢ Line Control.
- ➢ Image Control.
- Data Control.
- > Object Linkking and Embedding (OLE) Control.



CLASS : II B.Sc. (CT) COUR COURSE CODE: 17CTU302 SEMESTER III

COURSE NAME : PROGRAMMING WITH VISUAL BASIC 2 UNIT: I BATCH (2017-2020)

PROGRAMMING LANGUAGE (PROCEDURAL, OBJECT ORIENTED, EVENT DRIVEN)

Programming Language

A program is a set of instructions following the rules

of the chosen language.

• Without programs, computers are useless.

• A program is like a recipe.

• It contains a list of ingredients (called variables) and a list of directions (called statements) that tell thecomputer what to do with the variables.

You eventually need to convert your program intomachine language so that the computer can understand it. There are two ways to do this:

- Compile the program
- Interpret the program

Procedural Language

The Procedural Programming Approach to Programming We will begin this course with a brief discussion of the programming methodologies that you are most likely accustom to in your previous Visual Basics.Net introductory courses. Programming as it was done in the past and still being done today in many cases is based on the Event-Driven and Procedural Programming approach. These methods of programming are based on what's known as Structured Programming. Structure programming has been the traditional way of programming.

Procedural Programming If you have taken a course in C, Visual Basic, Pascal, FORTRAN, Cobol etc. the programs you wrote were Procedural. In procedural programming, the focus of the programs was to solve a problem. For example, supposed you were asked to write a program to solve the following problem: Write a Video Management System that will process the rental/return of videos tapes for a retail store such as a program used for Blockbuster Video. Using a language like C or can be done even with VB.NET, this is usually done as follows:



CLASS : II B.Sc. (CT)COURSE NAME : PROGRAMMING WITH VISUAL BASICCOURSE CODE: 17CTU302UNIT: ISEMESTER IIIBATCH (2017-2020)

1. Analyze the problem required to be solved: Design flow chart, algorithm etc.

2. Break the problem into smaller manageable pieces, such as the Rental Module, Return Module, Customer Information Module etc.

3. Design the UI for each of the pieces using Forms, controls or any structure supplied by the language to implement the UI

4. Write code to implement each piece using variables, functions & procedures to implement each of the modular pieces. Note that the focus is on solving the problem via the programming code and breaking the problem into smaller manageable pieces.

Object oriented

- Visual Basic provides full support for object-oriented programming including encapsulation, inheritance, and polymorphism.
- Encapsulation means that a group of related properties, methods, and other members are treated as a single unit or object.
- > Inheritance describes the ability to create new classes based on an existing class.
- Polymorphism means that you can have multiple classes that can be used interchangeably, even though each class implements the same properties or methods in different ways. This section describes the following concepts:
- Classes and objects
 - Class members
 Properties and fields
 Methods
 - wichious
 - Constructors
 - Destructors
 - Events
 - Nested classes
 - Access modifiers and access levels



CLASS : II B.Sc. (CT) COURSI COURSE CODE: 17CTU302 SEMESTER III

COURSE NAME : PROGRAMMING WITH VISUAL BASIC 2 UNIT: I BATCH (2017-2020)

- Instantiating classes
- Shared classes and members
- Anonymous types
- > Inheritance
 - Overriding members
- ➢ Interfaces
- ➤ Generics
- > Delegates

Classes and objects

The terms *class* and *object* are sometimes used interchangeably, but in fact, classes describe the *type* of objects, while objects are usable *instances* of classes. So, the act of creating an object is called *instantiation*. Using the blueprint analogy, a class is a blueprint, and an object is a building made from that blueprint.

To define a class:

Class SampleClass EndClass

> Visual Basic also provides a light version of classes called structures that are useful when you need to create large array of objects and do not want to consume too much memory for that.

To define a structure:

```
Structure SampleStructure EndStruct
```

Methods

A method is an action that an object can perform. +



CLASS : II B.Sc. (CT)COURSE NAME : PROGRAMMING WITH VISUAL BASICCOURSE CODE: 17CTU302UNIT: ISEMESTER IIIBATCH (2017-2020)

Note

In Visual Basic, there are two ways to create a method: the Sub statement is used if the method does not return a value; the Function statement is used if a method returns a value.

To define a method of a class:

```
Class SampleClass
Public Function SampleFunc(ByVal SampleParam As String)
' Add code here
End Function
End Class
```

A class can have several implementations, or *overloads*, of the same method that differ in the number of parameters or parameter types.

To overload a method:

```
OverloadsSub Display(ByVal theChar As Char)
' Add code that displays Char data.
End Sub
OverloadsSub Display(ByValtheIntegerAsInteger)
' Add code that displays Integer data.
End Sub
```

In most cases you declare a method within a class definition. However, Visual Basic also supports extension methods that allow you to add methods to an existing class outside the actual definition of the class.

Constructors

Constructors are class methods that are executed automatically when an object of a given type is created. Constructors usually initialize the data members of the new object. A constructor can run only once when a class is created. Furthermore, the code in the constructor always runs before any other code in a class. However, you can create multiple constructor overloads in the same way as for any other method.

To define a constructor for a class:



CLASS : II B.Sc. (CT) COU COURSE CODE: 17CTU302 SEMESTER III

COURSE NAME : PROGRAMMING WITH VISUAL BASIC 2 UNIT: I BATCH (2017-2020)

VBCopy

Class SampleClass

Sub New(ByVal s As String)

// Add code here.

End Sub

End Class

For more information, see: Object Lifetime: How Objects Are Created and Destroyed.

Destructors

Destructors are used to destruct instances of classes. In the .NET Framework, the garbage collector automatically manages the allocation and release of memory for the managed objects in your application. However, you may still need destructors to clean up any unmanaged resources that your application creates. There can be only one destructor for a class.

For more information about destructors and garbage collection in the .NET Framework, see Garbage Collection.

Events

Events enable a class or object to notify other classes or objects when something of interest occurs. The class that sends (or raises) the event is called the *publisher* and the classes that receive (or handle) the event are called *subscribers*. For more information about events, how they are raised and handled, see Events.

- To declare events, use the Event Statement.
- To raise events, use the RaiseEvent Statement.
- To specify event handlers using a declarative way, use the With Events statement and the Handles clause.

To be able to dynamically add, remove, and change the event handler associated with an event, use the AddHandler Statement and RemoveHandler Statement together with the AddressOf Operator.



CLASS : II B.Sc. (CT) COUF COURSE CODE: 17CTU302 SEMESTER III

COURSE NAME : PROGRAMMING WITH VISUAL BASIC 2 UNIT: I BATCH (2017-2020)

Nested classes

A class defined within another class is called *nested*. By default, the nested class is private.

```
Class Container
Class Nested
' Add code here.
EndClass
EndClass
```

Access modifiers and access levels

All classes and class members can specify what access level they provide to other classes by using *access modifiers*.

The following access modifiers are available:

Visual Basic Modifier	Definition
Public	The type or member can be accessed by any other code in the same assembly or another assembly that references it.
Private	The type or member can only be accessed by code in the same class.
Protected	The type or member can only be accessed by code in the same class or in a derived class.
Friend	The type or member can be accessed by any code in the same assembly, but not from another assembly.
Protected Friend	The type or member can be accessed by any code in the same assembly, or by any derived class in another assembly.



CLASS : II B.Sc. (CT) COU COURSE CODE: 17CTU302 SEMESTER III

COURSE NAME : PROGRAMMING WITH VISUAL BASIC 2 UNIT: I BATCH (2017-2020)

Instantiating classes

To create an object, you need to instantiate a class, or create a class instance.

Dim sampleObject as New SampleClass()

After instantiating a class, you can assign values to the instance's properties and fields and invoke class methods.

Inheritance

Inheritance enables you to create a new class that reuses, extends, and modifies the behavior that is defined in another class. The class whose members are inherited is called the *base class*, and the class that inherits those members is called the *derived class*. However, all classes in Visual Basic implicitly inherit from the Object class that supports .NET class hierarchy and provides low-level services to all classes.

Note :Visual Basic doesn't support multiple inheritance. That is, you can specify only one base class for a derived class.

To inherit from a base class:

ClassDerivedClass InheritsBaseClass EndClass

By default all classes can be inherited. However, you can specify whether a class must not be used as a base class, or create a class that can be used as a base class only.

To specify that a class cannot be used as a base class:

NotInheritableClassSampleClass EndClass



CLASS : II B.Sc. (CT) COU COURSE CODE: 17CTU302 SEMESTER III

COURSE NAME : PROGRAMMING WITH VISUAL BASIC 2 UNIT: I BATCH (2017-2020)

Interfaces

Interfaces, like classes, define a set of properties, methods, and events. But unlike classes, interfaces do not provide implementation. They are implemented by classes, and defined as separate entities from classes. An interface represents a contract, in that a class that implements an interface must implement every aspect of that interface exactly as it is defined.

To define an interface:

```
PublicInterfaceISampleInterface
SubDoSomething()
EndInterface
```

To implement an interface in a class:

```
ClassSampleClass
ImplementsISampleInterface
SubDoSomething
' Method implementation.
End Sub
EndClass
```

Generics

Classes, structures, interfaces and methods in .NET can include *type parameters* that define types of objects that they can store or use. The most common example of generics is a collection,

where you can specify the type of objects to be stored in a collection.

To define a generic class:

```
Class SampleGeneric(Of T)
Public Field As T
EndClass
```

To create an instance of a generic class:

```
Dim sampleObject As New SampleGeneric(OfString)
```



CLASS : II B.Sc. (CT) COU COURSE CODE: 17CTU302 SEMESTER III

COURSE NAME : PROGRAMMING WITH VISUAL BASIC 2 UNIT: I BATCH (2017-2020)

sampleObject.Field = "Sample string"

Delegates

A *delegate* is a type that defines a method signature, and can provide a reference to any method with a compatible signature. You can invoke (or call) the method through the delegate. Delegates are used to pass methods as arguments to other methods. Note

Event handlers are nothing more than methods that are invoked through delegates. For more information about using delegates in event handling, see <u>Events</u>.

To create a delegate:

```
DelegateSubSampleDelegate(ByValstrAsString)
```

To create a reference to a method that matches the signature specified by the delegate:

```
ClassSampleClass
' Method that matches the SampleDelegate signature.
SubSampleSub(ByValstrAsString)
' Add code here.
End Sub
' Method that instantiates the delegate.
SubSampleDelegateSub()
DimsdAsSampleDelegate = AddressOfSampleSub
sd("Sample string")
EndSub
EndClass
```

Event Driven

Visual Basic is an event-driven programming language. Before proceeding to the next chapter, it is very important to have a good concept of event-driven programming. The common events are



CLASS : II B.Sc. (CT)COURSE NAME : PROGRAMMING WITH VISUAL BASICCOURSE CODE: 17CTU302UNIT: ISEMESTER IIIBATCH (2017-2020)

Click, DblClick, Load, MouseMove, MouseDown, MouseUp, KeyPress, KeyUp, KeyDown, GotFocus, LostFocus, etc.

When you click, press a key, move the mouse or fire other events, the particular block of code of the corresponding event procedure is executed, and then the program behaves in a certain way. This is called event-driven programming.

When you fire an event, the code in the event procedure is executed, and then visual basic performs its operations as per the instructions written in the event procedure code. For example, in the first sample program, when you click the 'Print' button, the click event is fired, and then the code in the click event procedure gets executed. The code tells Visual Basic to print a text on the form. So as a result, you see a text printed on the form.

Example:

Write the following code in the DblClick event procedure of the form.

Private Sub Form_DblClick()

Print "You have double-clicked" End Sub



CLASS : II B.Sc. (CT) COU COURSE CODE: 17CTU302 SEMESTER III

COURSE NAME : PROGRAMMING WITH VISUAL BASIC 2 UNIT: I BATCH (2017-2020)

Output:

🔄 Form1	
You have double-clicked	

Figure 1.2

When you double-click on the form, the DblClick event procedure of the Form object is invoked,

and then the code in the DblClick event procedure is executed. Thus, the code instructs Visual

Basic to print a text on the form.

THE VISUAL BASIC ENVIRONMENT IDE

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

Figure 1.2The Visual Basic Start-up Dialog Box



CLASS : II B.Sc. (CT) CO COURSE CODE: 17CTU302 SEMESTER III

COURSE NAME : PROGRAMMING WITH VISUAL BASIC 2 UNIT: I BATCH (2017-2020)

		n 	• 🗖		~ /		
Microsoft Vis	New Project					? ×	
Elle Edit View		Micros	oft ual B	asi	C		×
•	New Existing	g Recent				1	
	D	2	and the second s		2	-	
	Standard EXE	ActiveX EXE	ActiveX DLL	ActiveX Control	VB Application Wizard		×
	2	B.	5	5	5	priz	ed]
	VB Wizard Manager	Data Project	IIS Application	Addin	ActiveX Document Dll		
	I Pa 🏠		Fa &		<u>O</u> pen		
					Cance		×
					Help		1
	Don't show th	is dialog in the f <u>i</u>	lture				
-							
			Figure 1	.3			

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 $\xrightarrow{}$ open project.



CLASS : II B.Sc. (CT) CO COURSE CODE: 17CTU302 SEMESTER III

COURSE NAME : PROGRAMMING WITH VISUAL BASIC 2 UNIT: I BATCH (2017-2020)

Visual Basics Environment



Figure 1.4

Visual Basic is a high level programming paradigm. Its concepts are based upon <u>Event driven</u> <u>programming</u>. 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 <u>Toolbox</u> control which allows us to add objects on to Form window. We can change the properties using the <u>properties</u> windows for all the objects on the form. We can also edit/create the event handlers using the <u>Code</u>Window. When creating applications in Visual Basic it is quite common to use multiple forms, modules etc. The <u>project explorer</u> 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.



CLASS : II B.Sc. (CT)COURSE NAME : PROGRAMMING WITH VISUAL BASICCOURSE CODE: 17CTU302UNIT: ISEMESTER IIIBATCH (2017-2020)

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 y 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



CLASS : II B.Sc. (CT) CO COURSE CODE: 17CTU302 SEMESTER III

COURSE NAME : PROGRAMMING WITH VISUAL BASIC 2 UNIT: I BATCH (2017-2020)



Docked on the right side of the screen, just under the tool bar is the Project Explorer Window. The Project Explorer Window as show 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 - Form1 \times Form1 Form -Alphabetic Categorized Name) Form1 -Appearance 1 - 3D AutoRedraw Ealse ackColor 8H800000F& orderStyle 2 - Sizable aption Form1 lipControls True ontrolBox True)rawMode)rawStyle 13 - Copy Pen 0 - Solid . rawWidth Enabled 8H00000008 illColor -illStyle 1 - Transparent MS Sans Serif ont ontTranspar rent True ■ &H80000012& oreColor leight lelpContextID 0 (Icon) eyPreview False

PROPERTIES WINDOW

Figure 1.6

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



CLASS : II B.Sc. (CT) CO COURSE CODE: 17CTU302 SEMESTER III

COURSE NAME : PROGRAMMING WITH VISUAL BASIC 2 UNIT: I BATCH (2017-2020)



Figure 1.7

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.







CLASS : II B.Sc. (CT)COURSE NAME : PROGRAMMING WITH VISUAL BASICCOURSE CODE: 17CTU302UNIT: ISEMESTER IIIBATCH (2017-2020)

PictureBox	Displays icons/bitmaps and metafiles. It displays text or acts as a visual container for other controls.	
TextBox	Used to display message and enter text.	
Frame	Serves as a visual and functional container for controls	
CommandButton	Used to carry out the specified action when the user chooses it.	
CheckBox	Displays a True/False or Yes/No option.	
OptionButton	OptionButton control which is a part of an option group allows the user to select only one option even it displays mulitiple choices.	
ListBox	Displays a list of items from which a user can select one.	
ComboBox	Contains a TextBox and a ListBox. This allows the user to select an ietm from the dropdown ListBox, or to type in a selection in the TextBox.	
HScrollBar and VScrollBar	These controls allow the user to select a value within the specified range of values	
Timer	Executes the timer events at specified intervals of time	
DriveListBox	Displays the valid disk drives and allows the user to select one of them.	
DirListBox	Allows the user to select the directories and paths, which are displayed.	
FileListBox	Displays a set of files from which a user can select the desired one.	



CLASS : II B.Sc. (CT)COURSE NAME : PROGRAMMING WITH VISUAL BASICCOURSE CODE: 17CTU302UNIT: ISEMESTER IIIBATCH (2017-2020)

Shape	Used to add shape (rectangle, square or circle) to a Form	
Line	Used to draw straight line to the Form	
Image	used to display images such as icons, bitmaps and metafiles. But less capability than the PictureBox	
Data	Enables the use to connect to an existing database and display information from it.	
OLE	Used to link or embed an object, display and manipulate data from other windows based applications.	
Label	Displays a text that the user cannot modify or interact with.	



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



CLASS : II B.Sc. (CT)COURSE NAME : PROGRAMMING WITH VISUAL BASICCOURSE CODE: 17CTU302UNIT: ISEMESTER IIIBATCH (2017-2020)

(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.

Immediate, Locals, And WatchWindows

These additional windows are provided for use in debugging your application. They are only available when you are running your application within the IDE.

PROPERTIES WINDOW

Besides a Project Explorer window, each form (or control) contains a Properties window. Properties are used to change the appearance or behavior of objects. Figure illustrates the default Properties dialog box for a form. If this box is not visible, you can bring it into view by pressing the F4 function key, or choosing View, Properties Window. With the form having the focus, press your right mouse button and drag down the Properties list.

Properties - Form1 🛛 🛛 🛛			
Form1 Form	-		
Alphabetic Cate	Alphabetic Categorized		
(Name)	Form1		
Appearance	1 - 3D		
AutoRedraw	False		
BackColor	8H800000F&		
BorderStyle	2 - Sizable		
Caption	Form1		
ClipControls	True		
ControlBox	True		
DrawMode	13 - Copy Pen		
DrawStyle	0 - Solid		
DrawWidth	1		
Enabled	True		
FillColor	8H00000008		
FillStyle	1 - Transparent		
Font	MS Sans Serif		
FontTransparent	True		
ForeColor	8H800000128		
Height	3600		
HeinContextID	n 💌		
Caption Returns/sets the text displayed in an object's title bar or below an object's icon.			

Figure 1.10

- Seven properties deal with the way in which the form or text appears on the display screen (Caption, BorderStyle, MinButton, MaxButton, ControlBox, WindowSize and StartUpPosition).
- The identification of an object is specified by the Name property.
- Font settings are controlled using the Fonts dialog box (FontName, FontSize, FontBold, FontItalic and FontStrikethru).
- Color settings (ForeColor and BackColor) use a special palette.



CLASS : II B.Sc. (CT)COURSE NAME : PROGRAMMING WITH VISUAL BASICCOURSE CODE: 17CTU302UNIT: ISEMESTER IIIBATCH (2017-2020)

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 \rightarrow Code. The Code Window has the following sections see figure



Figure 1.11

- *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*: IntelliSence 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 suntax for that element. For Example, Quickinfo feature work for the Msgbox Statement, which pops up a simple box.



CLASS : II B.Sc. (CT) CO COURSE CODE: 17CTU302 SEMESTER III

COURSE NAME : PROGRAMMING WITH VISUAL BASIC 2 UNIT: I BATCH (2017-2020)

1	Project1 - Form1 (Code)	
Fo	rm 🔽 Load	
	Private Sub Form_Load() msgbox(
	MsgBox(Prompt, [Buttons As VbMsgBo	xStyle = vbOKOnly], [<i>Title</i>], [<i>HelpFile</i>], [<i>Context</i>]) As VbMsgBoxRes
3		

Figure 1.12

SETTING A BORDER AND STYLE, THE SHAPE CONTROL, THE LINE CONTROL

How to Set Border Style

You can use BorderStyle property to add a border to the label control. This property is typically used to differentiate a Label that labels another control from a Label that displays the status of a process in an application.

The following code example demonstrates how to set different type of bordestype to a Label control.

FixedSingle :

Private Sub CreateBorder_FixedSingle()

Label1.BorderStyle = BorderStyle.FixedSingle

End Sub

Fixed3D :

Private Sub CreateBorder_Fixed3D()

Label1.BorderStyle = BorderStyle.Fixed3D

End Sub

Using the Shape control

There are four basic controls in VB6 that you can use to draw graphics on your form: the line control, the shape control, the image box and the picture box. To draw a straight line, just click on the line control and then use your mouse to draw the line on the form. After drawing the line,



CLASS : II B.Sc. (CT)COURSE NAME : PROGRAMMING WITH VISUAL BASICCOURSE CODE: 17CTU302UNIT: ISEMESTER IIIBATCH (2017-2020)

you can then change its color, width and style using the BorderColor, BorderWidth and BorderStyleproperties.Similarly, to draw a shape, just click on the shape control and draw the shape on the form. The default shape is a rectangle, with the default shape property set at 0. You can change the shape to square, oval, circle and rounded rectangle by changing the shape property's value to 1, 2, 3, 4, and 5 respectively. In addition, you can change its background color using the BackColor property, its border style using the BorderStyle property, its border color using the BorderColor property as well its border width using the BorderWidth property.

Example

This example will draw a circle, a sector, an arc and an ellipse with different colors and different DrawStyles

The Code

Dim pi As Single

Private Sub Form_Load() 'Change the coordinates of the origin Scale (-4000, 4000)-(4000, -4000) Me.Width = 8000 Me.Height = 8000

End Sub

Private Sub cmd_Draw_Click(Index As Integer)

pi = 4 * Atn(1) Select Case Index Case Is = 0 Me.FillColor = vbRed Me.FillStyle = vbSolid Me.DrawStyle = vbDot Me.DrawWidth = 1 Me.Circle (0, 0), 800, vbYellow ' Draw a circle with red border Case Is = 1 Me.FillColor = vbCyan



COURSE NAME : PROGRAMMING WITH VISUAL BASIC CLASS: II B.Sc. (CT) **COURSE CODE: 17CTU302 SEMESTER III**

UNIT: I **BATCH (2017-2020)**

Me.FillStyle = vbSolid Me.DrawStyle = vbDot Me.DrawWidth = 3Me.Circle (-2000, 0), 800, vbBlack, -pi, -pi / 2 Case Is = 2Me.FillColor = vbYellow Me.FillStyle = vbSolid Me.DrawStyle = vbDashDot Me.DrawWidth = 1Me.Circle (-2000, -2000), 800, vbBlue, 0, pi Case Is = 3Me.FillColor = vbBlue Me.FillStyle = vbSolid Me.DrawStyle = vbDashDotDot Me.DrawWidth = 1Me.Circle (-1500, 2000), 800, vbCyan, , , 1.3 End Select

End Sub



CLASS : II B.Sc. (CT)COURSE NAME : PROGRAMMING WITH VISUAL BASICCOURSE CODE: 17CTU302UNIT: ISEMESTER IIIBATCH (2017-2020)

The Output



Figure 1.13

The Line Control

The Line Method

Although the Pset method can be used to draw a straight line on the form, it is a little slow. It is better to use the Line method if you want to draw a straight line faster. The format of the Line command is shown below. It draws a line from the point (x1, y1) to the point (x2, y2) and the color constant will determine the color of the line.

Line (x1, y1)-(x2, y2), color


CLASS : II B.Sc. (CT)COURSE NAME : PROGRAMMING WITH VISUAL BASICCOURSE CODE: 17CTU302UNIT: ISEMESTER IIIBATCH (2017-2020)

For example, the following command will draw a red line from the point (0, 0) to the point (1000, 2000).

Line (0, 0)-(1000,2000), VbRed

The Line method can also be used to draw a rectangle. The syntax is

Line (x1-y1)-(x2, y2), color, B

The four corners of the rectangle are (x1-y1), (x2-y1), (x1-y2) and (x2, y2)

Another variation of the Line method is to fill the rectangle with a certain color. The syntax is

Line (x1, y1)-(x2, y2), color, BF

If you wish to draw the graphics in a picture box, you can use the following syntaxes

Picture1.Line (x1, y1)-(x2, y2), color Picture1.Line (x1-y1)-(x2, y2), color, B Picture1.Line (x1-y1)-(x2, y2), color, BF Picture1.Circle (x1, y1), radius, color

Example The Bar Graph Plotter

We shall use the knowledge we gained from the Line method to create a bar graph. In this program, we shall insert a picture box for drawing the bar graph. In addition, we insert six text boxes for accepted the user input. We also insert two command buttons for drawing and reseting. Besides that, we need to define a new origin using the **Scale method**, otherwise, the bar graph will be upside down. The code is

Picture1.Scale (0, 5000)-(5000, 0)

The Code

Private Sub Command1_Click() Dim sale1, sale2, sale3, sale4, sale5, sale6 As Integer

sale1 = Val(Txt_Jan.Text)
sale2 = Val(Txt_Feb.Text)



CLASS: II B.Sc. (CT) **COURSE NAME : PROGRAMMING WITH VISUAL BASIC COURSE CODE: 17CTU302 SEMESTER III**

UNIT: I **BATCH (2017-2020)**

sale3 = Val(Txt_Mac.Text) sale4 = Val(Txt_Apr.Text) sale5 = Val(Txt_May.Text) sale6 = Val(Txt Jun.Text)

Picture1.Line (100, 0)-(600, sale1 * 50), vbRed, BF Picture1.Line (700, 0)-(1200, sale2 * 50), vbRed, BF Picture1.Line (1300, 0)-(1800, sale3 * 50), vbRed, BF Picture1.Line (1900, 0)-(2400, sale4 * 50), vbRed, BF Picture1.Line (2500, 0)-(3000, sale5 * 50), vbRed, BF Picture1.Line (3100, 0)-(3600, sale6 * 50), vbRed, BF Picture1.Line (3700, 0)-(4200, sale7 * 50), vbRed, BF Picture1.Line (4300, 0)-(4800, sale8 * 50), vbRed, BF

End Sub

Private Sub Command2_Click() Txt Jan.Text = "" Txt Feb.Text = "" Txt Mac.Text = "" Txt_Apr.Text = "" Txt May.Text = "" Txt_Jun.Text = ""

Picture1.Cls

End Sub

Private Sub Form_Load() 'To redefine the coordinates of the origin Picture1.Scale (0, 5000)-(5000, 0)

End Sub



CLASS : II B.Sc. (CT) COURS COURSE CODE: 17CTU302 SEMESTER III

COURSE NAME : PROGRAMMING WITH VISUAL BASIC 2 UNIT: I BATCH (2017-2020)

The Output



Figure 1.14



CLASS : II B.Sc. (CT) COU COURSE CODE: 17CTU302 SEMESTER III

COURSE NAME : PROGRAMMING WITH VISUAL BASIC 2 UNIT: I BATCH (2017-2020)

POSSIBLE QUESTIONS

Part – B (2 Marks)

- 1. What is Event Driven Programming?
- 2. What is Graphical user interface?
- 3. Differentiate between GUI and CUI?
- 4. Write any two features of VB
- 5. Write any three advantages of Visual Basic.

Part – C (6 Marks)

- List the events for the following controls with example (i) TextBox (ii) ListBox (iii) Option Button
- 2. Explain the following (i) Project Explorer (ii) Properties (iii) Code window
- 3. Explain Programming Languages in Visual Basic
- 4. Explain (i) Define Keyboard Access Keys (ii) Setting Tab-Order for Controls
- 5. Explain how to work with multiple controls with their properties.
- 6. Write a program to draw several shapes and fill with different colors.
- 7. Explain components of Integrated Development Environment in detail.
- 8. Explain Tool Box in Visual Basic 6.0.
- 9. Explain Visual Basic Development Environment in detail.
- 10. Discuss about List Box and Combo Box in detail

(Deemed University)

(Established Under Section 3 of UGC Act 1956)

Coimbatore - 641 021

(For the candidates admitted in 2017 onwards)

CLASS : II B.Sc CT

SUBJECT : Programming with visual basic

	UNIT-1				
Questions	opt1	opt2	opt3	opt4	Answer
visual basic was developed form the					
programming language	FORTRAN	С	BASIC	C++	BASIC
visual bais is developed in the year	1978	1980	1970	1972	1970
A complete installation of the most powerful version of visual basic 6.0, the enterprise edition requires of hard disk space	250MB	250GB	460MB	460GB	250MB
Visual Basic 6.0 requires of					
RAM	18MB	16MB	18GB	16GB	16MB
	Integrated development	Integrated developed environmen	Integration developme nt environme	nono	Integrated developmen t environmen
IDE is also commonly referred to as	environment	ı developme	111	none	L
environment	interface	nt	design	integrated	desian
IDE was designed as a -	OLE	SDI	MDI	none	SDI
The windows associated with the project will stay with in a single container called	child	parent	code	none	parent
quick access to the commonly used commands	toolbars	toolbox	project window	form	toolbars
helps to move and resize the controls and forms	label	shape	OLE	pointer	pointer
action when the user choose it	option button	control	control	button	button
messages and enter text	tool box	text box	button	button	text box
also acts as a coontainer for child forms and some controls is a window that contains an	SDI	MDI	IDE	none	MDI
application code and has other objects place	properties	menu	form	border	form

is an actioon that can be					
performed on objects	form	move	click	method	method
control enables the user to					
connect to an existing database and display					
information from it.	object	properties	data	file	data
serves as a window that can be	-				
customized and controls graphics and pictures					
can also be added to it.	object	properties	form	file	form
Each and every form in an application is	· ·				
considered an	obiect	properties	file	none	obiect
Characteristics of an object are called its	,				,
· · · · · · · · · · · · · · · · · · ·	obiects	properties	project	none	properties
method is used to display the form			[]		
object	load	visible	show	display	show
method is used to load a					0.1011
form or control into memory but doesnot					
display it	load	visible	show	display	load
displays a text that the user	1000	VISIBIC	311011	alopidy	1000
displays a text that the user	lahel	text box	timer	line	lahel
has the zorder event in it	list box	Me flox arid	arid	label	liet box
flas the 2010er event in it.		IVIS HEX GILU	gnu		
	arid	Mefloyarid	DB arid	none	Mefloyarid
	gnu		DD gliu	none	wishexyriu
diaplays information at run time	atatua bar	button	toxt box	rich toyt boy	toxt box
displays information at full time	status bai	imaga			
	snape	antrol	nieture boy	abaak bay	antrol
the code entered in the	control	control		CHECK DOX	control
the code entered in the event mes					
when there is a change in the contents of the		allala	mouse		
text box.	cnange	CIICK	move	none	cnange
the event fires when the textbox			mouse		
control is clicked.	change	CIICK	move	none	CIICK
the event fires when the mouse			mouse		mouse
is moved over the textbox	change	click	move	none	move
A complete repaint of a form or control can be					
enforced by method	refresh	set focus	change	repaint	refresh
is used to link or embed object,					- · -
display and manipulate data from other	.VBP	.FRM	OLE	none	OLE
is the named attribute of a					
programming object	property	window	object	none	property
is the named attribute of a					
programming object	window	object	property	none	property
lets windows decide whrer	center		center	windows	windows
the form should be shown	screen	manual	owner	default	default
indicates whether the window is	window	windows			windows
shown normally, maximized or minimized.	state	default	both	none	default
event occurs when the form is					
closed by user.	-				
	load	unload	show	none	unioad
Writing a VB program involves	load	unload	show	none	unioad

steps involves designing an	visual	code			visual
application with various tools that come along	programmin	programmi			programmin
with VB package	g	ng	both	none	g
	visual	code			code
steps involves writing programs	programmin	programmi			programmin
using text editor.	g	ng	both	none	g
0	0	code			code
is the name property for the form		programmi	form		programmin
object	display prog	na	display	none	a
is the caption property for the		5			
command button if name is cmdexit	&exit	& clear	&display	none	&exit
is the name property for the					
command button if caption property is &clear	cmdexit	cmdclear	cmddisplay	none	cmdclear
command battor in capiton property is delear.	onidexit	omacical	omaalopiay	none	omaoicai
is the caption property for					
the command button if name is cmddisplay	&ovit	& clear	& display	none	& display
and command buttor in name is circulsplay	GEAIL	u ciedi	auispiay		
is the name property for the					
is the name property for the	omdovit	omdoloor	amddiaplay	nono	amddiaplay
command button if caption is adisplay	CITUEXIL	cinuclear			
VD project files are equal with an externion					
vo project liles are saved with an extension					
form files are saved with an extension	.VBP		.OLE	.EXE	
har Parks a dia ana at					
box displays the name of					
the selected object associated with the form	object	tool	text	command	object
Intersection of row and column is	coll	record	field	none	cell
the cole and rows properties are used to	Cell	Tecolu		none	
determine the number of columns and rows in					
	flowgrid	mo flovarid	arid		mo flovarid
	nexgna	ms nexgria	gna	none	ins nexgrid
event occurs whenever the size	- :		h alah t		
or form is analiged	size	resize	neight	width	resize
Kinds of rows/columns are					
created in the mattering contori.	4	3	2	1	2
		Vaa		VOO	
Ms flexgrid control is an control	UCX	XCO	COX	XUC	
user can change the current cell at run time			I		
using	mouse	arrow keys	both	none	both
To display the wrapped text, we need to					
the cell's column width	increase	decrease	normal	hide	increase
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
To add the flexgrid, choose complnents from			-		I man to at
To add the flexgrid, choose complnents from	browser	property	project	none	project
To add the flexgrid, choose complnents from menu After setting the properties, the ms flexgrid	browser	property	project	none	project
To add the flexgrid, choose complnents from menu After setting the properties, the ms flexgrid control is enlarged vertically and horizontally	browser	property	project	none	project
To add the flexgrid, choose complnents from menu After setting the properties, the ms flexgrid control is enlarged vertically and horizontally by its handles.	browser	property resizina	project hidina	none dragaina	dragging
To add the flexgrid, choose complnents from menu After setting the properties, the ms flexgrid control is enlarged vertically and horizontally by its handles.	browser moving	property resizing	project hiding	none dragging	dragging

Ms flexgrid has the properties	col	cols	both	none	both
property is set to change the					
height of a cell	row height	height row	height	none	row height



CLASS : II B.Sc. (CT) **COURSE CODE: 17CTU302 SEMESTER III**

COURSE NAME : PROGRAMMING WITH VISUAL BASIC UNIT: II **BATCH (2017-2020)**

UNIT II

Dealing with data: Operators-Variables-declaring variables- types of variables - data types constants - arrays - declaring arrays - specifying arrays - Multidimensional arrays dynamic arrays – arrays of arrays. Val function, Arithmetic operations, formatting data. Error functions and types. Introducing to Menu editor.

