QUICK GUIDE TO
VISUAL BASIC ® EXPRESS

©Microsoft Corporation

VBDESIGNER © EDITION

©Dr. James D. Fabrey

West Chester University

May 2007
Preface

General education classes in computer science (for non-majors) frequently include one or more of the following categories of instruction:

- Skills (such as word processing, spreadsheet, etc.)
- Concepts (an overview of computing and technology)
- Programming appreciation (a sampling of a programming course)
- Information management

This textbook can be used in courses for which programming appreciation is approximately 1/3 of the course, as it is at West Chester University of Pennsylvania. There are numerous textbooks on programming in Visual Basic Express, but most of are much lengthier, suitable for a one or even two semester course, solely in programming.

This textbook is based on demo files and folders, which the student can download and unzip online, as directed in Chapter 0, to a flash drive. Instructions are also given for submitting projects to the instructor through the Blackboard © course management system.

Chapters 1 – 3 each present a demonstration program or project based on a fictional phone store. The students learn about programming topics through these projects and practice building them. Each chapter includes a lab project, based on a fictional computer store, to be graded for credit. It is assumed that the course will also have 2 – 3 quiz projects to be done in the lab for credit, with no assistance from the instructor.

Each chapter also includes an optional lab exercise, not to be graded for credit. It is quite possible that the instructor might present his/her favorite exercises, also for practice.
About The Author

At the time of this writing, I have been the Chairperson of the Computer Science Department at West Chester University of Pennsylvania for 5 years. Originally receiving my PhD in Mathematical Physics at the Massachusetts Institute of Technology, I switched my primary field to computer science at the University of North Carolina at Chapel Hill and have taught that subject at West Chester University (WCU) for over 30 years.

I have seen the general education course at WCU pass through the following phases:

- In the 1970’s: 1/3 concepts, 2/3 programming appreciation
- In the 1980’s: 1/3 concepts, 2/3 skills
- In the 1990’s: 1/10 concepts, 9/10 skills
- In the 2000’s: 2/3 concepts, 1/3 programming appreciation

Perhaps this unusual cycling can be attributed to students finally acquiring computer skills before coming to college, yet still not really being exposed to programming in a meaningful way. Also, concepts have become more important because the pace of technological change seems to have accelerated – making it harder for the average person to stay current.

Programming appreciation is comparable to the approach taken in general education courses in biology or chemistry (or art history, for that matter), where it is not assumed that the student will go on in the subject but will profit from a well-rounded background. However, at WCU, for example,
students can continue on with a full course in programming in Visual Basic, which in turn can lead to an Information Technology minor.

Several editions of this book have been used over the past five years at WCU. I am particularly proud of this VBDesigner Edition – it makes use of my VBDesigner program. It has been very popular with my students. Writing a project exclusively in Visual Basic requires 5 steps:

- Selecting the objects for the window being programmed
- Entering text for these objects
- Choosing fonts and colors
- Naming the objects
- Writing the code (instructions) for some of the objects

The first four steps are done in VBDesigner, where everything is greatly simplified. Only the last step is still done in Visual Basic. As a result, project work has gone much more smoothly in the lab, and student grades have increased by 10% – 15%.

I hope that you will enjoy using this VBDesigner approach, as well as Visual Basic itself.

It should be noted that Visual Basic can be used to program web pages, but that for simplicity in this book, we restrict ourselves to programming windows.
# TABLE OF CONTENTS

CHAPTER 0: INTRODUCTION ................................................................. 6  
Files and Folders .............................................................................. 7  

CHAPTER 1: LABELS, LISTBOXES, TEXTBOXES, BUTTONS .......... 9  
Programming Appreciation ............................................................... 9  
Reviewing the Design (Properties) ................................................... 13  
Reviewing the Code (Instructions) ...................................................... 14  
Practice Building the Project ............................................................ 22  
Lab Exercise 1 (Optional: No Credit) ............................................... 34  
Lab Assignment 1 (Credit) ................................................................. 36  
Lab Assignment 1: Comments .......................................................... 40  

CHAPTER 2: CALCULATIONS, MESSAGE BOXES .................. 41  
Reviewing the Design (Properties) ................................................... 46  
Reviewing the Code (Instructions) ...................................................... 47  
Practice Building the Project ............................................................ 54  
Lab Exercise 2 (Optional: No Credit) ............................................... 59  
Lab Assignment 2 (Credit) ................................................................. 62  
Lab Assignment 2: Comments .......................................................... 68  

CHAPTER 3: LISTS, LOOPS, FORMATTING ......................... 69  
Reviewing the Design (Properties) ................................................... 72  
Reviewing the Code (Instructions) ...................................................... 73  
Practice Building the Project ............................................................ 80  
Lab Exercise 3 (Optional: No Credit) ............................................... 84  
Lab Assignment 3 (Credit) ................................................................. 86  
Lab Assignment 3: Comments ......................................................... Error! Bookmark not defined.
CHAPTER 0: INTRODUCTION

The Visual Basic 2005 Express programming language can be used to learn how computer programmers instruct the computer to perform specified tasks. We will consider pre-written demonstration projects and will

- run them
- examine how they were programmed
- program them ourselves

Visual Basic 2005 Express automatically opens in the lab whenever you open a project file. You should be able to complete all of your work in the lab, but if you want it for your own computer, you can download it for free via www.microsoft.com.

TERMS

We will use the terms program and project interchangeably. Typically, the word project is used in naming files and folders that we will be using.

We will also use the terms window and form interchangeably. The term form is really short for business form. We will learn how to design the objects in the form.

We will use the terms instructions and code interchangeably. We will learn how to write the code that makes the project operate properly.
Files and Folders

Here are the steps to be followed and the files and folders involved with each step (this picture is the rotated login screen in VBDesigner).
NOTE: If you ever have “glitches” in Visual Basic Express 2005 or VBDesigner, and your instructor cannot get past them, they are probably “folder” problems. If this happens, get a fresh start – using Spare Project Folder.

TO LOAD THE DEMO AND LAB PROJECTS FOR THE COURSE

(THIS ONLY HAS TO BE DONE ONCE, UNLESS YOU LOSE YOUR FLASH DRIVE!)

• MAKE SURE YOU HAVE A FLASH DRIVE!!!
• Using an Internet browser, go to www.cs.wcupa.edu/~jfabrey/Student.zip
• Save the .zip file to the Flash Drive
• Right-click the Start Button to Explore the Flash Drive
• Right-click the .zip file and Open with Compressed Folder
• Extract Button or File Menu-Extract Command
• Remove \Student from Flash Drive location
• Next Button, then Finish Button

FOR EACH SECTION OF EACH CHAPTER:

WHAT SHOULD YOU LOG ONTO?

FOR ALL SECTIONS

Right-click the green Start Button and choose Explore. Then log onto your flash drive and your course folder.

FOR EACH SECTION

The phrase “logging onto” means within the course folder.
CHAPTER 1: LABELS, LISTBOXES, TEXTBOXES, BUTTONS

Programming Appreciation

For Object Properties & Sample Run

1A Phone Sale Demo Folder:

Executable Phone Sale File

If you point the mouse - without clicking - over an object, Visual Basic 2005 Express’s tooltip feature will display – for a short time – the type of object and its key properties, such as

- **BackColor** (background color)
- **ForeColor** (foreground color)
- **font type** (Comic Sans MS, Courier New, Microsoft Sans Serif, Times New Roman)
- **font size** (8, 10, 12, or 14)
- **ReadOnly for TextBoxes** (False or True)

There are certain properties with default values set by **VBDesigner**.
Visual Basic 2005 Express is capable of using a very large number of objects, but for simplicity, we will only use the following:

- **Labels** (to make the purpose of the other objects understandable)
- **TextBoxes**
  - for typing text or numbers
  - the code for ListBoxes and Buttons can change the contents of TextBoxes
- **ListBoxes**
  - for choices of up to 4 items
  - for longer lists of items from which to select
- **Buttons** (clicking on these invokes their code as described by the text on the Button)

For simplicity, the colors are limited to Black, White, LightGray, Red, Green, Blue, Yellow, Aqua, and Silver. For greater readability, all text is boldfaced.

