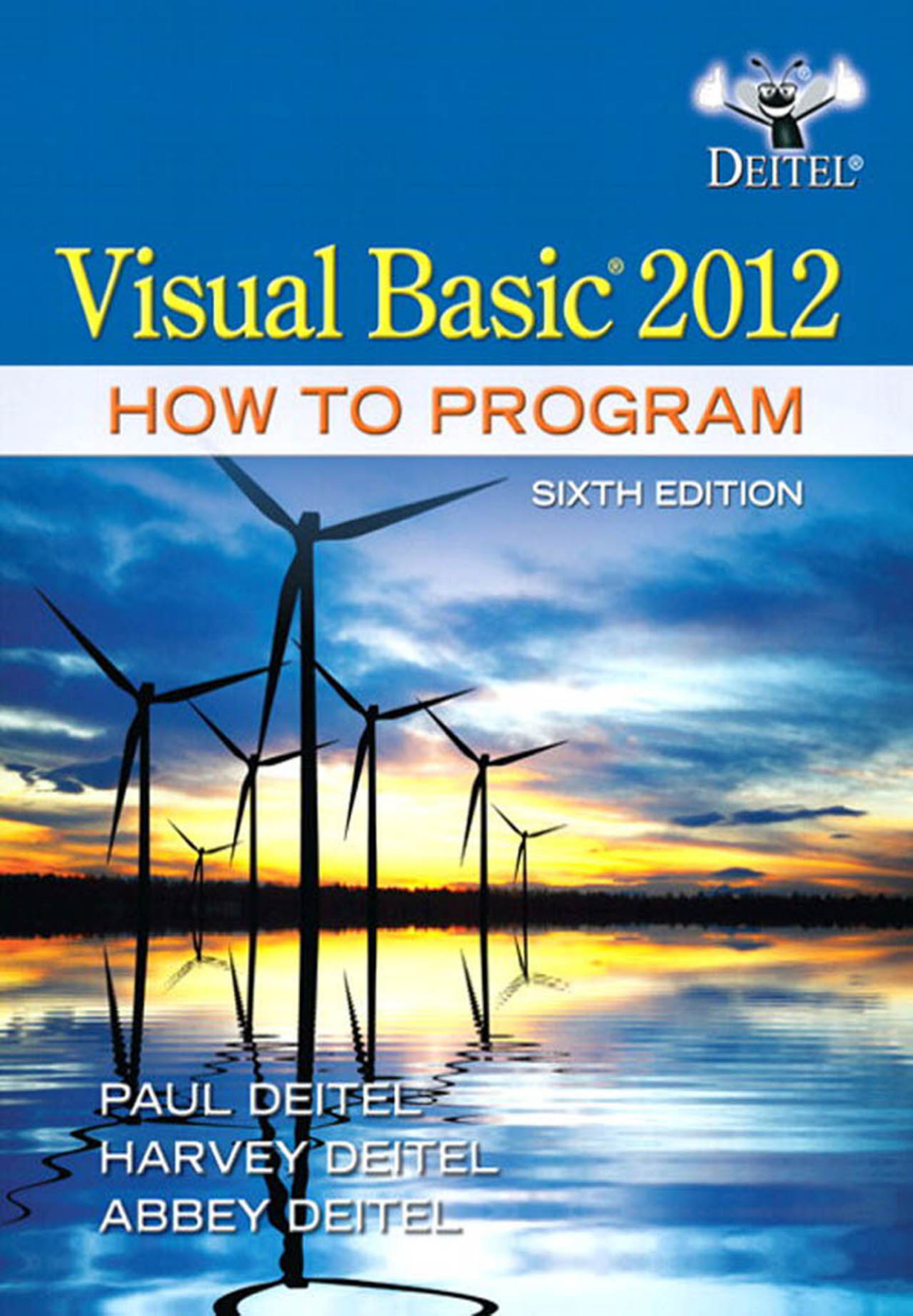
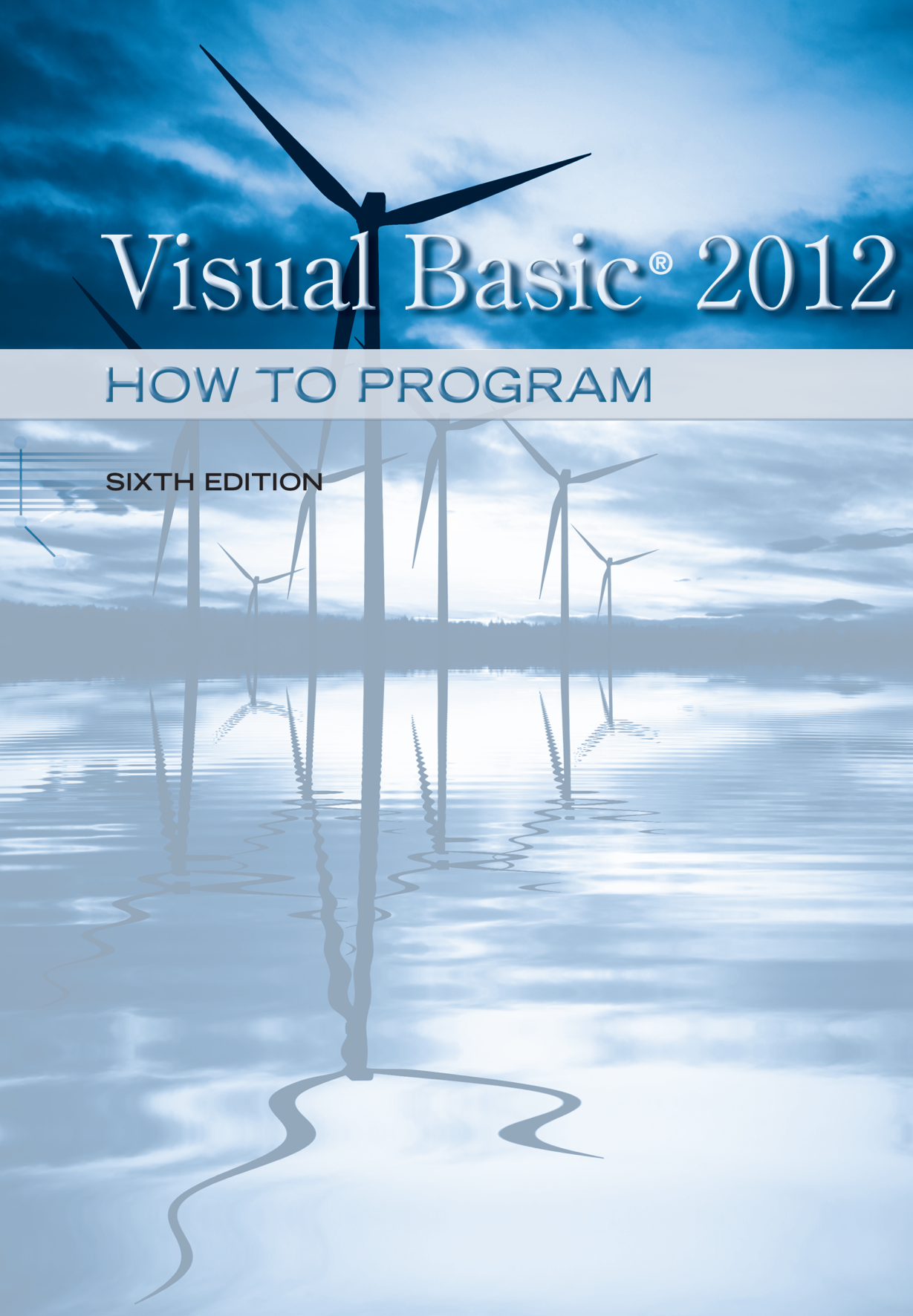

DEITEL®
Visual Basic® 2012
HOW TO PROGRAM
SIXTH EDITION
PAUL DEITEL
HARVEY DEITEL
ABBEY DEITEL


Visual Basic® 2012
HOW TO PROGRAM
SIXTH EDITION

Deitel® Series Page

How To Program Series

Android How to Program
C How to Program, 7/E
C++ How to Program, 9/E
C++ How to Program, Late Objects Version, 7/E
Java™ How to Program, 9/E
Java™ How to Program, Late Objects Version, 8/E
Internet & World Wide Web How to Program, 5/E
Visual Basic® 2012 How to Program
Visual C#® 2012 How to Program, 5/E
Visual C++® 2008 How to Program, 2/E
Small Java™ How to Program, 6/E
Small C++ How to Program, 5/E

