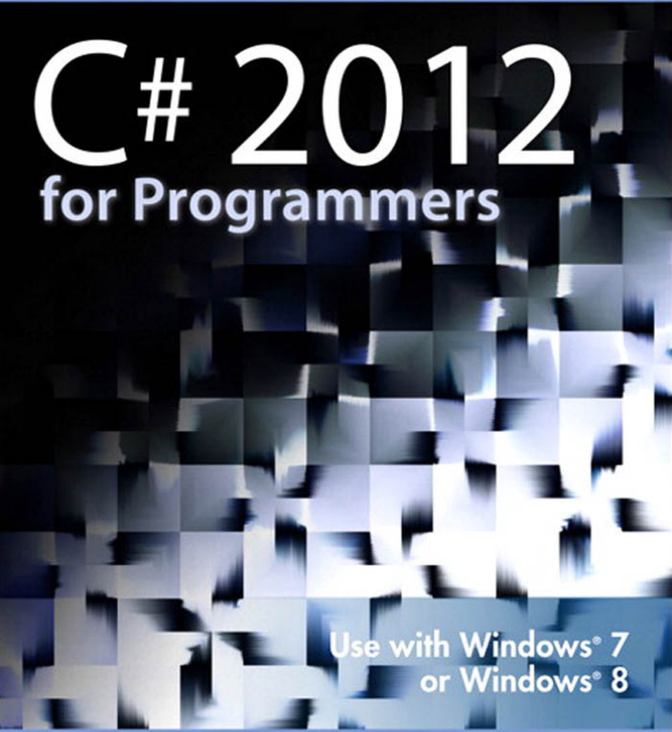
DEITEL® DEVELOPER SERIES

C# 2012 for Programmers



Use with Windows® 7 or Windows® 8

PAUL DEITEL • HARVEY DEITEL

FREE SAMPLE CHAPTER

SHARE WITH OTHERS

C# 2012 for Programmers
Fifth Edition

Deitel® Developer Series

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the publisher was aware of a trademark claim, the designations have been printed with initial capital letters or in all capitals.

The authors and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

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.

The publisher offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales, which may include electronic versions and/or custom covers and content particular to your business, training goals, marketing focus, and branding interests. For more information, please contact:

U. S. Corporate and Government Sales
(800) 382-3419
corpsales@pearsontechgroup.com

For sales outside the U. S., please contact:

International Sales
international@pearsoned.com

Visit us on the Web: informit.com/ph

Library of Congress Cataloging-in-Publication Data

On file

© 2014 Pearson Education, Inc.

All rights reserved. Printed in the United States of America. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. To obtain permission to use material from this work, please submit a written request to Pearson Education, Inc., Permissions Department, One Lake Street, Upper Saddle River, New Jersey 07458, or you may fax your request to (201) 236-3290.

ISBN-13: 978-0-13-344057-7
ISBN-10: 0-13-344057-5

Text printed in the United States at Edwards Brothers Malloy in Ann Arbor, Michigan.
First printing, September 2013

C# 2012 FOR PROGRAMMERS
FIFTH EDITION
DEITEL® DEVELOPER SERIES

Paul Deitel
Deitel & Associates, Inc.

Harvey Deitel
Deitel & Associates, Inc.

DEITEL®

PRENTICE HALL

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

Trademarks

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

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

UNIX is a registered trademark of The Open Group.

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

To our review team

Shay Friedman
Octavio Hernandez
Stephen Hustedde
José Antonio González Seco
Shawn Weisfeld

We are grateful for your guidance and expertise.

Paul and Harvey Deitel

This page intentionally left blank

Contents

Preface **xxiii**

Before You Begin **xxxi**

1 Introduction **1**

1.1 Introduction 2
1.2 Object Technology 2
1.3 C# 5
1.3.1 Object-Oriented Programming 5
1.3.2 Event-Driven Programming 5
1.3.3 Visual Programming 6
1.3.4 An International Standard; Other C# Implementations 6
1.3.5 Internet and Web Programming 6
1.3.6 Introducing `async`/`await` 6
1.4 Microsoft's .NET 6
1.4.1 .NET Framework 6
1.4.2 Common Language Runtime 7
1.4.3 Platform Independence 7
1.4.4 Language Interoperability 7
1.5 Microsoft's Windows® Operating System 8
1.6 Windows Phone 8 for Smartphones 9
1.6.1 Selling Your Apps in the Windows Phone Marketplace 10
1.6.2 Free vs. Paid Apps 10
1.6.3 Testing Your Windows Phone Apps 10
1.7 Windows Azure™ and Cloud Computing 11
1.8 Visual Studio Express 2012 Integrated Development Environment 11
1.9 **Painter** Test-Drive in Visual Studio Express 2012 for Windows Desktop 11
1.10 **Painter** Test-Drive in Visual Studio Express 2012 for Windows 8 15

2 Dive Into® Visual Studio Express 2012 for Windows Desktop **20**

2.1 Introduction 21
2.2 Overview of the Visual Studio Express 2012 IDE 21
2.3 Menu Bar and Toolbar 26

2.4 Navigating the Visual Studio IDE 28
2.4.1 **Solution Explorer** 30
2.4.2 **Toolbox** 31
2.4.3 **Properties** Window 31
2.5 Using Help 33
2.6 Using Visual App Development to Create a Simple App that Displays Text and an Image 34
2.7 Wrap-Up 44
2.8 Web Resources 45

3 Introduction to C# Apps 46

3.1 Introduction 47
3.2 A Simple C# App: Displaying a Line of Text 47
3.3 Creating a Simple App in Visual Studio 52
3.4 Modifying Your Simple C# App 58
3.5 Formatting Text with `Console.Write` and `Console.WriteLine` 60
3.6 Another C# App: Adding Integers 61
3.7 Arithmetic 65
3.8 Decision Making: Equality and Relational Operators 67
3.9 Wrap-Up 71

4 Introduction to Classes, Objects, Methods and `string`s 72

4.1 Introduction 73
4.2 Classes, Objects, Methods, Properties and Instance Variables 73
4.3 Declaring a Class with a Method and Instantiating an Object of a Class 74
4.4 Declaring a Method with a Parameter 78
4.5 Instance Variables and Properties 82
4.6 UML Class Diagram with a Property 86
4.7 Software Engineering with Properties and `set` and `get` Accessors 87
4.8 Auto-Implemented Properties 88
4.9 Value Types vs. Reference Types 89
4.10 Initializing Objects with Constructors 90
4.11 Floating-Point Numbers and Type `decimal` 93
4.12 Wrap-Up 99

5 Control Statements: Part 1 101

5.1 Introduction 102
5.2 Control Structures 102
5.3 `if` Single-Selection Statement 104
5.4 `if...else` Double-Selection Statement 105
5.5 `while` Repetition Statement 109

5.6 Counter-Controlled Repetition 110
5.7 Sentinel-Controlled Repetition 113
5.8 Nested Control Statements 118
5.9 Compound Assignment Operators 121
5.10 Increment and Decrement Operators 122
5.11 Simple Types 125
5.12 Wrap-Up 125

6 Control Statements: Part 2 126

6.1 Introduction 127
6.2 Essentials of Counter-Controlled Repetition 127
6.3 `for` Repetition Statement 128
6.4 Examples Using the `for` Statement 132
6.5 `do...while` Repetition Statement 136
6.6 `switch` Multiple-Selection Statement 137
6.7 `break` and `continue` Statements 145
6.8 Logical Operators 147
6.9 Wrap-Up 153

7 Methods: A Deeper Look 154

7.1 Introduction 155
7.2 Packaging Code in C# 155
7.3 `static` Methods, `static` Variables and Class `Math` 156
7.4 Declaring Methods with Multiple Parameters 158
7.5 Notes on Declaring and Using Methods 162
7.6 Method-Call Stack and Activation Records 163
7.7 Argument Promotion and Casting 163
7.8 The .NET Framework Class Library 165
7.9 Case Study: Random-Number Generation 167
 7.9.1 Scaling and Shifting Random Numbers 171
 7.9.2 Random-Number Repeatability for Testing and Debugging 171
7.10 Case Study: A Game of Chance; Introducing Enumerations 172
7.11 Scope of Declarations 177
7.12 Method Overloading 179
7.13 Optional Parameters 182
7.14 Named Parameters 183
7.15 Recursion 184
7.16 Passing Arguments: Pass-by-Value vs. Pass-by-Reference 187
7.17 Wrap-Up 191

8 Arrays; Introduction to Exception Handling 192

8.1 Introduction 193
8.2 Arrays 193

8.3 Declaring and Creating Arrays 195
8.4 Examples Using Arrays 196
8.4.1 Creating and Initializing an Array 196
8.4.2 Using an Array Initializer 197
8.4.3 Calculating a Value to Store in Each Array Element 198
8.4.4 Summing the Elements of an Array 199
8.4.5 Using Bar Charts to Display Array Data Graphically 200
8.4.6 Using the Elements of an Array as Counters 202
8.4.7 Using Arrays to Analyze Survey Results; Introduction to Exception Handling 203
8.5 Case Study: Card Shuffling and Dealing Simulation 206
8.6 `foreach` Statement 210
8.7 Passing Arrays and Array Elements to Methods 212
8.8 Passing Arrays by Value and by Reference 214
8.9 Case Study: `GradeBook` Using an Array to Store Grades 218
8.10 Multidimensional Arrays 223
8.11 Case Study: `GradeBook` Using a Rectangular Array 228
8.12 Variable-Length Argument Lists 234
8.13 Using Command-Line Arguments 236
8.14 Wrap-Up 238

9 Introduction to LINQ and the `List` Collection 239

9.1 Introduction 240
9.2 Querying an Array of `int` Values Using LINQ 241
9.3 Querying an Array of `Employee` Objects Using LINQ 245
9.4 Introduction to Collections 250
9.5 Querying a Generic Collection Using LINQ 253
9.6 Wrap-Up 255
9.7 Deitel LINQ Resource Center 255

10 Classes and Objects: A Deeper Look 256

10.1 Introduction 257
10.2 `Time` Class Case Study 257
10.3 Controlling Access to Members 261
10.4 Referring to the Current Object's Members with the `this` Reference 262
10.5 `Time` Class Case Study: Overloaded Constructors 264
10.6 Default and Parameterless Constructors 270
10.7 Composition 271
10.8 Garbage Collection and Destructors 274
10.9 `static` Class Members 275
10.10 `readonly` Instance Variables 278
10.11 Data Abstraction and Encapsulation 279
10.12 **Class View** and **Object Browser** 281

10.13 Object Initializers 283
10.14 Wrap-Up 283

11 Object-Oriented Programming: Inheritance 285
11.1 Introduction 286
11.2 Base Classes and Derived Classes 287
11.3 protected Members 289
11.4 Relationship between Base Classes and Derived Classes 290
11.4.1 Creating and Using a CommissionEmployee Class 290
11.4.2 Creating a BasePlusCommissionEmployee Class without Using Inheritance 295
11.4.3 Creating a CommissionEmployee–BasePlusCommissionEmployee Inheritance Hierarchy 300
11.4.4 CommissionEmployee–BasePlusCommissionEmployee Inheritance Hierarchy Using protected Instance Variables 303
11.4.5 CommissionEmployee–BasePlusCommissionEmployee Inheritance Hierarchy Using private Instance Variables 308
11.5 Constructors in Derived Classes 313
11.6 Software Engineering with Inheritance 314
11.7 Class object 314
11.8 Wrap-Up 315

12 OOP: Polymorphism, Interfaces and Operator Overloading 317
12.1 Introduction 318
12.2 Polymorphism Examples 320
12.3 Demonstrating Polymorphic Behavior 321
12.4 Abstract Classes and Methods 324
12.5 Case Study: Payroll System Using Polymorphism 326
12.5.1 Creating Abstract Base Class Employee 327
12.5.2 Creating Concrete Derived Class SalariedEmployee 329
12.5.3 Creating Concrete Derived Class HourlyEmployee 331
12.5.4 Creating Concrete Derived Class CommissionEmployee 333
12.5.5 Creating Indirect Concrete Derived Class BasePlusCommissionEmployee 334
12.5.6 Polymorphic Processing, Operator is and Downcasting 336
12.5.7 Summary of the Allowed Assignments Between Base-Class and Derived-Class Variables 341
12.6 sealed Methods and Classes 342
12.7 Case Study: Creating and Using Interfaces 342
12.7.1 Developing an IPayable Hierarchy 344
12.7.2 Declaring Interface IPayable 345
12.7.3 Creating Class Invoice 345
12.7.4 Modifying Class Employee to Implement Interface IPayable 347

12.7.5 Modifying Class SalariedEmployee for Use with IPayable 349
12.7.6 Using Interface IPayable to Process Invoices and Employees Polymorphically 350
12.7.7 Common Interfaces of the .NET Framework Class Library 352
12.8 Operator Overloading 353
12.9 Wrap-Up 356

13 Exception Handling: A Deeper Look 358

13.1 Introduction 359
13.2 Example: Divide by Zero without Exception Handling 360
13.3 Example: Handling DivideByZeroExceptions and FormatExceptions 363
13.3.1 Enclosing Code in a try Block 365
13.3.2 Catching Exceptions 365
13.3.3 Uncaught Exceptions 366
13.3.4 Termination Model of Exception Handling 367
13.3.5 Flow of Control When Exceptions Occur 367
13.4 .NET Exception Hierarchy 368
13.4.1 Class SystemException 368
13.4.2 Determining Which Exceptions a Method Throws 369
13.5 finally Block 369
13.6 The using Statement 376
13.7 Exception Properties 377
13.8 User-Defined Exception Classes 381
13.9 Wrap-Up 385

14 Graphical User Interfaces with Windows Forms: Part 1 386

14.1 Introduction 387
14.2 Windows Forms 388
14.3 Event Handling 390
14.3.1 A Simple Event-Driven GUI 390
14.3.2 Auto-Generated GUI Code 392
14.3.3 Delegates and the Event-Handling Mechanism 394
14.3.4 Another Way to Create Event Handlers 395
14.3.5 Locating Event Information 396
14.4 Control Properties and Layout 397
14.5 Labels, TextBoxes and Buttons 401
14.6 GroupBoxes and Panels 404
14.7 CheckBoxes and RadioButtons 407
14.8 PictureBoxes 415
14.9 ToolTips 417
14.10 NumericUpDown Control 419
14.11 Mouse-Event Handling 421

14.12 Keyboard-Event Handling 424
14.13 Wrap-Up 427

15 Graphical User Interfaces with Windows Forms: Part 2 428

15.1 Introduction 429
15.2 Menus 429
15.3 `MonthCalendar` Control 438
15.4 `DateTimePicker` Control 439
15.5 `LinkLabel` Control 442
15.6 `ListBox` Control 446
15.7 `CheckedListBox` Control 450
15.8 `ComboBox` Control 453
15.9 `TreeView` Control 457
15.10 `ListView` Control 462
15.11 `TabControl` Control 468
15.12 Multiple Document Interface (MDI) Windows 473
15.13 Visual Inheritance 480
15.14 User-Defined Controls 485
15.15 Wrap-Up 489

16 Strings and Characters: A Deeper Look 490

16.1 Introduction 491
16.2 Fundamentals of Characters and Strings 492
16.3 `string` Constructors 493
16.4 `string` Indexer, `Length` Property and `CopyTo` Method 494
16.5 Comparing `strings` 495
16.6 Locating Characters and Substrings in `strings` 498
16.7 Extracting Substrings from `strings` 501
16.8 Concatenating `strings` 502
16.9 Miscellaneous `string` Methods 503
16.10 Class `StringBuilder` 504
16.11 `Length` and `Capacity` Properties, `EnsureCapacity` Method and Indexer of Class `StringBuilder` 505
16.12 `Append` and `AppendFormat` Methods of Class `StringBuilder` 507
16.13 `Insert`, `Remove` and `Replace` Methods of Class `StringBuilder` 509
16.14 `Char` Methods 512
16.15 (Online) Introduction to Regular Expressions 514
16.16 Wrap-Up 515

17 Files and Streams 516

17.1 Introduction 517
17.2 Data Hierarchy 517

17.3 Files and Streams 519
17.4 Classes `File` and `Directory` 520
17.5 Creating a Sequential-Access Text File 529
17.6 Reading Data from a Sequential-Access Text File 538
17.7 Case Study: Credit Inquiry Program 542
17.8 Serialization 548
17.9 Creating a Sequential-Access File Using Object Serialization 549
17.10 Reading and Deserializing Data from a Binary File 553
17.11 Wrap-Up 555

18 Generics 557

18.1 Introduction 558
18.2 Motivation for Generic Methods 559
18.3 Generic-Method Implementation 561
18.4 Type Constraints 564
18.5 Overloading Generic Methods 566
18.6 Generic Classes 567
18.7 Wrap-Up 576

19 Collections 577

19.1 Introduction 578
19.2 Collections Overview 578
19.3 Class `Array` and Enumerators 581
19.4 Nongeneric Collections 584
19.4.1 Class `ArrayList` 584
19.4.2 Class `Stack` 589
19.4.3 Class `Hashtable` 591
19.5 Generic Collections 596
19.5.1 Generic Class `SortedDictionary` 597
19.5.2 Generic Class `LinkedList` 599
19.6 Covariance and Contravariance for Generic Types 603
19.7 Wrap-Up 606

20 Databases and LINQ 607

20.1 Introduction 608
20.2 Relational Databases 609
20.3 A Books Database 610
20.4 LINQ to Entities and the ADO.NET Entity Framework 614
20.5 Querying a Database with LINQ 615
20.5.1 Creating the ADO.NET Entity Data Model Class Library 616
20.5.2 Creating a Windows Forms Project and Configuring It to Use the Entity Data Model 620
20.5.3 Data Bindings Between Controls and the Entity Data Model 622

20.6 Dynamically Binding Query Results 627
20.6.1 Creating the **Display Query Results** GUI 628
20.6.2 Coding the **Display Query Results** App 629
20.7 Retrieving Data from Multiple Tables with LINQ 632
20.8 Creating a Master/Detail View App 637
20.8.1 Creating the Master/Detail GUI 638
20.8.2 Coding the Master/Detail App 639
20.9 Address Book Case Study 641
20.9.1 Creating the **Address Book** App's GUI 642
20.9.2 Coding the **Address Book** App 643
20.10 Tools and Web Resources 647
20.11 Wrap-Up 647

21 Web App Development with ASP.NET 649

21.1 Introduction 650
21.2 Web Basics 651
21.3 Multitier App Architecture 652
21.4 Your First Web App 654
21.4.1 Building the `WebTime` App 656
21.4.2 Examining `WebTime.aspx`'s Code-Behind File 665
21.5 Standard Web Controls: Designing a Form 666
21.6 Validation Controls 670
21.7 Session Tracking 677
21.7.1 Cookies 678
21.7.2 Session Tracking with `HttpSessionState` 679
21.7.3 `Options.aspx`: Selecting a Programming Language 680
21.7.4 `Recommendations.aspx`: Displaying Recommendations Based on Session Values 684
21.8 Case Study: Database-Driven ASP.NET Guestbook 685
21.8.1 Building a Web Form that Displays Data from a Database 687
21.8.2 Modifying the Code-Behind File for the `Guestbook` App 692
21.9 Case Study: ASP.NET AJAX 693
21.10 Case Study: Password-Protected Books Database App 694
21.11 Wrap-Up 694

22 XML and LINQ to XML 695

22.1 Introduction 696
22.2 XML Basics 696
22.3 Structuring Data 699
22.4 XML Namespaces 705
22.5 Document Type Definitions (DTDs) 708
22.6 W3C XML Schema Documents 711
22.7 Extensible Stylesheet Language and XSL Transformations 719
22.8 LINQ to XML: Document Object Model (DOM) 728

22.9 LINQ to XML Class Hierarchy 731
22.10 LINQ to XML: Namespaces and Creating Documents 740
22.11 XSLT with Class `XslCompiledTransform` 744
22.12 Wrap-Up 746

23 Windows 8 UI and XAML 747

23.1 Introduction 748
23.2 **Welcome** App: Introduction to XAML Declarative GUI Programming 749
23.2.1 Running the **Welcome** App 750
23.2.2 `MainPage.xaml` for the **Welcome** App 751
23.2.3 Creating the **Welcome** App's Project 753
23.2.4 Building the App's GUI 755
23.2.5 Overview of Important Project Files and Folders 756
23.2.6 Splash Screen and Logos 757
23.2.7 `Package.appmanifest` 758
23.3 **Painter** App: Layouts; Event Handling 758
23.3.1 General Layout Principles 759
23.3.2 `MainPage.xaml`: Layouts and Controls in the **Painter** App 760
23.3.3 Event Handling 765
23.4 **CoverViewer** App: Data Binding, Data Templates and Styles 771
23.4.1 `MainPage.xaml` for the **CoverViewer** App 772
23.4.2 Defining Styles and Data Templates 776
23.4.3 `MainPage.xaml.cs`: Binding Data to the `ListView`'s `ItemSource` 777
23.5 App Lifecycle 778
23.6 Wrap-Up 779

24 Windows 8 Graphics and Multimedia 780

24.1 Introduction 781
24.2 Basic Shapes 782
24.2.1 `Rectangles` 784
24.2.2 `Line` 784
24.2.3 `Ellipses` 784
24.3 `Polylines` and `Polygons` 784
24.4 `SolidColorBrushes` and `ImageBrushes` 786
24.5 `GradientBrushes` 788
24.5.1 Setting the `Rectangle`'s `Fill` to a Gradient 791
24.5.2 `GradientStops` 792
24.5.3 Defining the Gradient in the IDE 793
24.5.4 Code-Behind File 793
24.6 Transforms 794
24.6.1 `MainPage` Instance Variables and Constructor 797
24.6.2 `MainPage` Method `RotateAndDrawStars` 797
24.6.3 Applying a `RotateTransform` 797

24.7 Windows 8 Customization: A Television GUI 797
24.7.1 XAML for the TV GUI 798
24.7.2 XAML for the Power CheckBox's ControlTemplate 798
24.7.3 Creating a Basic ControlTemplate 800
24.7.4 Default ControlTemplate Markup 800
24.7.5 Defining the CheckBox's New Look-And-Feel 800
24.7.6 Using Animation to Change a Control's Look-and-Feel in Response to State Changes 801
24.7.7 XAML for the Play, Pause and Stop RadioButtons' ControlTemplate 802
24.7.8 XAML for the GUI 803
24.7.9 TV GUI Code-Behind File 805
24.8 Wrap-Up 807

25 Building a Windows Phone 8 App 808

25.1 Introduction 809
25.2 Downloading the Windows Phone 8 SDK 810
25.3 **Tip Calculator** App Introduction 811
25.4 Test-Driving the **Tip Calculator** App 812
25.5 **Tip Calculator** Technologies Overview 812
25.5.1 Classes App and PhoneApplicationPage 812
25.5.2 Arranging Controls with Grid 813
25.5.3 Controls 813
25.5.4 IDE Features 813
25.6 Building the App's GUI 813
25.6.1 Grid Introduction 813
25.6.2 Creating the TipCalculator Project 815
25.6.3 Changing the Default GUI 816
25.6.4 Adding the TextBlocks, Borders, a TextBox and a Slider 817
25.7 Adding Functionality to the App with C# 820
25.8 WMAppManifest.xml 825
25.9 Windows Phone Dev Center 825
25.9.1 Microsoft Account 826
25.9.2 Windows Phone Dev Center Account 826
25.9.3 Registering a Windows Phone Device for Development 826
25.10 Selling Your Apps in the Windows Phone Marketplace 827
25.10.1 Free vs. Paid Apps 827
25.10.2 Submitting Your Apps and In-App Products 827
25.10.3 Monetizing Apps with Microsoft Advertising pubCenter 828
25.11 Other Popular Mobile App Platforms 828
25.12 Developer Documentation 829
25.13 Additional Windows Phone 8 Resources 829
25.14 Wrap-Up 831

26 Asynchronous Programming with async and await 832

26.1 Introduction 833
26.2 Basics of async and await 834
26.3 Executing an Asynchronous Task from a GUI App 835
26.3.1 Performing a Task Asynchronously 835
26.3.2 Method calculateButton_Click 838
26.3.3 Task Method Run: Executing Asynchronously in a Separate Thread 838
26.3.4 awaiting the Result 838
26.3.5 Calculating the Next Fibonacci Value Synchronously 839
26.4 Sequential Execution of Two Compute-Intensive Tasks 839
26.5 Asynchronous Execution of Two Compute-Intensive Tasks 841
26.5.1 Method startButton_Click: awaiting Multiple Tasks with Task Method WhenAll 844
26.5.2 Method StartFibonacci 845
26.5.3 Method AppendText: Modifying a GUI from a Separate Thread 845
26.5.4 awaiting One of Several Tasks with Task Method WhenAny 845
26.6 Invoking a Flickr Web Service Asynchronously with WebClient 846
26.7 Wrap-Up 852

27 Web App Development with ASP.NET: A Deeper Look 854

27.1 Introduction 855
27.2 Case Study: Password-Protected Books Database App 855
27.2.1 Examining the ASP.NET Web Forms Application Template 856
27.2.2 Test-Driving the Completed App 859
27.2.3 Configuring the Website 861
27.2.4 Modifying the **Home** and **About** Pages 864
27.2.5 Creating a Content Page That Only Authenticated Users Can Access 866
27.2.6 Linking from the Default.aspx Page to the Books.aspx Page 866
27.2.7 Modifying the Master Page (Site.master) 866
27.2.8 Customizing the Password-Protected Books.aspx Page 867
27.3 ASP.NET Ajax 871
27.3.1 Traditional Web Apps 871
27.3.2 Ajax Web Apps 872
27.3.3 Testing an ASP.NET Ajax App 873
27.3.4 The ASP.NET Ajax Control Toolkit 874
27.3.5 Using Controls from the Ajax Control Toolkit 875
27.3.6 ToolkitScriptManager 875
27.3.7 Grouping Information in Tabs Using the TabContainer Control 875
27.3.8 Partial-Page Updates Using the UpdatePanel Control 876
27.3.9 Adding Ajax Functionality to ASP.NET Validation Controls Using Ajax Extenders 876

27.3.10 Changing the Display Property of the Validation Controls 877
27.3.11 Running the App 877
27.4 Wrap-Up 877

28 Web Services 879

28.1 Introduction 880
28.2 WCF Services Basics 881
28.3 HTTP `get` and `post` Requests 881
28.4 Representational State Transfer (REST) 882
28.5 JavaScript Object Notation (JSON) 882
28.6 Publishing and Consuming REST-Based XML Web Services 883
28.6.1 WCF Web Service Project 883
28.6.2 Implementing a REST-Based XML WCF Web Service 883
28.6.3 Building a REST WCF Web Service 884
28.6.4 Deploying the `WelcomeRESTXMLService` 886
28.6.5 Consuming a REST-Based XML WCF Web Service 888
28.7 Publishing and Consuming REST-Based JSON Web Services 889
28.7.1 Creating a REST-Based JSON WCF Web Service 889
28.7.2 Consuming a REST-Based JSON WCF Web Service 891
28.8 Equation Generator: Returning User-Defined Types 893
28.8.1 Creating the REST-Based XML `EquationGenerator` Web Service 895
28.8.2 Consuming the REST-Based XML `EquationGenerator` Web Service 897
28.8.3 Creating the REST-Based JSON WCF `EquationGenerator` Web Service 900
28.8.4 Consuming the REST-Based JSON WCF `EquationGenerator` Web Service 901
28.9 Wrap-Up 905

29 Building a Windows Azure™ Cloud Computing App 906

29.1 Introduction 907
29.2 Installing the Windows Azure SDK for Visual Studio 2012 909
29.3 Windows Azure Cloud Services Accounts 909
29.3.1 Signing Up for a Windows Azure Cloud Services Account 910
29.3.2 Windows Azure Educator Grant 910
29.3.3 Windows Azure for MSDN Subscribers 910
29.4 **Favorite Twitter Searches**: Introduction 910
29.5 **Favorite Twitter Searches**: Test-Drive 911
29.6 **Favorite Twitter Searches**: Technologies Overview 914
29.7 **Favorite Twitter Searches**: Code 915
29.7.1 `TableEntity` for Storing Data in Windows Azure Table Storage 915
29.7.2 Storing and Retrieving `TableEntity` Objects 916
29.8 Security, Privacy and Reliability 921

29.9 Microsoft Windows Azure Resources 921
29.10 Microsoft Windows Azure Code Samples 923
29.11 Additional Web Resources 924
29.12 Wrap-Up 926

30 GUI with Windows Presentation Foundation 927

30.1 Introduction 928
30.2 Windows Presentation Foundation (WPF) 928
30.3 Declarative GUI Programming Using XAML 929
30.4 Creating a WPF App 931
30.5 Laying Out Controls 932
30.5.1 General Layout Principles 933
30.5.2 Layout in Action 934
30.6 Event Handling 938
30.7 Commands and Common Application Tasks 946
30.8 WPF GUI Customization 951
30.9 Using Styles to Change the Appearance of Controls 951
30.10 Customizing `Windows` 957
30.11 Defining a Control's Appearance with Control Templates 960
30.12 Data-Driven GUIs with Data Binding 965
30.13 Wrap-Up 971

31 WPF Graphics and Multimedia 972

31.1 Introduction 973
31.2 Controlling Fonts 973
31.3 Basic Shapes 975
31.4 Polygons and Polylines 977
31.5 Brushes 980
31.6 Transforms 987
31.7 WPF Customization: A Television GUI 989
31.8 Animations 999
31.9 Speech Synthesis and Speech Recognition 1001
31.10 Wrap-Up 1008

32 ATM Case Study, Part 1: Object-Oriented Design with the UML 1009

32.1 Introduction 1010
32.2 Examining the ATM Requirements Document 1010
32.3 Identifying the Classes in the ATM Requirements Document 1018
32.4 Identifying Class Attributes 1025
32.5 Identifying Objects' States and Activities 1029
32.6 Identifying Class Operations 1033

32.7 Identifying Collaboration Among Objects 1040
32.8 Wrap-Up 1047

33 ATM Case Study, Part 2: Implementing an Object-Oriented Design 1052

33.1 Introduction 1053
33.2 Starting to Program the Classes of the ATM System 1053
33.3 Incorporating Inheritance and Polymorphism into the ATM System 1058
33.4 ATM Case Study Implementation 1065
33.4.1 Class `ATM` 1066
33.4.2 Class `Screen` 1071
33.4.3 Class `Keypad` 1072
33.4.4 Class `CashDispenser` 1073
33.4.5 Class `DepositSlot` 1074
33.4.6 Class `Account` 1075
33.4.7 Class `BankDatabase` 1077
33.4.8 Class `Transaction` 1080
33.4.9 Class `BalanceInquiry` 1081
33.4.10 Class `Withdrawal` 1082
33.4.11 Class `Deposit` 1086
33.4.12 Class `ATMCaseStudy` 1089
33.5 Wrap-Up 1089

A Operator Precedence Chart 1092

B Simple Types 1094

C ASCII Character Set 1096

D Number Systems 1097

D.1 Introduction 1098
D.2 Abbreviating Binary Numbers as Octal and Hexadecimal Numbers 1101
D.3 Converting Octal and Hexadecimal Numbers to Binary Numbers 1102
D.4 Converting from Binary, Octal or Hexadecimal to Decimal 1102
D.5 Converting from Decimal to Binary, Octal or Hexadecimal 1103
D.6 Negative Binary Numbers: Two's Complement Notation 1105

E UML 2: Additional Diagram Types 1107

E.1 Introduction 1107
E.2 Additional Diagram Types 1107

F Unicode® 1109
F.1 Introduction 1110
F.2 Unicode Transformation Formats 1111
F.3 Characters and Glyphs 1112
F.4 Advantages/Disadvantages of Unicode 1112
F.5 Using Unicode 1113
F.6 Character Ranges 1115

Index 1117

Preface

"Live in fragments no longer, only connect."
—Edgar Morgan Forster

Welcome to Visual C#® 2012 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.

