

James Foxall



STARTER KIT

DVD includes
Visual Basic® 2008
Express Edition

Sams **Teach Yourself**

Visual Basic® 2008

in **24**
Hours

SAMS

Sams Teach Yourself Visual Basic 2008 in 24 Hours

Copyright © 2008 by Pearson Education, Inc.

All rights reserved. No part of this book shall be reproduced, stored in a retrieval system, or transmitted by any means, electronic, mechanical, photocopying, recording, or otherwise, without written permission from the publisher. No patent liability is assumed with respect to the use of the information contained herein. Although every precaution has been taken in the preparation of this book, the publisher and author assume no responsibility for errors or omissions. Nor is any liability assumed for damages resulting from the use of the information contained herein.

This material may be distributed only subject to the terms and conditions set forth in the Open Publication License, v1.0 or later (the latest version is presently available at <http://www.opencontent.org/openpub/>).

ISBN-13: 978-0-672-32984-5

ISBN-10: 0-672-32984-0

Library of Congress Cataloging-in-Publication Data:

Foxall, James D.

Sams teach yourself Visual BASIC 2008 in 24 hours : complete starter kit / James Foxall.
p. cm.

ISBN-10: 0-672-32984-0

ISBN-13: 978-0-672-32984-5

1. Microsoft Visual BASIC. 2. BASIC (Computer program language) I. Title.
QA76.73.B3F69528 2008
005.2'762—dc22

2008010868

Printed in the United States of America

First Printing May 2008

Trademarks

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Sams Publishing cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

Warning and Disclaimer

Every effort has been made to make this book as complete and accurate as possible, but no warranty or fitness is implied. The information provided is on an “as is” basis. The author and the publisher shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book or from the use of the CD or programs accompanying it.

Bulk Sales

Sams Publishing offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales. For more information, please contact

U.S. Corporate and Government Sales

1-800-382-3419

corpsales@pearsontechgroup.com

For sales outside the U.S., please contact

International Sales

international@pearson.com

Editor-in-Chief

Karen Gettman

Executive Editor

Neil Rowe

Development Editor

Mark Renfrow

Technical Editor

Todd Meister

Managing Editor

Gina Kanouse

Project Editor

Anne Goebel

Copy Editor

Gayle Johnson

Indexer

Erika Millen

Proofreader

Kathy Ruiz

Publishing Coordinator

Cindy Teeters

Multimedia Developer

DPL

Cover Designer

Gary Adair

Composition

Nonie Ratcliff



This Book Is Safari Enabled

The Safari® Enabled icon on the cover of your favorite technology book means the book is available through Safari Bookshelf. When you buy this book, you get free access to the online edition for 45 days.

Safari Bookshelf is an electronic reference library that lets you easily search thousands of technical books, find code samples, download chapters, and access technical information whenever and wherever you need it.

To gain 45-day Safari Enabled access to this book:

- ▶ Go to <http://www.informit.com/onlineedition>.
- ▶ Complete the brief registration form.
- ▶ Enter the coupon code ZEHX-CMHH-HV33-B7H6-BTH4.

If you have difficulty registering on Safari Bookshelf or accessing the online edition, please email customer-service@safaribooksonline.com.

Introduction

Visual Basic 2008 is Microsoft's latest incarnation of the enormously popular Visual Basic language, and it's fundamentally different from the versions that came before it. Visual Basic 2008 is now more powerful and more capable than ever before, and its features and functionality are on par with "higher-level" languages such as C++. One consequence of this newfound power is added complexity. Gone are the days when you could sit down with Visual Basic and the online Help and teach yourself what you needed to know to create a functional program.

Audience and Organization

This book is targeted toward those who have little or no programming experience or who might be picking up Visual Basic as a second language. The book has been structured and written with a purpose: to get you productive as quickly as possible. I've used my experiences in writing large commercial applications with Visual Basic and teaching Visual Basic to create a book that I hope cuts through the fluff and teaches you what you need to know. All too often, authors fall into the trap of focusing on the technology rather than on the practical application of the technology. I've worked hard to keep this book focused on teaching you practical skills that you can apply immediately toward a development project. Feel free to post your suggestions or success stories at www.jamesfoxall.com/forums.

This book is divided into five parts, each of which focuses on a different aspect of developing applications with Visual Basic 2008. These parts generally follow the flow of tasks you'll perform as you begin creating your own programs using Visual Basic 2008. I recommend that you read them in the order in which they appear.

- ▶ Part I, "The Visual Basic 2008 Environment," teaches you about the Visual Basic environment, including how to navigate and access Visual Basic's numerous tools. In addition, you'll learn about some key development concepts such as objects, collections, and events.
- ▶ Part II, "Building a User Interface," shows you how to build attractive and functional user interfaces. In this part, you'll learn about forms and controls—the user interface elements such as text boxes and list boxes.

Sams Teach Yourself Visual Basic 2008 in 24 Hours

- ▶ Part III, “Making Things Happen: Programming,” teaches you the nuts and bolts of Visual Basic 2008 programming—and there’s a lot to learn. You’ll discover how to create modules and procedures, as well as how to store data, perform loops, and make decisions in code. After you’ve learned the core programming skills, you’ll move into object-oriented programming and debugging applications.
- ▶ Part IV, “Working with Data,” introduces you to working with graphics, text files, and programming databases and shows you how to automate external applications such as Word and Excel. In addition, this part teaches you how to manipulate a user’s file system and the Windows Registry.
- ▶ Part V, “Deploying Solutions and Beyond,” shows you how to distribute an application that you’ve created to an end user’s computer. In Hour 24, “The 10,000-Foot View,” you’ll learn about Microsoft’s .NET initiative from a higher, less-technical level.

Many readers of previous editions have taken the time to give me input on how to make this book better. Overwhelmingly, I was asked to have examples that build on the examples in the previous chapters. In this book, I have done that as much as possible. Now, instead of learning concepts in isolated bits, you’ll be building a feature-rich Picture Viewer program throughout the course of this book. You’ll begin by building the basic application. As you progress through the chapters, you’ll add menus and toolbars to the program, build an Options dialog box, modify the program to use the Windows Registry and a text file, and even build a setup program to distribute the application to other users. I hope you find this approach beneficial in that it allows you to learn the material in the context of building a real program.

Conventions Used in This Book

This book uses several design elements and conventions to help you prioritize and reference the information it contains:

By the Way

By the Way boxes provide useful sidebar information that you can read immediately or circle back to without losing the flow of the topic at hand.

Did you Know?

Did You Know? boxes highlight information that can make your Visual Basic programming more effective.

Watch Out! boxes focus your attention on problems or side effects that can occur in specific situations.

**Watch
Out!**

New terms appear in a **semibold** typeface for emphasis.

In addition, this book uses various typefaces to help you distinguish code from regular English. Code is presented in a monospace font. Placeholders—words or characters that represent the real words or characters you would type in code—appear in *italic monospace*. When you are asked to type or enter text, that text appears in **bold**.

Menu options are separated by a comma. For example, when you should open the File menu and choose the New Project menu option, the text says “Select File, New Project.”

Some code statements presented in this book are too long to appear on a single line. In these cases, a line-continuation character (an underscore) is used to indicate that the following line is a continuation of the current statement.

Onward and Upward!

This is an exciting time to be learning how to program. It's my sincerest wish that when you finish this book, you feel capable of creating, debugging, and deploying modest Visual Basic programs using many of Visual Basic's tools. Although you won't be an expert, you'll be surprised at how much you've learned. And I hope this book will help you determine your future direction as you proceed down the road to Visual Basic mastery.

I love programming with Visual Basic, and sometimes I find it hard to believe I get paid to do so. I hope you find Visual Basic as enjoyable as I do!

HOUR 1

Jumping in with Both Feet: A Visual Basic 2008 Programming Tour

What You'll Learn in This Hour:

- ▶ Building a simple (yet functional) Visual Basic application
- ▶ Letting a user browse a hard drive
- ▶ Displaying a picture from a file on disk
- ▶ Getting familiar with some programming lingo
- ▶ Learning about the Visual Studio 2008 IDE

Learning a new programming language can be intimidating. If you've never programmed before, the act of typing seemingly cryptic text to produce sleek and powerful applications probably seems like a black art, and you might wonder how you'll ever learn everything you need to know. The answer, of course, is one step at a time. I believe the first step to mastering a programming language is *building confidence*. Programming is part art and part science. Although it might seem like magic, it's more akin to illusion. After you know how things work, a lot of the mysticism goes away, and you are free to focus on the mechanics necessary to produce the desired result.

Producing large, commercial solutions is accomplished by way of a series of small steps. After you've finished this hour, you'll have a feel for the overall development process and will have taken the first step toward becoming an accomplished programmer. In fact, you will build on the examples in this hour in subsequent chapters. By the time you complete this book, you will have built a robust application, complete with resizable screens, an intuitive interface including menus and toolbars, and robust code with professional error handling. But I'm getting ahead of myself.

In this hour, you'll complete a quick tour of Visual Basic that takes you step by step through creating a complete, albeit small, Visual Basic program. Most introductory programming books start by having the reader create a simple Hello World program. I've yet to see a Hello World program that's the least bit helpful. (They usually do nothing more than print hello world to the screen—what fun!) So, instead, you'll create a Picture Viewer application that lets you view Windows bitmaps and icons on your computer. You'll learn how to let a user browse for a file and how to display a selected picture file on the screen. The techniques you learn in this chapter will come in handy in many real-world applications that you'll create, but the goal of this chapter is for you to realize just how much fun it is to program using Visual Basic 2008.

Starting Visual Basic 2008

Before you begin creating programs in Visual Basic 2008, you should be familiar with the following terms:

- ▶ **Distributable component:** The final, compiled version of a project. Components can be distributed to other people and other computers, and they don't require the Visual Basic 2008 development environment (the tools you use to create a .NET program) to run (although they do require the .NET runtime, which I'll discuss in Hour 23, "Deploying Applications"). Distributable components are often called **programs**. In Hour 23, you'll learn how to distribute the Picture Viewer program that you're about to build to other computers.
- ▶ **Project:** A collection of files that can be compiled to create a distributable component (program). There are many types of projects, and complex applications might consist of multiple projects, such as Windows application projects, and support dynamic link library (DLL) projects.
- ▶ **Solution:** A collection of projects and files that make up an application or component.

By the Way

In the past, Visual Basic was an autonomous language. This has changed. Now, Visual Basic is part of a larger entity known as the **.NET Framework**. The .NET Framework encompasses all the .NET technology, including Visual Studio .NET (the suite of development tools) and the common language runtime (CLR), which is the set of files that make up the core of all .NET applications. You'll learn about these items in more detail as you progress through this book. For now, realize that Visual Basic is one of many languages that exist within the Visual Studio family. Many other languages, such as C#, are also .NET languages, make use of the CLR, and are developed within Visual Studio.

Visual Studio 2008 is a complete development environment, and it's called the **IDE** (short for **integrated development environment**). The IDE is the design framework in which you build applications; every tool you'll need to create your Visual Basic projects is accessed from within the Visual Basic IDE. Again, Visual Studio 2008 supports development using many different languages, Visual Basic being the most popular. The environment itself is not Visual Basic, but the language you'll be using within Visual Studio 2008 is Visual Basic. To work with Visual Basic projects, you first start the Visual Studio 2008 IDE.

Start Visual Studio 2008 now by choosing Microsoft Visual Basic 2008 Express Edition from the Start/Programs menu. If you are running the full retail version of Visual Studio, your shortcut may have a different name. In this case, locate the shortcut on the Start menu and click it once to start the Visual Studio 2008 IDE.

Creating a New Project

When you first start Visual Studio 2008, you see the Start Page tab within the IDE, as shown in Figure 1.1. You can open projects created previously or create new projects from this Start page. For this quick tour, you'll create a new Windows application, so select File, New Project to display the New Project dialog box, shown in Figure 1.2.

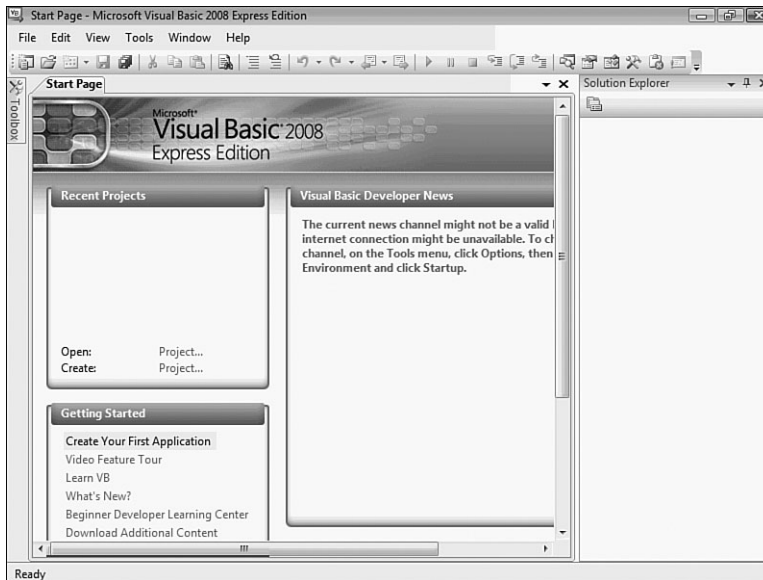
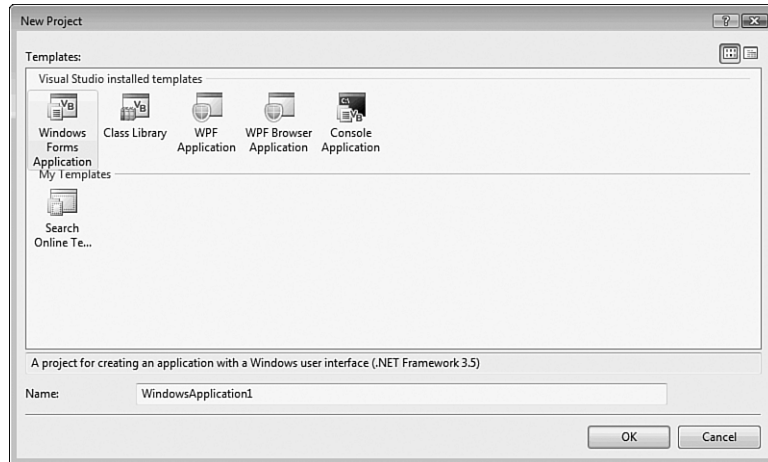


FIGURE 1.1
You can open existing projects or create new projects from the Visual Studio Start page.

**By the
Way**

If your Start page doesn't look like the one shown in Figure 1.1, chances are that you've changed the default settings. In Hour 2, "Navigating Visual Basic 2008," I'll show you how to change them back.

FIGURE 1.2
The New Project dialog box enables you to create many types of .NET projects.