Simply Series

Simply C++: An App-Driven Tutorial Approach
Simply Java™ Programming: An App-Driven Tutorial Approach
Simply Visual Basic® 2010, 4/E: An App-Driven Tutorial Approach
Simply Visual Basic 2010: An App-Driven Approach, 4/E
Visual Basic® 2012 How to Program
Visual Basic® 2010 How to Program
Visual C#® 2012 How to Program, 5/E
Visual C#® 2010 How to Program, 4/E

CourseSmart Web Books

www.deitel.com/books/CourseSmart/

C++ How to Program, 7/E, 8/E & 9/E
Simply C++: An App-Driven Tutorial Approach
Java™ How to Program, 7/E, 8/E & 9/E

Deitel® Developer Series

C++ for Programmers, 2/E
Android for Programmers: An App-Driven Approach
C# 2010 for Programmers, 3/E
Dive Into® iOS 6: An App-Driven Approach
iOS 6 for Programmers: An App-Driven Approach
Java™ for Programmers, 2/E
JavaScript for Programmers

LiveLessons Video Learning Products

www.deitel.com/books/LiveLessons/

Android® App Development Fundamentals
C++ Fundamentals
C# Fundamentals
iOS 6 App Development Fundamentals
Java™ Fundamentals
JavaScript Fundamentals
Visual Basic® Fundamentals

To receive updates on Deitel publications, Resource Centers, training courses, partner offers and more, please register for the free *Deitel® Buzz Online* e-mail newsletter at:

www.deitel.com/newsletter/subscribe.html

and join the Deitel communities on Twitter®

@deitel

Facebook®

facebook.com/DeitelFan

and Google+

gplus.to/deitel

To communicate with the authors, send e-mail to:

deitel@deitel.com

For information on government and corporate *Dive-Into® Series* on-site seminars offered by Deitel & Associates, Inc. worldwide, visit:

www.deitel.com/training/

or write to

deitel@deitel.com

For continuing updates on Prentice Hall/Deitel publications visit:

www.deitel.com
www.pearsonhighered.com/deitel/

Visit the Deitel Resource Centers that will help you master programming languages, software development, Android and iPhone/iPad app development, and Internet- and web-related topics:

www.deitel.com/ResourceCenters.html

Visual Basic® 2012

HOW TO PROGRAM

SIXTH EDITION

Paul Deitel
Deitel & Associates, Inc.

Abbey Deitel
Deitel & Associates, Inc.

Harvey Deitel
Deitel & Associates, Inc.

PEARSON

Boston Columbus Indianapolis New York San Franciscoa Upper Saddle River
Amsterdam Cape Town Dubai London Madrid Milan Munich Paris Montréal Toronto
Delhi Mexico City São Paulo Sydney Hong Kong Seoul Singapore Taipei Tokyo

Vice President and Editorial Director: ***Marcia J. Horton***
Executive Editor: ***Tracy Johnson***
Associate Editor: ***Carole Snyder***
Director of Marketing: ***Christy Lesko***
Marketing Manager: ***Yezan Alayan***
Marketing Assistant: ***Jon Bryant***
Director of Production: ***Erin Gregg***
Managing Editor: ***Scott Disanno***
Associate Managing Editor: ***Robert Engelhardt***
Operations Specialist: ***Lisa McDowell***
Art Director: ***Anthony Gemmellaro***
Cover Design: ***Abbey S. Deitel, Harvey M. Deitel, Anthony Gemmellaro***
Cover Photo Credit: © ***Shutterstock/Pati Photo***
Media Project Manager: ***Renata Butera***

Credits and acknowledgments borrowed from other sources and reproduced, with permission, in this textbook appear on page vi.

Library of Congress Cataloging-in-Publication Data on file.

10 9 8 7 6 5 4 3 2 1

ISBN-10: 0-13-340695-4
ISBN-13: 978-0-13-340695-5

PEARSON

To the Microsoft Visual Basic Language Team

Paul, Abbey and Harvey Deitel

Trademarks

DEITEL, the double-thumbs-up bug and DIVE INTO are registered trademarks of Deitel and Associates, Inc.

Microsoft® and Windows® are registered trademarks of the Microsoft Corporation in the U.S.A. and other countries. This book is not sponsored or endorsed by or affiliated with the Microsoft Corporation.

UNIX is a registered trademark of The Open Group.

Throughout this book, trademarks are used. Rather than put a trademark symbol in every occurrence of a trademarked name, we state that we are using the names in an editorial fashion only and to the benefit of the trademark owner, with no intention of infringement of the trademark.

Brief Contents

Chapters 16–31 are PDF documents posted online at the book's Companion Website (located at `www.pearsonhighered.com/deitel/`).

Companion Website Online Content

Chapters 16–31 and Appendix F are PDF documents posted online at the book's Companion Website (located at `www.pearsonhighered.com/deitel`).

Contents

Chapters 16–31 are PDF documents posted online at the book's Companion Website (located at `www.pearsonhighered.com/deitel/`).

Companion Website Online Content

Chapters 16–31 and Appendix F are PDF documents posted online at the book's Companion Website (located at `www.pearsonhighered.com/deitel`).

Preface

Welcome to the Visual Basic® 2012 computer programming language and the world of Microsoft® Windows® and Internet and web programming with Microsoft's .NET platform. Please read the book's back cover and inside back cover—these concisely capture the book's essence. In this Preface we provide more details.

This book is appropriate for information technology and business students in novice-level and intermediate-level Visual Basic courses. The book is also used by professional programmers.

At the heart of the book is the Deitel signature *live-code approach*—rather than using code snippets, we present concepts in the context of complete working programs followed by sample executions. Read the Before You Begin section after this Preface for instructions on setting up your computer to run the code examples. The source code is available at `www.deitel.com/books/vb2012htp` and `www.pearsonhighered.com/deitel`. Use the source code we provide to *compile and run each program* as you study it—this will help you master Visual Basic and related Microsoft technologies faster and at a deeper level.

We believe that this book and its supplements for students and instructors will give you an informative, engaging, challenging and entertaining introduction to Visual Basic. If you have questions, we're easy to reach at `deitel@deitel.com`—we'll respond promptly. For book updates, visit `www.deitel.com/books/vb2012htp`, join our social media communities on Facebook (`www.deitel.com/DeitelFan`), Twitter (`@deitel`), Google+ (`gplus.to/deitel`) and LinkedIn (`bit.ly/DeitelLinkedIn`), and subscribe to the *Deitel® Buzz Online* newsletter (`www.deitel.com/newsletter/subscribe.html`).

Visual Basic® 2012, the Visual Studio® 2012 IDE, .NET 4.5, Windows® 7 and Windows® 8

The new Visual Basic 2012 and its associated technologies motivated us to write *Visual Basic 2012 How to Program*. These are some of the key features of this new edition:

- ***Use with Windows 7, Windows 8 or both.*** The book is designed so that you can continue to use Windows 7 now and begin to evolve to Windows 8, if you like, or you can move right to Windows 8. All of the code examples in Chapters 1–19 and 23–31 were tested on *both* Windows 7 and Windows 8. The code examples for the Windows-8-specific chapters—Chapter 20 (Windows 8 UI and XAML), Chapter 21 (Windows 8 Graphics and Multimedia) and Chapter 22 (Building a Windows Phone 8 App)—were tested *only* on Windows 8.
- ***Modular multi-GUI treatment with Windows Forms, Windows 8 UI and WPF.*** The printed book features Windows Forms GUI; optional online chapters contain treatments of WPF GUI and the new Windows 8 UI. Windows 8 UI apps are

called *Windows Store apps*. In Chapter 20, you'll learn how to create and test Windows Store apps and upload them to Microsoft's Windows Store.

- ***Modular treatment of graphics and multimedia with Windows 8 and WPF.*** The book features optional online chapters on both Windows 8 Graphics and Multimedia (Chapter 21) and WPF Graphics and Multimedia (Chapter 28).
- ***Database with LINQ to Entities.*** In the previous edition of this book, we discussed LINQ (Language Integrated Query) to SQL (Microsoft's SQL Server database system). Microsoft stopped further development on LINQ to SQL in 2008 in favor of the newer and more robust LINQ to Entities and the ADO.NET Entity Framework, which we've switched to in this edition, keeping the discussion friendly for novices.
- ***SQL Server database.*** We use Microsoft's free SQL Server Express 2012 (which installs with the free Visual Studio Express 2012 for Windows Desktop) to present the fundamentals of database programming. Chapters 12–13 and online Chapters 24–25 use database and LINQ capabilities to build an address-book desktop app, a web-based guestbook app, a bookstore app and an airline reservation system app.
- ***ASP.NET 4.5.*** Microsoft's .NET server-side technology, ASP.NET, enables you to create robust, scalable web-based apps. In Chapter 13, you'll build several apps, including a web-based guestbook that uses ASP.NET and the ADO.NET Entity Framework to store data in a database and display data in a web page. The chapter also discusses the IIS Express web server for testing your web apps on your local computer.
- ***Building a Windows Phone 8 App.*** Windows Phone 8 is Microsoft's latest operating system for smartphones. It features multi-touch support for touchpads and touchscreen devices, enhanced security features and more. In online Chapter 22, you'll build a complete working Windows Phone 8 app and test it on the Windows Phone simulator; we'll discuss how to upload apps to the Windows Phone Store.
- ***Building a Windows Azure™ Cloud Computing App.*** Windows Azure is a cloud computing platform that allows you to develop, manage and distribute your apps in the cloud. Online Chapter 26 shows you how to build a Windows Azure app that can store data in the cloud.
- ***Asynchronous programming with `async` and `await`.*** Asynchronous programming is simplified in Visual Basic 2012 with the new `Async` and `Await` capabilities. We introduce asynchronous programming with `Async` and `Await` in online Chapter 23.