We focus on software engineering best practices. At the heart of the book is the Deitel signature "live-code approach"—concepts are presented in the context of complete working programs, rather than in code snippets. Each complete code example is accompanied by live sample executions. All the source code is available at

```
www.deitel.com/books/cs2012fp/
```

If you have questions as you read the book, we're easy to reach at `deitel@deitel.com`—we'll respond promptly. For book updates, visit `www.deitel.com/books/cs2012fp`, join our social media communities on Facebook (`www.deitel.com/DeitelFan`), Twitter (`@deitel`), Google+ (`gplus.to/deitel`) and LinkedIn (`bit.ly/DeitelLinkedIn`), and subscribe to the *Deitel® Buzz Online* newsletter (`www.deitel.com/newsletter/subscribe.html`).

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

The new Visual C# 2012 and its associated technologies motivated us to write *C# 2012 for Programmers, 5/e*. 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 1–22 and 26–33 were tested on *both* Windows 7 and Windows 8. The code examples for the Windows-8-specific chapters—Chapter 23 (Windows 8 UI and XAML), Chapter 24 (Windows 8 Graphics and Multimedia) and Chapter 25 (Building a Windows Phone 8 App)—were tested *only* on Windows 8, because Visual Studio Express 2012 for Windows 8 and Visual Studio Express 2012 for Windows Phone run only on Windows 8.
- ***C# and Visual C#.*** The C# language has been standardized internationally by ECMA and ISO (the standards document is available free of charge at `bit.ly/ECMA334`). This enables other implementations of the language besides Microsoft's Visual C#, such as Mono (`www.mono-project.com`), which runs on Linux systems, iOS (for Apple's iPhone, iPad and iPod Touch), Google's Android and Windows.

- *Modular multi-GUI treatment with Windows Forms, Windows 8 UI and WPF.* The book features three different GUI treatments, starting with Windows Forms GUI; later chapters contain treatments of the new Windows 8 UI (user interface) and WPF GUI. Windows 8 UI apps are called *Windows Store apps*. In Chapter 23, you'll learn how to create and test Windows Store apps.

- *Modular treatment of graphics and multimedia with Windows 8 and WPF.* The book features chapters on both the new Windows 8 Graphics and Multimedia (Chapter 24) and WPF Graphics and Multimedia (Chapter 31).

- *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.

- *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 20–21 and 27 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 21, 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 smartphone operating system. It features multi-touch support for touchpads and touchscreen devices, enhanced security features and more. In Chapter 25, you'll build a complete working Windows Phone 8 app and test it on the Windows Phone emulator; we 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. Chapter 29 shows you how to build a Windows Azure app that can store data in the cloud. You'll test your app on the Windows Azure Storage Emulator.

- *Asynchronous programming with* **`async`** *and* **`await`**. Asynchronous programming is simplified in C# 2012 with the new *`async`* and *`await`* capabilities. We introduce asynchronous programming with `async` and `await` in Chapter 26. To take advantage of multicore architecture you need to write applications that can process tasks asynchronously. Asynchronous programming is a technique for writing apps containing tasks that can execute asynchronously, which can improve app performance and GUI responsiveness in apps with long-running or compute-intensive tasks.

Object-Oriented Programming

- *Early-objects approach.* The basic concepts and terminology of object technology are introduced in Chapter 1. In Chapter 2, Dive Into® Visual Studio 2012 Express for Windows Desktop, you'll *visually* manipulate objects, such as labels and images. In Chapter 3, Introduction to C# Apps, you'll write C# *program code* that manipulates *existing* objects. You'll develop your first *customized* classes and objects in Chapter 4.
- ***A clear, example-driven presentation of classes, objects, inheritance, polymorphism and interfaces.***
- ***Case study: Using the UML to develop an object-oriented design and C# 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. Chapters 32 and 33 include a 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 C# implementation. Readers often report a "light bulb moment"—the case study helps them "tie it all together" and understand object orientation more deeply.
- ***Multiple programming paradigms.*** We discuss *structured programming*, *object-oriented programming*, *generic programming* and some *functional programming*.

Other Features

- ***We use LINQ to query files, databases, XML and collections.*** The introductory LINQ to Objects chapter (Chapter 9) will get you started using LINQ technology early. Later in the book, we take a deeper look, using LINQ to Entities (Chapters 20–21 and 27) and LINQ to XML (Chapters 22 and 26).
- ***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.
- ***Files and strings.***
- ***Generics and collections.***
- ***Integrated exception handling.*** We introduce exception handling early (Chapter 8, Arrays) to watch for attempts to access array elements outside the array's bounds. Chapter 10, Classes and Objects: A Deeper Look, shows how to in-

dicate an exception when a member function receives an invalid argument. We cover the complete details of exception handling in Chapter 13, Exception Handling: A Deeper Look.

- ***C# XML capabilities.*** Extensible Markup Language (XML) is pervasive in the software-development industry and throughout the .NET platform. In Chapter 22, 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 Chapters 23–24 and 30–31.
- ***Web app development with ASP.NET 4.5 and ASP.NET Ajax.*** Chapter 27 extends Chapter 21's ASP.NET discussion with a case study on building a password-protected, web-based bookstore app. Also, we introduce in Chapter 27 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.
- ***Windows Communication Foundation (WCF) web services.*** Web services enable you to package app functionality in a manner that turns the web into a library of *reusable* services. Chapter 28 includes a case study on building a math question generator web service that's called by a math tutor app.
- ***WPF (Windows Presentation Foundation) GUI, graphics and multimedia.*** Chapters 30–31 provide an introduction to Windows Presentation Foundation (WPF)—a XAML-based Microsoft framework that preceded Windows 8 UI and integrates GUI, graphics and multimedia capabilities. WPF was designed as a replacement for Windows Forms GUI technologies. We implement a painting app, a text editor, a color chooser, a book-cover viewer, a television video player, various animations, and speech synthesis and recognition apps.

Training Approach

C# 2012 for Programmers, 5/e stresses program clarity and concentrates on building well-engineered software.

Live-Code Approach. The book includes hundreds of "live-code" examples—each new concept is presented in the context of a complete working C# app that is immediately followed by one or more actual executions showing the program's inputs and outputs. We include a broad range of example programs selected from computer science, business, simulation, game playing, graphics, multimedia and many other areas.

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

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

Code Highlighting. We place gray rectangles around each program's key code segments.

Using Fonts for Emphasis. We place the key terms and the index's page reference for each defining occurrence in ***bold italic*** text for easier reference. We emphasize on-screen components in the **bold Helvetica** font (e.g., the **File** menu) and emphasize C# program text in the `Lucida` font (e.g., `int x = 5`).

Web Access. All of the source-code examples can be downloaded from:

```
www.deitel.com/books/cs2012fp
```

Objectives. Each chapter begins with a list of chapter objectives.

Programming Tips. The book includes hundreds of programming tips and practices that represent the best we've gleaned from a combined eight 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 of the tips describe aspects of C# that prevent bugs from getting into programs in the first place.

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.

Obtaining the Software Used in This Book

We wrote the code examples in *C# 2012 for Programmers, 5/e* using Microsoft's *free* Visual Studio Express 2012 products, including:

- Visual Studio Express 2012 for Windows Desktop (Chapters 1–20, 22, 26, 29 and 30–33), which includes Visual C# and other Microsoft development tools. This runs on Windows 7 *and* 8.
- Visual Studio Express 2012 for Web (Chapters 21 and 27–28
- Visual Studio Express 2012 for Windows 8 (Chapters 23–24)
- Visual Studio Express 2012 for Windows Phone (Chapter 25)

Each of these is available for download at

```
www.microsoft.com/visualstudio/eng/products/
   visual-studio-express-products
```

C# 2012 Fundamentals: Parts I, II, III and IV, Second Edition LiveLessons Video Training

Our *C# 2012 Fundamentals: Parts I, II, III and IV* LiveLessons video training shows you what you need to know to start building robust, powerful software with C# 2012. It includes approximately 40 hours of expert training synchronized with *C# 2012 for Programmers, 5/e*. For additional information about Deitel LiveLessons video products available on Safari Books Online and other electronic channels, visit

```
www.deitel.com/livelessons
```

or contact us at `deitel@deitel.com`.

Acknowledgments

We'd like to thank Abbey Deitel and Barbara Deitel of Deitel & Associates, Inc. for long hours devoted to this project. Abbey co-authored this Preface and Chapter 1 and she and Barbara painstakingly researched the new capabilities of Visual C# 2012, .NET 4.5, Windows 8, Windows Phone 8, Windows Azure and other key topics.

We're fortunate to have worked on this project with the dedicated publishing professionals at Prentice Hall/Pearson. We appreciate the extraordinary efforts and mentorship of our friend and professional colleague Mark L. Taub, Editor-in-Chief of Pearson Technology Group. Carole Snyder did a great job recruiting distinguished members of the C# community to review the manuscript and managing the review process. Chuti Prasertsith designed the cover with creativity and precision. John Fuller does a superb job managing the production of all of our Deitel Developer Series books and LiveLessons video products.

Reviewers

We wish to acknowledge the efforts of the reviewers whose constructive criticisms helped us shape the recent editions of this content. They scrutinized the text and the programs and provided countless suggestions for improving the presentation: Shay Friedman (Microsoft Visual C# MVP), Octavio Hernandez (Microsoft Certified Solutions Developer), Stephen Hustedde (South Mountain College), José Antonio González Seco (Parliament of Andalusia, Spain), Shawn Weisfeld (Microsoft MVP and President and Founder of UserGroup.tv), Huanhui Hu (Microsoft Corporation), Narges Kasiri (Oklahoma State University), Charles Liu (University of Texas at San Antonio), Dr. Hamid R. Nemati

(The University of North Carolina at Greensboro), Jeffrey P. Scott (Blackhawk Technical College), 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*), James Edward Keysor (Florida Institute of Technology), Helena Kotas (Microsoft), 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.), Timothy Ng (Microsoft), Akira Onishi (Microsoft), Joe Stagner (Senior Program Manager, Developer Tools & Platforms), Erick Thompson (Microsoft), Jesús Ubaldo Quevedo-Torrero (University of Wisconsin–Parkside, Department of Computer Science) and Zijiang Yang (Western Michigan University).

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

```
    deitel@deitel.com
```

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

Paul Deitel
Harvey Deitel

About the Authors

Paul Deitel, CEO and Chief Technical Officer of Deitel & Associates, Inc., is a graduate of MIT, where he studied Information Technology. Through Deitel & Associates, Inc., he has delivered hundreds of programming courses to industry clients, including Cisco, IBM, Siemens, Sun Microsystems, Dell, Fidelity, NASA at the Kennedy Space Center, the National Severe Storm Laboratory, White Sands Missile Range, Rogue Wave Software, Boeing, SunGard Higher Education, Nortel Networks, Puma, iRobot, Invensys and many more. He and his co-author, Dr. Harvey M. Deitel, are the world's best-selling programming-language textbook/professional book/video authors.

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

2012/2013 C# MVP

Dr. Harvey Deitel, Chairman and Chief Strategy Officer of Deitel & Associates, Inc., has 52 years of experience in the computer field. Dr. Deitel earned B.S. and M.S. degrees in Electrical Engineering from MIT and a Ph.D. in Mathematics from Boston University, all with an emphasis in Computer Science. In the 1960s, through Advanced Computer Techniques and Computer Usage Corporation, he worked on the teams building various IBM operating systems. In the 1970s, he built commercial software systems and more recently committed to a career in Computer Science education. 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 Corporate 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 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 C#®, Visual Basic®, C++, Visual C++®, C, Java™, XML®, Python®, object technology, Internet and web programming, Android app development, Objective-C and iOS 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 professional books, college textbooks 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
```

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 book uses the following software:

- Microsoft Visual Studio Express 2012 for Windows Desktop
- Microsoft Visual Studio Express 2012 for Web (Chapters 21 and 27–28)
- Microsoft Visual Studio Express 2012 for Windows 8 (Chapters 23–24)
- Microsoft Visual Studio Express 2012 for Windows Phone (Chapter 25)

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 *C# 2012 for Programmers, 5/e* 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 *C# 2012 for Programmers, 5/e* are available for download at

```
www.deitel.com/books/cs2012fp/
```

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/cs2012fp/`. 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 C# 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 C# studies with *C# 2012 for Programmers, 5/e*. We hope you enjoy the book!

This page intentionally left blank

3

Introduction to C# Apps

Objectives

In this chapter you'll:

- Input data from the keyboard and output data to the screen.
- Declare and use data of various types.
- Use arithmetic operators.
- Write decision-making statements.
- Use relational and equality operators.

Outline

3.1 Introduction
3.2 A Simple C# App: Displaying a Line of Text
3.3 Creating a Simple App in Visual Studio
3.4 Modifying Your Simple C# App
3.5 Formatting Text with `Console.Write` and `Console.WriteLine`
3.6 Another C# App: Adding Integers
3.7 Arithmetic
3.8 Decision Making: Equality and Relational Operators
3.9 Wrap-Up

3.1 Introduction

We now introduce C# app programming. Most of the C# apps you'll study in this book process information and display results. In this chapter, we introduce **console apps**—these input and output text in a *console window*, which in Windows is known as the **Command Prompt**.

We begin with several examples that simply display messages on the screen. We then demonstrate an app that obtains two numbers from a user, calculates their sum and displays the result. You'll perform various arithmetic calculations and save the results for later use. Many apps contain logic that makes *decisions*—the last example in this chapter demonstrates decision-making fundamentals by comparing numbers and then displaying messages based on the comparison results. For example, the app displays a message indicating that two numbers are equal only if they have the same value.

3.2 A Simple C# App: Displaying a Line of Text

Let's consider a simple app that displays a line of text. The app and its output are shown in Fig. 3.1, which illustrates several important C# language features. Each program we present in this book includes line numbers, which are *not* part of actual C# code. In the Before You Begin section that follows the Preface, we show how to display line numbers for your C# code. We'll soon see that line 10 does the real work of the app—namely, displaying the phrase `Welcome to C# Programming!` on the screen. Let's now do a code walkthrough of the app.

Comments

Line 1

```
    // Fig. 3.1: Welcome1.cs
```

begins with `//`, indicating that the remainder of the line is a **comment**. You'll insert comments to document your apps and improve their readability. The C# compiler ignores comments, so they do *not* cause the computer to perform any action when the app is run. We begin every app with a comment indicating the figure number and the name of the file in which the app is stored.

A comment that begins with `//` is called a **single-line comment**, because it terminates at the end of the line on which it appears. A `//` comment also can begin in the middle of a line and continue until the end of that line (as in lines 7, 11 and 12).

```
 1  // Fig. 3.1: Welcome1.cs
 2  // Text-displaying app.
 3  using System;
 4
 5  public class Welcome1
 6  {
 7     // Main method begins execution of C# app
 8     public static void Main( string[] args )
 9     {
10        Console.WriteLine( "Welcome to C# Programming!" );
11     } // end Main
12  } // end class Welcome1
```

```
Welcome to C# Programming!
```

Fig. 3.1 | Text-displaying app.

Delimited comments such as

```
/* This is a delimited comment.
   It can be split over many lines */
```

can be split over several lines. This type of comment begins with the delimiter `/*` and ends with the delimiter `*/`. All text between the delimiters is ignored by the compiler.

Common Programming Error 3.1

Forgetting one of the delimiters of a delimited comment is a syntax error.

Line 2

```
// Text-displaying app.
```

is a single-line comment that describes the purpose of the app.

`using` Directive

Line 3

```
using System;
```

is a **`using` directive** that tells the compiler where to look for a class that's used in this app. A great strength of Visual C# is its rich set of predefined classes that you can *reuse* rather than "reinventing the wheel." These classes are organized under **namespaces**—named collections of related classes. Collectively, .NET's namespaces are referred to as the **.NET Framework Class Library**. Each `using` directive identifies a namespace containing predefined classes that a C# app should be able to use. The `using` directive in line 3 indicates that this example intends to use classes from the `System` namespace, which contains the predefined `Console` class (discussed shortly) used in line 10, and many other useful classes.

Error-Prevention Tip 3.1

Forgetting to include a `using` directive for a namespace that contains a class used in your app typically results in a compilation error, containing a message such as "`The name 'Console' does not exist in the current context.`"

For each new .NET class we use, we indicate the namespace in which it's located. This information is important, because it helps you locate descriptions of each class in the **.NET documentation**. A web-based version of this documentation can be found at

```
msdn.microsoft.com/en-us/library/ms229335.aspx
```

This can also be accessed via the **Help** menu. You can click the name of any .NET class or method, then press the *F1* key to get more information. Finally, you can learn about the contents of a given namespace by going to

```
msdn.microsoft.com/namespace
```

So, `msdn.microsoft.com/System` takes you to namespace `System`'s documentation.

Blank Lines and Whitespace

Line 4 is simply a *blank line*. Blank lines and space characters make code easier to read, and together with tab characters are known as **whitespace**. Space characters and tabs are known specifically as **whitespace characters**. Whitespace is ignored by the compiler.

Class Declaration

Line 5

```
public class Welcome1
```

begins a **class declaration** for the class `Welcome1`. Every app consists of at least one class declaration that's defined by you. These are known as **user-defined classes**. The **`class` keyword** introduces a class declaration and is immediately followed by the **class name** (`Welcome1`). Keywords (sometimes called **reserved words**) are reserved for use by C# and are always spelled with all lowercase letters. The complete list of C# keywords is shown in Fig. 3.2.

C# Keywords and contextual keywords				
abstract	as	base	bool	break
byte	case	catch	char	checked
class	const	continue	decimal	default
delegate	do	double	else	enum
event	explicit	extern	false	finally
fixed	float	for	foreach	goto
if	implicit	in	int	interface
internal	is	lock	long	namespace
new	null	object	operator	out
override	params	private	protected	public
readonly	ref	return	sbyte	sealed
short	sizeof	stackalloc	static	string
struct	switch	this	throw	true
try	typeof	uint	ulong	unchecked
unsafe	ushort	using	virtual	void
volatile	while			

Fig. 3.2 | C# keywords and contextual keywords. (Part 1 of 2.)

C# Keywords and contextual keywords

Contextual Keywords

`add`	`alias`	`ascending`	`async`	`await`
`by`	`descending`	`dynamic`	`equals`	`from`
`get`	`global`	`group`	`into`	`join`
`let`	`on`	`orderby`	`partial`	`remove`
`select`	`set`	`value`	`var`	`where`
`yield`				

Fig. 3.2 | C# keywords and contextual keywords. (Part 2 of 2.)

Class Name Convention

By convention, all class names begin with a capital letter and capitalize the first letter of each word they include (e.g., `SampleClassName`). This convention is known as **upper camel casing**. A class name is an **identifier**—a series of characters consisting of letters, digits and underscores (_) that does not begin with a digit and does not contain spaces. Some valid identifiers are `Welcome1`, `identifier`, `_value` and `m_inputField1`. The name `7button` is *not* a valid identifier because it begins with a digit, and the name `input field` is *not* a valid identifier because it contains a space. Normally, an identifier that does not begin with a capital letter is not the name of a class. C# is **case sensitive**—that is, uppercase and lowercase letters are distinct, so `a1` and `A1` are different (but both valid) identifiers.[1]

Good Programming Practice 3.1

By convention, always begin a class name's identifier with a capital letter and start each subsequent word in the identifier with a capital letter.

Common Programming Error 3.2

C# is case sensitive. Not using the proper uppercase and lowercase letters for an identifier normally causes a compilation error.

`public` *Class*

In Chapters 3–9, every class we define begins with the keyword **`public`**. For now, we'll simply require this keyword. You'll learn more about classes in Chapter 10. When you save your `public` class declaration in a file, the file name is usually the class name followed by the `.cs` file-name extension. For our app, the file name is `Welcome1.cs`.

Good Programming Practice 3.2

By convention, a file that contains a single `public` class should have a name that's identical to the class name (plus the `.cs` extension) in both spelling and capitalization.

1. Identifiers may also be preceded by the `@` character. This indicates that a word should be interpreted as an identifier, even if it's a keyword (e.g., `@int`). This allows C# code to use code written in other .NET languages where an identifier might have the same name as a C# keyword. The **contextual keywords** in Fig. 3.2 can be used as identifiers outside the contexts in which they're keywords, but for clarity this is not recommended.

Body of a Class Declaration

A **left brace** (in line 6 in Fig. 3.1), `{`, begins the **body** of every class declaration. A corresponding **right brace** (in line 12), `}`, must end each class declaration. Lines 7–11 are indented. This indentation is a *spacing convention*. We define each spacing convention as a *Good Programming Practice*.

Good Programming Practice 3.3

Indent the entire body of each class declaration one "level" of indentation between the left and right braces that delimit the body of the class. This format emphasizes the class declaration's structure and makes it easier to read. You can let the IDE format your code by selecting **Edit > Advanced > Format Document**.

Common Programming Error 3.3

It's a syntax error if braces do not occur in matching pairs.

`Main` Method

Line 7

```
    // Main method begins execution of C# app
```

is a comment indicating the purpose of lines 8–11 of the app. Line 8

```
    public static void Main( string[] args )
```

is the starting point of every app. The **parentheses** after the identifier `Main` indicate that it's an app building block called a **method**. Class declarations normally contain one or more methods. Method names usually follow the same capitalization conventions used for class names. For each app, one of the methods in a class *must* be called `Main` (which is typically defined as shown in line 8); otherwise, the app will not execute. Methods are able to perform tasks and return information when they complete their tasks. Keyword **`void`** (line 8) indicates that this method will *not* return any information after it completes its task. Later, we'll see that many methods do return information. You'll learn more about methods in Chapters 4 and 7. We discuss the contents of `Main`'s parentheses in Chapter 8. For now, simply mimic `Main`'s first line in your apps.

Body of a Method Declaration

The left brace in line 9 begins the **body of the method declaration**. A corresponding right brace must end the method's body (line 11). Line 10 in the body of the method is indented between the braces.

Displaying a Line of Text

Line 10

```
    Console.WriteLine( "Welcome to C# Programming!" );
```

instructs the computer to **perform an action**—namely, to display the **string** of characters between the double quotation marks, which *delimit* the string. A string is sometimes called a **character string**, a **message** or a **string literal**. We refer to them simply as strings. Whitespace characters in strings are *not* ignored by the compiler.

Class **`Console`** provides **standard input/output** capabilities that enable apps to read and display text in the console window from which the app executes. The **`Console.WriteLine` method** displays a line of text in the console window. The string in the parentheses in line 10 is the **argument** to the method. Method `Console.WriteLine` performs its task by displaying its argument in the console window. When `Console.WriteLine` completes its task, it positions the **screen cursor** (the blinking symbol indicating where the next character will be displayed) at the beginning of the next line in the console window. This movement of the cursor is similar to what happens when a user presses the *Enter* key while typing in a text editor—the cursor moves to the beginning of the next line in the file.

Statements

The entire line 10, including `Console.WriteLine`, the parentheses, the argument `"Welcome to C# Programming!"` in the parentheses and the **semicolon** (`;`), is called a **statement**. Most statements end with a semicolon. When the statement in line 10 executes, it displays the message `Welcome to C# Programming!` in the console window. A method is typically composed of one or more statements that perform the method's task.

Matching Left (`{`) and Right (`}`) Braces

You may find it difficult when reading or writing an app to match the left and right braces (`{` and `}`) that delimit the body of a class declaration or a method declaration. To help, you can include a comment after each closing right brace (`}`) that ends a method declaration and after each closing right brace that ends a class declaration. For example, line 11

```
      } // end Main
```

specifies the closing right brace of method `Main`, and line 12

```
   } // end class Welcome1
```

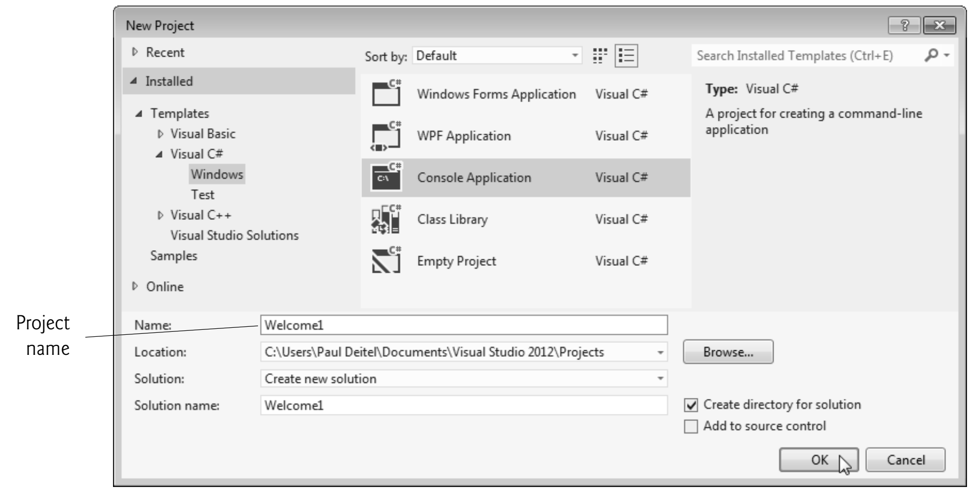
specifies the closing right brace of class `Welcome1`. Each of these comments indicates the method or class that the right brace terminates. Visual Studio can help you locate matching braces in your code. Simply place the cursor immediately in front of the left brace or immediately after the right brace, and Visual Studio will highlight both.

3.3 Creating a Simple App in Visual Studio

Now that we've presented our first console app (Fig. 3.1), we provide a step-by-step explanation of how to create, compile and execute it using Visual Studio 2012 Express for Windows Desktop, which we'll refer to simply as Visual Studio from this point forward.

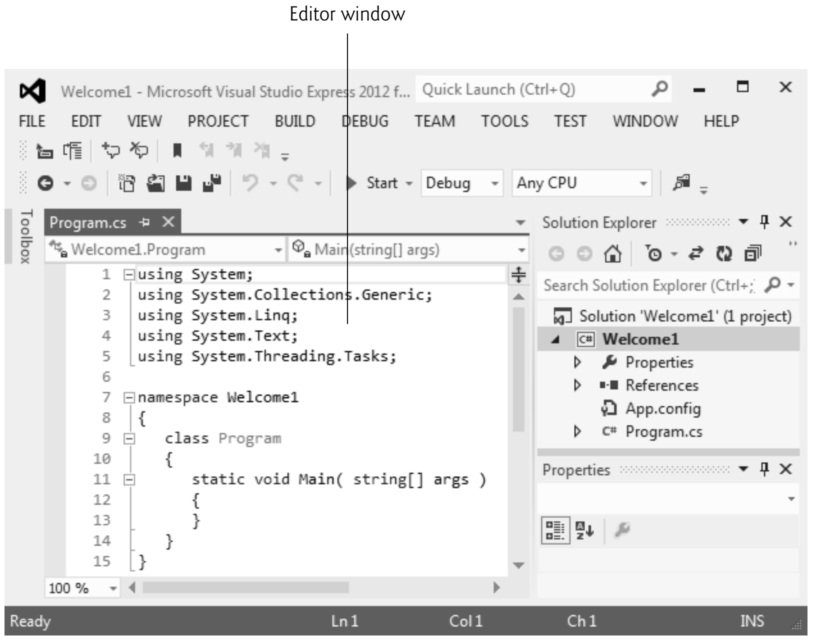
Creating the Console App

After opening Visual Studio, select **FILE > New Project…** to display the **New Project** dialog (Fig. 3.3). At the left side of the dialog, under **Installed > Templates > Visual C#** select the **Windows** category, then in the middle of the dialog select the **Console Application** template. In the dialog's **Name** field, type `Welcome1`, then click **OK** to create the project. By default, the project's folder will be placed in your account's `Documents` folder under `Visual Studio 2012\Projects`. The IDE now contains the open console app, as shown in Fig. 3.4. The editor window already contains some code provided by the IDE. Some of



Name: Welcome1
Location: C:\Users\Paul Deitel\Documents\Visual Studio 2012\Projects
Solution: Create new solution
Solution name: Welcome1
☑ Create directory for solution
☐ Add to source control

Fig. 3.3 | Creating a **Console Application** with the **New Project** dialog.



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Welcome1
{
    class Program
    {
        static void Main( string[] args )
        {
        }
    }
}
```

Fig. 3.4 | IDE with an open console app.

this code is similar to that of Fig. 3.1. Some is not, and uses features that we have not yet discussed. The IDE inserts this extra code to help organize the app and to provide access to some common classes in the .NET Framework Class Library—at this point in the book, this code is neither required nor relevant to the discussion of this app; delete all of it.

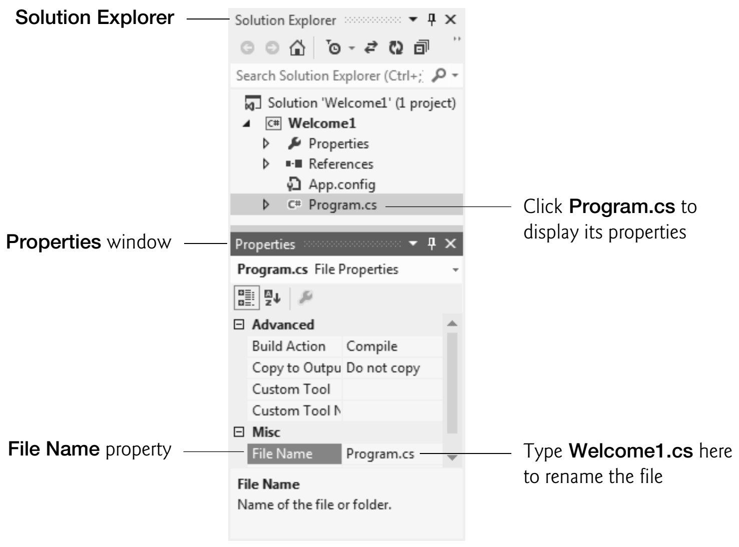
The code coloring scheme used by the IDE is called **syntax-color highlighting** and helps you visually differentiate app elements. For example, keywords appear in blue and comments appear in green. We syntax-shade our code similarly—bold for keywords, gray for comments, bold gray for literals and constants, and black for other text. One example of a literal is the string passed to `Console.WriteLine` in line 10 of Fig. 3.1. You can customize the colors shown in the code editor by selecting **Tools > Options…**. This displays the **Options** dialog. Then expand the **Environment** node and select **Fonts and Colors**. Here you can change the colors for various code elements.

Configuring the Editor Window

Visual Studio provides many ways to personalize your coding experience. In the Before You Begin section that follows the Preface, we show how to configure the IDE to display line numbers at the left side of the editor window and how to specify indent sizes that match our code examples.

Changing the Name of the App File

For the apps we create in this book, we change the default name of the source-code file (i.e., `Program.cs`) to a more descriptive name. To rename the file, click `Program.cs` in the **Solution Explorer** window. This displays the app file's properties in the **Properties** window (Fig. 3.5). Change the **File Name property** to `Welcome1.cs` and press *Enter*.


