

INTERNATIONAL
EDITION



Visual Basic® 2012

How To Program

SIXTH EDITION

Paul Deitel • Harvey Deitel • Abbey Deitel



ALWAYS LEARNING

PEARSON



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

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

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

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](https://plus.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.pearsoninternationaleditions.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.

International Edition contributions by

B. R. Chandavarkar

National Institute of Technology Karnataka, Surathkal



DEITEL

PEARSON

Boston Columbus Indianapolis New York San Francisco 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**
Publisher, International Edition: **Angshuman Chakraborty**
Publishing Administrator and Business Analyst, International Edition: **Shokhi Shah Khandelwal**
Associate Print & Media Editor, International Edition: **Anuprova Dey Chowdhuri**
Acquisitions Editor, International Edition: **Sandhya Ghoshal**
Publishing Administrator, International Edition: **Hema Mehta**
Project Editor, International Edition: **Karthik Subramanian**
Senior Manufacturing Controller, Production, International Edition: **Trudy Kimber**
Operations Specialist: **Lisa McDowell**
Art Director: **Anthony Gemmellaro**
Cover Design: **Jodi Notowitz**
Cover Photo: **majeczka/Shutterstock**
Media Project Manager: **Renata Butera**

Pearson Education Limited
Edinburgh Gate
Harlow
Essex CM20 2JE
England

and Associated Companies throughout the world

Visit us on the World Wide Web at:
www.pearsoninternationaleditions.com

© Pearson Education Limited 2014

The rights of Paul Deitel, Abbey Deitel and Harvey Deitel to be identified as authors of this work have been asserted by them in accordance with the Copyright, Designs and Patents Act 1988.

Authorized adaptation from the United States edition, entitled Visual Basic 2012 - How to Program, 6th edition, ISBN 978-0-13-340695-5, by Paul Deitel, Abbey Deitel and Harvey Deitel, published by Pearson Education © 2014.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without either the prior written permission of the publisher or a license permitting restricted copying in the United Kingdom issued by the Copyright Licensing Agency Ltd, Saffron House, 6-10 Kirby Street, London EC1N 8TS.

All trademarks used herein are the property of their respective owners. The use of any trademark in this text does not vest in the author or publisher any trademark ownership rights in such trademarks, nor does the use of such trademarks imply any affiliation with or endorsement of this book by such owners.

Microsoft and/or its respective suppliers make no representations about the suitability of the information contained in the documents and related graphics published as part of the services for any purpose. All such documents and related graphics are provided "as is" without warranty of any kind. Microsoft and/or its respective suppliers hereby disclaim all warranties and conditions with regard to this information, including all warranties and conditions of merchantability, whether express, implied or statutory, fitness for a particular purpose, title and non-infringement. In no event shall Microsoft and/or its respective suppliers be liable for any special, indirect or consequential damages or any damages whatsoever resulting from loss of use, data or profits, whether in an action of contract, negligence or other tortious action, arising out of or in connection with the use or performance of information available from the services.

The documents and related graphics contained herein could include technical inaccuracies or typographical errors. Changes are periodically added to the information herein. Microsoft and/or its respective suppliers may make improvements and/or changes in the product(s) and/or the program(s) described herein at any time. Partial screen shots may be viewed in full within the software version specified.

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.

ISBN 10: 0-273-79328-4

ISBN 13: 978-0-273-79328-1

British Library Cataloguing-in-Publication Data

A catalogue record for this book is available from the British Library

10 9 8 7 6 5 4 3 2 1

14 13 12 11 10

Typeset in AGaramond Regular by GEX Publishing Services

Printed and bound by Courier Westford in The United States of America

The publisher's policy is to use paper manufactured from sustainable forests.

To the Microsoft Visual Basic Language Team

Paul, Abbey and Harvey Deitel

Trademarks

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.pearsoninternationaleditions.com/deitel).

Preface	17
Before You Begin	31
1 Introduction to Computers, the Internet and Visual Basic	35
2 Dive Into Visual Studio Express 2012 for Windows Desktop	60
3 Introduction to Visual Basic Programming	93
4 Introduction to Problem Solving and Control Statements	135
5 Problem Solving and Control Statements: Part 2	180
6 Methods	222
7 Arrays	272
8 Files	320
9 Object-Oriented Programming: Classes and Objects	346
10 Object-Oriented Programming: Inheritance and Polymorphism	386
11 Introduction to LINQ	416
12 Databases and LINQ	433
13 Web App Development with ASP.NET	480
14 Windows Forms GUI: A Deeper Look	532
15 Graphics and Multimedia	588

Online Chapters	631
A Operator Precedence Chart	632
B Primitive Types	634
C Number Systems	635
D ASCII Character Set	647
E Unicode®	648
F Creating Console Applications	659
Index	663