If you actually click the mouse on an object, its code (if any) is executed. The project that you are running is described below and illustrated by screen snapshots.

The project is a sales promotion by a fictional phone store. There are computer instructions, or code, which have been programmed for the
**ListBox** and the **Exit** button. **Clicking on the object invokes the code.** Clicking on **Advertise** puts a two-sentence message into the **TextBox.** Clicking on **Clear** removes the message and positions the cursor inside the **TextBox** for the salesperson to put in a customized message. Clicking on **Exit** will terminate the project. In summary, you may do the following:

- click on **Advertise**
- click on **Clear** and then type whatever note you wish
- click on the **Exit** button

Try these actions! When you are done, we are ready to learn more about how this project works. We show some demo screens below.
Sale!!

Type in whatever sale message you want and use word wrap in this text box.

Sale!!

Special Sale Today:
- 20% Off
- Voice/Internet Plans:
  - 30% Off

Choose one

Clear

Exit
Reviewing the Design (Properties)

For Object Properties

VBDesigner – Step 2

1A Phone Sale Demo Folder: Objects File

or

For Object Properties

1A Phone Sale Demo Folder

Executable Phone Sale File

In VBDesigner, click on each object in the right panel (Phone Sale) and observe its properties in the left panel (ToolBox and Properties). This is the same screen that you will use when running VBDesigner – Steps 1 & 2 to practice building all of your projects (the Demo projects for practice and the Lab and Quiz projects for credit).

For comparison, you can simultaneously run the Executable Phone Sale file and look at the Tool Tips. If you point the mouse - without clicking - over an object, Visual Basic 2005 Express’s tooltip feature will display – for a short time – the type of object and its key properties.
Reviewing the Code (Instructions)

VBDesigner – Step 3

1A Phone Sale Demo Folder: Objects File

Click on each object in the right panel (Phone Sale) and observe its code in the left panel (View Code for Objects). Note that Labels and TextBoxes do not have code. Labels are not involved in the code in any way, but TextBoxes sometimes are involved in the code of ListBoxes and Buttons.

Let us try to understand what the code for each object means and how it works. All of our projects will always have code for

- the Exit button: clicking this should end the project
- the form load event: when starting the project
  - all TextBoxes should be cleared
  - the focus should be given to one of the TextBoxes (so that the blinking cursor in this box invites the user to enter text into it)
  - all ListBoxes should have an item selected as default

The btnExit Object

Let us start with the object with the shortest code: the Exit button, which is named btnExit.

All buttons are automatically given a name starting with"btn" followed by the text of the button (with blanks removed, if any).
The code for the button is called **btnExit_Click**.

The code for an object is identified by its name, connected by “_” (underscore or underline) to the name of the method (or action), such as click.

The actual code showing in your View Code for Objects panel is:

```vbnet
Private Sub btnExit_Click
  End
End Sub
```

This is the **Click** method for the **btnExit** object, and its contents is the **End** instruction, which is exactly what the project does when you click on **btnExit**. Visual Basic could have used other similar words (such as quit or stop), but **End** is the only approved keyword to accomplish this task. When you write your own projects and type in the **End** instruction, it will automatically be highlighted in blue as a keyword.

What about **Private Sub** and **End Sub**? These denote the beginning and the end of a **sub** program, or set of one or more instructions that apply just to the object and the method or action named in the **Private Sub** line. The good news is that you never have to type these when writing projects. In **Visual Basic 2005 Express**, you will select your object and method from pull-down lists, and it automatically supplies these lines and positions the cursor between them, ready for your instructions!
KEYWORDS

End is a keyword in Visual Basic. Words such as this will be highlighted in blue when we use them in Visual Basic. When the programmer types a keyword, IntelliSense changes the color to blue automatically.

The Form1 Object

The form, or entire window is also considered an object, with name Form1. The method that we use for it is not Click, but Load. The Load event occurs when you start the project running. Whatever code you put here automatically is executed by the computer before you click anything! When writing code in Visual Basic 2005 Express, you will select Form1 events and Load from pull-down lists. According to our rule, the code for the form is called Form1_Load.

The actual code showing in your View Code for Objects panel is:

Private Sub Form1_Load

'Clear the TextBox and give it the Focus

txtAdvertise.Clear()

txtAdvertise.Focus()

'Preselect the heading for the ListBox

lstChooseOne.SelectedItem = "Choose One"

End Sub
COMMENT LINES

Lines that begin with a quote are comment lines. Lines such as this will be highlighted in green when we use them in Visual Basic. When the programmer types a single quote and then an ordinary English phrase or sentence, IntelliSense changes the color to green automatically. They are in the form file to make the project more understandable.

CHARACTER STRINGS

Character strings begin and end with quotes. Strings such as this will be highlighted in red when we use them in Visual Basic. When the programmer types a quote, IntelliSense changes the color to red automatically.

On the following page, we see how TextBoxes and ListBoxes are named.
All **TextBoxes** are given a name starting with "**txt**" followed by the text of the TextBox (with blanks removed, if any).

All **ListBoxes** are given a name starting with "**lst**" followed by the first item of the ListBox (with blanks removed, if any).

**NOTE:** Text for an object is not allowed to be blank because it is needed to form the name of the object. If text is entered as blank or empty, the current text is retained.

**NOTE:** Text may include letters, digits, and underscore ( _ ). All other characters, including blank, are removed automatically.

**NOTE:** if text for an object would result in two objects with the same name, it is not accepted. *If the object names differ only in capitalization, such as txtCD and txtCd, this is not accepted either.*

The content of this code includes two instructions. In these instructions, we use the name of an object, connected by a dot to a method (action) or property.

The first instruction `txtAdvertise.Clear()` executes the **Clear** method for the **txtAdvertise** TextBox. The parentheses ( ) distinguishes a method from a property.

The second instruction `lstChooseOne.SelectedItem = "Choose One"` is an assignment statement. The equals sign means that the expression on the right is assigned to the **SelectedItem** property of the **lstChooseOne** ListBox.on the left. **Assignment statements always go from right to left!** This **preselects** and **highlights** the first item of the **ListBox**, which really serves as a heading for the second and third items “**Advertise**” and “**Clear**”.

"Page 18"
Thus, there are two ways for the code of one object to refer to another object:

Name of the object, followed by a dot, followed by a method

Name of the object, followed by a dot, followed by a property

NOTE: when we later write code ourselves using Visual Basic 2005 Express, all we have to do is type the name of the object, followed by a dot, and a list of methods and properties will display and we choose the one we want. This time-saving feature is called IntelliSense.

We have also discovered the answer to a common question that beginners have:

Where does code go: in the object causing the action or the object receiving the action?

The code must go in the object CAUSING the action, NOT in the object RECEIVING the action.

