Training Manual

FOR

Microsoft

Visual Basic Programming

Chapter-1: Introduction to Visual Basic Programming

VISUAL BASIC is one of the easiest programming tool to master. With some basic guidance, anybody could come up with a nice little windows-based program within a short time, age is not the limit. However, do not ever think that Visual Basic can only build simple programs, you could actually develop very advance and powerful applications. Indeed, Visual Basic 6.0 even allows you to develop web applications.

VISUAL BASIC is a high level programming language evolved from the earlier DOS version called BASIC. BASIC means Beginners' All-purpose Symbolic Instruction Code. It is a fairly easy programming language to learn. The codes look a bit like English Language.

Visual Basic (VB) is a beginner programming language for authoring Windows©-based software. In the past, VB was known as the language of the hobbyist, but since version 5.0, professional programmers have taken notice of the potential of VB. Applications written with VB will only run on computers with a minimum Windows 95.

1.1 What is Visual Basic?

Visual Basic is a high level programming language developed from the earlier DOS version called BASIC. Learning Visual Basic is quite easier than other programming languages, such as C++, C#, Java and so on. This is because Visual Basic enables you to work in a graphical user interface (GUI) where you can just drag and drop controls that you want to work with where you have to write bunches of code to create in C++ or C# or even in Java. If you are new to programming and want to start it in the smoothest and easiest way, then Visual Basic is your right choice!

1.2 Terminology

Before you begin to write your first program in Visual Basic, it is important that you understand some of the basic terminologies. This section describes Visual Basic terminologies.

Solutions vs. Projects

A project is a set of one or more files that contain the code and resources that define an application's executable (.EXE) or a dynamic link library (.DLL). A solution is a container that stores one or more related projects. For example, you might store the project for an application's executable and the dynamic link libraries it uses in the same solution.

Forms and Controls

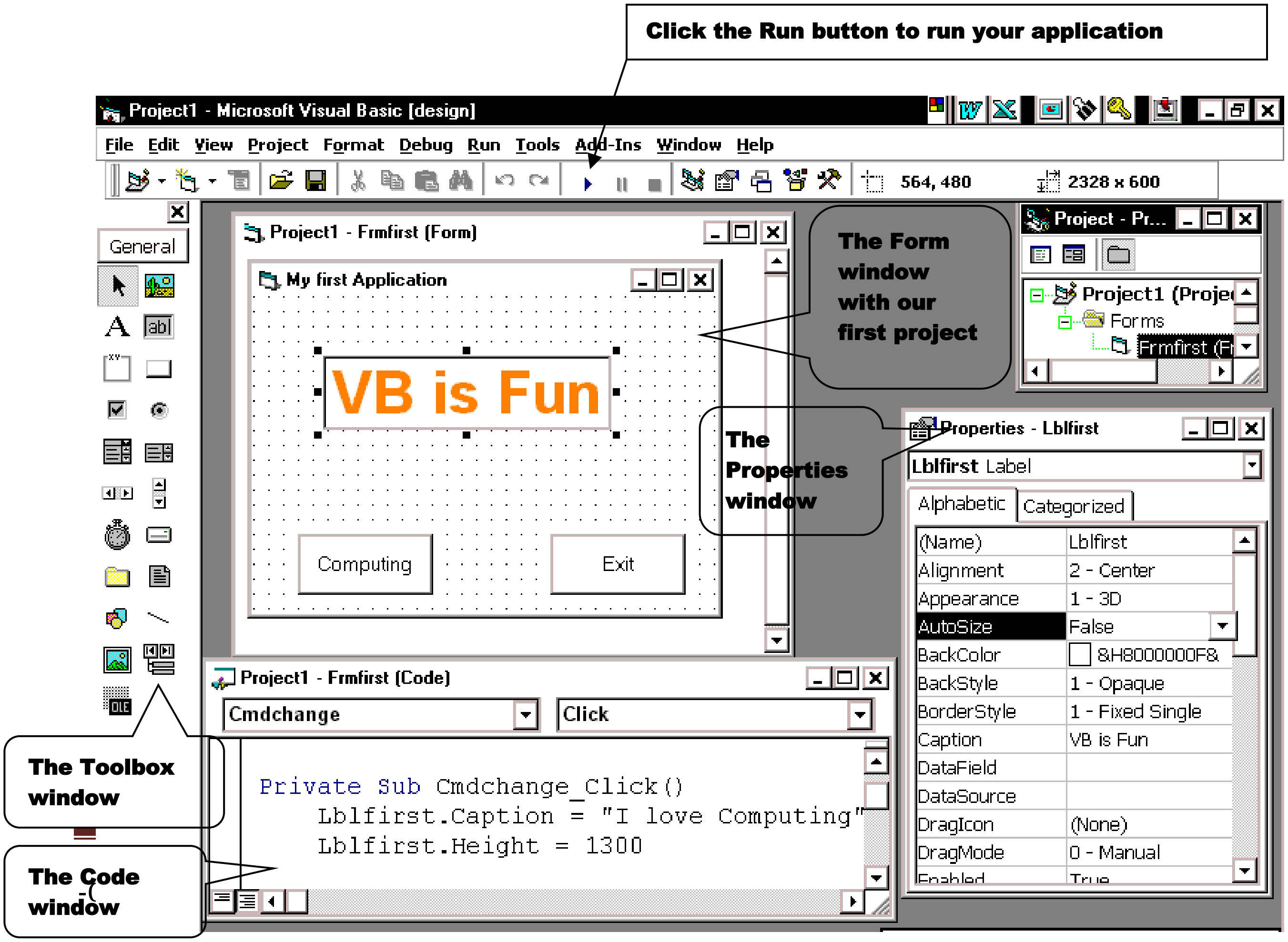
Visual Basic programs are built using forms and controls. A form is a container. Each window you view when you execute an application is built using a form. A control is an object that a user can interact with. Some of the control types include Buttons, lists, and textboxes.

