Mastering™ Visual C#™.NET

Jason Price
Mike Gunderloy

SYBEX®

Mastering
Visual C#.NET

This page intentionally left blank

Mastering™
Visual C#™.NET

Jason Price

Mike Gunderloy

San Francisco London

Associate Publisher: Richard Mills

Acquisitions Editor: Denise Santoro-Lincoln

Developmental Editor: Tom Cirtin

Editor: Kim Wimpsett

Production Editor: Erica Yee

Technical Editor: Gregory Beamer

Book Designer: Maureen Forys, Happenstance Type-O-Rama

Graphic Illustrator: Tony Jonick

Electronic Publishing Specialist: Jill Niles, Judy Fung, Scott Benoit

Proofreaders: Emily Hsuan, Nelson Kim, Laurie O'Connell, Yariv Rabinovitch, Nancy Riddiough

Indexer: Nancy Guenther

Cover Designer: Design Site

Cover Illustrator/Photographer: Sergie Loobkoff

Library of Congress Card Number: 2001094588

ISBN: 0-7821-2911-0

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

This book is dedicated to my family. You're still in my heart, even though you are far away.
—Jason Price

This one's for Steve, who is long overdue for a dedication.
—Mike Gunderloy

This page intentionally left blank

Acknowledgments

MANY THANKS TO ALL the great, hard-working people at Sybex, including Tom Cirtin, Denise Santoro Lincoln, Kim Wimpsett, and Erica Yee. Thanks also to Gregory Beamer for his thorough technical review.

—*Jason Price*

As much as some authors (including myself) are solitary creatures, in this electronic age it's impossible to be truly isolated. This book is better for that lack of isolation because it allowed me to bother Ken Getz, Steve White, and Mary Chipman with questions. Mary was especially helpful as I tried to figure out .NET security. The remaining errors, of course, are entirely mine.

Thanks to the editorial staff at *MCP Magazine*—Di Schaffhauser, Keith Ward, Kristen McCarthy, and Michael Domingo—for understanding and for some slack as I tried to juggle book deadlines with magazine deadlines.

Thanks to Denise Santoro Lincoln and the lead author of this book, Jason Price, for bringing me into this project. And of course I'd like to add my thanks to Jason's for the great team at Sybex.

My online family was always there when I wanted to rant about failing code, impossible deadlines, or the general annoyances associated with developing software. Thanks, folks.

And special thanks, as always, to Dana and Adam, who put up with my late nights and distraction as this one went through the publishing process. Without a loving family, I wouldn't have the sanity to write a single word.

—*Mike Gunderloy*

Contents at a Glance

Introduction . . . . . *xxii*

Part 1 • Fundamental C# Programming 1

Chapter 1 • Introduction to C# 3

Chapter 2 • Basic C# Programming 19

Chapter 3 • Expressions and Operators 47

Chapter 4 • Decisions, Loops, and Preprocessor Directives 71

Chapter 5 • Object-Oriented Programming 97

Chapter 6 • More about Classes and Objects 141

Chapter 7 • Derived Classes 165

Chapter 8 • Interfaces 209

Chapter 9 • Strings, Dates, Times, and Time Spans 239

Chapter 10 • Arrays and Indexers 301

Chapter 11 • Collections 339

Chapter 12 • Delegates and Events 397

Chapter 13 • Exceptions and Debugging 417

Part 2 • Advanced C# Programming 449

Chapter 14 • Threads 451

Chapter 15 • Streams and Input/Output 487

Chapter 16 • Assemblies 539

Chapter 17 • Attributes and Reflection 569

Chapter 18 • Remoting 603

Chapter 19 • Security 619

Chapter 20 • XML 645

Chapter 21 • Other Classes in the Base Class Library 677

Part 3 • .NET Programming with C# 719

Chapter 22 • Introduction to Databases 721

Chapter 23 • Active Data Objects: ADO.NET 757

Chapter 24 • Introduction to Windows Applications 819

Chapter 25 • Active Server Pages: ASP.NET 847

Chapter 26 • Web Services 881

Appendices 903

Appendix A • C# Keywords 903

Appendix B • C# Compiler Options 909

Appendix C • Regular Expressions 921

Index *935*

Contents

Introduction . . . . . *xxii*

Part 1 • Fundamental C# Programming 1

Chapter 1 • Introduction to C# 3

Developing Your First C# Program 3
Understanding the Main() Method 5
Compiling a Program 7
Introducing the Microsoft Intermediate Language (MSIL) 8
Introducing Visual Studio .NET 9
Starting Visual Studio .NET and Creating a Project 10
Compiling and Running the Program 14
Using the .NET Documentation 15
Accessing the Documentation Using the .NET SDK 15
Accessing the Documentation Using VS .NET 17
Summary 18