The lstChooseOne Object

The actual code showing in your View Code for Objects panel is listed on the following page.
Private Sub lstChooseOne_Click
    'Declare string variables
    Dim sentence1, sentence2 As String

    If lstChooseOne.SelectedItem = "Advertise" Then
        'Assign strings of characters to the sentences
        sentence1 = "Special Sale Today - 20% Off! "
        sentence2 = "Voice/Internet Plans - 30% Off!"
        'Put the sentences into the TextBox
        txtAdvertise.Text = sentence1 & sentence2
    End If

    If lstChooseOne.SelectedItem = "Clear" Then
        'Clear the TextBox and give it the Focus
        txtAdvertise.Clear()
        txtAdvertise.Focus()
    End If

End Sub

Sentence1 and sentence2 are called variable names, given by the programmer, not special words built into the Visual Basic system.
Variables can be used to store numbers or, in this case, to store strings of characters. The keyword “Dim” followed by the variables followed by “as string” tell the computer to set up the dimensions of these variables in the computer’s memory (not displayed as objects on the screen) to store strings.

The If statement acts exactly as it would in plain English. The format of an If statement is

```
If condition Then
    Statements to be executed if the condition is true
End If
```

When we practice writing the code for the project in Visual Basic 2005 Express, we type the If line and press Enter after Then. The End If line automatically appears below, and room for the statements, indented for readability, appears between the If and End If.

In this case, the condition uses the equals sign as a comparison operator to compare `lstChooseOne.SelectedItem` to “Advertise” or “Clear”. Thus, we now are using another property:

```
SelectedItem is a property of ListBoxes, and it really is the item selected by the user in clicking within a ListBox.
```

A quote symbol is put at the beginning and at the end of a character string. The equal sign stands for the assignment operator. What happens is the right hand side is assigned to the left hand side in an instruction using the assignment operator.

The expression `sentence1 & sentence2` illustrates the concatenation operator &, which puts two character strings together. The result is then assigned to the text property of the TextBox
The reason that the text fits into the dimensions of this TextBox is that the **MultiLine** property of this TextBox has been set to be true. It behaves like word wrap in a word processor. Note that the space at the end of Sentence1 keeps it from running directly into Sentence2.

**Focus** is not a property of txtAdvertise but rather a method, like **Clear**, acting upon it. What it does is shift the focus to this TextBox. What this means is that the blinking vertical insertion symbol will appear in the TextBox – inviting the user to enter text there.

---

**Practice Building the Project**

You will use **VBDesigner** to choose the objects and their properties, and then use **Visual Basic 2005 Express** to write the code.

---

**1. Design (Objects And Properties)**

<table>
<thead>
<tr>
<th>For Object Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>1A Phone Sale Demo Folder</td>
</tr>
<tr>
<td>Executable Phone Sale File</td>
</tr>
</tbody>
</table>

**To Design**

**VBDesigner – Step 1 (Later Step 2 If Needed)**

| 1B Phone Sale Build Folder |
FILES IN THE LAB FOLDER

In your lab folder, there are 3 folders and numerous files. Here is a summary.

FOLDERS: DO NOT USE!!!

Bin, My Project, Obj

: Do not use any of these – even with VBDesigner!!!

FILES

(There are 6 files, but only 3 of them are to be used)

The project file: the name ends in .vbproj – after using VBDesigner, open this to do the coding in Visual Basic 2005 Express.

The design and code files: these are always called

Form1.Designer.vb (design file)

Form1.vb (code file)

For projects to be graded for credit (labs and quizzes), submit BOTH of these to your instructor as directed.

If you point the mouse - without clicking - over an object in the Executable Phone Sale file, Visual Basic 2005 Express’s tooltip
feature will display – for a short time – the type of object and its key properties. Use this as your guide in building your own project.

Now – to build your own project, run VBDesigner – Step 1. Open a Phone Sale objects file in the 1B Phone Sale Build Folder. In the left panel (ToolBox and Properties), you can

- select Labels, TextBoxes, ListBoxes, and Buttons
- move them to the right panel (your project form or window)
- if you accidentally choose the wrong object, move it back past the vertical black line from the right panel (your project) to anywhere in the left panel (ToolBox & Properties) – this removes it from your project
- for each object, enter text and choose colors, font, and size
- for ListBoxes, enter text for each desired row and leave unused row numbers alone – they will be discarded when you deploy your objects to the project
- if you accidentally enter text into a row of a ListBox, simply reenter the row number to get rid of the text
- for the TextBox, keep the Read/Write setting so that text can be entered when running the project

You exit by clicking the Exit Button.

If you do not have time to complete your design, you can run VBDesigner – Step 2 at a later time. You will have to reopen your objects file and then you can continue.

HOW MANY OBJECTS ARE THERE?

When you move objects from the left ToolBox and Properties panel to the right panel, there are additional objects underneath. In fact, there are stacks of 10 Labels, 10 TextBoxes, 5 ListBoxes, and 5 Buttons!
2. Code (Instructions)

To Code

1B Phone Sale Build Folder
Project File
This Opens Visual Basic 2005 Express

To Review Code Needed
Read the Description Below

Open the project file. Your Project window is always called Form1.vb – if you cannot find it:

- click on View Menu – Solution Explorer
- double-click Form1.vb in Solution Explorer
- if any other windows ARE open, close all of them with the X button for each window to be closed

You should now only have open the form design window for Form1.vb.
In order to write code for your objects, you need to open the form code window for **Form1.vb**:

Click on **View Menu – Code**

If you ever need to return to the design window:

Click on **View Menu – Design**

After first opening the code window, you will have **tabs** for both the code and design windows – you can use these as shortcuts.

In the code window, there are two pull-down lists indicated by the ▼ **button**. They identify themselves by pointing to them. The one on the left is called **Class Names** and has all of your designed objects. The one on the right is called **Method Names** and lists various methods or actions or code (such as **Click** and **Load**) to be written for the objects.

To view the code that you will enter, run **VBDesigner – Step 3**. Log onto the objects file in the **1A Phone Sale Demo Folder**.

Click on each object in the right panel (**Phone Sale**) and observe its code in the left panel (**View Code for Objects**).

### The btnExit Object

All projects have **Exit** code. Therefore, let us begin by pulling down the objects, selecting **BtnExit**, pulling down the methods, and selecting **Click**. What we get is:
Private Sub btnExit_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles btnExit.Click
End Sub

The code indicated in VBDesigner by clicking on the Exit button was:

Private Sub btnExit_Click  
End  
End Sub

Why the difference in the Private Sub line?

The expression (ByVal sender As Object, ByVal e As System.EventArgs) is called a parameter list, only to be used by experienced programmers for advanced features. It should be left in your code, but for simplicity we do not list it in VBDesigner.

The expression Handles.btnExit.Click means that this code handles the event that arises when the user clicks btnExit. It should be left in your code, but for simplicity we do not list it in VBDesigner.

Your cursor should be located indented on a blank line between Private Sub and End Sub. It is inviting you to type the code, which is simply the word End. If you type it correctly, even in lower case, the E will be made upper case and the word will be highlighted in blue to indicate a keyword.

The Form1 Object

All projects have Form1_Load code. When we begin by pulling down the objects, we see two possible choices: Form1 and (Form1 Events).
Make the second choice **(Form1 Events)** because **Load** is an event that occurs when you start the project and the form is loaded into the computer’s memory – before the user does anything. Now pull down the methods and select **Load – not Click**. What we get is:

```vbnet
Private Sub Form1_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles MyBase.Load
    End Sub
```

The code indicated in **VBDesigner** by clicking on the form was:

```vbnet
Private Sub Form1_Load

    'Clear the TextBox and give it the Focus
    txtAdvertise.Clear()
    txtAdvertise.Focus()

    'Preselect the heading for the ListBox
    lstChooseOne.SelectedItem = "Choose One"

    End Sub
```

Your cursor should be located indented on a blank line between **Private Sub** and **End Sub**. It is inviting you to type the comments and 3 lines of code. Recall what we learned in the **Reviewing The Code** section. There are two ways for the code of one object to refer to another object:

- **Name of the object, followed by a dot, followed by a method**
- **Name of the object, followed by a dot, followed by a property**
NOTE: when we later write code ourselves using Visual Basic 2005 Express, all we have to do is type the name of the object, followed by a dot, and a list of methods and properties will display and we choose the one we want. This time-saving feature is called IntelliSense.