Companion Website Online Content

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

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



Contents

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

Preface 17

Before You Begin 31

1 Introduction to Computers, the Internet and Visual Basic 35

- 1.1 Introduction 36
- 1.2 Hardware and Moore’s Law 36
- 1.3 Data Hierarchy 37
- 1.4 Computer Organization 39
- 1.5 Machine Languages, Assembly Languages and High-Level Languages 41
- 1.6 Object Technology 42
- 1.7 Internet and World Wide Web 44
- 1.8 Visual Basic 45
 - 1.8.1 Object-Oriented Programming 46
 - 1.8.2 Event-Driven Programming 46
 - 1.8.3 Visual Programming 46
 - 1.8.4 Internet and Web Programming 47
 - 1.8.5 Other Key Contemporary Programming Languages 47
- 1.9 Microsoft’s .NET 48
 - 1.9.1 .NET Framework 48
 - 1.9.2 Common Language Runtime 48
- 1.10 Microsoft’s Windows® Operating System 48
- 1.11 Windows Phone 8 for Smartphones 50
 - 1.11.1 Selling Your Apps in the Windows Phone Marketplace 50
 - 1.11.2 Free vs. Paid Apps 50
 - 1.11.3 Testing Your Windows Phone Apps 51
- 1.12 Windows Azure™ and Cloud Computing 51
- 1.13 Visual Studio Integrated Development Environment 51
- 1.14 Test-Driving the Visual Basic **Advanced Painter** App in Visual Studio 2012 52

2 Dive Into Visual Studio Express 2012 for Windows Desktop 60

- 2.1 Introduction 61

10 Contents

2.2	Overview of the Visual Studio 2012 IDE	61
2.3	Menu Bar and Toolbar	66
2.4	Navigating the Visual Studio IDE	69
2.4.1	Solution Explorer	71
2.4.2	Toolbox	72
2.4.3	Properties Window	73
2.5	Using Help	74
2.6	Using Visual App Development to Create a Simple App that Displays Text and an Image	75
2.7	Wrap-Up	85
2.8	Web Resources	85

3 Introduction to Visual Basic Programming 93

3.1	Introduction	94
3.2	Programmatically Displaying Text in a Label	95
3.2.1	Analyzing the Program	96
3.2.2	Modifying ASimpleApp to Programmatically Change the Label's Text Property	98
3.3	Addition Program	102
3.4	Building the Addition Program	105
3.5	Memory Concepts	112
3.6	Arithmetic	113
3.7	Decision Making: Equality and Relational Operators	117
3.8	Wrap-Up	122

4 Introduction to Problem Solving and Control Statements 135

4.1	Introduction	136
4.2	Algorithms	136
4.3	Pseudocode Algorithm	137
4.4	Control Structures	137
4.5	If...Then Selection Statement	140
4.6	If...Then...Else Selection Statement	141
4.7	Nested If...Then...Else Selection Statements	142
4.8	Repetition Statements	143
4.9	Compound Assignment Operators	145
4.10	Formulating Algorithms: Counter-Controlled Repetition	147
4.11	Formulating Algorithms: Nested Control Statements	153
4.12	Using the Debugger: Locating a Logic Error	159
4.12.1	Breakpoints and Running the Program	161
4.12.2	Data Tip Box	162
4.12.3	Locals Window	162
4.12.4	Using the Step Over Command to Execute Statements	163
4.13	Wrap-Up	164

5 Problem Solving and Control Statements: Part 2 180

5.1	Introduction	181
5.2	For...Next Repetition Statement	181
5.2.1	For...Next Statement Header Components	183
5.2.2	General Form of a For...Next Statement	183
5.2.3	Declaring the Control Variable Before a For...Next Statement	184
5.2.4	Using Expressions in the For...Next Statement's Header	184
5.2.5	For...Next Statement UML Activity Diagram	184
5.2.6	Local Type Inference	184
5.3	Examples Using the For...Next Statement	186
5.4	App: Interest Calculator	186
5.5	Formulating Algorithms: Nested Repetition Statements	190
5.6	Select...Case Multiple-Selection Statement	193
5.7	Do...Loop While and Do...Loop Until Repetition Statements	198
5.8	Using Exit to Terminate Repetition Statements	199
5.9	Using Continue in Repetition Statements	200
5.10	Logical Operators	200
5.11	App: Dental Payment Calculator	203
5.12	Wrap-Up	207