Solution Explorer
Properties window
File Name property
Click Program.cs to display its properties
Type Welcome1.cs here to rename the file
Solution Explorer
Search Solution Explorer (Ctrl+;
Solution 'Welcome1' (1 project)
Welcome1
Properties
References
App.config
Program.cs
Properties
Program.cs File Properties

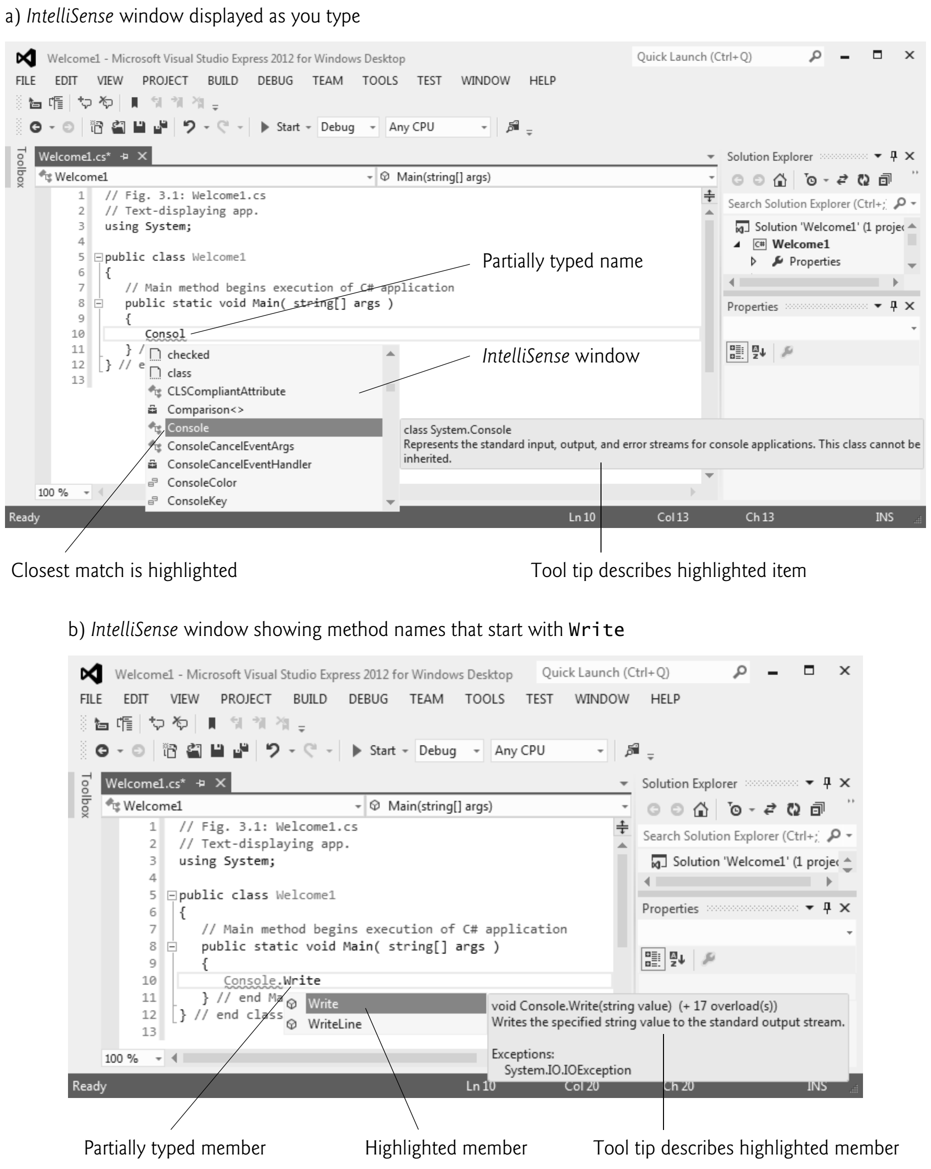
Advanced
Build Action Compile
Copy to Outpu Do not copy
Custom Tool
Custom Tool N
Misc
File Name Program.cs

File Name
Name of the file or folder.

Fig. 3.5 | Renaming the program file in the **Properties** window.

Writing Code and Using IntelliSense

In the editor (Fig. 3.4), replace the IDE-generated code with the code in Fig. 3.1. As you begin typing `Console` (line 10), an ***IntelliSense*** window is displayed (Fig. 3.6(a)). As you type, *IntelliSense* lists various items that start with or contain the letters you've typed so far. *IntelliSense* also displays a tool tip containing a description of the first matching item. You can


a) *IntelliSense* window displayed as you type
Welcome1 - Microsoft Visual Studio Express 2012 for Windows Desktop
Quick Launch (Ctrl+Q)
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST WINDOW HELP
Start Debug Any CPU
Welcome1.cs*
Welcome1
Main(string[] args)
Solution Explorer
Search Solution Explorer (Ctrl+;
Solution 'Welcome1' (1 projec
Welcome1
Properties
checked
class
CLSCompliantAttribute
Comparison<>
Console
ConsoleCancelEventArgs
ConsoleCancelEventHandler
ConsoleColor
ConsoleKey
100 %
Ready
Ln 10
Col 13
Ch 13
INS
Partially typed name
IntelliSense window
Closest match is highlighted
Tool tip describes highlighted item
b) *IntelliSense* window showing method names that start with `Write`
Write
WriteLine
Ln 10
Col 20
Ch 20
Partially typed member
Highlighted member
Tool tip describes highlighted member

```
// Fig. 3.1: Welcome1.cs
// Text-displaying app.
using System;

public class Welcome1
{
   // Main method begins execution of C# application
   public static void Main( string[] args )
   {
      Consol
```

class System.Console
Represents the standard input, output, and error streams for console applications. This class cannot be inherited.

```
// Fig. 3.1: Welcome1.cs
// Text-displaying app.
using System;

public class Welcome1
{
   // Main method begins execution of C# application
   public static void Main( string[] args )
   {
      Console.Write
   } // end Main
} // end class
```

void Console.Write(string value) (+ 17 overload(s))
Writes the specified string value to the standard output stream.

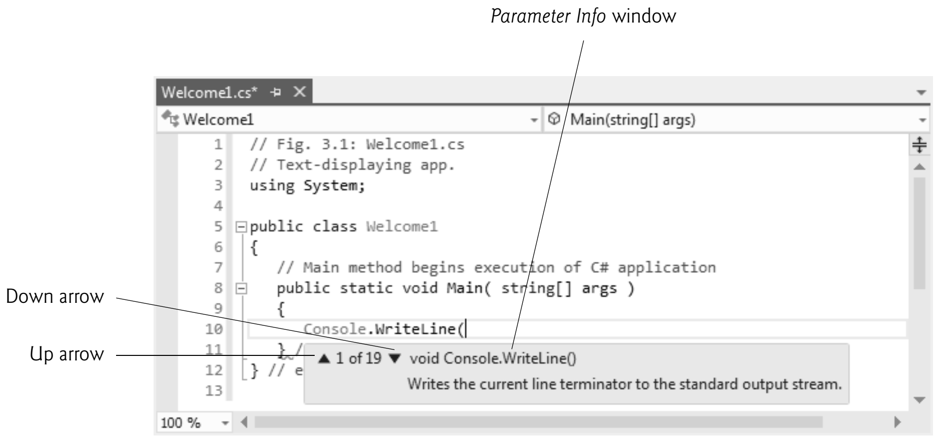
Exceptions:
System.IO.IOException

Fig. 3.6 | *IntelliSense*.

either type the complete item name (e.g., `Console`), double click the item name in the member list or press the *Tab* key to complete the name. Once the complete name is provided, the *IntelliSense* window closes. While the *IntelliSense* window is displayed, pressing the *Ctrl* key makes the window transparent so you can see the code behind the window.

When you type the dot (`.`) after `Console`, the *IntelliSense* window reappears and shows only the members of class `Console` that can be used on the right of the dot (Fig. 3.6(b)). When you type the open parenthesis character, `(`, after `Console.WriteLine`, the ***Parameter***

Info window is displayed (Fig. 3.7). This window contains information about the method's parameters. A class can define several methods that have the *same* name, as long as they have *different* numbers and/or types of parameters—a concept known as *overloaded methods*. These methods normally all perform similar tasks. The *Parameter Info* window indicates how many versions of the selected method are available and provides up and down arrows for scrolling through the different versions. For example, there are 19 versions of the `WriteLine` method—we use one of these in our app. The *Parameter Info* window is one of many features provided by the IDE to facilitate app development. In the next several chapters, you'll learn more about the information displayed in these windows. The *Parameter Info* window is especially helpful when you want to see the different ways in which a method can be used. From the code in Fig. 3.1, we already know that we intend to display one string with `WriteLine`, so, because you know exactly which version of `WriteLine` you want to use, you can simply close the *Parameter Info* window by pressing the *Esc* key.



```
// Fig. 3.1: Welcome1.cs
// Text-displaying app.
using System;

public class Welcome1
{
   // Main method begins execution of C# application
   public static void Main( string[] args )
   {
      Console.WriteLine(|
      } /
} // e
```

▲ 1 of 19 ▼ void Console.WriteLine()
Writes the current line terminator to the standard output stream.

Fig. 3.7 | *Parameter Info* window.

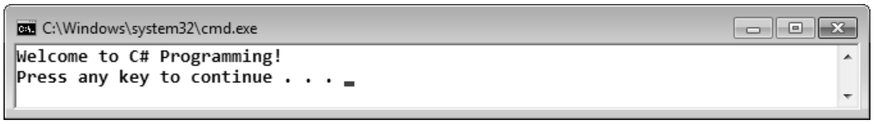
Saving the App

After you type the app's code, select **FILE > Save All** to save the project.

Compiling and Running the App

You're now ready to compile and execute your app. Depending on the project's type, the compiler may compile the code into files with the **`.exe`** (**executable**) **extension**, the **`.dll`** (**dynamically linked library**) **extension** or one of several other extensions. Such files are called **assemblies** and are the packaging units for compiled C# code. These assemblies contain the Microsoft Intermediate Language (MSIL) code for the app.

To compile the app, select **BUILD > Build Solution**. If the app contains no syntax errors, this will create an executable file (named `Welcome1.exe`, in one of the project's subdirectories). To execute it, type *Ctrl* + *F5*, which invokes the `Main` method (Fig. 3.1). If you attempt to run the app before building it, the IDE will build the app first, then run it only if there are no compilation errors. The statement in line 10 of `Main` displays `Welcome to C# Programming!`. Figure 3.8 shows the results of executing this app, displayed in a console (**Command Prompt**) window. Leave the app's project open in Visual Studio; we'll go back to it later in



```
Welcome to C# Programming!
Press any key to continue . . .
```

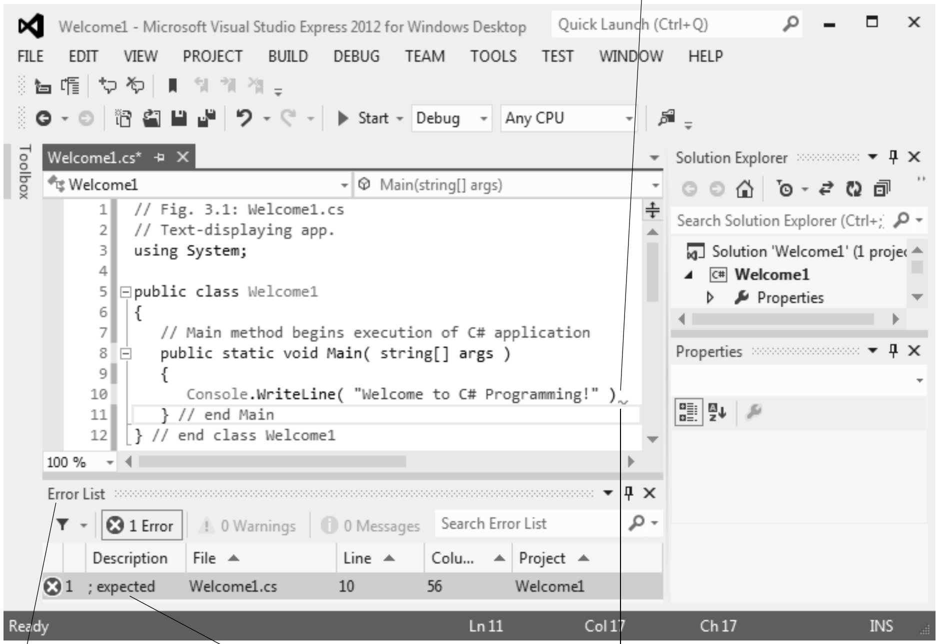
Fig. 3.8 | Executing the app shown in Fig. 3.1.

this section. [*Note:* The console window normally has a black background and white text. We reconfigured it to have a white background and black text for readability. If you'd like to do this, click the ■ icon in the upper-left corner of the console window, then select **Properties**. You can change the colors in the **Colors** tab of the dialog that appears.]

Syntax Errors, Error Messages and the Error List *Window*

Go back to the app in Visual Studio. As you type code, the IDE responds either by applying syntax-color highlighting or by generating a **syntax error**, which indicates a violation of Visual C#'s rules for creating correct apps. Syntax errors occur for various reasons, such as missing parentheses and misspelled keywords.

When a syntax error occurs, the IDE underlines the location of the error with a red squiggly line and provides a description of it in the **Error List window** (Fig. 3.9). If the


Intentionally omitted semicolon (syntax error)
Welcome1 - Microsoft Visual Studio Express 2012 for Windows Desktop
Quick Launch (Ctrl+Q)
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST WINDOW HELP
Start
Debug
Any CPU
Toolbox
Welcome1.cs*
Welcome1
Main(string[] args)
Solution Explorer
Search Solution Explorer (Ctrl+;
Solution 'Welcome1' (1 projec
Welcome1
Properties
Properties
100 %
Error List
1 Error
0 Warnings
0 Messages
Search Error List
Ready
Ln 11
Col 17
Ch 17
INS
Error List window
Error description(s)
Squiggly underline indicates a syntax error

```
// Fig. 3.1: Welcome1.cs
// Text-displaying app.
using System;

public class Welcome1
{
   // Main method begins execution of C# application
   public static void Main( string[] args )
   {
      Console.WriteLine( "Welcome to C# Programming!" )
   } // end Main
} // end class Welcome1
```

	Description	File	Line	Colu...	Project
1	; expected	Welcome1.cs	10	56	Welcome1

Fig. 3.9 | Syntax error indicated by the IDE.

Error List window is not visible in the IDE, select **VIEW > Error List** to display it. In Figure 3.9, we intentionally omitted the semicolon at the end of the statement in line 10. The error message indicates that the semicolon is missing. You can double click an error message in the **Error List** to jump to the error's location in the code.

Error-Prevention Tip 3.2

One syntax error can lead to multiple entries in the **Error List** *window. Each error that you address could eliminate several subsequent error messages when you recompile your app. So when you see an error you know how to fix, correct it and recompile—this may make several other errors disappear.*

3.4 Modifying Your Simple C# App

This section continues our introduction to C# programming with two examples that modify the example of Fig. 3.1.

Displaying a Single Line of Text with Multiple Statements

Class `Welcome2`, shown in Fig. 3.10, uses two statements to produce the same output as that shown in Fig. 3.1. From this point forward, we highlight the new and key features in each code listing, as shown in lines 10–11 of Fig. 3.10.

```
 1  // Fig. 3.10: Welcome2.cs
 2  // Displaying one line of text with multiple statements.
 3  using System;
 4
 5  public class Welcome2
 6  {
 7     // Main method begins execution of C# app
 8     public static void Main( string[] args )
 9     {
10        Console.Write( "Welcome to " );
11        Console.WriteLine( "C# Programming!" );
12     } // end Main
13  } // end class Welcome2
```

```
Welcome to C# Programming!
```

Fig. 3.10 | Displaying one line of text with multiple statements.

The app is almost identical to Fig. 3.1. We discuss the changes here. Line 2

```
// Displaying one line of text with multiple statements.
```

states the purpose of this app. Line 5 begins the `Welcome2` class declaration.

Lines 10–11 of method `Main`

```
Console.Write( "Welcome to " );
Console.WriteLine( "C# Programming!" );
```

display one line of text in the console window. The first statement uses `Console`'s method **`Write`** to display a string. Unlike `WriteLine`, after displaying its argument, `Write` does *not* position the screen cursor at the beginning of the next line in the console window—the next character the app displays will appear immediately after the last character that `Write` displays. Thus, line 11 positions the first character in its argument (the letter "`C`") immediately *after* the last character that line 10 displays (the space character before the string's closing double-quote character). Each `Write` statement resumes displaying characters from where the last `Write` statement displayed its last character.

Displaying Multiple Lines of Text with a Single Statement

A single statement can display multiple lines by using newline characters, which indicate to `Console` methods `Write` and `WriteLine` when they should position the screen cursor to the beginning of the next line in the console window. Like space characters and tab characters, newline characters are whitespace characters. The app of Fig. 3.11 outputs four lines of text, using newline characters to indicate when to begin each new line.

```
 1  // Fig. 3.11: Welcome3.cs
 2  // Displaying multiple lines with a single statement.
 3  using System;
 4
 5  public class Welcome3
 6  {
 7     // Main method begins execution of C# app
 8     public static void Main( string[] args )
 9     {
10        Console.WriteLine( "Welcome\nto\nC#\nProgramming!" );
11     } // end Main
12  } // end class Welcome3
```

```
Welcome
to
C#
Programming!
```

Fig. 3.11 | Displaying multiple lines with a single statement.

Most of the app is identical to the apps of Fig. 3.1 and Fig. 3.10, so we discuss only the changes here. Line 2

```
// Displaying multiple lines with a single statement.
```

states the purpose of this app. Line 5 begins the `Welcome3` class declaration.

Line 10

```
Console.WriteLine( "Welcome\nto\nC#\nProgramming!" );
```

displays four separate lines of text in the console window. Normally, the characters in a string are displayed exactly as they appear in the double quotes. Note, however, that the two characters \ and `n` (repeated three times in the statement) do *not* appear on the screen. The **backslash** (\) is called an **escape character**. It indicates to C# that a "special character" is in the string. When a backslash appears in a string of characters, C# combines the next character with the backslash to form an **escape sequence.** The escape sequence `\n` represents the **newline character**. When a newline character appears in a string being output with `Console` methods, the newline character causes the screen cursor to move to the beginning of the next line in the console window. Figure 3.12 lists several common escape sequences and describes how they affect the display of characters in the console window.

Escape sequence	Description
`\n`	Newline. Positions the screen cursor at the beginning of the next line.
`\t`	Horizontal tab. Moves the screen cursor to the next tab stop.
`\"`	Double quote. Used to place a double-quote character (") in a string—e,g., `Console.Write( "\"in quotes\"" );` displays `"in quotes"`.
`\r`	Carriage return. Positions the screen cursor at the beginning of the current line—does not advance the cursor to the next line. Any characters output after the carriage return overwrite the characters previously output on that line.
`\\`	Backslash. Used to place a backslash character in a string.

Fig. 3.12 | Some common escape sequences.

3.5 Formatting Text with Console.Write and Console.WriteLine

`Console` methods `Write` and `WriteLine` also have the capability to display formatted data. Figure 3.13 outputs the strings `"Welcome to"` and `"C# Programming!"` with `WriteLine`.

```
 1  // Fig. 3.13: Welcome4.cs
 2  // Displaying multiple lines of text with string formatting.
 3  using System;
 4
 5  public class Welcome4
 6  {
 7     // Main method begins execution of C# app
 8     public static void Main( string[] args )
 9     {
10        Console.WriteLine( "{0}\n{1}", "Welcome to", "C# Programming!" );
11     } // end Main
12  } // end class Welcome4
```

```
Welcome to
C# Programming!
```

Fig. 3.13 | Displaying multiple lines of text with string formatting.

Line 10

```
    Console.WriteLine( "{0}\n{1}", "Welcome to", "C# Programming!" );
```

calls method `Console.WriteLine` to display the app's output. The method call specifies three arguments. When a method requires multiple arguments, the arguments are separated with **commas** (`,`)—this is known as a **comma-separated list**.

Good Programming Practice 3.4

Place a space after each comma (,) in an argument list to make apps more readable.

Most statements end with a semicolon (`;`). Therefore, line 10 represents only one statement. Large statements can be split over many lines, but there are some restrictions.

Common Programming Error 3.4

Splitting a statement in the middle of an identifier or a string is a syntax error.

Format Strings and Format Items

Method `WriteLine`'s first argument is a **format string** that may consist of **fixed text** and **format items**. Fixed text is output by `WriteLine`, as in Fig. 3.1. Each format item is a placeholder for a value. Format items also may include optional formatting information.

Format items are enclosed in curly braces and contain characters that tell the method which argument to use and how to format it. For example, the format item `{0}` is a *placeholder* for the first additional argument (because C# starts counting from 0), `{1}` is a *placeholder* for the second, and so on. The format string in line 10 specifies that `WriteLine` should output two arguments and that the first one should be followed by a newline character. So this example substitutes `"Welcome to"` for the `{0}` and `"C# Programming!"` for the `{1}`. The output shows that two lines of text are displayed. Because braces in a formatted string normally indicate a placeholder for text substitution, you must type two left braces (`{{`) or two right braces (`}}`) to insert a single left or right brace into a formatted string, respectively. We introduce additional formatting features as they're needed in our examples.

3.6 Another C# App: Adding Integers

Our next app reads (or inputs) two **integers** (whole numbers, like –22, 7, 0 and 1024) typed by a user at the keyboard, computes the sum of the values and displays the result. This app must keep track of the numbers supplied by the user for the calculation later in the app. Apps remember numbers and other data in the computer's memory and access that data through app elements called **variables**. The app of Fig. 3.14 demonstrates these concepts. In the sample output, we highlight data the user enters at the keyboard in bold.

```
// Fig. 3.14: Addition.cs
// Displaying the sum of two numbers input from the keyboard.
using System;

public class Addition
{
   // Main method begins execution of C# app
   public static void Main( string[] args )
   {
      int number1; // declare first number to add
      int number2; // declare second number to add
      int sum; // declare sum of number1 and number2
```

Fig. 3.14 | Displaying the sum of two numbers input from the keyboard. (Part 1 of 2.)

```
13
14          Console.Write( "Enter first integer: " ); // prompt user
15          // read first number from user
16          number1 = Convert.ToInt32( Console.ReadLine() );
17
18          Console.Write( "Enter second integer: " ); // prompt user
19          // read second number from user
20          number2 = Convert.ToInt32( Console.ReadLine() );
21
22          sum = number1 + number2; // add numbers
23
24          Console.WriteLine( "Sum is {0}", sum ); // display sum
25       } // end Main
26    } // end class Addition
```

```
Enter first integer: 45
Enter second integer: 72
Sum is 117
```

Fig. 3.14 | Displaying the sum of two numbers input from the keyboard. (Part 2 of 2.)

Comments
Lines 1–2

```
// Fig. 3.14: Addition.cs
// Displaying the sum of two numbers input from the keyboard.
```

state the figure number, file name and purpose of the app.

Class `Addition`
Line 5

```
public class Addition
```

begins the declaration of class `Addition`. Remember that the body of each class declaration starts with an opening left brace (line 6) and ends with a closing right brace (line 26).

Function `Main`
The app begins execution with `Main` (lines 8–25). The left brace (line 9) marks the beginning of `Main`'s body, and the corresponding right brace (line 25) marks the end of `Main`'s body. Method `Main` is indented one level within the body of class `Addition` and the code in the body of `Main` is indented another level for readability.

Declaring Variable `number1`
Line 10

```
int number1; // declare first number to add
```

is a **variable declaration statement** that specifies the name (`number1`) and type of a variable (`int`) used in this app. Variables are typically declared with a **name** and a **type** before they're used. A variable's name can be any valid identifier. A variable's type specifies what kind of information is stored at that location in memory and how much space should be set aside to store that value. Like other statements, declaration statements end with a semicolon (`;`).

Type `int`

The declaration in line 10 specifies that the variable named `number1` is of type **`int`**—it will hold **integer** values (whole numbers such as `7`, `-11`, `0` and `31914`). The range of values for an `int` is –2,147,483,648 (`int.MinValue`) to +2,147,483,647 (`int.MaxValue`). We'll soon discuss types **`float`**, **`double`** and **`decimal`**, for specifying real numbers, and type **`char`**, for specifying characters. Real numbers contain decimal points, as in `3.4`, `0.0` and `-11.19`. Variables of type `float` and `double` store approximations of real numbers in memory. Variables of type `decimal` store real numbers precisely (to 28–29 significant digits), so `decimal` variables are often used with *monetary calculations*. Variables of type `char` represent individual characters, such as an uppercase letter (e.g., `A`), a digit (e.g., `7`), a special character (e.g., `*` or `%`) or an escape sequence (e.g., the newline character, `\n`). Types such as `int`, `float`, `double`, `decimal` and `char` are often called **simple types**. Simple-type names are keywords and must appear in all lowercase letters. Appendix B summarizes the characteristics of the simple types (`bool`, `byte`, `sbyte`, `char`, `short`, `ushort`, `int`, `uint`, `long`, `ulong`, `float`, `double` and `decimal`).

Declaring Variables `number2` *and* `sum`

The variable declaration statements at lines 11–12

```
int number2; // declare second number to add
int sum; // declare sum of number1 and number2
```

similarly declare variables `number2` and `sum` to be of type `int`.

Variable declaration statements can be split over several lines, with the variable names separated by commas (i.e., a comma-separated list of variable names). Several variables of the same type may be declared in one declaration or in multiple declarations. For example, lines 10–12 can also be written as follows:

```
int number1, // declare first number to add
    number2, // declare second number to add
    sum; // declare sum of number1 and number2
```

Good Programming Practice 3.5

Declare each variable on a separate line. This format allows a comment to be easily inserted next to each declaration.

Good Programming Practice 3.6

By convention, variable-name identifiers begin with a lowercase letter, and every word in the name after the first word begins with a capital letter. This naming convention is known as ***lower camel casing****.*

Prompting the User for Input

Line 14

```
Console.Write( "Enter first integer: " ); // prompt user
```

uses `Console.Write` to display the message `"Enter first integer: "`. This message is called a **prompt** because it directs the user to take a specific action.

Reading a Value into Variable **`number1`**

Line 16

```
    number1 = Convert.ToInt32( Console.ReadLine() );
```

works in two steps. First, it calls the `Console`'s **`ReadLine`** method, which waits for the user to type a string of characters at the keyboard and press the *Enter* key. As we mentioned, some methods perform a task then return the result of that task. In this case, `ReadLine` returns the text the user entered. Then, the `string` is used as an argument to class **`Convert`**'s **`ToInt32`** method, which converts this sequence of characters into data of type `int`. In this case, method `ToInt32` returns the `int` representation of the user's input.

Possible Erroneous User Input

Technically, the user can type anything as the input value. `ReadLine` will accept it and pass it off to the `ToInt32` method. This method assumes that the string contains a valid integer value. In this app, if the user types a noninteger value, a runtime logic error called an exception will occur and the app will terminate. C# offers a technology called *exception handling* that will help you make your apps more robust by enabling them to handle exceptions and continue executing. We introduce exception handling in Section 8.4, then use it again in Chapter 10. We take a deeper look at exception handling in Chapter 13.

Assigning a Value to a Variable

In line 16, the result of the call to method `ToInt32` (an `int` value) is placed in variable `number1` by using the **assignment operator**, `=`. The statement is read as "`number1` gets the value returned by `Convert.ToInt32`." Operator `=` is a **binary operator**, because it works on two pieces of information. These are known as its **operands**—in this case, the operands are `number1` and the result of the method call `Convert.ToInt32`. Everything to the right of the assignment operator, `=`, is always evaluated *before* the assignment is performed.

Good Programming Practice 3.7

Place spaces on either side of a binary operator to make the code more readable.

Prompting the User for Input and Reading a Value into Variable **`number2`**

Line 18

```
    Console.Write( "Enter second integer: " ); // prompt user
```

prompts the user to enter the second integer. Line 20

```
    number2 = Convert.ToInt32( Console.ReadLine() );
```

reads a second integer and assigns it to the variable `number2`.

Summing the **`number1`** *and* **`number2`**

Line 22

```
    sum = number1 + number2; // add numbers
```

calculates the sum of `number1` and `number2` and assigns the result to variable `sum` by using the assignment operator, `=`. The statement is read as "`sum` gets the value of `number1 + number2`." Most calculations are performed in assignment statements. When `number1 + number2` is encountered, the values stored in the variables are used in the calculation. The addition operator is a binary operator—its two **operands** are `number1` and `number2`. Portions of statements that contain calculations are called **expressions**. In fact, an expression is any portion of a statement that has a value associated with it. For example, the value of the expression `number1 + number2` is the sum of the numbers. Similarly, the value of the expression `Console.ReadLine()` is the string of characters typed by the user.

Displaying the `sum`

After the calculation has been performed, line 24

```
      Console.WriteLine( "Sum is {0}", sum ); // display sum
```

uses method `Console.WriteLine` to display the `sum`. The format item `{0}` is a placeholder for the first argument after the format string. Other than the `{0}` format item, the remaining characters in the format string are all *fixed* text. So method `WriteLine` displays `"Sum is "`, followed by the value of `sum` (in the position of the `{0}` format item) and a newline.

Performing Calculations in Output Statements

Calculations can also be performed inside output statements. We could have combined the statements in lines 22 and 24 into the statement

```
      Console.WriteLine( "Sum is {0}", ( number1 + number2 ) );
```

The parentheses around the expression `number1 + number2` are not required—they're included for clarity to emphasize that the value of the expression `number1 + number2` is output in the position of the `{0}` format item.

3.7 Arithmetic

Most apps perform arithmetic calculations. The **arithmetic operators** are summarized in Fig. 3.15. Note the various special symbols not used in algebra. The **asterisk** (`*`) indicates multiplication, and the **percent sign** (**%**) is the **remainder operator** (called *modulus* in some languages), which we'll discuss shortly. The arithmetic operators in Fig. 3.15 are binary operators—for example, the expression `f + 7` contains the binary operator `+` and the two operands `f` and `7`.

C# operation	Arithmetic operator	Algebraic expression	C# expression
Addition	`+`	$f + 7$	`f + 7`
Subtraction	`–`	$p - c$	`p - c`
Multiplication	`*`	$b \cdot m$	`b * m`
Division	`/`	x / y or $\frac{x}{y}$ or $x \div y$	`x / y`
Remainder	`%`	$r \bmod s$	`r % s`

Fig. 3.15 | Arithmetic operators.

If both operands of the division operator (`/`) are integers, **integer division** is performed and the result is an integer—for example, the expression `7 / 4` evaluates to `1`, and the expression `17 / 5` evaluates to `3`. Any fractional part in integer division is simply *truncated* (i.e., discarded)—*no rounding* occurs. C# provides the remainder operator, `%`, which yields the remainder after division. The expression `x % y` yields the remainder after `x` is divided by `y`. Thus, `7 % 4` yields `3`, and `17 % 5` yields `2`. This operator is most commonly used with integer operands but can also be used with `float`s, `double`s, and `decimal`s.

Arithmetic Expressions in Straight-Line Form

Arithmetic expressions must be written in **straight-line form** to facilitate entering apps into the computer. Thus, expressions such as "`a` divided by `b`" must be written as `a / b`, so that all constants, variables and operators appear in a straight line. The following algebraic notation is not acceptable to compilers:

$$\frac{a}{b}$$

Parentheses for Grouping Subexpressions

Parentheses are used to group terms in C# expressions in the same manner as in algebraic expressions. For example, to multiply `a` times the quantity `b + c`, we write

```
a * ( b + c )
```

If an expression contains **nested parentheses**, such as

```
( ( a + b ) * c )
```

the expression in the *innermost* set of parentheses (`a + b` in this case) is evaluated first.

Rules of Operator Precedence

C# applies the operators in arithmetic expressions in a precise sequence determined by the following **rules of operator precedence**, which are generally the same as those followed in algebra (Fig. 3.16). These rules enable C# to apply operators in the correct order.[2]

Operators	Operations	Order of evaluation (associativity)
Evaluated first		
`*`	Multiplication	If there are several operators of this type,
`/`	Division	they're evaluated from left to right.
`%`	Remainder	
Evaluated next		
`+`	Addition	If there are several operators of this type,
`-`	Subtraction	they're evaluated from left to right.

Fig. 3.16 | Precedence of arithmetic operators.

2. We discuss simple examples here to explain the order of evaluation of expressions. More subtle order of evaluation issues occur in complex expressions. For more information, see the following blog posts from Eric Lippert: `blogs.msdn.com/ericlippert/archive/2008/05/23/precedence-vs-associativity-vs-order.aspx` and `blogs.msdn.com/oldnewthing/archive/2007/08/14/4374222.aspx`.