So – use IntelliSense to make your coding much easier! If the IntelliSense list of methods and properties does not appear after your dot, you did not type the object’s name correctly. To select from the list, you can type enough of the method or property to get to it and press one of:

- Enter for a method (it then goes to the next line)
- An = sign for a property (it stays on the same line)

Sometimes IntelliSense even guesses what you want!

**The lstChooseOne Object**

Pull down lstChooseOne on the left and pull down Click on the right. What we get is:

```vbnet
Private Sub lstChooseOne_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles lstChooseOne.Click
End Sub
```

The code indicated in VBDesigner by clicking on the ListBox was:
Private Sub lstChooseOne_Click

'Declare string variables
Dim sentence1, sentence2 As String

If lstChooseOne.SelectedItem = "Advertise" Then

'Assign strings of characters to the sentences
sentence1 = "Special Sale Today - 20% Off! "
sentence2 = "Voice/Internet Plans - 30% Off!"

'Put the sentences into the TextBox
txtAdvertise.Text = sentence1 & sentence2

End If

If lstChooseOne.SelectedItem = "Clear" Then

'Clear the TextBox and give it the Focus
txtAdvertise.Clear()
txtAdvertise.Focus()
lstChooseOne.SelectedItem = "Choose One"

End If

End Sub
Your cursor should be located indented on a blank line between `Private Sub` and `End Sub`. It is inviting you to type all of the lines of code. Most projects have a section requiring a lot of keyboarding, such as we have here. **However, use IntelliSense to make it easier!** IntelliSense not only responds to the dot with a list of methods and properties, but it also automatically puts in the `End If` statement after you finish the `If ... Then` line and indents between them for you! It also automatically puts your comment lines in green – all you have to do is type the quote and then the comment in English.

Now we are finally ready to try out the code that you have written!

---

**THE RUN WINDOW**

You are always either designing, coding, or running your project. This is the third main window. Use the **Debug Menu & Start command** or the ▶ button to run the Project. When the project ends, the code window returns. If the project does not successfully end, the ■ button displays. **The code cannot be edited until this ■ button is clicked to stop the project!**

If your project will not run, due to statements not allowed in **Visual Basic 2005 Express**, a MessageBox will tell you that there are build errors and ask if you want to continue. You should click the No button.

A list of error messages will display at the bottom of your screen. Sometimes they are understandable (such as a missing parenthesis) and sometimes not. Simply click the error message. It will then go to the line of code where the error is located and display a wavy blue line under the problem, similar to a spell-checker in word processing.

Occasionally, in running your project, you might observe that you forgot something in **VB Designer** when you were designing properties for your objects. There are 2 ways to fix this problem – reuse VB Designer or use the Properties window in **Visual Basic 2005 Express**.
REUSE VBDESIGNER

Reuse VBDesigner to go back and review the design of the objects. Make any necessary changes and save them. Back in Visual Basic 2005 Express you will get a MessageBox saying:

“This file has been modified outside of the source editor. Do you want to reload it?”

You should click the Yes button and return to work on the code.

THE PROPERTIES WINDOW

This is to be used only if needed for adjustments to your design. While in the design window, click on View Menu – Properties. To see the properties of an object, click on the object and look in the properties window. You will find all of the properties that you designed, as well as many more that come with the system.

Just as with word processing, it is important to save your work frequently.

SAVING YOUR PROJECT

Click the Save All button (its icon is a stack of disks). Another way to do this is to click on File Menu – Save All.

You will always enter your code between Private Sub and End Sub lines given to you by pulling down and selecting the object and the method or action. However, if you scroll to the top of the code window, you will discover that Visual Basic 2005 Express has already put a great deal of built-in code, hidden from view by a + Windows Form Designer Generated Code line. You should be warned never to make any changes to this, as indicated on the following page:
VISUAL BASIC 2005 EXPRESS BUILT-IN CODE – WARNING!

At the top of your code window, you will see:

Public Class Form1
    Inherits System.Windows.Forms.Form

+ Windows Form Designer Generated Code

and at the bottom, you will see:

End Class

Never remove or change any of these lines!!!

Always enter your methods (code) after the + windows line and before the End Class line

If you click on the + sign to expand the built-in code, you will see all that comes with the system! Then click on the – sign to collapse all of this code to concentrate on your own!
Lab Exercise 1 (Optional: No Credit)

For Object Properties & Sample Run

Lab Exercise 1 Folder

Executable Lab Exercise 1 File

To Design

VBD Benefactor – Step 1 (Later Step 2 If Needed)

Lab Exercise 1 Folder-

NOT MY PROJECT SUBFOLDER!!!

To Code

Lab Exercise 1 Folder

Project File

This Opens Visual Basic 2005 Express

Build a project for President’s Day. If you click on “Washington”, it displays one sentence, and if you click on “Lincoln”, it displays another sentence, as illustrated by the demo screens on the following page.
Abraham Lincoln was the best President of the United States!

George Washington was the first President of the United States!
Lab Assignment 1 (Credit)

For Object Properties & Sample Run

1C Lab Folder
Executable Computer Sale File

To Design

VBDesigner – Step 1 (Later Step 2 If Needed)
1C Lab Folder- NOT MY PROJECT SUBFOLDER!!!

To Code

1C Lab Folder
Project File

This Opens Visual Basic 2005 Express

Build a project called Computer Sale, which is similar to the phone store project Phone Sale, except that it is used by a computer store. It has an Aqua background. It has two TextBoxes instead of using the same TextBox for two different notes. For both, ReadOnly=False, so that text can be entered into either one. The focus begins with the first TextBox. There is a ListBox with three items below a heading Pick One:
Quick Guide to Visual Basic Express

- CD-ROM – this displays a two-sentence sale note in the first TextBox
- DVD - this displays a different two-sentence sale note in the second TextBox
- Clear – this clears both TextBoxes and returns the Focus back to the first TextBox

We show some demo screens as follows..
SALE!!!

We are having a great CD-ROM sale! Discount of 25% off!

We are having a great DVD sale! Discount of 20% off.