Object-Oriented Programming

- ***Late objects approach.*** We defer the discussion of creating custom classes until Chapter 9, but in the early chapters, we still use lots of existing objects. Chapter 10 discusses how to create powerful new classes quickly by using inheritance to "absorb" the capabilities of existing classes, and presents the crucial concepts of polymorphism, abstract classes and interfaces.
- ***Rich coverage of programming fundamentals.*** Chapters 4 and 5 present a friendly treatment of control statements and problem solving.

- ***A clear, example-driven presentation of classes, objects, inheritance, polymorphism and interfaces.***
- ***Optional case study: Using the UML to develop an object-oriented design and Visual Basic implementation of an Automated Teller Machine (ATM).*** The UML™ (Unified Modeling Language™) is the industry-standard graphical language for modeling object-oriented systems. We introduce the UML in the early chapters. Optional online Chapters 30 and 31 include an *optional* case study on object-oriented design using the UML. We design and implement the software for a simple automated teller machine. We analyze a typical *requirements document* that specifies the system to be built. We determine the *classes* needed to implement that system, the *attributes* the classes need to have, the *behaviors* the classes need to exhibit and we specify how the classes must *interact* with one another to meet the system requirements. From the design we produce a complete working Visual Basic implementation. Students often report a "light bulb moment"—the case study helps them "tie it all together" and truly understand object orientation.
- ***Three programming paradigms.*** We discuss *structured programming*, *object-oriented programming* and *generic programming*.

Interesting, Entertaining and Challenging Exercises

- Extensive self-review exercises *and* answers are included for self-study.
- Many chapters include a multiple-choice Quick Quiz.
- Each chapter concludes with a substantial set of exercises, which generally includes simple recall of important terminology and concepts, identifying the errors in code samples, writing individual program statements, writing small portions of Visual Basic classes, writing complete programs and implementing major projects. Figure 1 lists a small sampling of the book's hundreds of exercises, including selections from our *Making a Difference* exercises set, which encourage you to use computers and the Internet to research and solve significant social problems—we hope you'll approach these exercises with *your own* values, politics and beliefs.

Exercises		
Abstract Methods	Cafeteria Survey App	`DateInformation` Class
Account Information App	Calculator GUI	Diameter, Circumference and Area
Account Inheritance Hierarchy	Carbon Footprint Calculator	Digit Extraction
Airline Reservations system	Car-Pool Savings Calculator	Displaying Tabular Data
Alarm Clock GUI	Coin Tossing	`Concantenating Strings`
Arithmetic Calculator App	Computer-Assisted Instruction: Reducing Student Fatigue	Duplicate Elimination
Array Sorting App	Computerization of Health Records	Duplicate Word Removal
Average Calculator App	Concentric Circles	`Employee` Class
Baseball Database App	Credit Checker App	Enforcing Privacy with Cryptography
Blackjack Modification		
Body Mass Index Calculator		

Fig. 1 | A sampling of the book's exercises. (Part 1 of 2.)

Exercises		
Enhanced Drawing App	Polling	Table of Decimal, Octal, Hexadecimal and Binary Equivalents
Enhanced Exam Analysis App	Polymorphism and Extensibility	Table of Powers App
Evaluating Poker Hands	Present Value Calculator App	Target-Heart-Rate Calculator
Find the Smallest and Largest Values	Protected vs. Private Access	Tax Plan Alternatives; The "Fair Tax"
Gas Pump	Pyramid	Telephone-Number Word Generator
Guess the Number App	Querying an Array of Invoice Objects	Temperature Converter App
Image Flasher	Quiz Average App	Triangles of Asterisks
Image Reflector App	Radio GUI	Using the Debugger: Discount Calculator App
Inheritance Advantages	Random Lines	Using the Debugger: Factorial App
`Invoice` Class	Reading Grades into a Two-Dimensional Array	Using the Debugger: Savings Calculator App
Large-Type Displays for People with Low Vision	Retail Sales Calculator App	Using the Debugger: Sibling Survey App
Lottery Number Generator	Road Sign Test App	Vending Machine App
MDI Text Editor	Sales Commissions	Wage Calculator with Tax Calculations
Miles Per Gallon App	`Savings Account` Class	Web-Based Address Book
Modifying the Internal Data Representation of a Class	Screen Saver App	World Population Growth
Multiples	Shape Hierarchy	Writing a Grade Report to a File
Notepad GUI	Simple Calculator App	
Nutrition Information XML Document	Simple Drawing App	
Package Inheritance Hierarchy	Snake PolyLine App	
Page Hit Counter	Sorting and Ascending and Descending Order	
Parking Charges	Speech-Controlled Drawing App	
Phone-Book Web Service	`Square` Class	
Pig Latin	Student Inheritance Hierarchy	

Fig. 1 | A sampling of the book's exercises. (Part 2 of 2.)

Other Features

- ***Illustrations and figures.*** Abundant tables, line drawings, UML diagrams, programs and program outputs are included.
- ***Focus on business and personal utility examples.***
- ***Windows Forms GUI is integrated throughout the core chapters.*** The core content focuses on Windows Forms GUI apps.
- ***We use LINQ to query files, databases, XML and collections.*** The introductory LINQ to Objects chapter (Chapter 11), is intentionally simple and brief to encourage instructors to begin covering LINQ technology early. We take a deeper look, using LINQ to Entities (Chapters 12–13 and online Chapters 24–25) and LINQ to XML (online Chapters 19, 25 and 26).
- ***Integrated Using the Debugger sections and exercises*** in the core printed book. Students use the debugger to locate and fix logic errors.
- ***Strings, files and databases are covered early.***

- ***Introduction to Web app development with ASP.NET is in the core print book.***
- ***Local type inference.*** When you initialize a local variable in its declaration, you can omit the variable's type—the compiler *infers* it from the initializer value.
- ***Object initializers.*** For new objects, you can use object initializer syntax (similar to array initializer syntax) to assign values to the new object's `public` properties and `public` instance variables.
- ***We emphasize the IDE's* IntelliSense *feature*** that helps you write code faster and with fewer errors.
- ***Optional parameters.*** You can specify method parameters with default values—if a corresponding method argument is not provided in the method call, the compiler inserts the optional parameter's default value in the call.
- ***"Quick Fix" window.*** We show how to use the IDE's **Error Correction Options** window to quickly fix certain common programming errors simply by clicking the suggested fix, which is displayed in a window in the code editor.
- ***We show how to use* DataTips *and visualizers*** to view object contents in the code window during debugging.
- ***Integrated exception handling.*** We introduce exception handling early (Chapter 7, Arrays) to ensure that we do not access an array element outside the array's bounds. Chapter 9, Object-Oriented Programming: Classes and Objects, shows how to indicate an exception when a member function receives an invalid argument. We cover the complete details of exception handling in online Chapter 16, Exception Handling: A Deeper Look.
- ***Visual Basic XML capabilities.*** Extensible Markup Language (XML) is pervasive in the software-development industry, e-business and throughout the .NET platform. In optional online Chapter 19, we introduce XML syntax and programmatically manipulate the elements of an XML document using LINQ to XML. XAML is an XML vocabulary that's used to describe graphical user interfaces, graphics and multimedia. We discuss XAML in optional online Chapters 20–21 and 27–28.
- ***Web app development with ASP.NET 4.5 and ASP.NET AJAX.*** Optional online Chapter 24 extends Chapter 13's ASP.NET discussion with a case study on building a password-protected, web-based bookstore app. We also introduce in Chapter 24 ASP.NET AJAX controls and use them to add AJAX functionality to web apps to give them a look and feel similar to that of desktop apps.