Classes

Each object in Visual Basic .NET (and later) is actually a class. A class is similar to a template that describes an object, such as a form or control. When a class is instantiated (loaded into memory), it becomes an object.

Properties, Methods, and Events

Each class has properties, methods, and events. A property is a piece of information about an object. It defines how an object looks or the data it contains. For example, a form has a Backcolor property that defines the color of its background, a Text property that defines the text on its title bar, and a Width and Height property that defines its size. The figure below depicts Visual Basic opening screen.



Click the Run button to run your application

The Form window with our first project

The Properties window

The Toolbox window

The Code window

```
Private Sub Cmdchange_Click()
    Lblfirst.Caption = "I love Computing"
    Lblfirst.Height = 1300
```

(Name)	Lblfirst
Alignment	2 - Center
Appearance	1 - 3D
AutoSize	False
BackColor	□ &H8000000F&
BackStyle	1 - Opaque
BorderStyle	1 - Fixed Single
Caption	VB is Fun
DataField	
DataSource	
DragIcon	(None)
DragMode	0 - Manual
Enabled	True

A method is an action that can be performed on an object. For example, you can display a form by calling the form's Show method and close a form by calling its Close method.

An Event is generally something a user causes to happen. Objects can listen for events and respond when they occur by executing the code you add to the event procedure. This is known as event-driven programming.

Variables and Constants

A variable is a named location in memory that stores a piece of data. You can modify the data in a variable during execution. A constant is also a named location in memory. The data in a constant is defined when the program is loaded. A constant's value cannot be modified.

Design View and Code View

When building a project, you will work in both design view and code view. Design view gives you access to a toolbox, which allows you to place controls on a form. Code view allows you to write the code that defines how your program will execute.

Debugging Your Program

When writing code, it is important to test it. When you debug your program, you run it inside the Visual Studio environment. You can set breakpoints to cause your program to stop running and enter break mode when a condition is met. In break mode, you can step through the code one line at a time and view the contents of properties and variables.

Knowing these terminologies will get you started. However, as you program in Visual Basic, you will encounter many more terms and concepts. Visual Basic is a powerful programming language, full of many features and, therefore, many terms. This section also includes the details specific to Labels, Text Boxes, and Form Properties.

What are Labels?

Programmers use labels to place titles around the form and to label text boxes. This helps users know what to type into the text box. When you use the Label control, your code can change the label's text so that different messages can appear when needed.

When you place labels on a form, you will almost always set the Label control's **Name property** and type a new **Caption value**. In addition, you will want to change the Font property and possibly the label's color and style.

The following are the most common Label control properties that you will set as you work with the Label control.