The New Project dialog box is used to specify the type of Visual Basic project to create. (You can create many types of projects with Visual Basic, as well as with the other supported languages of the .NET Framework.) The options shown in Figure 1.2 are limited because I am running the Express edition of Visual Basic for all examples in this book. If you are running the full version of Visual Studio, you will have many more options available.

Create a new Windows Forms Application now by following these steps:

1. Make sure that the Windows Forms Application icon is selected. (If it's not, click it once to select it.)
2. At the bottom of the New Project dialog box is a Name text box. This is where, oddly enough, you specify the name of the project you're creating. Enter **Picture Viewer** in the Name text box.
3. Click OK to create the project.

**Did you
Know?**

Always set the Name text box to something meaningful before creating a project, or you'll have more work to do later if you want to move or rename the project.

When Visual Basic creates a new Windows Forms Application project, it adds one form (the empty gray window) for you to begin building the **interface** for your application, as shown in Figure 1.3.

Within Visual Studio 2008, **form** is the term given to the design-time view of a window that can be displayed to a user.

**By the
Way**

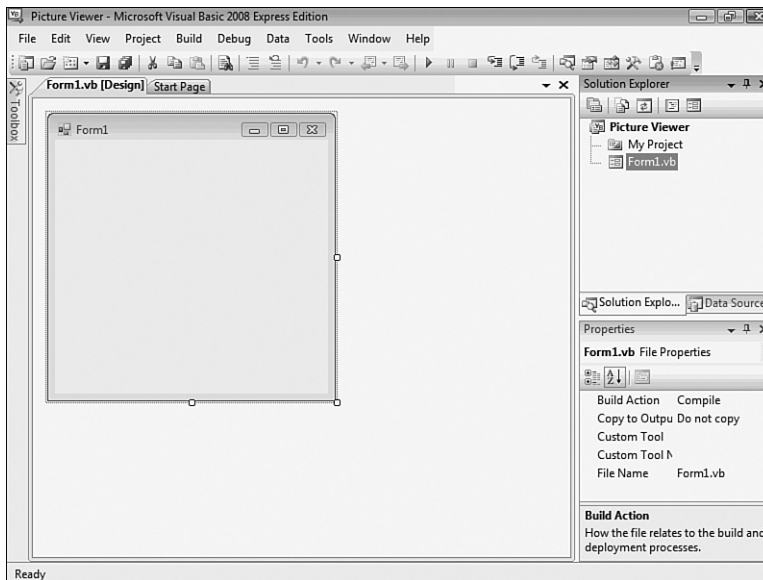


FIGURE 1.3
New Windows Forms Applications start with a blank form; the fun is just beginning!

Your Visual Studio 2008 environment might look different from that shown in the figures in this hour due to the edition of Visual Studio 2008 you're using, whether you've already played with Visual Studio 2008, and other factors, such as your monitor's resolution. All the elements discussed in this hour exist in all editions of Visual Studio 2008, however. (If a window shown in a figure doesn't appear in your IDE, use the View menu to display it.)

To create a program that can be run on another computer, you start by creating a project and then compiling the project into a component such as an **executable** (a program a user can run) or a **DLL** (a component that can be used by other programs and components). The compilation process is discussed in detail in Hour 23. The important thing to note at this time is that when you hear someone refer to *creating* or *writing a program*, just as you're creating the Picture Viewer program now, that person is referring to the completion of all steps up to and including compiling the project to a distributable file.

**By the
Way**

Understanding the Visual Studio 2008 Environment

The first time you run Visual Studio 2008, you'll notice that the IDE contains a number of windows, such as the Properties window on the right, which is used to view and set properties of objects. In addition to these windows, the IDE contains a number of tabs, such as the vertical Toolbox tab on the left edge of the IDE (refer to Figure 1.3). Try this now: Click the Toolbox tab to display the Toolbox window (clicking a tab displays an associated window). You can hover the mouse over a tab for a few seconds to display the window as well. To hide the window, simply move the mouse off the window (if you hovered over the tab to display it) or click another window. To close the window, click the Close (X) button in the window's title bar.

By the Way

If you opened the toolbox by clicking its tab rather than hovering over the tab, the toolbox will not automatically close. Instead, it will stay open until you click another window.

You can adjust the size and position of any of these windows, and you can even hide and show them as needed. You'll learn how to customize your design environment in Hour 2.

Watch Out!

Unless specifically instructed to do so, don't double-click anything in the Visual Studio 2008 design environment. Double-clicking most objects produces an entirely different result than single-clicking does. If you mistakenly double-click an object on a form (discussed shortly), a code window appears. At the top of the code window is a set of tabs: one for the form design and one for the code. Click the tab for the form design to hide the code window, and return to the form.

The Properties window on the right side of the design environment is perhaps the most important window in the IDE, and it's the one you'll use most often. If your computer display resolution is set to 800×600, you can probably see only a few properties at this time. This makes it difficult to view and set properties as you create projects. All the screen shots in this book were captured at 800×600 due to size constraints, but you should run at a higher resolution if you can. I highly recommend that you develop applications with Visual Basic at a screen resolution of 1024×768 or higher to have plenty of work space. To change your display settings, right-click the desktop and select Personalize. Keep in mind, however, that end users might be running at a lower resolution than you are using for development.

Changing the Characteristics of Objects

Almost everything you work with in Visual Basic is an object. Forms, for instance, are objects, as are all the items you can put on a form to build an interface, such as list boxes and buttons. There are many types of objects, and objects are classified by type. For example, a form is a `Form` object, whereas items you can place on a form are called `Control` objects, or controls. (Hour 3, “Understanding Objects and Collections,” discusses objects in detail.) Some objects don’t have a physical appearance but exist only in code. You’ll learn about these kinds of objects in later hours.

You’ll find that I often mention material coming up in future chapters. In the publishing field, we call these *forward references*. For some reason, these tend to unnerve some people. I do this only so that you realize you don’t have to fully grasp a subject when it’s first presented; the material will be covered in more detail later. I try to keep forward references to a minimum, but unfortunately, teaching programming is not a perfectly linear process. There will be times I’ll have to touch on a subject that I feel you’re not ready to dive into fully yet. When this happens, I give you a forward reference to let you know that the subject will be covered in greater detail later.

**Watch
Out!**

Every object has a distinct set of attributes known as **properties** (regardless of whether the object has a physical appearance). Properties define the characteristics of an object. Even you have certain properties, such as your height and hair color. Visual Basic objects have properties as well, such as `Height` and `BackColor`. When you create a new object, the first thing you need to do is set its properties so that the object appears and behaves the way you want it to. To display an object’s properties, click the object in its designer (the main work area in the IDE).

Click anywhere in the default form now, and check to see that its properties are displayed in the Properties window. You’ll know because the drop-down list box at the top of the Properties window will contain the form’s name: `Form1`. `System.Windows.Forms.Form`. `Form1` is the name of the object, and `System.Windows.Forms.Form` is the type of object.

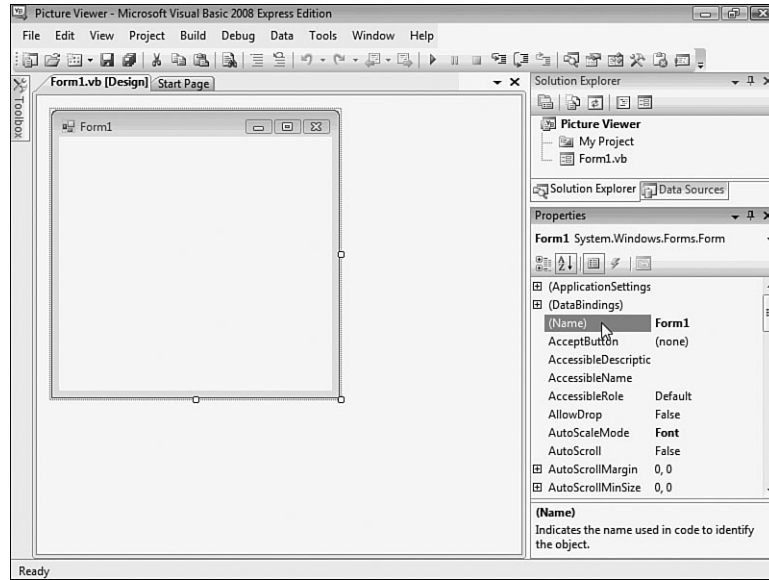
Naming Objects

The property you should always set first when creating any new object is the `Name` property. Press F4 to display the Properties window (if it’s not already visible), and scroll toward the top of the properties list until you see the `(Name)` property, as shown in Figure 1.4. If the `Name` property isn’t one of the first properties listed, the

Properties window is set to show properties categorically instead of alphabetically. You can show the list alphabetically by clicking the Alphabetical button that appears just above the properties grid.

FIGURE 1.4

The Name property is the first property you should change when you add a new object to your project.



By the Way

I recommend that you keep the Properties window set to show properties in alphabetical order; doing so makes it easier to find properties that I refer to in the text. Note that the Name property always stays toward the top of the list and is called (Name). If you're wondering why it has parentheses around it, it's because the parentheses force the property to the top of the list, because symbols come before letters in an alphabetical sort.

When saving a project, you choose a name and a location for the project and its files. When you first create an object within the project, Visual Basic gives the object a unique, generic name based on the object's type. Although these names are functional, they simply aren't descriptive enough for practical use. For instance, Visual Basic named your form Form1, but it's common to have dozens (or even hundreds) of forms in a project. It would be extremely difficult to manage such a project if all forms were distinguishable only by a number (Form2, Form3, and so forth).

What you're actually working with is a **form class**, or **template**, that will be used to create and show forms at runtime. For the purposes of this quick tour, I simply call it a form. See Hour 5, "Building Forms: The Basics," for more information.

**By the
Way**

To better manage your forms, give each one a descriptive name. Visual Basic gives you the chance to name new forms as they're created in a project. Visual Basic created this default form for you, so you didn't get a chance to name it. It's important not only to change the form's name but also to change its filename. Change the programmable name and the filename by following these steps:

1. Click the Name property, and change the text from Form1 to **ViewerForm**. Notice that this does not change the form's filename as it's displayed in the Solution Explorer window, located above the Properties window.
2. Right-click Form1.vb in the Solution Explorer window (the window above the Properties window).
3. Choose Rename from the context menu that appears.
4. Change the text from Form1.vb to **ViewerForm.vb**.

I use the Form suffix here to denote that the file is a form class. Suffixes are optional, but I find that they really help you keep things organized.

**By the
Way**

The form's Name property is actually changed for you automatically when you rename the file. In future examples, I will have you rename the form file so that the Name property is changed automatically. I had you set it in the Properties window here so that you could see how the Properties window works.

Setting the Text Property of the Form

Notice that the text that appears in the form's title bar says Form1. This is because Visual Basic sets the form's title bar to the name of the form *when it's first created* but doesn't change it when you change the name of the form. The text in the title bar is determined by the value of the form's Text property. Change the text now by following these steps:

1. Click the form once more so that its properties appear in the Properties window.
2. Use the scrollbar in the Properties window to locate the Text property.

3. Change the text to **Picture Viewer**. Press the Enter key or click a different property. You'll see the text in the title bar of the form change.

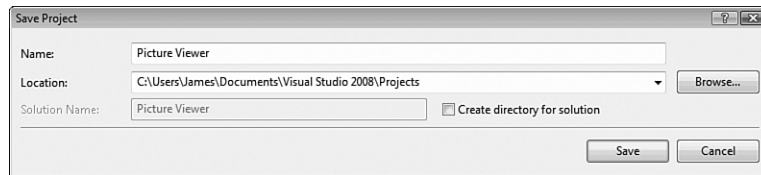
Saving a Project

The changes you've made so far exist only in memory. If you were to turn off your computer at this time, you would lose all your work up to this point. Get into the habit of frequently saving your work, which commits your changes to disk.

Click the Save All button on the toolbar (the picture of a stack of floppy disks) now to save your work. Visual Basic displays the Save Project dialog box, shown in Figure 1.5. Notice that the Name property is already filled in because you named the project when you created it. The Location text box is where you specify the location in which to save the project. Visual Basic creates a subfolder in this location using the value in the Name text box (in this case, Picture Viewer). You can use the default location or change it to suit your purposes. You can have Visual Basic create a solution folder in which the project folder gets placed. On large projects, this is a handy feature. For now, it's an unnecessary step, so uncheck the Create directory for solution box, and then click Save to save the project.

FIGURE 1.5

When saving a project, choose a name and location for the project and its files.



Giving the Form an Icon

Everyone who's used Windows is familiar with icons—the little pictures that represent programs. Icons most commonly appear on the Start menu next to the name of their respective programs. In Visual Basic, not only do you have control over the icon of your program file, you also can give every form in your program a unique icon if you want to.

By the Way

The following instructions assume that you have access to the source files for the examples in this book. They are available at <http://www.sampublishing.com>. You can also get these files, as well as discuss this book, at my website at <http://www.jamesfoxall.com/books.aspx>. When you unzip the samples, a folder will be created for each hour, and within each hour's folder will be subfolders for the sample projects. You'll find the icon for this example in the folder Hour 01\Picture Viewer.

You don't have to use the icon I've provided for this example; you can use any icon. If you don't have an icon available (or you want to be a rebel), you can skip this section without affecting the outcome of the example.

To give the form an icon, follow these steps:

1. In the Properties window, click the Icon property to select it.
2. When you click the Icon property, a small button with three dots appears to the right of the property. Click this button.
3. Use the Open dialog box that appears to locate the Picture Viewer.ico file or another icon file of your choice. When you've found the icon, double-click it, or click it once to select it and then choose Open.

After you've selected the icon, it appears in the Icon property along with the word "Icon." A small version of the icon appears in the upper-left corner of the form as well. Whenever this form is minimized, this is the icon displayed on the Windows taskbar.

This doesn't change the icon for the project as a whole. In Hour 23, you'll learn how to assign an icon to your distributable file.

**By the
Way**

Changing the Size of the Form

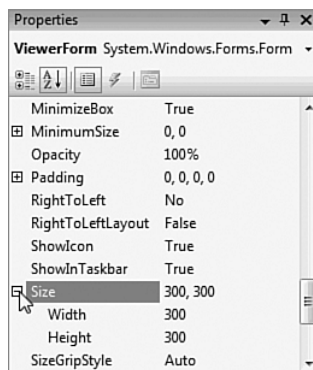
Next, you'll change the form's Width and Height properties. The Width and Height values are shown collectively under the Size property; Width appears to the left of the comma, and Height to the right. You can change the Width or Height property by changing the corresponding number in the Size property. Both values are represented in pixels. (That is, a form that has a Size property of 200,350 is 200 pixels wide and 350 pixels tall.) To display and adjust the Width and Height properties separately, click the small plus sign (+) next to the Size property (see Figure 1.6). (After you click it, it changes to a minus sign (-).)