When we say that operators are applied from left to right, we're referring to their **associativity**. You'll see that some operators associate from right to left. Figure 3.16 summarizes these rules of operator precedence. The table will be expanded as additional operators are introduced. Appendix A provides the complete operator precedence chart.

3.8 Decision Making: Equality and Relational Operators

A **condition** is an expression that can be either **true** or **false**. This section introduces a simple version of C#'s **`if` statement** that allows an app to make a **decision** based on the value of a condition. For example, the condition "grade is greater than or equal to 60" determines whether a student passed a test. If the condition in an `if` statement is true, the body of the `if` statement executes. If the condition is false, the body does *not* execute. We'll see an example shortly.

Conditions in `if` statements can be formed by using the **equality operators** (**==** and **!=**) and **relational operators** (**>**, **<**, **>=** and **<=**) summarized in Fig. 3.17. The two equality operators (== and !=) each have the same level of precedence, the relational operators (>, <, >= and <=) each have the same level of precedence, and the equality operators have lower precedence than the relational operators. They all associate from left to right.

Common Programming Error 3.5

Confusing the equality operator, ==, with the assignment operator, =, can cause a logic error or a syntax error. The equality operator should be read as "is equal to," and the assignment operator should be read as "gets" or "gets the value of." To avoid confusion, some programmers read the equality operator as "double equals" or "equals equals."

Standard algebraic equality and relational operators	C# equality or relational operator	Sample C# condition	Meaning of C# condition
Relational operators			
>	`>`	`x > y`	`x` is greater than `y`
<	`<`	`x < y`	`x` is less than `y`
≥	`>=`	`x >= y`	`x` is greater than or equal to `y`
≤	`<=`	`x <= y`	`x` is less than or equal to `y`
Equality operators			
=	`==`	`x == y`	`x` is equal to `y`
≠	`!=`	`x != y`	`x` is not equal to `y`

Fig. 3.17 | Relational and equality operators.

Using the `if` Statement

Figure 3.18 uses six `if` statements to compare two integers entered by the user. If the condition in any of these `if` statements is true, the assignment statement associated with that `if` statement executes. The app uses class `Console` to prompt for and read two lines of text from the user, extracts the integers from that text with the `ToInt32` method of class `Con-`

vert, and stores them in variables number1 and number2. Then the app compares the numbers and displays the results of the comparisons that are true.

```
// Fig. 3.18: Comparison.cs
// Comparing integers using if statements, equality operators
// and relational operators.
using System;

public class Comparison
{
    // Main method begins execution of C# app
    public static void Main( string[] args )
    {
        int number1; // declare first number to compare
        int number2; // declare second number to compare

        // prompt user and read first number
        Console.Write( "Enter first integer: " );
        number1 = Convert.ToInt32( Console.ReadLine() );

        // prompt user and read second number
        Console.Write( "Enter second integer: " );
        number2 = Convert.ToInt32( Console.ReadLine() );

        if ( number1 == number2 )
            Console.WriteLine( "{0} == {1}", number1, number2 );

        if ( number1 != number2 )
            Console.WriteLine( "{0} != {1}", number1, number2 );

        if ( number1 < number2 )
            Console.WriteLine( "{0} < {1}", number1, number2 );

        if ( number1 > number2 )
            Console.WriteLine( "{0} > {1}", number1, number2 );

        if ( number1 <= number2 )
            Console.WriteLine( "{0} <= {1}", number1, number2 );

        if ( number1 >= number2 )
            Console.WriteLine( "{0} >= {1}", number1, number2 );
    } // end Main
} // end class Comparison
```

```
Enter first integer: 42
Enter second integer: 42
42 == 42
42 <= 42
42 >= 42
```

Fig. 3.18 | Comparing integers using if statements, equality operators and relational operators. (Part 1 of 2.)

```
Enter first integer: 1000
Enter second integer: 2000
1000 != 2000
1000 < 2000
1000 <= 2000
```

```
Enter first integer: 2000
Enter second integer: 1000
2000 != 1000
2000 > 1000
2000 >= 1000
```

Fig. 3.18 | Comparing integers using `if` statements, equality operators and relational operators. (Part 2 of 2.)

Class `Comparison`

The declaration of class `Comparison` begins at line 6

```
      public class Comparison
```

The class's `Main` method (lines 9–39) begins the execution of the app.

Variable Declarations

Lines 11–12

```
      int number1; // declare first number to compare
      int number2; // declare second number to compare
```

declare the `int` variables used to store the values entered by the user.

Reading the Inputs from the User

Lines 14–16

```
      // prompt user and read first number
      Console.Write( "Enter first integer: " );
      number1 = Convert.ToInt32( Console.ReadLine() );
```

prompt the user to enter the first integer and input the value. The input value is stored in variable `number1`. Lines 18–20

```
      // prompt user and read second number
      Console.Write( "Enter second integer: " );
      number2 = Convert.ToInt32( Console.ReadLine() );
```

perform the same task, except that the input value is stored in variable `number2`.

Comparing Numbers

Lines 22–23

```
      if ( number1 == number2 )
         Console.WriteLine( "{0} == {1}", number1, number2 );
```

compare the values of the variables `number1` and `number2` to determine whether they're equal. An `if` statement always begins with keyword `if`, followed by a condition in parentheses. An `if` statement expects *one* statement in its body. Line 23 executes only if the numbers stored in variables `number1` and `number2` are equal (i.e., the condition is true). The `if` statements in lines 25–26, 28–29, 31–32, 34–35 and 37–38 compare `number1` and `number2` with the operators `!=`, `<`, `>`, `<=` and `>=`, respectively. If the condition in any of the `if` statements is true, the corresponding body statement executes.

No Semicolon at the End of the First Line of an `if` *Statement*

There's no semicolon (`;`) at the end of the first line of each `if` statement. Such a semicolon would result in a logic error at execution time. For example,

```
if ( number1 == number2 ); // logic error
   Console.WriteLine( "{0} == {1}", number1, number2 );
```

would actually be interpreted by C# as

```
if ( number1 == number2 )
   ; // empty statement

Console.WriteLine( "{0} == {1}", number1, number2 );
```

where the semicolon in the line by itself—called the **empty statement**—is the statement to execute if the condition in the `if` statement is true. When the empty statement executes, no task is performed in the app. The app then continues with the output statement, which always executes, regardless of whether the condition is true or false, because the output statement is not part of the `if` statement.

Whitespace

Note the use of whitespace in Fig. 3.18. Recall that whitespace characters, such as tabs, newlines and spaces, are normally ignored by the compiler. So statements may be split over several lines and may be spaced according to your preferences without affecting the meaning of an app. It's incorrect to split identifiers, strings, and multicharacter operators (like `>=`). Ideally, statements should be kept small, but this is not always possible.

Good Programming Practice 3.8

Place no more than one statement per line in an app. This format enhances readability.

Good Programming Practice 3.9

A lengthy statement can be spread over several lines. If a single statement must be split across lines, choose breaking points that make sense, such as after a comma in a comma-separated list, or after an operator in a lengthy expression. If a statement is split across two or more lines, indent all subsequent lines until the end of the statement.

Precedence and Associativity of the Operators We've Discussed So Far

Figure 3.19 shows the precedence of the operators introduced in this chapter. The operators are shown from top to bottom in decreasing order of precedence. All these operators, with the exception of the assignment operator, `=`, associate from left to right. Addition is left associative, so an expression like `x + y + z` is evaluated as if it had been written as `(x + y) + z`. The assignment operator, `=`, associates from right to left, so an expression like `x = y = 0` is evaluated as if it had been written as `x = (y = 0)`, which, as you'll soon see, first assigns the value `0` to variable `y` then assigns the result of that assignment, `0`, to `x`.

Operators	Associativity	Type
`*` `/` `%`	left to right	multiplicative
`+` `-`	left to right	additive
`<` `<=` `>` `>=`	left to right	relational
`==` `!=`	left to right	equality
`=`	right to left	assignment

Fig. 3.19 | Precedence and associativity of operations discussed so far.

3.9 Wrap-Up

You learned many important features of C# in this chapter. First you learned how to display data on the screen in a **Command Prompt** using the `Console` class's `Write` and `WriteLine` methods. Next, we showed how to use format strings and format items to create formatted output strings. You learned how to input data from the keyboard using the `Console` class's `ReadLine` method. We discussed how to perform calculations using C#'s arithmetic operators. Finally, you made decisions using the `if` statement and the relational and equality operators. As you'll see in Chapter 4, C# apps typically contain just a few lines of code in method `Main`—these statements normally create the objects that perform the work of the app. You'll implement your own classes and use objects of those classes in apps.

Index

Symbols

^, boolean logical exclusive OR 147, **149**
    truth table 149
--, predecrement/postdecrement 122
--, prefix/postfix decrement 122
-, private visibility symbol **87**
-, subtraction 65, 66
!, logical negation 147, **150**
    truth table 150
!=, not equals 67
?:, ternary conditional operator 106, 124
.NET 4.5 8
.NET Framework 579
"", empty string **90**
{, left brace 51
}, right brace 51
@ verbatim string character 492
@ XPath attribute symbol **723**
* asterisk occurrence indicator 726
*, multiplication 65, 66
*=, multiplication compound assignment operator 122
/ forward slash in end tags **697**
/ XPath root selector 722
/, division 65, 66
/* */ delimited comment 48
//, single-line comment 47
\, escape character 60
\", double-quote escape sequence 60
\=, division compound assignment operator 122
\n, newline escape sequence 59, 60
\r, carriage-return escape sequence 60
\t, horizontal tab escape sequence 60
\u*yyyy* notation 1113
&, boolean logical AND 147, **149**
&, menu access shortcut 430, 432
&&, conditional AND **147**, 148
    truth table 148
%, remainder 65, 66
%=, remainder compound assignment operator 122
+, addition 65, 66
+, concatenation operator 502
+, public visibility symbol **78**
++, prefix/postfix increment 122
++, preincrement/postincrement 122
+=, addition compound assignment operator 121, 122
<, less than 67
<!--…--> XML comment tags 700
<? and ?> XML processing instruction delimiters **720**
<=, less than or equal 67
<>, angle brackets for XML elements **696**
=, assignment operator 64
-=, subtraction compound assignment operator 122
==, comparison operator 495
==, is equal to 67
=>, lambda operator **626**
>, greater than 67
>=, greater than or equal to 67
|, boolean logical inclusive OR 147, **149**
||, conditional OR 147, **148**
    truth table **148**

A

abbreviating assignment expressions 121
ABCs of a WCF service 881
Abs method of Math 156
absolute addressing (XPath) **722**
absolute positioning 660
absolute value 156
abstract base class 1063
abstract class 319, **324**, 325, 326, 343, 1063
abstract data type (ADT) **280**
abstract keyword 303, **325**
abstract method **325**, 326, 328
abstract operation in the UML **1060**
AcceptButton property of class Form 390
AcceptsReturn property of class TextBox 402
access management 908
access modifier **75**, 84, 1053
    private 261, 289
    protected 261, **289**
    public **75**, 261, 289
access rule in ASP.NET **863**
access shortcut 429
accessor **84**
Account class
    ATM case study 1017, 1020, 1024, 1026, 1034, 1042, 1043, 1044, 1045, 1047, 1058, 1090
Account class with a constructor to initialize instance variable balance 95
accounts-receivable file 519
action 105, 109, 280
action expression in the UML **103**, 106, 1031
action of an object 1030
action state in the UML **103**, 1030
action state symbol **103**
action/decision model of programming **105**
activation in a UML sequence diagram **1045**
Activation property of class ListView **462**, 463
activation record **163**
active control **398**
active tab **25**
active window **389**
ActiveLinkColor property of class LinkLabel 443
ActiveMdiChild property of class Form 474
activity **103**
activity (UML) **1017**, **1029**, 1033
activity diagram 103, 105
    do…while statement 137
    for statement 131
    if statement 105
    if…else statement 106
    in the UML 109
    sequence statement 103
    switch statement 144
    while statement 109
activity diagram (UML) **1017**, 1030, 1032, 1050
actor in use case (UML) **1016**
add a database to a project 617
add a reference to a class library 621
add a row to an entity collection 693
Add method of class ArrayList 585, **586**
Add method of class Dictionary **528**
Add method of class Hashtable **595**
Add method of class HttpSessionState **683**
Add method of class List<T> **251**
Add method of class ObjectCollection 448
Add method of class PointCollection **978**
Add method of class XContainer **743**
Add method of UIElementCollection class **769**, **942**
Add Tab menu item 469
Add User Control… option in Visual Studio .NET 487
Add Windows Form… option in Visual Studio 473
AddDay method of structure DateTime **442**
addition 65
AddLast method of class LinkedList **602**
address of a WCF service **881**
AddYears method of structure DateTime **442**
ADO.NET Entity Data Model **614**
    data source for data binding 622
    entities 619

ADO.NET Entity Framework **609**
  add a row to a table 693
  DbContext class 626
  DBExtensions class **627**
  entity data model **614**
ADT (abstract data type) 280
advertisement 677
advertising revenue 828
AfterSelected event of class TreeView 458
aggregation **1023**
AJAX (Asynchronous Javascript and XML) **693**
Ajax (Asynchronous Javascript and XML) **871**
AJAX web app **693**
Ajax web application **871**
algebraic notation 66
algorithm 106
all XML Schema element **718**
AllowsTransparency property of Window control **957**
alpha (transparency) values 788
Alphabetic icon 32
alphabetizing 495
Alt key 424
Alt key shortcut 429
Alt property of class KeyEventArgs 425, 427
analysis stage of the software life cycle 1015
ancestor node of a DOM tree **728**
anchor a control 398, **398**
Anchor property of class Control 400
Anchoring demonstration 399
Android
  operating system 8
angle bracket (<>) for XML elements **696**
anonymous method **626**
anonymous object-creation expression 635
anonymous type **249**, 635
  Equals method 635
  ToString method 635
Any extension method of interface IEnumerable<T> **249**, 739
Apache HTTP Server 651
app 50
app bar 9
app platforms
  Amazon Kindle 829
  Android 828
  BlackBerry 829
  iPhone 829
  Windows Mobile 829
App.xaml **756**, **932**
App.xaml.cs **757**
App.xaml.vb **932**
AppBar control (Windows 8 UI) **762**
  IsOpen property **762**
app-development process 280
Appearance property of class CheckBox 407
Append method of class StringBuilder 507
AppendFormat method of class StringBuilder 508, 509
AppendText method of class File 521
Apple Computer, Inc. 1110
application 21
Application class **437**
Application counts the number of occurrences of each word in a string and stores them in a generic sorted dictionary 597
Application counts the number of occurrences of each word in a string and stores them in a hash table 593
Application.Current.Shutdown method **950**
Application.Exit method 450
Application.xaml
  StartupUri property **932**
Applicaton class
  Exit method **437**
Applying an XSLT style sheet to an XML document 744
Applying transforms to a Polygon 795, 988
arbitrary number of arguments 234
ARGB values **788**, 788
args parameter of Main method 236
argument promotion **163**
argument to a method **52**, 79
ArgumentException class **595**
ArgumentOutOfRangeException class **258**, 267, 268, 493, 502, 509, 588
arithmetic calculation 65
arithmetic operators 65
arithmetic overflow **280**
ArrangeIcons value of enumeration MdiLayout 475
array **193**
  bounds checking **203**
  ignoring element zero 205
  Length property **194**
  pass an array element to a method 212
  pass an array to a method 212
array-access expression **193**
Array class **580**, 581, 583
Array class static methods for common array manipulations 581
array-creation expression **195**
array initializer **197**
  for jagged array 225
  for rectangular array 224
  nested **224**
array-access expression
  for jagged arrays 225
  for rectangular arrays 224
ArrayList class **580**, **585**, 586
  property Capacity 585
  property Count 585
arrays as references 214
arrow 103
arrowhead in a UML sequence diagram 1045, 1046
article.xml displayed by Internet Explorer 702
artifact in the UML **1107**
as operator 739
as operator (downcasting) 340
ascending modifier of a LINQ orderby clause **244**
ascending order 726
ASCII (American Standard Code for Information Interchange) 1111
ASCII (American Standard Code for Information Interchange) character set 145, 1096
ASCII character, test for 426
ASP.NET **6**, 6, **650**
  AJAX **693**
  ASP.NET Web Forms Application
    template 855, 856, 857, 860, 861, 877
  login 694
  membership capabilities 857
  registration 694
  server control **650**
  start page 663, 666, 671, 680, 687
  validation control **670**
ASP.NET Ajax **871**
ASP.NET Ajax Control Toolkit **874**
ASP.NET Web Forms Application
  template 855, 856, 857, 860, 861, 877
aspect ratio **788**, **983**, 1001
ASPX file **650**
.aspx filename extension **650**
assembly 621
assembly (compiled code) 56
Assets folder (Windows 8 UI) **757**
Assigning base class and derived class references to base class and derived class variables 322
assignment operator, = **64**, 67
assignment operators **121**
associate
  left to right 124
  right to left 117, 124
association (in the UML) **1021**, 1022, 1023, 1055, 1056
  name 1021
associativity of operators **67**, 71, 124
  right to left 67
asterisk (*) occurrence indicator **709**
async modifier xxiv, 6, **834**, 838, 889
asynchronous call **1044**
asynchronous programming 6, **833**
asynchronous request **872**
asynchronous task **833**, 834
ATM (automated teller machine) case study 1010, 1015, 1042
ATM class (ATM case study) 1020, 1021, 1022, 1025, 1026, 1027, 1029, 1034, 1041, 1042, 1043, 1044, 1054
ATM system 1015, 1016, 1018, 1020, 1025, 1029, 1034, 1053
attached properties 774
attached property 784
attached property (Windows 8 UI) **769**
attached property (WPF) **937**
ATTLIST attribute-list declaration (DTD) **709**
attribute 1055, 1056
  compartment in a class diagram 1027
  declaration in the UML 1027, 1029
  in the UML 4, 78, 1020, 1024, 1025, 1026, **1027**, 1029, 1033
  name in the UML 1027
  of an object 4
attribute (XAML) 929
attribute (XML) **704**
attribute element **718**
attribute-list declaration **709**

Attribute method of class XElement **738**
attribute node 723
attribute value in XML **704**
authenticating a user **860**
authentication 859
AuthorISBN table of Books database 610, 612
Authors table of Books database 610, 611
auto-implemented property **88**
auto-hide 29
auto-implemented properties 532
autoincremented database column **611**
automated teller machine (ATM) 1010, 1015
  user interface 1011
automatic garbage collection 369
automatically implemented property **88**
AutoPopDelay property of class ToolTip 418
AutoPostBack property of a DropDownList ASP.NET control **868**
AutoScroll property of class Form 390
AutoScroll property of class Panel 404
AutoSize property of class TextBox **38**
average 110, 112
await xxiv, 6, 889
await expression **834**, 838
await multiple Tasks 844
await operator **834**
awaitable entity 834

B

BackColor property of a form **36**
BackColor property of class Control 398
background color 36
Background property of a Canvas **765**
Background property of a control (Windows 8 UI) **752**
Background property of TextBlock control **983**
Background property of WPF controls 951
BackgroundImage property of class Control 398
backslash, (\) **59**
backup 908
BalanceInquiry class (ATM case study) 1020, 1023, 1026, 1027, 1030, 1031, 1034, 1041, 1042, 1043, 1044, 1045, 1054, 1058, 1059, 1060, 1061
BankDatabase class (ATM case study) 1020, 1024, 1026, 1034, 1035, 1041, 1042, 1043, 1044, 1045, 1047, 1054, 1056
bar chart 200, 201
Bar chart printing application 200
bar of asterisks 200, 201
base
  for constructor initializers **302**
  for invoking overridden methods 311
  keyword 289, 302, 311, 313
base attribute of element extension **717**
base case 184
base class **286**, 1059
  constructor 293
  default constructor 293
  direct **286**, 288
  indirect **286**, 288
  method overridden in a derived class 311
base type (XML Schema) **717**
BasicShapes example 782, 975
behavior 1033
behavior of a system 1029, 1030, 1033
behavior of the system 1043
bidirectional navigability in the UML **1054**
big data 907
BigInteger struct 187
binary (base 2) number system 1098
binary arithmetic operators 117
binary digit **517**
binary operator **64**, 65, 150
BinaryFormatter class **548**
  Deserialize method **548**
  Serialize method 550
BinarySearch method of class Array **583**
BinarySearch method of class ArrayList 588
Binding class (Windows 8 UI) **775**
  ElementName property **775**
  Path property **775**
  Source property **775**
Binding class (WPF) **965**
  ElementName property **965**
  IsAsync property **971**
  Path property **965**
  Source property **965**
Binding markup extension 956
binding of a WCF service **881**
BindingNavigator class **616**, **623**
BindingSource class **623**
  DataSource property **627**
  EndEdit method **627**
  MoveFirst method **631**
bit **517**
bit (size of unit) 1111
bit manipulation **517**
BitArray class **580**
bitwise operators 409
bitwise Xor operator 438
BlackBerry OS 8
blank line 49
Blend for Visual Studio **929**
BLOB service (Windows Azure) 914
block 160
block of statements **108**, 116
Bluetooth 810
BlurEffect **995**
BMP (Windows bitmap) 41
body
  of a class declaration 51
  of a loop 109
  of a method **51**
  of an if statement 67
Books database 610
  table relationships 613
bool attribute **1025**
bool simple type **106**, 1094
boolean expression **106**
boolean logical AND, & 147, **149**
boolean logical exclusive OR, ^ 147, **149**
  truth table 149
boolean logical inclusive OR, | **149**
Border control 934, 995
Border control (Windows 8 UI) 775
Border control (Windows Phone 8) **813**
BorderStyle property of class Panel 404
bottom tier **653**
BottomAppBar property of a Page **762**
boundary of control 486
bounds checking **203**
boxing 591, 595
braces ({ and }) 108, 116, 129
braces not required 142
braces, { } 197
break keyword **141**
break statement 141, 145
  exiting a for statement 145
bricks-and-mortar store 677
brittle software **307**
brush 980
Brush class (WPF) 951
  ImageBrush class 958
bubbling events (WPF) **946**
buffer **520**
BufferedStream class **520**
buffering **520**
BUILD menu 26
Build Web Site command 665
built-in array capabilities 580
built-in data types 715
Business letter marked up as XML 703
business logic **653**
business rule 653
button 387
Button ASP.NET web control **670**
Button class 12, 16, 388, 402
  Click event 402
  FlatStyle property 402
  Text property 402
Button control 934
  Click event **942**, 944
  WPF **934**, 942, 951
Button control (Windows 8 UI) **764**, 769
  Click event **764**, **769**
  Name property **764**
  Style property 764
Button properties and events 402
Button property of class MouseEventArgs 422
ButtonBase class **402**
byte **518**
byte simple type 1094

C

C format specifier 98
.cs file name extension 50
C# keywords 49
C# Language Specification 563
C# programming language **5**
Calculating values to be placed into the elements of an array 198
calculations 71, 103
CalendarForeColor property of class DateTimePicker 440

CalendarMonthBackground property
of class DateTimePicker 440
call stack 377
callback function **872**
callback method 834
calling method (caller) **75**
CancelAsync method of class
WebClient **850**
CancelButton property of class Form
390
CanExecute event of CommandBinding
class **950**
CanExecute method of ICommand
interface **947**
Canvas class 16
Canvas control (Windows 8 UI) 760,
765, 783
Background property **765**
Left attached property **769**
Margin property **765**
Name property **765**
SetLeft method **769**
SetTop method **769**
Top attached property **769**
Canvas control (WPF) **937**, 977
Left attached property **937**, 942
SetLeft method **942**
SetTop method **942**
Top attached property **937**, 942
ZIndex attached property **938**
capacity of a collection **585**
Capacity property of class ArrayList
585, **588**
Capacity property of class List<T>
250, **253**
Capacity property of class
StringBuilder 505
Card class represents a playing card 206
card games 206
card shuffling
Fisher-Yates 209
Card shuffling and dealing
application 209
carriage return 60
carry bit 1105
Cascade value of enumeration
MdiLayout 475
cascaded window **475**
Cascading Style Sheets (CSS) 662
case **141**, 142
keyword **141**
case sensitive **50**
CashDispenser class (ATM case study)
1020, 1022, 1026, 1027, 1034, 1047
casino 167, 172
cast
downcast **321**
cast operator 117, 165, 176
catch
general catch clause **365**
catch all exception types 365
catch an exception **363**
Catch block **205**
catch block (or handler) **365**
catch block with no exception type 365
catch block with no identifier 365
catch-related errors 368
Categorized icon **32**
CDATA keyword (DTD) **710**
Ceiling method of Math 157
CenterX property of ScaleTransform
control **996**
CenterY property of ScaleTransform
control **996**
char
simple type 63
char array 493
char simple type 1094
Char struct 491
CompareTo method 514
IsDigit method 514
IsLetter method 514
IsLetterOrDigit method 514
IsLower method 514
IsPunctuation method 514
IsSymbol method 514
IsUpper method 514
IsWhiteSpace method 514
static character-testing methods
and case-conversion methods 512
ToLower method 514
ToUpper method 514
character 166, **517**
constant **145**
string **51**
character constant **492**
character data in XML **710**
character entity reference 710
character set **518**, 1111
check box **402**
CheckBox class 388, **407**
Appearance property 407
Checked property 407
CheckedChanged event 407
CheckState property 407
CheckStateChanged event 407
Text property 407
ThreeState property **407**
CheckBox control 995
WPF 951
CheckBox properties and events 407
CheckBoxes property of class ListView
462, 463
CheckBoxes property of class TreeView
458
Checked event handler 980
Checked event of a RadioButton
control **763**, **769**
Checked event of RadioButton control
942, 944
Checked property of class CheckBox 407
Checked property of class RadioButton
410
Checked property of class
ToolStripMenuItem 433, **437**
Checked property of class TreeNode 458
CheckedChanged event of class
CheckBox 407
CheckedChanged event of class
RadioButton 410
CheckedIndices property of class
CheckedListBox 451
CheckedItems property of class
CheckedListBox 451
CheckedListBox class 429, **446**, 450
CheckedIndices property 451
CheckedItems property 451
GetItemChecked method 451
CheckedListBox class (cont.)
ItemCheck event **450**, 451
SelectionMode property 451
CheckedListBox properties and events
451
CheckOnClick property of class
ToolStripMenuItem 433
CheckState property of class CheckBox
407
CheckStateChanged event of class
CheckBox 407
child element (XML) **701**, 704
child node **457**
child node (XML)
of a DOM tree **728**
child window **473**
child window maximized 475
child window minimized 475
children (in DOM tree) 728
Children property of Panel control
769, **942**
Choices **1007**
Choose Items... option in Visual Studio
488
chromeless window 9
class **3**, 155, 1028, 1034, 1038, 1053
class keyword 75
declaration 49, 50
declare a method **74**
instance variable **74**, 82, **157**
instantiating an object **74**
name **49**, 50, 390, 482, 1055
partial **665**
partial class **665**
user defined **49**
class average 110
class cannot extend a sealed class 342
class constraint **565**
class diagram
for the ATM system model 1023,
1049
in the UML **1017**, **1020**, 1023,
1025, 1027, 1053, 1056
class diagram (in the UML) 1034, 1061,
1062, 1063
class hierarchy **286**, 325
class keyword 75
Class Library **6**
class library **287**, 314, 481
add a reference 621
assembly 621
Class View (Visual Studio .NET) 281
“class-wide” information **275**
Classes
AppBar **762**
Application **437**
ArgumentException 595
ArgumentOutOfRangeException
267, 268, 588
Array **580**, 581, **583**, **584**
ArrayList 580, **585**, **586**, 588
BinaryFormatter **548**
Binding 775, **965**
BindingNavigator **623**
BindingSource **623**, **627**
BitArray **580**
Border 775
Brush 951
BufferedStream **520**

Classes (cont.)
- Button 402, 764
- ButtonBase 402
- Canvas 760, 765, 937, 977
- CheckBox 407, 995
- CheckedListBox 429, 446, 450
- CloudStorageAccount 914, 917
- CloudTableClient 917
- CollectionView 971
- ColorAnimation 1001
- ComboBox 429, 453
- CommandBinding 947, 950
- Console 51, 52, 58, 60, 519, 520
- ContentControl 763, 934
- ContentPresenter 931, 934, 960, 963
- Control 397, 398, 400, 486
- ControlTemplate 963
- Convert 64, 237
- DataContext 620
- DataContractJsonSerializer 891, 891
- DataGridView 616
- DataTemplate 777, 968
- DateTimePicker 439
- DbContext 615, 620
- Delegate 395
- Dictionary 528, 529, 580
- Directory 521, 525
- DirectoryInfo 467, 521
- DispatchTimer 959
- DivideByZeroException 362, 364, 368
- DockPanel 934
- DoubleAnimation 1001
- DropDownList 669
- Ellipse 784, 976
- Ellipse (Windows 8 UI) 769
- Enumerable 615
- EventArgs 392, 769, 942
- Exception 368
- ExecutionEngineException 368
- File 520, 521, 524, 529
- FileInfo 467
- FileStream 520
- Font 409
- Form 389, 390, 474
- FormatException 362, 364
- GradientStop 792, 986
- Graphics 424, 456
- Grid 752, 760, 774, 936, 937
- GridViewColumn 968
- GroupBox 404, 936
- Hashtable 592, 593, 595
- HttpClient 888
- HttpSessionState 679, 683, 683, 684, 685
- Image 667
- ImageBrush 787, 788, 958, 983
- ImageList 458, 462
- IndexOutOfRangeException 205
- InvalidCastException 340, 579, 596
- InvalidOperationException 584, 591, 602
- ItemCheckEventArgs 451
- KeyEventArgs 424, 425, 427
- KeyNotFoundException 599

Classes (cont.)
- Label 752, 931
- LinearGradientBrush 788, 984
- LinkedList 580, 599, 602
- LinkedListNode 599
- LinkLabel 429, 442, 443
- List 580
- List<T> 250, 251, 253, 580
- ListBox 429, 446
- ListBox.ObjectCollection 447
- ListView 462
- ListViewItem 463
- Match 491, 514
- Math 156
- MediaElement 805, 983
- MemoryStream 520
- MenuStrip 430
- MonthCalendar 438
- MouseButtonEventArgs 943
- MouseEventArgs 422
- MouseWheelEventArgs 943
- MulticastDelegate 395
- NullReferenceException 368
- NumericUpDown 388, 419
- object 290, 314
- ObjectCollection 447, 448, 450
- ObservableCollection<T> 627, 631
- OpenFileDialog 542, 547
- OutOfMemoryException 368
- Page 665, 676, 681
- PaintEventArgs 485
- Panel 404, 760, 769, 942
- Path 462, 528
- PictureBox 415, 476
- Point 978
- PointAnimation 1001
- PointCollection 786, 978, 980
- PointerRoutedEventArgs 769
- Polygon 784, 977
- PolyLine 784, 977
- Process 445
- Queryable 615
- Queue 580, 580
- RadioButton 407, 410, 769, 942
- Random 167, 797, 989
- ResourceManager 417
- Resources 417
- RoutedEventArgs 769, 942, 944
- SaveFileDialog 536
- ScaleTransform 996
- Setter 963
- SolidBrush 424
- SortedDictionary 580, 597, 598
- SortedList 580
- SortedSet<T> 605
- Stack 567, 580, 589
- StackOverflowException 368
- StackPanel 760
- Stream 520, 520
- StreamReader 520
- StreamWriter 520
- string 491
- StringBuilder 491, 504, 507, 508, 509
- SystemException 368
- TabControl 468
- TableEntity 914
- TableQuery 918