6 Methods 222

6.1	Introduction	223
6.2	Classes and Methods	223
6.3	Subroutines: Methods That Do Not Return a Value	225
6.4	Functions: Methods That Return a Value	229
6.5	Implicit Argument Conversions	231
6.6	Option Strict and Data-Type Conversions	232
6.7	Passing Arguments: Pass-by-Value vs. Pass-by-Reference	234
6.8	Scope of Declarations	237
6.9	Case Study: Random-Number Generation	240
6.9.1	Scaling and Shifting of Random Numbers	242
6.9.2	Randomly Selecting Images	243
6.9.3	Rolling Dice Repeatedly and Displaying Statistics	245
6.10	Case Study: A Game of Chance	247
6.11	Method Overloading	252
6.12	Optional Parameters	254
6.13	Using the Debugger: Debugging Commands	257
6.14	Wrap-Up	258

7 Arrays 272

7.1	Introduction	273
7.2	Arrays	273
7.3	Declaring and Allocating Arrays	274
7.4	Initializing the Values in an Array	275
7.5	Summing the Elements of an Array	276

12 Contents

7.6	Using Arrays to Analyze Survey Results	277
7.7	Die-Rolling App with an Array of Counters	280
7.8	Case Study: Flag Quiz	282
7.9	Passing an Array to a Method	286
7.10	For Each...Next Repetition Statement	289
7.11	Sorting an Array with Method Sort of Class Array	291
7.12	Searching an Array with Linear Search	293
7.13	Searching a Sorted Array with Array Method BinarySearch	295
7.14	Rectangular Arrays	296
7.15	Case Study: Maintaining Grades Using a Rectangular Array	298
7.16	Resizing an Array with the ReDim Statement	308
7.17	Wrap-Up	309

8 Files 320

8.1	Introduction	321
8.2	Data Hierarchy	321
8.3	Files and Streams	323
8.4	Test-Driving the Credit Inquiry App	324
8.5	Writing Data Sequentially to a Text File	326
8.5.1	Class CreateAccounts	329
8.5.2	Opening the File	330
8.5.3	Managing Resources with the Using Statement	331
8.5.4	Adding an Account to the File	332
8.5.5	Closing the File and Terminating the App	333
8.6	Building Menus with the Windows Forms Designer	334
8.7	Credit Inquiry App: Reading Data Sequentially from a Text File	336
8.7.1	Implementing the Credit Inquiry App	336
8.7.2	Selecting the File to Process	336
8.7.3	Specifying the Type of Records to Display	337
8.7.4	Displaying the Records	338
8.8	Wrap-Up	341

9 Object-Oriented Programming: Classes and Objects 346

9.1	Introduction	347
9.2	Classes, Objects, Methods and Instance Variables	347
9.3	Account Class	348
9.4	Value Types and Reference Types	354
9.5	Case Study: Card Shuffling and Dealing Simulation	355
9.6	Case Study: Time Class	361
9.7	Class Scope	368
9.8	Object Initializers	369
9.9	Auto-Implemented Properties	369
9.10	Using Me to Access the Current Object	370
9.11	Garbage Collection	370

9.12	Shared Class Members	371
9.13	Const and ReadOnly Fields	374
9.14	Shared Methods and Class Math	375
9.15	Object Browser	376
9.16	Wrap-Up	376

10 Object-Oriented Programming: Inheritance and Polymorphism **386**

10.1	Introduction	387
10.2	Base Classes and Derived Classes	387
10.3	Business Case Study: Commission Employees Class Hierarchy	389
	10.3.1 Creating Base Class <code>CommissionEmployee</code>	389
	10.3.2 Creating Derived Class <code>BasePlusCommissionEmployee</code>	392
	10.3.3 Testing Class <code>BasePlusCommissionEmployee</code>	395
10.4	Constructors in Derived Classes	397
10.5	Protected Members	397
10.6	Introduction to Polymorphism: A Polymorphic Video Game	398
10.7	Abstract Classes and Methods	399
10.8	Case Study: Payroll System Class Hierarchy Using Polymorphism	400
	10.8.1 Abstract Base Class <code>Employee</code>	401
	10.8.2 Concrete Derived Class <code>SalariedEmployee</code>	403
	10.8.3 Concrete Derived Class <code>CommissionEmployee</code>	404
	10.8.4 Indirect Concrete Derived Class <code>BasePlusCommissionEmployee</code>	406
	10.8.5 Demonstrating Polymorphic Processing	407
10.9	Online Case Study: Interfaces	409
10.10	Wrap-Up	410

11 Introduction to LINQ **416**

11.1	Introduction	417
11.2	Querying an Array of Primitive-Type Elements Using LINQ	418
11.3	Querying an Array of Reference-Type Elements Using LINQ	421
11.4	Deferred Execution and Transforming Query Results	427
11.5	LINQ Resource Center	428
11.6	Wrap-Up	429

12 Databases and LINQ **433**

12.1	Introduction	434
12.2	Relational Databases	435
12.3	A Books Database	436
12.4	LINQ to Entities and the ADO.NET Entity Framework	440
12.5	Querying a Database with LINQ	441
	12.5.1 Creating the ADO.NET Entity Data Model Class Library	442
	12.5.2 Creating a Windows Forms Project and Configuring It to Use the Entity Data Model	446