Pick One

CD-ROM

Pick One

DVD

Page 38
You should do the following steps:

- Run the **Executable Computer Sale file** in the **1C Lab folder**
  - Pointing to the objects lists their properties
- In **VBDesigner**, build your objects file in the **1C Lab folder**
  - Make sure the objects and properties match
- Use **Windows Explorer** to open the **project file** in the **1C Lab folder**
- Now you are in **Visual Basic 2005 Express** and must pull down objects and methods to write the code
  - There is **NO** code for you to copy this time, or the lab credits would be given to you without having to think about the solution!
  - You do have the already running **Executable Computer Sale file** to serve as your guide – it will show you **WHAT** it should do, but not **HOW** it is done.
  - The demo screens below will also show you **WHAT** it should do.
  - Refer in this Guide to the **Phone Sale** description to see how **similar** actions are programmed, such as declaring sentences and putting them into TextBoxes, clearing TextBoxes, etc. **Be aware that the object names and sentences themselves will be different!**
  - For this lab, your instructor **CAN** help you, but refer to this Guide first!
  - **NOTE**: for a quiz, your instructor **CANNOT** help you – that is why the best way to prepare for a quiz is to understand fully the demo and the lab projects.
Lab Assignment 1: Comments

To Submit the Project In Blackboard

• In the Blackboard menu, click on VB Quiz & Lab, or whatever name is specified by your instructor
• Enter a comment, such as the project name
• Click on browse to locate your 1C Lab Folder, Form1.Designer.vb file
• Click on Add Another File
• Click on browse to locate your 1C Lab Folder, Form1.vb file
• Click on Add Another File
• Click on Submit
• Your instructor’s comments and grade will show up in User Tools – View Grades

Note to Instructors

To Grade the Project in Blackboard

To grade student projects (for this and the other labs):

• Open a new project called Lab
• In Blackboard, save the student files Form1.Designer.vb and Form1.vb in Lab for each student
• Except for the first student, VB will say that files have been modified – do you wish to reload?
• Click Yes to get the next student’s project.
CHAPTER 2: CALCULATIONS, MESSAGE BOXES

What the Project Does

For Object Properties & Sample Run

2A Phone Quote Demo Folder:

Executable Phone Quote File

The project is a price quote by a fictional phone store. It begins with the insertion bar in the TextBox next to the Name Label. The salesperson types in the name of the customer and then presses the tab key or clicks to go to the TextBox next to the Rebate Label. The salesperson types in a rebate. The customer then makes a selection from the Voice Plan ListBox and a selection from the Internet Plan ListBox. The costs of these selections are placed in the TextBoxes labeled Voice Price and Internet Price. Clicking on the compute button will add the voice price to the internet price, subtract the value of the rebate to get a subtotal, multiply by 6% to get a sales tax, and add the tax to get a final total. NOTE: this total is not formatted for currency. We will concern ourselves with such formats when we get to the third demo project (Phone Pay) in this guide. Finally, the Exit button is self explanatory.

Try selecting 250 minutes voice and high-speed internet. These selections may both be clicked, but the phone store does not permit the
best internet connection for the lowest voice connection. This is detected when you click the Compute button. It uses an IF statement to detect the fact that both of these selections have been made and then uses a MsgBox (Message Box) statement to put a message box window on the screen, with the message "250 minutes has no high-speed internet". The OK button in the Message Box must be pressed. Then the Compute button finishes by clearing out everything, including the ListBoxes (reset to their headings), except for the name and rebate value.

In summary, you may do the following:

- enter a **name** in the first TextBox (the focus starts here)
- tab or click to the second TextBox to enter a **rebate** value
- select the **voice plan**
- select the **internet plan**
- **compute** the price quote

Try these actions! When you are done, we are ready to learn more about how this project works. We show some demo screens on the following pages.
<table>
<thead>
<tr>
<th>Name</th>
<th>Voice Plan</th>
<th>Internet Plan</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith</td>
<td>200 Minutes</td>
<td>Low Speed</td>
<td>500 Minutes</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>High Speed</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Voice Plan</th>
<th>Internet Plan</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith</td>
<td>300 Minutes</td>
<td>Low Speed</td>
<td>550 Minutes</td>
</tr>
<tr>
<td></td>
<td>45</td>
<td>High Speed</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Reviewing the Design (Properties)

For Object Properties
VBD组团ner – Step 2
2A Phone Quote Demo Folder: Objects File
or

For Object Properties
2A Phone Quote Demo Folder
Executable Phone Quote file

Click on each object in the right panel (Phone Quote) and observe its properties in the left panel (ToolBox and Properties).

For comparison, you can simultaneously run the Executable Phone Quote file and look at the Tool Tips. If you point the mouse - without clicking - over an object, Visual Basic 2005 Express’s tooltip feature will display – for a short time – the type of object and its key properties.
LINES

There are two lines in the form; is the line a new type of object? No, the line is simply a label that has been sized with the smallest height possible and with BackColor=Black, ForeColor=Black! The text does not matter because it will not show anyway. Other colored lines could be built in a similar manner.

Reviewing the Code (Instructions)

VBDesigner – Step 3

2A Phone Quote Demo Folder: Objects File

Click on each object in the right panel (Phone Quote) and observe its code in the left panel (View Code for Objects). Click on each object in the right panel (Phone Quote) and observe its code in the left panel (View Code for Objects). Note that Labels and TextBoxes do not have code. Labels are not involved in the code in any way, but TextBoxes sometimes are involved in the code of ListBoxes and Buttons.

Let us try to understand what the code for each object means and how it works. All of our projects will always have code for
• the **Exit** button: clicking this should end the project
• the **Form1_Load** event: when starting the project
  o all TextBoxes should be cleared
  o the focus should be given to one of the TextBoxes (so that the blinking cursor in this box invites the user to enter text into it)
  o all ListBoxes should have an item selected as default

**The btnExit Object**

Always start with this object – it has the shortest code.

```vbnet
Private Sub btnExit_Click
    End
End Sub
```

**The Form1 Object**

The actual code showing in your **View Code for Objects** panel is listed on the following page.
Private Sub Form1_Load
    'Clear all TextBoxes
    txtName.Clear()
    txtRebate.Clear()
    txtVoicePrice.Clear()
    txtInternetPrice.Clear()
    txtSubtotal.Clear()
    txtTax.Clear()
    txtTotal.Clear()
    'Put the Focus on the Name TextBox
    txtName.Focus()
    'Preselect the headings for the ListBoxes
    lstVoicePlan.SelectedItem = "Voice Plan"
    lstInternetPlan.SelectedItem = "Internet Plan"
End Sub
The lstVoicePlan Object

The actual code showing in your View Code for Objects panel is:

Private Sub lstVoicePlan_Click
    'Assign prices
    If lstVoicePlan.SelectedItem = "250 Minutes" Then
        txtVoicePrice.Text = 20
    End If
    If lstVoicePlan.SelectedItem = "500 Minutes" Then
        txtVoicePrice.Text = 30
    End If
End Sub

(These are assignment statements that put values into texts.)

The lstInternetPlan Object

The actual code showing in your View Code for Objects panel is listed on the following page.
Private Sub lstInternetPlan_Click

'Assign prices
If lstInternetPlan.SelectedItem = "Low Speed" Then
    txtInternetPrice.Text = 15
End If

If lstInternetPlan.SelectedItem = "High Speed" Then
    txtInternetPrice.Text = 25
End If

End Sub

(These are also assignment statements that put values into texts.)

The btnCompute Object

The actual code showing in your View Code for Objects panel is listed on the following page.
Private Sub btnCompute_Click

If lstVoicePlan.SelectedItem = "250 Minutes" And
    lstInternetPlan.SelectedItem = "High Speed" Then
    MsgBox("250 minutes has no High Speed Internet")
    'Clear all TextBoxes except Name and Rebate
    txtVoicePrice.Clear()
    txtInternetPrice.Clear()
    txtSubtotal.Clear()
    txtTax.Clear()
    txtTotal.Clear()
    'Reset headings for ListBoxes
    lstVoicePlan.SelectedItem = "Voice Plan"
    lstInternetPlan.SelectedItem = "Internet Plan"
Else
    'Calculations
    'Assume sales tax of 6% = .06
    txtTax.Text = Val(txtSubtotal.Text) * 0.06
End If

End Sub

The condition used in this If statement is a compound condition connected by the word **And**. It is used to check whether both 250 minutes and High Speed have been selected.

The message box statement has the form

MsgBox ("desired message goes between parentheses and quotes")
and causes a window with this message and an OK button to appear on the
screen. The user reads the message and then clicks on the OK button to
continue the project.

