

# Preface

---

*“Live in fragments no longer, only connect.”*  
—Edgar Morgan Forster

Welcome to Visual C#® 2008 and the world of Microsoft® Windows® and Internet and web programming with Microsoft’s .NET Framework 3.5 platform! This book presents leading-edge computing technologies for students, instructors, software developers and IT professionals.

We use the Deitel signature “live-code approach,” presenting most concepts in the context of complete working Visual C# 2008 programs, rather than using code snippets. Each code example is immediately followed by one or more sample executions. All the source code is available at [www.deitel.com/books/csharp3/](http://www.deitel.com/books/csharp3/).

At Deitel & Associates, we write programming-language textbooks and professional books for Prentice Hall, deliver corporate training courses worldwide and develop Web 2.0 Internet businesses. We have updated the previous edition of this book based on Visual Studio 2008 and .NET 3.5.

## New and Updated Features

Here are the updates we’ve made for *Visual C# 2008 How to Program, 3/e*:

- **LINQ.** Many Microsoft technical evangelists say that LINQ (Language Integrated Query) is the single most important new feature in Visual C# 2008 and Visual Basic 2008. LINQ provides a uniform syntax for querying data. Strong typing enables Visual Studio to provide *IntelliSense* support for LINQ operations and results. LINQ can be used on different types of data sources, including collections and files (LINQ to Objects, Chapters 9 and 19, respectively), databases (LINQ to SQL, Chapters 21–23) and XML (LINQ to XML, Chapters 20 and 24).
- **Early Introduction to LINQ and Generic Collections.** We introduce LINQ early in the book so that you can begin using it as soon as you’ve been introduced to data structures—our LINQ coverage begins immediately after Chapter 8 on arrays. To enable you to work with more flexible data structures throughout the book, we also now introduce the `List` generic collection—a dynamic data structure—in close proximity to arrays. This enables us to demonstrate the power of LINQ and how it can be applied to most data structures. In addition, the `List` class is a generic collection, which provides strong compile-time type safety—ensuring that all elements of the collection are of the appropriate type.
- **Databases.** We use the free Microsoft SQL Server Express Edition and real-world applications to teach the fundamentals of database programming. Chapters 21–23 discuss database and LINQ to SQL fundamentals, presented in the context of an

address-book desktop application, a web-based bookstore application and a web-based airline reservation system, respectively. Chapter 21 also demonstrates using the Visual Studio tools to build a GUI application that accesses a database using LINQ to SQL.

- ***Windows Presentation Foundation (WPF) GUI and Graphics.*** Graphical user interfaces (GUIs) and graphics make applications fun to create and easier to use. We begin our GUI discussion with the traditional Windows Forms controls in Chapters 14–15. We extend our coverage in Chapters 16 and 17, respectively, with an introduction to Windows Presentation Foundation (WPF)—Microsoft’s new framework that integrates GUI, graphics and multimedia capabilities. To demonstrate WPF GUI and graphics capabilities we present many examples, including a painting application, a text editor, a color chooser, a book-cover viewer, a television video player, a 3-D rotating pyramid and various animations.
- ***Windows Communication Foundation (WCF) Web Services.*** Microsoft’s .NET strategy embraces the Internet and web as integral to software development and deployment. Web-services technology enables information sharing, e-commerce and other interactions using standard Internet protocols and technologies, such as Hypertext Transfer Protocol (HTTP), Extensible Markup Language (XML) and Simple Object Access Protocol (SOAP). Web services enable programmers to package application functionality in a manner that turns the web into a library of reusable software components. We replaced our treatment of ASP.NET web services from the previous edition with a discussion of Windows Communication Foundation (WCF) services in Chapter 23. WCF is a set of technologies for building distributed systems in which system components communicate with one another over networks. In earlier versions of .NET, the various types of communication used different technologies and programming models. WCF uses a common framework for all communication between systems, so you need to learn only one programming model. Chapter 23 focuses on WCF web services that use either the SOAP protocol or REST (Representational State Transfer) architecture. The REST examples transmit both XML (eXtensible Markup Language) and JSON (JavaScript Object Notation).
- ***ASP.NET 3.5 and ASP.NET AJAX.*** The .NET platform enables developers to create robust, scalable web-based applications. Microsoft’s .NET server-side technology, ASP.NET 3.5, allows programmers to build web documents that respond to client requests. To enable interactive web pages, server-side programs process information that users input into HTML forms. ASP.NET provides enhanced visual programming capabilities, similar to those used in building Windows Forms for desktop programs. Programmers can create web pages visually, by dragging and dropping web controls onto web forms. Chapter 22 introduces these powerful technologies. We present a sequence of examples in which the student builds several web applications, including a web-based bookstore. The chapter culminates with an example that demonstrates the power of AJAX. Chapter 22 also discusses the ASP.NET Development Server (which enables you to test your web applications on your local computer), multitier architecture and web transactions. The chapter uses ASP.NET 3.5 and LINQ to build a guest-

book application that retrieves information from a database and displays it in a web page. We use the new `LinqDataSource` from a web application to manipulate a database. We use ASP.NET AJAX controls to add AJAX functionality to web applications to improve their responsiveness—in particular, we use the `UpdatePanel` control to perform partial-page updates.