A pixel is a unit of measurement for computer displays; it's the smallest visible "dot" on the screen. The resolution of a display is always given in pixels, such as 800×600 or 1024×768. When you increase or decrease a property by one pixel, you're making the smallest possible visible change to the property.

**By the
Way**

FIGURE 1.6

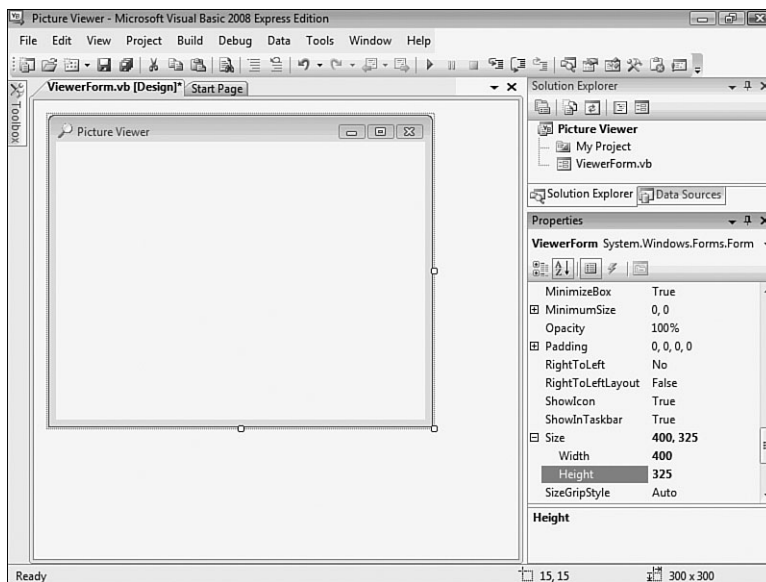
Some properties can be expanded to show more specific properties.



Change the Width property to 400 and the Height to 325 by typing in the corresponding box next to a property name. To commit a property change, press Tab or Enter, or click a different property or window. Your screen should now look like the one shown in Figure 1.7.

FIGURE 1.7

Changes made in the Properties window are reflected as soon as they're committed.



By the Way

You can also size a form by dragging its border, which you'll learn about in Hour 2, or by changing its properties using code, which you'll learn how to write in Hour 5.

Save the project now by choosing File, Save All from the menu or by clicking the Save All button on the toolbar—it has a picture of stacked floppy disks.

Adding Controls to a Form

Now that you've set the initial properties of your form, it's time to create a user interface by adding objects to the form. Objects that can be placed on a form are called **controls**. Some controls have a visible interface with which a user can interact, whereas others are always invisible to the user. You'll use controls of both types in this example. On the left side of the screen is a vertical tab titled Toolbox. Click the Toolbox tab to display the Toolbox window, and click the plus sign next to Common Controls to see the most commonly used controls (see Figure 1.8). The toolbox contains all the controls available in the project, such as labels and text boxes.

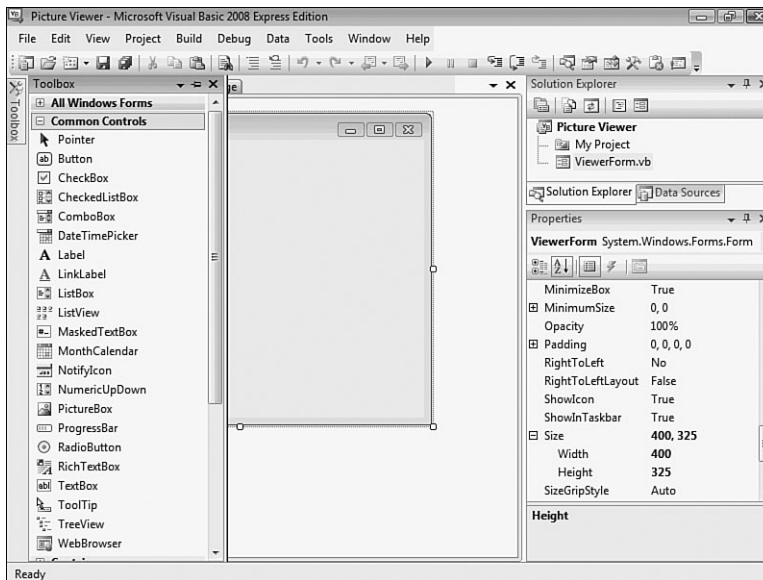


FIGURE 1.8

The toolbox is used to select controls to build a user interface.

The toolbox closes as soon as you've added a control to a form and when the pointer is no longer over the toolbox. To make the toolbox stay visible, you would click the little picture of a pushpin located in the toolbox's title bar.

I don't want you to add them yet, but your Picture Viewer interface will consist of the following controls:

- ▶ Two Button controls: The standard buttons that you're used to clicking in pretty much every Windows program you've ever run
- ▶ A PictureBox control: A control used to display images to a user
- ▶ An OpenFileDialog control: A hidden control that exposes the Windows Open File dialog box functionality

Designing an Interface

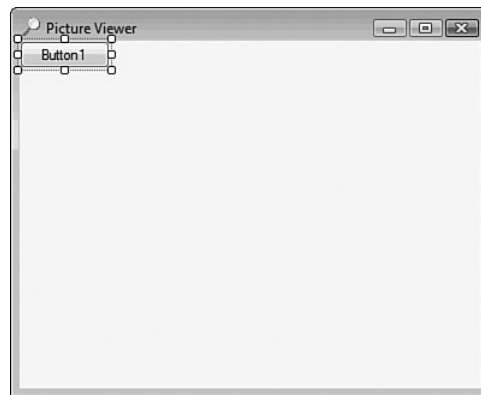
It's generally best to design a form's user interface and then add the code behind the interface to make the form functional. You'll build your interface in the following sections.

Adding a Visible Control to a Form

Start by adding a Button control to the form. Do this by double-clicking the Button item in the toolbox. Visual Basic creates a new button and places it in the upper-left corner of the form, as shown in Figure 1.9.

FIGURE 1.9

When you double-click a control in the toolbox, the control is added to the upper-left corner of the form.



Using the Properties window, set the button's properties as shown in the following list. Remember, when you view the properties alphabetically, the Name property is listed first, so don't go looking for it down in the list, or you'll be looking a while.

Property	Value
Name	btnSelectPicture
Location	295,10 (295 is the x coordinate; 10 is the y coordinate.)
Size	85,23
Text	Select Picture

Now you'll create a button that the user can click to close the Picture Viewer program. Although you could add another new button to the form by double-clicking the Button control on the toolbox again, this time you'll add a button to the form by creating a copy of the button you've already defined. This allows you to easily create a button that maintains the size and other style attributes of the original button when the copy was made.

To do this, right-click the Select Picture button, and choose Copy from its context menu. Next, right-click anywhere on the form, and choose Paste from the form's shortcut menu. (You can also use the keyboard shortcuts Ctrl+C to copy and Ctrl+V to paste.) The new button appears centered on the form, and it's selected by default. Notice that it retains almost all the properties of the original button, but the name has been reset. Change the properties of the new button as follows:

Property	Value
Name	btnQuit
Location	295,40
Text	Quit

The last visible control you need to add to the form is a PictureBox control. A PictureBox has many capabilities, but its primary purpose is to show pictures, which is precisely what you'll use it for in this example. Add a new PictureBox control to the form by double-clicking the PictureBox item in the toolbox, and set its properties as follows:

Property	Value
Name	picShowPicture
BorderStyle	FixedSingle
Location	8,8
Size	282,275

After you've made these property changes, your form will look like the one shown in Figure 1.10. Click the Save All button on the toolbar to save your work.

FIGURE 1.10
An application's interface doesn't have to be complex to be useful.



Adding an Invisible Control to a Form

All the controls you've used so far sit on a form and have a physical appearance when a user runs the application. Not all controls have a physical appearance, however. Such controls, called **nonvisual controls** (or **invisible-at-runtime controls**), aren't designed for direct user interactivity. Instead, they're designed to give you, the programmer, functionality beyond the standard features of Visual Basic.

To enable users to select a picture to display, you need to give them the ability to locate a file on their hard drives. You might have noticed that whenever you choose to open a file from within any Windows application, the dialog box displayed is almost always the same. It doesn't make sense to force every developer to write the code necessary to perform standard file operations, so Microsoft has exposed the functionality via a control that you can use in your projects. This control is called `OpenFileDialog`, and it will save you dozens of hours that would otherwise be necessary to duplicate this common functionality.

By the Way

Other controls in addition to the `OpenFileDialog` control give you file functionality. For example, the `SaveFileDialog` control provides features for allowing the user to specify a filename and path for saving a file.

Display the toolbox and scroll down using the down arrow in the lower part of the toolbox until you can see the `OpenFileDialog` control (it's in the `Dialogs` category),

and then double-click it to add it to your form. Note that the control isn't placed on the form; rather, it appears in a special area below the form (see Figure 1.11). This happens because the OpenFileDialog control has no form interface to display to the user. It does have an interface (a dialog box) that you can display as necessary, but it has nothing to display directly on a form.

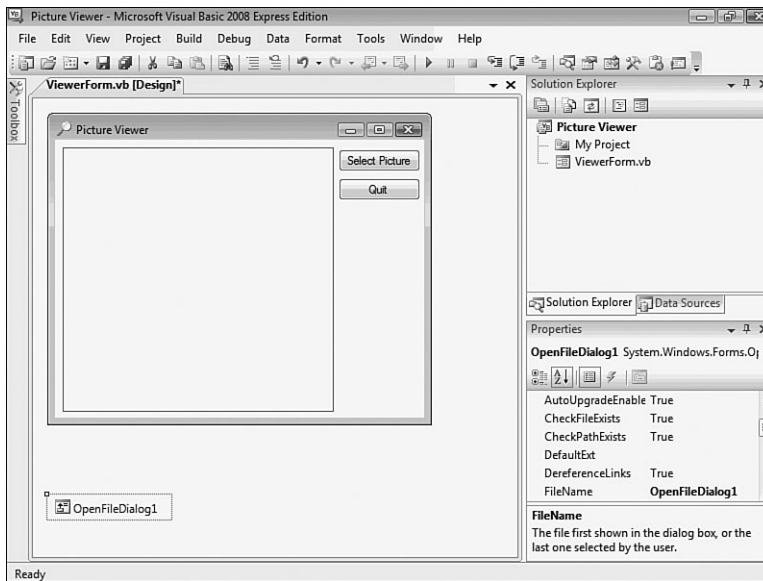


FIGURE 1.11
Controls that have no interface appear below the form designer.

Select the OpenFileDialog control, and change its properties as follows:

Property	Value
Name	ofdSelectPicture
Filename	<make empty>
Filter	Windows Bitmaps *.BMP JPEG Files .JPG
Title	Select Picture

Don't actually enter the text **<make empty>** for the filename; I really mean delete the default value and make this property value empty.

**Watch
Out!**

The Filter property is used to limit the types of files that will be displayed in the Open File dialog box. The format for a filter is description|filter. The text that

appears before the first pipe symbol is the descriptive text of the file type, whereas the text after the pipe symbol is the pattern to use to filter files. You can specify more than one filter type by separating each description|filter value with another pipe symbol. Text entered into the `Title` property appears in the title bar of the Open File dialog box.

The graphical interface for your Picture Viewer program is now finished. If you pinned the toolbox open, click the pushpin in the title bar of the toolbox now to close it.

Writing the Code Behind an Interface

You have to write code for the program to be capable of performing tasks and responding to user interaction. Visual Basic is an **event-driven** language, which means that code is executed in response to events. These events might come from users, such as a user clicking a button and triggering its `Click` event, or from Windows itself (see Hour 4, “Understanding Events,” for a complete explanation of events). Currently, your application looks nice, but it won’t do a darn thing. Users can click the Select Picture button until they can file for disability with carpal tunnel syndrome, but nothing will happen, because you haven’t told the program what to do when the user clicks the button. You can see this for yourself now by pressing F5 to run the project. Feel free to click the buttons, but they don’t do anything. When you’re finished, close the window you created to return to Design mode.

You’ll write code to accomplish two tasks. First, you’ll write code that lets users browse their hard drives to locate and select a picture file and then display it in the picture box (this sounds a lot harder than it is). Second, you’ll add code to the Quit button that shuts down the program when the user clicks the button.

Letting a User Browse for a File

The first bit of code you’ll write enables users to browse their hard drives, select a picture file, and then see the selected picture in the `PictureBox` control. This code executes when the user clicks the Select Picture button; therefore, it’s added to the `Click` event of that button.

When you double-click a control on a form in Design view, the default event for that control is displayed in a code window. The default event for a `Button` control is its `Click` event, which makes sense, because clicking is the most common action a user performs with a button. Double-click the Select Picture button now to access its `Click` event in the code window (see Figure 1.12).

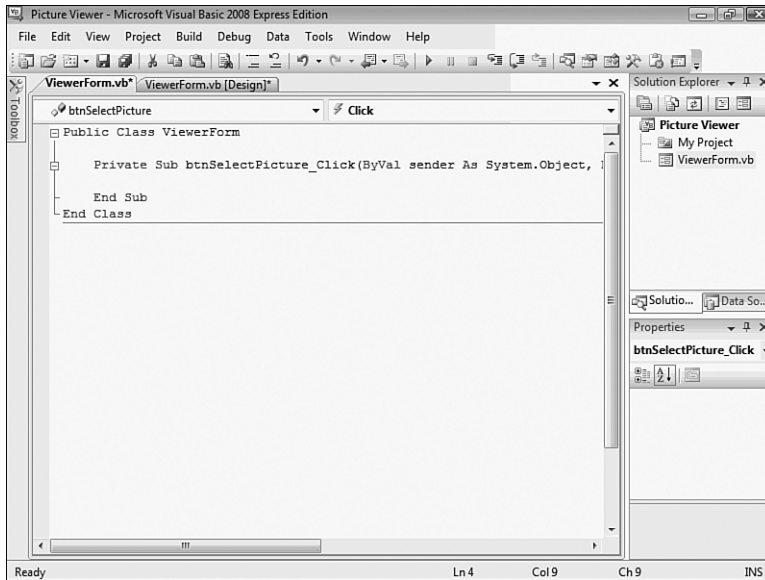


FIGURE 1.12
You'll write all your code in a window such as this.

When you access an event, Visual Basic builds an **event handler**, which is essentially a template procedure in which you add the code that executes when the event occurs. The cursor is already placed within the code procedure, so all you have to do is add code. Although this may seem daunting, by the time you're finished with this book, you'll be madly clicking and clacking away as you write your own code to make your applications do exactly what you want them to do—well, most of the time. For now, just enter the code as I present it here.