OPERATORS

An operator is a symbol that tells the compiler to perform specific mathematical or logical manipulations. VB is rich in built-in operators and provides following types of commonly used operators:

- Arithmetic Operators
- Comparison Operators
- Logical/Bitwise Operators
- Bit Shift Operators
- Assignment Operators
- Miscellaneous Operators

Arithmetic Operators

Following table shows all the arithmetic operators supported by VB. Assume variable **A** holds 2 and variable **B** holds 7, then: Show Examples

Operator	Description	Example
٨	Raises one operand to the power of another	B^A will give 49
+	Adds two operands	A + B will give 9
-	Subtracts second operand from the first	A - B will give -5
*	Multiplies both operands	A * B will give 14
/	Divides one operand by another and returns a floating point result	B / A will give 3.5
/	Divides one operand by another and returns an integer result	$\mathbf{B} \setminus \mathbf{A}$



CLASS : II B.Sc. (C1) COURSE CODE: 17CTU302 SEMESTER III		UNIT: II BATCH (2017-2020)		
			will give 3	
MOD	Modulus Operate	or and remainder of after an integer division	B MOD A will give 1	

Comparison Operators

Following table shows all the comparison operators supported by VB. Assume variable **A** holds 10 and variable **B** holds 20, then: Show Examples

Operator	Description	Example
=	Checks if the values of two operands are equal or not; if yes, then condition becomes true.	(A = B) is not true.
\Leftrightarrow	Checks if the values of two operands are equal or not; if values are not equal, then condition becomes true.	(A <> B) is true.
>	Checks if the value of left operand is greater than the value of right operand; if yes, then condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand; if yes, then condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand; if yes, then condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand; if yes, then condition becomes true.	(A <= B) is true.

Apart from the above, VB provides three more comparison operators, which we will be using in forthcoming chapters; however, we give a brief description here.

- Is Operator It compares two object reference variables and determines if two object references refer to the same object without performing value comparisons. If object1 and object2 both refer to the exact same object instance, result is **True**; otherwise, result is False.
- **IsNot** Operator It also compares two object reference variables and determines if two object references refer to different objects. If object1 and object2 both refer to the exact same object instance, result is **False**; otherwise, result is True.
- Like Operator It compares a string against a pattern.

Logical/Bitwise Operators



CLASS : II B.Sc. (CT) CO COURSE CODE: 17CTU302 SEMESTER III

COURSE NAME : PROGRAMMING WITH VISUAL BASIC UNIT: II

BATCH (2017-2020)

Following table shows all the logical operators supported by VB. Assume variable A holds Boolean value True and variable B holds Boolean value False, then:

Show Examples

Operator	Description	Example
And	It is the logical as well as bitwise AND operator. If both the operands are true, then condition becomes true. This operator does not perform short-circuiting, i.e., it evaluates both the expressions.	(A And B) is False.
Or	It is the logical as well as bitwise OR operator. If any of the two operands is true, then condition becomes true. This operator does not perform short-circuiting, i.e., it evaluates both the expressions.	(A Or B) is True.
Not	It is the logical as well as bitwise NOT operator. Use to reverses the logical state of its operand. If a condition is true, then Logical NOT operator will make false.	Not(A And B) is True.
Xor	It is the logical as well as bitwise Logical Exclusive OR operator. It returns True if both expressions are True or both expressions are False; otherwise it returns False. This operator does not perform short-circuiting, it always evaluates both expressions and there is no short-circuiting counterpart of this operator.	A Xor B is True.
AndAlso	It is the logical AND operator. It works only on Boolean data. It performs short-circuiting.	(A AndAlso B) is False.
OrElse	It is the logical OR operator. It works only on Boolean data. It performs short-circuiting.	(A OrElse B) is True.
IsFalse	It determines whether an expression is False.	
IsTrue	It determines whether an expression is True.	

Bit Shift Operators

We have already discussed the bitwise operators. The bit shift operators perform the shift operations on binary values. Before coming into the bit shift operators, let us understand the bit operations.

Bitwise operators work on bits and perform bit-by-bit operations. The truth tables for &, |, and $^$ are as follows:

p q p & q p \ q p ^ q	
-----------------------	--



CLASS : II B.Sc. (COURSE CODE: SEMESTER III	(CT) COU 17CTU302	RSE NAME : PRO UNIT BAT	GRAMMING WIT] : II CH (2017-2020)	H VISUAL BASIC
0	0	0	0	0
0	1	0	1	1
1	1	1	1	0
1	0	0	1	1

Assume if A = 60; and B = 13; now in binary format they will be as follows: A = 0011 1100

 $B = 0000 \ 1101$ _____

 $A\&B = 0000\ 1100$ $A|B = 0011 \ 1101$ $A^B = 0011\ 0001$ $\sim A = 1100\ 0011$

We have seen that the Bitwise operators supported by VB are And, Or, Xor and Not. The Bit shift operators are >> and << for left shift and right shift, respectively.

Assume that the variable A holds 60 and variable B holds 13, then:

Show Examples

Operator	Description	Example
And	Bitwise AND Operator copies a bit to the result if it exists in both operands.	(A AND B) will give 12, which is 0000 1100
Or	Binary OR Operator copies a bit if it exists in either operand.	(A Or B) will give 61, which is 0011 1101
Xor	Binary XOR Operator copies the bit if it is set in one operand but not both.	(A Xor B) will give 49, which is 0011 0001
Not	Binary Ones Complement Operator is unary and has the effect of 'flipping' bits.	(Not A) will give - 61, which is 1100 0011 in 2's complement form due to a signed binary



CLASS : II B.Sc. (CT)	COURSE NAME : PROGRAMMING WITH VISUAL BASIC
COURSE CODE: 17CTU302	UNIT: II
SEMESTER III	BATCH (2017-2020)

		number.
<<	Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.	A << 2 will give 240, which is 1111 0000
>>	Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.	A >> 2 will give 15, which is 0000 1111

Assignment Operators

There are following assignment operators supported by VB: Show Examples

Operator	Description	Example
=	Simple assignment operator, Assigns values from right side operands to left side operand	C = A + B assign value of $A + B$ into C
+=	Add AND assignment operator, It adds right operand to the left operand and assigns the result to left operand	C += A is equivalent to C = C + A
-=	Subtract AND assignment operator, It subtracts right operand from the left operand and assigns the result to left operand	C = A is equivalent to $C = C - A$
*_	Multiply AND assignment operator, It multiplies right operand with the left operand and assigns the result to left operand	C *= A is equivalent to C = C * A
/=	Divide AND assignment operator, It divides left operand with the right operand and assigns the result to left operand (floating point division)	C /= A is equivalent to C = C / A
\=	Divide AND assignment operator, It divides left operand with the	$C \models A$ is



CLASS : II COURSE C SEMESTER	B.Sc. (CT) COURSE NAME : PROGRAMMING WITH VIS ODE: 17CTU302 UNIT: II R III BATCH (2017-2020)	UAL BASIC
	right operand and assigns the result to left operand (Integer division)	equivalent to $C = C$ $\setminus A$
^=	Exponentiation and assignment operator. It raises the left operand to the power of the right operand and assigns the result to left operand.	$C^A = A$ is equivalent to $C = C^A$ A
<<=	Left shift AND assignment operator	C <<= 2 is same as C = C << 2
>>=	Right shift AND assignment operator	$C \implies 2$ is same as $C = C \implies 2$
&=	Concatenates a String expression to a String variable or property and assigns the result to the variable or property.	Str1 &= Str2 is same as Str1 Str1 &= Str1 & Str2 Str2

Miscellaneous Operators

There are few other important operators supported by VB. Show Examples

Operator	Description	Example
AddressOf	Returns the address of a procedure.	AddHandler Button1.Click, AddressOf Button1_Click
Await	It is applied to an operand in an asynchronous method or lambda expression to suspend execution of the method until the awaited task completes.	Dim result As res = Await AsyncMethodThatReturnsResult() Await AsyncMethod()
GetType	It returns a Type object for the specified type. The Type object provides information about the	MsgBox(GetType(Integer).ToString())



CLASS : II B.Sc. (CT)	COURSE NAME : PROGRAMMING WITH VISUAL BASIC
COURSE CODE: 17CTU302	UNIT: II
SEMESTER III	BATCH (2017-2020)

	type such as its properties, methods, and events.	
Function Expression	It declares the parameters and code that define a function lambda expression.	Dim add5 = Function(num As Integer) num + 5 'prints 10 Console.WriteLine(add5(5))
If	It uses short-circuit evaluation to conditionally return one of two values. The If operator can be called with three arguments or with two arguments.	Dim num = 5 Console.WriteLine(If(num >= 0, "Positive", "Negative"))

Operators Precedence in VB

Operator precedence determines the grouping of terms in an expression. This affects how an expression is evaluated. Certain operators have higher precedence than others; for example, the multiplication operator has higher precedence than the addition operator:

For example, x = 7 + 3 * 2; here, x is assigned 13, not 20 because operator * has higher precedence than +, so it first gets multiplied with 3*2 and then adds into 7.

Here, operators with the highest precedence appear at the top of the table, those with the lowest appear at the bottom. Within an expression, higher precedence operators will be evaluated first.

Show Examples

Operator	Precedence
Await	Highest
Exponentiation (^)	
Unary identity and negation (+, -)	
Multiplication and floating-point division (*, /)	
Integer division (\)	
Modulus arithmetic (Mod)	
Addition and subtraction (+, -)	
Arithmetic bit shift (<<, >>)	
All comparison operators (=, <>, <, <=, >, >=, Is, IsNot, Like, TypeOfIs)	



CLASS : II B.Sc. (CT) COURSE CODE: 17CTU302 SEMESTER III	I B.Sc. (CT) COURSE NAME : PROGRAMMING WITH VISUAL BASIC CODE: 17CTU302 UNIT: II ER III BATCH (2017-2020)			
Negation (Not)				
Conjunction (And, AndAlso)				
Inclusive disjunction (Or, OrEl	se)			
Exclusive disjunction (Xor)		Lowest		

VARIABLES

Variables are the memory locations which are used to store values temporarily. A defined naming strategy has to be followed while naming a variable. A variable name must begin with an alphabet letter and should not exceed 255 characters. It must be unique within the same scope. It should not contain any special character like %, &, !, #, @ or \$.

There are many ways of declaring variables in Visual Basic. Depending on where the variables are declared and how they are declared, we can determine how they can be used by our application. The different ways of declaring variables in Visual Basic are listed below and elucidated in this section.

- Explicit Declaration
- Using Option Explicit statement
- Scope of Variables

Explicit Declaration

Declaring a variable tells Visual Basic to reserve space in memory. It is not must that a variable should be declared before using it. Automatically whenever Visual Basic encounters a new variable, it assigns the default variable type and value. This is called implicit declaration. Though this type of declaration is easier for the user, to have more control over the variables, it is advisable to declare them explicitly. The variables are declared with a Dim statement to name the variable and its type. The As type clause in the Dim statement allows to define the data type or object type of the variable. This is called explicit declaration. Syntax

Dim variable [As Type] For example, Dim strName As String Dim intCounter As Integer

Using Option Explicit statement

It may be convenient to declare variables implicitly, but it can lead to errors that may not be recognized at run time. Say, for example a variable by name *intcount* is used implicitly and is assigned to a value. In the next step, this field is incremented by 1 by the following statement

Intcount = Intcount + 1

This calculation will result in *intcount* yielding a value of 1 as intcount would have been initialized to zero. This is because the intcount variable has been mityped as incont in the



CLASS: II B.Sc. (CT)

KARPAGAM ACADEMY OF HIGHER EDUCATION

COURSE NAME : PROGRAMMING WITH VISUAL BASIC 'U302 UNIT: II

COURSE CODE: 17CTU302 SEMESTER III

BATCH (2017-2020)

right hand side of the second variable. But Visual Basic does not see this as a mistake and considers it to be new variable and therefore gives a wrong result.

In Visual Basic, to prevent errors of this nature, we can declare a variable by adding the following statement to the general declaration section of the Form.

Option Explicit

This forces the user to declare all the variables. The Option Explicit statement checks in the module for usage of any undeclared variables and reports an error to the user. The user can thus rectify the error on seeing this error message.

The Option Explicit statement can be explicitly placed in the general declaration section of each module using the following steps.

- Click Options item in the Tools menu
- Click the Editor tab in the Options dialog box
- Check Require Variable Declaration option and then click the OK button

Scope of variables

A variable is scoped to a procedure-level (local) or module-level variable depending on how it is declared. The scope of a variable, procedure or object determines which part of the code in our application are aware of the variable's existence. A variable is declared in general declaration section of e Form, and hence is available to all the procedures. Local variables are recognized only in the procedure in which they are declared. They can be declared with *Dim* and *Static* keywords. If we want a variable to be available to all of the procedures within the same module, or to all the procedures in an application, a variable is declared with broader scope.

Local Variables

A local variable is one that is declared inside a procedure. This variable is only available to the code inside the procedure and can be declared using the Dim statements as given below. Dim sum As Integer

The local variables exist as long as the procedure in which they are declared, is executing. Once a procedure is executed, the values of its local variables are lost and the memory used by these variables is freed and can be reclaimed. Variables that are declared with keyword Dim exist only as long as the procedure is being executed.

Static Variables

Static variables are not reinitialized each time Visual Invokes a procedure and therefore retains or preserves value even when a procedure ends. In case we need to keep track of the number of times a command button in an application is clicked, a static counter variable has to be declared. These static variables are also ideal for making controls alternately visible or invisible. A static variable is declared as given below.

Static intPermanent As Integer

Variables have a lifetime in addition to scope. The values in a module-level and public variables are preserved for the lifetime of an application whereas local variables declared with Dim exist only while the procedure in which they are declared is still being executed. The value of a local variable can be preserved using the Static keyword. The following



CLASS : II B.Sc. (CT)COURSE NAME : PROGRAMMING WITH VISUAL BASICCOURSE CODE: 17CTU302UNIT: IISEMESTER IIIBATCH (2017-2020)

procedure calculates the running total by adding new values to the previous values stored in the static variable value.



CLASS : II B.Sc. (CT)COURSE NAME : PROGRAMMING WITH VISUAL BASICCOURSE CODE: 17CTU302UNIT: IISEMESTER IIIBATCH (2017-2020)Function RunningTotal ()

SEMESTER III Function RunningTotal () Static Accumulate Accumulate = Accumulate + num RunningTotal = Accumulate End Function

If the variable Accumulate was declared with Dim instead of static, the previously accumulated values would not be preserved accross calls to the procedure, and the procedure would return the same value with which it was called. To make all variables in a procedure static, the Static keyword is placed at the beginning of the procedure heading as given in the below statement.

Static Function RunningTotal ()

Example

The following is an example of an event procedure for a CommandButton that counts and displays the number of clicks made.

Private Sub Command1_Click () Static Counter As Integer Counter = Counter + 1 Print Counter End Sub

Modules

Code in Visual Basic is stored in the form of modules. The three kind of modules are Form Modules, Standard Modules and Class Modules. A simple application may contain a single Form, and the code resides in that Form module itself. As the application grows, additional Forms are added and there may be a common code to be executed in several Forms. To avoid the duplication of code, a separate module containing a procedure is created that implements the common code. This is a standard Module.

Class module (.CLS filename extension) are the foundation of the object oriented programming in Visual Basic. New objects can be created by writing code in class modules.

Each module can contain:

Declarations : May include constant, type, variable and DLL procedure declarations.

Procedures : A sub function, or property procedure that contain pieces of code that can be executed as a unit.

These are the rules to follow when naming elements in VB - variables, constants, controls, procedures, and so on:

- A name must begin with a letter.
- May be as much as 255 characters long (but don't forget that somebody has to type the stuff!).
- Must not contain a space or an embedded period or type-declaration characters used to specify a data type; these are ! # % \$ & @
- Must not be a reserved word (that is part of the code, like Option, for example)



CLASS : II B.Sc. (CT) COURSE NAME : PROGRAMMING WITH VISUAL BASIC UNIT: II

SEMESTER III

BATCH (2017-2020)

• The dash, although legal, should be avoided because it may be confused with the minus sign. Instead of First-name use First_name or FirstName.

Data types in Visual Basic 6

By default Visual Basic variables are of variant data types. The variant data type can store numeric, date/time or string data. When a variable is declared, a data type is supplied for it that determines the kind of data they can store. The fundamental data types in Visual Basic including variant are integer, long, single, double, string, currency, byte and boolean. Visual Basic supports a vast array of data types. Each data type has limits to the kind of information and the minimum and maximum values it can hold. In addition, some types can interchange with some other types. A list of Visual Basic's simple data types are given below.

1. Numeric

Byte	Store integer values in the range of 0 - 255			
Integer	Store integer values in the range of (-32,768) - (+ 32,767)			
Long	Store integer values in the range of (- 2,147,483,468) - (+ 2,147,483,468)			
Single	Store floating point value in the range of (-3.4x10-38) - (+ 3.4x1038)			
Double	Store large floating value which exceeding the single data type value			
Currency	store monetary values. It supports 4 digits to the right of decimal point and 15 digits to the left			

2. String

Use to store alphanumeric values. A variable length string can store approximately 4 billion characters

3. Date

Use to store date and time values. A variable declared as date type can store both date and time values and it can store date values 01/01/0100 up to 12/31/9999

4. Boolean

Boolean data types hold either a true or false value. These are not stored as numeric values and cannot be used as such. Values are internally stored as -1 (True) and 0 (False) and any non-zero value is considered as true.

5. Variant

Stores any type of data and is the default Visual Basic data type. In Visual Basic if we declare a variable without any data type by default the data type is assigned as default.

ARRAYS

An array is a consecutive group of memory locations that all have the same name and the same type. To refer to a particular location or element in the array, we specify the array name and the array element position number.

The Individual elements of an array are identified using an index. Arrays have upper and lower bounds and the elements have to lie within those bounds. Each index number in an array is allocated individual memory space and therefore users must evade declaring arrays of larger size than required. We can declare an array of any of the basic data types including variant, user-defined types and object variables. The individual elements of an array are all of the same data type.



CLASS : II B.Sc. (CT)COURSE NAME : PROGRAMMING WITH VISUAL BASICCOURSE CODE: 17CTU302UNIT: IISEMESTER IIIBATCH (2017-2020)

Declaring arrays

Arrays occupy space in memory. The programmer specifies the array type and the number of elements required by the array so that the compiler may reserve the appropriate amount of memory. Arrays may be declared as Public (in a code module), module or local. Module arrays are declared in the general declarations using keyword Dim or Private. Local arrays are declared in a procedure using Dim or Static. Array must be declared explicitly with keyword "As".

There are two types of arrays in Visual Basic namely:

• **Fixed-size array :** The size of array always remains the same-size doesn't change during the program execution.

• **Dynamic array :** The size of the array can be changed at the run time- size changes during the program execution.

Fixed-sized Arrays

When an upper bound is specified in the declaration, a Fixed-array is created. The upper limit should always be within the range of long data type.

Declaring a fixed-array

Dim numbers(5) As Integer

In the above illustration, numbers is the name of the array, and the number 6 included in the parentheses is the upper limit of the array. The above declaration creates an array with 6 elements, with index numbers running from 0 to 5.

If we want to specify the lower limit, then the parentheses should include both the lower and upper limit along with the To keyword. An example for this is given below.

Dim numbers (1 To 6) As Integer

In the above statement, an array of 10 elements is declared but with indexes running from 1 to 6.

A public array can be declared using the keyword Public instead of Dim as shown below. Public numbers(5) As Integer

Multidimensional Arrays

Arrays can have multiple dimensions. A common use of multidimensional arrays is to represent tables of values consisting of information arranged in rows and columns. To identify a particular table element, we must specify two indexes: The first (by convention) identifies the element's row and the second (by convention) identifies the element's column.

Tables or arrays that require two indexes to identify a particular element are called two dimensional arrays. Note that multidimensional arrays can have more than two dimensions. Visual Basic supports at least 60 array dimensions, but most people will need to use more than two or three dimensional-arrays.

The following statement declares a two-dimensional array 50 by 50 array within a procedure. Dim AvgMarks (50, 50)

It is also possible to define the lower limits for one or both the dimensions as for fixed size arrays. An example for this is given here.

Dim Marks (101 To 200, 1 To 100)

An example for three dimensional-array with defined lower limits is given below. Dim Details(101 To 200, 1 To 100, 1 To 100)



CLASS : II B.Sc. (CT) C COURSE CODE: 17CTU302 SEMESTER III

COURSE NAME : PROGRAMMING WITH VISUAL BASIC UNIT: II BATCH (2017-2020)

Static and dynamic arrays

Basically, you can create either static or dynamic arrays. Static arrays must include a fixed number of items, and this number must be known at compile time so that the compiler can set aside the necessary amount of memory. You create a static array using a Dim statement with a constant argument:

'This is a static array.

Dim Names(100) As String

Visual Basic starts indexing the array with 0. Therefore, the preceding array actually holds 101 items.

Most programs don't use static arrays because programmers rarely know at compile time how many items you need and also because static arrays can't be resized during execution. Both these issues are solved by dynamic arrays. You declare and create dynamic arrays in two distinct steps. In general, you declare the array to account for its visibility (for example, at the beginning of a module if you want to make it visible by all the procedures of the module) using a Dim command with an empty pair of brackets. Then you create the array when you actually need it, using a ReDim statement:

' An array defined in a BAS module (with Private scope) Dim Customers() As String

Sub Main() ' Here you create the array. ReDim Customer(1000) As String End Sub

If you're creating an array that's local to a procedure, you can do everything with a single ReDim statement:

Sub PrintReport() ' This array is visible only to the procedure. ReDim Customers(1000) As String ' ... End Sub

If you don't specify the lower index of an array, Visual Basic assumes it to be 0, unless an Option Base 1 statement is placed at the beginning of the module. My suggestion is this: Never use an Option Base statement because it makes code reuse more difficult. (You can't cut and paste routines without worrying about the current Option Base.) If you want to explicitly use a lower index different from 0, use this syntax instead:

ReDim Customers(1 To 1000) As String



CLASS: II B.Sc. (CT) **COURSE NAME : PROGRAMMING WITH VISUAL BASIC COURSE CODE: 17CTU302 SEMESTER III**

UNIT: II BATCH (2017-2020)

Dynamic arrays can be re-created at will, each time with a different number of items. When you re-create a dynamic array, its contents are reset to 0 (or to an empty string) and you lose the data it contains. If you want to resize an array without losing its contents, use the ReDim Preserve command:

ReDim Preserve Customers(2000) As String

When you're resizing an array, you can't change the number of its dimensions nor the type of the values it contains. Moreover, when you're using ReDim Preserve on a multidimensional array, you can resize only its last dimension:

ReDim Cells(1 To 100, 10) As Integer

ReDim Preserve Cells(1 To 100, 20) As Integer ' This works. ReDim Preserve Cells(1 To 200, 20) As Integer ' This doesn't.

Finally, you can destroy an array using the Erase statement. If the array is dynamic, Visual Basic releases the memory allocated for its elements (and you can't read or write them any longer); if the array is static, its elements are set to 0 or to empty strings.

You can use the LBound and UBound functions to retrieve the lower and upper indices. If the array has two or more dimensions, you need to pass a second argument to these functions to specify the dimension you need:

Print LBound(Cells, 1) 'Displays 1, lower index of 1st dimension Print LBound(Cells) ' Same as above Print UBound(Cells, 2) ' Displays 20, upper index of 2nd dimension 'Evaluate total number of elements. NumEls = $(UBound(Cells) _ LBound(Cells) + 1) * _$ $(UBound(Cells, 2) _ LBound(Cells, 2) + 1)$

ARRAY OF ARRAYS

While you can create two-dimensional arrays in Visual Basic, their structure isn't really flexible for at least two reasons: All rows in the array must have the same number of elements, and you can use ReDim Preserve to change the number of columns but you can't add new rows. The first point is especially important because it often leads you to declare an array that's far too large for your needs, thus allocating a lot of memory that in most cases remains largely unused. You can solve both problems using a structure known as an array of arrays.

The technique is conceptually simple: Since you can store an array in a Variant variable, you can build an array of Variants, where each item holds an array. Each subarray-a row of this pseudo-array-can hold a different number of elements, and you don't need to use more memory than is strictly necessary.



CLASS : II B.Sc. (CT) COURSE CODE: 17CTU302 SEMESTER III





Here's an example, based on an imaginary PIM (Personal Information Manager) program. In this program, you need to keep track of a list of appointments for each day of the year. The simplest solution would be to use an array in which each row corresponds to a day in the year and each column to a possible appointment. (For the sake of simplicity, let's assume that each appointment's data can be held in a string.)

ReDim apps(1 To 366, 1 To MAX_APPOINTMENTS) As String

Of course, you now have the problem of setting a reasonable value for the MAX_APPOINTMENTS symbolic constant. It should be high enough to account for all possible appointments in a day but not too high because you might be wasting a lot of memory without any real reason. Let's see how the array of arrays technique can help us save memory without posing any artificial limit to your application:

```
' A module-level variable
Dim apps(1 To 366) As Variant
'Add an appointment for a given day.
Sub AddNewAppointment(day As Integer, description As String)
Dim arr As Variant
If IsEmpty(apps(day)) Then
' This is the first appointment for this day.
apps(day) = Array(description)
Else
'Add the appointment to those already scheduled.
arr = apps(day)
ReDim Preserve arr(0 To UBound(arr) + 1) As Variant
arr(UBound(arr)) = description
apps(day) = arr
End If
End Sub
'Extract all the appointments for a given day.
Sub ListAppointments(day As Integer, lst As ListBox)
Dim i As Long
For i = 0 To UBound(apps(1))
```



CLASS : II B.Sc. (CT) COURSE CODE: 17CTU302 SEMESTER III lst.AddItem apps(1)(i) Next End Sub

COURSE NAME : PROGRAMMING WITH VISUAL BASIC UNIT: II BATCH (2017-2020)

Memory can be saved if each row of this array were built using an array of a more specific data type (String, in this case). Note the special syntax used to address an item in an array of arrays:

'Change the description for the Nth appointment. apps(day)(n) = newDescription

Introducing an array of arrays of arrays, and so on. If it is dealt with arrays in which each row can vary considerably in length, this approach is going to save a lot of memory and, in most cases, improve overall performance too. A key feature of an array of arrays is that you can process entire rows of your pseudo-array as if they were single entities. For example, you can swap them, replace them, add and delete them, and so on.

'Move the January 1st appointments to January 2nd.

apps(2) = apps(1)

apps(1) = Empty

Finally, an important advantage of this technique is that you can add new rows without losing the current contents of the array. (Remember that you can use ReDim Preserve on regular arrays only to modify the number of columns, not the number of rows.)

'Extend the appointment book for another nonleap year.

ReDim Preserve apps(1 to UBound(apps) + 365) As Variant

The Format Function

Formatting output is an important part of programming so that the visual interface can be presented clearly to the users. However, to better present the output, we can use a number of formatting functions in Visual Basic.

The three most common formatting functions in VB are Tab, Space, and Format

The Tab function

The syntax of a Tab function is **Tab** (**n**); **x**

The item x will be displayed at a position that is n spaces from the left border of the output form. There must be a semicolon in between Tab and the items you intend to display

Private Sub Form_Activate Print "I"; Tab(5); "like"; Tab(10); "to"; Tab(15); "learn"; Tab(20); "VB" Print Print Tab(10); "I"; Tab(15); "like"; Tab(20); "to"; Tab(25); "learn"; Tab(20); "VB" Print Print Tab(15); "I"; Tab(20); "like"; Tab(25); "to"; Tab(30); "learn"; Tab(35); "VB" End sub



CLASS : II B.Sc. (CT) COURSE CODE: 17CTU302 SEMESTER III

COURSE NAME : PROGRAMMING WITH VISUAL BASIC UNIT: II BATCH (2017-2020)

	orm	1						
1 11	ke	to I	learr like l	to like	Visu lear to	ial Basic n Vi learn	sual Basic Visual Basic	



The Space function

The **Space** function is very closely linked to the Tab function. However, there is a minor difference. While Tab (n) means the item is placed n spaces from the left border of the screen, the Space function specifies the number of spaces between two consecutive items. For example, the procedure

Private Sub Form_Activate() Print "Visual"; Space(10);"Basic" End Sub Means that the words Visual and Basic will be separated by 10 spaces

The Format function

The **Format** function is a very powerful formatting function which can display the numeric values in various forms. There are two types of Format function, one of them is the built-in or predefined format while another one can be defined by the users.

(a) The syntax of the predefined Format function is

Format (n, "style argument")

where n is a number and the list of style arguments is given in below table

Style argument	Explanation	Example
General Number	To display the number without having separators between thousands.	Format(8972.234, "General Number")=8972.234
Fixed	To display the number without having	Format(8972.2,

Table 12.1: List of Style Arguments



CLASS : II B.Sc. (CT)COURSE NAME : PROGRAMMING WITH VISUAL BASICCOURSE CODE: 17CTU302UNIT: IISEMESTED IIIBATCH (2017, 2020)

SEMESTER	I BATCH (2017-2020)					
	separators between thousands and rounds it up to two decimal places.	"Fixed")=8972.23				
Standard	To display the number with separators or separators between thousands and rounds it up to two decimal places.	Format(6648972.265, "Standard")= 6,648,972.27				
Currency	To display the number with the dollar sign in front has separators between thousands as well as rounding it up to two decimal places.	Format(6648972.265, "Currency")= \$6,648,972.27				
Percent	Converts the number to the percentage form and displays a % sign and rounds it up to two decimal places.	Format(0.56324, "Percent")=56.32 %				

Private Sub Form_Activate()

Print Format (8972.234, "General Number")

Print Format (8972.2, "Fixed")

Print Format (6648972.265, "Standard")

Print Format (6648972.265, "Currency")

Print Format (0.56324, "Percent")

End Sub

🗟 Form1

8972.234 8972.20 6,648,972.27 \$6,648,972.27 56.32%

Figure 2.3



CLASS : II B.Sc. (CT) CO COURSE CODE: 17CTU302 SEMESTER III

COURSE NAME : PROGRAMMING WITH VISUAL BASIC UNIT: II BATCH (2017-2020)

The syntax of the user-defined Format function is

Format (n, "user's format")

Format	Description	Output
Format(781234.576,"0")	Rounds to whole number without separators between thousands	781235
Format(781234.576,"0.0")	Rounds to 1 decimal place without separators between thousands	781234.6
Format(781234.576,"0.00")	Rounds to 2 decimal place without separators between thousands	781234.58
Format(781234.576,"#,##0.00")	Rounds to 2 decimal place with separators between thousands	781,234.58
Format(781234.576,"\$#,##0.00")	Displays dollar sign and Rounds to 2 decimal place with separators between thousands	\$781,234.58
Format(0.576,"0%")	Converts to percentage form without decimal place	58%
Format(0.5768,"0%")	Converts to percentage form with two decimal places	57.68%

Private Sub Form_Activate() Print Format(781234.57,"0") Print Format(781234.57, "0.0") Print Format(781234.576,"0.00") Print Format(781234.576,"#,##0.00") Print Format(781234.576,"\$#,##0.00") Print Format(0.576, "0%") Print Format(0.5768, "0.00%") End Sub

ERRORS HANDLING

Introduction

Error handling is an essential procedure in Visual Basic programming because it can help make the program error-free. An error-free program can run smoothly and efficiently, and the user does not have to face all sorts of problems such as program crash or system hang. Errors often occur due to incorrect input from the user. For example, the user might make the mistake of attempting to ask the computer to divide a number by zero which will definitely

cause a system error. Another example is the user might enter a text (string) to a box that is



CLASS: II B.Sc. (CT) **COURSE NAME : PROGRAMMING WITH VISUAL BASIC COURSE CODE: 17CTU302 UNIT: II SEMESTER III BATCH (2017-2020)**

designed to handle only numeric values such as the weight of a person, the computer will not be able to perform an arithmetic calculation for text, therefore will create an error. These errors are known as synchronous errors. Therefore a good programmer should be more alert to the parts of the program that could trigger errors and should write errors handling code to help the user in managing the errors. Writing errors handling code should be considered a good practice for Visual Basic programmers, so do not try to finish a program fast by omitting the errors handling code. However, there should not be too many errors handling code in the program as it creates problems for the programmer to maintain and troubleshoot the program later.

The syntax for errors handling is On Error GoTo program_label

where *program* label is the section of code that is designed by the programmer to handle the error committed by the user. Once an error is detected, the program will jump to the *program_label* section for error handling.

Example: Division by Zero

Private Sub CmdCalculate Click() Dim firstNum, secondNum As Double firstNum = Txt FirstNumber.Text secondNum = Txt SecondNumber.Text On Error GoTo error_handler Lbl_Answer.Caption = firstNum / secondNum Exit Sub

'To prevent error handling even the inputs are valid error_handler: Lbl_Answer.Caption = "Error" Lbl_ErrorMsg.Visible = True Lbl ErrorMsg.Caption = "You attempt to divide a number by zero!Try again!" End Sub

Private Sub Txt FirstNumber GotFocus() Lbl_ErrorMsg.Visible =False End Sub

Private Sub Txt_SecondNumber_GotFocus() Lbl_ErrorMsg.Visible = False End Sub

Explanation:

In this example, you design the interface. Name the first textbox as Txt_FirstNumber and the second textbox as Txt_SecondNumber. Insert one command button as label it as Calculate. Insert another label and name it as Lbl Answer to display the answers. If the user enter 0 in the second textbox as shown above, the program will jump to the label error_handler, and



CLASS : II B.Sc. (CT)COURSE NAME : PROGRAMMING WITH VISUAL BASICCOURSE CODE: 17CTU302UNIT: IISEMESTER IIIBATCH (2017-2020)

the error handling procedure is executed. It will show an error in the Txt_Answer label and an error message in the Lbl_ErrorMsg label.