14 Contents

12.5.3	Data Bindings Between Controls and the Entity Data Model	448
12.6	Dynamically Binding Query Results	453
12.6.1	Creating the Display Query Results GUI	454
12.6.2	Coding the Display Query Results App	455
12.7	Retrieving Data from Multiple Tables with LINQ	457
12.8	Creating a Master/Detail View App	462
12.8.1	Creating the Master/Detail GUI	463
12.8.2	Coding the Master/Detail App	465
12.9	Address Book Case Study	466
12.9.1	Creating the Address Book App's GUI	467
12.9.2	Coding the Address Book App	468
12.10	Tools and Web Resources	472
12.11	Wrap-Up	472

13 Web App Development with ASP.NET 480

13.1	Introduction	481
13.2	Web Basics	482
13.3	Multitier App Architecture	483
13.4	Your First Web App	485
13.4.1	Building the WebTime App	487
13.4.2	Examining WebTime.aspx 's Code-Behind File	496
13.5	Standard Web Controls: Designing a Form	496
13.6	Validation Controls	501
13.7	Session Tracking	507
13.7.1	Cookies	509
13.7.2	Session Tracking with HttpSessionState	509
13.7.3	Options.aspx : Selecting a Programming Language	511
13.7.4	Recommendations.aspx : Displaying Recommendations Based on Session Values	515
13.8	Case Study: Database-Driven ASP.NET Guestbook	516
13.8.1	Building a Web Form that Displays Data from a Database	518
13.8.2	Modifying the Code-Behind File for the Guestbook App	523
13.9	Online Case Study: ASP.NET AJAX	524
13.10	Online Case Study: Password-Protected Books Database App	525
13.11	Wrap-Up	525

14 Windows Forms GUI: A Deeper Look 532

14.1	Introduction	533
14.2	Controls and Components	533
14.3	Creating Event Handlers	535
14.4	Control Properties and Layout	537
14.5	GroupBoxes and Panels	540
14.6	ToolTips	542
14.7	Mouse-Event Handling	544
14.8	Keyboard-Event Handling	547

14.9	Menus	550
14.10	MonthCalendar Control	559
14.11	DateTimePicker Control	560
14.12	LinkLabel Control	563
14.13	ListBox and CheckedListBox Controls	565
14.14	Multiple Document Interface (MDI) Windows	569
14.15	Visual Inheritance	577
14.16	Animation with the Timer Component	580
14.17	Wrap-Up	581

15 Graphics and Multimedia 588

15.1	Introduction	589
15.2	Drawing Classes and the Coordinate System	589
15.3	Graphics Contexts and Graphics Objects	590
15.4	Colors	591
15.5	Fonts	598
15.6	Drawing Lines, Rectangles and Ovals	602
15.7	Drawing Arcs	605
15.8	Drawing Polygons and Polylines	608
15.9	Additional Brush Types	609
15.10	Loading, Displaying and Scaling Images	614
15.11	Windows Media Player	616
15.12	Printing	617
15.13	Wrap-Up	623

Online Chapters 631

A Operator Precedence Chart 632

B Primitive Types 634

C Number Systems 635

C.1	Introduction	636
C.2	Abbreviating Binary Numbers as Octal and Hexadecimal Numbers	639
C.3	Converting Octal and Hexadecimal Numbers to Binary Numbers	640
C.4	Converting from Binary, Octal or Hexadecimal to Decimal	640
C.5	Converting from Decimal to Binary, Octal or Hexadecimal	641
C.6	Negative Binary Numbers: Two's-Complement Notation	643

D ASCII Character Set 647

E Unicode® 648

E.1	Introduction	649
-----	--------------	-----

E.2	Unicode Transformation Formats	650
E.3	Characters and Glyphs	651
E.4	Advantages/Disadvantages of Unicode	652
E.5	Using Unicode	652
E.6	Character Ranges	654

F **Creating Console Applications** **659**

F.1	Introduction	659
F.2	A Simple Console Application	659
F.3	Creating a Console Application	660

Index **663**

Companion Website Online Content

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

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



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/vb2012http and www.pearsoninternationalitions.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/vb2012http, join our social media communities on Facebook (www.deitel.com/DeitelFan), Twitter (@deitel), Google+ ([gplus.to/deitel](https://plus.google.com/+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	Concatenating 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