- ***Silverlight.*** In Chapter 24, we introduce Silverlight, Microsoft’s technology for building Rich Internet Applications (RIA). Silverlight, a competitor to JavaFX and Adobe’s Flash and Flex technologies, allows programmers to create visually stunning, multimedia-intensive user interfaces for web applications using .NET languages such as Visual C#. Silverlight is a subset of WPF that runs in a web browser using a plug-in. One of Silverlight’s most compelling features is its ability to stream high-definition video. The chapter presents powerful multimedia applications, including a weather viewer, Flickr™ photo viewer, deep zoom book-cover collage and video viewer.
- ***New Language Features to Support LINQ.*** Many of the new Visual C# language features we cover in Chapter 10 were introduced to support LINQ. We show how to use extension methods to add functionality to a class without modifying the class’s source code. We enhanced our discussion of delegates (objects that hold method references) to support our discussion of C#’s new lambda expressions, which define anonymous functions. Lambda expressions can be used wherever delegates are needed—typically as arguments to method calls or to help create more powerful LINQ queries. You’ll learn how to use anonymous types to create simple classes that store data without writing a class definition—a feature used frequently in LINQ.
- ***Implicitly Typed Local Variables.*** When you initialize a local variable in its declaration, you can now omit the variable’s type—the compiler infers it from the type of the initializer value (introduced in Chapter 8). This is another feature used frequently in LINQ.
- ***Object and Collection Initializers.*** When creating a new object, you can use the new object initializer syntax to assign values to the new object’s properties (introduced in Chapter 10). Similarly, you can use the new collection initializer syntax (discussed in Chapter 9) to specify values for the elements of collections, just as you do with arrays.
- ***Auto-Implemented Properties.*** For cases in which a property of a class has a get accessor that simply returns a `private` instance variable’s value and a set accessor that simply assigns a value to the instance variable, C# now provides automatically implemented properties (also known as auto-implemented properties; introduced in Chapter 4). With an auto-implemented property, the C# compiler automatically creates a `private` instance variable and the get and set accessors for manipulating it. This gives you the software engineering benefits of having a property, but enables you to implement the property trivially.

We updated the entire text to reflect Microsoft’s latest release of Visual C# 2008. New items include:

- Screenshots updated to the Visual C# 2008 Express IDE.

- Updated keywords table (Chapter 3) to include the new contextual keywords—words that are considered keywords only in certain contexts. Outside those contexts, such keywords can still be used as valid identifiers. This minimizes the chance that older Visual C# code will break when upgrading to Visual C# 2008. Many of these contextual keywords are used with LINQ.
- Pointing out additional ways in which the IDE's *IntelliSense* helps you write code.
- Using implicitly typed local variables to determine the types of the control variables in many `foreach` statements.
- Using *DataTips* and visualizers to view object contents in the code window during debugging.
- Using LINQ to Objects to manipulate data in two file-processing examples.
- Using LINQ to SQL in all database-driven examples.

All of this has been carefully reviewed by distinguished academics and industry developers who worked with us on *Visual C# 2008 How to Program, 3/e*.

We believe that this book and its support materials will give students and professionals an informative, interesting, challenging and entertaining C# educational experience. We provide a suite of ancillary materials that help instructors maximize their students' learning experience.

As you read the book, if you have questions, send an e-mail to [deitel@deitel.com](mailto:deitel@deitel.com); we'll respond promptly. For updates on this book and the status of all supporting C# software, and for the latest news on all Deitel publications and services, visit [www.deitel.com](http://www.deitel.com). Sign up at [www.deitel.com/newsletter/subscribe.html](http://www.deitel.com/newsletter/subscribe.html) for the free *Deitel® Buzz Online* e-mail newsletter and check out our growing list of Visual C# and related Resource Centers at [www.deitel.com/ResourceCenters.html](http://www.deitel.com/ResourceCenters.html). Each week we announce our latest Resource Centers in the newsletter.

## Features

### *Early Classes and Objects Approach*

We introduce basic object-technology concepts and terminology in Chapter 1. Chapter 4 provides a carefully crafted, friendly introduction to classes and objects that gets students working with object orientation comfortably from the start. This chapter was developed with the guidance of a team of academic and industry reviewers. Chapters 5–8 have been carefully written with a friendly “early classes and objects approach.”

### *Carefully Tuned Treatment of Object-Oriented Programming in Chapters 10–12*

We performed a high-precision upgrade for *Visual C# 2008 How to Program, 3/e*. This edition is clearer and more accessible—especially if you are new to object-oriented programming.

### *Case Studies*

We include many case studies, some spanning multiple sections and chapters:

- GradeBook class in Chapters 4–8.

- Optional OOD/UML ATM system in the Software Engineering sections of Chapters 1, 3–8, 10 and 12. The complete code for the ATM is included in Appendix D.
- Time class in several sections of Chapter 10.
- Employee payroll application in Chapters 11–12.
- WPF painter application in Chapter 16.
- WPF text-editor application in Chapter 16.
- WPF color-chooser application in Chapter 16.
- WPF book cover viewer application in Chapter 16.
- WPF television application in Chapter 17.
- Address-book application in Chapter 21.
- Guestbook ASP.NET application in Chapter 22.
- Secure-books database ASP.NET application in Chapter 22.
- Airline reservation web service in Chapter 23.
- Blackjack web service in Chapter 23.
- Equation-generator web service and math-tutor application in Chapter 23.
- Silverlight weather-viewer application in Chapter 24.
- Silverlight Flickr™ photo-viewer application in Chapter 24.
- Silverlight deep zoom book-cover collage application in Chapter 24.
- Silverlight video-viewer application in Chapter 24.

### ***Integrated GradeBook Case Study***

To reinforce our early coverage of classes, we present an integrated case study using classes and objects in Chapters 4–8. We incrementally build a GradeBook class that represents an instructor's grade book and performs various calculations based on a set of student grades—finding the average, finding the maximum and minimum, and printing a bar chart. Our goal is to familiarize you with the important concepts of objects and classes through a real-world example of a substantial class. We develop this class from the ground up, constructing methods from control statements and carefully developed algorithms, and adding instance variables and arrays as needed to enhance the functionality of the class.

### ***The Unified Modeling Language (UML)—Using the UML 2.0 to Develop an Object-Oriented Design of an ATM***

The Unified Modeling Language™ (UML™) has become the preferred graphical modeling language for designing object-oriented systems. All the UML diagrams in the book comply with the UML 2.0 specification. We use UML class diagrams to visually represent classes and their inheritance relationships, and we use UML activity diagrams to demonstrate the flow of control in each of C#'s control statements.