Chapter 2 • Basic C# Programming 19

Using Statements, Whitespace, and Blocks 19
Adding Comments 21
Using Data Types, Variables, and Constants 22
Introducing Data Types 22
Looking at C#'s Built-in Types 23
Understanding Variables 29
Defining Constants 33
Introducing Strings 35
Understanding Enumerations 36
Specifying Values in an Enumeration 39
Specifying an Enumeration Base Type 40
Handling Input and Output 41
Reading a Single Character 41
Reading a String of Characters 42
Formatting Output 43
Summary 46

Chapter 3 • Expressions and Operators 47

Understanding Expressions and Operators 47
Assignment Operator 49
Arithmetic Operators 49
Comparison Operators 51
Boolean Logical Operators 53

Ternary Operator . . . 56
Bitwise Operators . . . 57
Shortcut Operators . . . 62
is Operator . . . 66
Operator Precedence . . . 67
Summary . . . 70

Chapter 4 • Decisions, Loops, and Preprocessor Directives . . . 71

Using the if Statement . . . 71
Replacing a Single Statement with a Block . . . 72
Using Nested if Statements . . . 74
Using Logical Operators with the if Statement . . . 75
Implementing the switch Statement . . . 76
Comparing String Values Using a switch Statement . . . 78
Introducing Fall-Through . . . 79
Using Loop Statements . . . 81
The while Loop . . . 81
The do…while Loop . . . 83
The for Loop . . . 84
The foreach Loop . . . 86
Understanding Jump Statements . . . 88
The break Statement . . . 88
The continue Statement . . . 89
The goto Statement . . . 90
Creating Preprocessor Directives . . . 92
Defining Symbols . . . 92
Using the #if, #elif, and #else Directives . . . 94
Summary . . . 96

Chapter 5 • Object-Oriented Programming . . . 97

Introducing Classes and Objects . . . 97
Declaring a Class . . . 98
Creating Objects . . . 99
Null Values . . . 101
Default Field Values and Initializers . . . 104
Using Methods . . . 106
Defining Methods . . . 106
Calling Methods . . . 108
Hiding . . . 110
The this Object Reference . . . 112
More on Parameters . . . 114
Method Overloading . . . 120
Using Access Modifiers . . . 122
Creating and Destroying Objects . . . 126
Using Constructors . . . 127
Using Destructors . . . 135

Introducing Structs 137
Summary 140

Chapter 6 • More about Classes and Objects 141

Introducing Static Members 141
Using Static Members 142
Using Constant Fields 145
Using Readonly Fields 146
Defining Properties 148
Introducing the "Has a" Relationship 150
Using the "Has a" Relationship 151
Nesting Classes 153
Learning about Namespaces 155
Nesting Namespaces into Hierarchies 158
The using Statement 161
Summary 163

Chapter 7 • Derived Classes 165

Introducing Inheritance 165
Learning about Polymorphism 170
Specifying Member Accessibility 174
Hiding Members 177
Versioning 182
Using the System.Object Class 184
Overriding the System.Object Class Methods 188
Boxing and Unboxing 189
Using Abstract Classes and Methods 191
Declaring Sealed Classes and Methods 194
Sealed Classes 194
Sealed Methods 194
Casting Objects 196
Upcasting 198
Downcasting 198
Operator Overloading 201
Overloading the Equal Operator 203
Overloading the Addition Operator 204
Overloading Other Operators 207
Summary 207

Chapter 8 • Interfaces 209

Defining an Interface 209
Implementing an Interface Using a Class 210
Implementing Multiple Interfaces 213
Inheriting from a Class and Implementing Interfaces 217
Casting an Object to an Interface 220
The is Operator and Interfaces 220
The as Operator and Interfaces 221

Using Derived Interfaces 224
Deriving an Interface from One Interface 224
Deriving an Interface from Multiple Interfaces 227
Understanding Explicit Interface Members 230
Implementing Explicit Interface Members 230
Hiding Interface Members 234
Summary 237