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 Pyramid	Tax Plan Alternatives; The “Fair Tax”
Gas Pump	Querying an Array of Invoice Objects	Telephone-Number Word Generator
Guess the Number App	Quiz Average App	Temperature Converter App
Image Flasher	Radio GUI	Triangles of Asterisks
Image Reflector App	Random Lines	Using the Debugger: Discount Calculator App
Inheritance Advantages	Reading Grades into a Two-Dimensional Array	Using the Debugger: Factorial App
Invoice Class	Retail Sales Calculator App	Using the Debugger: Savings Calculator App
Large-Type Displays for People with Low Vision	Road Sign Test App	Using the Debugger: Sibling Survey App
Lottery Number Generator	Sales Commissions	Vending Machine App
MDI Text Editor	Savings Account Class	Wage Calculator with Tax Calculations
Miles Per Gallon App	Screen Saver App	Web-Based Address Book
Modifying the Internal Data Representation of a Class	Shape Hierarchy	World Population Growth
Multiples	Simple Calculator App	Writing a Grade Report to a File
Notepad GUI	Simple Drawing App	
Nutrition Information XML Document	Snake PolyLine App	
Package Inheritance Hierarchy	Sorting and Ascending and Descending Order	
Page Hit Counter	Speech-Controlled Drawing App	
Parking Charges	Square Class	
Phone-Book Web Service	Student Inheritance Hierarchy	
Pig Latin		

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.pearsoninternationaleditions.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/vb2012http/jump_table.pdf). VideoNotes are free with the purchase of a *new* textbook.

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.
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 Services

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.

Instructor Supplements

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

- *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. 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

The publishers wish to thank Somitra Kr Sanadhya of Indraprastha Institute of Information Technology, Delhi for reviewing the content of the International Edition.

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/vb2012http/

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/vb2012http/. 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.



- 1.1 Introduction
- 1.2 Hardware and Moore's Law
- 1.3 Data Hierarchy
- 1.4 Computer Organization
- 1.5 Machine Languages, Assembly Languages and High-Level Languages
- 1.6 Object Technology
- 1.7 Internet and World Wide Web
- 1.8 Visual Basic
 - 1.8.1 Object-Oriented Programming
 - 1.8.2 Event-Driven Programming
 - 1.8.3 Visual Programming
 - 1.8.4 Internet and Web Programming
 - 1.8.5 Other Key Contemporary Programming Languages
- 1.9 Microsoft's .NET
 - 1.9.1 .NET Framework
 - 1.9.2 Common Language Runtime
- 1.10 Microsoft's Windows® Operating System
- 1.11 Windows Phone 8 for Smartphones
 - 1.11.1 Selling Your Apps in the Windows Phone Marketplace
 - 1.11.2 Free vs. Paid Apps
 - 1.11.3 Testing Your Windows Phone Apps
- 1.12 Windows Azure™ and Cloud Computing
- 1.13 Visual Studio Integrated Development Environment
- 1.14 Test-Driving the Visual Basic **Advanced Painter** App in Visual Studio 2012

Self-Review Exercises | Answers to Self-Review Exercises | Exercises | Making a Difference Exercises

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.

I.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.

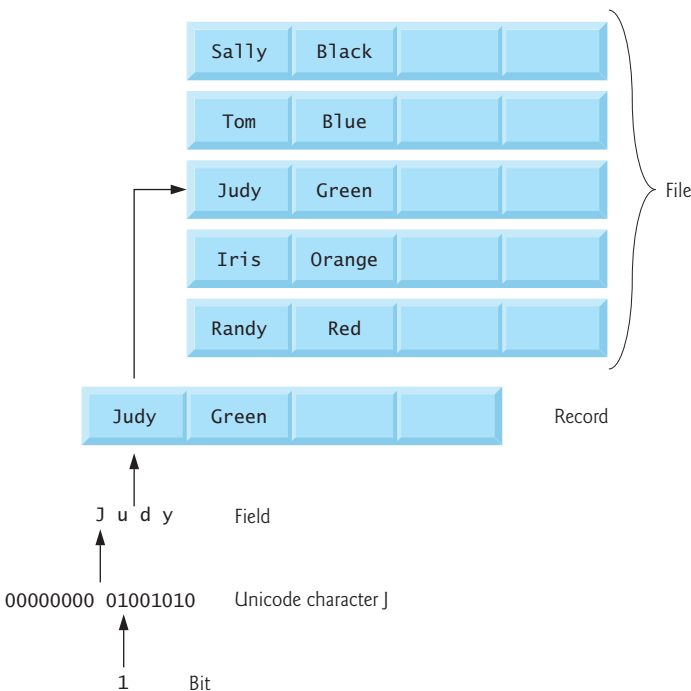


Fig. 1.1 | Data hierarchy.

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.

I.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, braking 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.