Notice that Exit sub after the division. It is to prevent the program from executing the error_handler code even though the user does not enter zero in the second textbox.Lastly, after the error message appeared, the user will click on the textboxes again. When this occur, the error message will disappear both from the answer label and error message label. This is achieved by using the event procedure GotFocus, as shown in the code.

The Output Window

5. Error Handling Example					
First Number	10	Calculate			
Second Number	0	L			
First Number ÷ Second Number = Error					
You attempt to divide a n	number by zero!Ti	ry again!			

Figure 2.4

Nested Error Handling Procedure

By referring to Example 35.1, we need to consider other types of errors made by the user, such as entering non-numeric inputs like letters. Therefore, we need to write error handling code for this error too. It should be put in the first place as soon as the user input something in the textboxes. And the error handler label *error_handler1* for this error should be put after the *error_handler2* label. This means the second error handling procedure is **nested** within the first error handling procedure. Notice that you have to put an Exit Sub for the second error handling procedure to prevent to execute the first error handling procedure again.

The Code Private Sub CmdCalculate_Click() Dim firstNum, secondNum As Double On Error GoTo error_handler1 firstNum = Txt_FirstNumber.Text secondNum = Txt_SecondNumber.Text On Error GoTo error_handler2 Lbl_Answer.Caption = firstNum / secondNum Exit Sub 'To prevent error handling even the inputs are valid



CLASS: II B.Sc. (CT) **COURSE NAME : PROGRAMMING WITH VISUAL BASIC COURSE CODE: 17CTU302 UNIT: II SEMESTER III BATCH (2017-2020)** error_handler2: Lbl_Answer.Caption = "Error" Lbl ErrorMsg.Visible = True Lbl_ErrorMsg.Caption ="You attempt to divide a number by zero!Try again!" Exit Sub error handler1: Lbl_Answer.Caption = "Error" Lbl ErrorMsg.Visible = True Lbl_ErrorMsg.Caption ="You are not entering a number! Try again!" End Sub

Private Sub Txt_FirstNumber_GotFocus() Lbl_ErrorMsg.Visible = False End Sub

Private Sub Txt_SecondNumber_GotFocus() Lbl_ErrorMsg.Visible = False End Sub

🕄 Error Handling Example					
First Number	a	Calculate			
Second Number	b				
First Number ÷ Second Number = Error					
You are not entering a nu	ımber! Try again!				

Figure 2.5

CREATING MENUS FOR YOUR APPLICATION

The menu bar is the standard feature of most Windows applications. The main purpose of the menus is for easy navigation and control of an application. Some of the most common menu items are File, Edit, View, Tools, Help and more. Each item on the main menu bar also provides a list of options in the form of a pull-down menu. When you create a Visual Basic 6 program, you need not include as many menu items as a full-fledged Windows application. What you need is to include those menu items that can improve the ease of usage by the user. There are two ways to add menus to your application, using the Visual Basic's Application Wizard and or the menu editor.



CLASS : II B.Sc. (CT) COURSE CODE: 17CTU302 SEMESTER III

COURSE NAME : PROGRAMMING WITH VISUAL BASIC UNIT: II BATCH (2017-2020)

Adding Menu Bar Using Visual Basic's Application Wizard

The easiest way to add a menu bar to your application is by using Visual Basic's Application Wizard. This wizard allows the user to insert fully customized standard Windows menu into his or her application. To start using Visual Basic's Application Wizard, click on the Application Wizard icon at the Visual Basic new project dialog box

N	ew Project					×
	Standard EXE	ActiveX EXE	ActiveX DLL	ActiveX Control	* III	OK Cancel <u>H</u> elp
	Addin	Data Project	DHTML Application	IIS Application	•	

Figure 2.6

New Project Window

When you click on the VB Application wizard, the introduction dialog box will appear. As you are not loading any default setting, just click on the Next button







CLASS : II B.Sc. (CT) COURSE CODE: 17CTU302 SEMESTER III

COURSE NAME : PROGRAMMING WITH VISUAL BASIC UNIT: II BATCH (2017-2020)

After clicking the Next button, the interface type dialog box will be displayed. There are three choices of interface available for your project. As we currently not creating a Multiple Document Interface (MDI), we choose Single Document Interface (SDI). You can also type the project name in the textbox below, here I am using MyFirstMenu.

Application Wizard - Inter	face Type
	What type of interface would you like for your application?
Hint! All windows exist at the same level.	 <u>M</u>ultiple Document Interface (MDI) <u>Single Document Interface (SDI)</u> <u>Explorer Style</u>
	<u>W</u> hat name do you want for the application? MyFirstMenu
Help	Cancel < <u>B</u> ack <u>N</u> ext > <u>F</u> inish

Figure 2.8



CLASS : II B.Sc. (CT) COURSE CODE: 17CTU302 SEMESTER III

COURSE NAME : PROGRAMMING WITH VISUAL BASIC UNIT: II

BATCH (2017-2020)

Clicking the Next button wiill bring up a list of menus and submenus that you can add them to your application. Check to select a menu item and uncheck to unselect a menu item. Let say we choose all the menus and click next, then you will get an interface comprises File, Edit, View and Help menus



Figure 2.9

🔄, MyFirstMenu	
<u>F</u> ile <u>E</u> dit <u>V</u> iew <u>H</u> elp	
D 😅 🖬 🎒 👗 🖪 🔀	
Status	19/02/2009 12:07 am

Figure 2.10

When you click on any menu item, a list of drop-down submenu items will be displayed. For example, if you click on the File menu, the list of submenu items such as New, Open, Save, Save As and more will be displayed.



CLASS : II B.Sc. (CT) COURSE CODE: 17CTU302 SEMESTER III

COURSE NAME : PROGRAMMING WITH VISUAL BASIC UNIT: II BATCH (2017-2020)

1. Droject1		
<u>File E</u> dit <u>V</u> iew <u>H</u> el	р	
New Open Close	Ctrl+N	3 <i>I</i> <u>U</u>
Save Save As Save All		
Properties		
Page Setup Print Preview Print		08:28 pm
Send		
Exit		

Figure 2.11

Clicking on any of the dropped down menu item will show the code associated with it, and this is where you can modify the code to suit your programming needs. For example, clicking on the item Open will reveal the following code:

```
😓 Project1 - frmMain (Code)
 mnuFileOpen
    Private Sub mnuFileOpen Click()
        Dim sFile As String
        With dlgCommonDialog
            .DialogTitle = "Open"
            .CancelError = False
            'ToDo: set the flags and attributes of the cor
            .Filter = "All Files (*.*) |*.*"
            .ShowOpen
            If Len(.FileName) = 0 Then
                Exit Sub
            End If
            sFile = .FileName
        End With
        'ToDo: add code to process the opened file
    End Sub
                         Figure 2.12
```


CLASS : II B.Sc. (CT)COURSE NAME : PROGRAMMING WITH VISUAL BASICCOURSE CODE: 17CTU302UNIT: IISEMESTER IIIBATCH (2017-2020)Now, I will show you how to modify the code in order to open a graphic file and display it in

Now, I will show you how to modify the code in order to open a graphic file and display it in an image box. For this program, you have to insert a Image box into the form. Next add the following lines so that the user can open graphic files of different formats.

.Filter =

"Bitmaps(*.BMP)|*.BMP|Metafiles(*.WMF)|*. WMF|Jpeg

Files(*.jpg)|*.jpg|GIF Files(*.gif)|*.gif|Icon

Files(*.ico)|*.ico|All

Files(*.*)|*.*".

Then, you need to load the image into the Image box with the following code:

Image1.Picture = LoadPicture(.FileName)

Also set the Stretch property of the Image box to true so that the image loaded can resize by itself. Please note that each menu item is a special control, so it has a name too. The name for the menu File in this example is mnuFileOpen.

The Code

Private Sub mnuFileOpen_Click() Dim sFile As String With dlgCommonDialog .DialogTitle = "Open" .CancelError = False 'Set the flags and attributes of the common dialog control .Filter = "Bitmaps(*.BMP)|*.BMP|Metafiles(*.WMF)|*. WMF|Jpeg Files(*.jpg)|*.jpg|GIF Files(*.gif)|*.gif|Icon Files(*.ico)|*.ico|All Files(*.*)|*.*" .ShowOpen Image1.Picture = LoadPicture(.FileName) If Len(.FileName) = 0 Then Exit Sub End If sFile = .FileName End With End Sub When you run the program and click on the File menu and then the submenu Open, the following Open dialog box will be displayed, where you can look for graphic files of various formats to load it into the image box.



LASS : II B.Sc. (CT) DURSE CODE: 17CTU3 CMESTER III	COUR 02	SE NAME : P U E	ROGRAM NIT: II SATCH (20	IMING WITH VI 017-2020)	SUAL BASI
🕄 Open					X
🖉 🗸 🖉 🖉 Sample F	victures	-	Sear	rch	Q
🎍 Organize 👻 🔡 Views	👻 📑 Ne	w Folder			•
Favorite Links	Name	Date taken	Tags	Size	»
📰 Desktop		No iter	ms match yo	ur search.	
Recent Places					
🖳 Computer					
Documents					
Pictures					
🚯 Music					
Recently Changed					
🕑 Searches					
📕 Public					
Folders					
File name			_ [Bitmans(* BMP)	
nie name.				Bitmaps(*.BMP)	
			1	Metafiles(*.WMF) lpeg Files(*.ipg)	
.DIALOGITC	re ope	11		GIF Files(*.gif)	
.CancelErr 'ToDo: set	or = rais the flac	se s and attri	butes of	All Files(*.*)	
.Filter =	"All File	s (*.*) *.*	"		
		Figure 2.1	.3	-	

For example, selecting the jpeg file will allow you to choose the images of jpeg format



CLASS : II B.Sc. (CT) COURSE CODE: 17CTU FMFSTFR III	COUI 302	RSE NAME	: PROGRA UNIT: II BATCH	MMING W	VITH VISUAL BAS
C, Open	le la la		DATCH	(2017-2020)	
Sample P	Pictures		• • Searc	h	٩
🔄 Organize 👻 🔛 Views	🔹 📑 New	Folder			0
Favorite Links	Name	Date taken	Tags	Size	>> *
Desktop					
Computer	Autumn Leaves	Creek	Desert Landscape	Dock	E
Pictures Music			and the		
 Recently Changed Searches 	Forest	Forest Flowers	Frangipani Flowers	Garden	
Folders	ge.	The.	4		-
File name:	:		✓ Jr	beg Files(*.jpg)	-
			C	Open	Cancel
		Figure	2 14		

Clicking on the particular picture will load it into the image box



Figure 2.15



CLASS : II B.Sc. (CT) COURSE CODE: 17CTU302 SEMESTER III

COURSE NAME : PROGRAMMING WITH VISUAL BASIC UNIT: II BATCH (2017-2020)

Adding Menu Bar Using Menu Editor

To start adding menu items to your application, open an existing project or start a new project, then click on Tools in the menu bar of the Visual Basic IDE and select Menu Editor. When you click on the Menu Editor, the Menu Editor dialog will appear. In the Menu Editor dialog , key in the first item File in the caption text box. You can use the ampersand (&) sign in front of F so that F will be underlined when it appears in the menu, and F will become the hot key to initiate the action under this item by pressing the Alt key and the letter F. After typing &File in the Caption text box, move to the name textbox to enter the name for this menu item, you can type in mnuFile here. Now, click the Next button and the menu item &File will move into the empty space below

Menu Editor	×
Caption: &File	ОК
Na <u>m</u> e: mnuFile	Cancel
Index: Shortcut: (None)	-
HelpContextID: 0 NegotiatePosition:	0 - None 🔻
Checked 🔽 Enabled 🔽 Visible	<u>W</u> indowList
← → ↑ ↓ <u>N</u> ext <u>I</u> nsert	Dele <u>t</u> e
8File	
1	

Figure 2.16

You can then add in other menu items on the menu bar by following the same procedure



CLASS : II B.Sc. (CT) COURSE NAME : PROGE	RAMMING WITH VISUAL BASIC
COURSE CODE: 17CTU302 UNIT: I	I
SEMESTER III BATCH	H (2017-2020)
Menu Editor	
Caption: SFile OK	
Name: mnuFile Cancel	
Inde <u>x</u> : <u>S</u> hortcut: (None)	
HelpContextID: 0 NegotiatePosition: 0 - None -	
☐ Checked ▼ Enabled ▼ Visible □ WindowList	
← → ↑ ↓ <u>N</u> ext <u>I</u> nsert Dele <u>t</u> e	
&File &Edit	
V&iew Hel&n	

Figure 2.17

When you click Ok, the menu items will be shown on the menu bar of the form.

	Ę	3		F	0	rr	n	1																							-			•))(2	×		
	I	i	le	2		E	d	it		1	Vį	ie	w.	v		ł	le	elį	p																					
				:	:	:	:		:					:	:	:			÷		÷	÷	÷	÷	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	
																					÷																			
			•																																					
			•	•	•	•	•	·	•	•			•	•	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	•	·	·	·	·	
	• •		•	•	·	·	•	·	·	•	•		•	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	
	• •		•	•	·	·	•	·	•	•	•		•	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	
	• •		•	•	•	•	•	•	•	•	•		•	•	•	•	•	•	•	•	·	·	·	·	·	·	·	•	•	•	•	•	•	•	•	•	•	•	•	
	• •		•	•	•	•	•	•	•	•	•		•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	
			:	:	:	:	:	:	:					:	:	:	:	:	÷	:	÷	÷	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	
,																																								
,			•											•																										
			•	•										•	·	·	·	·	·		·	·	·	·	·	·	·	·	·	·	·	·	·	·	•	·		·	·	
	• •		•	•	·	·	·	·	·	•	•		•	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	
	• •		•	•	·	·	•	·	•	•	•		•	•	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	
1	• •		•	•	•	•	•	•	•	•	•		•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	
ľ																																								
ľ																					÷	1	÷	1	1	1														
,																																								
,																																								
,			•	•	•	•	•	•	•	•				•	•	•	•	•	•	•		•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	
					1																								0.0							5.0		1		

Figure 2.18

Now, you may proceed to add the sub menus. In the Menu Editor, click on the Insert button between File and Exit and then click the right arrow key, and the dotted line will appear. This shows the second level of the menu, or the submenu. Now key in the caption and the name. Repeat the same procedure to add other submenu items. Here, we are adding New, Open, Save, Save As and Exit.



VOFILICIED

(tabled use sees a d'OCC AC, 1992) KARPAGAMACADE	MY OF HIGHER EDUCATION
CLASS : II B.Sc. (CT) COURSE NAI	ME : PROGRAMMING WITH VISUAL BASIC
COURSE CODE: 17CTU302	UNIT: II
SEMESTER III	BATCH (2017-2020)
Menu Editor	
Caption: Exit	ОК
Na <u>m</u> e: mnuExit	Cancel
Inde <u>x</u> :	Shortcut: (None)
HelpContextID: 0	NegotiatePosition: 0 - None
□ <u>C</u> hecked	✓ <u>V</u> isible
← → ↑ ↓ <u>N</u> ext	Insert Delete
Reile	
····New	
····Open ····Save	
····Save As	
····Exit	
V&iew	
Hel&p	
E'm	amo 2.10
Fig	ire 2.19
5. Form1	
<u>File Edit View Help</u>	
3	
••••••••••••••••	
Fig	ıre 2.20

Now click the OK button and go back to your form. You can see the dropped down submenus when you click on the item File, as shown.



KARPAGAM ACADEMY OF HIGHER EDUCATION CLASS : II B.Sc. (CT) COURSE NAME : PROGRAMMING WITH VISUAL BASIC COURSE CODE: 17CTU302 UNIT: II BATCH (2017-2020) **SEMESTER III** - • • 5. Form1 <u>File Edit View Help</u> New Open Save Save As Exit Figure 2.21

Finally, you can enter the code by clicking on any of the submenu items.



CLASS : II B.Sc. (CT) COURSE CODE: 17CTU302 SEMESTER III

COURSE NAME : PROGRAMMING WITH VISUAL BASIC UNIT: II BATCH (2017-2020)

POSSIBLE QUESTIONS

Part – B (2 Marks)

- 1. Define Variable.
- 2. Define Control Array.
- 3. Define Data type.
- 4. Define Constants.
- 5. Write the syntax of Val function with example.

Part – C (6 Marks)

- 1. Explain operators with example for each.
- 2. Explain (i) Multidimensional Array (ii) Dynamic Array.
- 3. Explain (i) Multidimensional Array (ii) Dynamic Array.
- 4. Explain how to create menu with sample program.
- 5. Explain operators with example for each.
- 6. Explain control array with calculator program.
- 7. What is Data type? Explain Different types of Data types in VB.
- 8. Explain Error handling in VB 6.0
- 9. What is Variables? Explain rules for declaring variables in VB.
- 10. Discuss about formatting data with example.

(Deemed University)

(Established Under Section 3 of UGC Act 1956)

Coimbatore - 641 021

(For the candidates admitted in 2017 onwards)

CLASS : II B.Sc CT

SUBJECT : Programming with visual basic

	0111-2				
Questions	opt1	opt2	opt3	opt4	Answer
data types etc,	constructing	building	fundamental	specified	building
Code in VB is stored in the form of	modules	subroutine	variables	standards	modules
There are kinds of modules	5	2	3	4	3
A simple application may contain a single form and the code resides in the itself	form	module	form module	none	form module
programming in VB	standard	class	form	none	class
is the extension for class module	.frm	.CLS	.std	none	.CLS
may include constant,type,variable and DLL procedure declarations.	declaration	procedure	variable	statement	declaration
type.	const	static	variant	none	variant
is the value range of byte	0 to 255	1 to 255	0 to 266	1 to 266	0 to 255
is the value range of integer	-32767 to 32768	-32768 to 3	32767 to -32768	32768 to -32767	-32768 to 32767
are used for storing values temporarily.	character	constant	variable	module	variable

UNIT-2

From the following which character is allowed while		1	I	I	
declaring variables	%	@	_	\$	_
There are ways for declaring variables	1	3	2	4	3
statement checks in the module for usege					
for any undeclared variables.	option explicit	option impl	int count	none	option explicit
variable is one that is declared inside a					
procedure	global	local	static	scope	local
only as long as the procedure is being executed	public	private	dim	double	dim
keyword is placed at the beginning of	public	private	dim	static	static
ne module-level variable is available to all the	nublic	nrivate	a and b	a or b	a or b
	public	private			
components called	procedures	programs	sub	event	procedures
by default	public	private	static	global	private
window is opened for the module			4	с	
runctions are especially useful for taking one of	code	add	туре	SUD	code
more pieces of data called	modules	arguments	procedures	programs	arguments
of program execution	control	property	while	do-while	control
statements, in order to execute one block	if then Else	if	then	if then	if then Else
execuion of one or ore statements	if then else	if	then	if then	if then
is an alternative to IfThenElse.	selectcase	casesele	select	none	selectcase
Selectcase structure evaluates an expression					
at the top of the structure	twice	once	thrice	none	once
The statement first executex the					

statemetn and then test the condition after each execution	dowhile	whiledo	selectcase	while	dowhile
until the condition is satisfied	do…loop	doloop ur	do whileloop	none	doloop until
do…loop until is loop	finite	infinite	long	small	infinite
function retrives only date	fornext	nextfor	exit for	exit do	fornext
statement	fornext	nextfor	exit for	for exit	fornext
statement	exit for	for exit	exit do	do exit	exit do
referred using	arrays	modules	sub-routines	none	arrays
The individual element of an array are identified using	modules	index	base	none	index
Fixed size array always	remains same	changes	. increased	decreased	remains same
Size of dynamic array is changed at time	execution	run	break	stop	run
a single variable to hold several related information	user defined	user like	user string	none	user defined
constant.	constant	const	consta	constan	const
The function retrieves the date and time	Now	Date	Time	none	Now
function retrives only time	Now	Date	Time	none	Time
function retrives only date	Now	Date	Time	none	Date
dates in terms of years, months or dates.	format	diffdate	datediff	relational	datediff
converts it to a string in format	specified	relational	logical	comparison	specified
returns the variant (string) containing a specified number of characters from a string	left()	right()	mid()	instr()	mid()
The function returns the length of the string	strcmp()	strrev()	len()	format()	len()

is a group of controls that share the	control arraya	orrovo	format arraya	2020	control orrova
mere are ways to create a control	control arrays	allays	ionnal anays	none	control arrays
array at design time	3	2	1	5	3
	TRUE	FALSE	0	INDEX%	index%
statement.	load	unload	format	unformat	load
array	load	unload	format	unformat	unload
executed as a unit	property	procedure	project	browser	procedure
A procedure name is always defined at level	routine	subroutine	procedure	project	procedure
of the code window and the debug window that displaya the procedures recognized for the object displayed in the object box.	project file	property pa	procedure box	none	procedure box
small sections of coding in applications	project	property	procedure	none	procedure
for other programs with slight modifications	building blocks	buildings	construction	module	building blocks
contains the control's actual name, an under score	sub	event	general	function	event
procedure dialog box from the tools	add	sub	mul	div	add
Boolean has function	cbool	cbol	cboolean	cboolea	cbool
Decimal has function	cdecimal	cdeci	cdemal	cdec	cdec



CLASS: II B.Sc.CT

COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: III

BATCH-2017-2020

<u>UNIT III</u>

Writing Code: Control flow statements – If – Then – If-then-else – Nested control statements – Select case – Loop statements – Do-loop – For-Next – While Wend – Exit statement . Displaying message in Message box, testing whether input is valid or not. Collections – procedures – Subroutines – Functions – Calling procedures – Object Browser – Creating classes and Objects – I/O statements

CONTROL FLOW 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.

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 of the If...Then selection

If <condition> Then statement End If

e.g.: If average>75 Then txtGrade.Text = "A" End If

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 of the If...Then...Else selection



CLASS: II B.Sc.CT	COURSE NAME: PRO	GRAMMING WITH VISUAL BASIC	
COURSE CODE: 17CTU302	UNIT: III	BATCH-2017-2020	
If <condition> Then</condition>			
statements			
Else			
statements			
End If			
e.g.: If average>50 Then			
txtGrade.Text = "Pass"			
Else			

txtGrade.Text = "Fail"

End If

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

You can use Nested If either of the methods as shown above

Method 1

If < condition 1 > Then statements ElseIf< condition 2 > Then statements ElseIf< condition 3 > Then statements Else Statements End If



CLASS: II B.Sc.CT	COURSE NAME: PRO	GRAMMING WITH VISUAL BASIC	
COURSE CODE: 17CTU302	UNIT: III	BATCH-2017-2020	
Method 2			
If < condition 1 > Then			
statements			
Else			
If < condition 2 > Then			
statements			
Else			
If < condition 3 > Then			
statements			
Else			
Statements			
End If			
End If			
EndIf			

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

If average > 75 Then txtGrade.Text = "A" ElseIf average > 65 Then txtGrade.Text = "B" ElseIf average > 55 Then txtGrade.text = "C" ElseIf average > 45 Then txtGrade.Text = "S" Else txtGrade.Text = "F" End If

Select...Case selection structure



CLASS: II B.Sc.CT COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: III BATCH-2017-2020

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 of the Select...Case selection structure

Select Case Index Case 0 Statements Case 1 Statements End Select

e.g.: 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



CLASS: II B.Sc.CT COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: III BATCH-2017-2020

MsgBox "Invalid average marks" End Select

LOOPING STATEMENTS

A repetition structure allows the programmer to that an action is to be repeated until given condition is true.

Do While... Loop Statement

The **Do While...Loop** is used to execute statements until a certain condition is met. The following Do Loop counts from 1 to 100.

```
Dim number As Integer
number = 1
Do While number <= 100
number = number + 1
Loop
```

A variable number is initialized to 1 and then the Do While Loop starts. First, the condition is tested; if condition is True, then the statements are executed. When it gets to the Loop it goes back to the Do and tests condition again. If condition is False on the first pass, the statements are never executed.

While... Wend Statement

A While...Wend statement behaves like the Do While...Loop statement. The following While...Wend counts from 1 to 100

Dim number As Integer



CLASS: II B.Sc.CT COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302	UNIT: III	BATCH-2017-2020	

number = 1 While number <=100 number = number + 1 Wend

Do...Loop While Statement

The **Do...Loop** While statement first executes the statements and then test the condition after each execution. The following program block illustrates the structure:

Dim number As Long number = 0 Do number = number + 1 Loop While number < 201

The programs 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.

Do Until...Loop Statement

Unlike the **Do While...Loop** and **While...Wend** repetition structures, 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.

An example for **Do Until...Loop** statement. The coding is typed inside the click event of the command button

Dim number As Long number=0 Do Until number > 1000



CLASS: II B.Sc.CT COURSE NAME: PROGRAMMING WITH VISUAL BASIC

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.

The For...Next Loop

The **For...Next** Loop is another way to make loops in Visual Basic. **For...Next** repetition structure handles all the details of counter-controlled repetition. The following loop counts the numbers from 1 to 100:

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

In order to count the numbers from 1 yo 50 in steps of 2, the following loop can be used

For x = 1 To 50 Step 2 Print x Next

The following loop counts numbers as 1, 3, 5, 7..etc

The above coding will display numbers vertically on the form. In order to display numbers horizontally the following method can be used.

For x = 1 To 50 Print x & Space\$ (2); Next



CLASS: II B.Sc.CT COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: III BATCH-2017-2020

To increase the space between the numbers increase the value inside the brackets after the & Space\$.

Following example is a **For...Next** repetition structure which is with the If condition used.

Dim number As Integer For number = 1 To 10 If number = 4 Then Print "This is number 4" Else Print number End If Next

In the output instead of number 4 you will get the "This is number 4".

EXIT STATEMENT

A **For...Next** loop condition can be terminated by an **Exit For** statement. Consider the following statement block.

Dim x As Integer For x = 1 To 10 Print x If x = 5 Then Print "The program exited at x=5" Exit For End If Next



The preceding code increments the value of x by 1 until it reaches the condition x = 5. The **Exit For** statement is executed and it terminates the **For...Next** loop. The Following statement block containing **Do...While** loop is terminated using **Exit Do**statement.

Dim x As Integer Do While x < 10Print x x = x + 1If x = 5 Then Print "The program is exited at x=5" Exit Do End If Loop

With...End With statement

When properties are set for objects or methods are called, a lot of coding is included that acts on the same object. It is easier to read the code by implementing the **With...End With**statement. Multiple properties can be set and multiple methods can be called by using the **With...End With** statement. The code is executed more quickly and efficiently as the object is evaluated only once. The concept can be clearly understood with following example.

With Text1 .Font.Size = 14 .Font.Bold = True .ForeColor = vbRed .Height = 230 .Text = "Hello World" End With



In the above coding, the object Text1, which is a text box is evaluated only once instead of every associated property or method. This makes the coding simpler and efficient.

MESSAGE BOXES

Message boxes used to display the information. Message boxes should be used for short messages or to give transient feedback. It can hold 1024 characters. Message boxes can be used in either two ways, depending on your needs. You can use for display information and also for decision making from the user. In either case the MsgBox Function is used by the following syntax:

MsgBox(prompt[, buttons] [, title] [, helpfile, context])

The MsgBox function syntax has these parts:

Part	Description
prompt	Required. String expression displayed as the message in the dialog box.
buttons	Optional. Numeric expression that is the sum of values specifying the number and type of buttons to display, the icon style to use, the identity of the default button, and the modality of the message box. If omitted, the default value for <i>buttons</i> is 0 (which causes only an OK button to be displayed with no icon).
title	Optional. String expression displayed in the title bar of the dialog box. If you omit <i>title</i> , the application name is placed in the title bar.
<i>helpfile</i> and <i>context</i>	Both optional. These arguments are only applicable when a Help file has been set up to work with the application.



CLASS: II B.Sc.CT

COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: III BATCH-2017-2020

Eg.1

C Project1 - Form1 (Form)				
Command1		<u></u> 	iput form	
Project1 - Form1 (Code)	lick			
Private Sub Command1 MsgBox "Welcome to V: End Sub	Click() sual Basic			
code window				

Figure 3.1 Input Form

<u>_</u>	🖻 Form1	
	Comma	and1Click this button
		Project1
		OK

Figure3.1 Output Form

In the running screen click on the command1 button you get an Msgbox named "Welcome to Visual Basic" followed by **OK** button. Now, Click on the **OK** button you will be back to Visual Basic screen.



CLASS: II B.Sc.CT

KARPAGAM ACADEMY OF HIGHER EDUCATION

COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: III BATC

BATCH-2017-2020

Eg.2

MsgBox "The Last Name field must not be blank.", _ VbExclamation, _ "Last Name"

 \rightarrow causes the following box to be displayed:



Figure 3.3

The buttons argument

The Buttons displayed in a message here

Button Layout	Value	Short Description
vbOKonly	0	Displays the OK button.
vbOKCancel	1	Displays the ok and cancel button.
vbAbortRetryIgnore	2	Displays the Abort , Retry , Ignore
vbYesNoCancel	3	Displays Yes, No and Cancel button
vbYesNo	4	Displays the Yes / No button
vbRetryCancel	5	Displays the retry and Cancel buttons.

The Icons displayed in the message box are here

Icon on message	Value	Short Description
vbCritical	16	Displays critical message icon
vbQuestion	32	Displays question icon



CLASS: II B.Sc.CT

COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302

UNIT: III

BATCH-2017-2020

vbExclamation	48	Displays exclamation icon
vbInformation	64	Displays information icon

The Default button displayed in a message form

Default Button	Value	Short Description
vbDefaultButton1	0	Button 1 is default
vbDefaultButton2	256	Button 2 is default
vbDefaultButton3	512	Button 3 is default

Msgbox Return Value

Return Value	Value	Short Description
vbOk	1	The User Clicked OK
vbCancel	2	The User Clicked Cancel
vbAbort	3	The User Clicked Abort
vbRetry	4	The User Clicked Retry
vbIgnore	5	The User Clicked Ignore
VbYes	6	The User Clicked Yes
VbNo	7	The User Clicked No

COLLECTIONS IN VB

Collection class provides an array-like container more flexible than an array in some ways and less flexible in other ways. More flexible in some ways than Collection is Dictionary class.

Creating a new collection:

Dim Cats As Collection

Prepared by Dr.S.Kowsalya, Asst Prof, Department of CS, CA & IT, KAHE



CLASS: II B.Sc.CT COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: III BATCH-2017-2020

Set Cats = **New** Collection

Dimensioning and creating a new collection in one line:

Dim Cats As New Collection

Adding an item:

Cats.Add "Item" Cats.Add "Item", "Key"

Accessing an item at an index, in a read-only way:

Cats (3) 'The third member of the collection Cats.Item(3) 'An alternative Cats.Item("Key 3") 'Works if an item has a key associated

Overwriting a item at an index:

NewValue = MyCollection(i) + 1 MyCollection.Add NewValue, Before:=i MyCollection.Remove Index:=i + 1

Removing an item:

Collection.Remove Index Collection.Remove HashKey

The size of a collection:



CLASS: II B.Sc.CT COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: III BATCH-2017-2020

Cats.Count

Iterating over a collection, read-only:

For Each Cat In Cats Rem Do something on Cat Next

Iterating over a collection, read-write:

'Fill the collection
Set MyCollection = New Collection
For i = 1 To 10
MyCollection.Add i
Next
'Increment each item by 1
For i = 1 To MyCollection.Count
NewValue = MyCollection(i) + 1
MyCollection.Add NewValue, Before:=i
MyCollection.Remove Index:=i + 1
Next

Testing the emptiness of a collection:

If Cats.Count=0 Then '... End If

Testing the presence of an element in a collection:



CLASS: II B.Sc.CT COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302UNIT: IIIBATCH-2017-2020

MatchFound = False For Each Member In MyCollection If Member = "SoughtValue" Then MatchFound = True Exit For End If Next

Appending one collection to another:

For Each Member In AppendedCollection ExtendedCollection.Add Member Next

Converting a collection to an array:

Dim MyArray

ReDim MyArray(MyCollection.Count - 1)

Index = 0

For Each Member In MyCollection

```
MyArray(Index) = Member
```

Index = Index + 1

Next

Using a collection as a queue:

```
Set Queue = New Collection
For i = 1 To 10
Queue.Add i
Next
```



CLASS: II B.Sc.CT COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: III BATCH-2017-2020

For i = 1 To 5

Queue.Remove 1 '*Remove from the queue*

Next

Using a collection as a stack:

Set Stack = New Collection
For i = 1 To 10
Stack.Add i
Next
For i = 1 To 5
Stack.Remove Stack.Count 'Remove from the stack
Next

Using a collection as a set:

' Using the key of a collection item will do the trick.
' The key has to be a string, hence "Str(i)" below.
' Note that Str(1) is " 1", with a leading space.
Set NumberSet = New Collection
For i = 1 To 10
NumberSet.Add i, Str(i) 'Value, Key
Next
For i = 5 To 15
'Add while catching errors resulting from existence of the key
On Error Resume Next
NumberSet.Add i, Str(i) 'Value, Key
On Error GoTo 0
'End If



CLASS: II B.Sc.CT COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302UNIT: IIIBATCH-2017-2020

Next	
For $i = 14$ To 16	
'Remove	
On Error Resume Next 'Catch error if element not present	
NumberSet.Remove Str(i)	
On Error GoTo 0	
Next	
'Membership test	
On Error Resume Next	
NumberSet.Item(Str(50))	
IsMember=Err.Number=0	
On Error GoTo 0	
,	
Set NumberSet2 = New Collection	
For i = 10 To 25	
NumberSet2.Add i, Str(i)	
Next	
'Union	
Set SetUnion = New Collection	
For Each Item In NumberSet	
SetUnion.Add Item, Str(Item) 'Value, Key	
Next	
For Each Item In NumberSet2	
On Error Resume Next	
SetUnion.Add Item, Str(Item) 'Value, Key	
On Error GoTo 0	
Next	
'Intersection	
Set SetIntersection = New Collection	



CLASS: II B.Sc.CT COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: III BATCH-2017-2020

For Each Item In NumberSet **On Error Resume Next** Dummy = NumberSet2(Str(Item)) If Err.Number = 0 Then SetIntersection.Add Item, Str(Item) 'Value, Key End If On Error GoTo 0 Next 'Set difference **Set** SetDifference = **New** Collection For Each Item In NumberSet **On Error Resume Next** Dummy = NumberSet2(Str(Item)) If Err.Number <> 0 Then SetDifference.Add Item, Str(Item) 'Value, Key End If On Error GoTo 0 Next

PROCEDURES IN VB

Visual Basic offers different types of procedures to execute small sections of coding in applications. The various procedures are elucidated in details in this section. Visual Basic programs can be broken into smaller logical components called Procedures. Procedures are useful for condensing repeated operations such as the frequently used calculations, text and control manipulation etc. The benefits of using procedures in programming are:

It is easier to debug a program a program with procedures, which breaks a program into discrete logical limits.