Companion Website

The printed book contains the core content (Chapters 1–15) for introductory course sequences. Several optional online chapters are available for advanced courses and professionals. Figure 2 lists the chapters that are available in searchable PDF format on the book's password-protected Companion Website at:

```
www.pearsonhighered.com/deitel
```

See the inside front cover of the book for an access code.

Online chapters	
Chapter 16, Exception Handling: A Deeper Look	Chapter 26, Building a Windows Azure Cloud Computing App
Chapter 17, Strings and Characters: A Deeper Look	Chapter 27, Windows Presentation Foundation (WPF) GUI
Chapter 18, Files and Streams: A Deeper Look	Chapter 28, WPF Graphics and Multimedia
Chapter 19, XML and LINQ to XML	Chapter 29, Data Structures and Generic Collections
Chapter 20, Windows 8 UI	Chapter 30, ATM Case Study, Part 1: Object-Oriented Design with the UML
Chapter 21, Windows 8 Graphics and Multimedia	Chapter 31, ATM Case Study, Part 2: Implementing an Object-Oriented Design
Chapter 22, Windows Phone 8 Case Study	Index (The online index includes the content from the printed book and the online content. The printed book index covers only the printed material.)
Chapter 23, Introduction to Concurrency: `Async` and `Await`	
Chapter 24, Web App Development with ASP.NET: A Deeper Look	
Chapter 25, Web Services	

Fig. 2 | Optional online chapters in *Visual Basic 2012 How to Program*.

VideoNotes

The Companion Website also includes extensive *VideoNotes*—watch and listen as co-author Paul Deitel discusses key code examples in the core chapters of the book. VideoNotes allow for self-paced instruction with easy navigation, including the ability to select, play, rewind, fast-forward and stop within each video.

We've created a jump table that maps each VideoNote to the corresponding figures in the book (`www.deitel.com/books/vb2012htp/jump_table.pdf`). VideoNotes are free with the purchase of a *new* textbook. If you have a *used* book you can purchase access to the VideoNotes for this book as follows:

1. Go to `www.pearsonhighered.com/deitel/`.
2. Scroll to *Visual Basic 2012 How to Program* and click **Companion Website**.
3. Click the **Register** button.
4. On the registration page, enter your student access code found beneath the scratch-off panel on the inside front cover of this book. Do not type the dashes. You can use lower- or uppercase. The access code can be used *only once*. This subscription is valid for twelve months upon activation and is *not transferable*. If this access code on your book has already been revealed, it may no longer be valid. If this is the case, click the **Website Purchase** link and follow the instructions.
5. Once your personal Login Name and Password are confirmed, you can begin using the *Visual Basic 2012 How to Program* Companion Website.

Book Overview and Chapter Dependencies

This section discusses the book's modular organization to help instructors plan their syllabi.

Introduction to Visual Basic and Visual Studio 2012 Express

Chapter 1, Introduction to Computers, the Internet and Visual Basic, introduces computing fundamentals and Microsoft's .NET platform. If you do not need to cover these fundamentals, you should still cover the **Painter** app test-drive. The vast majority of the book's examples will run on Windows 7 and Windows 8 using *Visual Studio Express 2012 for Windows Desktop*, which we test-drive in Section 1.14. Online Chapters 20–21 can be run *only* on Windows 8 using *Visual Studio Express 2012 for Windows 8*. There are other versions of *Visual Studio Express 2012* for web development and Windows Phone development—we cover these in the corresponding chapters.

Chapter 2, Dive Into® Visual Studio Express 2012 for Windows Desktop, shows how to develop a simple GUI app that displays text and an image. We'll look at Visual Studio Express 2012 for Windows 8 in more depth in online Chapter 20.

Introduction to Visual Basic Fundamentals

The chapters in this module of the book:

- Chapter 3, Introduction to Visual Basic Programming
- Chapter 4, Introduction to Problem Solving and Control Statements
- Chapter 5, Problem Solving and Control Statements: Part 2
- Chapter 6, Methods
- Chapter 7, Arrays
- Chapter 8, Files

present Visual Basic programming fundamentals (data types, operators, control statements, methods, arrays and files). These chapters should be covered in order. Chapter 7 introduces exception handling with an example that demonstrates accessing an element outside an array's bounds.

Object-Oriented Programming

The chapters in this module of the book:

- Chapter 9, Object-Oriented Programming: Classes and Objects
- Chapter 10, Object-Oriented Programming: Inheritance and Polymorphism
- Chapter 11, Introduction to LINQ
- Chapter 16, Exception Handling: A Deeper Look
- Chapter 30, ATM Case Study, Part 1: Object-Oriented Design with the UML
- Chapter 31, ATM Case Study, Part 2: Implementing an Object-Oriented Design

discuss object-oriented programming, including classes, objects, inheritance, polymorphism, interfaces and exception handling. Chapter 11, Introduction to LINQ, introduces Microsoft's Language Integrated Query (LINQ) technology, which provides a uniform syntax for manipulating data from various data sources, such as arrays and, as you'll see in later chapters, collections, XML and databases. This chapter can be deferred, but it's required for many of the later chapters starting with Chapter 12, Databases and LINQ. Online Chapters 30–31 present an *optional* object-oriented design and implementation case study

that requires the Visual Basic and object-oriented programming concepts presented in Chapters 3–7 and 9–10.

Windows `Forms` Graphical User Interfaces (GUIs), Graphics and Multimedia
There are now three GUI technologies in Windows—Windows `Forms` (which is a legacy technology), Windows 8 UI (available *only* on Windows 8) and Windows Presentation Foundation (WPF). We surveyed instructors teaching Visual Basic and they still prefer Windows `Forms` for their classes, so Windows `Forms` GUI is integrated throughout most of the book. Chapter 14, Windows Forms GUI: A Deeper Look, covers additional Windows `Forms` GUI controls and Chapter 15, Graphics and Multimedia, introduces graphics and multimedia. For those who wish to present or study Microsoft's more recent GUI, graphics and multimedia technologies, we provide online introductions to Windows 8 UI, graphics and multimedia (online Chapters 20–21) and WPF GUI, graphics and multimedia (online Chapters 27–28).

Strings and Files
We introduce `Strings` beginning in Chapter 3 and use them throughout the book. We introduce files beginning in Chapter 8. Online Chapter 17, Strings and Characters: A Deeper Look, investigates `Strings` in more depth, and online Chapter 18, Files and Streams: A Deeper Look, discusses files in more depth.

Databases and an Introduction to Web App Development
Chapter 12, Databases and LINQ, introduces database app development using the ADO.NET Entity Framework and LINQ to Entities. The chapter's examples require Visual Basic, object-oriented programming and Windows `Forms` concepts presented in Chapters 3–11. The final example in Chapter 13, Web App Development with ASP.NET requires the LINQ and database techniques presented in Chapter 12.

Extensible Markup Language (XML)
Online Chapter 19, XML and LINQ to XML, introduces XML, which is used in several later chapters. The first few sections of this chapter are required to understand the XAML markup that's used to build Windows 8 GUI, graphics and multimedia apps (Chapters 20–21), Windows Phone 8 apps (Chapter 22) and WPF GUI, graphics and multimedia apps (Chapters 27–28). The remainder of the chapter discusses LINQ to XML, which allows you to manipulate XML using LINQ syntax. These capabilities are used in Chapters 25 and 26.

Windows 8 UI, Graphics and Multimedia; Windows Phone
The online chapters in this module of the book:

- Chapter 20, Windows 8 UI
- Chapter 21, Windows 8 Graphics and Multimedia
- Chapter 22, Windows Phone 8 Case Study

present Windows 8 UI, graphics and multimedia, and Windows Phone 8 app development. These chapters can be used *only* on computers running Windows 8—they depend on event-handling concepts that are presented throughout the early chapters and the introduction to XML at the beginning of online Chapter 19 (see Section 19.1 for details). Developing a Windows Phone 8 app is similar to developing a Windows 8 UI app.

Asynchronous Programming
Online Chapter 23, Introduction to Concurrency: `Async` and `Await`, demonstrates .NET's and Visual Basic's new simplified asynchronous programming capabilities. These are commonly used in Web app and Web service development among many other uses.

Web App Development and Web Smervices
The chapters in this module of the book:

- Chapter 24, Web App Development with ASP.NET: A Deeper Look
- Chapter 25, Web Services
- Chapter 26, Building a Windows Azure™ Cloud Computing App

continue our discussion of Web app development from Chapter 13 and introduce web services, including a case study on cloud computing with Windows Azure. Online Chapters 25 and 26 depend on the LINQ to XML discussion in Chapter 19.

