Preface

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

Welcome to Visual C#® 2010, C# 4 and the world of Microsoft® Windows® and Internet and web programming with Microsoft's .NET 4 Framework! This book presents leading-edge computing technologies for professional software developers. We believe the book will give you an informative, challenging and entertaining C# educational experience.

We use the Deitel signature **live-code approach**, presenting most concepts in the context of complete working Visual C# 2010 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/csharpfp4/`.

At Deitel & Associates, we author programming-language professional books, LiveLessons video courses and textbooks under the Prentice Hall imprint of Pearson Higher Education, and deliver our Dive Into® Series professional instructor-led training courses worldwide on site at corporations, government agencies, branches of the military and academic institutions.

As you read the book, if you have questions, send an e-mail to `deitel@deitel.com`; we'll respond promptly. For updates on this book and its supporting Visual C# software, visit `www.deitel.com/books/csharpfp4/`, follow us on Twitter (`@deitel`) and Facebook (`www.deitel.com/deitelfan`), and subscribe to the *Deitel® Buzz Online* newsletter (`www.deitel.com/newsletter/subscribe.html`). Check out our growing list of C# and related Resource Centers at `www.deitel.com/ResourceCenters.html`.

New and Updated Features

Here are some key features of *C# 2010 for Programmers, 4/e:*

- ***LINQ.*** 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 17, respectively), databases (LINQ to SQL, Chapters 18, 19, 27 and 28) and XML (LINQ to XML, Chapters 26 and 29).
- ***Early Introduction to Generic Collections and LINQ.*** We introduce LINQ early so that you can begin using it with arrays. To enable you to work with more flexible data structures throughout the book, we 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. As a generic collection, the `List` class 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 present the fundamentals of database programming. Chapters 18, 27 and 28 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. Chapter 18 also demonstrates using the Visual Studio 2010 tools to build a GUI application that accesses a database using LINQ to SQL.
- ***Windows Presentation Foundation (WPF) GUI and Graphics.*** We begin our GUI discussion with the traditional Windows Forms controls in Chapters 14–15. We extend our coverage in Chapters 24 and 25 with an introduction to Windows Presentation Foundation (WPF)—Microsoft's framework that integrates GUI, graphics and multimedia 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), Simple Object Access Protocol (SOAP) and REST (Representational State Transfer). Web services enable you 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 an earlier edition with a discussion of Windows Communication Foundation (WCF) services in Chapter 28. WCF is a set of technologies for building distributed systems in which system components communicate with one another over networks. WCF uses a common framework for all communication between systems, so you need to learn only one programming model. Chapter 28 focuses on WCF web services that use either the SOAP protocol or REST architecture. The REST examples transmit both XML (eXtensible Markup Language) and JSON (JavaScript Object Notation).
- ***ASP.NET 4 and ASP.NET AJAX.*** The .NET platform enables you to create robust, scalable web-based applications. Microsoft's .NET server-side technology, ASP.NET 4, allows you 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. You can create web pages visually, by dragging and dropping web controls onto web forms. Chapters 19 and 27 introduce these powerful technologies. We present a sequence of examples in which you build several web applications, including a web-based bookstore. Chapter 27 culminates with an example that demonstrates the power of AJAX. We also discuss 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 4 and LINQ to build a guestbook application that retrieves information from a database and displays it in a web page. We use a `LinqDataSource` from a web application to manip-

ulate 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 29, 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 you 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.
- *Language Features to Support LINQ.* Many of the 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 use delegates (objects that hold method references) to support our discussion of 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 see 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 omit the variable's type—the compiler infers it from the type of the initializer value (introduced in Chapter 9). This is another feature used frequently in LINQ.
- *Object and Collection Initializers.* When creating an object, you can use the object initializer syntax (introduced in Chapter 9) to assign values to the new object's properties. Similarly, you can use the collection initializer syntax (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# provides automatically implemented properties (also known as auto-implemented properties; introduced in Chapter 4). With an auto-implemented property, the 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.
- **Other New Language Features.** We cover optional parameters, named parameters, covariance and contravariance.
- **Visual C# 2010 Express IDE.** All screenshots have been updated to the Visual C# 2010 Express IDE.
- **Contextual keywords.** The keywords table (Chapter 3) includes the 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# 2010. Many of these contextual keywords are used with LINQ.