Property	Description
Alignment	Determines whether the label's caption appears left-justified, centered, or right-justified within the label's boundaries.
AutoSize	Enlarges the label's size properties, when True, if you assign a caption that is too large to fit in the current label's boundaries at runtime.
BackColor	Specifies the label's background color. Click the BackColor's palette down arrow to see a list of colors and click Categorized to see a list of common Windows control colors.
BackStyle	Determines whether the background shows through the label or if the label covers up its background text, graphics, and color.
BorderStyle	Determines whether a single-line border appears around the label.
Caption	Holds the text that appears on the label.
Enabled	Determines whether the label is active. Often, you'll change the Enabled property at runtime with code when a label is no longer needed.
Font	Produces a Font dialog box in which you can set the caption's font name, style, and size.
ForeColor	Holds the color of the label's text.
Height	Holds the height of the label's outline in twips.

Left	Holds the number of twips from the label's left edge to the Form window's left edge.
MousePointer	Determines the shape of the mouse cursor when the user moves the mouse over the label.
TabIndex	Specifies the order of the label in the focus order. Although the label cannot receive the direct focus, the label can be part of the focus order.
ToolTipText	Holds the text that appears as a tooltip at runtime.
Top	Holds the number of twips from the label's top edge to the Form window's top edge.
Visible	Determines whether the label appears or is hidden from the user.
Width	Holds the width of the label in twips.
WordWrap	Determines whether the label expands to fit whatever text appears in the caption.

What are Text Boxes?

Text boxes accept user input. Although several other controls accept user input, text boxes are perhaps the easiest to set up and respond to. In addition, a text box is simple to use and hence can be seen on Windows forms all the time.

The table below lists the common properties associated with text boxes. By familiarizing yourself with the properties now, you will be able to more quickly produce applications as you learn more about Visual Basic.

Property	**Description**
Alignment	Determines whether the text box's text appears left-justified, centered, or right-justified within the text box's boundaries.
BackColor	Specifies the text box's background color. Click the BackColor property's palette down arrow to see a list of colors and click Categorized to see a list of common Windows control colors.
BorderStyle	Determines whether a single-line border appears around the text box.
Enabled	Determines whether the text box is active. Often, you'll change the Enabled

	property at runtime with code when a text box is no longer needed.
Font	Produces a Font dialog box in which you can set the Text property's font name, style, and size.
ForeColor	Holds the color of the text box's text.
Height	Holds the height of the text box's outline in twips.
Left	Holds the number of twips from the text box's left edge to the Form window's left edge.
Locked	Determines whether the user can edit the text inside the text box that appears.
MaxLength	Specifies the number of characters the user can type into the text box.
MousePointer	Determines the shape of the mouse cursor when the user moves the mouse over the text box.
MultiLine	Lets the text box hold multiple lines of text or sets the text box to hold only a single line of text. Add scrollbars if you wish to put text in a multiline text box so your users can scroll through the text.
PasswordChar	Determines the character that appears in the text box when the user enters a password (keeps prying eyes from knowing what the user enters into a text box).
ScrollBars	Determines whether scrollbars appear on the edges of a multiline text box.
TabIndex	Specifies the order of the text box in the focus order.
TabStop	Determines whether the text box can receive the focus.
Text	Holds the value of the text inside the text box. The Text property changes at runtime as the user types text into the text box. If you set an initial Text property value, that value becomes the default value that appears in the text box when the user first sees the text box.
ToolTipText	Holds the text that appears as a tooltip at runtime.
Top	Holds the number of twips from the text box's top edge to the Form window's top edge.

Visible	Determines whether the text box appears or is hidden from the user.
Width	Holds the width of the text box in twips.

What are Form Properties?

Forms have properties that you can and must set when you create an application. Being the background of your application, the form's properties help set the stage for the rest of the project. The form supports more property values than the other controls described here; however the list below lists only the most common properties that you'll need.

New Term: Pixel stands for picture element and represents the smallest addressable graphic dot on your monitor.

Property	**Description**
BackColor	Specifies the form's background color. Click the BackColor's palette down arrow to see a list of colors and click Categorized to see a list of common Windows control colors.
BorderStyle	Determines how the Form window appears. The BorderStyle property specifies whether the user can resize the form and also determines the kind of form you wish to display.
Caption	Displays text on the form's title bar at runtime.
ControlBox	Determines whether the form appears with the Control menu icon. The Control menu appears when your application's user clicks the Control menu icon.
Enabled	Determines whether the form is active. Often, you'll change the Enabled property at runtime with code when a form is no longer needed. Generally, only multiform applications, such as MDI applications, need to modify a form's Enabled property.
Font	Produces a Font dialog box in which you can set the text's font name, style, and size.
ForeColor	Holds the color of the form's text.