This edition continues to include an optional (and highly recommended) case study on object-oriented design using the UML. The case study was reviewed by a distinguished team of OOD/UML academic and industry professionals, including leaders in the field from Rational (the creators of the UML and now a division of IBM) and the Object Man-

agement Group (responsible for maintaining and evolving the UML). In the case study, we design and fully implement the software for a simple automated teller machine (ATM).

The Software Engineering Case Study sections at the ends of Chapters 1, 3–8, 10 and 12 present a carefully paced introduction to object-oriented design using the UML. We introduce a concise, simplified subset of the UML 2.0, then guide the reader through a first design experience intended for the novice object-oriented designer/programmer. The case study is not an exercise; rather, it is an end-to-end learning experience that concludes with a detailed walkthrough of the complete C# code.

The Software Engineering Case Study sections help readers develop an object-oriented design to complement the object-oriented programming concepts they begin learning in Chapter 1 and implementing in Chapter 4. In the first of these sections at the end of Chapter 1, we introduce basic OOD concepts and terminology. In the optional Software Engineering Case Study sections at the ends of Chapters 3–8, we consider more substantial issues, as we undertake a challenging problem with the techniques of OOD. We analyze a typical requirements document that specifies the system to be built; we determine the classes needed to implement that system, determine the attributes the classes need to have, determine the behaviors the classes need to exhibit and specify how the classes must interact with one another to meet the system requirements. In Appendix D, we include a complete C# implementation of the object-oriented system that we design in the earlier chapters. We employ a carefully developed, incremental object-oriented design process to produce a UML model for our ATM system. From this design, we produce a substantial working C# implementation using key programming notions, including classes, objects, encapsulation, visibility, composition, inheritance and polymorphism.

### ***Object-Oriented Programming***

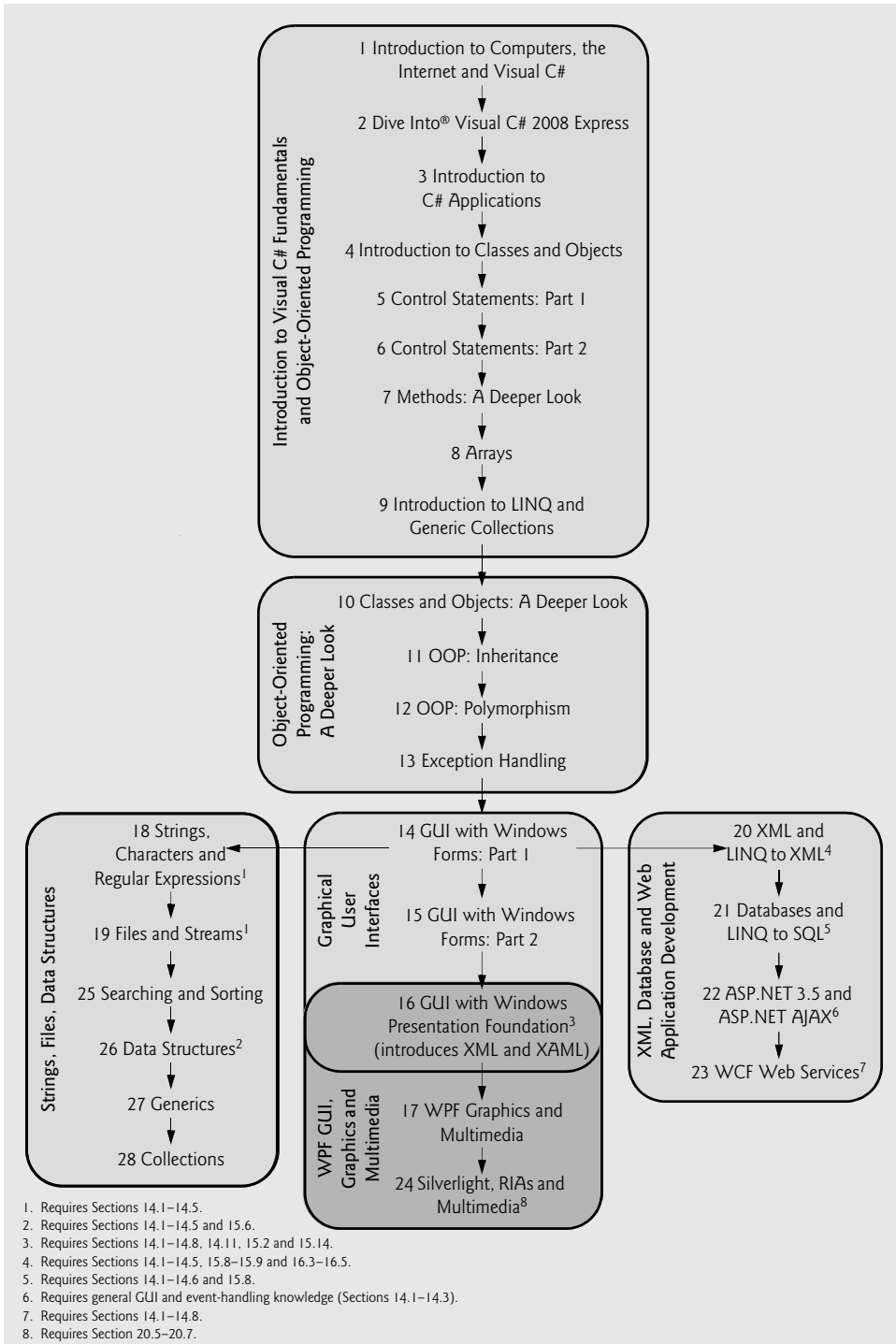
Object-oriented programming is today's most widely employed technique for developing robust, reusable software. This text offers a rich treatment of C#'s object-oriented programming features. Chapter 4 introduces how to create classes and objects. These concepts are extended in Chapter 10. Chapter 11 discusses how to create powerful new classes quickly by using inheritance to “absorb” the capabilities of existing classes. Chapter 12 familiarizes the reader with the crucial concepts of polymorphism, abstract classes, concrete classes and interfaces, all of which facilitate powerful manipulations among objects belonging to an inheritance hierarchy.