Chapter 9 • Strings, Dates, Times, and Time Spans 239

Using Strings 239
Creating Strings 240
Using String Properties and Methods 240
Creating Dynamic Strings 260
Creating StringBuilder Objects 260
Using StringBuilder Properties and Methods 261
Representing Dates and Times 267
Creating DateTime Instances 268
Introducing Time Spans 269
Using DateTime Properties and Methods 272
Using Time Spans 288
Creating TimeSpan Instances 288
Using TimeSpan Properties and Methods 289
Summary 299

Chapter 10 • Arrays and Indexers 301

Declaring and Creating Arrays 301
Using Arrays 302
Accessing an Array Using a Loop 303
Attempting to Access a Nonexistent Array Element 305
Initializing Arrays 307
Reading Command-Line Arguments 309
Introducing Array Properties and Methods 311
Sorting Array Elements Using the Sort() Method 312
Searching for an Array Element Using the BinarySearch() Method 314
Reversing the Elements of an Array Using the Reverse() Method 315
Searching for Array Elements Using the IndexOf() and LastIndexOf() Methods 315
Using Multidimensional Arrays 320
Two-Dimensional Rectangular Arrays 320
Three-Dimensional Rectangular Arrays 325
Jagged Arrays 327
Creating Arrays of Objects 330
Introducing Indexers 332
Defining an Indexer 333
Reading from the Fields of an Object Using an Indexer 335
Writing to the Fields of an Object Using an Indexer 336
Summary 338

Chapter 11 • Collections 339
Introducing Array Lists 339
Creating and Using an ArrayList 340
ArrayList Properties and Methods 342
Adding Objects to an ArrayList 355
Understanding Bit Arrays 364
Creating and Using a BitArray 364
BitArray Properties and Methods 366
Understanding Hash Tables 370
Creating and Using a Hashtable 371
Hashtable Properties and Methods 373
Understanding Sorted Lists 379
Creating and Using a SortedList 379
SortedList Properties and Methods 381
Understanding Queues 388
Creating and Using a Queue 388
Queue Properties and Methods 390
Understanding Stacks 391
Creating and Using a Stack 391
Stack Properties and Methods 393
Summary 394

Chapter 12 • Delegates and Events 397
Understanding Delegates 397
Declaring a Delegate Class 397
Creating and Using Delegate Objects 398
Delegate Multicasting 401
Calling Object Methods Using a Delegate 405
Understanding Events 409
Declaring an Event 409
Declaring the Delegate Class Used with an Event 410
Declaring the Reactor Class 411
Declaring the ReactorMonitor Class 412
Creating and Using a Reactor and ReactorMonitor Object 413
Summary 416

Chapter 13 • Exceptions and Debugging 417
Handling Exceptions 417
Using Try/Catch Blocks 418
Using Finally Blocks 418
Understanding Exception Objects 420
Handling Specific Exceptions 422
Using One catch Block 423
Using Multiple catch Blocks 425
Exploring Exception Propagation 427
Exception Propagation with a Nested try/catch Block 427

Exception Propagation with Methods 429
Unhandled Exceptions 432
Creating and Throwing Exception Objects 433
Declaring Custom Exceptions 435
Debugging 437
Creating the Program 437
Creating the New VS .NET Project 438
Debugging the Program 441
Summary 446

Part 2 • Advanced C# Programming 449

Chapter 14 • Threads 451
Understanding the .NET Framework Class Library 452
Introducing the Namespaces in the Class Library 452
Exploring Namespaces 455
Understanding Threads 458
Creating Threads 459
Setting Thread Properties and Methods 461
Setting Thread Priorities 463
Retrieving Thread States 465
Using Data Slots 468
Managing Threads 469
Using the Timer Class 470
Using the Join Method 471
Using Locks 472
Using the Interlocked Class 476
Using the Monitor Class 478
Using the Mutex Class 480
Dealing with Thread Problems 482
Avoiding Deadlocks 482
Avoiding Race Conditions 483
Thread Pooling 483
Summary 485

Chapter 15 • Streams and Input/Output 487
Dealing with Files and Directories 487
Browsing for Files 488
Retrieving File Information 491
Retrieving Directory Information 497
Walking the Hierarchy 501
Watching for Changes 502
Exploring Streams and Backing Stores 505
Stream 506
FileStream 507
NetworkStream 510
MemoryStream 515