Windows Presentation Foundation (WPF) GUI, Graphics and Multimedia
The chapters in this module of the book

- Chapter 27, Windows Presentation Foundation (WPF) GUI
- Chapter 28, WPF Graphics and Multimedia

discuss Windows Presentation Foundation GUI, graphics and multimedia. These chapters can be used on computers running Windows 7 or Windows 8 and depend on event-handling concepts that are presented throughout the early chapters and the introduction to XML at the beginning of online Chapter 19.

Teaching Approach

Visual Basic 2012 How to Program contains a rich collection of examples. We concentrate on building good software and stress program clarity.

Live-Code Approach. The book is loaded with "live-code" examples. Most new concepts are presented in the context of complete working Visual Basic apps, followed by one or more executions showing program inputs and outputs. In the few cases where we show a code snippet, to ensure correctness we first tested it in a complete working program then copied the code from the program and pasted it into the book.

Syntax Shading. For readability, we syntax shade the code, similar to the way most integrated-development environments and code editors syntax color code. Our syntax-shading conventions are:

```
comments appear like this
keywords appear like this
constants and literal values appear like this
all other code appears in black
```

Code Highlighting. We place light blue rectangles around each program's key code.

Using Fonts for Emphasis. We place the key terms and the index's page reference for each defining occurrence in **bold blue** text for easy reference. We emphasize on-screen compo-

nents in the **bold Helvetica** font (for example, the **File** menu) and Visual Basic program text in the `Lucida` font (for example, `Dim count As Integer = 5`).

Objectives. The opening quotes are followed by a list of chapter objectives.

Illustrations/Figures. Abundant tables, line drawings, UML diagrams, programs and program outputs are included.

Programming Tips. We include programming tips to help you focus on important aspects of program development. These tips and practices represent the best we've gleaned from a combined seven decades of programming and teaching experience.

Good Programming Practice

The Good Programming Practices *call attention to techniques that will help you produce programs that are clearer, more understandable and more maintainable.*

Common Programming Error

Pointing out these Common Programming Errors *reduces the likelihood that you'll make them.*

Error-Prevention Tip

These tips contain suggestions for exposing and removing bugs from your programs; many describe aspects of Visual Basic that prevent bugs from getting into programs.

Performance Tip

These tips highlight opportunities for making your programs run faster or minimizing the amount of memory that they occupy.

Portability Tip

The Portability Tips *help you write code that will run on a variety of platforms.*

Software Engineering Observation

The Software Engineering Observations *highlight architectural and design issues that affect the construction of software systems, especially large-scale systems.*

Look-and-Feel Observation

These observations help you design attractive, user-friendly graphical user interfaces that conform to industry norms.

Summary Bullets. We present a section-by-section, bullet-list summary of each chapter.

Terminology. We include an alphabetized list of the important terms defined in each chapter with the page number of each term's defining occurrence for easy reference.

Self-Review Exercises and Answers. Extensive self-review exercises *and* answers are included for self-study.

Exercises. Each chapter concludes with additional exercises including:

- simple recall of important terminology and concepts
- What's wrong with this code?

- What does this code do?
- Using the Debugger
- writing individual statements and small portions of methods and classes
- writing complete methods, classes and programs
- major projects.

Check out our Programming Projects Resource Center for lots of additional exercise and project possibilities (`www.deitel.com/ProgrammingProjects/`).

Index. We've included an extensive index for reference. Defining occurrences of key terms in the index are highlighted with a **bold blue** page number.

Software Included with *Visual Basic 2012 How to Program*

This book includes the Microsoft® Visual Studio® Express 2012 for Windows Desktop DVD, which runs on Windows 7 and 8. See the Before You Begin section that follows this preface for information on downloading the other Visual Studio Express 2012 Editions that we use in this book.

Instructor Supplements

The following supplements are available to ***qualified instructors only*** through Pearson Education's Instructor Resource Center (`www.pearsonhighered.com/irc`):

- ***Solutions Manual*** contains solutions to *most* of the end-of-chapter exercises. **Please do not write to us requesting access to the Pearson Instructor's Resource Center. Access is restricted to college instructors teaching from the book. Instructors may obtain access only through their Pearson representatives.** If you're not a registered faculty member, contact your Pearson representative or visit `www.pearsonhighered.com/educator/replocator/`. Exercise Solutions are *not* provided for "project" exercises. Check out our Programming Projects Resource Center for lots of additional exercise and project possibilities:

```
www.deitel.com/ProgrammingProjects
```

- ***Test Item File*** of multiple-choice questions (approximately two per book section)
- ***Customizable PowerPoint® slides*** containing all the code and figures in the text, plus bulleted items that summarize the key points in the text.

Microsoft DreamSpark™

Professional Developer and Designer Tools for Students

Microsoft provides many of its professional developer tools to students for free via a program called DreamSpark (`www.dreamspark.com`). See the website for details on verifying your student status so you take advantage of this program.

Acknowledgments

We'd like to thank Barbara Deitel of Deitel & Associates, Inc. for long hours devoted to this project. She painstakingly researched the new capabilities of Visual Basic 2012, .NET 4.5, Windows 8, Windows Phone 8, Windows Azure and other key topics.

We're fortunate to have worked with the dedicated team of publishing professionals at Pearson Higher Education. We appreciate the guidance, wisdom and energy of Tracy Johnson, Executive Editor, Computer Science. Carole Snyder did an extraordinary job recruiting the book's reviewers and managing the review process. Bob Engelhardt did a wonderful job bringing the book to publication.

Reviewers

We wish to acknowledge the efforts of our reviewers. The book was scrutinized by academics teaching Visual Basic courses and industry experts. They provided countless suggestions for improving the presentation. Any remaining flaws in the book are our own.

Sixth edition reviewers: Wu He (Old Dominion University), Ken Tucker (Microsoft MVP and Software Developer, Sea World), José Antonio González Seco (Parliament of Andalusia) and Jim Wooley (Slalom Consulting, Microsoft Visual Basic MVP, Author of LINQ in Action).

Other recent edition reviewers: Douglas B. Bock (MCSD.NET, Southern Illinois University Edwardsville), Dan Crevier (Microsoft), Amit K. Ghosh (University of Texas at El Paso), Marcelo Guerra Hahn (Microsoft), Kim Hamilton (Software Design Engineer at Microsoft and co-author of *Learning UML 2.0*), Huanhui Hu (Microsoft Corporation), Vitek Karas (Microsoft), Narges Kasiri (Oklahoma State University), James Edward Keysor (Florida Institute of Technology), Helena Kotas (Microsoft), Charles Liu (University of Texas at San Antonio), Chris Lovett (Software Architect at Microsoft), Bashar Lulu (INETA Country Leader, Arabian Gulf), John McIlhinney (Spatial Intelligence; Microsoft MVP 2008 Visual Developer, Visual Basic), Ged Mead (Microsoft Visual Basic MVP, DevCity.net), Anand Mukundan (Architect, Polaris Software Lab Ltd.), Dr. Hamid R. Nemati (The University of North Carolina at Greensboro), Timothy Ng (Microsoft), Akira Onishi (Microsoft), Jeffrey P. Scott (Blackhawk Technical College), Joe Stagner (Senior Program Manager, Developer Tools & Platforms), Erick Thompson (Microsoft) and Jesús Ubaldo Quevedo-Torrero (University of Wisconsin–Parkside, Department of Computer Science)

As you read the book, we'd sincerely appreciate your comments, criticisms and suggestions for improving the text. Please address all correspondence to:

```
deitel@deitel.com
```

We'll respond promptly. We really enjoyed writing this book—we hope you enjoy reading it!

Paul Deitel
Harvey Deitel

About the Authors

Paul Deitel, CEO and Chief Technical Officer of Deitel & Associates, Inc., is a graduate of MIT, where he studied Information Technology. Through Deitel & Associates, Inc., he has delivered hundreds of programming courses to industry clients, including Cisco, IBM, Siemens, Sun Microsystems, Dell, Fidelity, NASA at the Kennedy Space Center, the National Severe Storm Laboratory, White Sands Missile Range, Rogue Wave Software, Boeing, SunGard Higher Education, Nortel Networks, Puma, iRobot, Invensys and many

more. He and his co-author, Dr. Harvey M. Deitel, are the world's best-selling programming-language textbook/professional book/video authors.

Paul was named as a Microsoft® Most Valuable Professional (MVP) for C# in 2012. According to Microsoft, "the Microsoft MVP Award is an annual award that recognizes exceptional technology community leaders worldwide who actively share their high quality, real world expertise with users and Microsoft."