Object-Oriented Analysis and Design (OOAD)

Soon you'll be writing programs in Visual Basic. Perhaps, like many programmers, you'll simply turn on your computer and start typing. This approach may work for small programs (like the ones we present in the early chapters of this book), but what if you were asked to create a software system to control thousands of automated teller machines for a major bank? Or suppose you were asked to work on a team of thousands of software developers building the next U.S. air traffic control system? For projects so large and complex, you should not simply sit down and start writing programs.

To create the best solutions, you should follow a detailed **analysis** process for determining your project's **requirements** (i.e., defining *what* the system is supposed to do) and developing a **design** that satisfies them (i.e., deciding *how* the system should do it). Ideally, you'd go through this process and carefully review the design (and have your design reviewed by other software professionals) before writing any code. If this process involves analyzing and designing your system from an object-oriented point of view, it's called an **object-oriented analysis and design (OOAD) process**. Languages like Visual Basic are object oriented. Programming in such a language, called **object-oriented programming (OOP)**, allows you to implement an object-oriented design as a working system.

The UML (Unified Modeling Language)

Although many different OOAD processes exist, a single graphical language for communicating the results of *any* OOAD process has come into wide use. This language, known as the *Unified Modeling Language (UML)*, is now the most widely used graphical scheme for modeling object-oriented systems. We present our first simple UML diagrams in Chapters 4 and 5, then use them in our deeper treatment of object-oriented programming. In our optional ATM Software Engineering Case Study in Chapters 30–31 we present a subset of the UML's features as we guide you through a simple object-oriented design experience.

1.7 Internet and World Wide Web

In the late 1960s, ARPA—the Advanced Research Projects Agency of the United States Department of Defense—rolled out plans to network the main computer systems of approximately a dozen ARPA-funded universities and research institutions. The computers were to be connected with communications lines operating at a then-stunning 56 Kbps (1 Kbps is equal to 1,024 bits per second), at a time when most people (of the few who even had networking access) were connecting over telephone lines to computers at a rate of 110 bits per second. Academic research was about to take a giant leap forward. ARPA proceeded to implement what quickly became known as the ARPAnet, the precursor of today's **Internet**.

Things worked out differently from the original plan. Although the ARPAnet enabled researchers to network their computers, its main benefit proved to be the capability for quick and easy communication via what came to be known as electronic mail (e-mail). This is true even on today's Internet, with e-mail, instant messaging, file transfer and social media such as Facebook and Twitter, enabling billions of people worldwide to communicate quickly and easily.

The protocol (set of rules) for communicating over the ARPAnet became known as the **Transmission Control Protocol (TCP)**. TCP ensured that messages, consisting of

sequentially numbered pieces called *packets*, were properly routed from sender to receiver, arrived intact and were assembled in the correct order.

The Internet: A Network of Networks

In parallel with the early evolution of the Internet, organizations worldwide were implementing their own networks for both intraorganization (that is, within an organization) and interorganization (that is, between organizations) communication. A huge variety of networking hardware and software appeared. One challenge was to enable these different networks to communicate with each other. ARPA accomplished this by developing the Internet Protocol (IP), which created a true “network of networks,” the current architecture of the Internet. The combined set of protocols is now called **TCP/IP**.

Businesses rapidly realized that by using the Internet, they could improve their operations and offer new and better services to their clients. Companies started spending large amounts of money to develop and enhance their Internet presence. This generated fierce competition among communications carriers and hardware and software suppliers to meet the increased infrastructure demand. As a result, **bandwidth**—the information-carrying capacity of communications lines—on the Internet has increased tremendously, while hardware costs have plummeted.

The World Wide Web: Making the Internet User-Friendly

The **World Wide Web** (simply called “the web”) is a collection of hardware and software associated with the Internet that allows computer users to locate and view multimedia-based documents (documents with various combinations of text, graphics, animations, audios and videos) on almost any subject. The introduction of the web was a relatively recent event. In 1989, Tim Berners-Lee of CERN (the European Organization for Nuclear Research) began to develop a technology for sharing information via “hyperlinked” text documents. Berners-Lee called his invention the **HyperText Markup Language (HTML)**. He also wrote communication protocols such as **HyperText Transfer Protocol (HTTP)** to form the backbone of his new hypertext information system, which he referred to as the World Wide Web.

In 1994, Berners-Lee founded an organization, called the **World Wide Web Consortium (W3C)**, www.w3.org, devoted to developing web technologies. One of the W3C’s primary goals is to make the web universally accessible to everyone regardless of disabilities, language or culture.

In this book, you’ll use Visual Basic and other Microsoft technologies to build web-based apps.

I.8 Visual Basic

The **Visual Basic** programming language evolved from BASIC, developed in the 1960s at Dartmouth College for introducing novices to fundamental programming techniques. When Bill Gates founded Microsoft in the 1970s, he implemented BASIC on several early personal computers. In the late 1980s and the early 1990s, Microsoft developed the Microsoft Windows **graphical user interface (GUI)**—the visual part of the operating system with which users interact. With the creation of the Windows GUI, the natural evolution of BASIC was to Visual Basic, introduced by Microsoft in 1991 to make programming Windows apps easier.