BufferedStream 518
CryptoStream 521
Using Readers and Writers 521
BinaryReader and BinaryWriter 521
TextReader and TextWriter 524
StreamReader and StreamWriter 524
StringReader and StringWriter 527
Using Asynchronous I/O 529
Introducing Serialization 532
Summary 537

Chapter 16 • Assemblies 539

Looking at the Big Picture 539
Why Use Assemblies? 540
Enhanced Versioning 540
Side-by-Side Execution 541
What's in an Assembly? 541
The Assembly Manifest 541
Type Metadata 542
MSIL Code 542
Resources 542
Building Assemblies 542
Assembly Attributes 542
Single-File Assemblies 545
Multifile Assemblies 550
Viewing Assembly Contents 555
Understanding Strong Names and Signing 557
Strong Names 557
Code Signing 559
Assembly Versioning 561
Understanding Version Numbers 562
Retrieving Version Numbers 562
Version Compatibility and Policy Files 563
Working with the Global Assembly Cache 566
Finding an Assembly 567
Summary 568

Chapter 17 • Attributes and Reflection 569

Using Attributes 569
Using Intrinsic Attributes 570
Using Custom Attributes 574
Discovering Types at Run-Time 580
Building a Run-Time Library 581
Discovering Type Information 584
Late Binding via Reflection 593
What You Can Find with Reflection 595

Creating Types at Run-Time 596
Summary 601

Chapter 18 • Remoting 603

Understanding Application Domains 603
Creating an Application Domain 604
Using an Object in an Application Domain 606
Unloading an Application Domain 608
Understanding Marshaling with Proxies 609
Marshaling by Value 610
Marshaling by Reference 610
Understanding Contexts 612
Understanding Channels 613
Using Remoting 614
Defining an Interface 614
Building the Server 615
Building the Client 616
Testing the Code 618
Summary 618

Chapter 19 • Security 619

Using Code-Access Security 619
Understanding Permissions 620
Requesting Minimum Permissions 622
Code Groups 623
Permission Sets 624
Granting Permissions 624
Computing Permissions 626
Requesting Optional Permissions 628
Requesting Permission Sets 629
Refusing Permissions 629
Demanding Permissions 630
Using Role-Based Security 631
Identity and Principal Objects 632
Verifying Role Membership 634
Using the PrincipalPermission Class 635
Using Encryption 636
Symmetric and Asymmetric Cryptography 636
Encrypting a File 637
Decrypting a File 640
Using Asymmetric Cryptography 642
Summary 643

Chapter 20 • XML 645

Understanding XML 645
Introducing XML 646

Introducing XSLT 654
Introducing XSD 657
Reading and Writing XML 659
Writing XML Files 659
Reading XML Files 662
Using the Document Object Model 663
Understanding the Document Object Model 663
Introducing the XmlNode Class 663
Introducing the XmlDocument Class 666
Reading an XML Document with the XmlTextReader Class 668
Transforming XML 672
Summary 675

Chapter 21 • Other Classes in the Base Class Library 677

Understanding the Graphics Classes 678
Introducing the GDI+ 678
Using Pens, Lines, and Rectangles 678
Filling Shapes with a Brush 689
Working with Images 692
Supporting Globalization 695
Overview of Localization and Globalization 695
Understanding Cultures 696
Displaying Localized Information 704
Diagnostics and Debugging 705
The Trace and Debug Classes 706
Using Trace Listeners 708
Using Trace Output After Deployment 710
Using Advanced Facilities 711
Controlling the Garbage Collector 712
Starting and Stopping Services 713
Working with Active Directory 716
Summary 718

Part 3 • .NET Programming with C# 719

Chapter 22 • Introduction to Databases 721

Introducing Databases 721
Exploring the Northwind Database 722
Primary Keys 723
Foreign Keys 723
Null Values 724
The Customers Table 724
The Orders Table 726
The Order Details Table 728
The Products Table 728

Using the Structured Query Language (SQL) 729
Using the Query Analyzer 730
Understanding Data Manipulation Language (DML) Statements 732
Introducing Stored Procedures 751
Looking at an Example SQL Server Stored Procedure 751
Running a SQL Server Stored Procedure 752
Accessing a Database Using Visual Studio .NET 752
Summary 755