Classes (cont.)
- TabPage 468
- Task 838
- TextBlock 749, 983
- TextBox 388
- TextReader 520
- TextWriter 520
- Timer 487, 960
- ToolStripMenuItem 430, 432
- ToolTip 417, 418
- TreeNode 457, 458
- TreeNodeCollection 457
- TreeView 429, 457, 458, 731
- TreeViewEventArgs 458
- Type 315, 340
- UIElementCollection 769, 942
- UnauthorizedAccessException 462
- Uri 889
- UserControl 486
- ValueType 512
- WebClient 850
- XAttribute 738
- XComment 740
- XContainer 737
- XDocument 729
- XDocumentType 739
- XElement 729
- XName 731
- XNamespace 739, 740
- XNode 739
- XObject 740
- XProcessingInstruction 740
- XslCompiledTransform 745
- XText 740

Clear method of class Array 584
Clear method of class ArrayList 585
Clear method of class Dictionary 529
Clear method of class Graphics 456
Clear method of class List<T> 250
Clear method of class ObjectCollection 450
Clear method of class PointCollection 980
Clear method of UIElementCollection class 769, 942
ClearSelected method of class ListBox 447
click a Button 390, 402
Click event handler 980
Click event of a Button control 764, 769
Click event of Button control 942, 944
Click event of class Button 402
Click event of class PictureBox 415
Click event of class ToolStripMenuItem 432, 433
Clicks property of class MouseEventArgs 422
client code 321
client of a class 280, 1033, 1043
Client that consumes the WelcomeRESTXMLService 888
client tier 653
ClipRectangle property of class PaintEventArgs 485, 486
clock 487
cloning objects
- shallow copy 315

close a project 26
close a window 390
close box 43, 44
Close method of class Form 390
cloud computing **11**, 907
  Amazon Cloud Computing services 908
  Audiobox 909
  Box 908
  Cloudbees 909
  Dropbox 908
  Evernote 909
  Google Cloud Platform 908
  Heroku 909
  iCloud 909
  Microsoft SkyDrive 908
  NaviSite 908
  PrimaDesk 909
  Rackspace 908
  Salesforce 909
  SOS Online Backup 909
  Spotify 909
  Waze 909
Cloud Security Alliance 921
CloudStorageAccount class (Windows Azure) **914**, 917
  CreateCloudTableClient method **917**
CloudTable class (Windows Azure)
  CreateIfNotExists method **917**
  Execute method **919**
  ExecuteQuery method **918**
CloudTableClient class (Windows Azure) **917**, 917
  GetTableReference method **917**
CLR (Common Language Runtime) 7, 369, 383
CML (Chemical Markup Language) 697
code-behind file **650**
code reuse 286, 578
code snippets **89**
code snippets (IDE)
  switch 177
code value (Unicode) **1112**
code-behind class (Windows 8 UI) 757
code-behind class (WPF) 930, 932
coin tossing 167
collaboration diagram in the UML **1017**, **1043**
collaboration in the UML **1040**, 1041, 1044
collapse a tree 30
Collapse method of class TreeNode 458
collapse node 457
collection 250, 558, **578**
collection class **578**
collection initializers **254**
collection view (WPF) **971**
CollectionView class (WPF) **971**
CollectionViewSource.GetDefaultView method **971**
collision **592**
Color property of GradientStop control **792**, **986**
color resources 819
Color structure **424**
ColorAnimation (WPF) 1001
column 223
Column attached property of Grid control **774**, **937**
column index **228**
column of a database table **609**, 610
ColumnDefinition class
  associated with Grid control 774, **817**, **936**
  Width property 817
ColumnDefinitions property of Grid control **774**, **817**, **936**
columns of a two-dimensional array **223**
ColumnSpan attached property of Grid control **775**, **937**
ComboBox class 388, 429, **453**
  DropDownStyle property 453, **454**
  Items property 453, **454**
  MaxDropDownItems property **453**
  SelectedIndex property **454**
  SelectedIndexChanged event **454**
  SelectedItem property **454**
  Sorted property 454
ComboBox control
  SelectedIndexChanged event handler 631
ComboBox control (WPF) 951
ComboBox demonstration 453
ComboBox properties and an event 453
ComboBox used to draw a selected shape 454
ComboBoxStyle enumeration **454**
  DropDown value 454
  DropDownList value 454
  Simple value 454
comma (,) 133
comma in an argument list 61
comma-separated list 133
  of arguments **60**, 63
command binding (WPF) **947**
command library (WPF) 947
command-line argument **158**, **236**, 237
Command Prompt **47**, 236
CommandBinding class (WPF) **947**
  CanExecute event **950**
  Executed event **948**
  PreviewCanExecute event **950**
  PreviewExecuted event **948**
CommandBindings property of Window control **950**
commands (WPF) **946**
comment **47**
CommissionEmployee class represents a commission employee 291, 308
CommissionEmployee class that extends Employee 333
Common folder (Windows 8 UI) **757**
Common Language Runtime (CLR) 7, 369, 383
Common Programming Errors overview xxvii
communication diagram in the UML **1017**, **1043**, 1044
CompareTo method
  of IComparable **564**
  of interface IComparable 352
CompareTo method of struct Char 514
comparison operator 352
comparison operators **67**
compile 52
compile-time type safety **558**
complex content in XML Schema **716**
ComplexNumber class 353
complexType XML Schema element **715**
component **388**
component diagram in the UML **1107**
component in the UML **1107**
component selection drop-down list **32**
component tray **418**, 623
composite key **609**
composite primary key 612
composite structure diagram in the UML **1108**
composition **271**, 287, 289, **1022**, 1048
compound assignment operators 121, 123
  *= 122
  \= 122
  %= 122
  += 122
  -= 122
compound interest 133
compound-interest
  calculating with for 133
Concat method of class string 502
concatenate strings 277
concrete class **324**
concrete derived class 329
concurrency 833
concurrent operations 833
condition **67**, 136
conditional AND (&&) operator **147**, 149, 248
  truth table 148
conditional expression **106**
conditional operator, ?: 106, 124
conditional OR, || 147, **148**
  truth table **148**
confusing the equality operator == with the assignment operator = 67
connect to a database 615, **617**
connection string 618, 622
console app **47**, 52
Console class 519, 520
  ReadLine method 80
console window 47, 58, 60
Console.Write method **60**
Console.WriteLine method **52**, 58
const
  keyword 145, **157**, 198
const keyword 278
constant **145**, 157, 198
  declare 198
  must be initialized 198
constant integral expression **137**, 145
constant string expression **137**, 145
constant variable 199
Constants
  Nan of structure Double **362**, 383
  NegativeInfinity of structure Double **362**
  PositiveInfinity of structure Double **362**
constituent controls **486**
constructor 91
  naming 92
  parameter list 92

constructor constraint (new()) **566**
constructor initializer **267**, 302
with keyword base 302
consuming a web service **883**
container 388, **389**
container element (XML) **701**
Contains method of class ArrayList 585, **588**
Contains method of class List<T> 250, **253**
Contains method of class Stack **591**
ContainsKey method of class Dictionary **528**
ContainsKey method of Hashtable **595**
content control (Windows 8 UI) **752**
content control (WPF) **931**
content page in ASP.NET **857**
Content property of a RadioButton control **763**
Content property of ContentPresenter class **963**
ContentControl class (Windows 8 UI) 763
ContentControl class (WPF) **931**, 934
ContentPresenter class (WPF) **960**
Content property **963**
ContentPresenter element in a ControlTemplate **801**
context node (XPath) **726**
context-sensitive help **33**
contextual keywords 49, **50**, 85
continue keyword **145**
continue statement 145, 146
terminating an iteration of a for statement 146
contract of a WCF service **881**
contravariance **605**
control **25**, 31, 388, 812
control (Windows 8 UI) **751**
control (WPF) **930**, 960
control boundary 486
Control class **397**, 486
Anchor property 400
BackColor property 398
BackgroundImage property 398
Dock property 400
Enabled property **398**
Focused property 398
Font property 398
ForeColor property 398
Hide method 398
KeyDown event **424**, 425
KeyPress event **424**, 425
KeyUp event **424**, 425
Location property 400
MaximumSize property 400
MinimumSize property 400
MouseDown event 422
MouseEnter event 422
MouseHover event 422
MouseLeave event 422
MouseMove event 422
MouseUp event 422
MouseWheel event 422
OnPaint method **485**
Padding property 400
Select method **398**
Show method 398
Control class (cont.)
Size property 400
TabIndex property **398**
TabStop property 398
Text property 398
Visible property 398
control layout and properties 397
Control property of class KeyEventArgs 425, 427
control statement 104, 105, 127
nesting 104
stacking 104
control structure 102
control template (WPF) **960**, 961
control variable **110**, **127**, 128, 130
controller logic **653**
Controls **12**, **16**
987
BindingNavigator **623**
Border **775**, 934, 995
Border (Windows Phone 8) **813**
Button 12, 16, **670**, **764**, 769, **934**, 942, 951
Canvas 16, **765**, **937**, **938**, 977
CheckBox 951, 995
ComboBox 951
ContentControl 763, **931**, 934
DataGridView **616**
DockPanel 934
DoubleAnimation **1001**
DropDownList **669**, **868**
Ellipse **784**, **976**
EntityDataSource **689**
Expander 934
Form 930
GradientStop **792**, **986**
Grid **752**, 774, 931, **936**, 937
Grid (Windows Phone 8) 813
GridView **968**
GroupBox 12, 16, **936**
HyperLink **670**
Image **667**, 771, 775
ImageBrush **788**, 958, **983**
Label **25**, **34**, 37, **752**, **931**
Line **784**, **976**
LinearGradientBrush **788**, **984**
ListView 771, **965**
MediaElement **805**, **983**
Menu **950**
MenuItem **950**
naming convention 814
Page **751**
Panel 12, **931**
PhoneApplicationPage (Windows Phone 8) 812
PictureBox **25**, **34**, 40
Polygon **784**, **977**
Polyline **784**, **977**
RadialGradientBrush **984**
RadioButton 12, 16, 17, **763**, 769, **934**, 942, 951, 980, 995
RadioButtonList **670**
Rectangle **784**, **976**
RegularExpressionValidator **674**
RequiredFieldValidator **673**, **674**
RichTextBox **947**
RotateTransform **794**, **987**
Controls (cont.)
ScaleTransform **794**, **987**
ScriptManager **875**
Separator **951**
SkewTransform **794**, **987**
Slider **955**
Slider (Windows Phone 8) 811, **813**
SolidColorBrush **787**
StackPanel **762**, **936**
Storyboard **1000**
TabContainer **875**
TextBlock 16, **970**, **973**
TextBlock (Windows Phone 8) **813**
TextBox 793, **944**, 986
TextBox (Windows Phone 8) **813**
TextDecoration **975**
ToolBar **950**
ToolBarTray **951**
ToolkitScriptManager **875**
TranslateTransform **794**, **987**
ValidatorCalloutExtender **876**
VisualBrush **983**
Window **930**, 939, 958
WrapPanel 934
Controls property of class GroupBox 404, **405**
Controls property of class Panel 404
ControlTemplate (Windows 8 UI) 797, 798, 800
ControlTemplate (WPF) 995
ControlTemplate class (WPF) **963**
TargetType property **963**
Triggers property **963**
ControlToValidate property of an ASP.NET validation control **674**
converge on a base case 185
convert
a binary number to decimal 1103
a hexadecimal number to decimal 1103
an integral value to a floating-point value 165
an octal number to decimal 1103
Convert class **64**, 67, 237
ToDecimal method **98**
ToInt32 method 237
cookie **678**
deletion 678
expiration **678**
expiration date **678**
header 678
Copy method of class Array **583**
Copy method of class File 521
copying objects
shallow copy **315**
CopyTo method of class string 494
Cos method of Math 157
cosine 157
Count extension method of interface IEnumerable<T> **249**
Count method (LINQ) 528
Count property
of Hashtable **596**
Count property of class ArrayList 585, **588**
Count property of class HttpSessionState 683, **685**
Count property of class List<T> **252**

counter **110**
counter-controlled repetition **110**, 116, 119, 127, 128
  with the for repetition statement 129
  with the while repetition statement 127
covariance **604**
covariant
  interface 604
craps (casino game) 167, 172
Craps class simulates the dice game craps 172
create a reusable class 481
create an object of a class 77
Create and write to a sequential-access file 532
Create method of class File 521
CreateCloudTableClient method of class CloudStorageAccount **917**
CreateDirectory method of class Directory **521**
CreateIfNotExists method of class CloudTable **917**
CreateInstance method of class Array **584**
CreateText method of class File 521
creating a child Form to be added to an MDI Form 473
creating a generic method 574
Creating a WCF Service in Visual Studio Express 2012 for Web 885
creating and initializing an array 196
creating GradeBook object 113, 118
credit inquiry 543
cropping **1001**
.cs file name extension 30
.csproj file extension 41
CSS attribute 662
CSS class **662**
Ctrl key 141, 424
Ctrl + *z* 141
Current property of IEnumerator **583**
current time 488
CurrentValue property of class ItemCheckEventArgs 451
cursor 52, 59
custom control **485**, 486
  creation 486, 488
Custom palette 37
Custom tab **37**
Custom value of enumeration DateTimePickerFormat **439**
CustomFormat property of class DateTimePicker **440**
customize a Form 31
customize Visual Studio IDE 27
Customizing gradients 793, 986

D

D format specifier 98, 201
dangling-else problem **107**
data abstraction **280**
data binding **615**, 1001
data binding (Windows 8 UI) 771, 775
  to LINQ 775
data binding (WPF) **965**
  to LINQ 965
data hierarchy **518**
data in support of actions 280
data provider (WPF) **971**
data representation **280**
data source 243
data source (entity data model) 622
Data Source Configuration Wizard **622**
Data Sources window **622**
data structure **193**
data template 776
data template (WPF) **968**
data validation **88**
database **519**, **608**
  add to a project 617
  saving changes in LINQ to Entities 627
  schema **610**
database connection **617**
database management system (DBMS) **519**, **608**
database schema **610**, 614
database table **609**
DataBind method of a GridView **693**
DataContext class 620
  SaveChanges method **615**
DataContext of WPF controls **968**
DataContract attribute **890**
DataContractJsonSerializer class **891**
DataGridView class **616**
DataGridView control 616, 622
DataMember attribute 890, **895**
DataSource property
  BindingSource class **627**
DataTemplate class (Windows 8 UI) 771, 777
DataTemplate class (WPF) **968**
data-type attribute (XPath) **726**
Date property of a DateTime **439**
DateChanged event of class MonthCalendar **438**, 439
DateTime structure **488**
  AddDay method **442**
  AddYears method **442**
  DayOfWeek property **442**
  Now property **488**, 665
  ToLongDateString method **442**
  ToLongTimeString method **488**
DateTimePicker class **439**
  CalendarForeColor property 440
  CalendarMonthBackground property 440
  CustomFormat property **440**
  Format property **439**, 440
  MaxDate property 440, **442**
  MinDate property 440, **442**
  ShowCheckBox property 440
  ShowUpDown property 440
  Value property **439**, 440, 442
  ValueChanged event **440**
DateTimePickerFormat enumeration **439**
  Custom value 439
  Long value 439
  Short value 439
  Time value 439
DayOfWeek enumeration **442**
DayOfWeek property of structure DateTime **442**
DB2 608
DbContext class **615**, 626
  SaveChanges method 627
DBCS (double byte character set) **1113**
DBExtensions class **627**
  Load extension method **627**, 629
DBMS (database management system) **519**, **608**
dealing 206
debug a web application 664
DEBUG menu 27
Debugging 27
decimal 94
decimal (base 10) number system 1098
decimal digit **517**
decimal point 118
decimal simple type 63, **93**, 1095
DecimalPlaces property of class NumericUpDown **419**
decision **67**, 105
decision in the UML 1031
decision symbol **105**
declaration
  class 49, **51**
  method **51**
declarative programming **241**, **749**, **929**, 929
declare a constant 198
declare a method of a class **74**
decrement **127**
decrement operator, -- 122
default
  case in a switch 141
  keyword 141
default case 171
default constructor 270, 293
default event of a control **395**
default initial value of a field **85**
default namespace **707**
  demonstration 707
default namespace (XAML) 752, 930
default settings 13, 17
default type constraint (object) of a type parameter 570
default value 125
default value for optional parameter **182**, 183
deferred execution **254**
definite repetition 110
definitely assigned **112**, 175
Delegate 845
delegate **394**
  Delegate class **395**
  MulticastDelegate class **395**
  registering an event handler **394**
Delegate class **395**
delegate keyword **394**
Delete method of class Directory 521
Delete method of class File 521, **529**
delimited comments **48**
denial-of-service (DoS) attack 921
dependency property (Windows 8 UI)
  attached property **769**
dependency property (WPF) **957**, 963
  attached property **937**
dependent condition 149
deployment diagram in the UML **1107**

Deposit class (ATM case study) 1020, 1023, 1026, 1034, 1042, 1043, 1051, 1054, 1058, 1059, 1060
DepositSlot class (ATM case study) 1020, 1022, 1026, 1034, 1043, 1055
derived class **286**, 1059, 1060
descendant node of a DOM tree **728**
Descendants method of class XContainer **738**
descending modifier of a LINQ orderby clause **244**
descriptive words and phrases (OOD/UML case study) 1025, 1027
deselected state **410**
Deserialize method of class BinaryFormatter **548**
deserialized object **548**
design mode 43
design process **4**, 1010, 1016, 1035, 1040
design specification **1016**
Design view **23**, 35, 931
destructor 275
Development 749
development storage account 914
diacritic **1112**
dialog **23**
DialogResult enumeration **415**, **537**
diamond **103**, 105
dice game 172
Dictionary **1006**
dictionary **597**, 683
Dictionary(Of K, V) generic class **528**, 580
  Add method **528**
  Clear method **529**
  ContainsKey method **528**
  Keys property **529**
  Remove method **529**
DictionaryEntry structure **596**
digit 63, 1098
direct base class 286, 288
direct events, (WPF) **946**
Directory class **521**, 525
  CreateDirectory method **521**
  GetFiles method **528**
Directory class methods (partial list) 521
DirectoryInfo class **467**, **521**
  Exists method **468**
  FullName property **467**
  GetDirectories method **467**
  GetFiles method **467**
  Name property **467**
  Parent property **468**
Dispatcher **1007**
DispatcherTimer class **959**
  Interval property **960**
  IsEnabled property **960**
  Tick event **960**
display output 71
Display property of an ASP.NET validation control **674**
displaying line numbers in the IDE xxxii
DisplayMemberBinding property of GridViewColumn class **968**
Dispose method of interface IDisposable 353, 376
distance between values (random numbers) 171
Distinct extension method of interface IEnumerable<T> **249**
Distinct query operator (LINQ) 528
distributed computing 881
divide by zero 362, 364
DivideByZeroException class **362**, 364, 365, 368
division 65
division by zero is undefined 280
.dll file **56**, **482**
DNS (Domain Name System) server **651**
do keyword **136**
do...while repetition statement 104, 136, 137
dock a control **399**
Dock property of class Control 400, 624
docking demonstration 400
DockPanel control (WPF) 934
  LastChildFill property 934
DOCUMENT (representing a Web Form Properties window) **660**
Document navigation using XNode 732
Document Object Model (DOM) tree **728**
Document Outline window 783, 784, 812, 813, 818
Document property of class XObject **740**
document root **722**
Document Type Definition (DTD) **698**, 704, 708
  for a business letter 708
DOM (Document Object Model) tree **728**
  ancestor node **728**
  child node **728**
  descendant node **728**
  parent node **728**
  root node **728**
  sibling node **728**
DOM parser **728**
domain name system (DNS) server **651**
dotted line in the UML 103
Double 564
double-byte character set (DBCS) **1113**
(double) cast **117**
double equals, == 67
double-precision floating-point number **94**
double quote, " 51, 59, 60
double quotes ("") to delineate a string 704
double-selection statement 104
double simple type **63**, 114, 1095
Double.NaN **362**, 383
Double.NegativeInfinity **362**
Double.PositiveInfinity **362**
DoubleAnimation control **1001**
  Duration property **1001**
  From property **1001**
  To property **1001**
down-arrow button 37
downcast 339, 596
downcasting **321**
DownloadDataTaskAsync method of class WebClient **852**
DownloadStringTaskAsync method of class WebClient **850**
drag the mouse **32**
DragMove method of Window control **959**
Draw event of class ToolTip 418
draw on control 486
DrawEllipse method of class Graphics **456**
Drawing Polylines and Polygons 979
DrawPie method of class Graphics **457**
DrawPolygons example 977
DrawRectangle method of class Graphics **457**
drop-down list 388, **453**
DropDown value of enumeration ComboBoxStyle 454
DropDownList ASP.NET web control **669**
DropDownList value of enumeration ComboBoxStyle 454
DropDownStyle property of class ComboBox 453, **454**
DropShadowEffect **995**
DTD (Document Type Definition) **698**, 704
.dtd filename extension **704**
DTD repository 708
dummy value **114**
Duration property of DoubleAnimation control **1001**
dynamic binding **339**
dynamic resizing **240**
dynamic resizing of a List collection 251
dynamic resource (Windows 8 UI) **764**
dynamic resource (WPF) **956**
dynamically linked library **56**, **482**

E

E formatting code 98
EDIT menu 26
editable list 454
EF (ADO.NET Entity Framework) **609**
efficient (Unicode design basis) **1110**
element (XAML) 751, 929
element (XML) **696**
ELEMENT element type declaration (DTD) **709**
Element method of class XContainer **738**
element of an array **193**
element of chance 167
element type declaration **709**
element XML Schema element **714**
ElementName property of Binding class **775**, **965**
Elements method of class XContainer **731**
Elements method of class XElement **731**
element-to-element binding **956**
elided UML diagram **1021**
eligible for destruction **274**, 278
eligible for garbage collection **275**, 278
eliminate resource leak 370
Ellipse class (Windows 8 UI) **769**
Ellipse control **976**
Ellipse control (Windows 8 UI) **784**
ellipsis button **38**
Employee abstract base class 329, 348

Employee class with FirstName, LastName and MonthlySalary properties 245
Employee class with references to other objects 273
Employee hierarchy test application 336
empty element 704
EMPTY keyword (DTD) **710**
empty statement (semicolon by itself) **70**, 137
empty string
  "" **90**
  string.Empty **90**
EmptyStackException indicates a stack is empty 569
Enabled property of class Control **398**
encapsulation **4**, **84**, 88, 342
encoding **1110**
encoding attribute in xml declaration **700**
"end of data entry" 114
end-of-file (EOF)
  indicator **141**
end-of-file marker **519**
end tag **696**
end tag (XAML) 751, 929
EndEdit method of class BindingSource **627**
endpoint (of a WCF service) **881**
EndPoint property of LinearGradientBrush control **791**, **984**
endpointBehaviors element in web.config **886**
EndsWith method of class string 497, 498
EnsureCapacity method of class StringBuilder 505
Enter (or *Return*) key 35, 52
enter data from the keyboard 388
entities in an entity data model 619
entity
  &amp; 710
  &apos; 710
  &gt; 710
  &lt; 710
  &quot; 710
entity connection string 618
Entity Data Model 609, **614**, 626
  ADO.NET Entity Framework **614**
  create from database 616
  data source for data binding 622
  entities 619
EntityDataSource ASP.NET data control **689**
entity-relationship diagram **613**
entry point of an application **158**
enum **175**
  keyword 175
Enumerable class 615
enumeration **175**
  ComboBoxStyle 454
  DateTimePickerFormat 439
  DayOfWeek 442
  MdiLayout 475
  SelectionMode 446
  XmlNodeType 740
enumeration constant **175**
enumerator **578**, 583
  fail fast 584
  of a LinkedList 602
equal likelihood 169
Equals method of an anonymous type 635
Equals method of class object 315
Equals method of class string 495
Error List window **57**
Error property of class Console 519
ErrorMessage property of an ASP.NET validation control **674**
escape character **59**
escape sequence **59**, 63
  carriage return, \r 60
  escape character, \ 60
  horizontal tab, \t 60
  newline, \n 60, 63
event **390**, 1029
event driven **5**, **390**
event handler **390**, 394
event handling **390**
  Windows 8 UI 765
  WPF 938
event handling model 390
event multicasting **395**
event routing (WPF) 943
event sender **394**
EventArgs class 392, 769, 942
events **5**
events at an interval 487
EventSetter class (Windows 8 UI) 777
EventSetter class (WPF) **955**
exception 64, 203, **205**, **359**
  ArgumentException 595
  ArgumentOutOfRangeException 588
  handler **205**
  handling 203
  IndexOutOfRangeException 205
  InvalidCastException 340, 579, 596
  InvalidOperationException 584, 591, 602
  KeyNotFoundException 599
  Message property **206**
  parameter 205
Exception Assistant **366**
Exception class **368**, 368
exception handler **359**, 368
exception handling 64
.exe file name extension **56**
executable **56**
executable file 7
execute an application 52
Execute method of class CloudTable **919**
Execute method of ICommand interface **947**
Executed event of CommandBinding class **948**
ExecuteQuery method of class CloudTable **918**
ExecutionEngineException class **368**
exhausting memory 187
Exists method of class Directory 521
Exists method of class DirectoryInfo **468**
Exit method of class Application **437**, 450
exit point
  of a control statement 104
Exp method of Math 157
expand a tree 30
Expand method of class TreeNode 459
expand node 457
ExpandAll method of class TreeNode 459
Expander control (WPF) 934
expiration date of a cookie **678**
explicit conversion **117**
explicit type argument **563**
explicitly shut down a Windows 8 UI app 19
exponential complexity **838**
exponential method 157
exponentiation operator 135
expression **65**
extend a class **286**
extend an XML Schema data type 717
extender **876**
extensibility 321
Extensible Application Markup Language (XAML) **748**, **928**
Extensible HyperText Markup Language (XHTML) **719**
extensible language **77**
Extensible Stylesheet Language (XSL) **699**, 708, **719**
extension element
  base attribute **717**
extension method 249, 615
extension methods (Linq to Entities) 626
extension XML Schema element **717**
external DTD **704**

F

F formatting code 98
factorial 185
Factorial method 185, 186
false keyword **67**, 105, 106
fault-tolerant program **205**, **359**
Fibonacci series 838
field **82**, **518**
  in a database table **609**
field width **134**
fields **157**, 177
FIFO (first-in, first-out) 281
file 519
  as a collection of bytes 519
File class **520**, 524
  Delete method **529**
File class methods (partial list) 521
FILE menu 26
File name extensions
  .aspx **650**
  .cs 30
  .csproj 41
File Name property of a file in the **Solution Explorer** **54**
FileAccess enumeration 537
FileInfo class **467**
  FullName property **467**
  Name property **467**
file-position pointer 542
files **517**

FileStream class **520**, 537, 542, 553
  Seek method 548
Fill property of a shape control **784**, **976**
Fill value of Stretch property **1001**
FillEllipse method of class Graphics **424**, **457**
FillPie method of class Graphics **457**
FillRectangle method of class Graphics **457**
filter a collection using LINQ **240**
final state in the UML 103, 1031
final value 128
Finalize method
  of class object 315
finally block **365**, 370
finally blocks always execute, even when no exception occurs 371
Find method of class LinkedList **603**
Finished design of MasterDetail app 640
First extension method of interface IEnumerable<T> **249**
first-in, first-out (FIFO) data structure **281**
FirstDayOfWeek property of class MonthCalendar 439
FirstNode property of class TreeNode 458
Fisher-Yates shuffling algorithm 209
#FIXED keyword (DTD) **709**
fixed text 65
  in a format string **61**
flag **114**
FlatStyle property of class Button 402
Flickr 846
Flickr API key 846
float simple type 63, **93**, 1094
floating-point literal **94**
floating-point number 113, 114, 116
  division 117
  double precision **94**
  double simple type **93**
  float simple type **93**
  single precision **94**
floating-point **93**
floating-point literal
  double by default 94
Floor method of Math 157
flow of control 109, 116
  in the if statement 105
  in the if...else statement 106
  in the while statement 109
flow-based layout **759**, **932**
focus **389**
Focus method of a control (Windows Phone 8) **822**
Focused property of class Control 398
Font class **409**
  Style property **409**
Font dialog **38**
Font property of a Label **38**
Font property of class Control 398
Font property of class Form 390
font size 38
font style 38, 408
Font window 39
FontFamily property of a TextBlock **753**
FontFamily property of TextBlock control **974**
Fonts
  Arial 975
  Courier New 975
  Segoe UI 975
  Times New Roman 975
FontSize property of a TextBlock **753**
FontSize property of TextBlock control **975**
FontStyle enumeration **409**
FontStyle property of TextBlock control **975**
FontWeight property of TextBlock control **975**
for keyword **128**
for repetition statement 104, 128, 131, 133, 134
  activity diagram 131
  header **129**
foreach statement **210**, 584, 586, 591
  on rectangular arrays 232
ForeColor property of class Control 398
Foreground property of TextBlock control **787**, **983**
foreign key **612**, 614
form **25**
form background color 36
Form class 389
  AcceptButton property 390
  ActiveMdiChild property 474
  AutoScroll property 390
  CancelButton property 390
  Close method 390
  Font property 390
  FormBorderStyle property 390
  Hide method 390
  IsMdiChild property 474
  IsMdiContainer property **473**, 474
  LayoutMdi method 474, **475**
  Load event 390
  MaximumSize property **400**
  MdiChildActivate event 474
  MdiChildren property 474
  MdiParent property **473**
  MinimumSize property **400**
  Padding property **400**
  Show method 390
  Text property 390
Form control (Windows Forms) 930
Form properties, methods and events 390
Form1.cs 43
format item **61**
FORMAT menu 27
Format method of String **259**
Format property of class DateTimePicker **439**, 440
format specifier **98**
  F for floating-point numbers 118
format specifier D 201
format specifier table 98
format string **61**
FormatException class **362**, 364, 365
FormatString 509
formatted output
  field width **134**
  left justify **134**
  right justify **134**
formatting
  display formatted data 60
FormBorderStyle property of class Form 390
forward slash character (/) in end tags **697**, 722
fragile software **307**
frame (in the UML) 1047
Framework Class Library 564
from clause of a LINQ query **243**
From property of DoubleAnimation control **1001**
FromArgb method of class Color 793, 955
FromStream method of class Image 852
FullName property of class DirectoryInfo **467**
FullName property of class FileInfo **467**
FullName property of class Type 315
FullPath property of class TreeNode 458
FullStackException indicates a stack is full 569
fully qualified class name 82, 165, 390, 391, 482
Func<TResult> delegate **838**
function key **427**