### ***Visual Studio 2008 Debugger***

In Appendix H, we explain how to use key debugger features, such as setting “break-points” and “watches” and stepping into and out of methods. Most of the material in this appendix can be covered after Chapter 4. One example uses the conditional AND (&&) operator, which is explained in Chapter 6.

## **Dependency Chart**

Figure 1 (on the next page) illustrates the dependencies among chapters in the book. An arrow pointing into a chapter indicates that it depends on the content of the chapter from which the arrow points. Though other approaches may work for you, we recommend that you study all of a given chapter's dependencies before studying that chapter. Some of the dependencies apply only to sections of chapters, so we advise readers to browse the



**Fig. 1** | Visual C# 2008 How to Program, 3/e chapter dependency chart.

material before designing a course of study. We've commented on some additional dependencies in the diagram's footnotes.

## Teaching Approach

*Visual C# 2008 How to Program, 3/e*, contains a rich collection of examples. The book concentrates on the principles of good software engineering and stresses program clarity. We teach by example. We are educators who teach leading-edge topics in industry classrooms worldwide. Dr. Harvey M. Deitel has 20 years of experience in college teaching and 19 years in industry teaching. Paul Deitel has taught 17 years in industry. The Deitels have taught courses at all levels to government, industry, military and academic clients of Deitel & Associates.

**Live-Code Approach.** *Visual C# 2008 How to Program, 3/e*, is loaded with “live-code” examples. By this we mean that each new concept is presented in the context of a complete working C# program, followed immediately by one or more actual executions showing the program's inputs and outputs. This style exemplifies the way we teach and write about programming; we call it the “live-code approach.”

**Syntax Shading.** We syntax-shade all the C# code, similar to the way Visual C# 2008 Express and Visual Studio syntax-color code. This greatly improves code readability—an especially important goal, given that this book contains approximately 20,000 lines of code. Our syntax-shading conventions are as follows:

*comments appear in italic*  
**keywords appear in bold italic**  
**errors and ASP.NET script delimiters appear in bold black**  
**constants and literal values appear in bold gray**  
 all other code appears in plain black

**Code Highlighting.** We place white rectangles around the key code segments in each program.

**Programming Tips.** We include programming tips to help you focus on important aspects of program development. These tips and practices represent the best we have gleaned from a combined six decades of programming and teaching experience. One of our students—a mathematics major—likens this approach to the highlighting of axioms, theorems and corollaries in mathematics books; it provides a basis on which to build good software.



### Good Programming Practice

Good Programming Practices *will help you produce programs that are clearer, more understandable and more maintainable.*



### Common Programming Error

Students *tend to make certain kinds of errors frequently. Pointing out these Common Programming Errors reduces the likelihood that you'll make the same mistakes.*



### Error-Prevention Tip

These tips *contain suggestions for exposing bugs and removing them from your programs; many describe aspects of C# that prevent bugs from getting into programs in the first place.*



### Performance Tip

---

Students like to “turbocharge” their programs. These tips highlight opportunities for making your programs run faster or minimizing the amount of memory that they occupy.



### Portability Tip

---

We include Portability Tips to help you write code that will run on a variety of platforms and to explain how C# achieves its high degree of portability.



### 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

---

We provide Look-and-Feel Observations to highlight graphical-user-interface conventions. These observations help you design attractive, user-friendly graphical user interfaces that conform to industry norms.

**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 C# program text in the Lucida font (e.g., `int x = 5`).

**Web Access.** All of the source-code examples for *Visual C# 2008 How to Program, 3/e* are available for download from:

[www.deitel.com/books/csharp3/](http://www.deitel.com/books/csharp3/)

Site registration is quick and easy. Download all the examples, then run each program as you read the corresponding text discussions. Making changes to the examples and seeing the effects of those changes is a great way to enhance your C# learning experience.

**Objectives.** Each chapter begins with a statement of objectives. This lets you know what to expect and gives you an opportunity, after reading the chapter, to determine if you have met the objectives.

**Quotations.** The learning objectives are accompanied by quotations. Some are humorous; some are philosophical; others offer interesting insights. We hope that you enjoy relating the quotations to the chapter material.

**Outline.** The chapter outline helps you approach the material in a top-down fashion, so you can anticipate what is to come and set an effective learning pace.

**Illustrations/Figures.** Abundant charts, tables, line drawings, programs and program output are included. We model the flow of control in control statements with UML activity diagrams. UML class diagrams model the fields, constructors and methods of classes. We make extensive use of six major UML diagram types in the optional OOD/UML 2 ATM case study.

**Wrap-Up Section.** Each chapter ends with a brief “wrap-up” section that recaps the chapter content and transitions to the next chapter.

*Summary Bullets.* Each chapter ends with additional pedagogical features. We present a thorough, bullet-list-style summary of the chapter, section by section.

*Terminology.* We include an alphabetized list of the important terms defined in each chapter. Each term also appears in the index, with its defining occurrence highlighted with a *bold italic* page number.

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

*Exercises.* Each chapter concludes with a set of exercises, including simple recall of important terminology and concepts; identifying the errors in code samples; writing individual C# statements; writing small portions of functions and classes; writing complete C# functions, classes and programs; and writing major term projects. The numerous exercises enable instructors to tailor their courses to the unique needs of their students and to vary course assignments each semester. Instructors can use these exercises to form homework assignments, short quizzes, major examinations and term projects. See our Programming Projects Resource Center ([www.deitel.com/ProgrammingProjects/](http://www.deitel.com/ProgrammingProjects/)) for additional exercise and project possibilities.