It's important that you get in the habit of commenting your code, so the first statement you'll enter is a comment. Beginning a statement with an apostrophe (') designates that statement as a comment. The compiler won't do anything with the statement, so you can enter whatever text you want after the apostrophe. Type the following statement exactly as it appears, and press the Enter key at the end of the line:

```
' Show the open file dialog box.
```

The next statement you'll enter triggers a method of the OpenFileDialog control that you added to the form. Think of a method as a mechanism to make a control do something. The ShowDialog() method tells the control to show its Open dialog box and let the user select a file. The ShowDialog() method returns a value that indicates its success or failure, which you'll then compare to a predefined result (DialogResult.OK). Don't worry too much about what's happening here; you'll be

learning the details of all this in later hours. The sole purpose of this hour is to get your feet wet. In a nutshell, the `ShowDialog()` method is invoked to let a user browse for a file. If the user selects a file, more code is executed. Of course, there's a lot more to using the `OpenFileDialog` control than I present in this basic example, but this simple statement gets the job done. Enter the following statement and press Enter to commit the code (don't worry about capitalization; Visual Basic will fix the case for you!):

```
If ofdSelectpicture.ShowDialog = DialogResult.OK Then
```

By the Way

After you insert the statement that begins with `If` and press Enter, Visual Basic automatically creates the `End If` statement for you. If you type in `End If`, you'll wind up with two `End If` statements, and your code won't run. If this happens, delete one of the statements. Hour 13, "Making Decisions in Visual Basic Code," has all the details on the `If` statement.

It's time for another comment. The cursor is currently between the statement that starts with `If` and the `End If` statement. Leave the cursor there and type the following statement, remembering to press Enter at the end of the line:

```
' Load the picture into the picture box.
```

Did you Know?

Don't worry about indenting the code by pressing the Tab key or using spaces. Visual Basic automatically indents code for you.

This next statement, which appears within the `If` construct (between the `If` and `End If` statements), is the line of code that actually displays the picture in the picture box.

Enter the following statement:

```
picshowpicture.Image = Image.FromFile(ofdselectpicture.filename)
```

In addition to displaying the selected picture, your program also displays the path and filename of the picture in the title bar. When you first created the form, you changed its `Text` property using the Properties window. To create dynamic applications, properties need to be constantly adjusted at runtime, and you do this using code. Insert the following two statements, pressing Enter at the end of each line:

```
' Show the name of the file in the form's caption.  
Me.Text = "Picture Viewer(" & ofdselectpicture.FileName & ")"
```

After you've entered all the code, your editor should look like that shown in Figure 1.13.

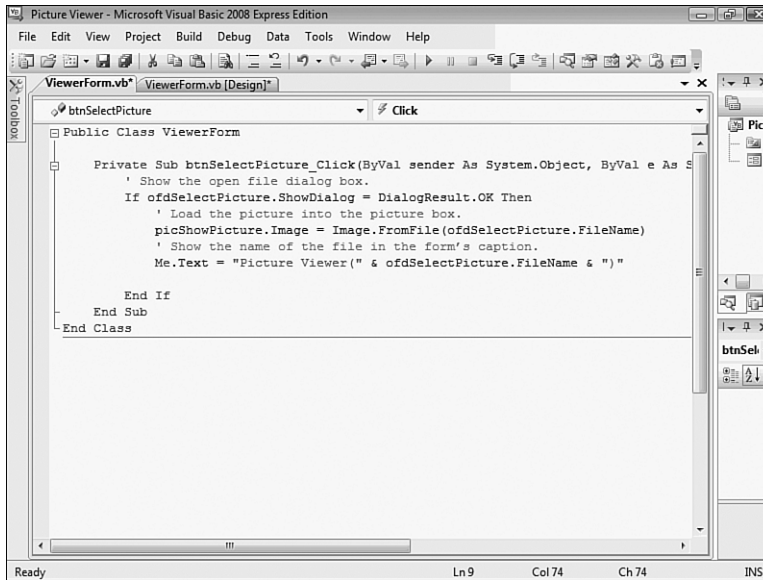


FIGURE 1.13
Make sure that your code exactly matches the code shown here.

Terminating a Program Using Code

The last bit of code you'll write terminates the application when the user clicks the Quit button. To do this, you'll need to access the Click event handler of the btnQuit button. At the top of the code window are two tabs. The current tab says ViewerForm.vb*. This tab contains the code window for the form that has the file-name ViewerForm.vb. Next to this is a tab that says ViewerForm.vb [Design]*. Click this tab to switch from Code view to the form designer. If you receive an error when you click the tab, the code you entered contains an error, and you need to edit it to make it the same as shown in Figure 1.13. After the form designer appears, double-click the Quit button to access its Click event.

Enter the following code in the Quit button's Click event handler; press Enter at the end of each statement:

```

' Close the window and exit the application
Me.Close()
  
```

**By the
Way**

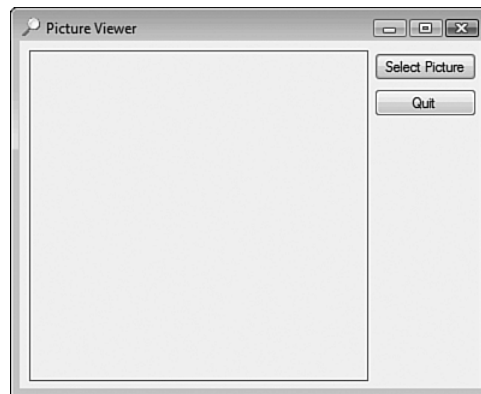
The `Me.Close()` statement closes the current form. When the last loaded form in a program is closed, the application shuts itself down—completely. As you build more robust applications, you'll probably want to execute all kinds of cleanup routines before terminating an application, but for this example, closing the form is all you need to do.

Running a Project

Your application is now complete. Click the Save All button on the toolbar (the stack of floppy disks), and then run your program by pressing F5. You can also run the program by clicking the button on the toolbar that looks like a right-facing triangle and resembles the Play button on a DVD player. (This button is called Start, and it can also be found on the Debug menu.) Learning the keyboard shortcuts will make your development process move along faster, so I recommend that you use them whenever possible.

When you run the program, the Visual Basic interface changes, and the form you've designed appears, floating over the design environment (see Figure 1.14).

FIGURE 1.14
When in Run mode, your program executes the same as it would for an end user.



You are now running your program as though it were a stand-alone application running on another user's machine; what you see is exactly what users would see if they ran the program (without the Visual Studio 2008 design environment in the background, of course). Click the Select Picture button to display the Select Picture dialog box, shown in Figure 1.15. Use this dialog box to locate a picture file. When you've found a file, double-click it, or click once to select it and then click Open. The selected picture is then displayed in the picture box, as shown in Figure 1.16.

When you click the Select Picture button, the default path shown depends on the last active path in Windows, so it might be different for you than shown in Figure 1.15.

By the Way

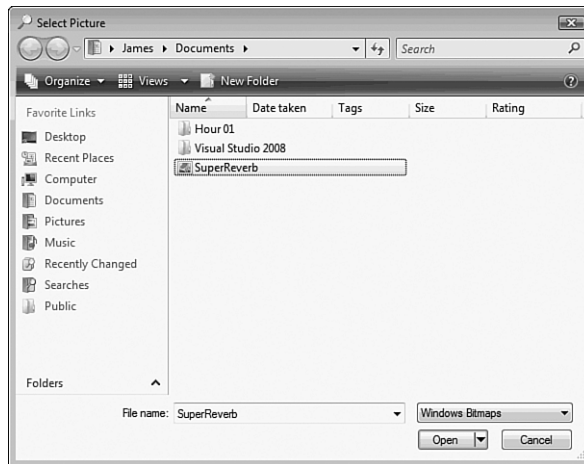


FIGURE 1.15

The OpenFileDialog control handles all the details of browsing for files. Cool, huh?



FIGURE 1.16

What could be prettier than a 1964 Fender Super Reverb amplifier?

If you want to select and display a picture from your digital camera, chances are the format is JPEG, so you'll need to select this from the Files of Type drop-down. Also, if your image is very large, you'll see only the upper-left corner of the image (what fits in the picture box). In later hours, I'll show you how you can scale the image to fit the picture box, and even resize the form to show a larger picture in its entirety.

By the Way

Summary

When you're finished playing with the program, click the Quit button to return to Design view.

That's it! You've just created a bona fide Visual Basic program. You've used the toolbox to build an interface with which users can interact with your program, and you've written code in strategic event handlers to empower your program to do things. These are the basics of application development in Visual Basic. Even the most complicated programs are built using this fundamental approach; you build the interface and add code to make the application do things. Of course, writing code to do things *exactly* the way you want things done is where the process can get complicated, but you're on your way.

If you take a close look at the organization of the hours in this book, you'll see that I start out by teaching you the Visual Basic (Visual Studio .NET) environment. I then move on to building an interface, and later I teach you about writing code. This organization is deliberate. You might be eager to jump in and start writing serious code, but writing code is only part of the equation—don't forget the word *Visual* in Visual Basic. As you progress through the hours, you'll build a solid foundation of development skills.

Soon, you'll pay no attention to the man behind the curtain—you'll be that man (or woman)!

Q&A

Q. *Can I show bitmaps of file types other than BMP and JPG?*

A. Yes. PictureBox supports the display of images with the extensions BMP, JPG, ICO, EMF, WMF, and GIF. PictureBox can even save images to a file using any of the supported file types.

Q. *Is it possible to show pictures in other controls?*

A. PictureBox is the control to use when you are just displaying images. However, many other controls allow you to display pictures as part of the control. For instance, you can display an image on a button control by setting the button's Image property to a valid picture.

Workshop

Quiz

1. What type of Visual Basic project creates a standard Windows program?
2. What window is used to change the attributes (location, size, and so on) of a form or control in the IDE?
3. How do you access the default event (code) of a control?
4. What property of a picture box do you set to display an image?
5. What is the default event for a button control?

Answers

1. Windows Forms Application
2. The Properties window
3. Double-click the control in the designer.
4. The Image property
5. The Click event

Exercises

1. Change your Picture Viewer program so that the user can also locate and select GIF files. (Hint: Change the Filter property of the OpenFileDialog control.)
2. Create a new project with a new form. Create two buttons on the form, one above the other. Next, change their position so that they appear next to each other.

Index

Symbols

* (asterisk), 162, 283
' (apostrophe), 339
\ (backslash), 283
, (comma), 311
& (concatentation) operator, 290-291
/ (division) operator, 283
= (equals sign), 66
^ (exponentiation) operator, 283
- (negation) operator, 283
() (parentheses), 302
. (period), 65, 68
+ (plus sign), 51, 282
(pound sign), 258
- (subtraction) operator, 283
_ (underscore), 77, 90, 233

A

Abort enumerator
(DialogResult), 386
AbortRetryIgnore enumerator
(MessageBoxButtons), 383
accelerator keys, 206
Accept buttons, 164-165
AcceptsReturn property (Text
Box control), 161
AcceptsTabs property (Text
Box control), 161
accessing
 events, 89-90
 Registry, 445-446
actions, 344-345
ActiveCaption system
 color, 410
ActiveCaptionText system
 color, 410
ActiveCell object, 490

ActiveMdiChild property

ActiveMdiChild property
 (forms), 150

Add Class command
 (Project menu), 364

Add Module command
 (Project menu), 230

Add New Item dialog box,
 106, 230

Add() method, 480, 489
 Items collection,
 172-173, 195
 Nodes collection, 197-199

addition, 282

addition (+) operator, 282

AddTwoNumbers()
 function, 239

ADO.NET
 connecting to
 databases, 469-471
 DataAdapter object
 creating, 472-473
 definition, 468
 Fill() method, 472
 Update() method,
 472, 479
 DataRow object, 475-476
 DataSet object, 468
 DataTable object
 creating records,
 480-481
 definition, 468
 deleting records,
 481-482

 editing records, 479
 navigating records,
 477-479
 overview, 472-474
 populating, 474
 disconnecting from
 databases, 471
 OleDbConnection object
 Close() method, 471
 connecting to
 databases, 469-471
 ConnectionString
 property, 470
 definition, 468
 disconnecting from
 databases, 471
 Open() method, 471
 overview, 468-469
 SqlConnection object, 468

**Advanced Appearance dialog
 box**, 409-410

alignment
 controls, 135-136
 text boxes, 157-158

ampersand (&), 290-291

Anchor property, 139-142

anchoring controls, 138-141

And operator, 288

apostrophe ('), 339

**Appearance Settings dialog
 box**, 409

Application object, 488

applications. See programs

Archive flag (files), 437

arithmetic
 addition, 282
 division, 283
 exponentiation, 283
 modulus arithmetic, 284
 multiplication, 283
 negation, 283
 operator precedence,
 284-286
 overview, 281-282
 subtraction, 283

arrays
 declaring, 263
 definition, 249, 263
 elements, 263
 multidimensional arrays,
 264-266
 referencing, 263-264

As keyword, 58

**assigning shortcut keys to
 menus**, 214-215

asterisk (*), 283

**auto-hiding design
 windows**, 40-41

**Automate Excel Properties
 command (Project
 menu)**, 486

**Automate Word Properties
 command (Project
 menu)**, 492

automation

definition, 485

Excel

creating workbooks, 489

instantiating automation
server, 488-489

making Excel

visible, 489

modifying workbooks,
490-491

referencing type

libraries, 486-488

testing, 492

overview, 485-486

Word

instantiating automation
server, 493

referencing type

libraries, 492-493

automation servers, creating**instances of**

Excel, 488-489

Word, 493

AutoScroll property

(forms), 146

AutoScrollMargin property

(forms), 146

AutoScrollMinSize property

(forms), 146

B**BackColor property**

(forms), 108-110

BackgroundImage property

(forms), 111-113

backgrounds (forms)

color, 108-110

images, 111, 113

backslash (\), 283**BaseDirectory() method, 461****bin prefix, 272****binding object references****to variables**

early binding, 374-376

late binding, 372-374

Bitmap() method, 407**bitmaps, 407-408****block scope, 267****Boolean data type, 251****Boolean logic**

And operator, 288

definition, 281

Not operator, 288-289

Or operator, 289

overview, 287-288

Xor operator, 289

borders, 115-117**break points**

actions, 344-345

creating, 343-344

definition, 343

BringToFront() method, 144**browsing files, 24-27****browsing scope, 81**

bugs, 337. *See also* debugging

build errors, 340-343**building confidence, 7****Button control. *See* buttons****Button property**