Chapter 23 • Active Data Objects: ADO.NET 757

Overview of ADO.NET 757
Overview of the ADO.NET Classes 758
Generic Data Classes and Objects 758
Managed Provider Classes and Objects 759
Performing a SQL SELECT Statement Using ADO.NET 760
Step 1: Formulate a String Containing the Details of the Database Connection 761
Step 2: Create a SqlConnection Object to Connect to the Database 761
Step 3: Formulate a String Containing the SELECT Statement 761
Step 4: Create a SqlCommand Object to Hold the SELECT Statement 762
Step 5: Set the CommandText Property of the SqlCommand Object to the SELECT String 762
Step 6: Create a SqlDataAdapter Object 762
Step 7: Set the SelectCommand Property of the SqlAdapter Object to the SqlCommand Object 762
Step 8: Create a DataSet Object to Store the Results of the SELECT Statement . . . 762
Step 9: Open the Database Connection Using the Open() Method of the SqlConnection Object 763
Step 10: Call the Fill() Method of the SqlDataAdapter Object to Retrieve the Rows from the Table 763
Step 11: Get the DataTable Object from the DataSet Object 763
Step 12: Display the Columns for Each Row in the DataTable 764
Step 13: Close the Database Connection 764
Connecting to a Microsoft Access Database 767
Connecting to an Oracle Database 768
Exploring the Details of the ADO.NET Classes 768
SqlConnection Class 768
SqlCommand Class 769
SqlDataReader Class 770
SqlDataAdapter Class 773
SqlTransaction Class 774
DataSet Class 775
DataTable Class 776
DataRow Class 778
DataColumn Class 779
DataRelation Class 780
Constraint Class 781
DataView Class 781

Performing SQL INSERT, UPDATE, and DELETE Statements Using ADO.NET 783
Step 1: Formulate a String Containing the SQL Statement 783
Step 2: Create a SqlCommand Object to Hold the SQL Statement 784
Step 3: Set the CommandText Property of the SqlCommand Object to the SQL String 784
Step 4: Use the Add() Method to Add the Parameters 784
Step 5: Set the Parameters to Specified Values Using the Value Property 786
Step 6: Use the ExecuteNonQuery() Method to Run the SQL Statement 786
Example Program 787
Modifying a DataTable Object and Synchronizing the Changes with the Database 793
Adding a New Row to a DataTable Object 793
Modifying a Row in a DataTable Object 794
Removing a Row from a DataTable Object 795
Examining an Example Program 796
Using a Transaction in ADO.NET 801
Using a DataView Object to Filter and Sort Rows 804
Defining and Using a Relationship between Two DataTable Objects 807
Running a SQL Server Stored Procedure Using ADO.NET 811
Writing and Reading XML Files Using ADO.NET 814
Using the WriteXml() Method 814
Using the WriteXmlSchema() Method 815
Using the ReadXml() Method 815
Summary 818

Chapter 24 • Introduction to Windows Applications 819

Developing a Simple Windows Application 820
Creating the Windows Application 820
Examining the Form1.cs File 824
Working with the Solution Explorer 828
Working with the Class View 829
Using Windows Controls 830
Using a DataGrid Control to Access a Database 831
Using the Data Form Wizard to Create a Windows Form 836
Data Binding 842
Adding Controls to the Form 843
Adding the Main() Method 844
Setting the pwd Property 844
Running the Form 845
Summary 846

Chapter 25 • Active Server Pages: ASP.NET 847

Creating a Simple ASP.NET Web Application 847
The WebForm1.aspx File 850
The WebForm1.aspx.cs File 852
Using Web Form Controls 854
Building a More Complex Application 856

Using a DataGrid Control to Access a Database 861
Creating the Web Application 861
Customizing the DataGrid 864
Using a DataList Control to Access a Database 872
Summary 879

Chapter 26 • Web Services 881
Exploring the Architecture of Web Services 881
Building a Simple Web Service 883
Understanding Discovery 884
Understanding Description 886
Using the Web Service 892
Watching the Conversation 893
Looking Inside the Web Services Proxy 895
Building a More Complex Web Service 897
Building a Client the Easy Way 898
Exploring Web Services Registries 900
Summary 901

Appendices 903

Appendix A • C# Keywords 903

Appendix B: • C# Compiler Options 909

Appendix C • Regular Expressions 921
Concepts 921
Metacharacters 922
The Regular Expression Classes 925
Regular Expression Examples 926
Groups and Captures 930

Index. 935

Introduction

WELCOME TO *MASTERING VISUAL C# .NET*! As you may already know, .NET is poised to become *the* hot platform for the next wave of technology deployment. .NET's strength is that it is built from the ground up to be used in a distributed environment—in other words, an environment that consists of computers and devices connected via a network.