2012 C# MVP

Dr. Harvey Deitel, Chairman and Chief Strategy Officer of Deitel & Associates, Inc., has over 50 years of experience in computing. Dr. Deitel earned B.S. and M.S. degrees in Electrical Engineering from MIT and a Ph.D. in Mathematics from Boston University. He has extensive college teaching experience, including earning tenure and serving as the Chairman of the Computer Science Department at Boston College before founding Deitel & Associates, Inc., in 1991 with his son, Paul Deitel. The Deitels' publications have earned international recognition, with translations published in Chinese, Korean, Japanese, German, Russian, Spanish, French, Polish, Italian, Portuguese, Greek, Urdu and Turkish. Dr. Deitel has delivered hundreds of programming courses to corporate, academic, government and military clients.

Deitel® Dive-Into® Series Programming Languages Training

Deitel & Associates, Inc., founded by Paul Deitel and Harvey Deitel, is an internationally recognized authoring and corporate training organization, specializing in computer programming languages, object technology, mobile app development and Internet and web software technology. The company's training clients include many of the world's largest companies, government agencies, branches of the military, and academic institutions. The company offers instructor-led training courses delivered at client sites worldwide on major programming languages and platforms, including Visual Basic®, Visual C#®, Visual C++®, C++, C, Java™, XML®, Python®, object technology, Internet and web programming, Android app development, Objective-C and iPhone app development and a growing list of additional programming and software development courses.

Through its 37-year publishing partnership with Prentice Hall/Pearson, Deitel & Associates, Inc., publishes leading-edge programming college textbooks, professional books and *LiveLessons* video courses. Deitel & Associates, Inc. and the authors can be reached at:

```
deitel@deitel.com
```

To learn more about Deitel's *Dive-Into® Series* Corporate Training curriculum, visit:

```
www.deitel.com/training
```

To request a proposal for worldwide on-site, instructor-led training at your organization, e-mail `deitel@deitel.com`.

Individuals wishing to purchase Deitel books and *LiveLessons* video training can do so through `www.deitel.com`. Bulk orders by corporations, the government, the military and academic institutions should be placed directly with Pearson. For more information, visit

```
www.informit.com/store/sales.aspx
```

This page intentionally left blank

Before You Begin

This section contains information you should review before using this book and instructions to ensure that your computer is set up properly for use with this book.

Font and Naming Conventions

We use fonts to distinguish between features, such as menu names, menu items, and other elements that appear in the program-development environment. Our convention is to emphasize IDE features in a sans-serif bold **Helvetica** font (for example, **Properties** window) and to emphasize program text in a sans-serif `Lucida` font (for example, `bool x = true`).

Software

This textbook uses the following software:

- Microsoft Visual Studio Express 2012 for Windows Desktop
- Microsoft Visual Studio Express 2012 for Web (Chapters 13 and 24–26)
- Microsoft Visual Studio Express 2012 for Windows 8 (Chapters 20–21)
- Microsoft Visual Studio Express 2012 for Windows Phone (Chapter 22)

Each is available free for download at `www.microsoft.com/express`. The Express Editions are fully functional, and there's no time limit for using the software.

Hardware and Software Requirements for the Visual Studio 2012 Express Editions

To install and run the Visual Studio 2012 Express Editions, ensure that your system meets the minimum requirements specified at:

```
www.microsoft.com/visualstudio/eng/products/compatibility
```

Microsoft Visual Studio Express 2012 for Windows 8 works *only* on Windows 8.

Viewing File Extensions

Several screenshots in *Visual Basic 2012 How to Program* display file names with file-name extensions (e.g., `.txt`, `.cs` or `.png`). Your system's settings may need to be adjusted to display file-name extensions. Follow these steps to configure your Windows 7 computer:

1. In the **Start** menu, select **All Programs**, then **Accessories**, then **Windows Explorer**.
2. Press *Alt* to display the menu bar, then select **Folder Options...** from **Windows Explorer**'s **Tools** menu.
3. In the dialog that appears, select the **View** tab.
4. In the **Advanced settings:** pane, uncheck the box to the left of the text **Hide extensions for known file types**. [*Note*: If this item is already unchecked, no action needs to be taken.]
5. Click OK to apply the setting and close the dialog.

Follow these steps to configure your Windows 8 computer:

1. On the **Start** screen, click the **Desktop** tile to switch to the desktop.
2. On the task bar, click the **File Explorer** icon to open the **File Explorer**.
3. Click the **View** tab, then ensure that the **File name extensions** checkbox is checked.

Obtaining the Code Examples

The examples for *Visual Basic 2012 How to Program* are available for download at

```
www.deitel.com/books/vb2012htp/
```

If you're not already registered at our website, go to `www.deitel.com` and click the **Register** link below our logo in the upper-left corner of the page. Fill in your information. There's no charge to register, and we do not share your information with anyone. We send you only account-management e-mails unless you register separately for our free e-mail newsletter at `www.deitel.com/newsletter/subscribe.html`. *You must enter a valid e-mail address.* After registering, you'll receive a confirmation e-mail with your verification code. Click the link in the confirmation email to go to `www.deitel.com` and sign in.

Next, go to `www.deitel.com/books/vb2012htp/`. Click the **Examples** link to download the ZIP archive file to your computer. Write down the location where you save the file—most browsers will save the file into your `Downloads` folder.

Throughout the book, steps that require you to access our example code on your computer assume that you've extracted the examples from the ZIP file and placed them at `C:\Examples`. You can extract them anywhere you like, but if you choose a different location, you'll need to update our steps accordingly. You can extract the ZIP archive file's contents using tools such as WinZip (`www.winzip.com`), 7-zip (`www.7-zip.org`) or the built-in capabilities of **Windows Explorer** on Window 7 or **File Explorer** on Windows 8.

Visual Studio Theme

Visual Studio 2012 has a **Dark** theme (the default) and a **Light** theme. The screen captures shown in this book use the **Light** theme, which is more readable in print. If you'd like to switch to the **Light** theme, in the **TOOLS** menu, select **Options...** to display the **Options** dialog. In the left column, select **Environment**, then select **Light** under **Color theme**. Keep the **Options** dialog open for the next step.

Displaying Line Numbers and Configuring Tabs

Next, you'll change the settings so that your code matches that of this book. To have the IDE display line numbers, expand the **Text Editor** node in the left pane then select **All Languages**. On the right, check the **Line numbers** checkbox. Next, expand the Visual Basic node in the left pane and select **Tabs**. Make sure that the option **Insert spaces** is selected. Enter **3** for both the **Tab size** and **Indent size** fields. Any new code you add will now use three spaces for each level of indentation. Click **OK** to save your settings.

Miscellaneous Notes

- Some people like to change the workspace layout in the development tools. You can return the tools to their default layouts by selecting **Window > Reset Window Layout**.

- Many of the menu items we use in the book have corresponding icons shown with each menu item in the menus. Many of the icons also appear on one of the toolbars at the top of the development environment. As you become familiar with these icons, you can use the toolbars to help speed up your development time. Similarly, many of the menu items have keyboard shortcuts (also shown with each menu item in the menus) for accessing commands quickly.

You are now ready to begin your Visual Basic studies with *Visual Basic 2012 How to Program*. We hope you enjoy the book!

This page intentionally left blank

Introduction to Computers, the Internet and Visual Basic

1

The chief merit of language is clearness.
——Galen

Our life is frittered away with detail. . . . Simplify, simplify.
—Henry David Thoreau

Man is still the most extraordinary computer of all.
—John F. Kennedy

Objectives

In this chapter you'll learn:

- Basic hardware, software and data concepts.
- The different types of programming languages.
- The history of the Visual Basic programming language and the Windows operating system.
- What cloud computing with Windows Azure™ is.
- Basics of object technology.
- The history of the Internet and the World Wide Web.
- The parts that Windows, .NET, Visual Studio 2012 and Visual Basic 2012 play in the Visual Basic ecosystem.
- To test-drive a Visual Basic 2012 drawing app.

Outline

1.1 Introduction

Welcome to Visual Basic 2012 which, from this point forward, we'll refer to simply as Visual Basic. Visual Basic is a powerful computer programming language that's appropriate for building substantial information systems. This book explains how to develop software with Visual Basic.

You're already familiar with the powerful tasks computers perform. Using this textbook, you'll write instructions commanding computers to perform those kinds of tasks and you'll prepare yourself to address new challenges.

Computers process *data* under the control of sequences of instructions called **computer programs**. These programs guide the computer through *actions* specified by people called **computer programmers**. The programs that run on a computer are referred to as **software**. In this book, you'll learn *object-oriented programming*—today's key programming methodology that's enhancing programmer productivity and reducing software development costs. You'll create many *software objects* that model both abstract and real-world *things*. And you'll build Visual Basic apps (applications) for a variety of environments including the *desktop* and *tablets*—and new to this edition of the book—"the *cloud*" and even *mobile devices* like *smartphones*.