[*NOTE: Please do not write to us requesting access to the Prentice Hall Instructor's Resource Center, which contains the exercise solutions and the book's ancillaries. Access is limited strictly to college instructors teaching from the book. Instructors may obtain access only through their Prentice Hall representatives.*]

*Thousands of Index Entries.* We have included a comprehensive index, which is especially useful when you use the book as a reference.

*"Double Indexing" of Live-Code Examples.* For every source-code program in the book, we index the figure caption both alphabetically and as a subindex item under "Examples." This makes it easier to find examples using particular features.

## **A Tour of the Optional Case Study on Object-Oriented Design with the UML**

This section tours the book's optional case study on object-oriented design with the UML. We preview the contents of the Software Engineering Case Study sections (in Chapters 1, 3–8, 10, 12 and Appendix D). After completing this case study, you'll be thoroughly familiar with an object-oriented design and implementation for a significant C# application.

The design presented in the ATM case study was developed at Deitel & Associates, Inc., and scrutinized by a distinguished developmental review team of industry professionals and academics. We crafted this design to meet the requirements of introductory course sequences. Real ATM systems used by banks and their customers worldwide are based on more sophisticated designs that take into consideration many more issues than we have addressed here. Our primary goal was to create a simple design that would be clear to OOD and UML novices, while still demonstrating key OOD concepts and the related UML modeling techniques. We worked hard to keep the design and the code relatively small so that they would work well in the introductory course sequence.

**Section 1.19—(Required) Software Engineering Case Study: Introduction to Object Technology and the UML**—introduces the object-oriented design case study with the UML. The section presents basic concepts and terminology of object technology,

including classes, objects, encapsulation and inheritance. We discuss the history of the UML. This is the only required section of the case study.

**Section 3.10—(Optional) Software Engineering Case Study: Examining the ATM Requirements Document**—discusses a *requirements document* specifying the requirements for a system that we'll design and implement the software for a simple automated teller machine (ATM). We investigate the structure and behavior of object-oriented systems in general. We discuss how the UML will facilitate the design process in subsequent Software Engineering Case Study sections by providing several additional types of diagrams to model our system. We include a list of URLs and book references on object-oriented design with the UML. We discuss the interaction between the ATM system and its user. Specifically, we investigate the scenarios that may occur between the user and the system itself—these are called *use cases*. We model these interactions, using UML *use case diagrams*.

**Section 4.12—(Optional) Software Engineering Case Study: Identifying the Classes in the ATM Requirements Documents**—begins to design the ATM system. We identify its classes by extracting the nouns and noun phrases from the requirements document. We arrange these classes into a UML class diagram that describes the class structure of our simulation. The diagram also describes relationships, known as *associations*, among the classes.

**Section 5.14—(Optional) Software Engineering Case Study: Identifying Class Attributes in the ATM System**—focuses on the attributes of the classes discussed in Section 3.10. A class contains both *attributes* (data) and *operations* (behaviors). As we see in later sections, changes in an object's attributes often affect its behavior. To determine the attributes for the classes in our case study, we extract the adjectives describing the nouns and noun phrases (which defined our classes) from the requirements document, then place the attributes in the class diagram we created in Section 4.12.

**Section 6.10—(Optional) Software Engineering Case Study: Identifying Objects' States and Activities in the ATM System**—discusses how an object, at any given time, occupies a specific condition called a *state*. A *state transition* occurs when the object receives a message to change state. The UML provides the *state machine diagram*, which identifies the set of possible states that an object may occupy and models that object's state transitions. An object also has an *activity*—the work it performs in its lifetime. The UML provides the *activity diagram*—a flowchart that models an object's activity. This section uses both diagram types to model behavioral aspects of our ATM system, such as how it carries out a withdrawal transaction and how it responds when the user is authenticated.

**Section 7.15—(Optional) Software Engineering Case Study: Identifying Class Operations in the ATM System**—identifies the operations, or services, of our classes. We extract from the requirements document the verbs and verb phrases that specify the operations for each class. We then modify the class diagram of Section 4.12 to include each operation with its associated class. At this point in the case study, we will have gathered all information possible from the requirements document. As future chapters introduce such topics as inheritance, we'll modify our classes and diagrams.

**Section 8.14—(Optional) Software Engineering Case Study: Collaboration Among Objects in the ATM System**—provides a “rough sketch” of the model for our ATM system. In this section, we see how it works. We investigate the behavior of the simulation by discussing *collaborations*—messages that objects send to each other to communicate. The class operations that we identified in Section 7.15 turn out to be the collaborations among the objects in our system. We determine the collaborations, then collect them into

a *communication diagram*—the UML diagram for modeling collaborations. This diagram reveals which objects collaborate and when. We present a communication diagram of the collaborations among objects to perform an ATM balance inquiry. We then present the UML *sequence diagram* for modeling interactions in a system. This diagram emphasizes the chronological ordering of messages. A sequence diagram models how objects in the system interact to carry out withdrawal and deposit transactions.

**Section 10.22—(Optional) Software Engineering Case Study: Starting to Program the Classes of the ATM System**—takes a break from designing the behavior of our system. We begin the implementation process to emphasize the material discussed in Chapter 8. Using the UML class diagram of Section 4.12 and the attributes and operations discussed in Section 5.14 and Section 7.15, we show how to implement a class in C# from a design. We do not implement all classes—because we have not completed the design process. Working from our UML diagrams, we create code for the `Withdrawal` class.

**Section 12.9—(Optional) Software Engineering Case Study: Incorporating Inheritance and Polymorphism into the ATM System**—continues our discussion of object-oriented programming. We consider inheritance: classes sharing common characteristics may inherit attributes and operations from a “base” class. In this section, we investigate how our ATM system can benefit from using inheritance. We document our discoveries in a class diagram that models inheritance relationships—the UML refers to these relationships as *generalizations*. We modify the class diagram of Section 4.12 by using inheritance to group classes with similar characteristics. This section concludes the design of the model portion of our simulation.