(MouseEventArgs), 92

buttons

Accept buttons, 164-165

adding to forms, 20-21

adding to message boxes,
383-385

Cancel buttons, 164-165

close buttons, 114-115

determining which button is
clicked, 385-386

drawing buttons

button properties, 73

Clear() method, 75

CreateGraphics
method, 74

Dispose() method, 77

DrawRectangle()
method, 76-77

overview, 73

testing, 78

variables, 74

maximize buttons, 114-115

minimize buttons, 114-115

overview, 163-164

buttons

- properties, 67-71
- radio buttons, 168-170, 315-316
- toolbar buttons, 217-219

ByRef keyword, 243

byt prefix, 272

Byte data type, 251

ByVal keyword, 243

C**calling**

- methods, 72
- procedures, 238-241

Cancel buttons, 164-165

**Cancel enumerator
(DialogResult), 386**

caret (^), 283

casting data types, 253-254

Catch statement, 352, 355

catching exceptions, 354

CBool() function, 253

CByte() function, 253

CChar() function, 253

CDate() function, 253

CDBl() function, 253

CDec() function, 253

**CenterParent value
(StartPosition property), 122**

**CenterScreen value
(StartPosition property), 122**

changing

- Excel workbooks, 490-491
- forms

- background
 - color, 108-110
- background images, 111-113
- borders, 115-117
- close buttons, 114-115
- icons, 113-114
- maximize buttons, 114-115
- minimize buttons, 114-115
- minimum/maximum size, 117
- names, 106
- title bar text, 108
- properties, 45-47, 67-71
- records, 479

Char data type, 251

check boxes, 166

CheckBox control, 166

**checked menu items,
creating, 208-209**

**CheckState property
(CheckBox), 166**

chr prefix, 272

CInt() function, 253

circles, 414

classes

- compared to standard modules, 363-364
- definition, 362
- encapsulation, 362-363
- form classes, 15
- interfaces
 - creating, 364-366
 - definition, 363
- methods
 - creating, 371
 - definition, 365
- object instantiation
 - early-binding object variables, 374-376
 - in variable declarations, 376
 - late-binding object variables, 372-374
 - object lifetimes, 377-378
 - overview, 371-372
 - releasing object references, 377
- overview, 362
- properties
 - declaring, 366-368
 - definition, 365
 - read-only properties, 370
 - readable properties, 368-369

ConnectionString property

- writable properties, 369
- write-only
 - properties, 370
- StreamReader, 457-459
- StreamWriter, 455-457
- cleanup code, 352**
- Clear() method, 75**
 - Graphics object, 415
 - Items collection, 175, 196
 - Nodes collection, 200
- clearing**
 - drawing surfaces, 415
 - lists, 175, 196
 - Tree View nodes, 200
- Click event, 24, 397**
 - menus, 210-211
 - text boxes, 163
- ClickOnce applications**
 - advanced settings, 505
 - creating with Publish Wizard, 500-503
 - overview, 499-500
 - testing, 504
 - uninstalling, 504-505
- Clicks property (MouseEventArgs), 92**
- client code, 364**
- clients, 364**
- CLng() function, 253, 346, 349**
- close buttons, 114-115**
- Close() method, 123, 457, 471**
- CObj() function, 253**
- code labels, 317**
- code modules. See modules**
- code windows, hiding, 12**
- collections**
 - Collections Example
 - form, 79-81
 - definition, 78
 - Items, 171-172, 217-219
 - iterative processing, 78
 - Nodes
 - Add() method, 197-199
 - Clear() method, 200
 - Remove() method, 199-200
 - structure of, 79
 - TabPage, 187
- Collections Example form, 79-81**
- color**
 - dithering, 49
 - forms
 - background color, 108-110
 - background images, 111-113
 - icons, 113-114
 - properties, 47-49
 - system colors, 409-412
- columns, 192-193, 475**
- Columns property (List View control), 192**
- combo boxes, 177-179**
- ComboBox control, 177-179**
- comma (,), 311**
- CommandBuilder object, 473**
- commands**
 - File menu, New Project, 36
 - Project menu
 - Add Class, 364
 - Add Module, 230
 - Automate Excel Properties, 486
 - Automate Word Properties, 492
 - regedit, 455
 - View menu, Properties Window, 38
- comments, 25**
 - adding to code, 338-339
 - definition, 338
- common language runtime, 510-511**
- common type system, 514**
- comparison operators, 286-287**
- compilers, 250**
- components, 53-54, 486. See also programs**
- concatenation, 80, 96, 290-291**
- concatenation (&) operator, 290-291**
- confidence, building, 7**
- connecting to databases, 469-471**
- ConnectionString property (OleDbConnection object), 470**

constants**constants**

- advantages of, 254
- defining, 255-256
- definition, 249
- naming conventions, 255
- scope
 - block scope, 267
 - global scope, 269-270
 - module-level scope, 268
 - name conflicts, 270
 - overview, 266
 - procedure-level scope, 267-268

containers

- definition, 147, 167
- frames, 167

Context Menu Strip

control, 212, 214

context menus

- assigning shortcut keys to, 214-215
- creating, 212-214

context-sensitive help, 59**continuing For...Next**

loops, 326

control objects. See controls**Control system color, 410****ControlDark system**

color, 411

ControlLight system

color, 411

controls

- adding to forms, 43-44, 128-129
 - by double-clicking from toolbox, 128
 - by dragging from toolbox, 128
 - by drawing, 129
- arranging on forms
 - alignment, 135-136
 - anchoring, 138-141
 - evenly spacing controls, 137
 - making controls same size, 137
 - snapping to grid, 129-131
 - snapping to lines, 132-133
- Button. *See* buttons
- CheckBox, 166
- ComboBox, 177-179
- Context Menu Strip, 212-214
- definition, 19, 64, 155
- Graphic objects, 406
- Group Box, 167
- Image List, 190-191
- invisible controls, 22-24
- Label, 155-156
- layering, 144

List View

- adding list items, 193-195
- columns, 192-193
- determining selected items, 196
- overview, 191-192
- removing all list items, 196
- removing list items, 196

ListBox

- adding items to lists, 172-173
- clearing lists, 175
- Items collection, 171-172
- overview, 170-171
- removing items from lists, 173-174
- retrieving item information, 175-176
- sorting lists, 177

Menu Strip, 204-206

naming conventions, 273

OpenFileDialog

- adding to forms, 22-24
- creating file filters, 428
- overview, 425-428
- showing Open File dialog box, 428-429

Panel, 167-168**PictureBox, 21-22, 30**

properties, 137-138

- Radio Button, 168-170
- SaveFileDialog, 429-431
- selecting groups
 - of, 133-135
- Status Bar, 221-223
- Tab, 186-190
- tab order, 142-144
- Text Box
 - events, 87, 163
 - maximum length, 161-162
 - multiline text boxes, 158-160
 - overview, 157
 - password fields, 162-163
 - scrollbars, 160
 - text alignment, 157-158
- Timer, 87-88, 183-186
- ToolStrip, 216-219
- Tree View
 - adding nodes, 197-199
 - clearing all nodes, 200
 - overview, 197
 - removing nodes, 199-200
 - visible controls, 20-22
- ControlText system color, 411**
- Copy() method, 432-433**
- copying files, 432-433**
- CreateDirectory() method, 440**
- CreateGraphics()**
 - method, 74, 406

- CreateSubKey()**
 - method, 446-447
- CByte() function, 253**
- CShort() function, 253**
- CSng() function, 253**
- CStr() function, 253**
- CUInt() function, 253**
- CULng() function, 253**
- current date/time,**
 - returning, 301
- CUShort() function, 253**
- custom dialog boxes, 387-391**
- Custom value (DashStyle property), 408**

D

- Dash value (DashStyle property), 408**
- DashDot value (DashStyle property), 408**
- DashDotDot value (DashStyle property), 408**
- DashStyle property (pens), 408-409**
- data types**
 - Boolean, 251
 - Byte, 251
 - casting, 253-254
 - Char, 251
 - common type system, 514

- Date, 251, 296-297
- Decimal, 251
- definition, 250
- determining, 250
- Double, 251
- guidelines for use, 252
- Integer, 251
- Long, 251
- naming conventions, 272
- Object, 251
- overview, 250
- REG_BINARY, 445
- REG_EXPAND_SZ, 445
- REG_MULTI_SZ, 445
- REG_SZ, 445
- Sbyte, 251
- Short, 251
- Single, 251
- strict typing
 - definition, 259-260
 - enabling, 261-262
- String, 251
- type conversion
 - functions, 253
- UInteger, 251
- ULong, 251
- UShort, 251
- DataAdapters, 468**
 - creating, 472-473
 - Fill() method, 472
 - Update() method, 472, 479

databases**databases. See also ADO.NET**

- connecting to, 469-471

DataAdapters

- creating, 472-473

- Fill() method, 472

- Update() method,
472, 479

- DataRow, 475-476

- DataSets, 468

DataTables

- creating records,
480-481

- definition, 468

- deleting records,
481-482

- editing records, 479

- navigating records,
477-479

- overview, 472-474

- populating, 474

- disconnecting from, 471

- overview, 467-468

records

- creating, 480-481

- deleting, 481-482

- editing, 479

- navigating, 477-479

- running database

- example, 482

DataRow, 475-476**DataSets, 468****DataTables**

- creating records, 480-481

- definition, 468

- deleting records, 481-482

- editing records, 479

- navigating records,
477-479

- overview, 472-474

- populating, 474

Date data type, 251, 296-297**Date variable, 296-297****DateAdd() function, 297-298****DateDiff() function, 299****DatePart() function, 299****dates**

- adding to/subtracting
from, 297-298

- Date data type, 251,
296-297

- determining intervals
between, 298-299

- file date/time information,
returning, 436

- formatting, 300-301

- querying whether values
are dates, 301

- retrieving current, 301

- retrieving parts of, 299

DateTime structure, 297, 301**dbl prefix, 272****debugging**

- build errors, 340-343

- comments, 338-339

exceptions

- Exception object, 354

- handling, 355-358

- overview, 337-338

Picture Viewer

- program, 452-455

- runtime errors, 340-343

structured error

- handling, 351

tools

- break points, 343-345

- Immediate window,
346-351

- Try...Catch...Finally
structure, 351-354

- unstructured error

- handling, 351

dec prefix, 272**Decimal data type, 251****decision-making constructs****Boolean logic**

- And operator, 288

- definition, 281

- Not operator,
288-289

- Or operator, 289

- overview, 287-288

- Xor operator, 289

comparison

- operators, 286-287

- GoTo statement, 317-318

- If...Then statement
 - Elseif statement, 309-310
 - If...Then...Else, 308-309
 - nesting, 310
 - simple example, 306-308
- overview, 305
- Select Case statement
 - example, 312-315
 - multiple comparisons, 311-312
 - overview, 310-311
 - radio buttons, evaluating, 315-316
- Try...Catch...Finally blocks, 319
- declaring**
 - arrays, 263
 - constants, 255-256
 - functions, 237
 - procedures, 58
 - properties, 366-368
 - subroutines, 232-236
 - variables, 57
 - explicit variable declaration, 259-260
 - simple example, 257-258
 - static variables, 270-271
- Define Color dialog box, 49**
- defining. See declaring**
- Delete() method, 481**
 - Directory object, 440
 - File object, 434-435
- DeleteSubKey() method, 447-448**
- DeleteSubKeyTree() method, 447**
- deleting**
 - directories, 440
 - files, 434-435
 - list items, 173-174, 196
 - menu items, 208
 - project files, 55-57
 - records, 481-482
 - Registry keys, 447
 - Tree View nodes, 199-200
- Delta property (MouseEventArgs), 92**
- deployment**
 - ClickOnce applications
 - advanced settings, 505
 - creating with Publish Wizard, 500-503
 - overview, 499-500
 - testing, 504
 - uninstalling, 504-505
 - overview, 499
- design windows**
 - auto-hiding, 40-41
 - docking, 39-40
 - floating, 38
 - showing/hiding, 38
 - states, 37-38
- Desktop system color, 411**
- device independence, 406**
- dialog boxes**
 - Add New Item, 106, 230
 - Advanced Appearance, 409-410
 - Appearance Settings, 409
 - custom dialog boxes, 387-391
 - Define Color, 49
 - New Project, 9-10, 34-35
 - Open File, 428-429
 - Save Project, 16
 - tabbed dialog boxes, 186-190
- DialogResult enumerations, 386**
- digital signatures, 504**
- Dim statement, 257-258**
- dimensioning**
 - arrays, 263
 - variables
 - explicit variable declaration, 259-260
 - simple example, 257-258
- directories, 440**
- Directory flag (files), 437**
- Directory object, 440**
- disconnecting from databases, 471**
- display settings, 12**

displaying

displaying

- design windows, 38
- forms, 118-119
 - initial display
 - position, 121-122
 - maximized
 - state, 120-121
 - minimized
 - state, 120-121
 - normal state, 120-121
- message boxes, 58-59
- object properties, 13
- Open File dialog
 - box, 428-429
- Picture Viewer log
 - files, 461-463
- properties, 45-47
- Registry options, 449-450
- text
 - labels, 155-156
 - text boxes, 157-163
- toolbars, 41
- Dispose() method, 77, 408, 457**
- distributable components.**
 - See programs**
- dithering, 49**
- division, 283**
- division (/) operator, 283**
- Do...Loop**
 - example, 332-334
 - exiting, 330-331
 - overview, 329
 - syntax, 330

docking

- design windows, 39-40
- toolbars, 42
- DoSomething() subroutine, 318**
- dot (.), 65, 68**
- Dot value (DashStyle property), 408**
- Double data type, 251**
- double-clicking mouse, 12**
- downward casting, 253**
- DrawBorder() function, 461**
- DrawEllipse() method, 414**
- drawing**
 - circles, 414
 - clearing drawing
 - surfaces, 415
 - controls on forms, 129
- Drawing project
 - button properties, 73
 - Clear() method, 75
 - CreateGraphics
 - method, 74
 - Dispose() method, 77
 - DrawRectangle()
 - method, 76-77
 - overview, 73
 - testing, 78
 - variables, 74
- ellipses, 414
- lines, 414
- rectangles, 412-414
- text, 415-416

Drawing project

- button properties, 73
- Clear() method, 75
- CreateGraphics method, 74
- Dispose() method, 77
- DrawRectangle()
 - method, 76-77
- overview, 73
- testing, 78
- variables, 74
- DrawLine() method, 414**
- DrawRectangle() method, 76-77, 414**
- DrawString() method, 415**
- drop-down lists, 177-179**
- dte prefix, 272**
- dynamic nature of methods, 72**

E

- e parameter (MouseDown event), 92**
- early-binding object variables, 374-376**
- editing**
 - Excel workbooks, 490-491
 - forms
 - background
 - color, 108-110
 - background images, 111-113
 - borders, 115-117