Height	Holds the height of the form's outline in twips.
Icon	Describes the icon graphic image displayed on the taskbar when the user minimizes the form.
Left	Holds the number of twips from the form's left edge to the screen's left edge.
MaxButton	Specifies whether a maximize window button appears on the form.
MinButton	Specifies whether a minimize window button appears on the form.
MousePointer	Determines the shape of the mouse cursor when the user moves the mouse over the form.
Moveable	Specifies whether the user can move the form at runtime.
Picture	Determines a graphic image that appears on the form's background at runtime.
ScaleMode	Determines whether the form's measurements appear in twips, pixels (the smallest graphic dot image possible), inches, centimeters, or other measurements.
ShowInTask bar	Determines whether the form appears on the Windows taskbar.
StartUpPosition	Determines the state (centered or default) of the form at application startup.
Top	Holds the number of twips from the form's top edge to the Form window's top edge.
Visible	Determines whether the form appears or is hidden from the user.
Width	Holds the width of the form in twips.
WindowState	Determines the initial state (minimized, maximized, or normal) in which the window appears at runtime.

1.3 Creating an Application

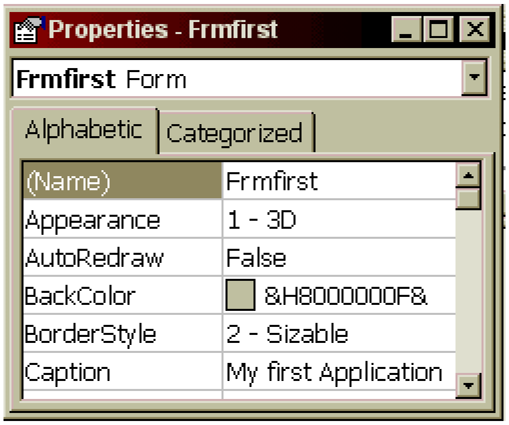
This section describes the ‘How To’ part of performing various activities as part of creating a successful application.

1.3.1 How to Start a New Project?

To start a new project, follow these steps:

1. Create a new project by going **File/New Project** and double-clicking the **Standard EXE** icon.

2. Change the form's Name property to `frmFirst` and change its Caption property to `My First Application`. The form's Caption property text appears in the title bar when you run the application.



Frmfirst Form

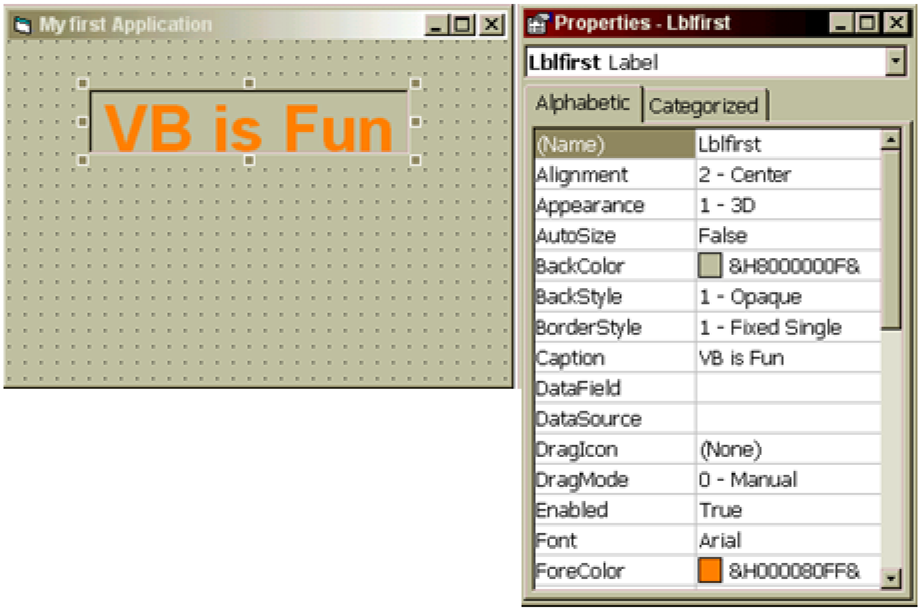
(Name)	Frmfirst
Appearance	1 - 3D
AutoRedraw	False
BackColor	▨ &H8000000F&
BorderStyle	2 - Sizable
Caption	My first Application

1.3.2 How to Create a Label?

To create a label, follow these steps:

1. Click the Label control and drag a Label on the form.
2. Modify the label's Name property to lblFirst. Also modify the label's Caption property to VB is fun.
3. Click the label's Font property value to display the Font dialog box for the label. Set the font to Arial, the size to 24 and set Bold property.