Thus, the **Compute** code checks to see if the user has checked the lower-
priced voice plan (250 minutes) and the higher priced internet plan (High
Speed).

**If so**, the code between **If** and **Else** is executed. First, the message "250
minutes has no High Speed Internet" appears in a window. After the user
clicks on the OK button, all the numbers except for the rebate are cleared,
and the ListBoxes are cleared in that the SelectedItem is set back to the
heading row.

**If not**, the code between **Else** and **End If** is executed. The calculations
take place here. The subtotal is gotten by adding the voice price and the
internet price and then subtracting the rebate. The subtotal is multiplied by
the tax rate (5% in this example) to get the sales tax. The total is gotten by
adding the sales tax and the subtotal.

**Val**: Whenever writing an arithmetic expression involving objects in the
form, **Val** (short for Value) is used for each such object. **Val** converts the
text in an object to its value in an arithmetic format. This is done by the
following: **Val(name of object.text)**. The arithmetic expression on the
right side of the equals (=) sign is then assigned to the text property of the
object on the left side of the equals (=) sign as follows:

| name of result object.text = arithmetic expression involving various values Val(name of object.text). |

This form of the **If** statement has the word **Else** on a line by itself in the
middle, and it acts exactly as they would in plain English. This format of
an **If** statement is as follows.
If condition Then

Statements to be executed if the condition is true

Else

Statements to be executed if the condition is false

End If

Recall that when we write the code for the project in Visual Basic 2005 Express, we type the If line and press Enter after Then. The End If line automatically appears below, and room for the statements, indented for readability, appears between the If and End If. We have to insert the word Else in the appropriate location, on a line by itself, and the indentation is automatically set up.

Practice Building the Project

You will use VBDesigner to choose the objects and their properties, and then use Visual Basic 2005 Express to write the code.
1. Design (Objects And Properties)

For Object Properties

2A Phone Quote Demo Folder

Executable Phone Quote File

To Design

VBDesigner – Step 1 (Later Step 2 If Needed)

2B Phone Quote Build Folder

If you point the mouse - without clicking - over an object in the Executable Phone Quote file, Visual Basic 2005 Express’s tooltip feature will display – for a short time – the type of object and its key properties. Use this as your guide in building your own project.

Now – to build your own project, run VBDesigner – Step 1. Open a Phone Quote objects file in the 2B Phone Quote Build Folder. In the left panel (ToolBox and Properties), you can

- select Labels, TextBoxes, ListBoxes, and Buttons
- move them to the right panel (your project form or window)
- for each object, enter text and choose colors, font, and size
• for ListBoxes, enter text for each desired row and leave unused row numbers alone – they will be discarded when you deploy your objects to the project
• if you accidentally enter text into a row of a ListBox, simply reenter the row number to get rid of the text
• for the txtName and txtRebate TextBoxes, keep the Read/Write setting so that text can be entered when running the project
• for all of the other TextBoxes, click on the ReadOnly setting – the user cannot enter anything but instead the numbers are put into these TextBoxes by the code of other objects

ReadOnly Property

TextBoxes start out with the Read/Write property, so that users can click on them and type into them, and code of other objects can put data into them. Changing to the ReadOnly property ensures that the TextBoxes can only be filled by pre-programmed code of other objects.

The BackColor of ReadOnly TextBoxes is overridden by a LightGray BackColor, and the contents are aligned on the right edge (right-aligned).

Read/Write TextBoxes can have any BackColor, and the contents are aligned on the left edge (left-aligned).

You exit by clicking the Exit Button.

If you do not have time to complete your design, you can run VBDesigner – Step 2 at a later time. You will have to reopen your objects file and then you can continue.
2. Code (Instructions)

To Code

2B Phone Quote Build Folder

Project File
This Opens Visual Basic 2005 Express

To Review Code Needed

Read “Reviewing the Code” in this Chapter

Open the project file. Your Project window is always called Form1.vb – if you cannot find it:

- click on View Menu – Solution Explorer
- double-click Form1.vb in Solution Explorer
- if any other windows ARE open, close all of them with the X button for each window to be closed

You should now only have open the form design window for Form1.vb.
In order to write code for your objects, you need to open the form code window for **Form1.vb**:

Click on **View Menu – Code**

If you ever need to return to the **design** window:

Click on **View Menu – Design**

After first opening the code window, you will have **tabs** for both the code and design windows – you can use these as shortcuts.

To view the code that you will enter, run **VBDesigner – Step 3**. Log onto the objects file in the **2A Phone Quote Demo Folder**.

Click on each object in the **right** panel (**Phone Quote**) and observe its code in the **left** panel (**View Code for Objects**).

Now go back and forth between **VBDesigner** and **Visual Basic 2005 Express** to transfer the code:

- click on an object in **VBDesigner** to view its code
- pull down the object and action in the **Visual Basic 2005 Express** code window and enter its code
- repeat this for each object that has code

Remember to save your work frequently!

Finally, try out the code that you have written and debug if necessary.
Lab Exercise 2 (Optional: No Credit)

For Object Properties & Sample Run
Lab Exercise 2 Folder
Executable Lab Exercise 2 File

To Design
VBD designer – Step 1 (Later Step 2 If Needed)
Lab Exercise 2 Folder-
NOT MY PROJECT SUBFOLDER!!!

To Code
Lab Exercise 2 Folder
Project File

This Opens Visual Basic 2005 Express

Build a project which is a dividing calculator – for simplicity, division is the only operation. You enter two numbers and then click on the Divide button to get the answer. If you accidentally divide by 0, a MessageBox should display and then the 0 and the previous answer should be cleared, as illustrated by the demo screens on the following pages.
Lab Assignment 2 (Credit)

For Object Properties & Sample Run
2C Lab Folder
Executable Computer Quote File

To Design
VBDesigner – Step 1 (Later Step 2 If Needed)
2C Lab Folder-NOT MY PROJECT SUBFOLDER!!!

To Code
2C Lab Folder
Project File
This Opens Visual Basic 2005 Express

Build a project called Computer Quote, which is similar to the phone store project Phone Quote, except that it is used by a computer store in order to calculate a customer order. It has a Yellow background. It has eight TextBoxes, six of which are ReadOnly; the other two can be entered: one for the quantity of CD-ROMs (at $7.99 each) and one for the quantity of DVDs (at $13.99 each). Focus starts with these two
TextBoxes. The customer then makes a selection from the **Choose One** ListBox and a selection from the **Discount** ListBox.