CLASS: II B.Sc.CT COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: III BATCH-2017-2020

Procedures used in one program can act as building blocks for other programs with slight modifications.

A Procedure can be Sub, Function or Property Procedure.

Sub Procedures

A sub procedure can be placed in standard, class and form modules. Each time the procedure is called, the statements between Sub and End Sub are executed. The syntax for a sub procedure is as follows:

[Private | Public] [Static] Sub Procedurename [(arglist)] [statements] End Sub

arglist is a list of argument names separated by commas. Each argument acts like a variable in the procedure. There are two types of Sub Procedures namely general procedures and event procedures.

Event Procedures

An event procedure is a procedure block that contains the control's actual name, an underscore(_), and the event name. The following syntax represents the event procedure for a Form_Load event.

Private Sub Form_Load()statement block.. End Sub

Event Procedures acquire the declarations as Private by default.

General Procedures



A general procedure is declared when several event procedures perform the same actions. It is a good programming practice to write common statements in a separate procedure (general procedure) and then call them in the event procedure.

In order to add General procedure:

- The Code window is opened for the module to which the procedure is to be added.
- The *Add Procedure* option is chosen from the Tools menu, which opens an Add

Procedure dialog box as shown in the figure given below.

- The name of the procedure is typed in the Name textbox
- Under *Type*, *Sub* is selected to create a Sub procedure, *Function* to create a Function procedure or *Property* to create a Property procedure.
- Under *Scope*, *Public* is selected to create a procedure that can be invoked outside the module, or Private to create a procedure that can be invoked only from within the module.

Add Procedure		
<u>N</u> ame:		ок
Type © Sub © Eunction	C <u>P</u> roperty C <u>E</u> vent	Cancel
Scope Public All Local variab	○ Pri <u>v</u> ate oles as Statics	

Figure 3.4

We can also create a new procedure in the current module by typing Sub ProcedureName, Function ProcedureName, or Property ProcedureName in the Code window. A Function procedure returns a value and a Sub Procedure does not return a value.

Function Procedures



Functions are like sub procedures, except they return a value to the calling procedure. They are especially useful for taking one or more pieces of data, called *arguments* and performing some tasks with them. Then the functions returns a value that indicates the results of the tasks complete within the function.

The following function procedure calculates the third side or hypotenuse of a right triangle, where A and B are the other two sides. It takes two arguments A and B (of data type Double) and finally returns the results.

Function Hypotenuse (A As Double, B As Double) As Double Hypotenuse = sqr $(A^2 + B^2)$ End Function

The above function procedure is written in the general declarations section of the Code window. A function can also be written by selecting the*Add Procedure* dialog box from the Tools menu and by choosing the required scope and type.

Property Procedures

A property procedure is used to create and manipulate custom properties. It is used to create read only properties for Forms, Standard modules and Class modules.Visual Basic provides three kind of property procedures-Property Let procedure that sets the value of a property, Property Get procedure that returns the value of a property, and Property Set procedure that sets the references to an object.

SUBROUTINES

Subroutines can be thought of as **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



CLASS: II B.Sc.CT COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: III BATCH-2017-2020

automatically called when a form loads. This is where you have been inserting your test code before. What you were actually doing was adding code to this subroutine.

Creating A Subroutine

Creating a subroutine involves two lines of code. Luckily though, the Visual Basic code editor is smart, and will insert the second line for you! 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. Don't worry about these yet, they will be covered later in the lesson.

Sub TestSub() End Sub

After you enter the first line and press Enter, the second line will automatically be added for you. These lines represent the start and end of the subroutine. Any code inserted between these lines will be executed when the subroutine is called. A subroutine can be called in one of two ways: using the Call keyword, or by simply stating its name.

```
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

You can also use the Call keyword, as mentioned above:



CLASS: II B.Sc.CT COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302UNIT: IIIBATCH-2017-2020

Sub TestSub()

MsgBox "Code in TestSub()"

End Sub

Private Sub Form_Load()

MsgBox "Code in Form_Load()" *'This line is functionally equal as the line in the previous example* Call TestSub MsgBox "Back in Form_Load()" End Sub

Comments

The previous listing introduced a new syntax element: the comment. A comment is a line that is **NOT** source code. Instead, it is there as guidance to the person reading the code. When programmers write programs, they usually (or at least they should) add comments describing large sections of code that could potentially be confusing. Comments begin with an apostrophe (" ' ") and end with a new line. Everything that you type after the apostrophe is ignored by Visual Basic, which is the entire point of comments. You will see comments in all future listings describing new concepts and syntax.

Subroutine Scope

The declaration of a subroutine has an optional keyword that can be used before "Sub". You may enter a keyword representing scope here. "Private" and "Public" are examples of scopes. You may have noticed that the Form_Load subroutine has the word "Private" before "Sub". This declares that this subroutine has a Private scope. If you don't specify the scope of a subroutine, Visual Basic will use a default scope. You should never rely on default scope, however. Get in the habit of always explicitly declaring the scope of your subroutines.


COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302

CLASS: II B.Sc.CT

BATCH-2017-2020

What Is Scope?

In the context of subroutines, scope represents where in your program the subroutine can be called from. Subroutines with a Private scope can only be called from the source file from where they were defined. Subroutines with a Public scope can be called from anywhere in your program. For example, a subroutine with a Private scope in a form can not be called from another form, whereas it can if it has a Public scope.

UNIT: III

Why Use Scope?

Why would you ever limit where you can call a subroutine from? Why wouldn't you make all subroutines Public so that there are no limitations on your program? Subroutine Scope is one of many tools that programmers can use to find bugs. If you know that your subroutine should never be called from a different source file, you should declare the subroutine to be Private. This will prevent you from making a silly mistake and calling this subroutine where it shouldn't be called. This will prevent mistakes that could be extremely damaging and hard to detect. Instead, your program will crash with an error rather than executing the subroutine with unpredictable results.

Parameters

Parameters, also called Arguments, are variables that can be "passed into" a subroutine. A subroutine with parameters looks like this:

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. The calling subroutine doesn't have to use numeric constants, however. It can use variables as well:

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

MsgBox x + y

End Sub

```
Private Sub Form_Load()
  Dim a As Integer
  Dim b As Integer
  a = 5
  b = 2
  DisplayAdd a, b
End Sub
```

This code has identical results. Note that DisplayAdd cannot access a and b. As far as DisplayAdd is concerned, a and b are represented as x and y. Attempting to access a or b from DisplayAdd would result in an error.



COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: III

BATCH-2017-2020

ByRef and ByVal

CLASS: II B.Sc.CT

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 = xx = yy =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
```



CLASS: II B.Sc.CT COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302UNIT: IIIBATCH-2017-2020

'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()



CLASS: II B.Sc.CT COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302UNIT: IIIBATCH-2017-2020

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

Functions Or Subroutines?

The best way to determine which is better for your application is by asking yourself a simple question. Will you need to return a value? If so, use Functions. If you don't, use subroutines.

Subroutines

Subroutines are another word for methods, and they're code blocks that will be executed by your application as you call them elsewhere in your applications. To call a subroutine, just call it by the subroutine's name (for instance, if you created your own subroutines).

The Public, Private and Protected keywords are incase you create multiple classes (explained below) and you want to choose whether other classes have access to use the respective methods (or subroutines) you have created. You can search online for the exact definition of these three keywords.

Say you have a subroutine that does some task of some kind, you call it and the code within that subroutine gets executed. However, the thing is, what if you wanted to make sure some code in the subroutine did it's job? What about using the Returnkeyword and return either true or false depending on whether the portions of the code block was executed? You may want to do this at some point in your Visual Basic development. Either way, subroutines cannot return values – they're there purely to execute code. If you wanted to do this, you'd



CLASS: II B.Sc.CT COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302

UNIT: III BATCH-2017-2020

need to use a Function in Visual Basic. It's like a Subroutine, except at some point you have to return a value of some form, like: Return TRUE etc.

Unlike Subroutines, with Functions you have to specify the data type of the Function (in other words, what sort of data *type* you're returning. Here's a basic example:

Public Function returnTrue() As String

Return "Executed"

End Function

The brackets are required there incase you may want to take in arguments as part of when some other part of the application call's the function. For example:

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)

Handles MyBase.Load

returnTrue("Hello world")

End Sub

Public Function returnTrue(ByVal n) As String

Return n

End Function

Ignore ByVal for now, you can learn that later – it's just telling Visual Basic how to handle the argument (the variable in the brackets). You can always search online about the ByVal keyword. Now, all this does is returns whatever you enter when calling the Function. If you forget to add ByVal when creating a Function or Subroutine, Visual Studio or Visual Basic Express will do it for you – incase you forget. You can also specify arguments (brackets) for Subroutines too. The only difference is Subroutines cannot return values, hence why Functions require an As Type keyword after them.

CREATING CLASSES AND OBJECT

An *object* is an encapsulation of *data*, and *methods* (procedures) that act on the data. The details of the data structures used and the implementation of the methods are hidden inside the object. All the programmer needs to know is what tasks the object can perform, and the parameters required by these tasks.



Control objects are predefined objects that can be created using the Visual Basic toolbox, such as text boxes, command buttons and other controls. No matter how many text boxes you create on a form, they all have the same properties and methods. Essentially, a *class* is a template from which an object is created, and specifies the properties and methods that will be common to all objects that are created from it. Each text box is therefore an*instance* of the class *TextBox*, for which you can set the properties or invoke the methods.

The value of a property is manipulated by two procedures, one to set the value (*Let*) and the other to retrieve the value (*Get*). These procedures are sometimes referred to as*accessor methods*.

Code objects must be created by the programmer, and are instances of user-defined types that are defined in a separate *class module*. Like control objects, they have properties and methods and can respond to events. The set of properties, methods, and events for a class is called the *class interface*. Two (or more) different classes may have the same interface, and carry out identical tasks, even though they carry out the task differently. The occurrence of two or more classes that have behaviors with the same name and functionality but different implementations is called *polymorphism*.

Events are declared in the general declarations section of class module with a statement of the form:

Public Event UserDefinedEvent (arg1, arg2, ...)

and triggered with a *RaiseEvent* statement. The statement that declares the object in the form's code must include the keyword *WithEvents* in order for the events coming from the object to be processed.

When Visual Basic is an *object-oriented* programming language. Applications communicate with objects through the properties and methods they *expose*, and the events they *raise*. When you program with Visual Basic, you are programming objects, such as the controls you place on a form, by manipulating their properties and calling their methods. The action takes place within the object's *event procedures*, which are executed in response to the occurrence of some event. Events are said to be raised (in other words they are considered to have occurred) when certain conditions are met.



CLASS: II B.Sc.CT

COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302

UNIT: III

BATCH-2017-2020





KARPAGAM KARPAGAM KARPAGAM KARPAGAM	RPAGAM ACADEMY OF	FHIGHER EDUCATION
CLASS: II B.Sc.CT	COURSE NAME: PROGE	RAMMING WITH VISUAL BASIC
COURSE CODE: 17CTU302	UNIT: III	BATCH-2017-2020

The relationship between an object and the application

EXAMPLE 1

Open a new Visual Basic project, save it to a new folder, and create a form similar to the one illustrated below.

	Students			×
		Create Stu	dent Record	
	Student Name			
	Student Number		Student Details	
		Enter Details		
		Display Details		
		Exit		
		Figu	ire 3.6	
The "frm	Students" for	m		
Propertie	s for "frmStuc	lents" form		
Control	Name	Caption	Additional Properties	
Form	frmStudent	s Students		
Label	lblTitle	Create Student Record	Alignment = 2 - Center	
Label	lblName	Name	Alignment = 1 - Right Justify	
TextBox	txtName			
Label	lblNumber	Number	Alignment = 1 - Right Justify	
TextBox	txtNumber			



CLASS: II B.Sc.CT COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: III BATCH-2017-2020

Button cmdEnter Enter Details

Button cmdDisplay Display Details

PicturePox picStudent

Button cmdQuit Exit

CREATING THE STUDENT CLASS

It is a convention to precede the name of a class with the letter 'C'. We will therefore use the

name *CStudent* . Use the following steps to create the *CStudent* class:

- 1. From the **Project** menu, select **Add Class Module** .
- 2. Double-click on **Class Module** in the **Add Class Module** dialog box.
- 3. If the **Properties** window is not visible, press **F4** to display it. (Note that the class has the default name *Class1*.)
- 4. Change the setting of the **Name** property to *CStudent*.
- 5. Type the following lines into the code module: Private mName As String

Private mNumber As String

These are the *member variables* used to hold data.

The word *Private* means they cannot be accessed directly.

- 6. From the **Tools** menu, click on **Add Procedure** .
- Type *Name* into the Name text box, select the Property radio button in the Type box, and click on OK. The following code will appear in the class module window: Public Property Get Name() As String

End Property

Public Property Let Name(ByVal vNewValue As Variant)

End Property

Change the word *Variant* in both procedures to *String*, and the word*vNewValue* in the second procedure to *vName*, and add code to the procedures as shown below: Public Property Get Name() As String



CLASS: II B.Sc.CT COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302UNIT: IIIBATCH-2017-2020

Name = mName

End Property

Public Property Let Name(ByVal vName As Variant)

mName = vName

End Property

The first procedure retrieves the value of the variable *mName*

'The second procedure assigns a value to mName

9. Create the procedures that will be used to retrieve and assign values to the variable *mNumber* as follows:Public Property Get Number() As String

Number = mNumber

End Property

Public Property Let Number(ByVal vNumber As Variant)

mNumber = vNumber

End Property

10. From the **File** menu, click on **Save CStudent As** and save the class module with the name *CStudents.cls*. The *.cls* extension is used for files holding class modules.

USING THE STUDENT CLASS

We can now write a program that creates an object that is an instance of the *CStudent* class, and uses the *Property Let* and *Property Get* procedures to assign values to and retrieve values from the member variables. Enter the following code in the form's code window:

Private Student As CStudent

Private Sub Form_Load()

Set Student = New CStudent

End Sub



CLASS: II B.Sc.CT COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302

UNIT: III

BATCH-2017-2020

Private Sub cmdEnter_Click()

Student.Name = txtName

Student.Number = txtNumber

txtName.Text = ""

txtNumber.Text = ""

End Sub

Private Sub cmdDisplay_Click()

picStudent.Cls

picStudent.Print "Student Name:"

picStudent.Print Student.Name

picStudent.Print

picStudent.Print "Student Number:"

picStudent.Print Student.Number

End Sub

Private Sub cmdQuit_Click()



Run the program and enter a student name and number, press the *Enter Details* button to send the data to the object, and press the *Display Details* button to retrieve the details from the object and display the student's name and number.

THE INITIALIZE EVENT PROCEDURE

The **Object** drop-down combo box in a class module window displays two items, *General* and *Class* (see below).



CLASS: II B.Sc.CT

COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302

UNIT: III

BATCH-2017-2020

1	Students - CStudent (Code)	
(0	General)	•
[(G	General)	
C	ASS FILVACE MUMMBEL AS JULING	-
	Public Property Get Name() As String	
	Name = mName	
	End Property	
	Public Property Let Name(ByVal vName As String)	
	mName = vName	
	End Property	
	Public Property Get Number() As String	
	Number = mNumber	
	End Property	
	Public Property Let Number(ByVal vNumber As String) mNumber = vNumber	
	End Property	
		_
E		► //.

Figure 3.7

The class module window

When you click on *Class*, the following template appears:

Private Sub Class_Initialize()

End Sub

This procedure is automatically invoked when an object is created from the class, and can be used to set default values for member variables, or to create other objects associated with this object. The counterpart to *Initialize* is the *Terminate* event procedure which has a similar format, and is automatically invoked when all references to the object are set to*Nothing*, or when the object falls out of scope (an object falls out of scope, remember, when the procedure that created it is exited). This procedure can be used to set any objects you may have created using the class module to *Nothing*.

CLASS RELATIONSHIPS

Three relationships can exist between classes:



CLASS: II B.Sc.CT COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: III BATCH-2017-2020

- *Use* one class *uses* another class if it manipulates or creates objects of that class. In general, class A uses class B if an object of class B is sent a message by a property or method of class A, or a method or property of class A returns, receives, or creates objects of class B.
- *Containment* class A *contains* class B when a member variable of class A has class B as its type.
- *Inheritance* this is the process by which one class (the *child* class), inherits the properties, methods, and events of another class (the *parent* class). Note that Visual Basic does not support the strict academic definition of inheritance.

I/O STATEMENTS IN VB

Visual Basic provides some excellent ways for input and output operations. Input can be taken using TextBox, InputBox, and output can be shown with the Print method, TextBox, Label, PictureBox and MsgBox.

Input and output using TextBox

The TextBox Control provides an efficient way for both input and operations. The following sample program shows it.

Example:

```
Private Sub Command1_Click()

Dim a As Integer

a = Val(Text1.Text)

a = a + 1

Text2.Text = a

End Sub
```

In the above program, Input is taken using Text1 and output is shown in Text2, where output is the incremented value of the input.

• Download sample program: <u>Sum Calculator</u>

The Sum Calculator program is explained here: <u>Some simple VB6 examples.</u>



CLASS: II B.Sc.CT COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: III BATCH-2017-2020

InputBox

InputBox is a function that prompts for user-input. InputBox shows a dialog box that inputs value from the user.

Syntax:

a=InputBox(promt, [Title], [Default], [xpos], [ypos])

where 'a' is a variable to which the value will be assigned. The texts inside the InputBox are optional except the "prompt" text. "prompt" text is the prompt message. "title" is the title of the message box window. "Default" is the default value given by the programmer. 'xpos' and 'ypos' are the geometric positions with respect to x and y axis respectively.

Note: Parameters in brackets are always optional. Do not write the brackets in your program.

Example: Private Sub cmdTakeInput_Click() Dim n As Integer n = InputBox("Enter the value of n : ") Print n End Sub

The above program prints the value of n taken from the InputBox function.

Example: InputBox with the title text and default value.

```
Private Sub cmdTakeInput_Click()

Dim n As Integer

n = InputBox("Enter the value of n : ", "Input", 5)

Print n

End Sub
```



CLASS: II B.Sc.CT COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: III BATCH-2017-2020

The InputBox dialog appears with the title "Input" and the highlighted default value in the provided text field is 5.

MsgBox

The MsgBox function shows a dialog box displaying the output value or a message.

Syntax:

MsgBox Prompt, [MsgBox style], [Title]

where Promt text is the prompt message, [MsgBox Style] is the msgbox style constants and [Title] is the text displayed in the title bar of the MsgBox dialog.

Example:

```
Private Sub cmdShowMessage_Click()
Dim n As Integer
n = 10
MsgBox "The value of n is " & n
End Sub
```

Example: MsgBox with the dialog type and title text.

```
Private Sub cmdShowMessage_Click()
Dim n As Integer
n = 10
MsgBox "The value of n is ", vbInformation, "Result" & n
End Sub
```

The MsgBox dialog box appears with the 'vbInformation' dialog type and the title text is "Result".

The list of dialog types is shown in the Quick Info text upon entering the Prompt text.



CLASS: II B.Sc.CT COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: III BATCH-2017-2020

Download the following programs:

- <u>InputBox & MsgBox 1</u>
- <u>InputBox & MsgBox2</u>

The above programs are explained here: <u>VB6 source code for beginners</u>.

Example: The following program shows how the MsgBox function returns value.

Private Sub Command1_Click() n = MsgBox("hello", vbOKCancel) If n = 1 Then Print "You have pressed ok" ElseIf n = 2 Then Print "You have pressed cancel" End If End Sub

When you click the button, the MsgBox dialog appears. Then the user either presses Ok or Cancel in this case. The MsgBox function returns 1 if you press Ok, it returns 2 if you press Cancel.

Line continuation character(_)

In your vb program you can type maximum of 1023 characters in a line. And sometimes, it doesn't fit in the window when we write so many characters in a line. So Line Continuation Character (underscore preceded by a space) is used to continue the same statement in the next line.

Example:

```
Private Sub cmdShow_Click()
MsgBox "Hello, welcome to www.vbtutes.com", _
vbInformation, "Welcome"
End Sub
```

Output using PictureBox



CLASS: II B.Sc.CT COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: III BATCH-2017-2020

PictureBox is also used for the output operation. The output is displayed using the Print method.

Example:

Private Sub cmdDisplay_Click() Dim n As Integer n = 40 picResult.Print n + 5 End Sub

The program prints 45 in the PictureBox.

Input-output with File

Input-output operations are also possible using File in Visual Basic discussed in the appropriate lessons. In this case, the input is taken from the data saved in Windows Notepad, in .txt extension and output can be displayed in the same file.

Output to the printer

The output can be displayed using printer . Visual Basic easily lets you display the output value or the output message using printer. Visual Basic treats the printer as an object named Printer. This topic is discussed in the appropriate lesson.





CLASS: II B.Sc.CT COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302UNIT: IIIBATCH-2017-2020

POSSIBLE QUESTIONS

Part – B (2 Marks)

- 1. Write the syntax for Select Case Statement
- 2. Write the uses of Object Browser
- 3. Write the syntax for Sub Procedure
- 4. Write the syntax for Input Box
- 5. List the I/O statements in visual basic.

Part – C (6 Marks)

- 1. Explain the various date and time function with example.
- 2. What are functions? How it is differ from procedure, explain?
- 3. Explain the creation of function in vb6.0 with sample program.
- 4. Explain Classes and Objects in detail.
- 5. What are the various ways of using the FOR-NEXT Loop? Explain exit statement with example
- 6. Discuss different string functions in detail with a sample program.
- 7. What are the various ways of using the FOR-NEXT Loop?
- 8. Explain Message Box and Input Box in Detail.
- 9. Describe Control and Looping Statements in VB with sample program.
- 10. Explain I/O Statements with example.

(Deemed University)

(Established Under Section 3 of UGC Act 1956)

Coimbatore - 641 021

(For the candidates admitted in 2017 onwards)

CLASS : II B.Sc CT

SUBJECT : Programming with visual basic

UNIT-3

Questions	opt1	opt2	opt3	opt4	Answer
contains the text box and list box	combo box	shape control	image control	timer control	combo box
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
The method is used to add items to a list at run time.	item	index	remove item	add item	add item
The method is used to remove an item from the list	item	index	remove item	add item	remove item
The property sets the index number of the currently selected item	index number	list index	list count	none	list index
The property returns the index of the last item added to the list	index number	list couont	new index	old index	new index
The sorted property is set to to enable a list to appear in alphanumeric order	0	1	TRUE	FALSE	TRUE
The sorted property is set to to display the items in the order in which they are added to the list	0	1	TRUE	FALSE	FALSE

is the style of dropdown combo box	style 0	style 2	style 3	style 1	style 0
is the style of simple combo box	style 0	style 1	style 3	style 2	style 1
is the style of dropdown list box	style 0	style 1	style 2	style 3	style 2
box saves the space on a form	list box	tool box	combo box	none	combo box
The property represents the minimum value in the scroll					
bar	min	max	minimum	maximum	min
The property represents the maximum value in					
the scroll bar	min	max	minimum	maximum	max
provides easy navigation through a list of items or					
a large amount of information	scroll bar	command button	tool bar	tool box	scroll bar
The style property in tab allows us to set the					
appearance of the control	line style	control	general	special	general
The property is used to set the style of the lines					
displayed between nodes	line style	special style	general style	node	line style
In a tree view control it is always better to first se the name					
property with a prefix	twv	t∨w	wvt	vwt	tvw
In tree view control 0 indicates line	tree	root	style	none	tree
In tree view control 1 indicates line	tree	root	style	none	root
enable of disable the automatic label editing feature of the					
control	label edit	line edit	enable	disable	label edit
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
configures the control for either manual / automatic					
dragging	OLE drag mod	OLE drop mode	label edit	image list	OLE drag mode
delimiter character used for the path returned by a node's full					
path property	path seperator	seperator	indentation	none	path seperator
uses to display the results of query on a					
DataBase.	tree view conti	view control	tree control	none	tree view contro

is the extension for the Bitmap files	.Btp	.BMP	.bmp	.dbs	.bmp
panels, which informs the user about the status of an					
application.	control bar	status bar	image bar	picture bar	status bar
An is an object that we place on a form to enable or					
enhance a user's interaction with an application.	Active X contro	Active Y control	Active Z control	none	Active X control
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
control serves as a visual and functional					
container for controls	form	frame	Ole	ADO	frame
displays a true/false of yes/no option	box	command button	text box	check box	check box
control is used link or embed on object, display and		100			
manipulate data from other windows based applications	Ole	ADO	DAO	none	Ole
control oddo o obono to o form		n i ati una	ahana		ahana
Control adds a shape to a form	image	picture	snape	none	snape
Box allows the user to select directories and paths	Dirlict	liet	tool	Dir tool	Dir liet
group applying the upper to collect and option over if it displays	Dir list	list	1001		Dir list
group addiws the user to select one option even in it displays	command	ontion	chock	imaga	option
how displays the valid disk drives and allows the	commanu	option	CHECK	linaye	οριιστ
box displays the valid disk drives and allows the	Dir list	Drivo list	list Drive	none	Drive list
				none	DIIVE IISt
box displays a set of files from which a user can					
select the lesired one	Drive list box	file list box	picture box	Dir list box	file list box
control draws a straight line to the form.	line	circle	oval	rectangle	line

control enable the user to connect to an existing					
DataBase and display information from it.	data base	data	file	list	data
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
An control is the rectangular portion into which picture files					
can be loaded.	line	shape	image	none	image
A bitmap also called graphics defines an image as					
a pattern of dots	paint type	circle type	oval type	drawing type	paint type
		_	_		
VB provides controls for creating graphical applications.	1	2	3	4	3
method sets the colour of the individual pixels	color	shape	control	none	color
method sets the colour of an individual pixel	set color	set pixel	set method	none	set method
OLE is	object linker ar	object linking and	object linking and	object linker and er	object linking an
DDE is	dynamic data e	dynemer data exc	dynamic datum ex	dynamic data exch	dynamic data e>
actually transfers control to the original	. – .				
application	ADO	DAO	OLE	DDE	OLE
The function can be used for creating the object in code	create object	get object	both	none	both
The form we work with during design time is a					_
	class	module	instance	sub routine	class
The OLE control can have only object at a time.	one	two	three	four	one
Each time an OLE control is drawn on a form object					
dialog box appears	delete	add	insert	none	insert
Paste special dialog box can be used to create an object during					
time.	design	run	execution	view	design
The method cab be used to specify an empty embedded					
object at runtime.	create embed	empty embed	OLE container	none	create embed

The method is used to display the insert object dialog					
box	obj dlg insert	insert obj dlg	dlg obj insert	none	insert obj dlg
property determines whether the OLE drop					
operations are allowed or not	OLE drop allow	OLE drop not allo	OLE drop mode pr	VB OLE drag auto	OLE drop allowe
The parameter describes the format of the data					
stored in the data parameter	object	data format	create	insert	data format
configures the Tree View control to enable / disable					
Ole drop operations	OLE drag mod	OLE drop mode	label edit	image list	OLE drop mode
The property determines the horizontal distance					
between nodes in the view	path seperator	seperator	indentation	none	indentation
function specifies the file name and assigns the					
picture to the picture property.	load picture	add picture	remove picture	none	load picture

I

id embedding

(change

d property



CLASS: II B.Sc.CT

COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: IV BATCH-2017-2020

SYLLABUS

Working with forms and procedures: Introducing to forms and types of forms and setting form properties, creating, adding, removing forms in project, hide, show method, load, unload, statement, Me keywords, Referring to objects on a different forms. Creating an application using controls: What is on the toolbar – Textbox control – Picture box – Image box – Label box – Frame – List box – Option button – Combo box – Command Button – check box – The Drive, Directory, File list controls – The Line & Shape control – Scroll Box – Data – Timer.

FORMS

The main characteristic of a Form is the title bar on which the Form's caption is displayed. On the left end of the title bar is the Control Menu icon. Clicking this icon opens the Control Menu. Maximize, Minimize and Close buttons can be found on the right side of the Form. Clicking on these buttons performs the associated function. The following figure illustrates the appearance of a Form



The control menu contains the following commands :

- Restore : Restores a maximized Form to the size it was before it was maximized; available only if the Form has been maximized.
- Move : Lets the user moves the Form around with the mouse
- Size : Lets the user resizes the control with the mouse
- Minimize: Minimizes the Form
- Maximize : Maximizes the Form
- Close : Closes the Form



CLASS: II B.Sc.CT COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: IV BATCH-2017-2020

Setting the Start-Up Form

A typical application has more than a single Form. When an application runs the main Form is loaded. By setting the Project properties you can control which Form is to be displayed in the Start-Up of the application. Following figure illustrates the Project property window.



By default, Visual Basic suggests the name of the first Form created when the project started. Loading and Unloading Forms

In order to load and unload the forms, Load and Unload statements are used. The Load statement has the following syntax :

Load FormName

And the Unload statement has the following syntax :

Unload FormName

The FormName variable is the name of the Form to be loaded or unloaded. Unlike the Show method which cares of both loading and displaying the Form, the load statement doesn't show the Form. You have to call the Form's Show method to display it on the desktop.

Showing Forms

Show method is used to Show a Form. If the Form is loaded but invisible, the Show method is used to bring the Form on Top every other window. If the Form is not loaded, the Show method loads it and then displays it.

Syntax of the Show method of the Form

FormName.Show mode

The FormName variable is the Form's name, and the optional argument mode determines whether the Form will be Modal or not. It can have one of the following syntax :

* 0-Modeless (default)

* 1-Modal



CLASS: II B.Sc.CT

COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: IV BATCH-2017-2020

- Modeless Forms are the normal Forms. Modeless Forms interact with the user and the user allowed switching to any other Form of the application. If you do not specify the optional mode argument, by default the mode is set to modeless.
- The Modal Forms takes the total control of the application where user cannot switch to any other Forms in the application unless the Form is closed. A modal Form, thus, must have a Close button or some means to close the Form in order to return to the Form where the Modal Form was loaded.

Hiding Forms

The Hide method is used to hide a Form. The following is the syntax of the Hide Method.

FormName.Hide

To hide a Form from within its own code, the following code can be used.

Me.Hide

You must understand that the Forms that are hidden are not unloaded ; they remains in the memory and can be displayed instantly with the Show Method. When a Form is hidden, you can still access its properties and code. For instance, you can change the settings of its Control Properties or call any Public functions in the Form.

The following is an example illustrates the Show method and Mode statement

* Open a new Project and save the Project

Design the application as shown below

Object	Property	Setting
	Caption	Form1
Form		
	Name	frm1
	Caption	Form2
Form		
	Name	frm2
*	Caption	Form3
Form		
	Name	frm3
		Clickon a
	Caption	button to
Label		display a Form
	Name	
		Label1



CLASS: II B.Sc.CT

COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: IV BATCH-2017-2020



The following code is typed in the Click event of the command buttons



Run the application. Clicking on the buttons will display the Forms respectively. But you can see that in the cmd2_Click() event additionally VbModal argument has been added. You can see the difference after you display the forms by clicking on the command buttons. You can notice that you cannot switch to any other Forms in the application unless you close the Form3.

TEXTBOX CONTROL

TextBox controls, which have a great many properties and events, are also among the most complex intrinsic controls.

Setting properties to a TextBox

- Text can be entered into the text box by assigning the necessary string to the text property of the control
- If the user needs to display multiple lines of text in a TextBox, set the MultiLine property to True
- To customize the scroll bar combination on a TextBox, set the ScrollBars property.
- Scroll bars will always appear on the TextBox when it's MultiLine property is set to True and its ScrollBars property is set to anything except None(0)
- If you set the MultilIne property to True, you can set the alignment using the Alignment property. The test is left-justified by default. If the MultiLine property is et to False, then setting the Alignment property has no effect.



CLASS: II B.Sc.CT COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: IV BATCH-2017-2020

Run-Time Properties of a TextBox control

The Text property is the one you'll reference most often in code, and conveniently it's the default property for the TextBox control. Three other frequently used properties are these:

The *SelStart* **property** sets or returns the position of the blinking caret (the insertion point where the text you type appears). Note that the blinking cursor inside TextBox and other controls is named caret, to distinguish it from the cursor (which is implicitly the mouse cursor). When the caret is at the beginning of the contents of the TextBox control, SelStart returns 0; when it's at the end of the string typed by the user, SelStart returns the value Len(Text). You can modify the SelStart property to programmatically move the caret.

The *SelLength* **property** returns the number of characters in the portion of text that has been highlighted by the user, or it returns 0 if there's no highlighted text. You can assign a nonzero value to this property to programmatically select text from code. Interestingly, you can assign to this property a value larger than the current text's length without raising a run-time error.

The *SelText* **property** sets or returns the portion of the text that's currently selected, or it returns an empty string if no text is highlighted. Use it to directly retrieve the highlighted text without having to query Text, SelStart, and SelLength properties. What's even more interesting is that you can assign a new value to this property, thus replacing the current selection with your own. If no text is currently selected, your string is simply inserted at the current caret position.