Microsoft has pledged its commitment and resources to making .NET a pervasive component of life in our technological society—you ignore .NET at your own peril. The bottom line is you need to learn .NET if you want to remain competitive in today's—and tomorrow's—marketplace.

In a nutshell, .NET is a completely new framework for writing many types of applications. Some of the applications you can write using .NET include Windows applications and web-based applications. You can use .NET to develop systems composed of interconnected services that communicate with each other over the Internet.

In addition, you can use .NET to create applications that run on devices such as handheld computers and cellular phones. Although other languages allow you to develop such applications, .NET was designed with the interconnected network in mind.

The .NET Framework consists of the following three primary components:

Development Languages and Tools The development languages that enable you to write .NET programs include C#, Visual Basic .NET (VB .NET), and Managed C++. Microsoft also has a Rapid Application Development (RAD) tool called Visual Studio .NET (VS .NET) that allows you to develop programs in an integrated development environment (IDE). You'll learn how to use C# and VS .NET in this book.

Common Language Runtime The Common Language Runtime (CLR) manages your running code and provides services such as memory management, thread management (which allows you to perform multiple tasks in parallel), and remoting (which allows objects in one application to communicate with objects in another application). The CLR also enforces strict safety and accuracy of your executable code to ensure no tampering occurs.

Framework Base Class Library The Framework Base Class Library is an extensive collection of code written by Microsoft that you can use in your own programs. For example, the Framework Base Class Library contains code that allows you to develop Windows applications, access directories and files on disk, connect to databases and retrieve information, and send and receive data across a network, among many other functions. You'll use the most important classes in the Framework Base Class Library in this book.

C# is one of languages you can use to write .NET applications. C# owes its heritage to languages such as Java, C, and C++—and, of course, those languages arose from the development of earlier languages. If you've programmed in Java or C++, you'll find C# easy to learn. C# isn't the only language for writing .NET applications. It is our opinion, however, that C# will be the primary language of choice for .NET development.

You might be wondering if you should learn C# or Java. Our answer is that you should learn both. Just because .NET is hot, it doesn't mean Java will be displaced any time soon. Both languages will coexist and evolve together—until something better comes along.

Who Should Read This Book?

This book was written for budding C# programmers. This book contains everything you need to know to master C#. No prior programming experience is assumed, but if you already know a programming language such as Java, C++, or Visual Basic, you'll be off to a running start. You may even know C# already; if so, you'll find that this book delves deeper into the advanced C# and .NET topics than most other books.

If you're a novice, you'll find Part I of this book easy to follow, and you'll get up to speed on the fundamental programming concepts used in C#. If you're an intermediate to advanced programmer, you'll be able to skim through the first five chapters of Part I quickly. Once you've mastered the C# language covered in Part I, you'll find Parts II and III of this book ideal for understanding the more advanced aspects of C# and .NET.

How to Use This Book

This book is divided up into three parts.

Part 1: "Fundamental C# Programming"

In Part I of this book, "Fundamental C# Programming," you'll learn everything you need to know to write simple C# programs. Part I consists of 13 chapters.

In Chapter 1, "Introduction to Visual C# and the .NET Framework," you'll be introduced to the C# language. You also learn about Microsoft's RAD tool, Visual Studio .NET. Visual Studio .NET allows you to develop programs in an IDE. Finally, you'll see how to use the extensive documentation from Microsoft that comes with .NET.

In Chapter 2, "Basic C# Programming," you'll learn how to write simple programs, how to store information in memory, and how to handle keyboard input and format screen output.

In Chapter 3, "Expressions and Operators," you'll explore how to use expressions and operators. An *expression* is typically a statement that performs some kind of calculation—using an *operator*—and returns a value.

In Chapter 4, "Decisions, Loops, and Preprocessor Directives," you'll learn how to execute different branches of code based on decisions and how to repeat statements using loops. You'll also learn about the preprocessor, which is program that reads your program source files prior to compilation. You can give the preprocessor instructions, known as *directives*, which can affect what parts of your program are actually compiled.

In Chapter 5, "Object-Oriented Programming," and Chapter 6, "More about Classes and Objects," you'll explore how to define your own types using classes and structs—and how to use them in your programs. You'll see how you can create objects from classes. You'll also learn how you can use classes and objects to model things that exist in the real world and how they can simplify the task of writing programs to solve complex problems.