G

G formatting code 98
game playing 167
garbage collection 370
garbage collector **274**, **275**
general catch clause **365**
general class average problem 113
generalization in the UML **1059**
generic class **250**, **558**, 567
Generic classes
  Dictionary 580
  LinkedList 580, **599**
  LinkedListNode **599**
  List 580
  Queue 580
  SortedDictionary 580, **597**, 598
  SortedList 580
  SortedSet<T> **605**
  Stack 580
generic interface **558**
Generic List<T> collection demonstration 251
generic method **558**, 561
  creating 574
  implementation 561
generics **558**
  class **558**
  class constraint **565**
  compile-time type safety **558**
  constructor constraint (new()) **566**
  default type constraint (object) of a type parameter 570
  interface **558**
  interface constraint **565**
  method 561
  overloading 566
  reference type constraint class **566**
  reusability 567
  scope of a type parameter 570

generics (cont.)
specifying type constraints **564**
Stack class 567
type argument **563**, 570
type checking **558**
type constraint of a type parameter **564**, 565
type parameter **562**
type parameter list **562**
value type constraint struct **566**
where clause **565**
get accessor of a property 74, 84, 87
get keyword **85**
get request (HTTP) **881**
GetCreationTime method of class Directory 521
GetCreationTime method of class File 521
GetCurrentPoint method of PointerRoutedEventArgs class **770**
GetDefaultView method of CollectionViewSource class **971**
GetDirectories method of class Directory 462, 521
GetDirectories method of class DirectoryInfo **467**
GetEnumerator method of interface IEnumerable **583**
GetExtension method of class Path **528**
GetFiles method of class Directory 521, **528**
GetFiles method of class DirectoryInfo **467**
GetHashCode method of class Object **593**
GetItemChecked method of class CheckedListBox 451
GetLastAccessTime method of class Directory 521
GetLastAccessTime method of class File 521
GetLastWriteTime method of class Directory 521
GetLastWriteTime method of class File 521
GetLength method of an array 227
GetNodeCount method of class TreeNode 459
GetObject method of class ResourceManager **417**
GetPosition method of MouseEventArgs class **943**
GetSelected method of class ListBox **447**
GetStringAsync (HttpClient) **889**
GetTableReference method of class CloudTableClient **917**
GetType method of class object 315, 340
GIF (Graphic Interchange Format) **41**
global namespace **82**
glyph 1112
Good Programming Practices overview xxvii
goto elimination 102
goto statement **102**
gradient **788**, **984**
GradientStop (Windows 8 UI) **792**
Color property **792**
Offset property **792**
GradientStop control **986**
Color property **986**
Offset property **986**
Grammar **1006**
graph information 201
Graphic Interchange Format (GIF) 41
graphical user interface (GUI) 23, 165, 387
Graphics class **424**, **456**
Clear method **456**
DrawEllipse method **456**
DrawPie method **457**
DrawRectangle method **457**
FillEllipse method **424**, **457**
FillPie method **457**
FillRectangle method **457**
Graphics property of class PaintEventArgs **485**, 486
Grid control (Windows 8 UI) **752**, 760
Column attached property **774**
ColumnDefinition class **774**
ColumnDefinitions property **774**
ColumnSpan attached property **775**
Row attached property **774**
RowDefinition class **774**
RowDefinitions property **774**
RowSpan attached property **775**
Grid control (Windows Phone 8) 813
ColumnDefinition 817
ColumnDefinitions property **817**
RowDefinition 817
RowDefinitions property **817**
Grid control (WPF) 931, **936**
Column attached property **937**
ColumnDefinition class **936**
ColumnDefinitions property **936**
ColumnSpan attached property **937**
Row attached property **937**
RowDefinition class **936**
RowDefinitions property **936**
RowSpan attached property **937**
GridView ASP.NET data control **686**
GridView class
DataBind method **693**
GridView control (WPF) **968**
GridViewColumn class (WPF) **968**
DisplayMemberBinding property **968**
GroupBox class 12
GroupBox control **404**
Controls property **405**
properties 404
Text property 404
GroupBox control (WPF) **936**
Header property **936**
guard condition 105, 106
guard condition in the UML 1031
guestbook on a website 685
GUI (graphical user interface) **23**, 387
GUI control 388
GUI control, basic examples 388

H

handle an event 394
handle an exception **363**
Handled property of RoutedEventArgs class **944**
has-a relationship **271**, **286**
has-a relationship (composition) **1022**
HasElements property of class XElement **731**
hash function **593**
hash table **592**
hashing **592**
Hashtable class **580**, **592**
ContainsKey method **595**
Count property **596**
method Add **595**
property Values **596**
Header property of GroupBox control **936**
height of a table row 813
Height property of a RowDefinition 817
Height property of a shape 976
Height property of a shape control 784
Height property of structure Size **400**
HELP menu 27, **33**
helper method **143**
HelpLink property of Exception 378
hexadecimal (base 16) number system 1098
"hidden" fields 177
hide implementation details 261, 279, 281
Hide method of class Control 398
Hide method of class Form 390
hiding implementation details 593
hierarchy 701
horizontal tab 60
HorizontalAlignment property of a TextBlock **752**
HorizontalAlignment property of Windows 8 UI controls **759**
HorizontalAlignment property of WPF controls **933**
host **651**
hostname **651**
hot key 429
HourlyEmployee class that extends Employee 331
HTML (HyperText Markup Language) 651
HTTP (HyperText Transfer Protocol) 651, 677
HTTP (Hypertext Transfer Protocol) request type **881**
http://www.w3.org/2001/XMLSchema (XML Schema URI) **714**
HttpClient **888**
GetStringAsync method **889**
HttpSessionState class **679**, 683, 684, 685
Add method 684
Count property 683
IsNewSession property 683
Keys property 683, 685
SessionID property 683
Timeout property 683
HttpUtility class
UrlEncode method **920**
HyperLink ASP.NET web control **670**
HyperText Markup Language (HTML) 651

HyperText Transfer Protocol (HTTP) 651, 677, 881

I

IBM
  DB2 608
IBM Corporation 1110
ICollection interface 579
ICommand interface (WPF) **947**
  CanExecute method **947**
  Execute method **947**
IComparable interface 352
IComparable<T> interface **564**, **605**
  CompareTo method **564**
IComparer<T> interface **605**
IComponent interface 352, **388**
icon 27
IDE (Integrated Development Environment) 11, 21
identifier **50**, 62
identity column in a database table **611**
identity management 908
IDictionary interface 579
IDisposable interface 353, 376
  Dispose method 353
IE10 (Internet Explorer 10) 9
IEC 60559 1094
IEEE 754 1094
IEnumerable interface 942
  method GetEnumerator **583**
IEnumerable<T> interface **245**, 579, **583**, 615
  Any extension method **249**, 739
  Count extension method **249**
  Distinct extension method **249**
  First extension method **249**
IEnumerator interface 353, **583**
if single-selection statement **67**, 104, 105, 137
  activity diagram 105
if...else double-selection statement 104, 105, 106, 116, 137
  activity diagram 106
ignoring array element zero 205
IIS Express **656**, 664
IIS Express web server 884
IList interface 579
Image ASP.NET web control **667**
Image Collection Editor 463
Image control (Windows 8 UI) **771**, 775
Image property of class PictureBox **40**, 415
image resource **417**
ImageBrush
  Stretch property **983**
ImageBrush (Windows 8 UI) **788**
  ImageSource property **788**
ImageBrush control (WPF) 958, **983**
  ImageSource property **983**
ImageIndex property of class ListViewItem **463**
ImageIndex property of class TreeNode 458
ImageList class **458**, 462
  Images property **463**
ImageList property of class TabControl 470
ImageList property of class TreeView 458
Images property of class ImageList **463**
ImageSource property of ImageBrush control **788**, **983**
ImageUrl property of an Image web control **667**
immutable (strings) **277**
immutable string 494
imperative programming **241**
implement
  an interface **319**, **343**, 350
  multiple interfaces 347
implementation-dependent code 261
implementation of a function 329
implementation phase 1063
implementation process 1035, 1053
implementing a dispose method (link to MSDN article) 353
implicit conversion **117**
implicit conversions between simple types 164
implicitly typed local variable **243**, **244**, 250
#IMPLIED keyword (DTD) **709**
in-app advertising **828**
In property of class Console 519
in-app products **827**
in-app purchase 810
increment **127**, 133
  a control variable 128
  expression 146
  of a for statement 131
increment and decrement operators 122
increment operator, ++ 122
Increment property of class NumericUpDown **419**
indefinite repetition **114**
indentation 107
independent software vendor (ISV) 314
index **193**, 203
Index property of class ItemCheckEventArgs 451
index zero **193**
indexer 494
  of a Hashtable 595
  of an ArrayList 588
IndexOf method of class Array **584**
IndexOf method of class ArrayList 585, **588**
IndexOf method of class List<T> 250
IndexOf method of class string 498
IndexOfAny method of class string 498
IndexOutOfRangeException class 203, **205**, 368
indirect base class 286, 288
infer a local variable's type 243
infinite loop 109, 116, 130, 187
infinite recursion **187**
infinity symbol 614
information hiding **4**, **84**, 279, 342
information tier **653**
infrastructure 908
inherit from class Control 486
inherit from Windows Form control 486
inheritance **4**, **286**, 290, 314, 1058, 1059, 1062, 1063
  examples 287
  hierarchy **287**
  hierarchy for university CommunityMembers 288
  single **286**
Inheritance hierarchy for class Shape 288
inheritance with exceptions 368
Init event of a Web Form **663**
Init event of an ASP.NET web page 665
initial state **103**
initial state in the UML **1030**, 1031
initial value of an attribute 1027
initial value of control variable **127**
InitialDelay property of class ToolTip 418
initializer list **197**
Initializing jagged and rectangular arrays 226
initializing readonly arrays 586
Initializing the elements of an array with an array initializer 197
initializing two-dimensional arrays in declarations 226
inlining method calls **342**
InnerException property of Exception **377**, 381
innermost set of brackets 204
input data from the keyboard 71
input–output operations 103
InputGestures property **947**
InputScope property of a TextBox (Windows Phone 8) 818
Insert method of class ArrayList 585
Insert method of class List<T> **251**
Insert method of class TableOperation **919**
Insert Separator option 432
Insert Snippet window 89
inserting separators in a menu 432
installing Windows Azure SDK for Visual Studio 2012 909
instance **3**
instance of a class 83
instance variable **74**, 82, 83, 94, 157
instance variables **4**
instantiate an object of a class **74**
int simple type **63**, 114, 122, 1094
Int32 564
integer **61**
  division **65**, **113**
  mathematics 280
  value **63**
integer array 197
integer division without exception handling 360
integer division yields an integer result 117
Integrated Development Environment (IDE) **11**, 21
IntelliSense **54**, 243, 609, 614, 698
interaction diagram in the UML **1043**
interaction overview diagram in the UML **1108**
interactions among objects 280, 1040, 1044
interest rate 133
interface **245**, 319, 344, 350
  declaration **343**

interface constraint **565**
interface keyword **343**, 351
Interfaces
  ICollection 579
  ICommand **947**
  IComparable 352, **564**
  IComparable<T> **605**
  IComparer<T> **605**
  IComponent 352, **388**
  IDictionary 579
  IDisposable 353, 376
  IEnumerable 579, **583**, 615, 942
  IEnumerable<T> **245**, 615
  IEnumerator 353, **583**
  IList 579
  IQueryable<T> **615**
  ISerializable **549**
internal data representation 281
Internet Explorer 701
Internet Explorer 10 (IE10) 9, 809
Internet Information Services (IIS) 651
Interval property of class Timer **487**
Interval property of DispatcherTimer class **960**
Introduction to Cloud Computing Faculty Resource Kit 910
InvalidCastException 340
InvalidCastException class 579, 596
InvalidOperationException class 584, 591, 602
Invoice class implements IPayable 346
invoke a method **90**
Invoke method of class Control **845**
InvokeRequired property of class Control **845**
iOS 8
IP address **651**
IPayable interface declaration 345
IPayable interface hierarchy UML class diagram 345
IQueryable<T> interface **615**
is-a relationship **286**, 320
is operator **339**
IsAsync property of Binding class **971**
IsAsynchronous property of ObjectDataProvider class **971**
IsAsynchronous property of XmlDataProvider class **971**
IsChecked property of a RadioButton 763
IsDigit method of struct Char 514
IsEnabled property of DispatcherTimer class **960**
ISerializable interface **549**
IsLetter method of struct Char 514
IsLetterOrDigit method of struct Char 514
IsLower method of struct Char 514
IsMdiChild property of class Form 474
IsMdiContainer property of class Form 473, 474
IsMouseOver property of WPF controls **963**
IsNewSession property of class HttpSessionState 683
IsOpen property of the AppBar control **762**
IsPostBack property of class Page **676**
IsPunctuation method of struct Char 514
IsSymbol method of struct Char 514
IsUpper method of struct Char 514
IsValid property of class Page **676**
IsWhiteSpace method of struct Char 514
Italic FontStyle 975
ItemActivate event of class ListView 463
ItemCheck event of class CheckedListBox **450**, 451
ItemCheckEventArgs class 451
  CurrentValue property 451
  Index property 451
  NewValue property 451
Items property of class ComboBox 453, **454**
Items property of class ListBox 446, **447**
Items property of class ListView 463
ItemSize property of class TabControl 470
ItemsSource property of ListView **776**, **967**
ItemTemplate property of ListView control **775**, 777, **968**
iterating (looping) **112**
iteration
  of a loop **127**, 146
iteration (looping)
  of a for loop 204
iteration statement 104
iteration variable **210**
iterative 187
iterative model **1015**

J

jagged array **223**, 224, 226
JavaScript 670
JavaScript Object Notation (JSON) **882**
JIT (just-in-time) compiler 7
joining database tables **613**, 634
  LINQ to Entities 633
Joint Photographic Experts Group (JPEG) 41
JPEG (Joint Photographic Experts Group) 41
JSON (JavaScript Object Notation) **882**
JSON serialization **891**
just-in-time (JIT) compiler 7

K

key code **427**
key data 427
key event **424**, 425
key value 427
key–value pair **683**
keyboard 61, 388
keyboard shortcuts **429**
KeyChar property of class KeyPressEventArgs **424**
KeyCode property of class KeyEventArgs 425
KeyData property of class KeyEventArgs 425
KeyDown event (Windows 8 UI) 770
KeyDown event (WPF) 943
KeyDown event of class Control **424**, 425
KeyEventArgs class **424**
  Alt property 425, 427
  Control property 425, 427
  KeyCode property 425
  KeyData property 425
  KeyValue property 425
  Modifiers property 425
  Shift property 425, 427
KeyEventArgs properties 425
KeyNotFoundException class **599**
Keypad class (ATM case study) 1017, 1020, 1022, 1034, 1041, 1042, 1043, 1046, 1055
KeyPress event of class Control **424**, 425
KeyPressEventArgs class **424**
  KeyChar property **424**, 425
KeyPressEventArgs properties 425
KeyRoutedEventArgs class (Windows 8 UI) 770
keys
  function **427**
  modifier **424**
Keys enumeration **424**
Keys property of class Dictionary **529**
Keys property of Hashtable **595**
Keys property of HttpSessionState class 683, **685**
KeyUp event (Windows 8 UI) 770
KeyUp event (WPF) 943
KeyUp event of class Control **424**, 425
KeyValue property of class KeyEventArgs 425
Keywords **49**, 104
  abstract 303, **325**
  as 340
  async **834**, 838
  await **834**, 838
  base **289**, **302**, 311, 313
  bool **106**
  break **141**
  case **141**
  char **63**
  class **49**, 75
  const 145, **157**, **198**, 278
  continue **145**
  decimal **63**, **93**
  default **141**
  delegate **394**
  do **136**
  double **93**
  else 105
  enum **175**
  false **106**
  float **63**
  for **128**
  get **85**
  if **67**, 105
  interface **343**
  is **339**
  namespace 482
  new 77, 91, 195, 196, 225
  null 86, **90**, 125, 195
  operator **354**
  out **188**
  override **206**, **294**, 302, 303

Keywords (cont.)
params **234**
partial 392, **665**
private **84**, 261, 289, 1053
protected 261, **289**
public **50**, **75**, 84, 158, 261, 289, 1053
readonly **279**
ref **188**, 212
return **85**, 162
sealed **342**
set **85**
static **135**
struct **512**
switch **141**
this **262**, 278
true **106**
value (contextual) **85**
var **243**
virtual **303**
void **51**, 75
while 109, 136
Koenig, Andrew 359

L

Label 38
label 401
Label class **25**, **34**, 37
Label control 388, 401
Windows 8 UI **752**
Windows Forms 752
WPF **931**
label in a switch **141**, 142
lambda expression **626**, 631, 632, 834, 838
lambda operator **626**
lambda operator **626**
language independence 7
Language Integrated Query (LINQ) **240**
language interoperability 7
LargeImageList property of class ListView **463**
last-in first-out (LIFO) order 573
Last property of class LinkedList **603**
LastChildFill property of DockPanel control 934
LastIndexOf method of class Array **584**
LastIndexOf method of class string 498, 500
LastIndexOfAny method of class string 498
last-in-first-out (LIFO) **163**
last-in-first-out (LIFO) data structure 280
LastNode property of class TreeNode 458
late binding **339**
layout container (Windows 8 UI) **752**, 759, 769
layout container (WPF) **931**, 933, 942
layout, control 397
LayoutMdi method of class Form 474, **475**
leading 0 **201**
Left attached property of Canvas control **769**, **937**, 942
left brace, { **51**, 62
left justify output **134**
Length property of an array 194
Length property of class string 494, 495
Length property of class StringBuilder 505
let clause of a LINQ query **254**
letter **517**
lifeline of an object in a UML sequence diagram **1045**
LIFO (last-in-first-out) order 163, 280, 573
Line control **976**
Line control (Windows 8 UI) **784**
line numbers xxxii
LinearGradientBrush (Windows 8 UI) **788**
EndPoint property **791**
StartPoint property **791**
LinearGradientBrush control **984**
EndPoint property **984**
StartPoint property **984**
LinkArea property of class LinkLabel 443
LinkBehavior property of class LinkLabel 443
LinkClicked event of class LinkLabel 442, 443
LinkColor property of class LinkLabel 443
LinkedList generic class 580, **599**
AddFirst method **602**
AddLast method **602**
method Find **603**
method Remove **603**
property First 603
property Last **603**
LinkedListNode class
property Next **599**
property Previous **599**
property Value **599**
LinkedListNode generic class **599**
LinkLabel class 429, **442**
ActiveLinkColor property 443
LinkArea property 443
LinkBehavior property 443
LinkClicked event 442, 443
LinkColor property 443
LinkVisited property 443
Text property 443
UseMnemonic property 443
VisitedLinkColor property 443
LinkLabel properties and an event 443
LinkVisited property of class LinkLabel 443
LINQ (Language Integrated Query) **240**, 524
anonymous type **249**
ascending modifier **244**
Count method 528
deferred execution **254**
descending modifier **244**
Distinct query operator 528
from clause **243**
let clause **254**
LINQ to Entities 241, **609**
LINQ to Objects **240**, 609
LINQ to XML 241, **696**, **729**
orderby clause **244**
provider **241**
LINQ (Language Integrated Query) (cont.)
query expression **240**
range variable **243**
Resource Center 647
select clause **244**
where clause **244**
LINQ provider **241**
LINQ query syntax 615
LINQ to Entities 241, **609**, 610, 626
data binding **615**
DbContext class **615**, **620**
extension methods 626
Object data source **622**
primary keys 609
saving changes back to a database 627
LINQ to Objects **240**, 525, 609, 615, 739
using a List<T> 253
using an array of Employee objects 246
using an int array 241
LINQ to SQL 609
LINQ to XML 241, **696**, 729, 850, 851
LINQ usage throughout the book 240
LINQ, data binding to 775, 965
LINQPad (www.linqpad.net) 647
Linux 8
list, editable 454
List<T> generic class **250**, 580
Add method **251**
Capacity property 250, **253**
Clear method 250
Contains method 250, **253**
Count property **252**
IndexOf method 250
Insert method **251**
Remove method 250, **252**
RemoveAt method 251, **252**
RemoveRange method 251
Sort method 251
TrimExcess method 251
List<T> methods 250
ListBox control 388, 429, **446**
ClearSelected method 447
GetSelected method 447
Items property **447**
MultiColumn property 446
SelectedIndex property **447**
SelectedIndexChanged event **446**
SelectedIndices property **447**
SelectedItem property **447**
SelectedItems property **447**
SelectionMode property **446**
Sorted property 447
ListBox properties, method and event 446
ListBox.ObjectCollection class 447
ListView control **462**
Activation property **462**
CheckBoxes property **462**
ItemActivate event 463
Items property 463
LargeImageList property **463**
MultiSelect property **462**
SelectedItems property 463
SmallImageList property **463**
View property **462**

ListView control (Windows 8 UI) 771
 ItemsSource property 776
 ItemTemplate property 775, 777
ListView control (WPF) 965
 ItemsSource property 967
 ItemTemplate property 968
 SelectedItem property 965
ListView displaying files and folders 464
ListView properties and events 463
ListViewItem class 463
 ImageIndex property 463
literal 51
literals
 decimal 94
 floating point 94
 int 94
live-code approach xxvi
Load event 675
Load event of an ASP.NET web page 675
Load event of class Form 390
Load extension method of class DBExtensions 627, 629
load factor 592
Load method of class XDocument 729
Load method of class XslCompiledTransform 745
local variable 82, 112, 177, 179, 264
local variable "goes out of scope" 517
localization 1110
localize an application 1110
LocalName property of class XName 731
Location property of class Control 400
Location property of TextDecoration control 975
Log method of Math 157
logarithm 157
logic error 64, 129
logical negation, ! 150
 operator truth table 150
logical operators 147, 150
logical output operation 520
logical point 791, 984
logical tree (WPF) 960
logical XOR, ^ 149
login (ASP.NET) 694
long simple type 1094
long-term retention of data 517
Long value of enumeration DateTimePickerFormat 439
lookless control (WPF) 960
loop 110, 112
 body 136
 continuation condition 104
 counter 127
 infinite 109, 116
loop-continuation condition 127, 128, 132, 136, 146
lower camel casing 63
lowercase letter 49

M

m-by-*n* array 224
Mac OS X 8
machine-independent pixel 781, 973
magic numbers 199
magnetic disk 517
magnetic tape 517
Main method 51, 52, 62, 77
MainPage.xaml 757
make your point (game of craps) 172
managed code 7
manifest file 825
many-to-many relationship 614, 619
many-to-one relationship 1024
Margin property of a Canvas 765
Margin property of a TextBlock 753
Margin property of Windows 8 UI controls 759
Margin property of WPF controls 933
markup extension (XAML) 764, 775, 956, 963, 965
markup in XML 929
mask the user input 401
master page in ASP.NET 857
master pages 694
master/detail view 637
match attribute 722
Match class 491, 514
Math class 156
 Abs method 156
 Ceiling method 157
 Cos method 157
 E constant 157
 Exp method 157
 Floor method 157
 Log method 157
 Max method 157
 Min method 157
 PI constant 157
 Pow method 135, 136, 156, 157
 Sin method 157
 Sqrt method 156, 157, 164, 383
 Tan method 157
Math tutor using EquationGeneratorServiceXML to create equations 897
MathML 697
Max method of Math 157
MaxDate property of class DateTimePicker 440, 442
MaxDate property of class MonthCalendar 439
MaxDropDownItems property of class ComboBox 453
MaxHeight property of Windows 8 UI controls 759
Maximum method 158
Maximum property of a Slider 822
Maximum property of class NumericUpDown 419
MaximumSize property of class Control 400
MaximumSize property of class Form 400
maxOccurs XML Schema attribute 715
MaxSelectionCount property of class MonthCalendar 439
MaxWidth property of Windows 8 UI controls 759
MaxWidth property of WPF controls 933
MBCS (multi-byte character set) 1113
.mdf file extension 610
MDI (Multiple Document Interface) 473
MDI (Multiple Document Interface) window 388
MDI child 480
MDI parent and MDI child properties, method and event 474
MDI parent window class 477
MDI title bar 475
MdiChildActivate event of class Form 474
MdiChildren property of class Form 474
MdiLayout enumeration 475
 ArrangeIcons value 475
 Cascade value 475
 TileHorizontal value 475
 TileVertical value 475
MdiParent property of class Form 473
MdiWindowListItem property of class MenuStrip 475
MediaElement (Windows 8 UI) 805
 Source property 805
MediaElement control 983
 Source property 983
 Stretch property 1001
member access (.) operator 135, 156, 276
member access operator 77
membership capabilities (ASP.NET) 857
MemberwiseClone method of class object 315
memory consumption 578
memory leak 275, 369
MemoryStream class 520
menu 26, 387, 429
 BUILD 26
 DEBUG 27
 EDIT 26
 FILE 26
 FORMAT 27
 HELP 27, 33
 PROJECT 26
 TEAM 27
 TEST menu 27
 TOOLS 27
 VIEW 26, 28
 WINDOW 27
menu access shortcut 429
menu access shortcut, create 430
menu bar 26, 387
 in Visual Studio IDE 26
Menu control (WPF) 950
menu item 26, 429
menu, ellipsis convention 432
menu, expanded and checked 430
MenuItem control (WPF) 950
MenuItem property MdiWindowListItem example 476
MenuStrip class 430
 MdiWindowListItem property 475
 RightToLeft property 433
MenuStrip properties and events 433
merge in the UML 1031
merge symbol in the UML 109
message 51
message (sent to an object) 74
message in the UML 1040, 1043, 1044, 1045
message passing in the UML 1045
Message property of an exception object 206
Message property of Exception 374, 377

method **3**, **51**, 79
  call **74**
  header **76**
  in the UML 4
  parameter 80
  parameter list **80**
  return type **75**
  static **135**
method call 4, 160
method call stack **163**, 377
method declaration 160
method overloading 155, **179**, 180
method parameter list 234
MethodInvoker delegate 845
methods implicitly sealed 342
methods of class List<T> 250
Metro 9
Microsoft 1110
  SQL Server 608
  SQL Server Express **608**
Microsoft account 826
Microsoft Advertising pubCenter **828**
Microsoft Developer Network (MSDN) **22**, 910
Microsoft Global Foundation Services (GFS) 921
Microsoft Intermediate Language (MSIL) 7, 56
Microsoft.WindowsAzure.Storage namespace 914
Microsoft.WindowsAzure.Storage .Table namespace 914
Microsoft's Faculty Connection 910
middle tier **653**
Min method of Math 157
MinDate property of class DateTimePicker 440, **442**
MinDate property of class MonthCalendar 439
MinHeight property of Windows 8 UI controls **759**
MinHeight property of WPF controls **933**
minimized and maximized child window 475
Minimized constant of WindowState enum **964**
Minimum property of class NumericUpDown **419**
MinimumSize property of class Control 400
MinimumSize property of class Form **400**
minInclusive XML Schema element **717**
minOccurs XML Schema attribute **715**
minus sign (-) for container elements in Internet Explorer 702
minus sign (-) indicating private visibility in the UML 1053
MinWidth property of Windows 8 UI controls **759**
MinWidth property of WPF controls **933**
mobile advertising network 828
mobile app 908
modal dialog **537**
model 616
model designer 619
model of a software system 1021, 1029, 1061
modifier key **424**
Modifiers property of class KeyEventArgs 425
modulus operator (%) **65**
monetary calculations 135
monetizing apps (Windows Phone 8) 828
MonthCalendar class **438**
  DateChanged event **438**
  FirstDayOfWeek property 439
  MaxDate property 439
  MaxSelectionCount property 439
  MinDate property 439
  MonthlyBoldedDates property 439
  SelectionEnd property 439
  SelectionRange property 439
  SelectionStart property 439
MonthlyBoldedDates property of class MonthCalendar 439
More Windows... option in Visual Studio .NET 475
mouse 388
  pointer 28
mouse click **421**
mouse event **421**, 422
mouse move **421**
MouseButtonEventArgs class (WPF) **943**
MouseDown event 943
MouseDown event handler 980
MouseDown event of class Control 422
MouseEnter event of class Control 422
MouseEventArgs class **422**
  Button property 422
  Clicks property 422
  GetPosition method **943**
  X property 422
  Y property 422
MouseEventArgs class (WPF) **943**
MouseEventArgs properties 422
MouseEventHandler delegate **422**
MouseHover event of class Control 422
MouseLeave event of class Control 422
MouseLeftButtonDown event (WPF) **943**, 946
MouseLeftButtonUp event (WPF) **943**, 946
MouseMove event (WPF) **943**
MouseMove event of class Control 422
MouseRightButtonDown event (WPF) **943**
MouseRightButtonUp event (WPF) **943**
MouseUp event 943
MouseUp event of class Control 422
MouseWheel event (WPF) 943
MouseWheel event of class Control 422
MouseWheelEventArgs class (WPF) 943
Move method of class Directory 521
Move method of class File 521
MoveFirst method of class BindingSource **631**
MoveNext method of IEnumerator **583**
MSDN (Microsoft Developer Network) 49, 156, 910
  Windows Azure credits 910
MSDN (Microsoft Developers Network) **22**
MSIL (Microsoft Intermediate Language) **7**
mulit-byte character set (MBCS) **1113**
multicast delegate **395**
multicast event 395
MulticastDelegate class **395**
MultiColumn property of class ListBox 446
multidimensional array 223
MultiExtended value of enumeration SelectionMode **446**
Multiline property of class TabControl 470
Multiline property of class TextBox 402
Multilingual App Toolkit for Visual Studio 2012 810
multiple document interface (MDI) **388**, **473**
multiple-selection statement 104
multiplication, * **65**
multiplicative operators: *, / and % 118
multiplicity **1021**
MultiSelect property of class ListView **462**
MultiSimple value of enumeration SelectionMode **446**
multithreading **833**
multitier app **652**
mutual exclusion **410**
mutually exclusive options 410
MySQL 608

N

N format specifier 98
n-tier app **652**
name attribute (XPath) **727**
name collision **391**, 482
name conflict **391**, 482
name node-set function **726**
Name property 947
Name property of class DirectoryInfo **467**
Name property of class FileInfo **467**
Name property of class XElement **731**
name XML Schema attribute **715**
named constant **199**
named parameter **184**
namespace **48**, 155, 165
namespace
  declaration 390, 481
  keyword 482
Namespace property of class XName **743**
Namespaces
  Microsoft.WindowsAzure. Storage 914
  Microsoft.WindowsAzure. Storage.Table 914
  of the FCL 165
  prefix XML **705**, 717
  System 167
  System.Collections 166, 564, **578**
  System.Collections. Concurrent **578**

Namespaces (cont.)
  System.Collections.Generic 166, 250, 528, **578**, 598
  System.Collections.Specialized **578**
  System.Data.Entity 166, **615**
  System.Data.Linq 166
  System.Diagnostics **445**
  System.Drawing 409
  System.IO 166, 520
  System.Linq 166, 241, 615
  System.Net 850
  System.Net.Http 888
  System.Runtime.Serialization.Formatters.Binary **548**
  System.Runtime.Serialization.Json 891
  System.ServiceModel 883
  System.Text 166, 491
  System.Text.RegularExpressions 491
  System.Threading.Tasks **838**
  System.Web 166
  System.Web.UI **665**
  System.Web.UI.WebControls **665**
  System.Windows.Controls 165, 813
  System.Windows.Forms 165, 389
  System.Windows.Input 165
  System.Windows.Media 165
  System.Windows.Shapes 165
  System.Xml 166, **729**
  System.Xml.Linq 166, **729**
  System.Xml.XPath **738**
  System.Xml.Xsl **745**
  Windows.UI.XAML.Controls **752**, 752
  Windows.UI.Xaml.Shapes 782
  XAML 930
  XML 705
naming collision **705**
naming convention
  Controls 814
NaN constant of structure Double **362**, 383
natural logarithm 157
navigability arrow in the UML **1053**
Navigate method of class WebBrowser **917**, 920
NavigateUrl property of a HyperLink control **670**
navigation property **614**, 615, 622, 623
Near Field Communication (NFC) 810
NegativeInfinity constant of structure Double **362**
NegativeNumberException represents exceptions caused by illegal operations performed on negative numbers 382
nested array initializer 224
nested control statements 118, 170
nested element **701**
nested element (XAML) 751, 929
nested for statement 201, 226, 227, 228
nested foreach statement 228
nested if…else selection statement 106, 107, 108
nested message in the UML **1044**
nested parentheses 66
nested structure of a layout 818
nesting 106
.NET
  Framework **6**, 558
  Framework Class Library **48**, 155, 165, 352
  Framework Class Library (FCL) **5**, 6, 8
  Framework documentation **49**
  initiative **6**
.NET for Windows Store Apps 8
New keyword 635
new keyword 195, 196, 225
new operator 77, 91
New Project dialog **23**, 25, 35, 931
new project in Visual Studio 26
new() (constructor constraint) **566**
newline character **59**
newline escape sequence, \n 60, 63
NewValue property of class ItemCheckEventArgs 451
Next method of class Random **167**, 168, 170
Next property of class LinkedListNode **599**
NextBytes method of class Random 797, **989**
NextNode property of class TreeNode 458
NextNode property of class XNode **740**
Nginx web server 651
node **457**
  child **457**
  expand and collapse 457
  parent **457**
  root **457**
  sibling **457**
node (in DOM tree) 728
node-set function **726**
node set of an xsl
  for-each element **722**
Nodes method of class XContainer **739**
Nodes property of a TreeView **457**
Nodes property of class TreeNode 458
Nodes property of class TreeView 458
NodeType property of class XObject **740**
non-static class member 278
None value for Stretch property **1001**
None value of enumeration SelectionMode **446**
nonvalidating XML parser **699**
not selected state **410**
note (in the UML) 103
Notepad 443
noun phrase in requirements document 1018, 1019, 1025
Now property of structure DateTime **488**
NuGet package manager 874
null keyword 86, **90**, 125, 195
NullReferenceException class **368**
numeric input (Windows Phone 8) 813, 818
numeric keypad (Windows Phone 8) 811, 818
NumericUpDown control 388, **419**
  DecimalPlaces property **419**
  Increment property **419**
  Maximum property **419**
NumericUpDown control (cont.)
  Minimum property **419**
  ReadOnly property **421**
  UpDownAlign property 420
  Value property 420
  ValueChanged event 420
NumericUpDown properties and events 419