4. Modify the fore color to Orange, or any color you choose, and change the label's Alignment property to 2-Center.



(Name)	Lblfirst
Alignment	2 - Center
Appearance	1 - 3D
AutoSize	False
BackColor	&H8000000F&
BackStyle	1 - Opaque
BorderStyle	1 - Fixed Single
Caption	VB is Fun
DataField	
DataSource	
DragIcon	(None)
DragMode	0 - Manual
Enabled	True
Font	Arial
ForeColor	&H000080FF&

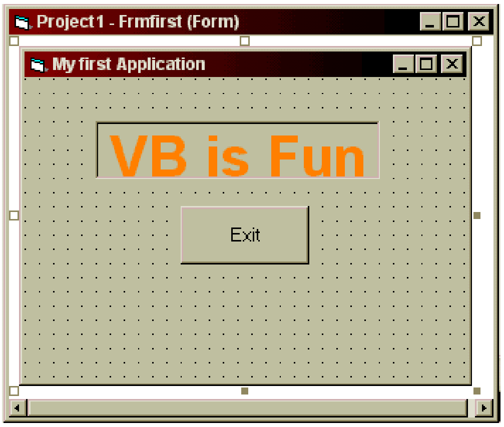
5. Modify the label's BorderStyle property to 1-FixedSingle. This property adds a single-line 3D border around the label. You can modify the shape of the label to fit the text.

1.3.3 How to Create a Command Button?

To create a command button, follow these steps:

1. Add a Command button by double-clicking the command button tool on the Toolbox window. The command button appears in the middle of the form and you can leave it where it is.

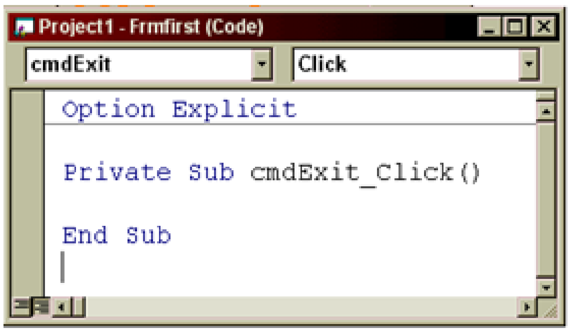
2. Modify the command button's Name property to cmdExit.



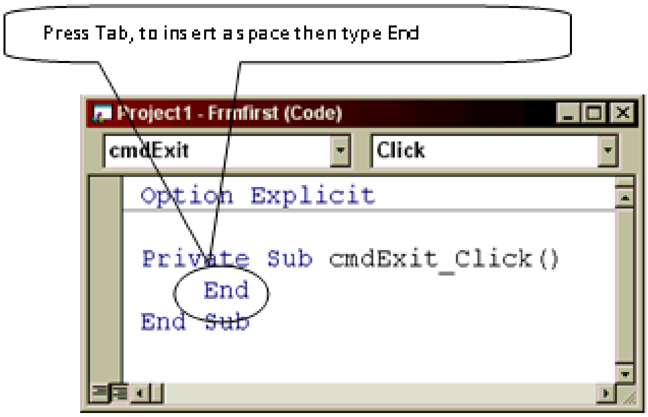
3. Modify the command button's Caption property to Exit. Watch the command button as you type the Caption property text. The command button's caption becomes the text you type.
4. Double-click the form's command button and Visual Basic instantly opens the Code window and displays the wrapper lines for the command button's Click event procedure. The command button will be used to exit the program. When you click the command button, your application must end. When you click a command button, a Click event occurs. Therefore, to respond to this event, you must write an event procedure for the command button. Refer Section 1.3.4 How to Write Code for more information.

1.3.4 How to Write Code?

The name of the procedure, cmdExit_Click(), describes both the control and the event being processed by the code you only need to fill in the body. Type End for the one-word body of the event procedure and close the Code window. End tells Visual Basic to end the running application. Therefore, the application will end when the user clicks the command button.