**Appendix D—ATM Case Study Code**—The majority of the case study involves designing the model (i.e., the data and logic) of the ATM system. In this appendix, we fully implement that model in C#, using all the UML diagrams we created. We apply the concepts of object-oriented design with the UML and object-oriented programming in C# that you learned in the chapters. By the end of this appendix, you’ll have completed the design and implementation of a real-world system and should feel confident tackling larger systems, such as those that professional software engineers build.

**Appendix E—UML 2: Additional Diagram Types**—overviews the UML 2 diagram types not discussed in the OOD/UML Case Study.

## **Microsoft Developer Network Academic Alliance (MSDNAA) and Microsoft DreamSpark**

### ***Microsoft Developer Network Academic Alliance (MSDNAA)—Free Microsoft Software for Academic and Research Purposes***

The MSDNAA provides free software for academic and research purposes. For software direct to faculty, visit [www.microsoft.com/faculty](http://www.microsoft.com/faculty). For software for your department, visit [www.msdnaa.com](http://www.msdnaa.com).

### ***Microsoft DreamSpark—Professional Developer and Designer Tools for Students***

Microsoft provides many of its developer tools to students for free via a program called DreamSpark ([downloads.channe18.msdn.com/](http://downloads.channe18.msdn.com/)). At the time of this writing, the DreamSpark website states that students in the United States, the United Kingdom, Canada, China, Germany, France, Finland, Spain, Sweden, Switzerland and Belgium can obtain this software after their student status has been verified.

## Software for the Book

We use Microsoft Visual Studio 2008 development tools, including the free Visual C#<sup>®</sup> 2008 Express Edition, Visual Web Developer<sup>™</sup> 2008 Express Edition and SQL Server 2005 Express Edition. Per Microsoft's website, Microsoft Express Editions are "light-weight, easy-to-use and easy-to-learn tools for the hobbyist, novice and student developer." The Express Editions provide rich functionality and can be used to build robust .NET applications. They are appropriate for academic courses and for professionals who do not have access to a complete version of Visual Studio 2008.

You may use the Express Editions to compile and execute all the example programs and solve all the exercises in the book (with the exception of Chapter 24, whose software requirements are presented below in the Other Software Requirements section). You may also use the full Visual Studio product to build and run the examples and exercises. All of the features supported by the Express Editions are also available in the complete Visual Studio 2008 editions.

This book includes the Microsoft<sup>®</sup> Visual Studio<sup>®</sup> 2008 Express Editions All-in-One DVD, which contains the Visual C# 2008 Express Edition, the Visual Web Developer 2008 Express Edition and the SQL Server 2005 Express Edition. You can also download the latest versions of these tools from:

[www.microsoft.com/express/](http://www.microsoft.com/express/)

When you install the software (discussed in the Before You Begin section that follows this Preface), you also should install the help documentation and SQL Server 2005 Express. Microsoft provides a dedicated forum for help using the Express Editions at:

[forums.microsoft.com/msdn/ShowForum.aspx?siteid=1&ForumID=24](http://forums.microsoft.com/msdn/ShowForum.aspx?siteid=1&ForumID=24)

As this book was sent to publication, Microsoft released SQL Server 2008 Express, which you can now download and use with this book. The instructions we provide for using SQL Server 2005 Express with our examples also apply to the new version.

### *Windows Vista and Windows XP*

Readers of this book can use either Windows Vista or Windows XP. We used Windows Vista while developing the book. We use the Windows Vista Segoe UI font in the graphical user interfaces. This font is accessible to Windows XP users—we tell you how to get it in the Before You Begin section. Several of our full-book reviewers tested all the programs on Windows XP and reported no problems. If any Windows XP-specific issues arise after the book is published, we'll post them at [www.deitel.com/books/csharp3/](http://www.deitel.com/books/csharp3/) with appropriate instructions. Write to us at [deitel@deitel.com](mailto:deitel@deitel.com) if you encounter any problems, and we'll respond promptly.

### *Other Software Requirements*

For Chapters 21–23, you'll need the SQL Server 2005 Express Edition or SQL Server 2008 Express Edition. Chapters 22 and 23 require Visual Web Developer 2008 Express (or a full Visual Studio 2008 edition).

We present Microsoft Silverlight in Chapter 24. At the time of this writing Silverlight 2 was in beta, and the tools for developing Silverlight applications were available only for Visual Studio 2008 (not Express Editions); tools for developing Silverlight applications

with the Express Editions will be available soon. When the final tools become available, we'll post updates at [www.deitel.com/books/csharp3/](http://www.deitel.com/books/csharp3/).

For updates on the software used in this book, subscribe to our free e-mail newsletter at [www.deitel.com/newsletter/subscribe.html](http://www.deitel.com/newsletter/subscribe.html) and visit the book's website at [www.deitel.com/books/csharp3/](http://www.deitel.com/books/csharp3/). Also, be sure to visit our Visual C# 2008 Resource Center ([www.deitel.com/VisualCSharp2008/](http://www.deitel.com/VisualCSharp2008/)) frequently for new Visual C# 2008 resources. We've created Resource Centers for all of the major technologies discussed in this book ([www.deitel.com/resourcecenters.html](http://www.deitel.com/resourcecenters.html))—each week we announce our latest Resource Centers in the newsletter.

### **Instructor Resources for *Visual C# 2008 How to Program, 3/e***

*Visual C# 2008 How to Program, 3/e*, has extensive instructor resources. The Prentice Hall *Instructor's Resource Center* contains the *Solutions Manual* with solutions to the vast majority of the end-of-chapter exercises, a *Test Item File* of multiple-choice questions (approximately two per book section) and PowerPoint® slides containing the code and figures in the text, plus bulleted summaries of the key points in the text. Instructors can customize the slides. If you are not already a registered faculty member, contact your Prentice Hall representative or visit [vig.prenhall.com/replocator/](http://vig.prenhall.com/replocator/).