When you want to append text to a TextBox control, you should use the following code (instead of using the concatenation operator) to reduce flickering and improve performance:

Text1.SelStart = Len(Text1.Text) Text1.SelText = StringToBeAdded

One of the typical operations you could find yourself performing with these properties is selecting the entire contents of a TextBox control. You often do it when the caret enters the field so that the user can quickly override the existing value with a new one, or start editing it by pressing any arrow key:

Private Sub Text1_GotFocus() Text1.SelStart = 0 ' A very high value always does the trick. Text1.SelLength = 9999 End Sub

Always set the SelStart property first and then the SelLength or SelText properties. When you assign a new value to the SelStart property, the other two are automatically reset to 0 and an empty string respectively, thus overriding your previous settings.

The selected text can be copied to the Clipboard by using SelText:



CLASS: II B.Sc.CT COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: IV BATCH-2017-2020

Clipboard.SelText text, [format]

In the above syntax, text is the text that has to be placed into the Clipboard, and format has three possible values.

- 1. VbCFLink conversation information
- 2. VbCFRTF Rich Text Format
- 3. VbCFText Text

We can get text from the clipboard using the GetText() function this way:

Clipboard.GetText ([format])

The following Figure summarizes the common TextBox control's properties and methods.

Property/ Method	Description			
Properties				
Enabled	specifies whether user can interact with this control or not			
Index	Specifies the control array index			
Locked	this control is set to True user can use it else if this control is set to alse the control cannot be used			
MaxLength	Specifies the maximum number of characters to be input. Default value is set to 0 that means user can input any number of characters			
MousePointer	Using this we can set the shape of the mouse pointer when over a TextBox			
Multiline	By setting this property to True user can have more than one line in the TextBox			
PasswordChar	This is to specify mask character to be displayed in the TextBox			
ScrollBars	This to set either the vertical scrollbars or horizontal scrollbars to make appear in the TextBox. User can also set it to both vertical and horizontal. This property is used with the Multiline property.			
Text	Specifies the text to be displayed in the TextBox at runtime			
ToolTipIndex	This is used to display what text is displayed or in the control			
Visible	By setting this user can make the Textbox control visible or invisible at runtime			
Method				
SetFocus	Transfers focus to the TextBox			
Event procedures				
Change	Action happens when the TextBox changes			

Prepared by Dr.S.Kowsalya, Asst Prof, Department of CS, CA & IT, KAHE



CLASS: II B.Sc.CT

COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: IV BAT

BATCH-2017-2020

Click	Action happens when the TextBox is clicked
GotFocus	Action happens when the TextBox receives the active focus
LostFocus	Action happens when the TextBox loses it focus
KeyDown	Called when a key is pressed while the TextBox has the focus
KeyUp	Called when a key is released while the TextBox has the focus

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


CLASS: II B.Sc.CT COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: IV BATCH-2017-2020

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:

DestPictBox.PaintPicture SrcPictBox.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



CLASS: II B.Sc.CT

COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: IV BATCH-2017-2020

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



CLASS: II B.Sc.CT

COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: IV BATCH-2017-2020

Loop ' Next column

x = x + picSource.ScaleWidthLoop

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.

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,



CLASS: II B.Sc.CT

COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: IV BATCH-2017-2020

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.

LABEL CONTROLS

Most people use Label controls to provide a descriptive caption and possibly an associated hot key for other controls, such as TextBox, ListBox, and ComboBox, that don't expose the Caption property. In most cases, you just place a Label control where you need it, set its Caption property to a suitable string (embedding an ampersand character in front of the hot key you want to assign), and you're done. Caption is the default property for Label controls. Be careful to set the Label's TabIndex property so that it's 1 minus the TabIndex property of the companion control.

Other useful properties are BorderStyle(if you want the Label control to appear inside a 3D border) and Alignment (if you want to align the caption to the right or center it on the control). In most cases, the alignment depends on how the Label control relates to its companion control: for example, if the Label control is placed to the left of its companion field, you might want to set its Alignment property to 1-Right Justify. The value 2-Center is especially useful for stand-alone Label controls.



CLASS: II B.Sc.CT

COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: IV BATCH-2017-2020



Different settings for the Alignment property of Label controls.

You can insert a literal & character in a Label control's Caption property by doubling it. For example, to see Research & Development you have to type &Research && Development. Note that if you have multiple but isolated &s, the one that selects the hot key is the last one and all others are ignored. This tip applies to all the controls that expose a Caption property. (The & has no special meaning in forms' Caption properties, however.)

If the caption string is a long one, you might want to set the Label's WordWrap property to True so that it will extend for multiple lines instead of being truncated by the right border of the control. Alternatively, you might decide to set the AutoSize property to True and let the control automatically resize itself to accommodate longer caption strings.

You sometimes need to modify the default value of a Label's BackStyle property. Label controls usually cover what's already on the form's surface (other lightweight controls, output from graphic methods, and so on) because their background is considered to be opaque. If you want to show a character string somewhere on the form but at the same time you don't want to obscure underlying objects, set the BackStyle property to 0-Transparent.

If you're using the Label control to display data read from elsewhere—for example, a database field or a text file—you should set its UseMnemonics property to False. In this case, & characters have no special meaning to the control, and so you indirectly turn off the control's hot key capability. I mention this property because in older versions of Visual Basic, you had to manually double each & character to make the ampersand appear in text. I don't think all developers are aware that you can now treat ampersands like regular characters.

As I said before, you don't usually write code in Label control event procedures. This control exposes only a subset of the events supported by other controls. For example, because Label controls can never get the input focus, they don't support GotFocus, LostFocus, or any keyboard-related events. In practice, you can take advantage only of their mouse events: Click, DblClick, MouseDown, MouseMove, and MouseUp. If you're using a Label control to display data read



CLASS: II B.Sc.CT

COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: IV BATCH-2017-2020

from a database, you might sometimes find it useful to write code in its Change event. A Label control doesn't expose a specific event that tells programmers when users press its hot keys. You can do some interesting tricks with Label controls. For example, you can use them to provide rectangular hot spots for images loaded onto the form. To create that context-sensitive ToolTip, I loaded the image on the form using the form's Picture property and then I placed a Label control over the Microsoft BackOffice logo, setting its Caption property to an empty string and the BackStyle property to 0-Transparent. These properties make the Label invisible, but it correctly shows its ToolTip when necessary. And because it still receives all mouse events, you can use its Click event to react to users' actions.

FRAME CONTROLS

Frame controls are similar to Label controls in that they can serve as captions for those controls that don't have their own. Moreover, Frame controls can also (and often do) behave as containers and host other controls. In most cases, you only need to drop a Frame control on a form and set its Caption property. If you want to create a borderless frame, you can set its BorderStyle property to 0-None.

Controls that are contained in the Frame control are said to be child controls. Moving a control at design time over a Frame control—or over any other container, for that matter—doesn't automatically make that control a child of the Frame control. After you create a Frame control, you can create a child control by selecting the child control's icon in the Toolbox and drawing a new instance inside the Frame's border. Alternatively, to make an existing control a child of a Frame control, you must select the control, press Ctrl+X to cut it to the Clipboard, select the Frame control, and press Ctrl+V to paste the control inside the Frame. If you don't follow this procedure and you simply move the control over the Frame, the two controls remain completely independent of each other, even if the other control appears in front of the Frame control.

Frame controls, like all container controls, have two interesting features. If you move a Frame control, all the child controls go with it. If you make a container control disabled or invisible, all its child controls also become disabled or invisible. You can exploit these features to quickly change the state of a group of related controls.