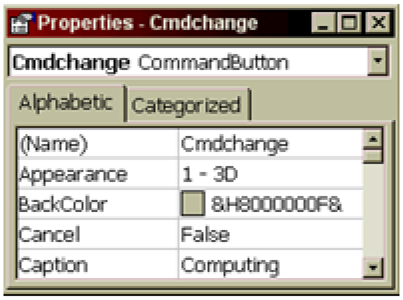


```
Option Explicit

Private Sub cmdExit_Click()

End Sub
```


Press Tab, to insert a space then type End

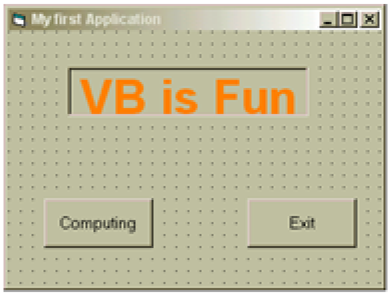
```
Option Explicit

Private Sub cmdExit_Click()
    End
End Sub
```

1. Press the Run button to run the program and watch your creation appear. In this example, we are going to make a button that will change the words VB is Fun, in the Label to I love computing.
2. Create another Command Button and set the Name to Cmdchange, and the Caption to Computing.


Properties - Cmdchange

Cmdchange CommandButton

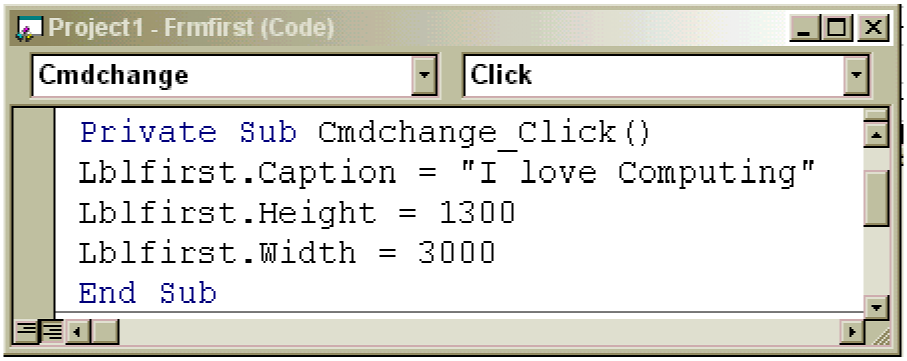
(Name)	Cmdchange
Appearance	1 - 3D
BackColor	&H8000000F&
Cancel	False
Caption	Computing



3. Double click on the button to open the Code window. Type in the following:

   Lblfirst.Caption = "I love Computing"

Note: Remember earlier we set the properties for the label caption as VB is fun. This code says for the caption field of Lblfirst put the words I love computing.


Project1 - Frmfirst (Code)

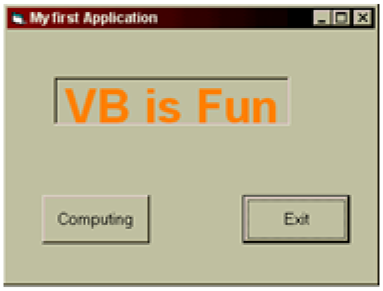
```
Private Sub Cmdchange_Click()
Lblfirst.Caption = "I love Computing"
Lblfirst.Height = 1300
Lblfirst.Width = 3000
End Sub
```

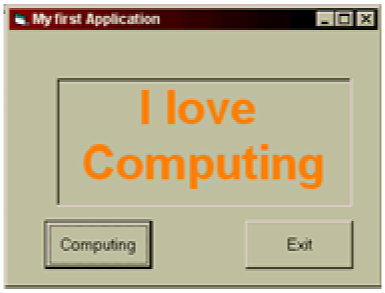
4. Set the height of the label to 1400 as shown below. The words will be too big for the Label window, so we need to make the font size smaller as well.