Visual Basic 6 was introduced in 1998. It was designed to make programming fast and easy, however it was not intended for building industrial strength apps. In 2002, Microsoft released Visual Basic .NET—a more robust language that’s appropriate for the most demanding app development tasks, especially for building today’s large-scale enterprise apps, and web-based, mobile and cloud apps.

Visual Basic 2012 is one of the most *productive* programming languages, requiring less development time than other popular high-level programming languages such as Java, C, C++ and C#—the sister language to Visual Basic.³

1.8.1 Object-Oriented Programming

Visual Basic is *object oriented*—we’ve discussed the basics of object technology and will present a rich treatment of object-oriented programming throughout the book. Visual Basic has access to the powerful **.NET Framework Class Library**—a vast collection of pre-built components that enable you to develop apps quickly (Fig. 1.3). We’ll say more about .NET in Section 1.9.

Some key capabilities in the .NET Framework Class Library	
Database	Computer networking
Building web apps	Debugging
Graphics	Multithreading
Input/output	File processing
Networking	Security
Permissions	Web communication
Mobile	Graphical user interface
String processing	Data structures

Fig. 1.3 | Some key capabilities in the .NET Framework Class Library.

1.8.2 Event-Driven Programming

Visual Basic is **event driven**. You’ll write programs that respond to user-initiated **events** such as mouse clicks, keystrokes, timer expirations and—new in Visual Basic 2012—*touches* and *finger swipes*—geatures that are so widely used on smartphones and tablets.

1.8.3 Visual Programming

Visual Basic is a *visual programming language*—in addition to writing program statements to build portions of your apps, you’ll also use Visual Studio’s graphical user interface (GUI) to conveniently drag and drop predefined objects like *buttons* and *textboxes* into place on your screen, and label and resize them. Visual Studio will write much of the GUI code for you.

3. www.drdoobs.com/jvm/the-comparative-productivity-of-programm/240005881.

1.8.4 Internet and Web Programming

Today's apps can be written with the aim of communicating among the world's computers. As you'll see, this is the focus of Microsoft's .NET strategy. In Chapters 13 and 24, you'll build web-based apps with Visual Basic and Microsoft's **ASP.NET** technology.

1.8.5 Other Key Contemporary Programming Languages

Figure 1.4 summarizes some other important programming languages. The last four are object-oriented languages that have capabilities comparable to those of Visual Basic. C, which is not object oriented, is an ancestor of the other four languages in the figure.

Programming language	Description
C	C was implemented in 1972 by Dennis Ritchie at Bell Laboratories. It initially became widely known as the UNIX operating system's development language. Today, most of the code for general-purpose operating systems is written in C or C++.
C++	C++, an extension of C, was developed by Bjarne Stroustrup in the early 1980s at Bell Laboratories. C++ provides several features that "spruce up" the C language, but more important, it provides capabilities for <i>object-oriented programming</i> . It's often used in applications with stringent performance requirements such as operating systems, real-time systems, embedded systems and communications systems. Visual C++ is Microsoft's version of the language.
Java	In the 1990s, Sun Microsystems (now part of Oracle) developed the C++-based object-oriented programming language called Java. A key goal of Java is to be able to write programs that will run on a great variety of computer systems and computer-control devices—this is sometimes called <i>write once, run anywhere</i> . Java is used to develop large-scale enterprise applications, to enhance the functionality of web servers (the computers that provide the content we see in our web browsers), to provide applications for consumer devices (e.g., smartphones, tablets, television set-top boxes, appliances, automobiles and more) and for many other purposes.
C#	In 2000, Microsoft announced the C# programming language. C# has roots in the C, C++ and Java programming languages. It has similar capabilities to Java and is appropriate for the most demanding application development tasks, especially for building today's large-scale enterprise applications, and web-based, mobile and cloud-based apps. The latest versions of Visual Basic have capabilities comparable to those of C#.
Objective-C	Objective-C is another object-oriented language based on C. It was developed at Stepstone in the early 1980s and later acquired by Next, which in turn was acquired by Apple. It has become the key programming language for the Mac OS X desktop operating system and all iOS-based devices, such as iPods, iPhones and iPads.

Fig. 1.4 | Other programming languages.

1.9 Microsoft's .NET

In 2000, Microsoft announced its **.NET initiative** (www.microsoft.com/net), a broad vision for using the Internet and the web in the development, engineering, distribution and use of software. Rather than forcing you to use a single programming language, .NET permits you to create apps in *any* .NET-compatible language (such as Visual Basic, C#, Visual C++ and many others). Part of the initiative includes Microsoft's ASP.NET technology, which is used to build web applications that users interact with via their web browsers.