[NOTE: Please do not write to us requesting access to the Prentice Hall Instructor's Resource Center, which contains the exercise solutions and the book's ancillaries. Access is limited strictly to college instructors teaching from the book. Instructors may obtain access only through their Prentice Hall representatives.]

### **Deitel® Buzz Online Free E-mail Newsletter**

Each week, the *Deitel® Buzz Online* newsletter announces our latest Resource Center(s) and includes commentary on industry trends and developments, links to free articles and resources from our published books and upcoming publications, product-release schedules, errata, challenges, anecdotes, information on our corporate instructor-led training courses and more. It's also a good way for you to keep posted about issues related to *Visual C# 2008 How to Program, 3/e*. To subscribe, visit

[www.deitel.com/newsletter/subscribe.html](http://www.deitel.com/newsletter/subscribe.html)

### **The Deitel Online Resource Centers**

Our website [www.deitel.com](http://www.deitel.com) provides more than 100 Resource Centers on various topics including programming languages, software development, Web 2.0, Internet business and open-source projects—see the complete list of Resource Centers in the first few pages of this book or visit [www.deitel.com/ResourceCenters.html](http://www.deitel.com/ResourceCenters.html). The Resource Centers evolved out of the research we've done to support our books and business endeavors. We've found many exceptional resources online, including tutorials, documentation, software downloads, articles, blogs, podcasts, videos, code samples, books, e-books and more—most of them are free. Each week we announce our latest Resource Centers in our newsletter, the *Deitel® Buzz Online* ([www.deitel.com/newsletter/subscribe.html](http://www.deitel.com/newsletter/subscribe.html)). The following Resource Centers may be of interest to you as you study *Visual C# 2008 How to Program, 3/e*:

- Visual C# 2008
- ASP.NET
- ASP.NET 3.5
- ASP.NET AJAX
- Visual Studio Team System
- Code Search Engines and Code Sites
- Computer Game Programming
- Computing Jobs
- LINQ
- Popfly
- Open Source
- Programming Projects
- .NET
- .NET 3.0
- .NET 3.5
- Silverlight
- Silverlight 2.0
- SQL Server 2008
- Web Services
- Windows Communication Foundation
- Windows Presentation Foundation
- Windows Vista

## Acknowledgments

It's a pleasure to acknowledge the efforts of people whose names do not appear on the cover, but whose hard work, cooperation, friendship and understanding were crucial to the book's production. Many people at Deitel & Associates, Inc., devoted long hours to this project—thanks especially to Abbey Deitel and Barbara Deitel.

We would also like to thank the participants of our Honors Internship and Co-op programs who contributed to this publication—Greg Ayer, a Computer Science major at Northeastern University; Nicholas Doiron, an Electrical and Computer Engineering major at Carnegie Mellon University; Joseph Itkis, a Mathematics major (Computer Science track) at Yeshiva University; David Keyworth, an Information Technology major at Rochester Institute of Technology; Jehhal Liu, an Electrical and Computer Engineering major at Cornell University; Matthew Pearson, a Computer Science major at Cornell University; Bruce Tu, an Information Science major at Cornell University; Scott Wehrwein, a Computer Science major at Middlebury College; and H. Shawn Xu, a Biomedical Engineering and Economics double major at Johns Hopkins University.

We are fortunate to have worked on this project with the talented and dedicated team of publishing professionals at Prentice Hall. We appreciate the extraordinary efforts of Marcia Horton, Editorial Director of Prentice Hall's Engineering and Computer Science Division. Carole Snyder, Lisa Bailey and Dolores Mars did an extraordinary job recruiting the book's review team and managing the review process. Francesco Santalucia (an independent artist) and Kristine Carney of Prentice Hall did a wonderful job designing the book's cover—we provided the concept, and they made it happen. Scott Disanno, Robert Engelhardt and Marta Samsel did a marvelous job managing the book's production. Our marketing manager Chris Kelly and his boss Margaret Waples did a great job marketing the book through academic and professional channels.

We wish to acknowledge the efforts of our reviewers. Adhering to a tight time schedule, they scrutinized the text and the programs, providing countless suggestions for improving the accuracy and completeness of the presentation:

### **Visual C# 2008 How to Program, 3/e, Reviewers**

**Academic Reviewers:** Mingsheng Hong (Cornell University), Stan Kurkovsky, Ph.D. (Central Connecticut State University), Markus Lumpe (Swinburne University of Technology), and Gavin Osborne (Saskatchewan Institute of Applied Science and Technology). **Microsoft Reviewers:** Vinay Ahuja (Architect), Dan Crevier, Marcelo Guerra Hahn, Helena Kotas, Eric Lippert, Kyrylo Osenkov (Visual C#) and Alex Turner (Visual C# Compiler Program Manager). **Industry Reviewers:** Rizwan Ahmed a.k.a. RizwanSharp

(C# MVP, Sr. Software Engineer, TEO), José Alarcón-Aguín (ASP.NET MVP, Krasis.com), Mostafa Arafa (C# MVP, Agility Logistics), Bonnie Berent (Microsoft C# MVP), Adam Calderon (C# MVP, InterKnowlogy), Octavio Hernandez (C# MVP, Plain Concepts), Ged Mead (DevCity.Net, Microsoft VB MVP—Visual Developer) and José Antonio González Seco (Parliament of Andalusia).