   Lblfirst.Height = 1300

5. Set the width as shown below to make the Label window wider to fit the words.

   Lblfirst.Width = 3000

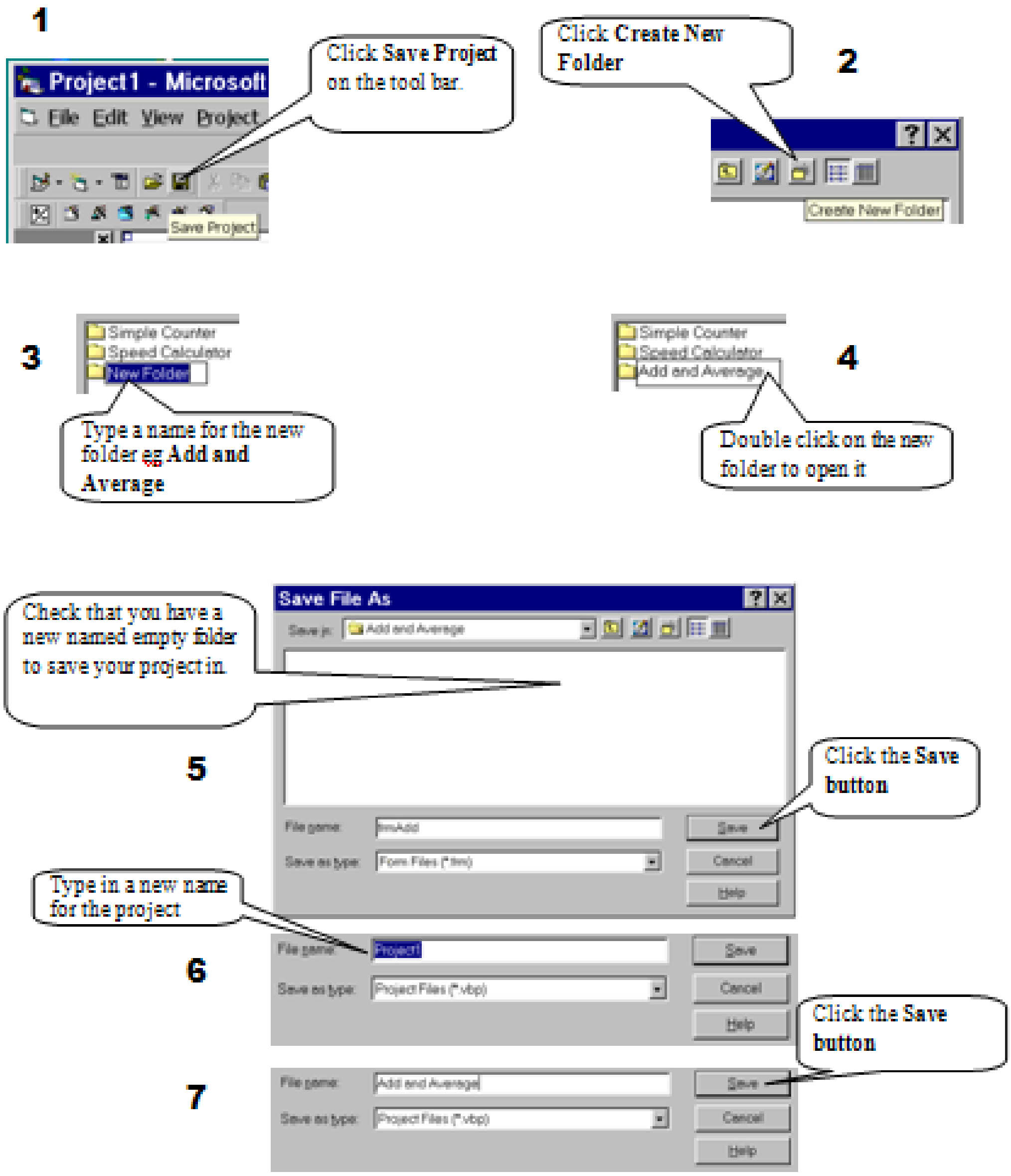




6. Run the program by clicking on the Run button and see if it works.

1.4 Saving a Visual Basic Program

This section depicts the flow to successfully save a Visual Basic program.



1 Click Save Project on the tool bar.

2 Click Create New Folder

3 Type a name for the new folder eg Add and Average

4 Double click on the new folder to open it

5 Check that you have a new named empty folder to save your project in.

Save in: Add and Average

File name: frmAdd

Save as type: Form Files (*.frm)

Click the Save button

6 Type in a new name for the project

File name: Project1

Save as type: Project Files (*.vbp)

7 File name: Add and Average

Save as type: Project Files (*.vbp)

Click the Save button

1.5 Modular Environment

Visual Basic supports modular environment. Code in Visual Basic is stored in the form of modules. The following are the three kinds of modules in Visual Basic:

- Form Modules
- Standard Modules
- 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) is 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
- 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)
- The dash, although legal, must be avoided because it may be confused with the minus sign. Instead of First-name, you can use First_name or FirstName