1.9.1 .NET Framework

The **.NET Framework** contains the .NET Framework **Class Library**, which provides many capabilities that you'll use to build substantial Visual Basic apps quickly and easily. The .NET Framework Class Library has *thousands* of valuable *prebuilt* classes that have been tested and tuned to maximize performance. You'll learn how to create your own classes, but you should *re-use* the .NET Framework classes when possible to speed up the software development process, while enhancing the quality and performance of the software you develop.

1.9.2 Common Language Runtime

The **Common Language Runtime (CLR)**, another key part of the .NET Framework, executes .NET programs and provides functionality to make them easier to develop and debug. The CLR is a **virtual machine (VM)**—software that manages the execution of programs and hides from them the underlying operating system and hardware. The source code for programs that are executed and managed by the CLR is called *managed code*. The CLR provides various services to managed code, such as integrating software components written in different .NET languages, error handling between such components, enhanced security, automatic memory management and more. Unmanaged-code programs do not have access to the CLR's services, which makes unmanaged code more difficult to write.⁴

1.10 Microsoft's Windows® Operating System

Microsoft's Windows is the most widely used desktop operating system worldwide. **Operating systems** are software systems that make using computers more convenient for users, app developers and system administrators. They provide *services* that allow each app to execute safely, efficiently and *concurrently* (i.e., in parallel) with other apps. Other popular desktop operating systems include Linux and Mac OS X. Popular *mobile operating systems* used in smartphones and tablets include Microsoft's Windows Phone, Google's Android, Apple's iOS (for iPhone, iPad and iPod Touch devices) and BlackBerry OS. Figure 1.5 presents the evolution of the Windows operating system.

4. msdn.microsoft.com/en-us/library/8bs2ecf4.aspx.

Version	Description
Windows in the 1990s	In the mid-1980s, Microsoft developed the Windows operating system based on a graphical user interface with buttons, textboxes, menus and other graphical elements. The various versions released throughout the 1990s were intended for personal computing. Microsoft entered the corporate operating systems market with the 1993 release of <i>Windows NT</i> .
Windows XP and Windows Vista	<i>Windows XP</i> was released in 2001 and combined Microsoft's corporate and consumer operating system lines. It remains popular today—according to a 2012 Netmarketshare study, it's used on more than 40% of Windows computers (netmarketshare.com/operating-system-market-share.aspx?qprid=10&qpcustomd=0). <i>Windows Vista</i> , released in 2007, offered the attractive new Aero user interface, many powerful enhancements and new apps and enhanced security. But Vista never caught on—today, it has “only” six percent of the total desktop operating systems market share (that's still a pretty significant number; netmarketshare.com/operating-system-market-share.aspx?qprid=10&qpcustomd=0).
Windows 7	<i>Windows 7</i> , now the most widely used version of Windows, includes enhancements to the Aero user interface, faster startup times, further refinement of Vista's security features, touch-screen with multi-touch support, and more. Windows 7 had a 43% market share, and overall, Windows (including Windows 7, Windows XP and Windows Vista) had over 90% of the desktop operating system market share worldwide (netmarketshare.com/operating-system-market-share.aspx?qprid=10&qpcustomd=0). The core chapters of this book use Windows 7, Visual Studio 2012 and Visual Basic 2012.
Windows 8 for Desktops and Tablets	Windows 8 , released in 2012 provides a similar platform (the underlying system on which apps run) and <i>user experience</i> across a wide range of devices including personal computers, tablets and the Xbox Live online game service. The new look-and-feel features a Start screen with <i>tiles</i> that represent each app, similar to that of <i>Windows Phone</i> —Microsoft's operating system for smartphones. Windows 8 features <i>multi-touch</i> support for <i>touchpads</i> and <i>touchscreen</i> devices, enhanced security features and more.
Windows 8 UI (User Interface)	Visual Basic 2012 supports the new Windows 8 UI (previously called “Metro”) which has a clean look-and-feel with minimal distractions to the user. Windows 8 apps feature a <i>chromeless window</i> —there's no longer a border around the window with the typical interface elements such as title bars and menus. These elements are <i>hidden</i> , allowing apps to fill the <i>entire</i> screen, which is particularly helpful on smaller screens such as tablets and smartphones. The interface elements are displayed in the <i>app bar</i> when the user <i>swipes</i> the top or bottom of the screen by holding down the mouse button, moving the mouse in the swipe direction and releasing the mouse button; this can be done with a <i>finger swipe</i> on a touch-screen device. We discuss Windows 8 and the Windows 8 UI in Chapter 20 and Windows Phone 8 in Chapter 22.

Fig. I.5 | The evolution of the Windows operating system.