- close buttons, 114-115
- icons, 113-114
- maximize buttons, 114-115
- minimize buttons, 114-115
- minimum/maximum size, 117
- names, 106
- title bar text, 108
- properties, 45-47, 67-71
- records, 479
- elements of arrays, 263**
- ellipses, 414**
- Else statement, 308-309**
- Elself statement, 309-310**
- empty strings, 346**
- Enabled property (Text Box control), 160**
- enabling strict typing, 261-262**
- encapsulation, 64, 362-363**
- End Function statement, 237**
- End If statement, 26**
- End Sub statement, 232-233**
- endless loops, 245**
- enumerations**
 - DialogResult, 386
 - MessageBoxButtons, 383
 - MessageBoxIcon, 383-384
- Environment Tutorial**
 - design windows
 - auto-hiding, 40-41
 - docking, 39-40
 - floating, 38
 - showing/hiding, 38
 - states, 37-38
- overview, 36
- Properties window
 - color properties, 47-49
 - overview, 44
 - property descriptions, 49-50
 - selecting objects, 45
 - viewing and changing properties, 45-47
- Solution Explorer
 - adding/removing files, 55-57
 - overview, 50-52
 - project components, 53-54
 - project properties, 54
 - solutions, 52-53
- toolbars
 - docking, 42
 - overview, 41
 - resizing, 42
 - showing/hiding, 41
- toolbox, 43-44
- equals sign (=), 66**
- errors. See also debugging;**
- exceptions**
 - build errors, 340-343
 - runtime errors, 340-343
 - structured error handling, 351
 - unstructured error handling, 351
- evenly spacing controls, 137**
- event driven languages, 24**
- event-driven programming model, 85-86. See also events**
- event handling**
 - definition, 25
- event procedures, 86**
- events**
 - accessing, 89-90
 - Click, 24, 397
 - menus, 210-211
 - text boxes, 163
 - compared to methods, 86
 - definition, 24
 - event-driven programming model, 85-86
 - event handlers, 25
 - event procedures, 86
 - keyboard events, 393-396
 - KeyDown, 394
 - KeyPress, 394-396
 - KeyUp, 394
 - mouse events, 396-399
 - MouseDown, 92, 163, 397
 - MouseEnter, 397
 - MouseHover, 397
 - MouseLeave, 397
 - MouseMove, 93-98, 163, 397
 - MouseUp, 163, 397
 - MultilineChanged, 87
 - names, 98-99
 - overview, 85

events

- parameters, 90-93
- recursive events, 88
- sample project
 - event handlers, 94-98
 - user interface, 93-94
- SelectedIndexChanged, 189
- TextChanged, 87, 163
- Tick, 87, 186
- triggering
 - by objects, 87-88
 - by operating systems, 88
 - overview, 86
 - through user interaction, 87

Excel automation

- creating workbooks, 489
- instantiating automation server, 488-489
- making Excel visible, 489
- modifying workbooks, 490-491
- referencing type libraries, 486-488
- testing, 492

Exception object, 354**exceptions. *See also* errors**

- catching, 354
- Exception object, 354
- handling anticipated exceptions, 355-358
- StackOverflow, 88-89

Exclamation enumerator (MessageBoxIcon)_, 383

executables, 11**Exists() method**

- Directory object, 440
- File object, 432

Exit Do statement, 330**Exit For statement, 326****Exit Function statement, 244****Exit Sub statement, 244****Exit Try statement, 354****exiting**

- Do...Loop, 330-331
- For...Next loops, 326
- procedures, 244
- Try...End Try structure, 354

explicit variable

- declaration, 259-260**

exponentiation (^)

- operator, 283**

expressions, 258-259**F**

File menu commands, New Project, 36

File object. *See also* files

- Copy() method, 432-433
- Delete() method, 434-435
- Exists() method, 432
- GetAttributes() method, 436-437

GetCreationTime()

- method, 436

GetLastAccessTime()

- method, 436

GetLastWriteTime()

- method, 436

Move() method, 433-434

FileLen() function, 439**files****attributes**

- determining, 436-437
- viewing in Picture Viewer program, 437-439

browsing, 24-27

copying, 432-433

deleting, 434-435

determining whether file exists, 432

File object

- Copy() method, 432-433

- Delete() method, 434-435

- Exists() method, 432

- GetAttributes() method, 436-437

- GetCreationTime() method, 436

- GetLastAccessTime() method, 436

- GetLastWriteTime() method, 436

- Move() method, 433-434

- log files
 - creating, 459-461
 - displaying, 461-463
 - testing, 463-464
- moving, 433-434
- OpenFileDialog control
 - creating file filters, 428
 - overview, 425-428
 - showing Open File dialog box, 428-429
- project files, 55-57
- renaming, 434
- returning date/time
 - information about, 436
- SaveFileDialog control, 429-431
- text files
 - Picture Viewer log files, 459-464
 - reading, 457-459
 - writing to, 455-457
- Fill() method, 472**
- filters, 428**
- Finally statement, 352**
- floating design windows, 38**
- Font object, 415**
- For statement, 324**
- For...Next loops**
 - continuing before Next is reached, 326
 - example, 326-329
 - exiting, 326
 - For statement, 324
 - Next statement, 324-325
 - Step keyword, 325-326
 - syntax, 323
- form modules, 229**
- Format() function, 300-302**
- Format16bppGrayScale value (PixelFormat argument), 407**
- Format16bppRgb555 value (PixelFormat argument), 407**
- Format24bppRgb value (PixelFormat argument), 407**
- formatting dates/times, 300-301**
- FormBorderStyle property (forms), 116-117**
- forms**
 - background color, 108-110
 - background images, 111-113
 - borders, 115-117
 - close buttons, 114-115
 - Collections Example, 79-81
 - context menus
 - assigning shortcut keys to, 214-215
 - creating, 212-214
 - controlling size of, 117
 - controls
 - adding, 43-44, 128-129
 - alignment, 135-136
 - anchoring, 138-141
 - Button. See buttons
 - CheckBox, 166
 - ComboBox, 177-179
 - Context Menu Strip, 212-214
 - definition, 19, 155
 - evenly spacing, 137
 - Group Box, 167
 - Image List, 190-191
 - invisible controls, 22-24
 - Label, 155-156
 - layering, 144
 - List View, 191-196
 - ListBox, 170-177
 - making controls same size, 137
 - Menu Strip, 204-206
 - OpenFileDialog controls, 22-24
 - Panel, 167-168
 - PictureBox controls, 21-22, 30
 - properties, 137-138
 - Radio Button, 168-170
 - selecting groups of, 133-135
 - snapping to grid, 129-131
 - snapping to lines, 132-133
 - Status Bar, 221-223
 - Tab, 186-190
 - Text Box, 157-163
 - Timer, 183-186
 - ToolStrip, 216-219

forms

- Tree View, 197-200
- visible controls, 20-22
- definition, 11
- displaying
 - initial display
 - position, 121-122
 - maximized state, 120-121
 - minimized state, 120-121
 - normal state, 120-121
- form classes, 15
- Graphic objects, 406
- hiding, 123
- icons, 16-17, 113-114
- lists. *See* lists
- maximize buttons, 114-115
- MDI (multiple-document interface) forms, 147-150
- minimize buttons, 114-115
- modal versus nonmodal, 119-120
- names, 13-15, 106
- overview, 105-106
- properties
 - ActiveMdiChild, 150
 - AutoScroll, 146
 - AutoScrollMargin, 146
 - AutoScrollMinSize, 146
 - BackColor, 108-110
 - BackgroundImage, 111-113
 - FormBorderStyle, 116-117
 - Height, 17-18
 - Icon, 17, 113
 - IsMdiContainer, 148
 - MaximumSize, 117
 - MdiParent, 150
 - MinimumSize, 117
 - Name, 13-15, 106
 - Opacity, 145
 - ShowInTaskbar, 123
 - StartPosition, 121-122
 - Text, 15-16, 108
 - TopMost, 144
 - Visible, 123
 - Width, 17-18
 - WindowState, 120-121
- resizing, 17-19
- scrollable forms, 145-146
- showing, 118-119
- Startup forms, 150-151
- status bars, 221-223
- tab order, 142-144
- templates, 15
- title bars, 108
- toolbars
 - buttons, 217-219
 - creating, 216
 - drop-down menus for, 221
 - programming, 220-221
- top-level menus
 - assigning shortcut keys to, 214-215
 - checked menu items, 208-209
 - creating, 204-206
 - creating menu items, 207
 - deleting menu items, 208
 - moving menu items, 208
 - programming, 210-212
- topmost nonmodal windows, 144
- transparent forms, 145
- unloading, 123-124
- forward references, 13**
- frames, 167**
- Friend keyword, 272, 367**
- FromImage() method, 408**
- FullRowSelect property (ListView control), 195**
- Function keyword, 58, 237**
- functions. *See also specific functions***
 - calling, 238-241
 - declaring, 237
 - definition, 232
 - exiting, 244
 - parameters, 242-244
 - recursive loops, 245

G**g_ prefix**, 273**garbage collection**, 514-515**GDI (Graphical Device Interface)**, 406**Get statement**, 368-369**GetAttributes()** method, 436-437**GetCreationTime()** method, 436**GetLastAccessTime()** method, 436**GetLastWriteTime()** method, 436**GetSetting()** function, 445**GetValue()** method, 448**global scope**, 269-270**GoTo statement**, 317-318**GotoExample()** subroutine, 317**Graphical Device Interface (GDI)**, 406**graphics**

adding to form back-grounds, 111-113

circles, 414

ellipses, 414

GDI (Graphical Device Interface), 406

Graphics object

creating for bitmaps, 407-408

creating for forms/controls, 406

overview, 405-406

icons

adding to forms, 16-17

adding to message

boxes, 383-385

assigning to

forms, 113-114

lines, 414

overview, 405

pens

creating, 408

DashStyle property, 408-409

definition, 408

Persisting Graphics project example, 417-422

persisting on forms, 416

rectangles, 412-414

storing in image

lists, 190-191

system colors, 409-412

text, drawing, 415-416

Graphics object. See**also graphics**

creating for bitmaps, 407-408

creating for forms/controls, 406

overview, 405-406

GrayText system color, 411**grid, snapping controls to**, 129-131**GridSize setting (grid)**, 130-131**Group Box control**, 167**group boxes**, 167**groups of controls**

properties, 137-138

selecting, 133-135

spacing, 137

H**Handles keyword**, 99**handling exceptions**, 355-358**Height property (forms)**, 17-18**Help**, 59**Hidden flag (files)**, 437**Hide()** method, 124**hiding**

code windows, 12

design windows, 38

forms, 123

toolbars, 41

Highlight system color, 411**HighlightText system color**, 411**HKEY_CLASSES_ROOT node (Registry)**, 444**HKEY_CURRENT_CONFIG node (Registry)**, 444**HKEY_CURRENT_USER node (Registry)**, 444, 447**HKEY_LOCAL_MACHINE node (Registry)**, 444**HKEY_USERS node (Registry)**, 444**hyphen (-)**, 283

Icon property**I****Icon property (forms), 17, 113****icons**

- adding to forms, 16-17
- adding to message boxes, 383-385
- assigning to forms, 113-114

IDE (integrated development environment), 9**If...Then statement**

- Elseif statement, 309-310
- If...Then...Else, 308-309
- nesting, 310
- simple example, 306-308

If...Then...Else statement, 308-309**Ignore enumerator (DialogResult), 386****IL (Intermediate Language), 511-512****Image List control, 190-191****ImageIndex property (List View control), 193****images**

- adding to form
 - backgrounds, 111-113
- circles, 414
- ellipses, 414
- GDI (Graphical Device Interface), 406

Graphics object

- creating for bitmaps, 407-408
- creating for forms/controls, 406
- overview, 405-406

icons

- adding to forms, 16-17
- adding to message boxes, 383-385
- assigning to forms, 113-114

lines, 414**overview, 405****pens**

- creating, 408
- DashStyle property, 408-409
- definition, 408

Persisting Graphics project example, 417—422**persisting on forms, 416****rectangles, 412-414**

storing in image lists, 190-191

system colors, 409-412**text, drawing, 415-416****ImageSize property (Image control), 191****Immediate window, 346-351****importing namespaces, 468-469****InactiveBorder system color, 411****InactiveCaption system color, 411****InactiveCaptionText system color, 411****Inflate() method, 413****Information enumerator (MessageBoxIcon), 383****inheritance, 64****initializing variables, 275-278****InputBox() function, 391-393****Insert() method, 173****instantiation, 74****automation servers**

- Excel, 488-489
- Word, 493

objects

- early-binding object variables, 374-376
- in variable declarations, 376
- late-binding object variables, 372-374
- object lifetimes, 377-378
- overview, 371-372
- releasing object references, 377

Instr() function, 293-294**int prefix, 272****Integer data type, 251****integrated development environment (IDE), 9****IntelliSense, 68**

interfaces

- creating, 11, 364-366
- definition, 363
- methods
 - creating, 371
 - definition, 365
- properties
 - declaring, 366-368
 - definition, 365
 - read-only
 - properties, 370
 - readable
 - properties, 368-369
 - writable properties, 369
 - write-only
 - properties, 370

Intermediate Language (IL), 511-512**invisible at runtime controls, 22-24****invisible controls, 22-24****invoking methods, 72****IsDate() function, 301****IsMdiContainer property (forms), 148****IsNumeric() function, 307, 349****items**

- list items
 - adding, 172-173, 193-196
 - clearing, 175

removing, 173-174, 196

retrieving information
about, 175-176

sorting, 177

menu items

checked menu items,
208-209

creating, 207

deleting, 208

moving, 208

Items collection, 171-172, 217-219**iterative processing, 78****J-K****JITter (just-in-time compiler), 512****keyboard events, handling, 393-396****KeyDown event, 394****KeyPress event, 394-396****keys****Registry keys**

creating, 446-447

deleting, 447

getting/setting key
values, 448-449

shortcut keys,
assigning to
menus, 214-215

KeyUp event, 394**keywords. See also statements**

As, 58

ByRef, 243

ByVal, 243

definition, 69

Friend, 272, 367

Function, 58, 237

Handles, 99

Let, 66

Me, 69

Mod, 284

New, 372, 376-377

Private, 91, 232, 268, 367

prohibited in names, 258

Public, 232, 269, 367

ReadOnly, 370

Return, 237

Step, 325-326

Sub, 58, 91, 233

To, 312

WriteOnly, 370

L**Label control, 155-156****labels**

code labels, 317

creating, 155-156

languages

languages

common language
runtime, 510-511

IL (Intermediate Language),
511-512

machine language, 512

**LargeImageList property (List
View control), 194**

**late-binding object variables,
372-374**

layering controls, 144

LayoutMode setting (grid), 130

Left() function, 291-292

Len() function, 291

Let keyword, 66

libraries

definition, 81

type libraries, 492-493

lifetimes of objects, 377-378

line continuation

character (`_`), 77

lines, 414

snapping controls
to, 132-133

list boxes

adding items to
lists, 172-173

clearing lists, 175

Items collection, 171-172

overview, 170-171

removing items from
lists, 173-174

retrieving item

information, 175-176

sorting lists, 177

List View control

adding list items, 193-196

columns, 192-193

overview, 191-192

removing all list items, 196

removing list items, 196

ListBox control

adding items to

lists, 172-173

clearing lists, 175

Items collection, 171-172

overview, 170-171

removing items from
lists, 173-174

retrieving item information,
175-176

sorting lists, 177

lists

adding items to, 172-173,
193-196

clearing, 175, 196

columns, 192-193

drop-down lists, 177-179

image lists, 190-191

list boxes

adding items to
lists, 172-173

clearing lists, 175

Items collection,
171-172

overview, 170-171

removing items from
lists, 173-174

retrieving item

information, 175-176

sorting lists, 177

List Views

adding list

items, 193-196

columns, 192-193

overview, 191-192

removing all list
items, 196

removing list items, 196

removing items from,
173-174, 196

retrieving item
information, 175-176

sorting, 177

literal values, 258

lng prefix, 272

loading Registry

options, 451-452

local (procedure-level)

scope, 267-268

Location property

(MouseEventArgs), 92

log files

creating, 459-461

displaying, 461-463

testing, 463-464

logic, Boolean

- And operator, 288
- definition, 281
- Not operator, 288-289
- Or operator, 289
- overview, 287-288
- Xor operator, 289