O

object 2, **2**
  send a message 90
object (or instance) 4, 1043
Object Browser (Visual Studio .NET) 281
object class **286**, **290**, 314
  Equals method 315
  Finalize method 315
  GetHashCode method 315
  GetType method 315, **340**
  MemberwiseClone method 315
  ReferenceEquals method 315
  ToString method 293, 315
object creation expression 77, 92
Object data source **622**
object diagram in the UML **1107**
object initializer **283**
object initializer list **283**
object methods that are inherited directly or indirectly by all classes 315
object of a class **74**
object of a derived class 321
object of a derived class is instantiated 313
object-oriented analysis and design (OOAD) **5**
object-oriented design (OOD) 1010, 1018, 1025, 1029, 1033, 1053
object-oriented language **5**
object-oriented programming (OOP) **2**, **5**, 286
object serialization **548**
ObjectAnimationUsingKeyFrames animation (Windows 8 UI) **801**
ObjectCollection collection
  Add method 448
  Clear method 450
  RemoveAt method 450
ObjectDataProvider class (WPF) **971**
  IsAsynchronous property **971**
Oblique FontStyle 975
ObservableCollection<T> class **627**, 631
occurrence indicator **709**
octal number system (base 8) 1098
ODF (Open Document Format) 697
off-by-one error **129**
Offset property of GradientStop control **792**, **986**
omit-xml-declaration attribute **722**
one statement per line 70
one-to-many relationship **613**, 614, **1024**
one-to-one relationship **1024**
One value of enumeration SelectionMode **446**
one's complement 1105
ones position 1098
OnPaint method of class Control **485**

OOAD (object-oriented analysis and design) 5
OOD (object-oriented design) 1010, 1016, 1018, 1025, 1029, 1033
OOP (object-oriented programming) **5**, 286
OOXML (Office Open XML) 697
opacity mask **989**
OpacityMask property of Rectangle control **996**
Open method of class File 521
OpenFileDialog class **542**, **547**
opening a project 26
OpenRead method of class File 521
OpenText method of class File 521
OpenWrite method of class File 521
operand 117
operands of a binary operator **65**
operating system **8**
operation compartment in a UML class diagram 1034
operation in the UML 78, 1020, 1033, 1034, 1038, 1055, 1056
operation parameter in the UML 81, 1034, 1038
OperationContract attribute **883**, 885
operations of an abstract data type **280**
operator keyword 354
operator overloading **353**
operator precedence 66
  operator precedence chart 118
  rules **66**
Operators 65, **626**
  ^, boolean logical exclusive OR 147, **149**
  --, prefix decrement/postfix decrement 122
  -, subtraction 65, 66
  !, logical negation 147, **150**
  !=, not equals 67
  ?:, ternary conditional operator 106, 124
  *, multiplication 65, 66
  *=, multiplication compound assignment 122
  /, division 65, 66
  \=, division compound assignment 122
  &, boolean logical AND 147, **149**
  &&, conditional AND **147**, 148
  %, remainder 65, 66
  %=, remainder compound assignment 122
  +, addition 65, 66
  ++, prefix increment/postfix increment 122
  +=, addition assignment operator **121**
  +=, addition compound assignment 122
  <, less than 67
  <=, less than or equal 67
  =, assignment operator **64**
  -=, subtraction compound assignment 122
  ==, is equal to 67
  >, greater than 67
  >=, greater than or equal to 67
Operators (cont.)
  |, boolean logical inclusive OR 147, **149**
  ||, conditional OR 147, **148**
  arithmetic **65**
  as 340, 739
  await **834**
  binary **64**, 65
  boolean logical AND, & 147, **149**
  boolean logical exclusive OR, ^ 147, **149**
  boolean logical inclusive OR, | **149**
  cast **117**, 176
  compound assignment operators 121, 123
  conditional AND, && **147**, 149, 248
  conditional operator, ?: **106**, 124
  conditional OR, || 147, **148**, 149
  decrement operator, -- 122
  increment and decrement 122
  increment operator, ++ 122
  is **339**
  logical negation, ! 150
  logical operators **147**, 150
  logical XOR, ^ **149**
  member access (.) 135, 276
  multiplicative: *, / and % **118**
  new 77, 91
  postfix decrement **122**
  postfix increment **122**
  precedence chart 1092
  prefix decrement **122**
  prefix increment **122**
  remainder, % **65**
optical disk 517
optimizing compiler 136
optional parameter **182**, 183
  default value **182**, 183
Oracle Corporation 608, 1110
order attribute **726**
orderby clause of a LINQ query **244**
  ascending modifier **244**
  descending modifier **244**
OrderBy extension method of class Queryable **626**, 631
Orientation of WrapPanel control 934
Orientation property of StackPanel control **762**, **936**
orthographic projection **996**
out keyword **188**
out-of-range array index 368
Out property of class Console 519
outer set of brackets 204
OutOfMemoryException class **368**
output 59
output parameter **188**
overloaded constructors **264**
overloaded generic methods 566
overloaded methods 56, 179, 180, 559
  with identical signatures 181
overloaded operators for complex numbers 355
override a base class method **289**, 293
override keyword 206, **294**, 302, 303

P

package **1107**
package diagram in the UML **1107**
Package.appmanifest file (Windows 8 UI) **757**
Padding property of class Control 400
Padding property of class Form **400**
Page class **665**, 676, 681
  Session property 681
Page control (Windows 8 UI) **751**
  BottomAppBar **762**
  TopAppBar **762**
page layout software 491
Page_Init event handler **663**, 665
Page_Load event handler **675**
Page.Resources element of a Page **776**
PaintEventArgs class **485**
  ClipRectangle property **485**
  Graphics property **485**
PaintEventArgs properties 486
palette **37**
Panel class 12
Panel class (Windows 8 UI) 760
  Children property **769**
Panel control 388, **404**
  AutoScroll property 404
  BorderStyle property 404
  Children property **942**
  Controls property 404
  WPF **931**
Panel properties 404
Panel with scrollbars 405
parallel operations 833
parameter **78**, 80
  output **188**
parameter in the UML 81, 1034, 1038
Parameter Info window **55**
parameter list **80**, 92
parameterless constructor **267**, 270, **566**
params keyword **234**
parent element **701**
parent menu 429
parent node **457**
parent node of a DOM tree **728**
Parent property of class DirectoryInfo **468**
Parent property of class XObject **740**
parent window **473**
parent/child relationships between data 701
parentheses **51**, 66
  nested **66**
parsed character data **710**
parser **698**
partial class **665**
partial class **389**
partial keyword 392
partial modifier **665**
partial page update **872**
PartitionKey property of a TableEntity **914**
pass an array element to a method 212
pass an array to a method 212
pass-by-reference **187**
pass-by-value **187**, 212
Passing an array reference by value and by reference 215
Passing arrays and individual array elements to methods 212

passing options to a program with command-line arguments 236
password TextBox **401**
Path class **462**, **528**
  GetExtension method **528**
Path property of Binding class 775, **965**
pattern of 0s and 1s 518
payroll file 519
payroll system 518
#PCDATA keyword (DTD) **710**
Peek method of class Stack **591**
perform a calculation 71
perform a task 76
perform an action **51**
performance 833
performing operations concurrently **833**
permission setting 463
persistent data **517**
personalization **677**
perspective projection **803**, **996**
PhoneAccentBrush predefined color resource (Windows Phone 8) 819
PhoneApplicationPage control 812, 821
physical output operation **520**
PI (processing instruction) 720
PictureBox class **25**, **34**, 40
PictureBox control 415, 476
  Click event 415
  Image property 415
  SizeMode property 415
PictureBox properties and event 415
pin icon 29
platform **9**
platform independence 7
plus sign (+) for container elements in Internet Explorer 702
plus sign (+) indicating public visibility in the UML 1053
plus sign (+) occurrence indicator **709**
PNG (Portable Network Graphics) **41**
Point class 978
PointAnimation (WPF) 1001
PointCollection class **786**, **978**
  Add method **978**
  Clear method **980**
pointer events (Windows 8 UI) 769
PointerCanceled event (Windows 8 UI) **770**
PointerCaptureLost event (Windows 8 UI) **770**
PointerEntered event (Windows 8 UI) **769**
PointerExited event (Windows 8 UI) **770**
PointerMoved event (Windows 8 UI) **770**
PointerPressed event (Windows 8 UI) **770**
PointerReleased event (Windows 8 UI) **770**
PointerRoutedEventArgs class
  GetCurrentPoint method **770**
PointerRoutedEventArgs class (Windows 8 UI) 769, **770**
PointerWheelChanged event (Windows 8 UI) **770**
Points property of a Polyline or Polygon **785**
Points property of Polyline control **980**
Poll analysis application 203
Polygon control **977**
Polygon control (Windows 8 UI) **784**
  Points property **785**
Polyline control **977**
  Points property **980**
Polyline control (Windows 8 UI) **784**
  Points property **785**
polymorphic screen manager 320
polymorphically process Invoices and Employees 350
polymorphism 145, 316, **318**, 1058
pop data off a stack **163**
Pop method of class Stack **590**, 591
portability 7, 1113
Portability Tips overview xxvii
Portable Network Graphics (PNG) 41
porting **7**
position number 193
positional notation 1098
positional value 1099
positional values in the decimal number system 1099
PositiveInfinity constant of structure Double **362**
post request (HTTP) **881**
postback event of an ASP.NET page **676**
postfix decrement operator **122**
postfix increment operator **122**
PostgreSQL 608
Pow method of Math 135, 136, 156, 157
power (exponent) 157
power of 3 larger than 100 109
precedence 70, 124
  arithmetic operators 66
  chart 67m 118
precedence chart appendix 1092
precision
  formatting a floating-point number 118
  of double values 1095
  of float values 1095
precision of a floating-point value **94**
predefined styles in Windows 8 UI **752**
predicate **244**
prefix decrement operator **122**
prefix increment operator **122**
prepackaged data structures 578
presentation logic **653**
presentation XAML namespace (XAML) **930**
PreviewCanExecute event of CommandBinding class **950**
PreviewExecuted event of CommandBinding class **948**
PreviewMouseLeftButtonDown event (WPF) **946**
PreviewMouseLeftButtonUp event (WPF) **946**
Previous property of class LinkedListNode **599**
PreviousNode property of class XNode **740**
PrevNode property of class TreeNode 458
primary key **609**, 614
  in LINQ to Entities 609
principal 133
principle of least privilege **278**
privacy 921
privacy protection 677
private
  access modifier 262, 1053
  keyword 1053
  static class member 276
private access modifier **84**, 289
private visibility in the UML 1053
probability 167
Process class **445**
  Start method **445**
processing instruction (PI) **720**
  target **720**
  value **720**
processor **698**
program execution stack **163**
program in the general 318
program in the specific 318
programming languages
  C# **5**
project **23**
PROJECT menu 26
projection **249**
prolog (XML) **701**
promotion **117**, 133, 677
promotion rules **164**
prompt **63**
properties 82
Properties window **31**, 32, 35, 39, 660, 931
property **74**
property declaration **84**
property for a form or control **31**
Property property of Trigger class **963**
proprietary class 314
protected
  access modifier 261
protected access modifier 289
protocolMapping element in web.config 886
pseudorandom number **167**, 171
public
  access modifier 75, 84, 158, 1053
  class 50
  interface **257**
  keyword 50, 84, 1053, 1056
  member of a derived class 289
  method 258, 261
  method encapsulated in an object 261
  service **257**
  static class members 276
  static method 276
public access modifier 289
public default constructor **91**
public visibility in the UML 1053
publishing a web service **883**
push data onto a stack **163**
Push method of class Stack 590

Q

query **240**, **608**, 610
query expression (LINQ) **240**
query string **881**

Queryable class **615**
  OrderBy extension method **626**, 631
  ThenBy extension method **626**
  Where extension method **631**
querying the table storage service 918
question mark (?) occurrence indicator **709**
queue **281**
Queue class **580**
Queue generic class 580

R

RadialGradientBrush 995
RadialGradientBrush control **984**
radians 157
radio button **402**, 410
radio button group **410**
radio buttons, using with TabPage 472
RadioButton control 12, 16, 17, **407**, 410, 980, 995
  Checked event **942**, 944
  Checked property 410
  CheckedChanged event 410
  Text property 410
RadioButton control (Windows 8 UI) **763**, 769
  Checked event **763**, **769**
  Content property **763**
  IsChecked property 763
RadioButton control (WPF) **934**, 942, 951
RadioButton properties and events 410
RadioButtonList ASP.NET web control **670**
Random class 167, 989
  Next method **167**, 168, 170
  NextBytes method **989**
random number generation 206
random numbers 171
  in a range 171
  scaling factor **167**, 171
  seed value **167**, **171**
  shifting value 171
range variable of a LINQ query **243**
raster-based graphics **973**, **929**
Rational Software Corporation 1016
Rational Unified Process™ 1016
Read method of class Console 520
readability 47
Reading an XML document and displaying it in a TextBox 729
Reading sequential-access files 538
ReadLine method of class Console **64**, 80, 141, 520
readonly
  keyword **279**
read-only property 1036
ReadOnly property of class NumericUpDown **421**
ReadOnly property of class TextBox 402
real number 114
realization in the UML **345**
reclaim memory 278
recognizing clients 677
record **518**, 609
Record for sequential-access file-processing applications 531
record key **519**
recovery 908
Rectangle control **976**
Rectangle control (Windows 8 UI) **784**
rectangular array **223**, 225
  with three rows and four columns 224, 225
recursion 462
recursion step 185
recursive call 185
recursive descent **727**
recursive evaluation 186
  of 5! 186
recursive factorial 185
recursive method 184
ref keyword **188**, 212
refer to an object **89**
reference **89**
reference type **89**, 193
reference type constraint class **566**
Reference, output and value parameters 189
ReferenceEquals method of object 315
Regex class 491
registration (ASP.NET) 694
regular expression **514**
RegularExpressionValidator ASP.NET validation control **674**
reinventing the wheel 48
relational database 608, 609
relational database management system (RDBMS) 653
relational database table 609
relational operators **67**
relative addressing (XPath) **723**
relative positioning 660
release resource 370
release unmanaged resources 353
reliability 921
remainder 66
remainder operator, % 65, 66
Remove method of class ArrayList 586, **588**
Remove method of class Dictionary **529**
Remove method of class LinkedList **603**
Remove method of class List<T> 250, **252**
Remove method of class StringBuilder 509
RemoveAt method of class ArrayList 586
RemoveAt method of class List<T> 251, **252**
RemoveAt method of class ObjectCollection **450**
RemoveAt method of UIElementCollection class 769, 942
RemoveRange method of class ArrayList 586
RemoveRange method of class List<T> 251
RenderTransform property of a Windows 8 UI element **797**
RenderTransform property of a WPF UI element **989**
repetition
  counter controlled 116, 119
  definite **110**
  sentinel controlled 116
repetition statement 102, **104**, 109
  do...while 104, 136, 137, 137
  for 104, 131
  foreach 104
  while 104, 109, 112, 116
repetition terminates 109
Replace method of class string 503, 504
Replace method of class StringBuilder 511
Representational State Transfer (REST) **880**, 882
request method **881**
#REQUIRED keyword (DTD) **709**
RequiredFieldValidator ASP.NET validation control **673**, **674**
requirements **4**, 1014
requirements document **1010**, 1014, 1016
requirements gathering **1014**
reserved word **49**, 104
Reset method of interface IEnumerator **583**
ReshowDelay property of class ToolTip 418
Resize method of class Array 240, 250
resolution independence 759, **929**, 932
resource **417**
resource (Windows 8 UI) 776
  dynamic resource **764**
  static resource **764**
resource (WPF) **952**
  dynamic resource **956**
  static resource **956**
resource binding (Windows 8 UI) **764**
resource binding (WPF) **956**, 961
resource leak **274**, **369**
ResourceManager class **417**
  GetObject method **417**
Resources class **417**
Resources property of WPF controls **955**
ResponseFormat property of the WebGet attribute **889**, 900
responses to a survey 203, 205
REST web service 846
RESTful web services **882**
restriction on built-in XML Schema data type **716**, 717
result of an uncaught exception **366**
Result property of class Task **839**
result tree (XSLT) **719**
resumption model of exception handling **367**
rethrow an exception **375**
return keyword **85**, 162
return message in the UML 1046
return statement **85**, 185
return type (in the UML) 1034, 1040
return type of a method **75**
reusability 567
reusable component 287
reusable software components 165
reuse 48
Reverse method of class Array **584**

RGBA values 984
RichTextBox control (WPF) **947**
right brace, } 51, 62, 112, 116
right justify output **134**
RightToLeft property of class MenuStrip 433
robust 64
robust application **359**
role name in the UML **1021**
Roll a six-sided die 6,000,000 times 202
Roll a six-sided die 6000 times 169
rolling two dice 175
root element (XAML) 751, 929, 930
root element (XML) **697**, 701, 704
root node **457**
root node of a DOM tree **728**
root node, create 459
Root property of class XDocument **729**
RotateTransform (Windows 8 UI) **794**
RotateTransform control **987**
rotation angle 794
round a floating-point number for display purposes 118
rounded rectangle (for representing a state in a UML state diagram) **1029**
rounding a number 66, 135, 157
routed events (WPF) **943**
  bubbling events **946**
  direct events **946**
  tunneling events **946**
RoutedEventArgs class 942, 944
RoutedEventArgs class (Windows 8 UI) 769
RoutedEventArgs class (WPF)
  Handled property **944**
  Source property **944**
Row attached property of Grid control **774**, **937**
row objects **614**
row of a database table **609**
RowDefinition **817**
  Height property 817
RowDefinition class associated with Grid control 774, **936**
RowDefinitions property of Grid control 774, **936**
RowDefinitions property of Grid control **817**
RowKey property of a TableEntity **915**
rows of a two-dimensional array **223**
RowSpan attached property of Grid control **775**, **937**
rules of operator precedence 66
Run command in Windows 445
Run method of class Task **838**, 844
run mode **43**
run-time logic error 64
running an app 445
runtime system 567

S

SalariedEmployee class that extends Employee 330, 349
Sample new-format input for the XMLCombine application 742
Sample old-format input for the XMLCombine application 742
SaveChanges method of a LINQ to Entities DbContext **615**, 627
SaveFileDialog class 536
saving changes back to a database in LINQ to Entities 627
savings account 133
sbyte simple type 1094
ScaleTransform 996
ScaleTransform (Windows 8 UI) **794**
ScaleTransform control **987**
  CenterX property **996**
  CenterY property **996**
  ScaleX property **996**
  ScaleY property **996**
ScaleX property of ScaleTransform control **996**
ScaleY property of ScaleTransform control **996**
scaling factor (random numbers) 167, 171
schema 711
schema (database) **610**
Schema (XML) **698**
schema invalid document **712**
schema repository **708**
schema valid XML document **712**
schema-valid XML document describing a list of books 712
schema XML Schema element **714**
scope **130**, 130
  static variable 276
Scope class demonstrates instance and local variable scopes 177
scope of a declaration **177**
scope of a type parameter 570
Screen class (ATM case study) 1020, 1022, 1034, 1041, 1042, 1043, 1044, 1046, 1047, 1055
screen cursor **52**, 59, 60
screen-manager program 320
script (Unicode) **1115**
ScriptManager control **875**
scrollbar **32**
ScrollBars property of class TextBox 402
scrollbox **32**
SDI (Single Document Interface) **473**
sealed
  class 342
  keyword **342**
  method 342
search operators (Twitter) 910
Searching for characters and substrings in strings 499
secondary storage device **517**
security 921
seed value (random numbers) 167, 171
Seek method of class FileStream 548
SeekOrigin enumeration 548
select attribute (XPath) **726**
select attribute of xsl:for-each element **722**
select clause of a LINQ query **244**
Select method of class Control **398**
Select Resource dialog **41**
selected state **410**
SelectedImageIndex property of class TreeNode 458
SelectedIndex property of class ComboBox **454**
SelectedIndex property of class ListBox **447**
SelectedIndex property of class TabControl 470
SelectedIndexChanged event handler ComboBox class 631
SelectedIndexChanged event of class ComboBox **454**
SelectedIndexChanged event of class ListBox **446**
SelectedIndexChanged event of class TabControl 470
SelectedIndices property of class ListBox **447**
SelectedItem property of class ComboBox **454**
SelectedItem property of class ListBox **447**
SelectedItem property of ListView control **965**
SelectedItems property of class ListBox **447**
SelectedItems property of class ListView 463
SelectedNode property of class TreeView 458
SelectedTab property of class TabControl 470
selecting an item from a menu 390
selecting data from a table 610
selection 104
selection statement 102, **104**
  if 104, 105, 137
  if...else 104, 105, 106, 116, 137
  switch 104, 137, 144
SelectionEnd property of class MonthCalendar 439
SelectionMode enumeration **446**
  MultiExtended value 446
  MultiSimple value 446
  None value 446
  One value 446
SelectionMode property of class CheckedListBox 451
SelectionMode property of class ListBox **446**, 447
SelectionRange property of class MonthCalendar 439
SelectionStart property of class MonthCalendar 439
semicolon (;) **52**, 62, 70
send a message to an object 90
sentinel-controlled repetition 116
sentinel value **114**, 116
separator bar **432**
Separator control (WPF) **951**
sequence diagram in the UML **1017**, **1043**
sequence of messages in the UML **1044**
sequence statement **102**
sequence structure 103
sequence-structure activity diagram 103
sequential-access file 532
sequential execution **102**
sequential file **519**
  created using serialization 550
  read using deserialzation 554

[Serializable] attribute **549**
Serializable attribute 891
serialization **891**
SerializationException class 550
Serialize method of class
  BinaryFormatter **548**, 550
serialized object **548**, **549**
server response **881**
server-side form handler **881**
service of a class 261
Service1.svc 883
ServiceContract attribute **883**, 885
session 678
session item **683**
Session property of Page class **681**
session tracking **678**, 679
SessionID property of
  HttpSessionState class **683**
set accessor of a property **74**, 84, 87
Set as Startup Project 621
set keyword **85**
SetLeft method of Canvas control
  769, **942**
Setter class (Windows 8 UI) 777
Setter class (WPF) **955**, 963
  TargetName property **963**
SetTop method of Canvas control **769**,
  942
shallow copy **315**
Shape class hierarchy 288
shift **168**
Shift key 424
Shift property of class KeyEventArgs
  425, 427
Shifted and scaled random integers 168
shifting value (random numbers) **168**,
  171
short-circuit evaluation **149**
short simple type 1094
Short value of enumeration
  DateTimePickerFormat **439**
shortcut key 429
ShortcutKeyDisplayString property
  of class ToolStripMenuItem **430**,
  433
ShortcutKeys property of class
  ToolStripMenuItem **430**, 433
shortcuts with the & symbol 432
Show All Files icon **30**
Show method of class Control 398
Show method of class Form 390, 473, 479
ShowCheckBox property of class
  DateTimePicker 440
ShowDialog method of class
  OpenFileDialog 542, **548**
ShowDialog method of class
  SaveFileDialog 537
ShowShortcutKeys property of class
  ToolStripMenuItem **430**, 433
ShowUpDown property of class
  DateTimePicker 440
shuffling 206
  Fisher-Yates 209
Shutdown method of
  Application.Current **950**
sibling node **457**
sibling node of a DOM tree **728**
side effect **149**, 187
signal value **114**
signature of a method **181**
simple condition **147**
simple content in XML Schema **716**
simple name **390**, 482
Simple Object Access Protocol (SOAP)
  880
simple type **63**, 125, 164, 716
Simple Types
  bool 1094
  byte 1094
  char 63, 1094
  decimal **63**, **93**, 1095
  double **63**, **93**, 114, 1095
  float **63**, **93**, 1094
  int 62, 63, 114, 122, 1094
  long 1094
  names are keywords 63
  sbyte 1094
  short 1094
  table of 1094
  uint 1094
  ulong 1094
  ushort 1094
Simple value of enumeration
  ComboBoxStyle 454
simpleContent XML Schema element
  717
simpleType XML Schema element **717**
Simulation Dashboard 810
simulator
  terminate a running app 14, 19
Sin method of Math 157
sine 157
Single Document Interface (SDI) **473**
single-entry/single-exit control statements
  104
single inheritance 286
single-precision floating-point number **94**
single-quote character (') 704
single-selection statement 104, 105
single-line comment **47**
single-selection statement
  if 105
Size property of class Control 400
Size structure **400**
  Height property **400**
  Width property **400**
SizeMode property of class PictureBox
  41, 415
sizing handle **35**
sizing handle, enabled 35
SkewTransform 996
SkewTransform (Windows 8 UI) **794**
SkewTransform control **987**
Slider control (Windows Phone 8) 811,
  813
  Maximum property **822**
  Value property **822**
Slider control (WPF) **955**
.sln file extension 41
SmallImageList property of class
  ListView **463**
smart tag menu **629**, 669
snap lines **401**
SOAP (Simple Object Access Protocol)
  880
software engineering
  encapsulation 88
Software Engineering Observations
  overview xxvii
software life cycle **1014**
software reuse 286, 482
solid circle 103
solid circle (for representing an initial
  state in a UML diagram) in the UML
  1029, 1031
solid circle enclosed in an open circle (for
  representing the end of a UML activity
  diagram) 1031
solid circle surrounded by a hollow circle
  103
solid diamond (representing
  composition) in the UML **1022**
SolidBrush class **424**
SolidColorBrush 980
SolidColorBrush (Windows 8 UI) **787**
solution 12, 15, **23**
Solution Explorer 931
Solution Explorer window **30**
Some methods of class ArrayList 585
Sort method of class Array **583**
Sort method of class ArrayList 586
Sort method of class List<T> 251
Sorted property of class ComboBox 454
Sorted property of class ListBox 447
SortedDictionary generic class 580,
  597, 598
SortedList generic class **580**
SortedSet<T> class **605**
sorting in XSL 726
source code 314
Source property of Binding class **775**,
  965
Source property of Exception 378
Source property of MediaElement
  control **805**, **983**
Source property of RoutedEventArgs
  class **944**
source tree (XSLT) **719**
space character 49
space/time trade-off **592**
spacing convention 51
special character 63, **492**
special symbol **517**
specialization in the UML **1059**
Speech recognizers **1001**
Speech Synthesis Markup Language 1007
Speech synthesizers **1001**
SpeechRecognizer **1006**
SpeechSynthesizer **1006**
speech-to-text systems **1001**
splash screen **757**
Split method of class Regex 595
Split method of class String **542**
SQL 240, **608**
SQL Server **608**
SQL Server Express 617
SQL Server Express 2012 LocalDB **608**
Sqrt method of class Math 383
Sqrt method of Math 156, 157, 164
square brackets, [] **193**
square root 157
SSML 1007
stack **163**, **279**, 567
Stack class **580**, 589
stack frame **163**

Stack generic class 567, 580
  Stack< double> 576
  Stack<int> 576
stack overflow **163**
stack trace **362**
stack unwinding 378
Stack unwinding and Exception class properties 378
StackOverflowException class **368**
StackPanel class (Windows 8 UI) 760, **762**
  Orientation property **762**
StackPanel control (WPF) **936**
  Orientation property **936**
StackTrace property of Exception **377**, 378, 381
standard error stream object **520**
standard input stream object **519**
standard input/output object **52**
standard output stream object **520**
standard reusable component **287**
standard time format 259
standard XAML namespace (XAML) **930**
StandardStyles.xaml 757
Start method of class Process **445**
Start Page **21**
start page for a web application 663, 666, 671, 680, 687
start tag **696**, 704
start tag (XAML) 751, 929
StartPoint property of LinearGradientBrush control **791**, **984**
StartsWith and EndsWith methods 498
StartsWith method of class string **254**, 497
startup project **30**
StartupUri property of Application.xaml **932**
state **1017**
state button **407**
state in the UML **1017**, 1031
state machine diagram for some of the states of the ATM object 1030
state machine diagram in the UML **1017**, **1029**
state of an object 1025, 1029
stateless protocol 677
statement
  empty **70**
statements **52**, **76**
  break 141, 145
  continue **145**
  control statements 104
  control-statement nesting **104**
  control-statement stacking **104**
  do...while 104, 136, 137
  double selection **104**
  for 104, 128, 131, 133, 134
  if **67**, 104, 105, 137
  if...else 104, 105, 106, 116, 137
  iteration **104**
  multiple selection **104**
  nested **118**
  nested if...else **106**, 107, 108
  repetition **102**, **104**, **109**
  return 162
  selection **102**, 104
statements (cont.)
  single selection **104**
  switch 104, 137, 144
  switch multiple-selection statement 171
  throw **258**, **374**
  try **205**, **367**
  using 376
  while 104, 109, 112, 116
static
  class member 275
  method 77, **135**
  variable **275**, 276
static binding **342**
Static member demonstration 277
static method cannot access non-static class members 278
static method Concat 502
static resource (Windows 8 UI) **764**
static resource (WPF) **956**
static variable scope 276
Static variable used to maintain a count of the number of Employee objects in memory 276
stereotype in the UML **87**
storage 908
storage account 914
storage services (Windows Azure) 914
Storyboard control **1000**
  TargetName property **1001**
  TargetProperty property **1001**
straight-line form **66**
stream
  standard error **520**
  standard input **519**
  standard output **520**
Stream class **520**
stream of bytes **519**
StreamReader class **520**
StreamWriter class **520**
Stretch property of an ImageBrush **788**
Stretch property of ImageBrush **983**
Stretch property of MediaElement control 1001
  Fill value **1001**
  None value **1001**
  Uniform value **1001**
  UniformToFill value **1001**
Stretch property of VisualBrush **984**
StretchImage value 41
string **51**, 166
  literal **51**
  of characters 51
  verbatim **492**
string
  verbatim **445**
string array 195
String class
  Split method **542**
string class 491
  Concat method 502
  constructors 493
  CopyTo method 494
  EndsWith method 497, 498
  Equals method 495
  Format method **259**
  immutable 494
  IndexOf method 498
string class (cont.)
  IndexOfAny method 498
  LastIndexOf method 498, 500
  LastIndexOfAny method 498, 501
  Length property 494, 495
  method ToLower **595**
  method ToUpper **603**
  Replace method 503, 504
  StartsWith method **254**, 497
  Substring method 501
  ToLower method 503, 504
  ToUpper method **254**, 503, 504
  Trim method 503, 504
  verbatim string literal 161
String Collection Editor in Visual Studio .NET 448
string concatenation **161**, 277
string constant **492**
string constructors 493
string format specifiers 98
string indexer 495
string indexer, Length property and CopyTo method 494
string literal **492**
string XML Schema data type 715
string.Empty **90**
StringBuilder class 491, 504
  Append method 507
  AppendFormat method 508
  Capacity property 505
  constructors 505
  EnsureCapacity method 505
  Length property 505
  Remove method 509
  Replace method 511
  ToString method 505
StringBuilder constructors 504
StringBuilder size manipulation 505
StringBuilder text replacement 511
Stroke property of a shape control **784**, **976**
StrokeThickness property of a shape control **784**, **976**
strongly typed languages **125**
Stroustrup, Bjarne 359
struct
  DateTime **488**
struct keyword **512**
structure of a system 1029
structured programming **102**
Structured Query Language (SQL) **608**
STT systems **1001**
Style (Windows 8 UI) 771, 777
  TargetType property 777
style (WPF) **951**
Style class (WPF) **955**
  TargetType property **955**
  Triggers property **963**
Style property of a Button 764
Style property of class Font **409**
style sheet **702**
stylesheet start tag **722**
submenu 429
Substring method of class string 501
substrings generated from strings 501
subtraction 65
sum function (XSL) **727**
summarizing responses to a survey 203

summing integers with the for statement 132
surrogates (Unicode) **1111**
suspended app 19
SVC file **883**
switch code snippet (IDE) 177
switch expression 141
switch keyword **141**
switch logic 145
switch multiple-selection statement 104, 137, 144, 171
  activity diagram with break statements 144
  case label 141, 142
  controlling expression **141**
  default label **141**, 171
Sybase 608
Sybase, Inc. 1110
synchronous call **1044**
synchronous programming 6
synchronous request **872**
syntax error **57**
syntax error underlining 57
syntax-color highlighting **54**
system **1016**
system behavior **1016**
SYSTEM keyword in XML **704**
System namespace 48, 167, 491
system requirements **1014**
system structure **1016**
System.Collections namespace 166, 564, **578**
System.Collections.Concurrent namespace **578**
System.Collections.Generic namespace 166, 250, 528, **578**, 598
System.Collections.Specialized namespace **578**
System.Data.Entity namespace 166, **615**
System.Diagnostics namespace **445**
System.Drawing namespace 409
System.IO namespace 166, 520
System.Linq namespace 166, 241, 615
System.Net namespace 850
System.Net.Http namespace 888
System.Runtime.Serialization.Formatters.Binary namespace **548**
System.Runtime.Serialization.Json 891
System.ServiceModel namespace 883
System.Speech.Recognition **1003**
System.Speech.Synthesis **1003**
System.Text namespace 166, 491
System.Text.RegularExpressions namespace 491
System.Threading.Tasks namespace **838**
System.Web namespace 166
System.Web.UI namespace **665**
System.Web.UI.WebControls namespace **665**
System.Windows.Controls namespace 165, 813
System.Windows.Forms namespace 165, 389
System.Windows.Input namespace 165
System.Windows.Media namespace 165
System.Windows.Shapes namespace 165
System.Xml namespace 166, **729**
System.Xml.Linq namespace 166, **729**
System.Xml.XPath namespace **738**
System.Xml.Xsl namespace **745**
SystemException class **368**, 383