COMMANDBUTTON CONTROL

Using CommandButton 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



CLASS: II B.Sc.CT

COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: IV BATCH-2017-2020

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 DblClick 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
```



CLASS: II B.Sc.CT

COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: IV BATCH-2017-2020

OPTIONBUTTON CONTROL

OptionButton controls are also known as radio buttons because of their shape. You always use OptionButton 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 OptionButton control are similar to those already described for CheckBox controls. You set an OptionButton 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 OptionButton'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 OptionButton controls is often hosted in a Frame control. This is necessary when there are other groups of OptionButton controls on the form. As far as Visual Basic is concerned, all the OptionButton 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— PictureBox, for example—but Frame controls are often the most reasonable choice.

CHECKBOX CONTROL

The CheckBox control is similar to the option button, except that a list of choices can be made using check boxes where you cannot choose more than one selection using an OptionButton. By ticking the CheckBox the value is set to True. This control can also be grayed when the state of the CheckBox is unavailable, but you must manage that state through code.

When you place a CheckBox control on a form, all you have to do, usually, is set its Caption property to a descriptive string. You might sometimes want to move the little check box to the right of its caption, which you do by setting the Alignment property to 1-Right Justify, but in most cases the default setting is OK. If you want to display the control in a checked state, you set its Value property to 1-Checked right in the Properties window, and you set a grayed state with 2-Grayed.



CLASS: II B.Sc.CT

COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: IV BATCH-2017-2020

The only important event for CheckBox controls is the Click event, which fires when either the user or the code changes the state of the control. In many cases, you don't need to write code to

handle this event. Instead, you just query the control's Value property when your code needs to process user choices. You usually write code in a CheckBox control's Click event when it affects the state of other controls. For example, if the user clears a check box, you might need to disable one or more controls on the form and reenable them when the user clicks on the check box again. This is how you usually do it (here I grouped all the relevant controls in one frame named Frame1):

Private Sub Check1_Click() Frame1.Enabled = (Check1.Value = vbChecked) End Sub

Note that Value is the default property for CheckBox controls, so you can omit it in code. I suggest that you not do that, however, because it would reduce the readability of your code.

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

Object	Property	Setting
	Caption	CheckBox
Form		
	Name	frmCheckBox
	Caption	Bold
CheckBox		
	Name	chkBold
	Caption	Italic
CheckBox		
	Name	chkItalic
	Caption	Underline
CheckBox		
	Name	chkUnderline
	Caption	Red
OptionButton		
	Name	optRed
	Caption	Blue
OptionButton		
	Name	optBlue
OptionButton	Caption	Green

* Design the Form as shown below



CLASS: II B.Sc.CT

COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302

BATCH-2017-2020

UNIT: IV

	Name	optGreen
	Name	txtDisplay
TextBox		
	Text	(empty)
	Caption	Exit
CommandButton		
	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()



CLASS: II B.Sc.CT

COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: IV BATCH-2017-2020

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

LISTBOX and COMBOBOX CONTROLS

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
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

Prepared by Dr.S.Kowsalya, Asst Prof, Department of CS, CA & IT, KAHE



CLASS: II B.Sc.CT

COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17C1	YU302 UNIT: IV BATCH-2017-2020
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.AddItemitem, Index

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

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

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

Prepared by Dr.S.Kowsalya, Asst Prof, Department of CS, CA & IT, KAHE



CLASS: II B.Sc.CT

COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: IV BATCH-2017-2020

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.





CLASS: II B.Sc.CT

COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: IV BATCH-2017-2020

- 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
	Caption	ListBox
Form		
	Name	frmListBox
	Text	(empty)
TextBox		
	Name	txtName

Label	Caption	Enter a name
	Name	lblName
ListBox	Name	lstName
Label	Caption	Amount Entered
	Name	lblAmount



CLASS: II B.Sc.CT

COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302

UNIT: IV

BATCH-2017-2020

	Caption	(empty)	
Label	Name	lblDisplay	
	Border Style	1 Fixed Single	
CommandButton	Caption	Add	
CommanuDucton	Name	cmdAdd	
CommandButton	Caption	Remove	
	Name	cmdRemove	
CommandButton	Caption	Clear	
	Name	cmdClear	
	Caption	Exit	
CommandButton	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



CLASS: II B.Sc.CT COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: IV BATCH-2017-2020

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



CLASS: II B.Sc.CT COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: IV BATCH-2017-2020

DriveListBox, DirListBox, And FileListBox Controls

The **DriveListBox** control is a specialized drop-down list that d drive string (such as "C:") to the Drive property in code. You can also read the Drive property to see which drive has been selected.

To make a DirListBox display the directories of the currently selected drive, you would set the **Path** property of the DirListBox control to the **Drive** property of the DriveListBox control in the **Change** event of the DriveListBox, as in the following statement:

Dir1.Path = Drive1.Drive

The **DirListBox** control displays a hierarchical list of the user's disk directories and subdirectories and automatically reacts to mouse clicks to allow the user to navigate among them. To synchronize the path selected in the DirListBox with a FileListBox, assign the **Path** property of the DirListBox to the **Path** property of the FileListBox in the**Change** event of the DirListBox, as in the following statement:

File1.Path = Dir1.Path

The **FileListBox** control lists files in the directory specified by its Path property. You can display all the files in the current directory, or you can use the **Pattern** property to show only certain types of files.

Similar to the standard ListBox and ComboBox controls, you can reference the **List**, **ListCount**, and **ListIndex** properties to access items in a DriveListBox, DirListBox, or FileListBox control. In addition, the FileListBox has a **MultiSelect** property which may be set to allow multiple file selection.

Example

Private Sub Command1_Click() End End Sub Private Sub Dir1_Change() File1.Path = Dir1.Path End Sub Private Sub Drive1_Change() Dir1.Path = Drive1.Drive End Sub



CLASS: II B.Sc.CT

COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302

UNIT: IV BATCH-2017-2020

Private Sub File1_Click() Dim f As String f = Dir1.Path & "/" & File1.FileName Picture1.Picture = LoadPicture(f) End Sub Private Sub Form_Load() File1.Pattern = "*.jpg" End Sub

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



CLASS: II B.Sc.CT

COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: IV BATCH-2017-2020

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

SCROLL BAR

The ScrollBar is a commonly used control, which enables the user to select a value by positioning it at the desired location. It represents a set of values. The Min and Max property represents the minimum and maximum value. The value property of the ScrollBar represents its current value, that may be any integer between minimum and maximum values assigned.

The HScrollBar and the VScrollBar controls are perfectly identical, apart from their different orientation. After you place an instance of such a control on a form, you have to worry about only a few properties: Min and Max represent the valid range of values, SmallChange is the variation in value you get when clicking on the scroll bar's arrows, and LargeChange is the variation you get when you click on either side of the scroll bar indicator. The default initial value for those two properties is 1, but you'll probably have to change LargeChange to a higher value. For example, if you have a scroll bar that lets you browse a portion of text, SmallChange should be 1 (you scroll one line at a time) and LargeChange should be set to match the number of visible text lines in the window.

The most important run-time property is Value, which always returns the relative position of the indicator on the scroll bar. By default, the Min value corresponds to the leftmost or upper end of the control:

' Move the indicator near the top (or left) arrow. VScroll1.Value = VScroll1.Min ' Move the indicator near the bottom (or right) arrow. VScroll1.Value = VScroll1.Max

While this setting is almost always OK for horizontal scroll bars, you might sometimes need to reverse the behavior of vertical scroll bars so that the zero is near the bottom of your form. This arrangement is often desirable if you want to use a vertical scroll bar as a sort of slider. You



CLASS: II B.Sc.CT

COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: IV BATCH-2017-2020

obtain this behavior by simply inverting the values in the Min and Max properties. (In other words, it's perfectly legal for Min to be greater than Max.)

There are two key events for scrollbar controls: the Change event fires when you click on the scroll bar arrows or when you drag the indicator; the Scroll event fires while you drag the indicator. The reason for these two distinct possibilities is mostly historical. First versions of Visual Basic supported only the Change event, and when developers realized that it wasn't possible to have continuous feedback when users dragged the indicator, Microsoft engineers added a new event instead of extending the Change event. In this way, old applications could be recompiled without unexpected changes in their behavior. At any rate, this means that you must often trap two distinct events:

'Show the current scroll bar's value. Private VScroll1_Change() Label1.Caption = VScroll1.Value End Sub Private VScroll1_Scroll() Label1.Caption = VScroll1.Value End Sub

The example shown in the following figure uses three VScrollBar controls as sliders to control the individual RGB (red, green, blue) components of a color. The three scroll bars have their Min property set to 255 and their Max property set to 0, while their SmallChange is 1 and LargeChange is 16. This example is also a moderately useful program in itself because you can select a color and then copy its numeric value to the clipboard and paste it in your application's code as a decimal value, a hexadecimal value, or an RGB function.



Scrollbar controls can receive the input focus, and in fact they support both the TabIndex and TabStop properties. If you don't want the user to accidentally move the input focus on a scrollbar control when he or she presses the Tab key, you must explicitly set its TabStop property to False. When a scrollbar control has the focus, you can move the indicator using the Left, Right, Up, Down, PgUp, PgDn, Home, and End keys. For example, you can take advantage of this behavior to create a read-only TextBox control with a numeric value that can be edited only through a tiny



CLASS: II B.Sc.CT

COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: IV BATCH-2017-2020

companion scroll bar. This scroll bar appears to the user as a sort of spin button, as you can see in the figure below. To make the trick work, you need to write just a few lines of code:

Private Sub Text1_GotFocus() ' Pass the focus to the scroll bar. VScroll1.SetFocus End Sub Private Sub VScroll1_Change() ' Scroll bar controls the text box value. Text1.Text = VScroll1.Value End Sub

DATE AND TIME FUNCTIONS

Not only does Visual Basic let you store date and time information in the specific Date data type, it also provides a lot of date- and time-related functions. These functions are very important in all business applications and deserve an in-depth look. Date and Time are internally stored as numbers in Visual Basic. The decimal points represents the time between 0:00:00 and 23:59:59 hours inclusive.

The system's current date and time can be retrieved using the Now, Date and Time functions in Visual Basic. The Now function retrieves the date and time, while Date function retrieves only date and Time function retrieves only the time.

To display both the date and time together a message box is displayed use the statement given below.

MsgBox "The current date and time of the system is" & Now

Here & is used as a concatenation operator to concentrate the string and the Now function. Selective portions of the date and time value can be extracted using the below listed functions.

Function	Extracted Portion
Year ()	Year (Now)
Month ()	Month (Now)
Day ()	Day (Now)
WeekDay ()	WeekDay (Now)



CLASS: II B.Sc.CT

COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: IV BATCH-2017-2020

Hour()	Hour (Now)
Minute ()	Minute (Now)
Second ()	Second (Now)

The calculation and conversion functions related to date and time functions are listed below.

Function	Description
DateAdd ()	Returns a date to which a specific interval has been added
DateDiff ()	Returns a Long data type value specifying the interval between the two values
DatePart ()	Returns an Integer containing the specified part of a given date
DateValue ()	Converts a string to a Date
TimeValue ()	Converts a string to a time
DateSerial ()	Returns a date for specified year, month and day

DateDiff Function

The DateDiff function returns the intervals between two dates in terms of years, months or days. The syntax for this is given below.

DateDiff (interval, date1, date2[, firstdayofweek[, firstweekofyear]])

Format Function

The format function accepts a numeric value and converts it to a string in the format specified by the format argument. The syntax for this is given below.

Format (expression[, format[, firstdayofweek[, firstweekofyear]]])

The Format function syntax has these parts:

Part	Description
Expression	Required any valid expression
format	Optional. A valid named or user-defined format expression.
firstdayofweek	Optional. A contant that specifies the first day of the week.



CLASS: II B.Sc.CT

KARPAGAM ACADEMY OF HIGHER EDUCATION

COURSE NAME: PROGRAMMING WITH VISUAL BASI(

COURSE CODE: 17CTU302UNIT: IVBATCH-2017-2020

firstweekofyear Optional. A contant that specifies the first week of the year

POSSIBLE QUESTIONS

Part – B (2 marks)

- 1. Write about hide and show methods of form with example.
- 2. Write down the operations of Combo Box
- 3. Write down the operations of List Box
- 4. How will you add picture to picture box at run time?
- 5. Name any five properties of Timer Control

Part – C (6 marks)

- 1. Explain the following controls with example (i) DriveListBox (ii) DirectoryListBox (iii) FileListBox
- 2. Write short notes on Timer Control. Write a program to animate the picture using timer control
- 3. Discuss about Text Box, Label Box, Command Button and Timer Controls in Visual Basic
- 4. Write a program to print the student grade for Five Subjects
- 5. Explain the following controls in detail :-(i) Check Boxes (ii) Images
- 6. Describe the purpose of each of the following methods or properties for a list box control with example: (i)ListIndex (ii) List (iii) RemoveItem.
- 7. Implement the following functionalities for a ListBox control. Adding item. Removing selected item, Count selected items and Clear the list of items and finally close the application. Use TextBox control to take item as an input and display output
- 8. Explain line and shape controls with program.
- 9. Explain the following controls with example : (i) CheckBox (ii) CommandButton
- 10. Create an application Having 3 horizontal scrollbars-Red, Green and Blue. The maximum value of any scrollbar should be 255. The application should also contain a picture box, whenever the value of any scrollbar is changed, the color corresponding to the combination of three values of 3 scrollbars should be applied to the Picturebox's background.

(Deemed University)

(Established Under Section 3 of UGC Act 1956)

Coimbatore – 641 021

(For the candidates admitted in 2017 onwards)

CLASS : II B.Sc CT

SUBJECT : Programming with visual basic

	01111 +				
Questions	opt1	opt2	opt3	opt4	Answer
is used for opening many windows at the same					
time	MDI	IDM	DIM	none	MDI
All the document windows are contained in a	child	parent	both	none	parent
VB application can have only MDI form.		1 2	3	3	4 1
Each time the user clicks new from the file menu, a new					
window is created and displayed.	parent	form	standard	child	child
'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
The resize event is fired whenever the size of the form is					
	fixed	not changeable	changed	none	changed
The is reserved word used in the form _ Resize()					
proccedure is a variable containing the names of the form	my	mine	myself	me	me
The ME keyword in VB behaves like an declared					
variable.	implicity	explicity	static	constant	implicity
is an area containing control to provide quick					
access to the most commonly used oprations	status bar	tool bar	grid	line	tool bar
Tool Bar is also called as	ribbon bar	control bar	both	none	both
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
Status bar appears at the of an MDI form.	top	bottom	middle	side	bottom
A can be used to add new commands to the existing					
menus, create new menus and menu bars, change or delete	Tool editor	Menu interface	Menu editor	none	Menu editor
	1			1	

UNIT-4

The includes all the menu controls of the current form.	Tool editor	Menu interface	Menu editor	none	Menu editor
A bar is a line which seperates the menu item.	join	include	divide	seperator	seperator
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
When the menu item is it is dimmed and cannot be					
selected, but viewed.	able	disable	view	visible	disable
To place a check mark in a menu item, the checked property					
is set to	TRUE	FALSE	0	1	TRUE
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
number of class modules can be created in a					
project.	less	1	2	any	any
Each class mocule can respond to only events.	1	2	3	4	2
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
The Event procedure contains code that needs					
to be executed at the moment the object is created.	Initialize	Terminate	stop	start	Initialize
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
Click the button to destroy the existing thing object					
and create a new one.	destroy old thir	create new thin	both	none	create new thing
Press the key bigger print to run the project.	F4	F5	F6	F9	F5
An is a combination of code and data that can be					
treated as a unit	class	module	object	none	object
The keyword is used only when we want VB to					
create a new instance of the original object.	old	new	fresh	begin	new
is the key concepts in working with objects.	polymorphism	inheritance	encapsulation	variables	encapsulation
combines data and characteristics into a single					
package.	encapsulation	inheritance	polymorphism	none	encapsulation
, the ability to make classes descend from other					
classes is not supported by VB.	encapsulation	inheritance	polymorphism	none	inheritance
In an Object Oriented Programming Language, the ability to					
redefine a routine in a derived class is called	encapsulation	inheritance	polymorphism	none	polymorphism

Is nelptul when creating new objects from old ones			1.1		
because it makes a programmers job simpler	polymorphism	encapsulation	inneritance	variables	polymorphism
IN VB, the setting or sttributes of an object are referred to as		a hia ata			
	properties	objects	methods	contrast	properties
The various procedures that can operate on the object are					
	objects	methods	properties	contrast	methods
Properties that we can set and those that we manipulate at run					
time are called	set manipulate	run time desigr	read write prop	none	read write proper
Properties that we can only read at run time	read only prop	only read prope	read write prop	none	read only property
can affect the values of properties	objects	classes	. modules	methods	methods
The role of is to free the space allocated for the					
constructor	destructor	collection	object	procedure	destructor
If multiple instances of the form are created using the DIM					
statement with the new keyword, a is created	collection objection	forms collection	constructors	destructors	collection object
A is an ordered set of items that can be considered					
as a unit.	collection objection	collections	constructors	destructors	collection object
method add items to the collection	remove	add	+	addition	+
method removes items from the collection	-	sub	remove	removal	remove
method returns the number of items present in the					
collection	return	count	number	none	count
method returns an item referenced by index					
property.	return	count	Item	ideal	ltem
' for eachnext ' statement uses an object called an					
	collection	control	enumerator	module	enumerator
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
is an array that contains all the controls on a		chamerator	0011011100101		chamerator
form in VB	Control Collec	class collection	module collection	none	Control Collectio
The property of the control collection returns the				none	Control Collectio
number count behave set	numbor	count	hohovo	cot	count
The statement is used to associate a particular		COUNT	Denave	301	count
control with an object variable	cot	rocot	class	object	cot
There are turned of acroll here are evoluble	561		CIASS		5 0 1
There are types of scroll bars are available.	1	2	3	4	2
The _ control is a visual element that contains several		line e	an atom alla		ahaaa
predefined snapes	snape	line	rectangle	square	snape

The control is a straight line segment that is drawn at					
design time.	square	rectangle	shape	line	line
The Fill color and Fill style properties in shape sontrol can be	not changed	changed	enabled	disabled	changed
The is a commonly used sontrol , which enables					
the user to select a value by positioning it at the desired					
location.	vertical scroll	Horizontal scro	scroll bar	status bar	scroll bar
control executes timer events at specifies					
intervals of time	time	timer	watch	none	timer
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 butto	option button	timer	option button
allows the user to select an item from the dropdown list					
box to type a selection in the text box.	combo box	command butto	check box	none	combo box
	TOUL				
I o create a child form, set its MDI child property to	IRUE	FALSE	0	1 1	IRUE

rties ∕



CLASS: II B.Sc.CT COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302

UNIT: V SYLLABUS

BATCH-2017-2020

Multiple Document Interface & Menus: Why MDI Forms – Features of an MDI forms – Loading MDI forms & child forms – creating a simple MDI forms –Accessing MDI froms – creating MENUS – POP-UP MENUS.

Data access controls: JET database Engine – ADODC – DAO Data control – ODBC Data Source Administrator – DATA REPORT.

MULTIPLE DOCUMENT INTERFACE (MDI)

The Multiple Document Interface (MDI) was designed to simplify the exchange of information among documents, all under the same roof. With the main application, you can maintain multiple open windows, but not multiple copies of the application. Data exchange is easier when you can view and compare many documents simultaneously.

You almost certainly use Windows applications that can open multiple documents at the same time and allow the user to switch among them with a mouse-click. Multiple Word is a typical example, although most people use it in single document mode. Each document is displayed in its own window, and all document windows have the same behavior. The main Form, or MDI Form, isn't duplicated, but it acts as a container for all the windows, and it is called the parent window. The windows in which the individual documents are displayed are called Child windows.

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

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

To create an MDI application, follow these steps:

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

Visual Basic automatically associates this new Form with the parent Form. This child Form can't exist outside the parent Form; in the words, it can only be opened within the parent Form.



CLASS: II B.Sc.CT COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: V BATCH-2017-2020

Parent and Child Menus

MDI Form cannot contain objects other than child Forms, but MDI Forms can have their own menus. However, because most of the operations of the application have meaning only if there is at least one child Form open, there's a peculiarity about the MDI Forms. The MDI Form usually has a menu with two commands to load a new child Form and to quit the application. The child Form can have any number of commands in its menu, according to the application. When the child Form is loaded, the child Form's menu replaces the original menu on the MDI Form

Following example illustrates the above explanation.

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

* Design a menu that has the following structure.

<>> MDIMenu Menu caption

- MDIOpen opens a new child Form
- MDIExit terminates the application

* Then design the following menu for the child Form

<> ChildMenu Menu caption

- Child Open opens a new child Form
- Child Save saves the document in the active child Form
- Child Close Closes the active child Form

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

Form1.Show

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

End

Double click on Child Close and enter the following code in the click event

Unload Me

Before run the application in the project properties set MDI Form as the start-up Form. Save and run the application. Following output will be displayed



CLASS: II B.Sc.CT COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: V BATCH-2017-2020

DATA CONTROL

The **ADO data control** presents a fast and easy way to create a bound form by providing builtin functions that define, navigate, display, add to, and update the records in a recordset. The resulting form makes it easy for a user (non-developer) to maintain records. Although the data control defines the recordset, you'll still need a control for each field to automatically display the bound records. The image below displays a simple form with several bound controls. The data control at the bottom of the form contains four navigation buttons. These buttons, from left to right, allow you to move to the first, previous, next, and last records in the form's recordset. If you update a record, VB will save that change when you move to another record. If you close the form before moving to another record. VB won't save changes you've made to the current record.

The ADO Data Control

Several of the controls found in Visual Basic's Toolbox can be data-bound, including the CheckBox, ComboBox, Image, Label, ListBox, PictureBox, and TextBox controls. Additionally, Visual Basic includes several data-bound ActiveX controls such as the DataGrid, DataCombo, Chart, and DataList controls. You can also create your own data-bound ActiveX controls, or purchase controls from other vendors.

Previous versions of Visual Basic featured the intrinsic Data control and the Remote Data control (RDC) for data access. Both controls are still included with Visual Basic for backward compatibility. However, because of the flexibility of ADO, it's recommended that new database applications be created using the ADO Data Control.

To create an ADO Data Control that exposes a Recordset in your application, at the minimum you need to do the following:

- Specify a Connection by filling in the ConnectionString property.
- Specify how to derive a Recordset by setting the RecordSource property (which is a complex property requiring its own dialog box to set up).

STEP BY STEP to Create an ADO Data Control

1. Add the Microsoft ADO DataControl 6.0 (OLEDB) from the Project, Components menu dialog box, as in Figure .The ADO Data Control icon should now appear in the VB toolbox.



CLASS: II B.Sc.CT COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: V BATCH-2017-2020

- 2. Place an instance of the ADO Data Control on the form (see Figure)
- 3. Change the control's Name and Caption from their default values. (The Caption is for information only, so you can set it to whatever you think will be most informative for the user.)
- 4. Set the ConnectionString property using steps 5–9.
- 5. Click the ellipsis next to the ConnectionString property in the ADO Data Control's Properties window to bring up the Property Page dialog box for this property, as shown in Figure .
- 6. As Source of Connection, choose one of the following three options:
 - Use Data Link File. If you choose this option, you will be able to click the Browse button to specify an existing *.UDL file).
 - Use ODBC Data Source Name. If you choose this option, you will be able to choose an existing ODBC DSN from the drop-down list, or you can create a new DSN by clicking the New button.
 - Use Connection String. If you choose this option, you will be able to click the Build button to bring up the Data Link Properties tabbed dialog box

.The following steps assume that you have chosen this option.

On the Provider tab of the Data Link Properties tabbed dialog box, choose an OLE DB data provider, such as Microsoft Jet 3.51 OLE DB (see Figure)

- 7. The Connection tab of the Data Link Properties tabbed dialog box will vary in appearance, depending on the provider specified in the preceding step. In the case of the Microsoft Jet 3.51 OLE DB, you are prompted to choose an Access data file and set some security options.
- 8. Click OK to accept the ConnectionString options you have built.

9. Still in the ADO Data Control's Properties window, navigate to the RecordSource property and click the ellipsis button.

10. On the RecordSource tab (see Figure 8.18) of the resulting Property Page dialog box, choose the CommandType (adCmdUnknown, adCmdText, adCmdTable, adCmdStoredProc).



CLASS: II B.Sc.CT COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: V BATCH-2017-2020

11. Complete the dialog box appropriately for the CommandType that you chose:

12.Click OK to end the RecordSource dialog box.

ADO Data Control Events

The ADO Data control features several events that you can program. The table below shows the events and when they occur; however the table is not meant to be a complete list all of the conditions when the events occur. For complete information, see the reference topic for the individual event.

Event	Occurs				
WillMove	On Recordset.Open, Recordset.MoveNext Recordset.Move, Recordset.MoveLast Recordset.MoveFirst, Recordset.MovePrevious Recordset.Bookmark, Recordset.AddNew Recordset.Delete, Recordset.Requery, Recordset.Resync				
MoveComplete	After WillMove				
WillChangeField	Before the Value property changes				
FieldChangeComplete	After WillChangeField				
WillChangeRecord	On Recordset.Update, Recordset.Delete, Recordset.CancelUpdate, Recordset.UpdateBatch, Recordset.CancelBatch				
RecordChangeComplete	After WillChangeRecord				
WillChangeRecordset	On Recordset.Requery, Recordset.Resync, Recordset.Close, Recordset.Open, Recordset.Filter				



CLASS: II B.Sc.CT	COURSE NAME: PROGRAMMING WITH VISUAL BASIC				
COURSE CODE: 17CTU302	UNIT: V	BATCH-2017-2020			
RecordsetChangeComplete	After WillChangeRecordset				
InfoMessage	When the data provider returns a result				

DATA ACCESS OBJECT CONTROL (DAO)

Declaring a Data Bound Control

A data bound control connects a data control in Visual Basic to a database table or query. Data controls are visual objects that are said to be data-aware. Data controls may include check boxes, images, labels, picture boxes, and text boxes .Visual Basic's data controls allow users to access stored database records. The data control establishes a link between the database and other controls in the interface, in a process called binding. When a control is bound, it displays database field contents when the Data control is present. Required property settings for the Data control include the **DatabaseName** property which specifies the complete path to the database and the **RecordSource** property, which indicates the table to use from the database. The **DataSource** property (specifies the name of the Data control to which it is bound) and the **DataField** property (specifies the name of a field in the database to which the control is linked) should be set for all form control that access the database information

Connecting a Data Bound Control with a Database

The properties list for a data control (see figures) includes a "DatabaseName" to specify the database the control is to connect to and a "Recordsource" to specify the table in the database, or the SQL statement, for the data. SQL stands for Structured Query Language--a high-level language used to manipulate and/or define databases.

Data controls rely on Data Access Objects (DAO) to perform processing. The DAO provide a more comprehensive interface to data using Visual Basic--they allow the user to run queries, update database table values and create the structure of databases (McManus, Database Access with Visual Basic 6.0).

In general, four parameters must be defined for data access:

- 1. Database Name--(DatabaseName) This is the full path to the database to which you would like to connect.
- 2. Table Name--(RecordSource) This is the name of the table in 1) to which you would like to connect.
- 3. Field Name--(DataField) This is the field to which your control should be connected.
- 4. Data Source--(DataSource) This is the name of the data control to be specified in linked components (such as text boxes). This is necessary because it is possible to have two data controls on one form.


CLASS: II B.Sc.CT COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: V BATCH-2017-2020

Manipulating Data Records with a Data Bound Control

Users may add records to the database with data controls in two ways:

- Using the **EOFAction** property, this indicates the action to take when the user tries to move past the last record in a table. Parameters for the EOFAction property are: 0 for Move Last (move the last record in the database), 1 for EOF (default), or 2 for Add New (adds a new record to the database).
- Using "AddNew" with the connected data control's record set.

Example. DataCustomer.Recordset.AddNew

Records may be updated using "Update" with the connected data control's record set in the same way "AddNew" is used.

Data binding features may be set for a control in the Procedure Attributes dialog box (see figure). The Procedure Attributes box is active after a procedure has been added. A procedure may be added to a module by choosing Add Procedure from the Tools menu. After adding the procedure, the full procedure attributes box is available under Tools, Procedure Attributes, and then Advanced. The "Name" property must be set to the name of the data control being bound, and the "Property is Data Bound" checkbox must be checked.

The checkbox labeled "This property binds to DataField" specifies that the property should appear alone under the DataField Property which is used when a control has a single bound property. The checkbox labeled "Show in DataBindings collection at design time" determines whether the bound property appears in the Properties window under the DataBindings property. The checkbox labeled "Property will call CanPropertyChange before changing" specifies that the control will check with the container to see if the property can be changed

Advantages and disadvantages of using DAO

On the plus side, DAO is fairly easy to use. And since DAO has been around longer than RDO or ADO and has been used in more projects, it pays to know how DAO works. Furthermore, if your application is running in a 16-bit environment, DAO is your only choice.

But DAO is older technology, and it doesn"t offer as much functionality as RDO and ADO. For instance, ADO can provide an interface to e-mail and file systems and custom business objects, as well as other sources. Microsoft is now focusing most of its improvements and advances on ADO, as well.

Generally, it's better to use DAO for accessing local databases where the speed is not the top priority and the number of users is limited, and to use either RDO or ADO for accessing remote databases and for larger scale projects.



CLASS: II B.Sc.CT COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: V BATCH-2017-2020

• 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.

VB projects that will process an Access-style (JET) database must include a reference to Microsoft DAO 3.51 Object Library. To include this reference, go to the VB Project menu and select References. From the resulting dialog box, check that reference (shown below):

Note: When you use the data control, setting this reference is not necessary (VB will set it automatically).

STRUCTURED QUERY LANGUAGE (SQL) BASICS

SQL is a common manipulation language used for database programming. *SQL* statements are like instructions which are sent to the *Database Server* and then processed. Statements range from creating, deleting or modifying data.

It is used to select the records form one or more tables in a relational database according to criteria that you have in that SQL statements. The most commonly SQL keyword used are SELECT, followed by one of these keywords WHERE, SELECT, FROM, HAVING, GROUPBY, or ORDER BY. If a data control has its database property set to Biblio.mdb you can use SQL statements.

The Queries

Data1.RecordSource = "SELECT [Name] FROM Publishers"

The FROM statement is require in every SQL SELECT statement. The FROM clause tells VB which table(s) or query(s) to examine to find the date.

The SELECT statement usually occurs first in as SQL statement It is almost followed by the field names. You can have multiple field names by using a comma between them.

Data1.RecordSource = "SELECT [Name], [State] FROM Publishers

Inorder, you can asterisk(*) to say you want all fields from the table

Data1.RecordSource = "SELECT * FROM Publishers"



CLASS: II B.Sc.CT COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: V BATCH-2017-2020

Incase, if you want a particular value from the table, the SQL statement is

Data1.RecordSource= "SELECT [Name] FROM Publishers WHERE State =

"NewYork" "

The single quotes inside the SQL statements is how you identify a string.

Finding Records Using SQL

You can use the four SQL Find methods combined with a SQL statement to examine th contents of a current Recordset attached to a data control. These functions are to find

- First Record to find the First Record
- FindLast to find the last record
- FindNext to theind the next record
- FindPrevious to find the previous record.

Syntax:

DataControlName.RecordSet.FindFirst SQL statement

The SQL criterion for the Findmethod is what would follow the Where clause in a SQL SELECT statement. For Example,

Data1.Recordset.FindFirst "State = ,,CA""

Use the NoMatch property of the RecordSet object to determine if a match was found.

Modifying a Table's Data through SQL

It is also possible that to write action queries that actually change data to match the conditions given in a SQL statement. The Keywords for Action Queries are

- UPDATE This method tells the access engine that changes should be made.
- SET it tells that which field should be changed and how.
- DELETE it allows the query to delete



CLASS: II B.Sc.CT COURSE NAME: PROGRAMMING WITH VISUAL BASIC

UNIT: V

COURSE CODE: 17CTU302

BATCH-2017-2020

• Execute Property – it carries out the change.

Example:

Dim ActionQuery As String ActionQuery = "UPDATE [ITEMS] " ActionQuery = ActionQuery & "SET [Current Price] = [Current Price]*.9 ActionQuery = ActionQuery & "WHERE [Placed On Shelf] <=1998" Data1.Database.Execute ActionQuery

Note: you can change several fields at the same time by separating commas.

DATABASE OBJECTS

The first thing you must do in your application is to open a database where your tables are stored. You need to declare a variable to hold your database in order to do this. This is done with:

Dim dbMyDB As Database

This gives you a variable/object that can hold a reference to your database. To open a simple Access database named "MyDatabase.mdb", do this:

Set dbMyDB = OpenDatabase("MyDatabase.mdb")

Note: You should really specify the complete path to the db, but if your current directory is the directory where the database is situated, this will work.

Hence, now you have opened a database. This won't give you any data. What you need to do is open a table in the database. You're not limited to open a single table; sometimes you have two or more tables that are related to each other and linked together with foreign keys, and there are ways to handle this to. But in this "Visual Basic - Database Primer" I will only show you how to open a single table.

RecordSet Object

Visual Basic uses an object called RecordSet to hold your table. To declare such an object and to open the table, do this:



CLASS: II B.Sc.CT COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: V Dim rsMyRS As RecordSet BATCH-2017-2020

Set rsMyRS = dbMyDB.OpenRecordSet("MyTable", dbOpenDynaset)

Here, I declared a RecordSet object and used the Database object's OpenRecordSet method to open a table of type Dynaset. You can open a RecordSet in several modes. VB's online help file explains the different modes and what they are for. The Dynaset mode is the mode I use mostly. It gives you a RecordSet that you can add, delete and modify records in.

Accessing records

Now that we have opened a table (referred to as RecordSet from now on) we want to access the records in it. The RecordSet object allows us to move in it by using the methods MoveFirst, MoveNext, MovePrevious, MoveLast (among others). I will use some of these to fill up a list box with the records of our RecordSet.

To get this example to work, make a database (with Access) called "MyDatabase.mdb" with the table "MyTable" in it. This table should have the fields "ID" of type "Counter" that you set to be the primary key, the field "Name" of type Text and a field "P hone" of type Text. Add some records to it. Put a list box on a form and call it "lstRecords".

Dim dbMyDB As Database Dim rsMyRS As RecordSet

Private Sub Form_Load()

Set dbMyDB = OpenDatabase("MyDatabase.mdb") Set rsMyRS = dbMyDB.OpenRecordSet("MyTable", dbOpenDynaset)

If Not rsMyRS.EOF Then rsMyRS.MoveFirst

Do While Not rsMyRS.EOF

lstRecords.AddItem rsMyRS!Name lstRecords.ItemData(lstRecords.NewIndex) =

rsMyRS!ID rsMyRS.MoveNext

Loop End Sub

This will make the list box fill up with your records when the form loads. I have introduced some new concepts with this example. We have all ready covered the first part where we open the



CLASS: II B.Sc.CT COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302UNIT: VBATCH-2017-2020table. The line that says If Not rsMyRS.EOF Then rsMyRS.M oveFirst tells the program to moveto the first record in case there are any records at all. The EOF is a Boolean property that is trueif the current record is the last. It is also true if there are no records in the RecordSet.

Then we make the program add the "Name" field of all records to the list box by adding the current records field "Name" and moving to the next record. You ask for a field of a RecordSet by putting a ! Between the name of the RecordSet object and the name of the field. The while loop checks to see if there are more records to add.

Searching the RecordSet

You might have wondered why I put the value of the field "ID" in the list box's ItemData property. I did this so that we would know the primary key for all the records in order to search for a record.

Put a text box somewhere on the form and call it "txtPhone". Then copy the following code to the project.

Private Sub lstRecords_Click()

rsMyRS.FindFirst "ID=" & Str(lstRecords.ItemData(lstRecords.ListIndex))

txtPhone.Text = rsMyRS!Phone

End Sub

This will display the phone number of the selected person when clicking in the list box. It uses the FindFirst method of the RecordSet object. This takes a string parameter that is like what is after WHERE in a SQL expression. You state the field that you want to search in (here "ID"), then the evaluation criteria (here "=") and last the value to search for (here the ItemData of the selected item in the list box).

So what we did was to search for the record with the "ID" field value that was the same as the ItemData property of the selected item in the list box. Then we show the value of the "Phone" field in the text.

Possible Questions

PART-B (Two marks)

1. Write any two features of MDI



CLASS: II B.Sc.CT COURSE NAME: PROGRAMMING WITH VISUAL BASIC

COURSE CODE: 17CTU302 UNIT: V

BATCH-2017-2020

- 2. Write any two differences between DAO and RDO
- 3. What is Record Set?
- 4. What is Database?
- 5. Define popup menus.

PART-C (6 marks)

- 1. Describe Data Access Controls in Visual Basic.
- 2. Write a Program to print the Employee Payroll with database Connectivity.
- 3. Describe Multiple Document interface in detail.
- 4. Write a Program to print the Students grade with database Connectivity.
- 5. Describe DAO Control in Visual Basic.
- 6. Write a Program for Railway Reservation and Cancellation with database connectivity.
- 7. Describe ADO Control in Visual Basic.
- 8. Write a program to draw different shapes and fill with different colors using MDI for
- 9. Explain how to generate report in visual basic with program.
- 10. Write a code that displays a popup menu with its top border centered on the form; when the user clicks the form with the right mouse button the popup menu will be displayed.

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

Coimbatore – 641 021

(For the candidates admitted in 2017 onwards)

CLASS : II B.Sc CT

SUBJECT : Programming with visual basic

UNIT-5

Questions	opt1	opt2	opt3	opt4	Answer
compile class-based projects					
such as components.	Active X	Active Y	Active Z	None	Active Y
process server.	in	out	up	down	in
, that can be used with	file make	object servers	compilation	none	object servers
trap run time errors in our project	error checking	error finding	error catching	none	error checking
pages that contain VB script code, which is executed	с	C++	VB	HTML	HTML
data sources through the JET database engine	Data Access Objects	data control	cointainer	relations	Data Access Objects
to microsoft access and other ODBC data sources.	DAO	ADO	Data control	Data	Data control
framework for using code to					
create and manipulate	RDO	RDC	ODBC	VBSQL	RDO
ODBC remote DataBase .	RDO	RDC	ODBC	VBSQL	RDC
to the open DataBase connectivity libraries and drivers to provide	RDO	RDC	ODBC	VBSQL	ODBC
of the DataBase library API specifically designed to provide	RDO	RDC	ODBC	VBSQL	VBSQL
model that eliminates the need to choose from among DAO and	ADO	RDC	ODBC	DB	ADO
collection of classes that model the structure of a	module	procedure	object	programe	object
to a stored table definition.	DataBase	TableDef	QueryDef	Record set	TableDef
query definition , which is a pre compiled SQL statement.	DataBase	TableDef	QueryDef	Record set	QueryDef

to a view into a database							
table or the results of a query.	cursored	pointer	manual	none	cursored		
rows of data in buffer and points							
to one row of data at a time called	previous	next	side	current	current		
parameter associated with a							
query def object created form a	index	user object	parameter	paranthesis	parameter		
information about one instance of							
a type of object.	group	relation	property	document	document		
built-in characteristic or a user							
defined characteristic of a DAO.	property	group	document	relation	property		
information describing the objects							
that are grouped into that	container	holder	DataBase	DataBase object	container		
DAO libraries supported by Visual	1	2	3	4	2		
the method of the							
workspace object is used.	open database	create database	database	none	open database		
a set of records from database.	set	records	record set	none	record set		
the first row in the record set.	move first	move next	move previous	move last	move first		
last row in the record set.	move first	move next	move previous	move last	move next		
the previous row in the record set.	move first	move next	move previous	move last	move previous		
the last row in the record set.	move first	move next	move previous	move last	move last		
when the user moves beyond the							
last record in the record set.	EOF	BOF	Both	none	EOF		
when the user has moved to a							
position before the first record in	EOF	BOF	Both	none	BOF		
are used to a record in a	manipulate	edit	change	modify	manipulate		
used to locate a record in a table							
type record set.	open	close	store	seek	seek		
perform action queries.	open	close	store	execute	execute		
existing fields can be deleted							
using the methods	append	delete	a or b	a and b	a or b		
The command is							
used to add rows to a table.	add	insert	sub	delete	insert		
based programming interface							
designed to access a wide variety	ADO	DAO	OLEDB	RDO	OLEDB		
objects is the object							

based interface to OLEDB.	Active X data	Active Y data	Active Z data	none	Active X data		
current connections we have built							
to databases, as well as any data	Data view window	Data report designer	Query designer	Data environment de	Data view window		
report headers , detail lines and							
many other common features,	Data view window	Data report designer	Query designer	Data environment des	Data report designer		
to design queries save them in	Dataview window	Data report	Query	Data Environment	Query		
we can link it to all our databases,							
tables and Queries with a single	Dataview window	Data report	Query	Data Environment	Data Environment		
ADO code that is designed to							
browse records in the same way	Data form wizard	Data view window	Query	Data Environment	Data form wizard		
RDO is	Remote Data Objects	Remotes Data Objects	Remote Datum Object	Remote Data Object	Remote Data Objects		
method is used to create a							
new record in the record set.	add new	create new	new add	none	add new		
to delete an existing record in							
dynaset of table type record set.	delete method	delete existing	delete	none	delete method		
The method is used to							
retrive data from the tables.	execute	run	stop	open record set	open record set		
open a table Dynaset of Snapshot							
record set from the tabledef object	open record set	create record set	open table	create table	open record set		
and refreshes any attached table							
links for the tabledef object.	new link	fresh link	refresh link	none	refresh link		
to create and store a user-defined	create index	create property	create field	create table	create property		
an index to the table def object.	add index	new index	create index	fresh index	create index		
a new field to an existing tabledef	add field	new field	create field	fresh field	create field		
statements that perform specific							
actions on the database.	SQL	action	calculation	None	action		
with similar access rights.	set	gang	group	object	group		
a relationship between fields in							
tables or queries.	property	document	relation	None	relation		
define and enforce database							
	user	relation	property	document	user		
to a column of data type and set							
of properties.	field	Table	Query	record	field		
represents the object.	database	visual basic	HTML	DHTML	database		

identical to the snapshot record					
set except that we can only scorll	Backward only type record	Reverse only type recor	Forward only type reco	none	Forward only type rec
can refer to any table, attached	snap shot	backward	forward	dynaset	snap shot
A type record set cannot be update and does not reflect any changes made by the users.	dvnaset	forward	backward	snap shot	snap shot

ord set

Reg. No -----

[17CTU302]

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University)

(Established Under Section 3 of UGC Act 1956) FIRST INTERNAL EXAMINATION, JULY 2018

Third Semester

COMPUTER TECHNOLOGY

PROGRAMMING IN VISUAL BASIC

Time : 2 Hours Class: II B.Sc. CT Date/Session: Maximum : 50

PART-A Answer ALL the questions

[20 * 1 = 20 Marks]

1.	met	thod is used to di	splay the for	m object		
	a.Load	b. visibl	e c.show	w d.dis	play	
2.		method is use	d to load a fo	orm or control	into memory but does no	ot
	display it				·	
	a.Load	b.visible d	c.show	d.display		
3.		displays a text th	nat the user c	annot modify	or interact with	
	a.Label	b.text be	ox c.time	er d. lin	e	
4.		is the foundation	n of object or	iented program	nming in VB	
	a. Standard	b.class of	c.form	d.reference		
5.	Visual basic w	was developed fo	rm the progra	amming langu	age	
	a.FORTRAN	b. C	c. BA	SIC d. C-	-+	
6.	Visual basic is	s developed in th	e year			
	a.1978	b. 1980	c.1970	0 d.197	72	
7.	IDE is					
	a. Integrated d	levelopment env	ironment	b. Integrated	developed environment	
	c. Integration	development env	vironment	d.none	L.	
8.	IDE is also co	mmonly referred	l to as		- environment	
	a.interface	b.develo	pmentc.desi	gn	d.integrated	
9.	The windows	associated with	he project w	ill stay with in	a single container called	1
			1 0	-	-	
	a.child	b.parent	c.code	e	d.module	
10.		lets windows	decide wheth	her the form sh	ould be shown	
	a.center screen	n b.manua	al c.cent	er owner	d.windows default	
11.	A complete re	paint of a form of	or control can	be enforced b	y method	
	a.refresh	b.set for	cus c.char	nge	d.repaint	
12.		box displays	the name of	the selected of	bject associated with the	form
	a.object	b.tool	c.text		d.command	
13.	i	s an action that c	an be perform	ned on objects	5	
	a.form	b.move	c.clicl	ζ. ž	d.method	
14.	va	riable is one that	is declared i	nside a proced	lure	
	a.global	b.local	c.stati	c	d.scope	

- 15. Variables that are declared with keyword _____ exist only as long as the procedure is being executed
 - a.public b.private c.dim d.double
- 16. The ______ in the menubar provide quick access to the commonly used commands
- a.Toolbars b.toolbox c.project window d.form
- 17. _____ helps to move and resize the controls and forms
- a. Label b.shape c.OLE d.pointer

18. _____ carries out the specified action when the user choose it
 a.option button
 b.image control
 c.shape control
 d.command button
 19. is the control used to display messages and enter text

- a.tool box b.text box c.command button d.option button
- 20._____ control draws a straight line to the form. a.line b.circle c.oval d.rectangle

PART-B

[3 * 2= 6 Marks]

Answer ALL of the following

- 21. Write any two features of VB.
- 22. What is Graphical user interface?
- 23. Define Variable.

PART-C

[3 * 8 = 24 Marks]

Answer ALL of the following

24. a. Explain Visual Basic Development Environment in detail.

(**OR**)

- b. Explain Tool Box in Visual Basic 6.0
- 25. a. Explain the following (i) Project Explorer (ii) Properties (iii) Code window (**OR**)
- b. Write a program to draw several shapes and fill with different colors.
- 26. a. What is Variables? Explain rules for declaring variables in VB.

(**OR**)

b. What is Data type? Explain Different types of Data types in VB.

Reg. No -----

[17CTU302]

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University) (Established Under Section 3 of UGC Act 1956) FIRST INTERNAL EXAMINATION, JULY 2018

Third Semester

COMPUTER TECHNOLOGY PROGRAMMING IN VISUAL BASIC

Time : 2 Hours Class: II B.Sc. CT Date/Session: Maximum : 50

[20 * 1 = 20 Marks]

PART-A

Answer All the questions

1.	me	thod is used to disp	lay the form obj	ect	
	a. Load	b. visible	c.show	d.display	
2.		method is used	to load a form	n or control into memory but does no	t
	display it			·	
	a.Load	b.visible c.sł	how d.di	splay	
3.		displays a text that	the user cannot	modify or interact with	
	a. Label	b.text box	c.timer	d. line	
4.		is the foundation of	f object oriented	l programming in VB	
	a. Standard	b.class c.fe	orm d.re	ference	
5.	Visual basic v	vas developed form	the programming	ng language	
	a. FORTRAN	b. C	c. BASIC	d. C++	
6.	Visual basic i	s developed in the y	/ear		
	a.1978	b. 1980	c. 1970	d.1972	
7.	IDE is				
	a. Integrated	development envir	ronment b. Ir	ntegrated developed environment	
	c Integration	dervelopment enviro	. 1		
	c. megration	development enviro	onment d.nc	one	
8.	IDE is also co	ommonly referred to	onment d.nc	one environment	
8.	IDE is also co a.interface	mmonly referred to b.developm	onment d.nc o as nent c. desig	ne environment n d.integrated	
8. 9.	IDE is also co a.interface The windows	b.development enviro b.developm associated with t	onment d.no o as nent c. desig he project will	one environment gn d.integrated stay with in a single container called	1
8. 9.	IDE is also co a.interface The windows	mmonly referred to b.developm associated with t	onment d.nc o as nent c. desig he project will	one environment gn d.integrated stay with in a single container called	b
8. 9.	IDE is also co a.interface The windows a.child	b.parent	nent d.no o as nent c. desig he project will c.code	one environment gn d.integrated stay with in a single container called d.module	ł
 8. 9. 10. 	IDE is also co a.interface The windows a.child	b.development enviro b.developm associated with t b. parent lets windows de	nment d.no o as nent c. desig he project will c.code cide whether the	one environment gn d.integrated stay with in a single container called d.module e form should be shown	b
8. 9. 10.	IDE is also co a.interface The windows a.child a.center screet	b.development enviro b.developm associated with t b. parent lets windows dea n b.manual	nent d.nc o as nent c. desig he project will c.code cide whether the c.center ow	one environment gn d.integrated stay with in a single container called d.module e form should be shown oner d. windows default	b
 8. 9. 10. 11. 	IDE is also co a.interface The windows a.child a.center screet A complete re	b.development enviro b.developm associated with t b. parent lets windows dee b.manual epaint of a form or c	nent d.no o as he project will c.code cide whether the c.center ow control can be er	one environment gn d.integrated stay with in a single container called d.module e form should be shown oner d.windows default nforced by method	t
 8. 9. 10. 11. 	IDE is also co a.interface The windows a.child a.center screet A complete re a. refresh	b.development enviro b.developm associated with t b. parent lets windows dea b.manual paint of a form or c b.set focus	onment d.nc o as he project will c.code cide whether the c.center ow control can be er c.change	one environment gn d.integrated stay with in a single container called d.module e form should be shown mer d. windows default nforced by method d.repaint	đ
 8. 9. 10. 11. 12. 	IDE is also co a.interface The windows a.child a.center screet A complete re a. refresh	b.parent b.manual b.manual b.parent b.parent b.manual b.manual b.set focus b.set focus	onment d.no o as nent c. desig he project will c.code cide whether the c.center ow control can be er c.change e name of the se	environment environment gn d.integrated stay with in a single container called d.module e form should be shown mer d.windows default nforced by method d.repaint elected object associated with the form	£
 8. 9. 10. 11. 12. 	IDE is also co a.interface The windows a.child a.center screet A complete re a. refresh a. object	b.parent b.developm b.developm associated with t b.parent lets windows dea b.manual paint of a form or c b.set focus b.set focus b.tool	onment d.no o as nent c.desig he project will c.code cide whether the c.center ow control can be er c.change e name of the se c.text	environment environment gn d.integrated stay with in a single container called d.module e form should be shown mer d.windows default nforced by method d.repaint elected object associated with the form d.command	f
 8. 9. 10. 11. 12. 13. 	IDE is also co a.interface The windows a.child a.center screet A complete re a. refresh a. object i	b.parent b.development enviro b.developm associated with t b.parent lets windows dea n b.manual paint of a form or c b.set focus box displays th b.tool s an action that can	onment d.nc o as nent c. desig he project will c.code cide whether the c.center ow control can be er c.change e name of the se c.text be performed o		b
 8. 9. 10. 11. 12. 13. 	IDE is also co a.interface The windows a.child a.center screet A complete re a. refresh a. object i a.form	b.parent b.development enviro b.developm associated with t b.parent lets windows dee n b.manual epaint of a form or c b.set focus box displays th b.tool s an action that can b.move	onment d.no o as nent c.desig he project will c.code cide whether the c.center ow control can be er c.change e name of the se c.text be performed of c.click		1
 8. 9. 10. 11. 12. 13. 14. 	IDE is also co a.interface The windows a.child a.center scree: A complete re a. refresh a. object a.form	b.parent b.development enviro b.developm associated with t b.parent lets windows dea b.manual paint of a form or c b.set focus box displays th b.tool s an action that can b.move wriable is one that is	onment d.no o as nent c.desig he project will c.code cide whether the c.center ow control can be er c.change e name of the se c.text be performed of c.click declared inside		d

- 15. Variables that are declared with keyword _____ exist only as long as the procedure is being executed
 - a.public b.private c.**dim** d.double
- 16. The ______ in the menubar provide quick access to the commonly used commands
- a.**Toolbars** b.toolbox c.project window d.form
- 17. _____ helps to move and resize the controls and forms
- a. Label b.shape c.OLE d.**pointer**
- 18. ______ carries out the specified action when the user choose it a.option button b.image control c.shape control d.command button
 19. ______ is the control used to display messages and enter text
- a.tool box b.**text box** c.command button d.option button

20._____ control draws a straight line to the form.a.lineb.circlec.ovald.rectangle

PART-B Answer ALL of the following [3 * 2= 6 Marks]

21. Write any two features of VB.

Features of visual basic:-

- It introduce the concept of event driven programming.
- Easier comparison
- User friendliness
- Faster application development

22. What is Graphical user interface?

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.

23. Define Variable?

Variables are the memory locations which are used to store values temporarily. A defined naming strategy has to be followed while naming a variable.

PART-C Answer ALL of the following

[3 * 8 = 24 Marks]

24.a. Explain Visual Basic Development Environment in detail.

The Visual Basic Environment IDE:

To launch Visual Basic, on the Taskbar, click Start -> (All) Programs -> Microsoft Visual Studio 6.0 -> Microsoft Visual Basic 6.0.



- Double Click on the Standard EXE option.
- VB environment comes up.
- 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).
- 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 CodeWindow. 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.

Visual Basics Environment:-



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 y 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 show 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.

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

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.



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

]	F)	C)	R	21	V	1]	D]	E	S	Ι	(Ĵ	N	Ţ	F]	R																			
		3	F	0	5		1																							[C	1		×	<	
F	1	1	1	1	1	1	1					1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
Ł							•			-				•	•						•	•							•					-	•	•	•		
Ł	•	•			•	•	•	•	•		-	•		•	•	•			•	•	•	•	•	•	•	•	•	•	•	•	•	-	•	•	•	•	-	•	
ŀ.	-			-	-		•	-		-	-	-	-	•	•	•		-	-	-	•	•	-		-	-	-	•	•	-		-	-	-	•	•	-		
r.							•				-				•						•								•							•			
Ε.	1		1	1				1					1	1			1	1	1	1	1		1	2	1	1	1	1		1	2	1	2		1	2	÷.	0.1	
[1	
Ł		-	-	-	-	-			-	-	-	-					-	-	-			-		-	-	-	-				-	-	-	-					
Ł				-	-					-	-	-							-			-				-								-					
Ł	•	•				•	•	•	•		-			•	•	•					•	•		•			•	•	•	•	•		•	•	•	•	•		
ł.						•	•	•			-			•	•	•					•	•						•	•					•	•	•	•		
ł.	•						•	•						•	•						•	•						•	•						•	•	•		
Ē.										-																							Ξ.					÷.	
[- 2	- 1																						2						
Ļ.																																							
ŀ.																																							
ŀ																																							
Ł	·	•	-	-	-	•	•	•	•	-	-	-	•	·	•	•	•	•	-	•	·	-	•	•	-	-	•	·	•	•	•	•	-	-	·	•	•	•	
F	•	•	-	-	-	•	•	•	•	-	-	-	•	•	•	•	-	•	-	•	•	•	•	•	-	-	•	•	•	•	•	•	-	-	•	•	•		
ŀ.	•			-	-		•	•		-	-	-	•	•	•				-	•	•	-			-	-	-	•	•				-	-	•	•	•		
t	1		1					1					1	1				1		1	1	1	1	1	1			1		1	1	1	1		1	1	1	1	
Į.																			1					1	1	1	1				1								

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:

- Present in view menu (or)press F2 function key
- Left columns lists out the object&classes that are available in the project.
- Object picked from the class. we can see its members

(**OR**)

b.Explain Tool Box in Visual Basic 6.0? 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.



Pointer Provides a way to move and resize the controls form

PictureBox Displays icons/bitmaps and metafiles. It displays text or acts as a visual container for other controls.

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

25. a. Explain the following (i) Project Explorer (ii) Properties (iii) Code window? 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 show 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.

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.

Properties - Form	Properties - Form1 🛛 🔀							
Form1 Form								
Alphabetic Cate	gorized							
(Name)	Form1							
Appearance	1 - 3D							
AutoRedraw	False							
BackColor	8H800000F&							
BorderStyle	2 - Sizable							
Caption	Form1							
ClipControls	True							
ControlBox	True							
DrawMode	13 - Copy Pen							
DrawStyle	0 - Solid							
DrawWidth	1							
Enabled	True							
FillColor	8H00000008							
FillStyle	1 - Transparent							
Font	MS Sans Serif							
FontTransparent	True							
Fore⊂olor	&H80000012&							
Height	6285							
HelpContextID	0							
Icon	(Icon)							
KeyPreview	False							
L off	n 🗾							

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 \rightarrow Code.



- **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**: IntelliSence 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 suntax for that element.

(OR) b.Write a program to draw several shapes and fill with different colors? Using the Shape control:

There are four basic controls in VB6 that you can use to draw graphics on your form: the line control, the shape control, the image box and the picture box. To draw a straight line, just click on the line control and then use your mouse to draw the line on the form. After drawing the line, you can then change its color, width and style using the BorderColor, BorderWidth and BorderStyleproperties.Similarly, to draw a shape, just click on the shape control and draw the shape on the form. The default shape is a rectangle, with the default shape property set at 0. You can change the shape to square, oval, circle and rounded rectangle by changing the shape property's value to 1, 2, 3, 4, and 5 respectively. In addition, you can change its background color using the BorderColor property, its border style using the BorderStyle property, its border color using the BorderColor property as well its border width using the BorderWidth property.

Example:

This example will draw a circle, a sector, an arc and an ellipse with different colors and different DrawStyles

The Code

Dim pi As Single Private Sub Form Load() 'Change the coordinates of the origin Scale (-4000, 4000)-(4000, -4000) Me.Width = 8000Me.Height = 8000End Sub Private Sub cmd Draw Click(Index As Integer) pi = 4 * Atn(1)Select Case Index Case Is = 0Me.FillColor = vbRed Me.FillStyle = vbSolid Me.DrawStyle = vbDotMe.DrawWidth = 1Me.Circle (0, 0), 800, vbYellow ' Draw a circle with red border Case Is = 1Me.FillColor = vbCyan Me.FillStyle = vbSolid Me.DrawStyle = vbDotMe.DrawWidth = 3Me.Circle (-2000, 0), 800, vbBlack, -pi, -pi / 2 Case Is = 2Me.FillColor = vbYellow Me.FillStyle = vbSolid Me.DrawStyle = vbDashDot Me.DrawWidth = 1Me.Circle (-2000, -2000), 800, vbBlue, 0, pi Case Is = 3Me.FillColor = vbBlue Me.FillStyle = vbSolid Me.DrawStyle = vbDashDotDot Me.DrawWidth = 1Me.Circle (-1500, 2000), 800, vbCyan, , , 1.3 End Select End Sub

The Output



26. a. What are Variables? Explain rules for declaring variables in VB?

VARIABLES:

Variables are the memory locations which are used to store values temporarily. A defined naming strategy has to be followed while naming a variable.

Rules of variables:

A variable name must begin with an alphabet letter and should not exceed 255 characters. It must be unique within the same scope. It should not contain any special character like %, &, !, #, @ or \$.

There are many ways of declaring variables in Visual Basic:

Depending on where the variables are declared and how they are declared, we can determine how they can be used by our application. The different ways of declaring variables in Visual Basic are listed below and elucidated in this section.

- Explicit Declaration
- Using Option Explicit statement
- Scope of Variables

Explicit Declaration:

Declaring a variable tells Visual Basic to reserve space in memory. It is not must that a variable should be declared before using it. Automatically whenever Visual Basic encounters a new variable, it assigns the default variable type and value. This is called **implicit declaration**. Though this type of declaration is easier for the user, to have more control over the variables, it is advisable to declare them explicitly. The variables are declared with a Dim statement to name the

variable and its type. The As type clause in the Dim statement allows to define the data type or object type of the variable. This is called **explicit declaration**.

Syntax: Dim variable [As Type] For example, Dim str as string. Dim a as integer.

Option Explicit:

This forces the user to declare all the variables. The Option Explicit statement checks in the module for usage of any undeclared variables and reports an error to the user. The user can thus rectify the error on seeing this error message.

The Option Explicit statement can be explicitly placed in the general declaration section of each module using the following steps.

- Click Options item in the Tools menu
- Click the Editor tab in the Options dialog box
- Check Require Variable Declaration option and then click the OK button

Scope of variables:

A variable is scoped to a procedure-level (local) or module-level variable depending on how it is declared. The scope of a variable, procedure or object determines which part of the code in our application are aware of the variable's existence. A variable is declared in general declaration section of e Form, and hence is available to all the procedures. Local variables are recognized only in the procedure in which they are declared.

Local variables:

A local variable is one that is declared inside a procedure. This variable is only available to the code inside the procedure and can be declared using the Dim statements as given below.

Dim sum As Integer

The local variables exist as long as the procedure in which they are declared, is executing. Once a procedure is executed, the values of its local variables are lost and the memory used by these variables is freed and can be reclaimed.

Static variables:

These variable are not reinitialized each time. The value are preserved value even when a procedure ends. These variables are used to keep back of no.of.times a control is clicked. They are also ideal for making controls alternatively visible (or)invisible.

eg:

```
static p as integer
private sub command1_click()
static counter as integer
counter=counter+1
end sub
```

(**OR**)

b. What is Data type? Explain Different types of Data types in VB.

Data types in Visual Basic 6:

By default Visual Basic variables are of variant data types. The variant data type can store numeric, date/time or string data. When a variable is declared, a data type is supplied for it that determines the kind of data they can store. The fundamental data types in Visual Basic including variant are integer, long, single, double, string, currency, byte and boolean. Visual Basic supports a vast array of data types. Each data type has limits to the kind of information and the minimum and maximum values it can hold. In addition, some types can interchange with some other types.

A list of Visual Basic's simple data types are given below

•

Byte	Store integer values in the range of 0 - 255						
Integer	Store integer values in the range of (-32,768) - (+ 32,767)						
Long	Store integer values in the range of (- 2,147,483,468) - (+ 2,147,483,468)						
Single	Store floating point value in the range of (-3.4x10-38) - (+ 3.4x1038)						
Double	Store large floating value which exceeding the single data type value						
Currency	store monetary values. It supports 4 digits to the right of decimal point and 15 digits to the left						

2. String

Use to store alphanumeric values. A variable length string can store approximately 4 billion characters

3. Date

Use to store date and time values. A variable declared as date type can store both date and time values and it can store date values 01/01/0100 up to 12/31/9999

4. Boolean

Boolean data types hold either a true or false value. These are not stored as numeric values and cannot be used as such. Values are internally stored as -1 (True) and 0 (False) and any non-zero value is considered as true.

5. Variant

Stores any type of data and is the default Visual Basic data type. In Visual Basic if we declare a variable without any data type by default the data type is assigned as default

Reg. No -----

[17CTU302]

KARPAGAM ACADEMY OF HIGHER EDUCATION (Deemed to be University) (Established Under Section 3 of UGC Act 1956) **SECOND INTERNAL EXAMINATION, AUGUST 2018** Third Semester **COMPUTER TECHNOLOGY PROGRAMMING IN VISUAL BASIC** Time : 2 Hours Maximum : 50 Class: II B.Sc. CT Date/Session: 14.8.2018 (AN) PART-A [20 * 1 = 20 Marks]**Answer ALL the questions** 1. is the extension for class module a. .frm b. .cls c. .std d. .doc 2. _____ may include constant, type, variable and DLL procedure declarations. c. variable a. declaration b. procedure d. statement 3. By default VB variables are of the _____ data type. a. const b. static c. variant d. none 4. _____ is the foundation of object oriented programming in VB a. standard b. class c. form d. reference 5. Fixed size array always _____ a. remains same c. increased b. changes d. decreased 6. Size of dynamic array is changed at ______ time a. execution b. run c. break d. stop 7. The statement first executes the statement and then test the condition after each execution a. do..while b. while..do c. select..case d. while 8. ______ structure executes the statements until the condition is satisfied a. do…loop c. do while...loop b. do..loop until d. none 9. Do…loop until is ----- loop a. finite b. infinite c. long d. small 10. _____ function retrieves only date a. for...next b.next...for c. exit for d.exit do 11. _____ are used for storing values temporarily. a. character b. constant d. module c. variable 12. From the following which character is allowed while declaring variables a. % b. @ d. \$ c._ 13. The ______ function returns the length of the string a. strcmp() b. strrev() c. len() d. format() 14. ______ is a group of controls that share the same name and type a. control arrays c. format arrays b. arrays d. index

15.	5. To make all variables in a procedure static,			key	_ keyword is placed at the beginning			
	a.	public	b. private	с.	dim	d. static		
16.	The m	odule-level v	ariable is available to all the p	rocedures	in the mod	dule. They are declared		
	using _	keyw	ords			-		
	a.	public		с.	private an	d public		
	b.	private		d.	private or	public		
17.		is a na	med sequence of statements e	xecuted as	a unit			
	a.	property	b. procedure	c. pr	oject	d. browser		
18.		is th	e extension for the Bitmap file	es				
	a.	.Btp	bBMP	с.	.bmp	ddbs		
19.	Α	loop	can be terminated by an exit	for statem	ent			
	a.	fornext	b. nextfor	c. ez	xit for	d. for exit		
20.	dov	while loop is t	erminated using	statement				
	a.	exit for	b. for exit	с.	exit do	d. do exit		

PART-B

[3 * 2= 6 Marks]

Answer ALL of the following

21. List the I/O statements in visual basic.

22. Write the syntax for Input Box.

23. Write the syntax for Select Case Statement

PART-C

[3 * 8 = 24 Marks]

Answer ALL of the following

24. a. Describe Control and Looping Statements in VB with sample program.

(**OR**)

b. Explain Input Output Statements in VB with relevant example.

25. a. Explain the creation of function in VB 6.0 with sample program.

(**OR**)

b. Explain Message Box and Input Box in Detail.

26. a. Explain the two methods used to create a menu with sample program.

(**OR**)

b. Explain Classes and Objects in detail.

[17CTU302]

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University)

(Established Under Section 3 of UGC Act 1956) SECOND INTERNAL EXAMINATION, AUGUST 2018

Third Semester

COMPUTER TECHNOLOGY

PROGRAMMING IN VISUAL BASIC

Maximum : 50

Time : 2 Hours Class: II B.Sc. CT Date/Session: 14.8.2018 (AN)

PART-A

Answer ALL the questions

1.	is	the extension	on for class n	nodule						
1.	a fr	m	bcls	104410	c. sto	ł		d d	loc	
2	u	ma	v include con	stant type va	riable a	and I	OLL proce	dure dec	larati	ions
2.	a. de	 claration		stant, type, va	C C	vari	iable		iuiui	10115.
	h pro	ocedure			d.	stat	ement			
3	By def	fault VB va	riables are of	the	data ty	vne	ement			
5.		nst	h static		$\frac{1}{C} = \frac{C}{V}$, pc. ariai	nt	d	none	
4	u. co.	is t	he foundation	of object orig	ented n	roors	amming in	VB	none	, ,
	a sta	15 ti undard	h class		c fo	rm		d r	efere	nce
5	Fixed	size arrav a	lways		c. 10			u. 1	cicic	liee
	a.	remains s	ame				c increas	sed		
	b.	changes					d decrea	sed		
6	Size o	f dynamic a	rray is chang	ed at	time		a. accrea	504		
0.	a.	execution	h.	run		C.	break		d	l stop
7.	The	checation	statement fi	rst executes th	ne stater	ment	and then t	est the c	condit	tion after each
	execut	ion							01101	
	a.	dowhile	b.	whiledo		c.	selectcas	e	d.	while
8.		stru	icture execute	es the stateme	nts unti	l the	condition	- is satisf	ied	
	a.	doloop					c. do whi	leloo	n	
	b.	doloop u	ıntil				d. none		r	
9.	Do…l	oop until is	loop							
	a.	finite	b.	infinite		c.	long		d	l. small
10.		func	tion retrieves	only date			U			
	a.	fornext	b.:	nextfor		c.	exit for		d	.exit do
11.		are used fo	r storing valu	les temporaril	y.					
	a.	character	b.	constant	-	c.	variable		d.	module
12.	From	the followir	ng which char	acter is allow	ed whil	le de	claring var	iables		
	a.	%	b. (<u>@</u>		c	U		d. \$	5
13.	The	funct	ion returns th	e length of the	e string	ς				
	a.	strcmp()	b.	strrev()	U	c.	len()		d.	format()
14.		is a	group of con	trols that shar	the sa	ame	name and t	type		**
	a.	control a	rays				c. format	arrays		
	b.	arrays	-				d. index	-		

[20 * 1 = 20 Marks]

15. To ma	ke all variables in a p	keyword is place	keyword is placed at the beginning			
of the	procedure.					
a.	public	b. private	c. dim	d. static		
16. The m	odule-level variable	is available to all the	procedures in the module	e. They are declared		
using	keywords					
a.	public		c. private and p	ublic		
b.	private		d. private or pu	ıblic		
17	is a named sec	juence of statements e	executed as a unit			
a.	property	b. procedure	c. project	d. browser		
18	is the extens	sion for the Bitmap fil	les			
a.	.Btp	bBMP	cbmp	ddbs		
19. A	loop can be	terminated by an exit	for statement			
a.	fornext	b. nextfor	c. exit for	d. for exit		
20. do	while loop is terminat	ed using	statement			
a.	exit for	b. for exit	c. exit do	d. do exit		

PART-B

[3 * 2= 6 Marks]

Answer ALL of the following

21. List the I/O statements in visual basic?

- ➢ Input & Output using text box.
- \succ Input box.
- ➢ Message box.
- > Line continuation character.
- Output using picture box.

22.Write the syntax for Input Box ?

a=InputBox(promt, [Title], [Default], [xpos], [ypos])

23.Write the syntax for Select Case Statement ?

Syntax:

Select Case Index Case 0 Statements Case 1 Statements End Select

Answer ALL of the following

24.a. Describe Control and Looping Statements in VB with sample program.

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.

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 of the If...Then selection

If <condition> Then statement End If

Example:-If average>75 Then txtGrade.Text = "A" End If

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 of the If...Then...Else selection

If <condition > Then statements Else statements End If

Example: If average>50 Then txtGrade.Text = "Pass" Else txtGrade.Text = "Fail" End If

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 Method 1
If < condition 1 > Then statements ElseIf< condition 2 > Then statements ElseIf< condition 3 > Then statements Else Statements End If

Method 2

If < condition 1 > Then statements Else If < condition 2 > Then statements Else If < condition 3 > Then statements Else Statements End If End If EndIf

Example: If average > 75 Then txtGrade.Text = "A" ElseIf average > 65 Then txtGrade.Text = "B" ElseIf average > 55 Then txtGrade.text = "C" ElseIf average > 45 Then txtGrade.Text = "S" Else txtGrade.Text = "F" End If

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 of the Select...Case selection structure Select Case Index Case 0 Statements Case 1 Statements End Select Example: 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

Looping Statements

A repetition structure allows the programmer to that an action is to be repeated until given condition is true.

Do While... Loop Statement

The **Do While...Loop** is used to execute statements until a certain condition is met. The following Do Loop counts from 1 to 100.

Dim number As Integer number = 1 Do While number <= 100 number = number + 1

Loop

A variable number is initialized to 1 and then the Do While Loop starts. First, the condition is tested; if condition is True, then the statements are executed. When it gets to the Loop it goes back to the Do and tests condition again. If condition is False on the first pass, the statements are never executed.

While... Wend Statement

A While...Wend statement behaves like the Do While...Loop statement. The following While...Wend counts from 1 to 100

Dim number As Integer number = 1 While number <=100 number = number + 1 Wend

Do...Loop While Statement

The **Do...Loop** While statement first executes the statements and then test the condition after each execution. The following program block illustrates the structure: Dim number As Long number = 0 Do number = number + 1 Loop While number < 201

The programs 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.

Do Until...Loop Statement

Unlike the **Do While...Loop** and **While...Wend** repetition structures, 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.

An example for **Do Until...Loop** statement. The coding is typed inside the click event of the command button 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.

The For...Next Loop

The **For...Next** Loop is another way to make loops in Visual Basic. **For...Next** repetition structure handles all the details of counter-controlled repetition. The following loop counts the numbers from 1 to 100:

Dim x As Integer For x = 1 To 50 Print x Next In order to count the numbers from 1 yo 50 in steps of 2, the following loop can be used For x = 1 To 50 Step 2 Print x Next The following loop counts numbers as 1, 3, 5, 7..etc

The above coding will display numbers vertically on the form. In order to display numbers horizontally the following method can be used. For x = 1 To 50 Print x & Space\$ (2); Next To increase the space between the numbers increase the value inside the brackets after the & Space^{\$}. Following example is a **For...Next** repetition structure which is with the If condition used. Dim number As Integer For number = 1 To 10If number = 4 Then Print "This is number 4" Else Print number End If Next

In the output instead of number 4 you will get the "This is number 4".

Exit Statement

A For...Next loop condition can be terminated by an Exit For statement. Consider the following statement block.

Dim x As Integer For x = 1 To 10 Print x If x = 5 Then Print "The program exited at x=5" Exit For End If Next The preceding code increments the value of x by 1 until it reaches the condition x = 5. The **Exit** For statement is executed and it terminates the For...Next loop. The Following statement block containing **Do...While** loop is terminated using **Exit Do**statement. Dim x As Integer Do While x < 10Print x x = x + 1If x = 5 Then Print "The program is exited at x=5" Exit Do

End If Loop

With...End With statement

When properties are set for objects or methods are called, a lot of coding is included that acts on the same object. It is easier to read the code by implementing the **With...End With**statement. Multiple properties can be set and multiple methods can be called by using the **With...End With** statement. The code is executed more quickly and efficiently as the object is evaluated only once. The concept can be clearly understood with following example.

With Text1 .Font.Size = 14 .Font.Bold = True .ForeColor = vbRed .Height = 230 .Text = "Hello World" End With In the above coding, the

In the above coding, the object Text1, which is a text box is evaluated only once instead of every associated property or method. This makes the coding simpler and efficient.

(**OR**)

b. Explain Input Output Statements in VB with relevant example.

I/O Statements in VB

Visual Basic provides some excellent ways for input and output operations. Input can be taken using TextBox, InputBox, and output can be shown with the Print method, TextBox, Label, PictureBox and MsgBox.

Input and output using TextBox

The TextBox Control provides an efficient way for both input and operations. The following sample program shows it.

Example:

```
Private Sub Command1_Click()

Dim a As Integer

a = Val(Text1.Text)

a = a + 1

Text2.Text = a

End Sub
```

InputBox:-

InputBox is a function that prompts for user-input. InputBox shows a dialog box that inputs value from the user.

Syntax:

a=InputBox(promt, [Title], [Default], [xpos], [ypos])

where 'a' is a variable to which the value will be assigned. The texts inside the InputBox are optional except the "prompt" text. "prompt" text is the prompt message. "title" is the title of the message box window. "Default" is the default value given by the programmer. 'xpos' and 'ypos' are the geometric positions with respect to x and y axis respectively.

Example:

```
Private Sub cmdTakeInput_Click()
Dim n As Integer
n = InputBox("Enter the value of n : ")
Print n
End Sub
```

The above program prints the value of n taken from the InputBox function.

MsgBox:-

The MsgBox function shows a dialog box displaying the output value or a message.

Syntax:

MsgBox Prompt, [MsgBox style], [Title]

where Promt text is the prompt message, [MsgBox Style] is the msgbox style constants and [Title] is the text displayed in the title bar of the MsgBox dialog.

Example:

```
Private Sub cmdShowMessage_Click()
Dim n As Integer
n = 10
MsgBox "The value of n is " & n
End Sub
```

Example: MsgBox with the dialog type and title text. Private Sub cmdShowMessage_Click() Dim n As Integer n = 10 MsgBox "The value of n is ", vbInformation, "Result" & n End Sub

The MsgBox dialog box appears with the 'vbInformation' dialog type and the title text is "Result".

Line continuation character(_):-

In your vb program you can type maximum of 1023 characters in a line. And sometimes, it doesn't fit in the window when we write so many characters in a line. So Line Continuation Character (underscore preceded by a space) is used to continue the same statement in the next line.

Example:

Private Sub cmdShow_Click() MsgBox "Hello, welcome to www.vbtutes.com", _ vbInformation, "Welcome" End Sub

Output using PictureBox

PictureBox is also used for the output operation. The output is displayed using the Print method.

Example:

```
Private Sub cmdDisplay_Click()
Dim n As Integer
n = 40
picResult.Print n + 5
End Sub
```

The program prints 45 in the PictureBox.

25. a. Explain the creation of function in VB 6.0 with sample program.

Functions

Functions are like sub procedures, except they return a value to the calling procedure. They are especially useful for taking one or more pieces of data, called *arguments* and performing some tasks with them. Then the functions returns a value that indicates the results of the tasks complete within the function.

The following function procedure calculates the third side or hypotenuse of a right triangle, where A and B are the other two sides. It takes two arguments A and B (of data type Double) and finally returns the results.

Function Hypotenuse (A As Double, B As Double) As Double Hypotenuse = sqr $(A^2 + B^2)$ End Function

The above function procedure is written in the general declarations section of the Code window. A function can also be written by selecting the *Add Procedure* dialog box from the Tools menu and by choosing the required scope and type. EXAMPLE:-

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)

(OR) b.Explain Message Box and Input Box in Detail?

MESSAGE BOXES:-

End Sub

Message boxes used to display the information. Message boxes should be used for short messages or to give transient feedback. It can hold 1024 characters. Message boxes can be used in either two ways, depending on your needs. You can use for display information and also for decision making from the user. In either case the MsgBox Function is used by the following syntax:

MsgBox(prompt[, buttons] [, title] [, helpfile, context])

The MsgBox function syntax has these parts:

Part	Description
prompt	Required. String expression displayed as the message in the dialog box.
buttons	Optional. Numeric expression that is the sum of values specifying the number
	and type of buttons to display, the icon style to use, the identity of the default
	button, and the modality of the message box. If omitted, the default value for
	buttons is 0 (which causes only an OK button to be displayed with no icon).
title	Optional. String expression displayed in the title bar of the dialog box. If you
	omit <i>title</i> , the application name is placed in the title bar.
<i>helpfile</i> and	Both optional. These arguments are only applicable when a Help file has been
context	set up to work with the application.

🖻 Project1 - Form1 (Form) 🔤 🗖 🔀	
► Form1	
Input form	٦
Command1	
Project1 - Form1 (Code)	
Command1 Click V	
Command1 Click Private Sub Command1_Click() MsgBox "Welcome to Visual Basic "	
Commandi V Click V Private Sub Commandi_Click() MsgBox "Welcome to Visual Basic" End Sub	
Commandi V Click Private Sub Commandi_Click() MsgBox "Welcome to Visual Basic" End Sub = I I	
Commandi V Click Private Sub Commandi_Click() MsgBox "Welcome to Visual Basic" End Sub Code window	

Figure3.1 Output Form

🛢 Form1	
Comr	nand1Click this button
	Project1
	Welcome to Visual Basic
	ОК

In the running screen click on the command1 button you get an Msgbox named "Welcome to Visual Basic" followed by **OK** button. Now, Click on the **OK** button you will be back to Visual Basic screen.

Eg.2 :-

MsgBox "The Last Name field must not be blank.", _ VbExclamation, _ "Last Name" → causes the following box to be displayed:



<u>The buttons argument :-</u> The Buttons displayed in a message here

Button Layout	Value	Short Description
vbOKonly	0	Displays the OK button.
vbOKCancel	1	Displays the ok and cancel button.
vbAbortRetryIgnore	2	Displays the Abort , Retry , Ignore
vbYesNoCancel	3	Displays Yes, No and Cancel button
vbYesNo	4	Displays the Yes / No button
vbRetryCancel	5	Displays the retry and Cancel buttons.

The Icons displayed in the message box are here

Icon on message	Value	Short Description
vbCritical	16	Displays critical message icon
vbQuestion	32	Displays question icon
vbExclamation	48	Displays exclamation icon
vbInformation	64	Displays information icon

The Default button displayed in a message form

Default Button	Value	Short Description
vbDefaultButton1	0	Button 1 is default
vbDefaultButton2	256	Button 2 is default
vbDefaultButton3	512	Button 3 is default

Msgbox Return Value

Return Value	Value	Short Description
vbOk	1	The User Clicked OK
vbCancel	2	The User Clicked Cancel
vbAbort	3	The User Clicked Abort
vbRetry	4	The User Clicked Retry
vbIgnore	5	The User Clicked Ignore
VbYes	6	The User Clicked Yes
VbNo	7	The User Clicked No

InputBox:-

InputBox is a function that prompts for user-input. InputBox shows a dialog box that inputs value from the user.

Syntax:

```
a=InputBox( promt, [Title], [Default], [xpos], [ypos])
```

where 'a' is a variable to which the value will be assigned. The texts inside the InputBox are optional except the "prompt" text. "prompt" text is the prompt message. "title" is the title of the message box window. "Default" is the default value given by the programmer. 'xpos' and 'ypos' are the geometric positions with respect to x and y axis respectively.

Example:

```
Private Sub cmdTakeInput_Click()
Dim n As Integer
n = InputBox("Enter the value of n : ")
Print n
End Sub
```

The above program prints the value of n taken from the InputBox function.

Example: InputBox with the title text and default value. Private Sub cmdTakeInput_Click() Dim n As Integer n = InputBox("Enter the value of n : ", "Input", 5) Print n End Sub

The InputBox dialog appears with the title "Input" and the highlighted default value in the provided text field is 5.

26. a. Explain the two methods used to create a menu with sample program?

The menu bar is the standard feature of most Windows applications. The main purpose of the menus is for easy navigation and control of an application. Some of the most common menu items are File, Edit, View, Tools, Help and more. Each item on the main menu bar also provides a list of options in the form of a pull-down menu. When you create a Visual Basic 6 program, you need not include as many menu items as a full-fledged Windows application. What you need is to include those menu items that can improve the ease of usage by the user. There are two ways to add menus to your application, using the Visual Basic's Application Wizard and or the menu editor.

Adding Menu Bar Using Visual Basic's Application Wizard

The easiest way to add a menu bar to your application is by using Visual Basic's Application Wizard. This wizard allows the user to insert fully customized standard Windows menu into his or her application. To start using Visual Basic's Application Wizard, click on the Application Wizard icon at the Visual Basic new project dialog box, as shown in Figure 37.1 below:



New Project Window

When you click on the VB Application wizard, the introduction dialog box will appear, as shown in Figure as below. As you are not loading any default setting, just click on the Next button.

Application Wizard - Intro	oduction 📃 🔀
	The Application Wizard will help you create a new Visual Basic Application. You can press Back at any time to change your selections. Please click Next to begin. From what profile do you want to load your settings? (None)
Help	Cancel < Back Next > Einish

After clicking the Next button, the interface type dialog box will be displayed, as shown in Figure as below.. There are three choices of interface available for your project. As we currently not creating a Multiple Document Interface (MDI), we choose Single Document Interface (SDI).

Application Wizard - Interface Type				
	What type of interface would you like for your application?			
Hint! All windows exist at the same	C Multiple Document Interface (MDI)			
level.	Single Document Interface (SDI)			
	C Explorer Style			
	<u>W</u> hat name do you want for the application? MyFirstMenu			
Help	Cancel < <u>B</u> ack <u>N</u> ext > <u>F</u> inish			

Clicking the Next button will bring up a list of menus and submenus that you can add them to your application. Check to select a menu item and uncheck to unselect a menu item. We choose all the menus and click next, then you will get an interface comprises File, Edit, View and Help

Figure

menus,	as	shov	vn	in
Application Wizard	- Menus			×
	Select the application You can a the applic	e Menus and Sub Me n? Ilways use the Men ation is created.	enus you would like u Editor to modify th	in your ne menus after
Menus	<u>S</u> ub	Menus		
✓ &File	↔ 	&New	<u>^</u> +	
✓ &Edit ✓ &View	×	&Open &Close	= ×	
&Tools		[Separator]		
&Window		&Save Save &As	<u>Ť</u>	
	€	Save A&l	÷	
I		[Separator]	-	<u>R</u> eset
Help	Cancel	< <u>B</u> ack	Next >	<u>F</u> inish

When you click on any menu item, a list of drop-down submenu items will be displayed. For example, if you click on the File menu, the list of submenu items such as New, Open, Save, Save

63. P	roject1		
<u>F</u> ile	<u>E</u> dit <u>V</u> iew	<u>H</u> elp	
	New Open Close	Ctrl+N	3 <u>7 U</u>
	Save Save As Save All		
	Properties		
	Page Setup Print Preview Print		08:28 pm
	Send		
	Exit		

Clicking on any of the dropped down menu item will show the code associated with it, and this is where you can modify the code to suit your programming needs. For example, clicking on the item Open will reveal the following code:

```
😓 Project1 - frmMain (Code)
 mnuFileOpen
    Private Sub mnuFileOpen Click()
        Dim sFile As String
        With dlgCommonDialog
             .DialogTitle = "Open"
             .CancelError = False
             'ToDo: set the flags and attributes of the con
             .Filter = "All Files (*.*) |*.*"
             .ShowOpen
             If Len(.FileName) = 0 Then
                 Exit Sub
             End If
             sFile = .FileName
        End With
         'ToDo: add code to process the opened file
   End Sub
Next add the following lines so that the user can open graphic files of different formats.
```

.Filter = "Bitmaps(*.BMP)|*.BMP|Metafiles(*.WMF)|*. WMF|Jpeg Files(*.jpg)|*.jpg|GIF Files(*.gif)|*.gif|Icon Files(*.ico)|*.ico|All Files(*.*)|*.*". Then, you need to load the image into the Image box with the following code:

Image1.Picture = LoadPicture(.FileName)

Also set the Stretch property of the Image box to true so that the image loaded can resize by itself. Please note that each menu item is a special control, so it has a name too. The name for the menu File in this example is mnuFileOpen.

Private Sub mnuFileOpen_Click() Dim sFile As String With dlgCommonDialog .DialogTitle = "Open" .CancelError = False 'Set the flags and attributes of the common dialog control .Filter = "Bitmaps(*.BMP)|*.BMP|Metafiles(*.WMF)|*. WMF|Jpeg Files(*.jpg)|*.jpg|GIF Files(*.gif)|*.gif|Icon Files(*.ico)|*.ico|All Files(*.*)|*.*" .ShowOpen Image1.Picture = LoadPicture(.FileName) If Len(.FileName) = 0 Then Exit Sub End If sFile = .FileName End With End Sub



Clicking on the particular picture will load it into the image box, as shown in Figure 36.10 below



Adding Menu Bar Using Menu Editor

To start adding menu items to your application, open an existing project or start a new project, then click on Tools in the menu bar of the Visual Basic IDE and select Menu Editor. When you click on the Menu Editor, the Menu Editor dialog will appear. In the Menu Editor dialog, key in the first item File in the caption text box. You can use the ampersand (&) sign in front of F so that F will be underlined when it appears in the menu, and F will become the hot key to initiate the action under this item by pressing the Alt key and the letter F. After typing &File in the Caption text box, move to the name textbox to enter the name for this menu item, you can type in mnuFile here. Now, click the Next button and the menu item &File will move into the empty space.

Menu Editor	×
Caption: 8File	ОК
Name: mnuFile	Cancel
Inde <u>x</u> : Shortcut: (None)	-
HelpContextID: 0 NegotiatePosition:	0 - None 💌
□ Checked □ Enabled □ Visible □	<u>W</u> indowList
← → ↑ ↓ <u>N</u> ext <u>I</u> nsert	Dele <u>t</u> e
8File	
1	

You can then add in other menu items on the menu bar by following the same procedure, as shown in Figure 37.12 below:

Menu Editor
Caption: SFile OK
Name: mnuFile Cancel
Inde <u>x</u> : Shortcut: (None)
HelpContextID: 0 NegotiatePosition: 0 - None 💌
□ <u>C</u> hecked □ <u>E</u> nabled □ <u>V</u> isible □ <u>W</u> indowList
← → ↑ ↓ <u>N</u> ext Insert Dele <u>t</u> e
&Edit V&iew Hel&p

When you click Ok, the menu items will be shown on the menu bar of the form.

		C	3.	F	0	rn	n1	L																				C		2	וכ		•))(Σ	3	
		F	il	e		E	d	it		1	/ <u>i</u>	ev	v		ŀ	łe	l	<u>p</u>																				
	•	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:
	•	·	•	·	·	•	·	·	•	·	•	·	·	·	·	·	•	•	·	·	·	·	•	·	·	·	·	·	·	•	·	•	·	•	•	·	·	•
	1	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:
	•	·	•	·	•	•	·	•	•	·	•	·	•	•	•	·	•	•	•	•	•	•	•	•	·	·	•	·	·	•	•	•	•	•	•	·	·	•
-		:	:	2	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	2	:	2	:	:	:	2	:
		·	•	·	•	•	·	•	•	·	•	·	•	•	•	·	•	•	•	•	•	•	•	•	·	·	•	·	·	•	•	•	•	•	•	•	•	•
	•	•	:	:	:	:	•	:	:	:	:	•	:	:	:	:	:	:	:	:	:	:	:	:	•	•	:	:	:	:	•	:	•	:	:	•	•	:
																							:					:		:		:		:	:			
	•	·	·	·	·	·	·	·	·	•	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	•	·	·	·	·	·	·	·	·	•
	1	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:
	·	·	•	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	•	·	·	·	·	·	·	•	·	•	·	•	•	·	·	•
		:			:			:					:	:	:	:			:	:	:	:	:	:			:	:	:	:	1	:	1	:	:	:	:	:
	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
		·		·		ċ				·	ċ					i		ċ																				

Now, you may proceed to add the sub menus. In the Menu Editor, click on the Insert button between File and Exit and then click the right arrow key, and the dotted line will appear. This shows the second level of the menu, or the submenu. Now key in the caption and the name. Repeat the same procedure to add other submenu items. Here, we are adding New, Open, Save, Save As and Exit.

Menu Editor	X
Caption: Exit	ОК
Name: mnuExit	Cancel
Inde <u>x</u> : Shortcut: (None)	•
HelpContextID: 0 NegotiatePosition:	0 - None 💌
Checked 🔽 Enabled 🔽 Visible	<u>W</u> indowList
← → ↑ ↓ <u>N</u> ext <u>I</u> nsert	Dele <u>t</u> e
&File ····New ····Open ····Save	
····Save As	
&Edit V&iew Hel&p	

Now click the OK button and go back to your form. You can see the dropped down submenus when you click on the item File, as shown.

C3. Fo	orm1		
<u>F</u> ile	<u>E</u> dit V <u>i</u> ew He	el <u>p</u>	
	New Open Save		
	Save As Exit		

(**OR**)

b.Explain Classes and Objects in detail.

CREATING CLASSES AND OBJECT

An *object* is an encapsulation of *data*, and *methods* (procedures) that act on the data. The details of the data structures used and the implementation of the methods are hidden inside the object. All the programmer needs to know is what tasks the object can perform, and the parameters required by these tasks.

Control objects are predefined objects that can be created using the Visual Basic toolbox, such as text boxes, command buttons and other controls. No matter how many text boxes you create on a

form, they all have the same properties and methods. Essentially, a *class* is a template from which an object is created, and specifies the properties and methods that will be common to all objects that are created from it. Each text box is therefore an*instance* of the class *TextBox*, for which you can set the properties or invoke the methods.

The value of a property is manipulated by two procedures, one to set the value (*Let*) and the other to retrieve the value (*Get*). These procedures are sometimes referred to asaccessor methods.

Code objects must be created by the programmer, and are instances of user-defined types that are defined in a separate *class module*. Like control objects, they have properties and methods and can respond to events. The set of properties, methods, and events for a class is called the *class interface*. Two (or more) different classes may have the same interface, and carry out identical tasks, even though they carry out the task differently. The occurrence of two or more classes that have behaviors with the same name and functionality but different implementations is called *polymorphism*.

Events are declared in the general declarations section of class module with a statement of the form:

Public Event UserDefinedEvent (arg1, arg2, ...)

and triggered with a *RaiseEvent* statement. The statement that declares the object in the form's code must include the keyword *WithEvents* in order for the events coming from the object to be processed.

When Visual Basic is an *object-oriented* programming language. Applications communicate with objects through the properties and methods they *expose*, and the events they *raise*. When you program with Visual Basic, you are programming objects, such as the controls you place on a form, by manipulating their properties and calling their methods. The action takes place within the object's *event procedures*, which are executed in response to the occurrence of some event. Events are said to be raised (in other words they are considered to have occurred) when certain conditions are met.



Figure 3.5

EXAMPLE

Open a new Visual Basic project, save it to a new folder, and create a form similar to the one illustrated below.

Students			
	Create Stud	lent Record	
Student Name			
Student Number		Student Details	
	Enter Details		
	Display Details		
	Exit		

The "frmStudents" form

Properties	for "frmStud	ents" form	
Control	Name	Caption	Additional Properties
Form	frmStudents	Students	
Label	lblTitle	Create Student Record	Alignment = $2 - Center$
Label	lblName	Name	Alignment = 1 - Right Justify
TextBox	txtName		
Label	lblNumber	Number	Alignment = 1 - Right Justify
TextBox	txtNumber		
Button	cmdEnter	Enter Details	
Button	cmdDisplay	Display Details	
PicturePox	picStudent		
_			

Button cmdQuit Exit

CREATING THE STUDENT CLASS

It is a convention to precede the name of a class with the letter 'C'. We will therefore use the name *CStudent* . Use the following steps to create the *CStudent* class:

- 1. From the **Project** menu, select **Add Class Module** .
- 2. Double-click on **Class Module** in the **Add Class Module** dialog box.
- 3. If the **Properties** window is not visible, press **F4** to display it. (Note that the class has the default name *Class1*.)
- 4. Change the setting of the **Name** property to *CStudent*.
- 5. Type the following lines into the code module: Private mName As String
 Private mNumber As String
 These are the *member variables* used to hold data.
 The word *Private* means they cannot be accessed directly.
- 6. From the **Tools** menu, click on **Add Procedure** .
- Type *Name* into the Name text box, select the Property radio button in theType box, and click on OK. The following code will appear in the class module window:Public Property Get Name() As String End Property

Public Property Let Name(ByVal vNewValue As Variant) End Property

8. Change the word *Variant* in both procedures to *String*, and the word*vNewValue* in the second procedure to *vName*, and add code to the procedures as shown

below:Public Property Get Name() As String

Name = mName End Property

Public Property Let Name(ByVal vName As Variant) mName = vName End Property

The first procedure retrieves the value of the variable *mName*

The second procedure assigns a value to mName

9. Create the procedures that will be used to retrieve and assign values to the variable *mNumber* as follows:Public Property Get Number() As String

Number = mNumber End Property

Public Property Let Number(ByVal vNumber As Variant) mNumber = vNumber

End Property

10. From the **File** menu, click on **Save CStudent As** and save the class module with the name *CStudents.cls*. The *.cls* extension is used for files holding class modules.

USING THE STUDENT CLASS

We can now write a program that creates an object that is an instance of the *CStudent* class, and uses the *Property Let* and *Property Get* procedures to assign values to and retrieve values from the member variables. Enter the following code in the form's code window:

Private Student As CStudent