Long data type, 251**loops**

- definition, 323
- Do...Loop
 - example, 332-334
 - exiting, 330-331
 - overview, 329
 - syntax, 330
- For...Next
 - continuing before Next is reached, 326
 - example, 326-329
 - exiting, 326
 - For statement, 324
 - Next statement, 324-325
 - Step keyword, 325-326
 - syntax, 323
 - recursive loops, 245
 - While...End While, 334

LTrim() function, 295**M****m_ prefix, 273****machine language, 512****macros, 490****magic numbers, 254****managed code, 510****Manual value (StartPosition property), 122****math**

- addition, 282
- division, 283
- exponentiation, 283
- modulus arithmetic, 284
- multiplication, 283
- negation, 283
- operator precedence, 284-286
- overview, 281-282
- subtraction, 283

maximize buttons, 114-115**maximized state (forms), 120-121****maximum length of text boxes, 161-162****MaximumSize property (forms), 117****MaxLength property (Text Box control), 161-162****MDI (multiple-document interface) forms, 147-150****MdiParent property (forms), 150****Me keyword, 69****Me.Close() statement, 28****menu commands**

- File menu, New Project, 36
- Project menu
 - Add Class, 364
 - Add Module, 230
 - Automate Excel Properties, 486
 - Automate Word Properties, 492
- View menu, Properties Window, 38

Menu Strip control, 204-206**Menu system color, 411****menus**

- context menus
 - assigning shortcut keys to, 214-215
 - creating, 212-214
- overview, 204
- top-level menus
 - assigning shortcut keys to, 214-215
 - checked menu items, 208-209
 - creating, 204-206
 - creating menu items, 207
 - deleting menu items, 208
 - moving menu items, 208
 - programming, 210-212

MenuText system color**MenuText system color, 411****message boxes**

- buttons/icons, 383-385
- determining which button is clicked, 385-386
- displaying, 58-59, 381-382
- message-writing guidelines, 386-387

MessageBox.Show()

- method, 58, 80

MessageBoxButtons, 383**MessageBoxIcon, 383-384****metadata, 514****methods. *See also specific methods***

- definition, 71
- dynamic nature of, 72
- invoking, 72

Microsoft.VisualBasic

- namespace, 513

Mid() function, 292-293**minimize buttons, 114-115****minimized state**

- (forms), 120-121

MinimumSize property

- (forms), 117

Mod keyword, 284**modal forms, 119-120****modifying**

- Excel workbooks, 490-491
- forms
 - background color, 108-110
 - background images, 111-113

- borders, 115-117

- close buttons, 114-115

- icons, 113-114

- maximize buttons, 114-115

- minimize buttons, 114-115

- minimum/maximum size, 117

- names, 106

- title bar text, 108

- properties, 45-47, 67-71

- records, 479

module-level scope, 268**modules**

- compared to classes, 363-364
- definition, 229
- form modules, 229
- standard (class) modules, 229-231

modulus arithmetic, 284**mouse**

- double-clicking, 12
- event handling, 396-399
 - MouseDown event, 92, 163, 397
 - MouseEnter event, 397
 - MouseHover event, 397
 - MouseLeave event, 397
 - MouseMove event, 93-98, 163, 397
 - MouseUp event, 163, 397

- MouseDown event, 92, 163, 397

- MouseEnter event, 397

- MouseHover event, 397

- MouseLeave event, 397

- MouseMove event, 93-98, 163, 397

- MouseUp event, 163, 397

Move() method

- Directory object, 440
- File object, 433-434

moving

- directories, 440
- files, 433-434
- menu items, 208

multidimensional

- arrays, 264-266

- Multiline property (Text Box control), 158

- multiline text boxes, 158-160

- MultilineChanged event, 87

- multiple-document interface (MDI) forms, 147-150

- multiplication (*) operator, 283

N

- Name property, 13-15, 106

- namespace (global) scope, 269-270

namespaces, 514

- importing, 468-469
- overview, 513

- System.Data, 468
- table of, 513
- naming conventions**
 - constants, 255
 - controls, 273
 - data types, 272
 - events, 98-99
 - forms, 106
 - name conflicts (scope), 270
 - objects, 13-15
 - reserved words, 258
 - variables, 273
- navigating records, 477-479**
- negation (-) operator, 283**
- nesting If...Then statements, 310**
- .NET Framework**
 - common language runtime, 510-511
 - common type system, 514
 - definition, 8, 510
 - garbage collection, 514-515
 - IL (Intermediate Language), 511-512
 - namespaces, 514
 - importing, 468-469
 - overview, 513
 - System.Data, 468
 - table of, 513
 - overview, 509
 - recommended reading, 515

- New keyword, 372, 376-377**
- New Project command (File menu), 36**
- New Project dialog box, 9-10, 34-35**
- NewRow() method, 480**
- Next statement, 324-325**
- No enumerator (DialogResult), 386**
- nodes (Tree View)**
 - adding, 197-199
 - clearing, 200
 - definition, 81
 - removing, 199-200
- Nodes collection**
 - Add() method, 197-199
 - Clear() method, 200
 - Remove() method, 199-200
- None enumerator**
 - DialogResult, 386
 - MessageBoxIcon, 383
- nonmodal forms, 119-120**
- nonvisual controls, 22-24**
- Normal flag (files), 437**
- normal state (forms), 120-121**
- Not operator, 288-289**
- Nothing value (variables), 377**
- numbers, magic, 254**

O

- obj prefix, 272**
- Object Browser, 81**
- Object data type, 251**
- object libraries. creating**
 - references to
 - Excel, 486-488
 - Word, 492-493
- object models, 485**
- object-oriented programming (OOP), 230**
- object-oriented programming, 64**
- objects**
 - ActiveCell, 490
 - ADO.NET objects.
 - See ADO.NET
 - Application, 488
 - collections
 - Collections Example form, 79-81
 - definition, 78
 - iterative processing, 78
 - structure of, 79
 - CommandBuilder, 473
 - containers, 167
 - control objects.
 - See controls
 - definition, 64
 - Directory, 440
 - Exception, 354

objects

File. *See also* files

Copy() method, 432-433

Delete() method,
434-435

Exists() method, 432

GetAttributes()
method, 436-437

GetCreationTime()
method, 436

GetLastAccessTime()
method, 436

GetLastWriteTime()
method, 436

Move() method, 433-434

Font, 415

form objects. *See* forms

Graphics. *See also* graphics

creating for bitmaps,
407-408

creating for forms/
controls, 406

overview, 405-406

instantiation, 74

early-binding object
variables, 374-376

in variable
declarations, 376

late-binding object
variables, 372-374

object lifetimes,
377-378

overview, 371-372

releasing object
references, 377

methods

definition, 71

dynamic nature of, 72

invoking, 72

naming, 13-15

Object Browser, 81

object-oriented
programming, 64

overview, 63-64

properties

color properties, 47-49

definition, 13

displaying, 13

getting, 65-67

modifying, 67-71

overview, 65

property
descriptions, 49-50

referencing, 65-66

setting, 65-67

viewing and
changing, 45-47

Range, 490

Rectangle, 412-413

sample project

interface, 73

object-based
code, 74-77

overview, 73

testing, 78

selecting in Properties
window, 45

Selection, 494

SqlConnection, 468

triggering events, 87-88

OK enumerator

(DialogResult), 386

OK enumerator

(MessageBoxButtons), 383

OKCancel enumerator

(MessageBoxButtons), 383

OleDbConnection object

Close() method, 471

connecting to databases,
469-471

ConnectionString
property, 470

definition, 468

disconnecting from
databases, 471

Open() method, 471

On Error statements, 351

**OOP (object-oriented
programming), 230**

Opacity property (forms), 145

Open File dialog box, 428-429

Open() method, 471

OpenFileDialog control

adding to forms, 22-24

creating file filters, 428

overview, 425-428

showing Open File dialog
box, 428-429

opening projects, 36

**OpenPicture() function,
234-235, 356, 459**

Picture Viewer program

operators

- addition (+), 282
- And, 288
- comparison operators, 286-287
- concatenation (&), 290-291
- division (/), 283
- exponentiation (^), 283
- multiplication (*), 283
- negation (-), 283
- Not, 288-289
- Or, 289
- precedence, 284-286
- subtraction (-), 283
- Xor, 289

Or operator, 289

P

Panel control, 167-168

parameters

- defined, 91
- definition, 58, 90
- overview, 91-93
- passing, 242-244

parentheses (), 302

passing

- literal values to variables, 258
- parameters, 242-244

PasswordChar, 162

PasswordChar property (Text Box control), 162

passwords, 162-163

Pen() method, 408

pens

- creating, 408
- DashStyle property, 408-409
- definition, 408

period (.), 65, 68

persistence of graphics, 416

Persisting Graphics

project, 417-422

picShowPicture_MouseMove procedure, 95

Picture Viewer program

browsing files, 24-27

ClickOnce file

- creating, 500-503
- testing, 504
- uninstalling, 504-505

controls

- Button controls, 20-21
- definition, 19
- invisible controls, 22-24
- OpenFileDialog controls, 22-24
- PictureBox controls, 21-22, 30
- visible controls, 20-22

drawing button

- Clear() method, 75
- CreateGraphics method, 74

Dispose() method, 77

DrawRectangle() method, 76-77

overview, 73

properties, 73

variables, 74

file properties, 437-439

log files

creating, 459-461

displaying, 461-463

testing, 463-464

MouseMove event

event handlers, 94-98

user interface, 93-94

project, creating, 9-11

quitting, 27

Registry

displaying options

from, 449-450

loading options

from, 451-452

saving options to, 451

running, 28-29

saving, 16

testing, 78, 452-455

variables

creating, 274

initializing, 275-278

ViewerForm

icon, 16-17

naming, 13-15

resizing, 17-19

Text property, 15-16

PictureBox controls

PictureBox controls, 21-22, 30

pictures. *See* graphics

PixelFormat argument

(Bitmap() method), 407

pixels, 17

plus sign (+), 51, 282

polymorphism, 64

populating DataTables, 474

pound sign (#), 258

*Practical Standards for
Microsoft Visual Basic .NET,
Second Edition*, 273, 319

precedence of operators,
284-286

prefixes

for controls, 273

for data types, 272

for variables, 273

Private keyword, 91,
232, 268, 367

procedural languages, 85

procedure-level scope, 267-268

procedures. *See also*
specific procedures

calling, 238-241

declaring, 58

definition, 58, 229, 232

event procedures, 86

exiting, 244

parameters, 242-244

picShowPicture_

MouseMove, 95

recursive loops, 245

subroutines, 232-236

processor independent
code, 512

programming

menus, 210-212

toolbars, 220-221

programs

automation

definition, 485

Excel, 486-492

overview, 485-486

Word, 492-493

ClickOnce applications

advanced settings, 505

creating with Publish

Wizard, 500-503

overview, 499-500

testing, 504

uninstalling, 504-505

creating, 11

definition, 8, 52

executables, 11

interfaces, 11

Picture Viewer. *See* Picture
Viewer program

quitting, 27

self-contained

applications, 500

Project menu commands

Add Class, 364

Add Module, 230

Automate Excel

Properties, 486

Automate Word

Properties, 492

projects. *See also*
specific projects

components, 53-54

creating, 9-11, 34-36

definition, 52

managing with Solution
Explorer

adding/removing

files, 55-57

overview, 50-52

project components,
53-54

project properties, 54

solutions, 52-53

opening, 36

properties, 54

running, 28-29

saving, 16

properties. *See also*
specific properties

button properties, 67-71

color properties, 47-49

control properties, 137-138

declaring, 366-368

definition, 13, 365

displaying, 13

getting, 65

modifying, 67-71

object properties, 65-67

project properties, 54

property descriptions, 49-50

read-only properties,
67, 370

- readable properties, 368-369
- referencing, 65-66
- setting, 65
- viewing and
 - changing, 45-47
- writable properties, 369
- write-only properties, 370
- Properties window, 12**
 - color properties, 47-49
 - overview, 44
 - property descriptions, 49-50
 - selecting objects, 45
 - viewing and changing properties, 45-47
- Properties Window command (View menu), 38**
- Public Function statement, 237**
- Public keyword, 232, 269, 367**
- Public Sub statement, 232-233**
- Publish Wizard, 500-503**
- Q-R**
- Question enumerator (MessageBoxIcon), 383**
- Radio Button control, 168-170**
- radio buttons, 168-170, 315-316**
- Range object, 490**
- read-only properties, 67, 370**
- readable properties, 368-369**
- reading text files, 457-459**
- ReadLine() method, 458**
- ReadOnly keyword, 370**
- ReadToEnd() method, 458**
- ReadyOnly flag (files), 437**
- records**
 - creating, 480-481
 - deleting, 481-482
 - editing, 479
 - navigating, 477-479
- rectangles, 412-414**
- recursive events, 88**
- recursive loops, 245**
- reference, passing by, 242-244**
- reference-tracing garbage collection, 515**
- referencing**
 - arrays, 263-264
 - DataRows, 475-476
 - object properties, 65-66
 - type libraries
 - Excel, 486-488
 - Word, 492-493
- REG_BINARY data type, 445**
- REG_EXPAND_SZ data type, 445**
- REG_MULTI_SZ data type, 445**
- REG_SZ data type, 445**
- regedit command, 455**
- Registry**
 - accessing, 445-446
 - displaying options from, 449-450
- keys**
 - creating, 446-447
 - deleting, 447
 - getting/setting key values, 448-449
 - loading options from, 451-452
 - overview, 443
 - regedit command, 455
 - saving options to, 451
 - structure, 444-445
- releasing object references, 377**
- Remove() method**
 - Items collection, 173-174, 196
 - Nodes collection, 199-200
- RemoveAt() method, 174**
- removing**
 - directories, 440
 - files, 434-435
 - list items, 173-174, 196
 - menu items, 208
 - project files, 55-57
 - records, 481-482
 - Registry keys, 447
 - Tree View nodes, 199-200
- renaming files, 434**
- Replace() function, 295**
- replacing text within strings, 295**

reserved words**reserved words**

- As, 58
- ByRef, 243
- ByVal, 243
- definition, 69
- Friend, 272, 367
- Function, 58, 237
- Handles, 99
- Let, 66
- Me, 69
- Mod, 284
- New, 372, 376-377
- Private, 91, 232, 268, 367
- prohibited in names, 258
- Public, 232, 269, 367
- ReadOnly, 370
- Return, 237
- Step, 325-326
- Sub, 58, 91, 233
- To, 312
- WriteOnly, 370

resizing

- forms, 17-19
- toolbars, 42

resolution (screen), 12**Retry enumerator**

- (DialogResult), 386

RetryCancel enumerator

- (MessageBoxButtons), 383

Return keyword, 237**Right() function, 292****routines. See procedures****rows, 475-476****RTrim() function, 295****running**