1.2 Hardware and Moore's Law

A computer consists of various devices referred to as **hardware**, such as the keyboard, screen, mouse, hard disks, memory, DVD drives, printer and processing units. Every year or two, the capacities of computer hardware have approximately *doubled* inexpensively. This remarkable trend often is called **Moore's Law**, named for the person who identified it, Gordon Moore, co-founder of Intel—the leading manufacturer of the processors in to-

day's computers and **embedded systems**, such as smartphones, appliances, game controllers, cable set-top boxes and automobiles.

Moore's Law and *related* observations apply especially to

- the amount of *memory* that computers have for running programs and processing data
- the amount of *secondary storage* (such as hard disk storage) they have to hold programs and data over longer periods of time
- their *processor speeds*—the speeds at which computers **execute** their programs (i.e., do their work).

Similar growth has occurred in the communications field, in which costs have plummeted as enormous demand for communications *bandwidth* (i.e., information-carrying capacity) has attracted intense competition. We know of no other fields in which technology improves so quickly and costs fall so rapidly. Such phenomenal improvement is truly fostering the *Information Revolution* and creating significant career opportunities.

1.3 Data Hierarchy

Data items processed by computers form a **data hierarchy** that becomes larger and more complex in structure as we progress from the simplest data items (called "bits") to richer data items, such as characters, fields, and so on. Figure 1.1 illustrates a portion of the data hierarchy.

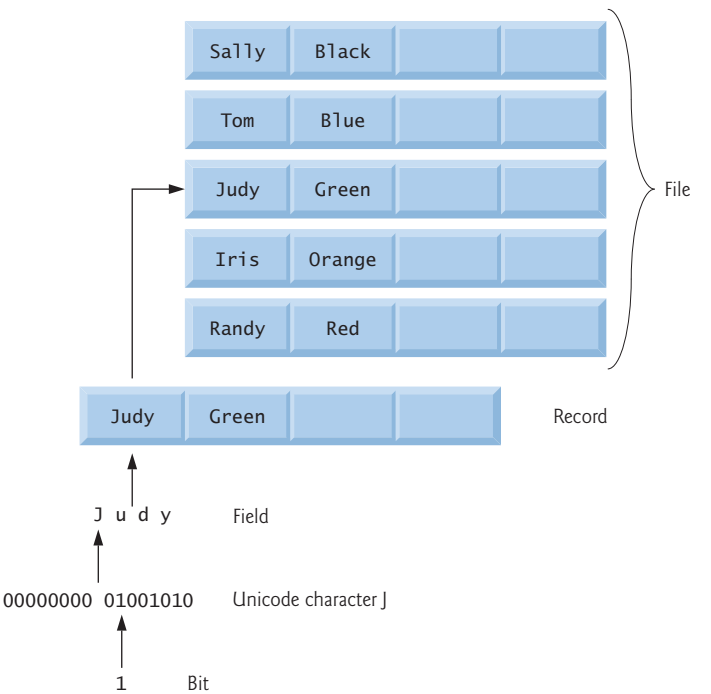


Fig. 1.1 | Data hierarchy.

Bits

The smallest data item in a computer can assume the value 0 or the value 1. Such a data item is called a **bit** (short for "binary digit"—a digit that can assume either of *two* values). It's remarkable that the impressive functions performed by computers involve only the simplest manipulations of 0s and 1s—*examining a bit's value*, *setting a bit's value* and *reversing a bit's value* (from 1 to 0 or from 0 to 1). We discuss binary numbers in more detail in Appendix C, Number Systems.

Characters

It's tedious for people to work with data in the low-level form of bits. Instead, we prefer to work with *decimal digits* (0–9), *uppercase letters* (A–Z), *lowercase letters* (a–z), and *special symbols* (e.g., $, @, %, &, *, (,), –, +, ", :, ? and /). Digits, letters and special symbols are known as **characters**. The computer's **character set** is the set of all the characters used to write programs and represent data items on that device. Computers process only 1s and 0s, so every character is represented as a pattern of 1s and 0s. The **Unicode** character set contains characters for many of the world's languages. Visual Basic supports several character sets, including 16-bit Unicode® characters that are composed of two **bytes**—each byte is composed of eight bits. See Appendix D for more information on the **ASCII (American Standard Code for Information Interchange)** character set—the popular *subset* of Unicode that represents uppercase and lowercase letters in the English alphabet, digits and some common *special characters*.

Fields

Just as characters are composed of bits, **fields** are composed of characters or bytes. A field is a group of characters or bytes that conveys meaning. For example, a field consisting of uppercase and lowercase letters could be used to represent a person's name, and a field consisting of decimal digits could represent a person's age.

Records

Several related fields can be used to compose a **record**. In a payroll system, for example, the record for an employee might consist of the following fields (possible types for these fields are shown in parentheses):

- Employee identification number (a whole number)
- Name (a string of characters)
- Address (a string of characters)
- Hourly pay rate (a number with a decimal point)
- Year-to-date earnings (a number with a decimal point)
- Amount of taxes withheld (a number with a decimal point)

Thus, a record is a group of related fields. In the preceding example, all the fields belong to the *same* employee. A company might have many employees and a payroll record for each one.

Files

A **file** is a group of related records. [*Note:* More generally, a file contains arbitrary data in arbitrary formats. In some operating systems, a file is viewed simply as a *sequence of bytes*—

any organization of the bytes in a file, such as organizing the data into records, is a view created by the programmer.] It's not unusual for an organization to have thousands or even millions of files, some containing billions or even trillions of characters of information. You'll work with files in Chapter 8.

Database

A **database** is a collection of data that's organized for easy access and manipulation. The most popular database model is the *relational database* in which data is stored in simple *tables*. A table includes *records* composed of *fields*. For example, a table of students might include first name, last name, major, year, student ID number and grade point average fields. The data for each student is a record, and the individual pieces of information in each record are the fields. You can *search*, *sort* and otherwise manipulate the data based on its relationship to multiple tables or databases. For example, a university might use data from the student database in combination with data from databases of courses, on-campus housing, meal plans, etc. We discuss databases in Chapters 12–13.

Big Data

The amount of data being produced worldwide is enormous and growing explosively. Figure 1.2 shows various common byte measurements. According to IBM, approximately 2.5 quintillion bytes (2.5 *exabytes*) of data are created daily and 90% of the world's data was created in just the past two years![1] According to an IDC study, approximately 1.8 *zettabytes* (equal to 1.8 trillion gigabytes) of data was used worldwide in 2011.[2]

Unit	Consists of	Which is approximately
1 kilobyte (KB)	1024 bytes	10^3 (1024 bytes, exactly)
1 megabyte (MB)	1024 kilobytes	10^6 (1,000,000 bytes)
1 gigabyte (GB)	1024 megabytes	10^9 (1,000,000,000 bytes)
1 terabyte (TB)	1024 gigabytes	10^{12} (1,000,000,000,000 bytes)
1 petabyte (PB)	1024 terabytes	10^{15} (1,000,000,000,000,000 bytes)
1 exabyte (EB)	1024 petabytes	10^{18} (1,000,000,000,000,000,000 bytes)
1 zettabyte (ZB)	1024 exabytes	10^{21} (1,000,000,000,000,000,000,000 bytes)

Fig. 1.2 | Byte measurements.

1.4 Computer Organization

Regardless of differences in *physical* appearance, computers can be envisioned as divided into various **logical units** or sections.

Input Unit

This "receiving" section obtains information (data and computer programs) from **input devices** and places it at the disposal of the other units for processing. Most information is

1. www-01.ibm.com/software/data/bigdata/.
2. www.emc.com/collateral/about/news/idc-emc-digital-universe-2011-infographic.pdf.

entered into computers through keyboards, touch screens and mouse devices. Other forms of input include receiving voice commands, scanning images and barcodes, reading from secondary storage devices (such as hard drives, DVD drives, Blu-ray Disc™ drives and USB flash drives—also called "thumb drives" or "memory sticks"), receiving video from a webcam or smartphone and having your computer receive information from the Internet (such as when you download videos from YouTube or e-books from Amazon). Newer forms of input include position data from GPS devices, and motion and orientation information from accelerometers in smartphones or game controllers (such as Microsoft® Kinect™, Nintendo's Wii™ Remote and Sony's PlayStation® Move).

Output Unit

This "shipping" section takes information that the computer has processed and places it on various **output devices** to make it available for use outside the computer. Most information that's output from computers today is displayed on screens; printed on paper ("going green" discourages this); played as audio or video on PCs and media players (such as Apple's iPods) and giant screens in sports stadiums; transmitted over the Internet or used to control other devices, such as robots, 3D printers and "intelligent" appliances.