```
Private Sub Form_Load()
  Set Student = New CStudent
End Sub
Private Sub cmdEnter_Click()
  Student.Name = txtName
  Student.Number = txtNumber
  txtName.Text = ""
  txtNumber.Text = ""
End Sub
Private Sub cmdDisplay_Click()
  picStudent.Cls
  picStudent.Print "Student Name:"
  picStudent.Print Student.Name
  picStudent.Print
  picStudent.Print "Student Number:"
  picStudent.Print Student.Number
End Sub
```

Private Sub cmdQuit_Click()



Run the program and enter a student name and number, press the *Enter Details* button to send the data to the object, and press the *Display Details* button to retrieve the details from the object and display the student's name and number.

THE INITIALIZE EVENT PROCEDURE

The **Object** drop-down combo box in a class module window displays two items, *General* and *Class* (see below).

18	Students - CStudent (Code)	
	(General)	•
	General)	-
	Class	-
	Public Property Get Name() As String	
	Name = mName	
	End Property	
	Public Property Let Name (ByVal vName As String)	
	mName = vName	
	End Property	
	Public Property Get Number() As String	
	Number = mNumber	
	End Property	_
	Public Property Let Number (ByVal vNumber As String)	
	mNumber = vNumber	
	End Property	
=		

The class module window When you click on *Class*, the following template appears: Private Sub Class_Initialize()

End Sub

This procedure is automatically invoked when an object is created from the class, and can be used to set default values for member variables, or to create other objects associated with this object. The counterpart to *Initialize* is the *Terminate* event procedure which has a similar format, and is automatically invoked when all references to the object are set to*Nothing*, or when the object falls out of scope (an object falls out of scope, remember, when the procedure that created it is exited). This procedure can be used to set any objects you may have created using the class module to *Nothing*.

CLASS RELATIONSHIPS

Three relationships can exist between classes:

- *Use* one class *uses* another class if it manipulates or creates objects of that class. In general, class A uses class B if an object of class B is sent a message by a property or method of class A, or a method or property of class A returns, receives, or creates objects of class B.
- *Containment* class A *contains* class B when a member variable of class A has class B as its type.
- *Inheritance* this is the process by which one class (the *child* class), inherits the properties, methods, and events of another class (the *parent* class). Note that Visual Basic does not support the strict academic definition of inheritance.

Reg. No -----

[17CTU302]

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University)

(Established Under Section 3 of UGC Act 1956)

THIRD INTERNAL EXAMINATION, OCTOBER 2018

Third Semester

COMPUTER TECHNOLOGY

PROGRAMMING WITH VISUAL BASIC

Time : 2 Hours Class: II B.Sc. CT Date/Session: 8.10.2018 (AN)

PART-A

Answer ALL the questions

1.	is us	ed for opening mar	ny windows at the same ti	me
	a.MDI	b.IDM	c.DIM	d.SDI
2.	All the document	nt windows are cor	ntained in a	
	a.Child	b. parent	c.both	d.mdi
3.	VB application	can have only	MDI form.	
	a.1	b.2	c.3	d.4
4.	Each time the up	ser clicks new fron	n the file menu, a new	window is created and
	displayed.			
	a.Parent	b.form	c.standard	d.child
5.	binds data	-aware controls to	microsoft access and othe	er ODBC data sources.
	a.DAO	b.ADO	c.Data control	d.Data
6.	is an	n API call interface	e to the open DataBase con	nnectivity libraries and drivers to
	provide data acc	cess to Oracle	-	-
	a.RDO	b.RDC	c.ODBC	d.VBSQL
7.	The control	which is a part of a	an option group allows the	e user to select one option even if
	it displays multi	ple choices.		-
	a check box	h comm	and button contion bu	then ditimen
	u.eneen oon	0.0011111	and button c.option bu	uon a.umer
8.	allows th	e user to select an	item from the dropdown l	ist box to type a selection in the
8.	allows th text box.	e user to select an	item from the dropdown l	ist box to type a selection in the
8.	allows th text box. a.combo box	e user to select an b.command but	item from the dropdown l	ist box to type a selection in the d.none
8. 9.	allows th text box. a.combo box To create a child	e user to select an b.command but d form, set its MDl	item from the dropdown l ton c.check box I child property to	d.timer ist box to type a selection in the d.none
8. 9.	allows the text box. a.combo box To create a child a.TRUE	e user to select an b.command but d form, set its MDI b.FALSE	ton c.check box I child property to c.0 d.1	ist box to type a selection in the d.none
 8. 9. 10. 	allows th text box. a.combo box To create a child a.TRUE A data access of	e user to select an b.command but d form, set its MDI b.FALSE bject is a collection	ton c.check box I child property to c.0 d.1 n of classes that	d.none at model the structure of a
8. 9. 10.	allows th text box. a.combo box To create a child a.TRUE A data access ol relational databa	b.command but b.command but d form, set its MDI b.FALSE bject is a collection ase system.	ton c.check box I child property to c.0 d.1 n of classes that	d.umer ist box to type a selection in the d.none at model the structure of a
8. 9. 10.	allows the text box. a.combo box To create a child a.TRUE A data access of relational databa a.Module	e user to select an b.command but d form, set its MDI b.FALSE bject is a collectior ase system. b.procedure	ton c.check box I child property to c.0 d.1 n of classes that c.object	d.none at model the structure of a d.program
 8. 9. 10. 11. 	allows th text box. a.combo box To create a child a.TRUE A data access of relational databa a.Module	e user to select an b.command but d form, set its MDI b.FALSE bject is a collection ase system. b.procedure ox allows the user to	item from the dropdown l ton c.check box I child property to c.0 d.1 n of classes that c.object o select directories and pa	d.umer ist box to type a selection in the d.none d.none at model the structure of a d.program ths which are displayed
 8. 9. 10. 11. 	allows the text box. a.combo box To create a child a.TRUE A data access of relational databa a.Module	e user to select an b.command but d form, set its MDI b.FALSE bject is a collection ase system. b.procedure w allows the user to b.list	item from the dropdown l ton c.check box I child property to c.0 d.1 n of classes tha c.object o select directories and pa c.tool	 a.umer ist box to type a selection in the d.none at model the structure of a d.program aths which are displayed d.Dir tool
 8. 9. 10. 11. 12. 	allows the text box. a.combo box To create a child a.TRUE A data access of relational databa a.Module Bo a.Dir list . The r	b.command but b.command but d form, set its MDI b.FALSE bject is a collection ase system. b.procedure ox allows the user t b.list method moves to th	item from the dropdown l ton c.check box I child property to c.0 d.1 n of classes tha c.object o select directories and pa c.tool ne first row in the record s	 at model the structure of a d.program at which are displayed d.Dir tool et.
8.9.10.11.12.	allows th text box. a.combo box To create a child a.TRUE A data access of relational databa a.Module Bo a.Dir list . The r a.move first	e user to select an b.command but d form, set its MDI b.FALSE bject is a collection ase system. b.procedure ox allows the user t b.list nethod moves to th b.move next	item from the dropdown l ton c.check box I child property to c.0 d.1 n of classes tha c.object o select directories and pa c.tool ne first row in the record s c.move previous	 at model the structure of a d.program at which are displayed d.Dir tool et. d.move last
 8. 9. 10. 11. 12. 13. 	allows th text box. a.combo box To create a child a.TRUE A data access of relational databa a.Module Bo a.Dir list The r a.move first	b.command but b.command but d form, set its MDI b.FALSE bject is a collection ase system. b.procedure ox allows the user t b.list nethod moves to th b.move next thod moves to the 1	item from the dropdown l ton c.check box I child property to c.0 d.1 n of classes that c.object o select directories and pa c.tool ne first row in the record s c.move previous last row in the record set.	d.umer ist box to type a selection in the d.none d.none at model the structure of a d.program aths which are displayed d.Dir tool et. d.move last
 8. 9. 10. 11. 12. 13. 	allows the text box. a.combo box To create a child a.TRUE A data access of relational databas a.Module Bo a.Dir list The r a.move first The me a.move first	b.command but b.command but d form, set its MDI b.FALSE bject is a collection ase system. b.procedure ox allows the user t b.list method moves to the b.move next thod moves to the b. b.move next	item from the dropdown l ton c.check box I child property to c.0 d.1 n of classes tha c.object o select directories and pa c.tool ne first row in the record s c.move previous last row in the record set. c.move previous	 at model the structure of a d.program at which are displayed d.Dir tool et. d.move last d.move last
 8. 9. 10. 11. 12. 13. 14. 	allows th text box. a.combo box To create a child a.TRUE A data access ol relational databa a.Module Bo a.Dir list The r a.move first . The me a.move first	e user to select an b.command but d form, set its MDI b.FALSE bject is a collection ase system. b.procedure ix allows the user t b.list nethod moves to th b.move next thod moves to the 1 b.move next ox displays the val	item from the dropdown l item from the dropdown l ton c.check box I child property to c.0 d.1 n of classes that c.object o select directories and pa c.tool ne first row in the record s c.move previous last row in the record set. c.move previous id disk drives and allows	 at model the structure of a d.program at model the structure of a d.program at which are displayed d.Dir tool et. d.move last d.move last the use to select one of them

Maximum Marks : 50

[20 * 1 = 20 Marks]

15. box displays a set of files from which a user can select the desired one a.Drive list box b.file list box c.picture box d.Dir list box 16. communicates with data sources through the JET database engine b.data control c.container d.relations a.Data Access Objects helps to move and resize the controls and forms 17. ____ a. label b. shape c. OLE d. pointer carries out the specified action when the user choose it 18. a.option button b.image control c.shape control d.command button 19. is the control used to display messages and enter text a.tool box c.command button b.text box d.option button bar control is a frame that can consist of several panels, which informs the 20.A user about the status of an application. b.status bar a.control bar c.image bar d.picture bar

PART-B

[3 * 2= 6 Marks]

Answer ALL of the following

21. Write down the operations of Combo Box.

22. Name any five properties of Timer Control

23. Write any two features of MDI

PART-C

[3 * 8 = 24 Marks]

Answer ALL of the following

24. a. Explain the following controls with example : (i) CheckBox (ii) CommandButton

(**OR**)

b. Create an application Having 3 horizontal scrollbars-Red, Green and Blue. The maximum value of any scrollbar should be 255. The application should also contain a picture box, whenever the value of any scrollbar is changed, the color corresponding to the combination of three values of 3 scrollbars should be applied to the Picturebox's background.

25. a. Explain line and shape controls with a sample program.

(**OR**)

b.Describe Multiple Document Interface in detail.

26. a.Write a Program to print the Students grade with database Connectivity.

(**OR**)

b. Write a Program to print the Employee Payroll with database Connectivity.

[17CTU302]

KARPAGAM ACADEMY OF HIGHER EDUCATION

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

THIRD INTERNAL EXAMINATION, AUGUST 2018

Third Semester COMPUTER TECHNOLOGY

PROGRAMMING IN VISUAL BASIC

Time : 2 Hours Class: II B.Sc. CT Date/Session:

PART-A

[20 * 1 = 20 Marks]

Maximum : 50

Answer ALL the questions

1.	is used	l for opening many wir	ndows at the same time	
	a.MDI	b. IDM	c.DIM	d.SDI
2.	All the document	windows are contained	d in a	
	a.Child	b. parent	c.both	d.mdi
3.	VB application ca	in have only M	IDI form.	
	a.1	b. 2	c.3	d.4
4.	Each time the use	r clicks new from the f	ile menu, a new	- window is created and
	displayed.			
	a.Parent	b.form	c.standard	d. child
5.	binds data-a	ware controls to micro	soft access and other C	DBC data sources.
	a.DAO	b.ADO	c. Data control	d. Data
6.	is an A	API call interface to the	e open DataBase conne	ectivity libraries and drivers to
	provide data acce	ss to Oracle		
	a.RDO	b.RDC	c.ODBC d.VBS	QL
7.	The control w	hich is a part of an opt	ion group allows the u	ser to select one option even if
	it displays multipl	le choices.		
	a.check box	b.command b	utton c. option butt	on d. timer
8.	allows the	user to select an item f	rom the dropdown list	box to type a selection in the
	text box.			
	a.combo box	b. command button	c.check box	d.none
9.	To create a child	form, set its MDI child	property to	
	a.TRUE	b.FALSE	c.0 d.1	
10.	A data access obj	act is a collection of	1 .1 .	
	A data access obj	ect is a confection of	classes that r	nodel the structure of a
	relational databas	e system.	classes that r	nodel the structure of a
	relational databas a.Module	e system. b.procedure	classes that r	d. program
11.	relational databas a.Module Box	e system. b.procedure allows the user to sele	classes that r c. object ct directories and paths	nodel the structure of a d. program s which are displayed
11.	relational databas a.Module a.Dir list	e system. b.procedure allows the user to select b. list	classes that r c. object ct directories and paths c.tool	nodel the structure of a d. program s which are displayed d.Dir tool
11. 12.	a.Module a.Dir list	e system. b.procedure allows the user to select b. list ethod moves to the first	classes that r c. object ct directories and paths c.tool t row in the record set.	nodel the structure of a d. program which are displayed d.Dir tool
11. 12.	relational databas a.Module a.Dir list The me a.move first	 e system. b.procedure allows the user to selection b.list ethod moves to the first b.move next 	classes that r c. object ct directories and paths c.tool t row in the record set. c.move previous	nodel the structure of a d. program s which are displayed d.Dir tool d.move last
11. 12. 13.	a.Module Box a.Dir list The meth	e system. b.procedure allows the user to select b. list ethod moves to the first b. move next od moves to the last ro	classes that r c. object ct directories and paths c.tool t row in the record set. c.move previous w in the record set.	nodel the structure of a d. program s which are displayed d.Dir tool d.move last

14. _____ box displays the valid disk drives and allows the use to select one of them b.Drive list a.Dirlist **c.**list Drive d.none 15. box displays a set of files from which a user can select the desired one a.Drive list box b.file list box **c.**picture box d.Dir list box communicates with data sources through the JET database engine 16. _ a.Data Access Objects b.data control c.container d.relations helps to move and resize the controls and forms 17. a. label c. OLE b. shape d. pointer ____ carries out the specified action when the user choose it 18.____ a.option button b.image control c.shape control d.command button 19._____ is the control used to display messages and enter text a.tool box b.text box **c.**command button d.option button bar control is a frame that can consist of several panels, which informs the 20.A user about the status of an application.

a.control bar b.**status bar c.**image bar d.picture bar

PART-B

[3 * 2= 6 Marks]

Answer ALL of the following

21. Write down the operations of Combo Box.

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.

22. Name any five properties of Timer Control

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

23. Write any two features of MDI

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

PART-C

[3 * 8 = 24 Marks]

Answer ALL of the following

24. a. Explain the following controls with example :

(i) CheckBox

The CheckBox control is similar to the option button, except that a list of choices can be made using check boxes where you cannot choose more than one selection using an OptionButton. By ticking the CheckBox the value is set to True. This control can also be grayed when the state of the CheckBox is unavailable, but you must manage that state through code.

When you place a CheckBox control on a form, all you have to do, usually, is set its Caption property to a descriptive string. You might sometimes want to move the little check box to the right of its caption, which you do by setting the Alignment property to 1-Right Justify, but in most cases the default setting is OK. If you want to display the control in a checked state, you set its Value property to 1-Checked right in the Properties window, and you set a grayed state with 2-Grayed.

The only important event for CheckBox controls is the Click event, which fires when either the user or the code changes the state of the control. In many cases, you don't need to write code to handle this event. Instead, you just query the control's Value property when your code needs to process user choices. You usually write code in a CheckBox control's Click event when it affects the state of other controls. For example, if the user clears a check box, you might need to disable one or more controls on the form and reenable them when the user clicks on the check box again. This is how you usually do it (here I grouped all the relevant controls in one frame named Frame1):

Private Sub Check1_Click() Frame1.Enabled = (Check1.Value = vbChecked) End Sub

Note that Value is the default property for CheckBox controls, so you can omit it in code. I suggest that you not do that, however, because it would reduce the readability of your code.

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
	Name	frmCheckBox
CheckBox	Caption	Bold
	Name	chkBold
ChackBoy	Caption	Italic
CHECKDOX	Name	chkItalic
ChaokBoy	Caption	Underline
CHECKDOX	Name	chkUnderline

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 (ii) CommandButton

Using CommandButton 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 DblClick 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.

- \Box 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.

 \Box 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
```

(**OR**)

b.Create an application Having 3 horizontal scrollbars-Red, Green and Blue. The maximum value of any scrollbar should be 255. The application should also contain a picture box, whenever the value of any scrollbar is changed, the color corresponding to the combination of three values of 3 scrollbars should be applied to the Picturebox's background.

There are two key events for scrollbar controls: the Change event fires when you click on the scroll bar arrows or when you drag the indicator; the Scroll event fires while you drag the indicator. The reason for these two distinct possibilities is mostly historical. First versions of Visual Basic supported only the Change event, and when developers realized that it wasn't possible to have continuous feedback when users dragged the indicator, Microsoft engineers added a new event instead of extending the Change event. In this way, old applications could be recompiled without unexpected changes in their behavior. At any rate, this means that you must often trap two distinct events:

'Show the current scroll bar's value. Private VScroll1_Change() Label1.Caption = VScroll1.Value End Sub Private VScroll1_Scroll() Label1.Caption = VScroll1.Value End Sub

The example shown in the following figure uses three VScrollBar controls as sliders to control the individual RGB (red, green, blue) components of a color. The three scroll bars have their Min property set to 255 and their Max property set to 0, while their SmallChange is 1 and LargeChange is 16. This example is also a moderately useful program in itself because you can select a color and then copy its numeric value to the clipboard and paste it in your application's code as a decimal value, a hexadecimal value, or an RGB function.



Scrollbar controls can receive the input focus, and in fact they support both the TabIndex and TabStop properties. If you don't want the user to accidentally move the input focus on a scrollbar control when he or she presses the Tab key, you must explicitly set its TabStop property to False.

When a scrollbar control has the focus, you can move the indicator using the Left, Right, Up, Down, PgUp, PgDn, Home, and End keys. For example, you can take advantage of this behavior to create a read-only TextBox control with a numeric value that can be edited only through a tiny companion scroll bar. This scroll bar appears to the user as a sort of spin button, as you can see in the figure below. To make the trick work, you need to write just a few lines of code:

Private Sub Text1_GotFocus() ' Pass the focus to the scroll bar. VScroll1.SetFocus End Sub Private Sub VScroll1_Change() ' Scroll bar controls the text box value. Text1.Text = VScroll1.Value End Sub

25. a. Explain line and shape controls with a sample program.

There are four basic controls in VB6 that you can use to draw graphics on your form: the line control, the shape control, the image box and the picture box. To draw a straight line, just click on the line control and then use your mouse to draw the line on the form. After drawing the line, you can then change its color, width and style using the BorderColor, BorderWidth and BorderStyleproperties.Similarly, to draw a shape, just click on the shape control and draw the shape on the form. The default shape is a rectangle, with the default shape property set at 0. You can change the shape to square, oval, circle and rounded rectangle by changing the shape property's value to 1, 2, 3, 4, and 5 respectively. In addition, you can change its background color using the BackColor property as well its border style using the BorderStyle property, its border style using the BorderStyle property.

Example

This example will draw a circle, a sector, an arc and an ellipse with different colors and different DrawStyles

The Code Dim pi As Single

Private Sub Form_Load() 'Change the coordinates of the origin Scale (-4000, 4000)-(4000, -4000) Me.Width = 8000

Me.Height = 8000End Sub Private Sub cmd_Draw_Click(Index As Integer) pi = 4 * Atn(1)Select Case Index Case Is = 0Me.FillColor = vbRed Me.FillStyle = vbSolid Me.DrawStyle = vbDotMe.DrawWidth = 1Me.Circle (0, 0), 800, vbYellow ' Draw a circle with red border Case Is = 1Me.FillColor = vbCyan Me.FillStyle = vbSolid Me.DrawStyle = vbDotMe.DrawWidth = 3Me.Circle (-2000, 0), 800, vbBlack, -pi, -pi / 2 Case Is = 2Me.FillColor = vbYellow Me.FillStyle = vbSolid Me.DrawStyle = vbDashDot Me.DrawWidth = 1Me.Circle (-2000, -2000), 800, vbBlue, 0, pi Case Is = 3Me.FillColor = vbBlue Me.FillStyle = vbSolid Me.DrawStyle = vbDashDotDot Me.DrawWidth = 1Me.Circle (-1500, 2000), 800, vbCyan, , , 1.3 End Select

End Sub

The Output





The Line Control

The Line Method

Although the Pset method can be used to draw a straight line on the form, it is a little slow. It is better to use the Line method if you want to draw a straight line faster. The format of the Line command is shown below. It draws a line from the point (x1, y1) to the point (x2, y2) and the color constant will determine the color of the line.

Line (x1, y1)-(x2, y2), color

For example, the following command will draw a red line from the point (0, 0) to the point (1000, 2000).

Line (0, 0)-(1000,2000), VbRed

The Line method can also be used to draw a rectangle. The syntax is

Line (x1-y1)-(x2, y2), color, B

The four corners of the rectangle are (x1-y1), (x2-y1), (x1-y2) and (x2, y2)

Another variation of the Line method is to fill the rectangle with a certain color. The syntax is

Line (x1, y1)-(x2, y2), color, BF

If you wish to draw the graphics in a picture box, you can use the following syntaxes

Picture1.Line (x1, y1)-(x2, y2), color Picture1.Line (x1-y1)-(x2, y2), color, B Picture1.Line (x1-y1)-(x2, y2), color, BF Picture1.Circle (x1, y1), radius, color

Example The Bar Graph Plotter

We shall use the knowledge we gained from the Line method to create a bar graph. In this program, we shall insert a picture box for drawing the bar graph. In addition, we insert six text boxes for accepted the user input. We also insert two command buttons for drawing and reseting. Besides that, we need to define a new origin using the **Scale method**, otherwise, the bar graph will be upside down. The code is

Picture1.Scale (0, 5000)-(5000, 0)

The Code

Private Sub Command1_Click() Dim sale1, sale2, sale3, sale4, sale5, sale6 As Integer

sale1 = Val(Txt_Jan.Text) sale2 = Val(Txt_Feb.Text) sale3 = Val(Txt_Mac.Text) sale4 = Val(Txt_Mar.Text) sale5 = Val(Txt_May.Text) sale6 = Val(Txt_Jun.Text)

Picture1.Line (100, 0)-(600, sale1 * 50), vbRed, BF Picture1.Line (700, 0)-(1200, sale2 * 50), vbRed, BF Picture1.Line (1300, 0)-(1800, sale3 * 50), vbRed, BF Picture1.Line (1900, 0)-(2400, sale4 * 50), vbRed, BF Picture1.Line (2500, 0)-(3000, sale5 * 50), vbRed, BF Picture1.Line (3100, 0)-(3600, sale6 * 50), vbRed, BF Picture1.Line (3700, 0)-(4200, sale7 * 50), vbRed, BF Picture1.Line (4300, 0)-(4800, sale8 * 50), vbRed, BF

End Sub

Private Sub Command2_Click() Txt_Jan.Text = "" Txt_Feb.Text = "" Txt_Mac.Text = "" Txt_Apr.Text = "" Txt_May.Text = "" Txt_Jun.Text = ""

Picture1.Cls

End Sub

Private Sub Form_Load() 'To redefine the coordinates of the origin Picture1.Scale (0, 5000)-(5000, 0) End Sub

The Output



(**OR**)

b.Describe Multiple Document Interface in detail.

The Multiple Document Interface (MDI) was designed to simplify the exchange of information among documents, all under the same roof. With the main application, you can maintain multiple open windows, but not multiple copies of the application. Data exchange is easier when you can view and compare many documents simultaneously.

You almost certainly use Windows applications that can open multiple documents at the same time and allow the user to switch among them with a mouse-click. Multiple Word is a typical example, although most people use it in single document mode. Each document is displayed in its own window, and all document windows have the same behavior. The main Form, or MDI Form, isn't duplicated, but it acts as a container for all the windows, and it is called the parent window. The windows in which the individual documents are displayed are called Child windows.

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

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

To create an MDI application, follow these steps:

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

Visual Basic automatically associates this new Form with the parent Form. This child Form can't exist outside the parent Form; in the words, it can only be opened within the parent Form.

Parent and Child Menus

MDI Form cannot contain objects other than child Forms, but MDI Forms can have their own menus. However, because most of the operations of the application have meaning only if there is at least one child Form open, there's a peculiarity about the MDI Forms. The MDI Form usually has a menu with two commands to load a new child Form and to quit the application. The child Form can have any number of commands in its menu, according to the application. When the child Form is loaded, the child Form's menu replaces the original menu on the MDI Form

Following example illustrates the above explanation.

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

* Design a menu that has the following structure.

<>> MDIMenu Menu caption

• MDIOpen opens a new child Form
• MDIExit terminates the application

* Then design the following menu for the child

Form <> ChildMenu Menu caption

- Child Open opens a new child Form
- Child Save saves the document in the active child Form
- Child Close Closes the active child Form

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

Form1.Show

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

End

Double click on Child Close and enter the following code in the click event

Unload Me

Before run the application in the project properties set MDI Form as the start-up Form. Save and run the application. Following output will be displayed

26.a.Write a Program to print the Students grade with database Connectivity.

Dim db As Database Dim rs As Recordset Private Sub Combo1 Click() rs.MoveFirst While Not rs.EOF If Combo1.Text = rs.Fields(1) Then Text1.Text = rs.Fields(0)Text2.Text = rs.Fields(1)Text3.Text = rs.Fields(2)Text4.Text = rs.Fields(3)Text5.Text = rs.Fields(4)Text6.Text = rs.Fields(5)Text7.Text = rs.Fields(6)Text8.Text = rs.Fields(7)Text9.Text = rs.Fields(8)Text10.Text = rs.Fields(9)Text11.Text = rs.Fields(10)Text12.Text = rs.Fields(11)Text13.Text = rs.Fields(12)Text14.Text = rs.Fields(13)

Text15.Text = rs.Fields(14)End If rs.MoveNext Wend End Sub Private Sub Combo1_GotFocus() rs.MoveFirst While Not rs.EOF Combo1.AddItem (rs.Fields(1)) rs.MoveNext Wend End Sub Private Sub Command1_Click() Text14.Text = Val(Text7.Text) + Val(Text10.Text) + Val(Text13.Text) End Sub Private Sub Command2_Click() If (Text14.Text ≥ 150) Then Text15.Text = "pass" Else Text15.Text = "fail" End If End Sub Private Sub Command3_Click() rs.AddNew rs.Fields(0) = Text1.Textrs.Fields(1) = Text2.Textrs.Fields(2) = Text3.Textrs.Fields(3) = Text4.Textrs.Fields(4) = Text5.Textrs.Fields(5) = Text6.Textrs.Fields(6) = Text7.Textrs.Fields(7) = Text8.Textrs.Fields(8) = Text9.Text rs.Fields(9) = Text10.Textrs.Fields(10) = Text11.Textrs.Fields(11) = Text12.Textrs.Fields(12) = Text13.Textrs.Fields(13) = Text14.Textrs.Fields(14) = Text15.TextMsgBox "data saved" rs.Update End Sub

Private Sub Command4_Click() rs.MoveFirst While Not rs.EOF If rs.Fields(1) = Combo1.Text Then rs.Delete MsgBox "Record deleted successfully", vbInformation, "student details" End If rs.MoveNext Wend Combo1.Clear End Sub Private Sub Command5_Click() Text1.Text = "" Text2.Text = "" Text3.Text = "" Text4.Text = "" Text5.Text = "" Text6.Text = "" Text7.Text = "" Text8.Text = "" Text9.Text = "" Text10.Text = "" Text11.Text = "" Text12.Text = "" Text13.Text = "" Text14.Text = "" Text15.Text = "" Combo1.Clear End Sub Private Sub Command6_Click() rs.MoveFirst While Not rs.EOF If rs.Fields(1) = Combo1.Text Then rs.Edit rs.Fields(0) = Text1.Textrs.Fields(1) = Text2.Textrs.Fields(2) = Text3.Textrs.Fields(3) = Text4.Textrs.Fields(4) = Text5.Textrs.Fields(5) = Text6.Textrs.Fields(6) = Text7.Textrs.Fields(7) = Text8.Textrs.Fields(8) = Text9.Text

```
rs.Fields(9) = Text10.Text
rs.Fields(10) = Text11.Text
rs.Fields(11) = Text12.Text
rs.Fields(12) = Text13.Text
rs.Fields(13) = Text14.Text
rs.Fields(14) = Text15.Text
MsgBox "Data updated successfully", vbInformation
rs.Update
End If
rs.MoveNext
Wend
Combo1.Clear
End Sub
Private Sub Form_Load()
```

Set db = OpenDatabase("Y:\student\studi.mdb") Set rs = db.OpenRecordset("studs") End Sub Private Sub Text7_Click() Text7.Text = Val(Text5.Text) + Val(Text6.Text) Text10.Text = Val(Text8.Text) + Val(Text9.Text) Text13.Text = Val(Text11.Text) + Val(Text12.Text) End Sub

Student personal Detail	Stud	lents Ma	rk Detail	
STU_NAME Gayatlui		INT	EXT	TOTAL
ROLL_NO	RDBMS	34	35	69
REG_NO 2	ENGLISH	33	34	67
CLASS it	WEB DESIGN	32	31	63
Add Delete	Total	199	Result	pass
Clear Update	T	OTAL	RESU	LT

STUDENT INFORMATION

stuname	rolino	regno	class	rbms	english	web_design	total	Result
shri	1001	6	it	90	91	97	278	pass
Gayathri	1002	2	it	69	67	63	199	pass
Suganya	1003	3	it	83	87	77	247	pass
hari	1004	4	it	90	78	82	250	pass
jeevi	1005	5	it	95	22	28	145	fail

(**OR**)

b. Write a Program to print the Employee Payroll with database Connectivity. /* EMPLOYEE PAYROLL */

Dim db As Database Dim rs As Recordset Private Sub Combo1_Click() rs.MoveFirst While Not rs.EOF If Combo1.Text = rs.Fields(0) Then Text1.Text = rs.Fields(0) Text2.Text = rs.Fields(1) Text3.Text = rs.Fields(2) Text4.Text = rs.Fields(3) Text5.Text = rs.Fields(3) Text5.Text = rs.Fields(4) Text6.Text = rs.Fields(5) Text7.Text = rs.Fields(6) Text8.Text = rs.Fields(7)Text9.Text = rs.Fields(8)Text10.Text = rs.Fields(9)Text11.Text = rs.Fields(10)End If rs.MoveNext Wend End Sub Private Sub Combo1_GotFocus() rs.MoveFirst While Not rs.EOF Combo1.AddItem (rs.Fields(0)) rs.MoveNext Wend End Sub Private Sub Command1_Click() //Calculate Text7.Text = Val(Text6.Text) Text8.Text = Text7.Text * 0.15Text9.Text = Text7.Text * 0.2 Text10.Text = Text7.Text * 0.08 Text11.Text = Text7.Text + (Text7.Text * 0.15) + (Text7.Text * 0.2) - (Text7.Text * 0.08)Combo1.Clear End Sub Private Sub Command2_Click() //Save rs.AddNew rs.Fields(0) = Text1.Textrs.Fields(1) = Text2.Textrs.Fields(2) = Text3.Textrs.Fields(3) = Text4.Textrs.Fields(4) = Text5.Textrs.Fields(5) = Text6.Textrs.Fields(6) = Text7.Textrs.Fields(7) = Text8.Textrs.Fields(8) = Text9.Text rs.Fields(9) = Text10.Textrs.Fields(10) = Text11.TextMsgBox "data saved successfully", vbInformation rs.Update Combo1.Clear End Sub Private Sub command3_Click() //Clear Text1.Text = ""

Text2.Text = "" Text3.Text = "" Text4.Text = "" Text5.Text = "" Text6.Text = "" Text7.Text = "" Text8.Text = "" Text9.Text = "" Text10.Text = "" Text11.Text = "" Combo1.Clear End Sub Private Sub Command4_Click() //Update rs.MoveFirst While Not rs.EOF If rs.Fields(0) = Combo1.Text Then rs.Edit rs.Fields(0) = Text1.Textrs.Fields(1) = Text2.Text rs.Fields(2) = Text3.Textrs.Fields(3) = Text4.Textrs.Fields(4) = Text5.Textrs.Fields(5) = Text6.Textrs.Fields(6) = Text7.Textrs.Fields(7) = Text8.Textrs.Fields(8) = Text9.Text rs.Fields(9) = Text10.Textrs.Fields(10) = Text11.TextMsgBox "data updataed successfully", vbInformation rs.Update End If rs.MoveNext Wend Combo1.Clear End Sub Private Sub Command5_Click() //Delete rs.MoveFirst While Not rs.EOF If rs.Fields(0) = Combo1.Text Then rs.Delete MsgBox "record deleted successfully", vbInformation, "pay roll" End If rs.MoveNext Wend

Combo1.Clear End Sub

Private Sub Form_Load() Set db = OpenDatabase("Y:\shri33\Shri Animation\shriemp.mdb") Set rs = db.OpenRecordset("emp33") End Sub

🖻 Form1							
Employee payslip							
EMP_ID	001 001 - 001 001 002	BASIC SALARY	400000				
EMP_NAME	shrimathi	HRA	60000				
ADDRESS	EE 11 north housing unit selvar	DA	80000				
DEPARTMENT	Π	PF	32000				
DESIGNATION	HR	NETPAY	508000				
SALARY	400000						
	calculate ADD	<u>clear</u> UPDAT	delete				