The project should multiply the number of CD-ROMs and DVDs by the unit prices listed to calculate a subtotal and then subtract a 20% discount if a discount has been selected. **However, it should use an IF statement with a message box in order NOT to permit a discount for a credit purchase!**

**NOTE: no formatting for currency with 2 decimal places is required.**

We show some demo screens on the following pages.
<table>
<thead>
<tr>
<th>Item</th>
<th>Quantity</th>
<th>Unit Price</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD-ROM 1</td>
<td>2</td>
<td>15.98</td>
<td>31.96</td>
</tr>
<tr>
<td>DVD 2</td>
<td>3</td>
<td>41.97</td>
<td>125.91</td>
</tr>
<tr>
<td>DVD 3</td>
<td>2</td>
<td>57.95</td>
<td>115.90</td>
</tr>
<tr>
<td><strong>Subtotal</strong></td>
<td></td>
<td></td>
<td><strong>46.96</strong></td>
</tr>
</tbody>
</table>

**Discount:** 20%

**Total:** 115.90
You should do the following steps:

- Run the **Executable Computer Quote file** in the **2C Lab folder**
  - Pointing to the objects lists their properties

- In **VBDesigner**, build your objects file in the **2C Lab folder**
  - Make sure the objects and properties match

- Use **Windows Explorer** to open the **project file** in the **2C Lab folder**

- Now you are in **Visual Basic 2005 Express** and must pull down objects and methods to write the code
  - There is **NO** code for you to copy this time, or the lab credits would be given to you without having to think about the solution!
  - You do have the already running **Executable Computer Quote file** to serve as your guide – it will show you **WHAT** it should do, but not **HOW** it is done.
  - The demo screens below will also show you **WHAT** it should do.
  - Refer in this Guide to the **Phone Quote** description to see how similar actions are programmed. **Be aware that the object names and sentences themselves will be different!**
  - For this lab, your instructor **CAN** help you, but refer to this Guide first!
  - **NOTE**: for a quiz, your instructor **CANNOT** help you – that is why the best way to prepare for a quiz is to understand fully the demo and the lab projects.
Lab Assignment 2: Comments

To Submit the Project in Blackboard

- In the Blackboard menu, click on VB Quiz & Lab, or whatever name is specified by your instructor
- Enter a comment, such as the project name
- Click on browse to locate your 2C Lab Folder, Form1.Designer.vb file
- Click on Add Another File
- Click on browse to locate your 2C Lab Folder, Form1.vb file
- Click on Add Another File
- Click on Submit
- Your instructor’s comments and grade will show up in User Tools – View Grades
CHAPTER 3: LISTS, LOOPS, FORMATTING

What the Project Does

For Object Properties & Sample Run

3A Phone Pay Demo Folder:

Executable Phone Pay File

The project is a total contract payment calculation by a fictional phone store. The salesperson enters the Startup amount in the TextBox, where the focus starts. The term of the contract (number of years) can be selected by means of a list stored in a ListBox. The monthly charge can be selected by means of another list stored in a ListBox. Clicking on the compute button calculates the total contract payment (formatted for currency with 2 decimal places) and puts it in a TextBox.

There are three differences between these ListBoxes and those of the previous chapters:

- The vertical scroll bar for a ListBox appears automatically if the list does not fit in the size of the ListBox.
- Labels are used for headings for these ListBoxes so that they will not scroll out of the ListBox.
• There are many items in these ListBoxes so they are not added in the design phase. Instead, we use loops in the `Form1_Load` code to add these items.

In summary, you may do the following:

• enter a startup charge
• enter the term of the contract by selecting from a list
• enter a monthly charge by selecting from a list
• click on the compute button to calculate and display the total contract payment

Try these actions! When you are done, we are ready for some terminology so that we can learn how this project works. We show some demo screens on the following page.
Reviewing the Design (Properties)

For Object Properties
VBDesigner – Step 2
3A Phone Pay Demo Folder: Objects File

or

For Object Properties
3A Phone Pay Demo Folder
Executable Phone Pay File

Click on each object in the right panel (Phone Pay) and observe its properties in the left panel (ToolBox and Properties).

For comparison, you can simultaneously run the Executable Phone Pay file and look at the Tool Tips. If you point the mouse - without clicking - over an object, Visual Basic 2005 Express’s tooltip feature will display – for a short time – the type of object and its key properties.
LISTBOX ITEMS

Notice that the items that appear when you run the Executable Phone Pay file do NOT appear in VBDesigner, which just has the top item in each ListBox.

The top item is used to form the name in VBDesigner, just as the text in a TextBox is used. DO NOT forget to enter this text! The label above the ListBox serves as the heading for this kind of list.

In Visual Basic 2005 Express, the code is used to clear this item and then add the desired items in each ListBox.

Reviewing the Code (Instructions)

VBDesigner – Step 3

3A Phone Pay Demo Folder: Objects File

Click on each object in the right panel (Phone Pay) and observe its code in the left panel (View Code for Objects). Click on each object in the right panel (Phone Pay) and observe its code in the left panel (View Code for Objects). Note that Labels and TextBoxes do not have code. Labels are not involved in the code in any way, but TextBoxes sometimes are involved in the code of ListBoxes and Buttons.

Let us try to understand what the code for each object means and how it works. All of our projects will always have code for
• the **Exit** button: clicking this should end the project
• the **form load** event: when starting the project
  o all **TextBoxes** should be cleared
  o the focus should be given to one of the **TextBoxes** (so that the blinking cursor in this box invites the user to enter text into it)
  o all **ListBoxes** should have the top item cleared because it is used for the name but not as a heading
  o all **ListBoxes** should have all desired items added
  o all **ListBoxes** should have an item selected as default

: **The btnExit Object**

Always start with this object – it has the shortest code.

    Private Sub btnExit_Click
        End
    End Sub

: **The Form1 Object**

The actual code showing in your **View Code for Objects** panel is listed on the following page.
Private Sub Form1_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles MyBase.Load

    'Clear TextBoxes
    txtStartupCharge.Clear()
    txtTotalPayment.Clear()
    'Clear ListBox headings before putting list items in
    lstMonthlyCharge.Items.Clear()
    lstYears.Items.Clear()

    'Declare loop variables for lists
    Dim monthlyCharge As Decimal '2 decimal places to be used
    Dim years As Integer 'whole numbers

    'Add formatted monthly charges to their ListBox
    For monthlyCharge = 10 To 20 Step 0.5
        lstMonthlyCharge.Items.Add(Format(monthlyCharge, 2))
    Next

    'Add formatted years to their ListBox
    For years = 1 To 5
        lstYears.Items.Add(Format(years, 0))
    Next

    'Preselect default items in lists
    lstMonthlyCharge.SelectedItem = "10.00"
    lstYears.SelectedItem = "5"

End Sub
In Chapter 1, we needed to declare variables, whose values were to be stored in memory to be used by the code. In that chapter, we declared string variables to store sentences. In this chapter, we need to declare an integer variable for whole numbers and a decimal variable for numbers with decimal places.

The values of these variables are added as formatted items into the ListBoxes by means of loops. So, we will discuss three concepts:

- formatting an item
- adding an item to a ListBox
- writing a loop to do this many times (repetition)

### Formatting an Item

We format the item being added for the number of decimal place (2 for charges and 0 for years), by using `FormatNumber`. As you can see, the legal way to format for two decimal places is

```plaintext
FormatNumber(item to be formatted, number of decimal places)
```

### Clearing and Adding Items to a ListBox

A ListBox has an `Items` property, which has `Clear` and `Add` methods to clear and add items into the list. As you can see above, the legal ways to do this are:

```plaintext
Name of ListBox.Items.Clear()

Name of ListBox.Items.Add(item to be added)
```

Use `IntelliSense` to save on typing: enter the name of the ListBox, enter a dot, select `Items` from a pull-down list, enter a dot, select `Add` from a pull-down list, and enter parentheses with the item to be added placed inside.
Writing a Loop (Repetition)

The legal form or syntax of the **For** loop is

```vbnet
For variable name = starting-value To ending-value Step step-value
    Statement or statements to be repeated (the **BODY** of the loop)
Next
```

The programmer declares and uses a variable name, such as `monthlyCharge` and then specifies its

- starting-value
- ending-value
- step-value (how much to **ADD** to the variable each time it repeats the **BODY** of the loop)

Just as an **If** statement requires a matching **End If** in a **DECISION** statement, a **For** statement requires a matching **Next** statement after the **BODY** of the loop.

**NOTE:** Visual Basic automatically puts **Next** at the bottom of the loop! Also, it automatically highlights in **blue** all of these key (reserved) words in **Visual Basic 2005 Express**.