T

tab 388
tab character, \t 49, 60
tab stops 60
Tabbed pages in Visual Studio .NET 469
tabbed window 25
TabContainer Ajax Control Toolkit control **875**
TabControl class **468**
  ImageList property 470
  ItemSize property 470
  Multiline property 470
  SelectedIndex property 470
  SelectedIndexChanged event 470
  SelectedTab property 470
  TabCount property 470
  TabPages property **469**, 470
TabControl with TabPages example 469
TabControl, adding a TabPage 470
TabCount property of class TabControl 470
TabIndex property of class Control **398**
table 223, 609
table element 223
table of simple types 1094
table of values **223**
Table Storage service (Windows Azure) **911**, 914, 915
TableEntity class (Windows Azure) 914
  PartitionKey property **914**
  RowKey property **915**
  TimeStamp property **915**
TableOperation class (Windows Azure)
  Insert method **919**
TableQuery class (Windows Azure) 918
TabPage class **468**
  Text property **469**
TabPage, add to TabControl 469
TabPage, using radio buttons 472
TabPages added to a TabControl 470
TabPages property of class TabControl **469**, 470
TabPanel class **876**
TabStop property of class Control 398
tabular format 197
tagging 846
Tan method of Math 157
tangent 157
Target property of a HyperLink control **670**
TargetName property of Setter class **963**
TargetName property of Storyboard control **1001**
targetNamespace XML Schema attribute **714**
TargetProperty property of Storyboard control **1001**
TargetSite property of Exception 378
TargetType property of a ControlTemplate **800**
TargetType property of ControlTemplate class **963**
TargetType property of Style class 777, **955**
Task class **838**
  Result property **839**
  Run method **838**, 844
  WhenAll method **844**
  WhenAny method **845**
Task Parallel Library 838
TEAM menu 27
template **23**
template binding (WPF) **963**
Template property of WPF controls **961**
temporary 117
temporary data storage 517
terminate an app running in the simulator 14, 19
termination housekeeping **275**
termination model of exception handling **367**
ternary operator **106**
test harness **223**
Testing class BasePlusCommissionEmployee 299, 306, 312
Testing class CommissionEmployee 294
Testing generic class Stack 570, 574
Tests interface IPayable with disparate classes 351
text editor 52, 491
text field 813
text file 728
text node-set function **726**
Text property **35**, 39
Text property of a TextBlock **753**
Text property of class Button 402
Text property of class CheckBox 407
Text property of class Control 398
Text property of class Form 390
Text property of class GroupBox 404
Text property of class LinkLabel 443
Text property of class RadioButton 410
Text property of class TabPage **469**
Text property of class TextBox 402
Text property of class ToolStripMenuItem 433
Text property of class TreeNode 458
TextAlign property of a Label **40**
TextAlignment property of a TextBlock **753**
TextBlock class (Windows 8 UI) **749**
  FontFamily **753**
  FontSize **753**
  HorizontalAlignment **752**
  Margin **753**
  Text **753**
  TextAlignment **753**
  TextWrapping **753**
  VerticalAlignment **753**

TextBlock control 16, **973**
  Background property **983**
  FontFamily property **974**
  FontSize property **975**
  FontStyle property **975**
  FontWeight property **975**
  Foreground property **983**
  TextDecorations property **975**
TextBlock control (Windows 8 UI)
  Foreground property **787**
TextBlock control (Windows Phone 8) **813**
TextBlock control (WPF) **970**
textbox **401**
TextBox ASP.NET web control **668**
TextBox control 388, 401, **944**, 986
  AcceptsReturn property 402
  Multiline property 402
  ReadOnly property 402
  ScrollBars property 402
  Text property 402
  TextChanged event 402
  UseSystemPasswordChar property **401**
TextBox control (Windows 8 UI) 793
TextBox control (Windows Phone 8) **813**
  InputScope property 818
TextChanged event of class TextBox 402
TextDecoration control **975**
  Baseline **975**
  Location property **975**
  Overline **975**
  Strikethrough **975**
  Underline **975**
TextDecorations property of TextBlock control **975**
Text-displaying application 48
TextReader class 520
text-to-speech systems **1001**
TextWrapping property of a TextBlock **753**
TextWriter class 520
theme (Windows Phone 8) 819
theme of a Windows Store app 754
ThenBy extension method of class Queryable **626**
this
  keyword 262, 278
  reference **262**
  to call another constructor of the same class 267
this used implicitly and explicitly to refer to members of an object 263
thread 1007
  of execution **833**
ThreeState property of class CheckBox **407**
throw an exception **205**, **258**, 267, **360**, 365
throw point **362**, 367
throw statement **374**
Tick event of class Timer **487**
Tick event of DispatcherTimer class **960**
tier in a multitier application **652**
tile 9
tiled window **475**
TileHorizontal value of enumeration MdiLayout 475
TileVertical value of enumeration MdiLayout 475
time and date 488
Time value of enumeration DateTimePickerFormat **439**
Time1 class declaration maintains the time in 24-hour format 257
Time1 object used in an app 260
Time2 class declaration with overloaded constructors 265
TimeOfDay property of DateTime **439**
Timeout property of HttpSessionState class **683**
Timer class **487**, 960
  Interval property **487**
  Tick event **487**
TimeSpan struct **960**
TimeStamp property of a TableEntity **915**
timing diagram in the UML **1108**
Tip Calculator app 809
title bar 35
title bar, MDI parent and child 475
Title property of a Web Form **660**
Titles table of Books database 610, 611
To property of DoubleAnimation control **1001**
ToDecimal method of class Convert **98**
ToInt32 method of class Convert **64**, 67, 237
tokenize a string 542
ToLongDateString method of structure DateTime **442**
ToLongTimeString method of structure DateTime **488**
ToLower method of class string 503, 504, **595**
ToLower method of struct Char 514
tool bar 387
tool tip **28**
toolbar **27**
ToolBar control (WPF) **950**
toolbar icon **27**
ToolBarTray control (WPF) **951**
Toolbox 31, 660, 931
ToolkitScriptManager control **875**
TOOLS menu 27
ToolStripMenuItem class **430**
  Checked property 433, **437**
  CheckOnClick property 433
  Click event **432**, 433
  ShortcutKeyDisplayString property **430**, 433
  ShortcutKeys property **430**, 433
  ShowShortcutKeys property **430**, 433
  Text property 433
ToolStripMenuItem properties and an event 433
ToolTip class **417**
  AutoPopDelay property 418
  Draw event 418
  InitialDelay property 418
  ReshowDelay property 418
ToolTip properties and events 418
Top attached property of Canvas control **769**, **937**, 942
top tier **653**
TopAppBar property of a Page **762**
ToString method
  anonymous type 635
  class Exception 381
  class object 293, 315
  class StringBuilder 505, 507
ToUpper method of class string **254**, 503, 504, **603**
ToUpper method of struct Char 514
trace **573**
tracking customers 677
Transaction class (ATM case study) 1059, 1060, 1061, 1063, 1064
transfer of control **102**
transform **794**, **987**
Transform method of class XslCompiledTransform **745**
Transforming an XML document and splicing its contents with another 741
transition **103**
transition arrow **103**, 105, 110
transition between states in the UML **1030**
TranslateTransform (Windows 8 UI) **794**
TranslateTransform control **987**
transparency 788
traverse an array **226**
tree **457**
tree structure 702
Tree structure for the document article.xml 728
TreeNode class **457**, 458
  Checked property 458
  Collapse method 458
  Expand method 459
  ExpandAll method 459
  FirstNode property 458
  FullPath property 458
  GetNodeCount method 459
  ImageIndex property 458
  LastNode property 458
  NextNode property 458
  Nodes property 458
  PrevNode property 458
  SelectedImageIndex property 458
  Text property 458
TreeNode Editor 459
TreeNode properties and methods 458
TreeNodeCollection class **457**
TreeView class 429, **457**, 731
  AfterSelected event 458
  CheckBoxes property 458
  ImageList property 458
  Nodes property 458
  SelectedNode property 458
TreeView displaying a sample tree 457
TreeView properties and an event 458
TreeView used to display directories 460
TreeViewEventArgs class **458**
trigger (WPF) **963**
trigger an event 388
Trigger class (WPF) **963**
  Property property **963**
  Value property **963**
Triggers property of ControlTemplate class **963**

Triggers property of Style class **963**
trigonometric cosine 157
trigonometric sine 157
trigonometric tangent 157
Trim method of class string 503
TrimExcess method of class List<T> 251
TrimToSize method of class ArrayList **585**, 586
true **67**, 105, 106
truncate 66, **113**
truth table **148**
truth tables
 for operator ^ 149
 for operator ! 150
 for operator && 148
 for operator || 148
try block **205**, **365**
try statement **205**, **367**
TryParse method of structure Int32 **365**
TTS systems **1001**
tunneling events (WPF) **946**
Turing Machine 102
TV GUI showing the versatility of WPF customization (code-behind) 997
24-hour clock format 257
Twilio 925
Twitter search 910
 operators 912
two-dimensional array **223**
two's complement 1105
twos position 1100
type **62**, 63
type argument **562**, **563**, 570
type attribute in a processing instruction **720**
type checking **558**
Type class 315, **340**
 FullName property 315
type constraint **564**
 specifying **564**
type inference **563**
type parameter **562**, 567, 576
 scope 570
type parameter list **562**, 567
type XML Schema attribute **715**
typesetting system 491
typing in a TextBox 390

U

U+*yyyy* (Unicode notational convention) 1112
UIElementCollection class **769**, **942**
 Add method **769**, **942**
 Clear method **769**, **942**
 RemoveAt method 769, 942
uint simple type 1094
ulong simple type 1094
UML
 activity diagram 103, 105, 109, 136
 arrow 103
 class diagram **78**
 diamond 105
 dotted line **103**
 final state **103**
 guard condition **105**
 merge symbol **109**
UML (cont.)
 modeling properties 86
 note **103**
 solid circle **103**
 solid circle surrounded by a hollow circle 103
 stereotype **87**
UML (Unified Modeling Language) 5, 1010, 1016, 1020, 1027, 1029, 1059
 abstract operation **1060**
 class diagram 1061, 1062, 1063
 diagram 1016
 frame 1047
 generalization **1059**
 private visibility 1053
 public visibility 1053
 Resource Center (www.deitel.com/UML/) 1018
 specialization **1059**
UML Activity Diagram
 solid circle (for representing an initial state) in the UML 1031
 solid circle enclosed in an open circle (for representing the end of an activity) in the UML 1031
UML class diagram 287
 attribute compartment 1027
 operation compartment 1034
UML Sequence Diagram
 activation **1045**
 arrowhead 1045, 1046
 lifeline **1045**
UML State Diagram
 rounded rectangle (for representing a state) in the UML **1029**
 solid circle (for representing an initial state) in the UML **1029**
UML Use Case Diagram
 actor **1016**
 use case 1016
unambiguous (Unicode design basis) **1110**
unary cast operator 117
unary operator **117**, 150
UnauthorizedAccessException class 462
unbounded value 715
uncaught exception **366**
uneditable text or icons 388
unhandled exception 362, **366**
Unicode Consortium **1110**
Unicode Standard **1110**
Unicode Standard design basis 1110
 efficient **1110**
 unambiguous **1110**
 uniform **1110**
 universal 1110
Unicode® character set 125, 145, **492**, **518**
Unified Modeling Language (UML) 5, 1010, 1016, 1020, 1027, 1029, 1059
uniform (Unicode design basis) **1110**
Uniform Resource Identifier (URI) **651**, **706**
Uniform Resource Locator (URL) 651, **706**
Uniform Resource Name (URN) **706**
Uniform value for Stretch property **1001**
UniformToFill **788**, **983**
UniformToFill value for Stretch property **1001**
unique session ID of an ASP.NET client **683**
universal (Unicode design principle) 1110
universal-time format 257, 258, 259
unmanaged resource 353
unobtrusive validation **676**
unqualified name **165**, 177, 390, 482
unwind a method from the call stack 380
UpdatePanel ASP.NET Ajax Extensions control **876**
UpDownAlign property of class NumericUpDown 420
upper camel casing **50**
uppercase letter 50, 63
Uri 889
URI (Uniform Resource Identifier) **651**, **706**
UriTemplate property of WebGet attribute **884**
URL (Uniform Resource Locator) **651**, **706**
UrlEncode method of class HttpUtility **920**
URN (Uniform Resource Name) **706**
use case diagram in the UML **1015**, 1017
use case in the UML **1015**, 1016
use case modeling **1015**
UseMnemonic property of class LinkLabel 443
user-defined classes 49
user interface 653
UserControl control **486**
UserControl defined clock 487
user-defined exception class **381**
user-defined method Maximum 159
user-defined types 715
UseSystemPasswordChar property of class TextBox **401**
ushort simple type 1094
using directive **48**, 82, 165
UsingAnimations example 999
UsingBrushes example 980
UsingGradients example 984
UTF-8 1111
UTF-16 **1111**
UTF-32 **1111**
utility method **143**

V

valid identifier 62
valid XML document **698**, 709
Validate property of Page class **676**
validating XML parser **698**
validation **96**
validation control **670**
ValidationExpression property of a RegularExpressionValidator control **674**
validator **670**
ValidatorCalloutExtender control **876**
validity checking **96**
Value **740**
value contextual keyword **85**

Value property of a Slider (Windows Phone 8) **822**
Value property of class DateTimePicker **439**, 440, **442**
Value property of class LinkedListNode **599**
Value property of class NumericUpDown 420
Value property of class XElement **731**
Value property of class XText **740**
Value property of Trigger class **963**
value type **89**, 193
value type constraint struct **566**
ValueChanged event of class DateTimePicker **440**
ValueChanged event of class NumericUpDown 420
Values property of class Hashtable **596**
ValueType class **512**
var keyword **243**
variable **61**, 62
  declaration statement **62**
  name **62**
variable is not modifiable 279
variable-length argument list **234**
variable scope 130
vector-based graphics **781**, **973**, **929**
verb phrase in requirements document 1034
verbatim string **445**
verbatim string **492**
verbatim string literal 161
verbatim string syntax(@) 492
version attribute (XSL) 722
version attribute in xml declaration **700**
VerticalAlignment property of a TextBlock **753**
VerticalAlignment property of Windows 8 UI controls **759**
VerticalAlignment property of WPF controls **933**
VIEW menu 26, 28
View property of class ListView **462**, 463
virtual
  keyword **303**
virtual directory 652
virtual goods 10, 810, 827
virtual machine (VM) 7
visibility in the UML **1053**
visibility marker in the UML **1053**
Visibility property of a WPF UI element **978**
  Collapsed state **978**
  Hidden state **978**
  Visible state **978**
visibility symbols in the UML 78, 87
Visible property of an ASP.NET web control **673**
Visible property of class Control 398
VisitedLinkColor property of class LinkLabel 443
visual app development **21**
visual inheritance 857
visual programming 389
Visual property of VisualBrush control **983**
Visual Studio **11**
  component tray **418**
  IDE (integrated development environment) 11
Visual Studio .NET Class View 281
Visual Studio .NET Object Browser 281
Visual Studio Express 2012 for Web **650**
Visual Studio Express 2012 for Windows Phone 810
Visual Studio® 2012 **21**
visual tree (WPF) **960**
VisualBrush
  Stretch property **984**
VisualBrush control **983**
  Visual property **983**
VisualStateManager.VisualGroups element for defining state changes in a ControlTemplate **800**, 801
vocabulary (XML) **697**
VoiceXML 697
void keyword 51, 75

W

W3C (World Wide Web Consortium) **696**
W3C XML Schema **698**
Wallet 810
waterfall model **1015**
WCF
  ABCs of a WCF service 881
  DataContract attribute 890
  DataMember attribute 890
  OperationContract attribute **883**
  ResponseFormat property of the WebGet attribute **889**
  Serializable attribute 891
  ServiceContract attribute 883
  UriTemplate property of WebGet attribute **884**
  WebGet attribute 884
WCF REST service to create random equations based on a specified operation and difficulty level 896
WCF Service Application project type **883**, **884**
WCF service class 883
WCF service endpoint **881**
Web app development **650**
web control **650**
Web Form **650**, **657**, 678, 684
  Init event **663**
  Load event 675
web server **651**
web service **846**, **880**
  host **881**
  processing user-defined types 893
Web Site Administration Tool **862**
Web.config file **664**, 883
WebBrowser class
  Navigate method **917**, 920
WebClient class 850
  CancelAsync method **850**
  DownloadDataTaskAsync method **852**
  DownloadStringTaskAsync method 850, **850**
WebGet attribute 884
webHttp Web.config property **886**
webHttpBinding Web.config binding setting **886**
WebMessageFormat.Json setting of ResponseFormat property 889, 900
WebMessageFormat.Xml setting of ResponseFormat property 889
well-formed XML document **698**
WhenAll method of class Task **844**
WhenAny method of class Task **845**
where clause **565**
where clause of a LINQ query **244**
Where extension method of class Queryable **631**
while keyword 136
while repetition statement 104, 109, 112, 116
  activity diagram in the UML 109
whitespace **49**, 51, 70
  characters **49**
whitespace character (regular expressions) 504
whole/part relationship 1022
widget **388**
width of a column 813
Width property of a ColumnDefinition 817
Width property of a shape 976
Width property of a shape control 784
Width property of structure Size **400**
window auto hide **29**
Window control (WPF) **930**, 939, 958
  AllowsTransparency property **957**
  CommandBindings property **950**
  DragMove method **959**
  WindowState property **964**
  WindowStyle property **957**
window gadget **388**
WINDOW menu 27
window tab 25
Windows
  Font 39
  Properties **31**, 32, 35, 39
  Solution Explorer **30**
Windows 8 **9**
Windows 8 UI 8, 9, 15, **748**, 781
  App.xaml **756**
  App.xaml.cs **757**
  Assets folder **757**
  Background property of a control **752**
  Binding class **775**
  Border **775**
  Button control **764**, 769
  Canvas control 760, **765**
  code-behind class 757
  Common folder **757**
  content control **752**
  ContentControl control 763
  control **751**
  data binding **771**, **775**
  DataTemplate **771**
  DataTemplate class **777**
  Ellipse control **784**
  event handling 765
  EventSetter class 777
  GradientStop control **792**
  Grid 760
  Grid control **752**

Windows 8 UI (cont.)
- HorizontalAlignment property **759**
- Image control 771, 775
- ImageBrush control **788**
- KeyDown event 770
- KeyRoutedEventArgs class 770
- KeyUp event 770
- Label control **752**
- layout container **752**, 769
- layout control 759
- Line control **784**
- LinearGradientBrush control **788**
- ListView control **771**
- MainPage.xaml 757
- Margin property **759**
- MaxHeight property **759**
- MaxWidth property **759**
- MediaElement control **805**
- MinHeight property **759**
- MinWidth property **759**
- ObjectAnimationUsingKeyFrames animation **801**
- Package.appmanifest file 757
- Page control **751**
- Panel 760
- pointer events 769
- PointerCanceled event **770**
- PointerCaptureLost event **770**
- PointerEntered event **769**
- PointerExited event **770**
- PointerMoved event **770**
- PointerPressed event **770**
- PointerReleased event **770**
- PointerRoutedEventArgs class 769, **770**
- PointerWheelChanged event **770**
- Polygon control **784**
- Polyline control **784**
- predefined styles **752**
- RadioButton control **763**, 769
- Rectangle control **784**
- resource **776**
- resource binding **764**
- RotateTransform control **794**
- RoutedEventArgs class 769
- ScaleTransform control **794**
- Setter class **777**
- SkewTransform control **794**
- SolidColorBrush control **787**
- StackPanel control 760, **762**
- StandardStyles.xaml 757
- Style class **777**
- TextBlock **749**
- TextBox control 793
- theme 754
- TranslateTransform control **794**
- VerticalAlignment property **759**

Windows Azure **11**, **907**
- academic pass **910**
- Active Directory Federation Services 921
- blob 925
- BLOB service 914
- cloud services 910
- development storage account 914
- HDInsight 925
- identity 925

Windows Azure (cont.)
- Microsoft resources 921
- Microsoft Virtual Academy training course 924
- SDK tools 924
- Service Management API 925
- storage account 914
- Storage Emulator **909**
- storage services 914
- Table Storage service **911**, 914, 915
- TableEntity 914
- use cases 907
- Windows Azure AD Access Control 921
- Windows Azure SDK for Visual Studio 2012 **909**
- Windows Azure Security Guidance 921
- Windows Azure Shared Access Signatures 921
- Windows Azure Storage NuGet package 914
- Windows Azure Trust Center 921
- Windows Identity Foundation (WIF) 921

Windows bitmap (BMP) 41
Windows Communication Foundation (WCF) 880
Windows Explorer 445
Windows Form **388**
Windows Forms app **23**
Windows operating system **8**
Windows Phone 7 8
Windows Phone 8 809
- Coding4Fun Toolkit 831
- developer features 809
- Emulator **810**
- SDK 810
- theme 819
- **Windows Phone App** template **815**
- Windows Phone Application Analysis 810
- Windows Phone Dev Center **825**, 826
- Windows Phone Marketplace **827**
- Windows Phone Power Tools 831
- Windows Phone Toolkit 831

Windows Phone 8 operating system **9**
Windows Phone App template **815**
Windows Phone Dev Center 10
Windows Phone Emulator **10**
Windows Phone Marketplace **10**
Windows Phone operating system 9
Windows Presentation Foundation (WPF) **928**, 973
Windows Store 9
Windows Store developer 9
Windows.UI.XAML.Controls namespace **752**
Windows.UI.Xaml.Shapes namespace (Windows 8 UI) 782
WindowState property of Window control **964**
WindowState.Minimized constant **964**
WindowStyle property of Window control **957**

Withdrawal class (ATM case study) 1020, 1021, 1023, 1026, 1031, 1034, 1042, 1043, 1045, 1046, 1047, 1055, 1056, 1058, 1059, 1060, 1061, 1063
WMAppManifest.xml 825
word processor 491, 498
workflow **103**
workflow of an object in the UML 1030
World Wide Web Consortium (W3C) **696**
WPF (Windows Presentation Foundation) **928**, 943
- App.xaml **932**
- App.xaml.vb **932**
- Background property 951
- Binding class **965**
- Border control 934, 995
- Brush class 951
- Button control **934**, 942, 951
- Canvas control **937**, 977
- CheckBox control 951, 995
- code-behind class 930, 932
- collection view **971**
- CollectionView class **971**
- ColorAnimation 1001
- ComboBox control 951
- command binding **947**
- command library 947
- CommandBinding class **947**
- commands **946**
- content control **931**
- ContentControl control **931**, 934
- ContentPresenter class **960**
- control **930**, 960
- control template **960**, 961
- ControlTemplate 995
- ControlTemplate class **963**
- data binding **965**
- data provider **971**
- data template **968**
- DataContext property **968**
- DataTemplate class **968**
- dependency property **957**, 963
- DockPanel control 934
- DoubleAnimation control **1001**
- Ellipse control **976**
- event handling 938
- EventSetter class **955**
- Expander control 934
- GradientStop control **986**
- Grid control 931, **936**
- GridView control **968**
- GridViewColumn class **968**
- GroupBox control **936**
- HorizontalAlignment property **933**
- ICommand interface **947**
- ImageBrush control **983**
- IsMouseOver property **963**
- KeyDown event 943
- KeyUp event 943
- Label control **931**
- layout container **931**, 942
- layout control 933
- Line control **976**
- LinearGradientBrush control **984**
- ListView control **965**
- logical tree **960**

WPF (Windows Presentation Foundation) (cont.)
  lookless control **960**
  Margin property **933**
  MaxWidth property **933**
  MediaElement control **983**
  Menu control **950**
  MenuItem control **950**
  MinHeight property **933**
  MinWidth property **933**
  MouseButtonEventArgs class **943**
  MouseEventArgs class **943**
  MouseLeftButtonDown event **943**, 946
  MouseLeftButtonUp event **943**, 946
  MouseMove event **943**
  MouseRightButtonDown event **943**
  MouseRightButtonUp event **943**
  MouseWheel event 943
  MouseWheelEventArgs class 943
  ObjectDataProvider class **971**
  Panel control **931**
  PointAnimation 1001
  Polygon control **977**
  Polyline control **977**
  PreviewMouseLeftButtonDown event **946**
  PreviewMouseLeftButtonUp event **946**
  RadialGradientBrush 995
  RadialGradientBrush control **984**
  RadioButton control **934**, 942, 951, 980, 995
  Rectangle control **976**
  resource **952**
  resource binding **956**, 961
  Resources property **955**
  RichTextBox control **947**
  RotateTransform control **987**
  routed events **943**
  RoutedEventArgs class 942, 944
  ScaleTransform 996
  ScaleTransform control **987**
  Separator control **951**
  Setter class **955**, 963
  SkewTransform 996
  SkewTransform control **987**
  Slider control **955**
  SolidColorBrush 980
  StackPanel control **936**
  Storyboard control **1000**
  style **951**
  Style class **955**
  template binding **963**
  Template property **961**
  TextBlock control **970**, **973**
  TextBox control **944**, 986
  ToolBar control **950**
  ToolBarTray control **951**
  TranslateTransform control **987**
  trigger **963**
  Trigger class **963**
  VerticalAlignment property **933**
  visual tree **960**
  VisualBrush control **983**
  Window control **930**, 939, 958
  WrapPanel control 934
  XmlDataProvider class **971**
WPF bitmap effect **989**
WrapPanel control (WPF) 934
  Orientation property 934
Write method of class Console **58**, 520
WriteLine method of class Console 51, 58, 520
WriteLine method of class StreamWriter **537**
WSDL (Web Services Description Language) 697
www.deitel.com/LINQ/ 255
www.oasis-open.org 708
www.unicode.org 518
www.w3.org/XML/Schema 710, 712
www.w3schools.com/schema/default.asp 712
www.xml.org 708

X

X format specifier 98
X property of class MouseEventArgs 422
x:Class attribute (XAML) **751**, **930**
x:Key property of a ControlTemplate **800**
x:Name property of a Button **764**
x:Name property of a Canvas **765**
x:Name property of a RadioButton **763**
x:Name property of a Windows 8 UI control **763**, **764**, **765**
x:Name property of a WPF control 938
XAML (Extensible Application Markup Language) **748**, **928**
  attribute 929
  default namespace 752, 930
  element 751, 929
  end tag 751, 929
  markup extension **764**, **775**, **956**, 963, 965
  namespace 752, 930
  nested element 751, 929
  presentation XAML namespace **930**
  root element 751, 929, 930
  standard XAML namespace **930**
  start tag 751, 929
  x:Class attribute **751**, **930**
XAML view **754**, **931**
XAttribute class **738**
XBRL (Extensible Business Reporting Language) 697
XComment class **740**
XContainer class 737
  Add method **743**
  Descendants method 738
  Element method **738**
  Elements method **731**
  Nodes method **739**
XDocument class **729**, 851
  Load method **729**
  Root property **729**
XDocumentType class **739**
XElement class **729**
  Attribute method **738**
  Elements method **731**
  HasElements property **731**
  Name property **731**
  Value property **731**
XHTML (Extensible HyperText Markup Language) **719**
XML (Extensible Markup Language)
  attribute **704**
  attribute value **704**
  child element **701**
  container element **701**
  declaration **700**, 703
  document 775, 965
  element **696**
  empty element 704
  end tag **696**
  instance document **716**, 718
  node **702**
  parent element **701**
  parser 698
  processor **698**
  prolog **701**
  root **702**
  root element **697**
  start tag **696**
  vocabulary **697**
XML document containing book information 723
XML document that describes various sports 720
XML document using the laptop element defined in computer.xsd 718
.xml file extension **698**
XML file generated by XMLCombine 743
xml namespace prefix 705
XML namespaces demonstration 706
XML Path Language (XPath) **719**
XML Schema **698**, 708, **712**, 715
  complex types **715**
  simple types **715**
XML Schema document defining simple and complex types 716
XML Schema document for book.xml 713
XML Schema URI 714
XML used to mark up an article 699
XML Working Group of the W3C 696
XmlDataProvider class (WPF) **971**
  IsAsynchronous property **971**
XmlNodeType enumeration **740**
xmlns attribute in XML **706**, 707
xmlns namespace prefix 705
XName class **731**
  LocalName property **731**
  Namespace property **743**
XNamespace class 739, **740**, 900
XNode class **739**
  NextNode property **740**
  PreviousNode property 740
  XPathSelectElements extension method **738**
XObject class **740**
  Document property **740**
  NodeType property **740**
  Parent property **740**
Xor bitwise operator 438
XPath (XML Path Language) **719**, 738
XPathSelectElements extension method of class XNode **738**
XProcessingInstruction class **740**
.xsd filename extension **712**
XSL (Extensible Stylesheet Language) **699**, 708, **719**
  variable **727**

XSL document that transforms
  sorting.xml into XHTML 724
.xsl filename extension **721**
XSL-FO (XSL Formatting Objects) **719**
XSL Formatting Objects (XSL-FO) **719**
XSL style sheet **719**, 727
XSL template **722**
xsl template element **722**
XSL Transformations (XSLT) 719
xsl:for-each element **722**
xsl:output element **722**
xsl:text element **726**
xsl:value-of element **723**
XslCompiledTransform class **745**
  Load method **745**
  Transform method **745**
XSLT processor 719
XSLT that creates elements and attributes
  in an XHTML document 721
XText class **740**
  Value property **740**

Y

Y property of class MouseEventArgs 422

Z

zeroth element **193**
ZIndex attached property of Canvas
  control **938**