Memory Unit

This rapid-access, relatively low-capacity "warehouse" section retains information that's entered through the input unit, making it immediately available for processing when needed. The memory unit also retains processed information until it can be placed on output devices by the output unit. Information in the memory unit is *volatile*—it's typically *lost* when the computer's power is turned off. The memory unit is often called either **memory** or **primary memory**—on desktop and notebook computers it commonly contains as much as 16 GB (gigabytes).

Arithmetic and Logic Unit (ALU)

This "manufacturing" section performs *calculations*, such as addition, subtraction, multiplication and division. It also contains the *decision* mechanisms that allow the computer, for example, to *compare* two items from the memory unit to determine whether they're equal. In today's systems, the ALU is usually implemented as part of the next logical unit, the CPU.

Central Processing Unit (CPU)

This "administrative" section *supervises* the operation of the other sections. The CPU tells the input unit when information should be read into the memory unit, tells the ALU when information from the memory unit should be used in calculations and tells the output unit when to send information from the memory unit to certain output devices. Many of today's computers have multiple CPUs and, hence, can perform many operations simultaneously. A **multi-core processor** implements multiple CPUs on a single "microchip"—a *dual-core processor* has *two* CPUs and a *quad-core processor* has *four* CPUs. Many of today's desktop computers have quad-core processors that can execute billions of instructions per second. In this book you'll use Microsoft's new `Async` technology to write programs that can keep CPUs running in parallel to get your computing tasks done faster.

Secondary Storage Unit

This is the long-term, high-capacity "warehousing" section. Programs or data not actively being used by the other units normally are placed on secondary storage devices (such as

your *hard drive*) until they're again needed, possibly hours, days, months or even years later. Information on secondary storage devices is *persistent*—it's *preserved* even when the computer's power is turned off. Secondary storage data takes much longer to access than information in primary memory, but the cost per unit of secondary storage is much less than that of primary memory. Examples of secondary storage devices include CD drives, DVD drives and flash drives, some of which can hold up to 768 GB. Typical hard drives on desktop and notebook computers can hold up to 2 TB. New to this edition, you'll see that storage in the cloud can be viewed as additional secondary storage accessible by your Visual Basic apps.

1.5 Machine Languages, Assembly Languages and High-Level Languages

Programmers write instructions in various programming languages (such as Visual Basic), some directly understandable by computers and others requiring intermediate *translation* steps.

Machine Languages

Any computer can *directly* understand *only* its own **machine language**, defined by its hardware architecture. Machine languages generally consist of numbers, ultimately reduced to 1s and 0s. Such languages are cumbersome for humans, who prefer meaningful words like "add" and "subtract" to indicate the operations to be performed, so the machine language numeric versions of these instructions are referred to as **code**. The term "code" has become more broadly used and now refers to the program instructions in *all* levels of programming languages.

Assembly Languages and Assemblers

Machine language was simply too slow and tedious for programmers to work with. Instead, they began using English-like *abbreviations* to represent elementary operations. These abbreviations form the basis of **assembly languages**. *Translator programs* called **assemblers** convert assembly-language code to machine language code quickly. Although assembly-language code is clearer to humans, it's incomprehensible to computers until translated to machine language code.

High-Level Languages, Compilers and Interpreters

To speed the programming process even further, **high-level languages** were developed in which single statements could be written to accomplish substantial tasks. High-level languages, such as Visual Basic, C#, C++, C, Objective-C and Java, allow you to write instructions that look almost like everyday English and contain commonly used mathematical expressions. Translator programs called **compilers** convert high-level language code into machine language code.

The process of compiling a large high-level language program into machine language can take a considerable amount of computer time. **Interpreter** programs were developed to execute high-level language programs directly (without the need for compilation), although more slowly than compiled programs.

1.6 Object Technology

Visual Basic is an object-oriented programming language. In this section we'll introduce the basics of object technology.

Building software quickly, correctly and economically remains an elusive goal at a time when demands for new and more powerful software are soaring. **Objects**, or more precisely the *classes* objects come from, are essentially *reusable* software components. There are date objects, time objects, audio objects, video objects, automobile objects, people objects, etc. Almost any *noun* can be reasonably represented as a software object in terms of *attributes* (such as name, color and size) and *behaviors* (such as calculating, moving and communicating). Software developers have discovered that using a modular, object-oriented design and implementation approach can make software-development groups much more productive than was possible with earlier techniques—object-oriented programs are often easier to understand, correct and modify.

The Automobile as an Object

Let's begin with a simple analogy. Suppose you want to *drive a car and make it go faster by pressing its accelerator pedal*. What must happen before you can do this? Well, before you can drive a car, someone has to *design* it. A car typically begins as engineering drawings, similar to the *blueprints* that describe the design of a house. These drawings include the design for an accelerator pedal. The pedal *hides* from the driver the complex mechanisms that actually make the car go faster, just as the brake pedal hides the mechanisms that slow the car, and the steering wheel hides the mechanisms that turn the car. This enables people with little or no knowledge of how engines, braking and steering mechanisms work to drive a car easily.

Before you can drive a car, it must be *built* from the engineering drawings that describe it. A completed car has an *actual* accelerator pedal to make the car go faster, but even that's not enough—the car won't accelerate on its own (we hope), so the driver must *press* the pedal to accelerate the car.

Methods and Classes

Let's use our car example to introduce some key object-oriented programming concepts. Performing a task in a program requires a **method**. The method houses the program statements that actually perform the task. It *hides* these statements from its user, just as a car's accelerator pedal hides from the driver the mechanisms of making the car go faster. In object-oriented programming languages, we create a program unit called a **class** to house the set of methods that perform the class's tasks. For example, a class that represents a bank account might contain one method to *deposit* money to an account, another to *withdraw* money from an account and a third to *inquire* what the account's current balance is. A class that represents a car might contain methods for accelerating, breaking and turning. A class is similar in concept to a car's engineering drawings, which house the design of an accelerator pedal, steering wheel, and so on.

Making Objects from Classes

Just as someone has to *build a car* from its engineering drawings before you can actually drive a car, you must *build an object* from a class before a program can perform the tasks that the class's methods define. The process of doing this is called *instantiation*. An object is then referred to as an **instance** of its class.

Reuse

Just as a car's engineering drawings can be *reused* many times to build many cars, you can *reuse* a class many times to build many objects. Reuse of existing classes when building new classes and programs saves time and effort. Reuse also helps you build more reliable and effective systems, because existing classes and components often have gone through extensive *testing* (to locate problems), *debugging* (to correct those problems) and *performance tuning*. Just as the notion of *interchangeable parts* was crucial to the Industrial Revolution, *reusable classes* are crucial to the software revolution that's been spurred by object technology.

Software Engineering Observation 1.1

Use a building-block approach to creating your programs. Avoid reinventing the wheel—use existing pieces wherever possible. This software reuse is a key benefit of object-oriented programming.

Messages and Method Calls

When you drive a car, pressing its gas pedal sends a *message* to the car to perform a task—that is, to go faster. Similarly, you *send messages to an object*. Each message is implemented as a **method call** that tells a method of the object to perform its task. For example, a program might call a particular bank-account object's *deposit* method to increase the account's balance.

Attributes and Instance Variables

A car, besides having capabilities to accomplish tasks, also has *attributes*, such as its color, its number of doors, the amount of gas in its tank, its current speed and its record of total miles driven (i.e., its odometer reading). Like its capabilities, the car's attributes are represented as part of its design in its engineering diagrams (which, for example, include an odometer and a fuel gauge). As you drive an actual car, these attributes are carried along with the car. Every car maintains its *own* attributes. For example, each car knows how much gas is in its own gas tank, but *not* how much is in the tanks of *other* cars.

An object, similarly, has attributes that it carries along as it's used in a program. These attributes are specified as part of the object's class. For example, a bank-account object has a *balance attribute* that represents the amount of money in the account. Each bank-account object knows the balance in the account it represents, but *not* the balances of the *other* accounts in the bank. Attributes are specified by the class's **instance variables**.

Encapsulation

Classes **encapsulate** (i.e., wrap) attributes and methods into objects—an object's attributes and operations are intimately related. Objects may communicate with one another, but they're normally not allowed to know how other objects are implemented—implementation details are *hidden* within the objects themselves. This **information hiding**, as we'll see, is crucial to good software engineering.

Inheritance

A new class of objects can be created quickly and conveniently by **inheritance**—the new class absorbs the characteristics of an existing class, possibly customizing them and adding unique characteristics of its own. In our car analogy, an object of class "convertible" certainly *is an* object of the more *general* class "automobile," but more *specifically*, the roof can be raised or lowered.