In Chapter 7, "Derived Classes," you'll learn how a class may be derived from another class. You'll also learn how you can overload an operator to perform your own code when a particular operator is used, rather than using the default provided by C#.

In Chapter 8, "Interfaces," you'll explore interfaces. An *interface* contains a list of declarations such as method and property declarations. You can then have your class implement a given interface; your class must then define the code for the declarations specified in that interface. By having your class implement an interface, you're guaranteeing that your class contains the code for the items declared in that interface.

In Chapter 9, "Strings, Dates, Times, and Time Spans," you'll examine the use of strings and how you manipulate strings in your programs. You'll also learn how to represent dates and times in your programs.

In Chapter 10, "Arrays and Indexers," you'll learn about *arrays,* which are sets of consecutive of storage compartments that can hold data elements of the same type. You use arrays to store multiple variables or objects.

In Chapter 11, "Collections," you'll explore collections. *Collections* are objects that store many elements and whose capacity you can change after you've created them. These objects offer flexible ways to access their elements. As you master C#, you'll find collections to be useful when you need to perform complex manipulation of data.

In Chapter 12, "Delegates and Events," you'll learn about delegates and events. A *delegate* acts like a pointer to a function, and you can use them to call different functions that you specify at run-time. Delegates are closely tied to events—*events* are in fact a special kind of delegate. You can use events to send notifications that something has occurred to a particular object.

In Chapter 13, "Exceptions and Debugging," you'll explore exceptions and debugging. When an error or abnormal condition occurs during your program's execution, an *exception* is thrown. An example of a program error, or *bug,* might be attempting to divide a number by zero. An example of an abnormal program condition might be running out of memory.

Part 2: "Advanced C# Programming"

In Part II of this book, "Advanced C# Programming," you'll learn about the more complex aspects of C# programming. Part II consists of 8 chapters.

In Chapter 14, "Threads," you'll learn how to use threads, which allow you to perform multiple tasks in parallel. You'll learn that there are some problems to consider when your application uses multiple threads. In particular, you need to worry about synchronizing threads and handling deadlocks that can arise when two threads are fighting for resources to do their work.

In Chapter 15, "Streams and Input/Output," you'll explore how to work with input/output streams. You'll look at basic file and directory operations, such as opening a file and examining the contents of a directory. You'll also learn how to read and write data in a variety of ways, including ways to use the network as a data transport.

In Chapter 16, "Assemblies," you'll examine how .NET groups code into units known as *assemblies*. .NET uses assemblies for version tracking, security, deployment, and type identity—among other things. You'll learn how to create assemblies and how to include more than one file in the same assembly.

In Chapter 17, "Attributes and Reflection," you'll learn about metadata, which describes the contents of assemblies. Metadata consists of attributes that describe types, members, modules, and assemblies. You'll explore intrinsic attributes that are provided by default and custom attributes; you'll see how to create of your own custom attributes to extend the metadata stored in assemblies.

In Chapter 18, "Remoting," you'll explore remoting, which allows objects in one application to communicate with objects in another application. Remoting allows you to develop distributed software components that interact with each other.

In Chapter 19, "Security," you'll learn about .NET's security system. You can use code-access security to specify exactly which operations an application is allowed to perform. Code-access security allows you to select individual permissions to apply to code based on the code's membership in code groups. A code group can be based on the publisher of the code, the identity of the code, the location from which the code was obtained, or other factors. .NET also includes a role-based security system that lets you determine whether a particular user should have access to a resource.

In Chapter 20, "XML," you'll explore how to use the Extensible Markup Language (XML) with .NET. .NET uses XML for many tasks, from storing configuration files to serializing objects for transmission to a remote application. You can read and write XML files, analyze their structure, transform XML files into new formats, or use them to work with data.

In Chapter 21, "Other Classes in the Base Class Library," you'll learn about some of the other .NET classes contained in the Framework Base Class Library. You'll explore the graphics classes that allow you to draw elements, globalization classes that allow your software to be used in other locales and other languages, debug and trace classes that allow you to see debugging information either at design-time or run-time, along with advanced classes that support the features of the .NET runtime itself.

Part 3: ".NET Programming with C#"

In Part III of this book, ".NET Programming with C#," you'll be introduced to databases, you'll learn how to access a database using a C# program, and you'll see how to program Windows and web applications. Part III consists of 5 chapters.