- database example, 482
- projects, 28-29

runtime errors, 340-343**S****s_ prefix, 273**

Sams Teach Yourself Object-Oriented Programming with Visual Basic .NET in 21 Days, Second Edition, 230

Save Project dialog box, 16

SaveFileDialog control, 429-431

SaveSetting() function, 445**saving**

- projects, 16
- Registry options, 451

Sbyte data type, 251**scope**

- block scope, 267
- browsing scope, 81
- definition, 250
- global scope, 269-270
- module-level scope, 268
- name conflicts, 270
- naming conventions, 273
- overview, 266

procedure-level

- scope, 267-268

screen resolution, 12**scrollable forms, 145-146****scrollbars, 160**

Scrollbars property (Text Box control), 160

SDI (single-document interface) forms. See forms, 147

security, digital signatures, 504**Select Case statement**

- example, 312-315
- multiple comparisons, 311-312
- overview, 310-311
- radio buttons, evaluating, 315-316

Select() method, 490

SelectedIndexChanged event, 189

selecting

- controls, 133-135
- objects, 45

Selection object, 494**self-contained applications, 500****SendToBack() method, 144****servers**

- automation servers, 488-489, 493
- definition, 363

Set statement, 369**SetValue() method, 448**

- shapes**
 - circles, 414
 - ellipses, 414
 - lines, 414
 - rectangles, 412-414
- sho prefix, 272**
- Short data type, 251**
- shortcut keys, assigning to menus, 214-215**
- shortcut menus**
 - assigning shortcut keys to, 214-215
 - creating, 212-214
- Show() method, 58, 80, 119, 381-382**
- ShowCurrentRecord() method, 476-477**
- ShowDialog() method, 25, 119**
- ShowGrid setting (grid), 130-132**
- showing**
 - design windows, 38
 - forms, 118-119
 - initial display position, 121-122
 - maximized state, 120-121
 - minimized state, 120-121
 - normal state, 120-121
 - message boxes, 58-59
 - object properties, 13
 - Open File dialog box, 428-429
 - Picture Viewer log files, 461-463
 - properties, 45-47
 - Registry options, 449-450
 - text
 - labels, 155-156
 - text boxes, 157-163
 - toolbars, 41
- ShowInTaskbar property (forms), 123**
- ShowInVisible Taskbar property (forms), 123**
- signatures, digital, 504**
- Single data type, 251**
- single-document interface (SDI) forms. See forms**
- sizing**
 - controls, 137
 - forms, 17-19, 117
 - grips, 222
 - toolbars, 42
- slash (/), 283**
- Sleep() function, 328**
- Snap to Lines layout feature, 132-133**
- SnapToGrid setting (grid), 130**
- sng prefix, 272**
- Solidvalue (DashStyle property), 408**
- Solution Explorer**
 - adding/removing project files, 55-57
 - overview, 50-52
 - project components, 53-54
 - project properties, 54
- solutions, 8, 52-53**
- Sorted property (ListBox control), 177**
- sorting lists, 177**
- spacing controls, 137**
- spaghetti code, 317**
- SqlConnection object, 468**
- StackOverflow exception, 88-89**
- stacks, 245**
- standard (class) modules**
 - compared to classes, 363-364
 - creating, 230-231
 - definition, 229
- Start Page**
 - creating new projects, 34-36
 - opening existing projects, 36
 - overview, 34
- starting Visual Basic 2008, 8-9**
- StartPosition property (forms), 121-122**
- Startup forms, 150-151**
- statements. See also keywords; loops**
 - Catch, 352-355
 - definition, 96
 - Dim, 257-258
 - Else, 308-309
 - Elseif, 309-310

statements

- End Function, 237
- End If, 26
- End Sub, 232-233
- Exit Do, 330
- Exit For, 326
- Exit Function, 244
- Exit Sub, 244
- Exit Try, 354
- Finally, 352
- For, 324
- Get, 368-369
- GoTo, 317-318
- If...Then
 - Elself statement, 309-310
 - If...Then...Else, 308-309
 - nesting, 310
 - simple example, 306-308
- Next, 324-325
- On Error, 351
- Public Function, 237
- Public Sub, 232-233
- Select Case
 - example, 312-315
 - multiple comparisons, 311-312
 - overview, 310-311
 - radio buttons, evaluating, 315-316
- Set, 369
- Try, 352
 - Try...Catch...Finally, 319, 351-354
- states, 37-38**
- static text, displaying with Label control, 155-156**
- static variables, 270-271**
- Status Bar control, 221-223**
- status bars, 221-223**
- Step keyword, 325-326**
- stepping into code, 345**
- stepping out of code, 345**
- stepping over code, 345**
- Stop enumerator (MessageBoxIcon), 384**
- str prefix, 272**
- StreamReader class, 457-459**
- StreamWriter class, 455-457**
- strict typing**
 - definition, 259-260
 - enabling, 261-262
- String data type, 251**
- strings**
 - concatenation, 80, 96, 290-291
 - determining number of characters in, 291
 - determining whether one string contains another, 293-294
 - replacing text within, 295
 - retrieving text from left side of, 291-292
 - retrieving text from right side of, 292
 - retrieving text within, 292-293
 - trimming, 294-295
 - zero-length strings, 346, 458
- structure (block) scope, 267**
- structured error handling, 351**
- structures, DateTime, 297, 301**
- Sub keyword, 58, 91, 233**
- SubItems property (List View control), 193**
- subroutines. *See also* specific subroutines**
 - calling, 238-241
 - declaring, 232-236
 - definition, 232
 - exiting, 244
 - parameters, 242-244
 - recursive loops, 245
- subtraction (-) operator, 283**
- suffixes (names), 15**
- system colors, 409-412**
- System flag (files), 437**
- System namespace, 513**
- System.Data namespace, 468, 513**
- System.Diagnostics namespace, 513**
- System.Drawing namespace, 513**
- System.IO namespace, 513**

System.Net namespace, 513
System.Security namespace, 513
System.Web namespace, 514
System.Windows.Forms namespace, 514
System.XML namespace, 514

T

Tab control, 186-190
tab order (forms), 142-144
tabbed dialog boxes, 186-190
TabIndex property (controls), 142-144
tables
 creating records, 480-481
 definition, 468
 deleting records, 481-482
 editing records, 479
 navigating records, 477-479
 overview, 472-474
 populating, 474
TabPage collection, 187
taskbar, hiding forms in, 123
templates, 15
Temporary flag (files), 437
terminating programs, 27
testing
 ClickOnce applications, 504
 drawing project, 78
 Excel automation, 492

Picture Viewer program, 452-455, 463-464

text

displaying on form title bars, 108
drawing, 415-416
Font object, 415
labels, 155-156
strings
 concatenation, 80, 96, 290-291
 determining number of characters in, 291
 determining whether one string contains another, 293-294
 replacing text within, 295
 retrieving text from left side of, 291-292
 retrieving text from right side of, 292
 retrieving text within, 292-293
 trimming, 294-295
 zero-length strings, 346, 458

text boxes

events, 163
maximum length, 161-162
multiline text boxes, 158-160
overview, 157

password fields, 162-163
scrollbars, 160
text alignment, 157-158
text files
 Picture Viewer log files, 459-464
 reading, 457-459
 writing to, 455-457

Text Box control

events, 87, 163
maximum length, 161-162
multiline text boxes, 158-160
overview, 157
password fields, 162-163
scrollbars, 160
text alignment, 157-158

Text property

forms, 15-16, 108
Label control, 156

TextAlign property (Text Box control), 157-158

TextChanged event, 87, 163

Tick event, 87, 186

TimeOfDay() function, 185

Timer control, 87-88, 183-186

times

adding to/subtracting from, 297-298
determining intervals between, 298-299

times

- file date/time information, returning, 436
- formatting, 300-301
- retrieving current, 301
- title bars (forms), 108**
- To keyword, 312**
- toolbars**
 - adding toolbar buttons, 217-219
 - creating, 216
 - docking, 42
 - drop-down menus for, 221
 - overview, 41
 - programming, 220-221
 - resizing, 42
 - showing/hiding, 41
- Toolbox window, 12, 43-44**
- ToolStrip control, 216-219.**
 - See also toolbars**
- top-level menus**
 - assigning shortcut keys to, 214-215
 - checked menu items, 208-209
 - creating, 204-206
 - creating menu items, 207
 - deleting menu items, 208
 - moving menu items, 208
 - programming, 210-212
- topmost nonmodal windows, 144**
- TopMost property (forms), 144**
- traditional controls.**
 - See controls**
- transparent forms, 145**
- TransparentColor property (Image List control), 191**
- Tree View control**
 - adding nodes, 197-199
 - clearing all nodes, 200
 - overview, 197
 - removing nodes, 199-200
- triggering**
 - events
 - by objects, 87-88
 - by operating systems, 88
 - overview, 86
 - through user interaction, 87
 - methods, 72
- Trim() function, 295**
- trimming strings, 294-295**
- Try statement, 352**
- Try...Catch...Finally structure, 319, 351-354**
- turning on strict typing, 261-262**
- type libraries, creating references to**
 - Excel, 486-488
 - Word, 492-493
- types**
 - Boolean, 251
 - Byte, 251
 - casting, 253-254
 - Char, 251
 - common type system, 514
 - Date, 251, 296-297
 - Decimal, 251
 - definition, 250
 - determining, 252
 - Double, 251
 - guidelines for use, 252
 - Integer, 251
 - Long, 251
 - naming conventions, 272
 - Object, 251
 - overview, 250
 - REG_BINARY, 445
 - REG_EXPAND_SZ, 445
 - REG_MULTI_SZ, 445
 - REG_SZ, 445
 - Sbyte, 251
 - Short, 251
 - Single, 251
 - strict typing
 - definition, 259-260
 - enabling, 261-262
 - String, 251
 - type conversion functions, 253
 - UInteger, 251
 - ULong, 251
 - UShort, 251
- TypeText() method, 494**

U

UInteger data type, 251
ULong data type, 251
underscore (_), 77, 90, 233
uninstalling ClickOnce applications, 504-505
unloading forms, 123-124
unmanaged code, 510
unstructured error handling, 351
Update() method, 472, 479
upward casting, 253
user interaction
 custom dialog
 boxes, 387-391
 InputDialog() function, 391-393
 keyboard events, 393-396
 message boxes
 determining which button is clicked, 385-386
 displaying buttons/icons, 383-385
 displaying with MessageBox.Show() function, 381-382
 message-writing guidelines, 386-387
 mouse events, 396-399
 overview, 381
 triggering events, 87
UShort data type, 251

V

value, passing by, 242-244
values, storing in variables, 57
variables. See also data types
 arrays. *See* arrays
 Date, 296-297
 declaring, 57, 257
 explicit variable declaration, 259-260
 simple example, 257-258
 static variables, 270-271
 definition, 57, 249
 Drawing project, 74
 in expressions, 258-259
 naming conventions, 273
 passing literal values to, 258
 Picture Viewer
 program variables creating, 274
 initializing, 275-278
 scope
 block scope, 267
 global scope, 269-270
 module-level scope, 268
 name conflicts, 270
 overview, 266
 procedure-level scope, 267-268
 static variables, 270-271

View menu commands, Properties Window, 38

ViewerForm

 icon, 16-17
 naming, 13-15
 properties
 Height, 17-18
 Icon, 17
 Name, 13-15
 Text, 15-16
 Width, 17-18
 resizing, 17-19

viewing

 design windows, 38
 forms, 118-119
 initial display position, 121-122
 maximized state, 120-121
 minimized state, 120-121
 normal state, 120-121
 message boxes, 58-59
 object properties, 13
 Open File dialog box, 428-429
 Picture Viewer log files, 461-463
 properties, 45-47
 Registry options, 449-450
 text
 labels, 155-156
 text boxes, 157-163
 toolbars, 41

views**views**

List View control

adding list

items, 193-196

columns, 192-193

overview, 191-192

removing all list

items, 196

removing list items, 196

Tree View control

adding nodes, 197-199

clearing all nodes, 200

overview, 197

removing

nodes, 199-200

visible controls, 20-22**W****Warning enumerator**

(MessageBoxIcon), 384

While...End While loops, 334**Width property (forms), 17-18****Window system color, 411****windows**

code windows, hiding, 12

design windows

auto-hiding, 40-41

docking, 39-40

floating, 38

showing/hiding, 38

states, 37-38

Immediate, 346-351

Properties, 12

color properties, 47-49

overview, 44

property

descriptions, 49-50

selecting objects, 45

viewing and changing

properties, 45-47

Toolbox, 12

topmost nonmodal

windows, 144

Windows Registry. See Registry**WindowsDefaultBounds value**

(StartPosition property), 122

WindowsDefaultLocation value

(StartPosition property), 122

WindowState property

(forms), 120-121

wizards, Publish

Wizard, 500-503

Word automation, 492

instantiating automation

server, 493

referencing type

libraries, 492-493

WordWrap property (Text

Box control), 160

workbooks (Excel)

creating, 489

modifying, 490-491

wrappers, 488**writable properties, 369****Write() method, 456****write-only properties, 370****WriteLine() method, 282,****350-351, 456-457****WriteOnly keyword, 370****writing to text files, 455-457****X-Y-Z****X property**

(MouseEventArgs), 92

Xor operator, 289**Y property**

(MouseEventArgs), 92

Yes enumerator

(DialogResult), 386

YesNo enumerator

(MessageBoxButtons), 383

YesNoCancel enumerator

(MessageBoxButtons), 383

zero-length strings, 346, 458