The **For** statement is responsible for giving the variable a starting value and checking to see when the ending value is reached. The **Next** statement is responsible for adding the step-value to the variable and then sending control back to the top of the loop.

**Within the loop, we format the item being added for the number of decimal places (2 for monthlyCharges and 0 for years), by using **FormatNumber**.**
Most Common Programming Mistake

Do not reverse the following:

- the **name of the object** `IstMonthlyCharge` (which contains all of the monthly charges in a ListBox)
- the **variable name** `monthlyCharge` (which is used as the loop variable, taking on one monthly charge value at a time)

Thus, it would be backwards to use either of the following:

```vbnet
For lstMonthlyCharge = 10 to 20 Step .5
    monthlyCharge.items.add(FormatNumber(lstMonthlyCharge,2))
```

**The btnCompute Object**

The actual code showing in your **View Code for Objects** panel is listed on the following page.
Private Sub btnCompute_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles btnCompute.Click
    'Declare variables
    Dim monthlyCharge As Decimal '2 decimal places to be used
    Dim years As Integer 'whole numbers

    'Find amounts selected
    monthlyCharge = lstMonthlyCharge.SelectedItem
    years = lstYears.SelectedItem

    'Calculations
    txtTotalPayment.Text = Val(txtStartupCharge.Text) + monthlyCharge * 12 * years
    'Formatting
    txtTotalPayment.Text = FormatCurrency(txtTotalPayment.Text)

End Sub

ListBoxes have a SelectedItem property, which in these two cases are the monthlyCharge and years selected by the user.

After placing the numerical result into the text for txtTotalPayment, we format it with the FormatCurrency method. The legal way to do this is

    Name of Object.Text = FormatCurrency(Name of Object.Text)

**NOTE:** it is legal to have the name of the object, in this case txtTotalPayment, on both sides. On the left, it means the "after" version - on the right, the "before" version.

**NOTE:** Just as with btnCompute, you need to declare the variables monthlyCharge and years and in order to use them. You may copy and paste these two lines from your btnCompute code.
**Practice Building the Project**

You will use **VBDesigner** to choose the objects and their properties, and then use **Visual Basic 2005 Express** to write the code.

### 1. DESIGN (OBJECTS AND PROPERTIES)

**For Object Properties**

- 3A Phone Pay Demo Folder
- Executable Phone Pay File

*To Design*

- VBDdesigner – Step 1 (Later Step 2 If Needed)
- 3B Phone Pay Build Folder

If you point the mouse - **without clicking** - over an object, **Visual Basic 2005 Express**’s **tooltip** feature will display – for a short time – the type of object and its key properties. Use this as your guide in building your own project.
Now – to build your own project, run **VBDesigner – Step 1**. Open a new **Phone Pay** objects file in the **3B Phone Pay Build Folder**. In the left panel (**ToolBox and Properties**), you can

- select Labels, TextBoxes, ListBoxes, and Buttons
- move them to the **right** panel (your project form or window)
- for each object, enter text and choose colors, font, and size
- for ListBoxes, enter text for each desired row and leave unused row numbers alone – they will be discarded when you **deploy** your objects to the project
- if you accidentally enter text into a row of a ListBox, **simply reenter the row number to get rid of the text**
- for the txtStartupCharge TextBox, keep the Read/Write setting so that text can be entered when running the project
- for the txtTotalPayment TextBox, click on the ReadOnly setting – the user cannot enter anything but instead the numbers are put into these TextBoxes by the code of other objects

You exit by clicking the **Exit Button**.

If you do not have time to complete your design, you can run **VBDesigner – Step 2** at a later time. You will have to reopen your objects file and then you can continue.
2. CODE (INSTRUCTIONS)

To Code

3B Phone Pay Build Folder

Project File
This Opens Visual Basic 2005 Express

To Review Code Needed
Read “Reviewing the Code” in this Chapter

Open the project file. You should now only have open the form design window for Form1.vb.

In order to write code for your objects, you need to open the form code window for Form1.vb:

Click on View Menu – Code

If you ever need to return to the design window:

Click on View Menu – Design

To view the code that you will enter, run VBDesigner – Step 3. Log onto the objects file in the 3A Phone Pay Demo Folder.
Click on each object in the **right** panel (**Phone Pay**) and observe its code in the **left** panel (**View Code for Objects**).

Now go back and forth between **VBDesigner** and **Visual Basic 2005 Express** to transfer the code:

- click on an object in **VBDesigner** to view its code
- pull down the object and action in the **Visual Basic 2005 Express** code window and enter its code
- repeat this for each object that has code

Remember to save your work frequently!

Finally, try out the code that you have written and debug if necessary.
Lab Exercise 3 (Optional: No Credit)

For Object Properties & Sample Run

Lab Exercise 3 Folder

Executable Lab Exercise 3 File

To Design

VBDesigner – Step 1 (Later Step 2 If Needed)

Lab Exercise 3 Folder-

NOT MY PROJECT SUBFOLDER!!!

To Code

Lab Exercise 3 Folder

Project File

This Opens Visual Basic 2005 Express

Build a project which converts quarters to dollars. Select the number of quarters from the ListBox and click on the Dollars button to display the value formatted in dollars and cents, as illustrated by the demo screens on the following page.
Page 85
Lab Assignment 3 (Credit)

For Object Properties & Sample Run

3C Lab Folder
Executable Computer Pay File

To Design

VBDesigner – Step 1 (Later Step 2 If Needed)
3C Lab Folder-NOT MY PROJECT SUBFOLDER!!

To Code

3C Lab Folder
Project File

This Opens Visual Basic 2005 Express

Build a project called Computer Pay, which is similar to the phone store project Phone Pay, except that it is used by a computer store in order to calculate a store employee’s pay. It has a Green background. It has ListBox for displaying the value of the hours worked by the employee (between 0 and 40, with a default of 30). It has a ListBox for the hourly pay rate from 6 to 10, with step size of .50, with a default of 7.00. When the compute Button is clicked, the pay is calculated and displayed inside a
ReadOnly TextBox, formatted to currency, to be the number of hours worked times the hourly rate.

You are REQUIRED to use a loop to put the hourly pay rates into the list!

**MOST COMMON PROGRAMMING MISTAKES:**

Confusing the loop variable name with the ListBox name in writing the loop.

Forgetting to format for two decimal places and for currency

We show some demo screens on the following page.
VBDesigner Edition

Employee Pay

Hours

Pay Rate

Gross Pay

$187.50

Compute

Exit

Page 88
You should do the following steps:

- Run the **Executable Computer Pay file** in the **3C Lab folder**
  - Pointing to the objects lists their properties
- In **VBDesigner**, build your objects file in the **3C Lab folder**
  - Make sure the objects and properties match
- Use **Windows Explorer** to open the **project file** in the **3C Lab folder**
- Now you are in **Visual Basic 2005 Express** and must pull down objects and methods to write the code
  - There is **NO** code for you to copy this time, or the lab credits would be given to you without having to think about the solution!
  - You do have the already running **Executable Computer Pay file** to serve as your guide – it will show you **WHAT** it should do, but not **HOW** it is done.
  - The demo screens below will also show you **WHAT** it should do.
  - Refer in this Guide to the **Phone Pay** description to see how **similar** actions are programmed. **Be aware that the object names and sentences themselves will be different!**
  - For this lab, your instructor **CAN** help you, but refer to this Guide first!
  - **NOTE**: for a quiz, your instructor **CANNOT** help you – that is why the best way to prepare for a quiz is to understand fully the demo and the lab projects.