### **Visual C# 2005 How to Program, 2/e, Reviewers**

**Academic Reviewers:** Rekha Bhowmik (California Lutheran University), Ayad Boudiab (Georgia Perimeter College), Harlan Brewer (University of Cincinnati), Sam Gill (San Francisco State University), Gavin Osborne (Saskatchewan Institute of Applied Science and Technology) and Catherine Wyman (DeVry-Phoenix). **Microsoft Reviewers:** George Bullock (Program Manager, Microsoft.com Community Team), Dharmesh Chauhan, Shon Katzenberger, Matteo Taveggia and Matt Tavis. **Industry Reviewers:** Alex Bondarev (Investor's Bank and Trust), Peter Bromberg (Senior Architect Merrill Lynch and C# MVP), Vijay Cinnakonda (TrueCommerce, Inc.), Jay Cook (Alcon Laboratories), Jeff Cowan (Magenic, Inc.), Ken Cox (Independent Consultant, Writer and Developer and ASP.NET MVP), Stochio Goutsev (Independent Consultant, writer and developer and C# MVP), James Huddleston (Independent Consultant), Rex Jaeschke (Independent Consultant and Editor of the *C# Standard ECMA-334, 2005*, produced by committee Ecma TC39/TG2), Saurabh Nandu (AksTech Solutions Pvt. Ltd.), Simon North (Quintiq BV), Mike O'Brien (State of California Employment Development Department), José Antonio González Seco (Andalusia's Parliament), Devan Shepard (XMaLpha Technologies), Pavel Tsekov (Caesar BSC), John Varghese (UBS) and Stacey Yasenka (Software Developer at Hyland Software and C# MVP).

Well, there you have it! Visual C# 2008 is a powerful programming language that will help you write programs quickly and effectively. It scales nicely into the realm of enterprise-systems development to help organizations build their business-critical and mission-critical information systems. As you read the book, we would sincerely appreciate your comments, criticisms, corrections and suggestions for improvement. Please address all correspondence to:

deitel@deitel.com

We'll respond promptly, and we'll post corrections and clarifications on the book's website:

[www.deitel.com/books/csharp3/](http://www.deitel.com/books/csharp3/)

We hope you enjoy reading *Visual C# 2008 How to Program, 3/e*, as much as we enjoyed writing it!

*Paul J. Deitel*

*Dr. Harvey M. Deitel*

### **About the Authors**

**Paul J. Deitel**, CEO and Chief Technical Officer of Deitel & Associates, Inc., has 21 years of experience in the computer field. Paul is a graduate of MIT's Sloan School of Management, where he studied Information Technology. Through Deitel & Associates, Inc., he

has delivered C#, Visual Basic, C++, C and Java courses to industry clients, including Cisco, IBM, Sun Microsystems, Dell, Lucent Technologies, Fidelity, NASA at the Kennedy Space Center, White Sands Missile Range, the National Severe Storm Laboratory, Rogue Wave Software, Boeing, Stratus, Hyperion Software, Adra Systems, Entergy, CableData Systems, Nortel Networks, Puma, iRobot, Invensys and many more. He holds the Sun Certified Java Programmer and Java Developer certifications and has been designated by Sun Microsystems as a Java Champion. He has lectured on Java and C++ for the Boston Chapter of the Association for Computing Machinery. He and his father, Dr. Harvey M. Deitel, are the world's best-selling programming-language textbook authors.

**Dr. Harvey M. Deitel**, Chairman and Chief Strategy Officer of Deitel & Associates, Inc., has 47 years of experience in the computer field. Dr. Deitel earned B.S. and M.S. degrees from MIT and a Ph.D. from Boston University. He has extensive college teaching experience, including earning tenure and serving as the Chairman of the Computer Science Department at Boston College before founding Deitel & Associates, Inc., with his son, Paul J. Deitel. Harvey and Paul are the co-authors of dozens of books and multimedia packages and they are writing many more. The Deitels' texts have earned international recognition with translations published in Japanese, German, Russian, Spanish, Traditional Chinese, Simplified Chinese, Korean, French, Polish, Italian, Portuguese, Greek, Urdu and Turkish. Dr. Deitel has delivered hundreds of professional seminars to major corporations, academic institutions, government organizations and the military.

## **About Deitel & Associates, Inc.**

Deitel & Associates, Inc., is an internationally recognized corporate training and authoring organization specializing in computer programming languages, Internet and web software technology, object-technology education and Internet business development through its Web 2.0 Internet Business Initiative. The company provides instructor-led courses on major programming languages and platforms, such as Visual C#, Visual Basic, Visual C++, C++, Java, C, XML, object technology, Internet and web programming, and a growing list of additional programming and software-development related courses. The founders of Deitel & Associates, Inc., are Paul J. Deitel and Dr. Harvey M. Deitel. The company's clients include many of the world's largest companies, government agencies, branches of the military, and academic institutions. Through its 32-year publishing partnership with Prentice Hall, Deitel & Associates, Inc., publishes leading-edge programming textbooks, professional books, interactive multimedia *Cyber Classrooms*, *LiveLessons* DVD-based and web-based video courses, and e-content for popular course-management systems. Deitel & Associates, Inc., and the authors can be reached via e-mail at:

[deitel@deitel.com](mailto:deitel@deitel.com)

To learn more about Deitel & Associates, Inc., its publications and its worldwide *Dive Into*<sup>®</sup> Series Corporate Training curriculum, see the last few pages of this book or visit:

[www.deitel.com](http://www.deitel.com)

and subscribe to the free *Deitel*<sup>®</sup> *Buzz Online* e-mail newsletter at:

[www.deitel.com/newsletter/subscribe.html](http://www.deitel.com/newsletter/subscribe.html)

Individuals wishing to purchase Deitel books, *LiveLessons* DVD and web-based training courses can do so through:

[www.deitel.com](http://www.deitel.com)

Bulk orders by corporations, the government, the military and academic institutions should be placed directly with Prentice Hall. For more information, visit

[www.prenhall.com/mishtml/support.html#order](http://www.prenhall.com/mishtml/support.html#order)