In Chapter 22, "Introduction to Databases," you'll learn the basics of databases and see how to use the Structured Query Language (SQL) to access a database. This chapter shows how to access a SQL Server database named Northwind. This database contains the information for the fictitious Northwind Company, which sells food products.

In Chapter 23, "Active Data Objects: ADO.NET," you'll explore how to access a database from a C# program using ADO.NET. ADO.NET is Active Data Objects for the .NET Framework. You'll learn how to connect to a database and issue SQL statements that retrieve and modify the information stored in a database.

In Chapter 24, "Introduction to Windows Applications," you'll learn about Windows programming. A Windows program takes advantage of the graphical environment for display and use of the mouse, as well as the keyboard, for input. Windows provides graphical items such as menus, text boxes, radio buttons, and text boxes that allow you to build a visual interface that is easy to use.

In Chapter 25, "Active Server Pages: ASP.NET," you'll explore Active Server Pages for .NET (ASP.NET). ASP.NET allows you to create web pages whose content may change at run-time and allows you to develop applications that are accessed using a web browser. For example, you could develop an application that allows users to order products over the Web, or you could create a stock-trading application that allows users to place trades for shares in companies.

In Chapter 26, "Web Services," you'll learn how to build a simple web service. A *web service* is a software component that delivers information over the Internet (or a private intranet) to fulfill a specific need. For example, you could build a web service that returns the current price of a company's stock.

Appendixes

This book also contains three appendixes for reference purposes.

Appendix A, "C# Keywords," summarizes the keywords used in the C# language.

Appendix B, "C# Compiler Options," lists the options used with the C# compiler, which enable you to compile your programs.

Appendix C, "Regular Expressions," summarizes the C# regular expressions, which allow you to search for a specified set of characters or pattern of characters.

Downloading the Example Programs

Throughout this book, you'll see many example programs that illustrate the concepts described in the text. These are marked with a listing number and title, such as the one shown here:

LISTING 1.1: THE "HELLO WORLD" PROGRAM

You can download a ZIP file containing these programs from the Sybex website at `www.sybex.com`. You can use a program such as WinZip to extract the contents of the ZIP file. The filenames will correspond to the listing numbers.

Part 1

Fundamental C# Programming

In this section you will find:

- Chapter 1: Introduction to C#
- Chapter 2: Basic C# Programming
- Chapter 3: Expressions and Operators
- Chapter 4: Decisions, Loops, and Preprocessor Directives
- Chapter 5: Object-Oriented Programming
- Chapter 6: More about Classes and Objects
- Chapter 7: Derived Classes
- Chapter 8: Interfaces
- Chapter 9: Strings, Dates, Times, and Time Spans
- Chapter 10: Arrays and Indexers
- Chapter 11: Collections
- Chapter 12: Delegates and Events
- Chapter 13: Exceptions and Debugging

This page intentionally left blank

Chapter 1

Introduction to C#

In this chapter, you'll be introduced to the C# language. You'll see a simple example program that displays the words *Hello World!* on your computer's screen, along with the current date and time.

You'll also learn about Microsoft's Rapid Application Development (RAD) tool, Visual Studio .NET. Visual Studio .NET enables you to develop and run programs in an integrated development environment. This environment uses all the great features of Windows, such as the mouse and intuitive menus, and increases your productivity as a programmer.

In the final sections of this chapter, you'll see how to use the extensive documentation from Microsoft that comes with the .NET Software Development Kit (SDK) and Visual Studio .NET. This documentation goes well beyond the text of this book, and you'll find it invaluable as you become an expert C# programmer.

NOTE *Before you can develop C# programs, you'll need to install the .NET SDK or Visual Studio .NET. You can download the .NET SDK at* `http://msdn.microsoft.com/downloads`*. Once you've downloaded the executable file, go ahead and run it. Follow the instructions on the screen to install the .NET SDK on your computer. You can also purchase a copy of Visual Studio .NET from Microsoft at their website.*

Featured in this chapter:

- Building Your First C# Program
- Learning about Visual Studio .NET
- Working with the .NET Documentation

Developing Your First C# Program

Learning a new programming language is sometimes a daunting task. To get you started, you'll begin with a variation on the classic "Hello World" program. This program traditionally starts all programming books—and who are we to argue with tradition?