- ***IntelliSense.*** We point out additional ways in which the IDE's *IntelliSense* helps you write code.
- ***Data Tips and Visualizers.*** We use *DataTips* and visualizers to view object contents in the code window during debugging.
- ***Tuned Treatment of Object-Oriented Programming.*** The book 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 presents the crucial concepts of polymorphism, abstract classes, concrete classes and interfaces, all of which facilitate powerful manipulations among objects in an inheritance hierarchy.
- ***Visual Studio 2010 Debugger.*** In Appendix G, we explain how to use key debugger features, such as setting "breakpoints" 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.

Case Studies

Among the hundreds of complete working C# programs we present are many case studies, including:

- `GradeBook` class in Chapters 4–8.
- OOD/UML ATM system in Chapters 30 and 31.
- `Time` class in Chapter 10.
- `Employee` payroll application in Chapters 11–12.
- WPF painter application in Chapter 24.
- WPF text-editor application in Chapter 24.
- WPF color-chooser application in Chapter 24.
- WPF book cover viewer application in Chapter 24.
- WPF television application in Chapter 25.
- Address-book application in Chapter 18.
- Guestbook ASP.NET application in Chapter 19.
- Password-protected books database ASP.NET application in Chapter 27.
- Airline reservation web service in Chapter 28.
- Blackjack web service in Chapter 28.
- Equation-generator web service and math-tutor application in Chapter 28.
- Silverlight weather-viewer application in Chapter 29.
- Silverlight Flickr® photo-viewer application in Chapter 29.
- Silverlight Deep Zoom book-cover collage application in Chapter 29.
- Silverlight video-viewer application in Chapter 29.

Object-Oriented Design Case Study: Designing and Implementing an ATM

In this case study, we design and fully implement the software for a simple automated teller machine (ATM). After completing this case study, you'll be familiar with an object-oriented design and implementation for a significant C# application.

The design was developed at Deitel & Associates, Inc., and reviewed by industry professionals and academics. We kept the design and the code small and simple so that they would work well in C# professional courses.

The Unified Modeling Language® (UML®) has become the preferred graphical modeling language for designing object-oriented systems. Chapters 30 and 31 present a carefully paced introduction to object-oriented design using the UML.

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.

Here's what the sections of the case study cover:

Section 1.9—Introduction to Object Technology—presents basic concepts and terminology of object technology, including classes, objects, encapsulation and inheritance.

Section 30.2—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 facilitates the design process in subsequent Case Study sections by providing several additional types of diagrams to model our system. 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—called *use cases*. We model these interactions, using UML *use case diagrams*.

Section 30.3—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 system. The diagram also describes relationships, known as *associations*, among the classes.

Section 30.4—Identifying Class Attributes—focuses on the attributes of the classes discussed in Section 30.3. 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 30.3.

Section 30.5—Identifying Objects' States and Activities—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 30.6—Identifying Class Operations—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 30.3 to include each operation with its associated class. As future chapters introduce such topics as inheritance, we'll modify our classes and diagrams.

Section 30.7—Identifying Collaboration Among Objects—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 system by discussing *collaborations*—messages that objects send to each other to communicate. The class operations that we identified in Section 30.6 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 31.2—Starting to Program the Classes of the ATM System—takes a break from designing the behavior of our system. We begin the implementation process. Using the UML class diagram of Section 30.3 and the attributes and operations discussed in Section 30.4 and Section 30.6, 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 31.3—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 30.3 by using inheritance to group classes with similar characteristics. This section concludes the design of the model portion of our simulation.

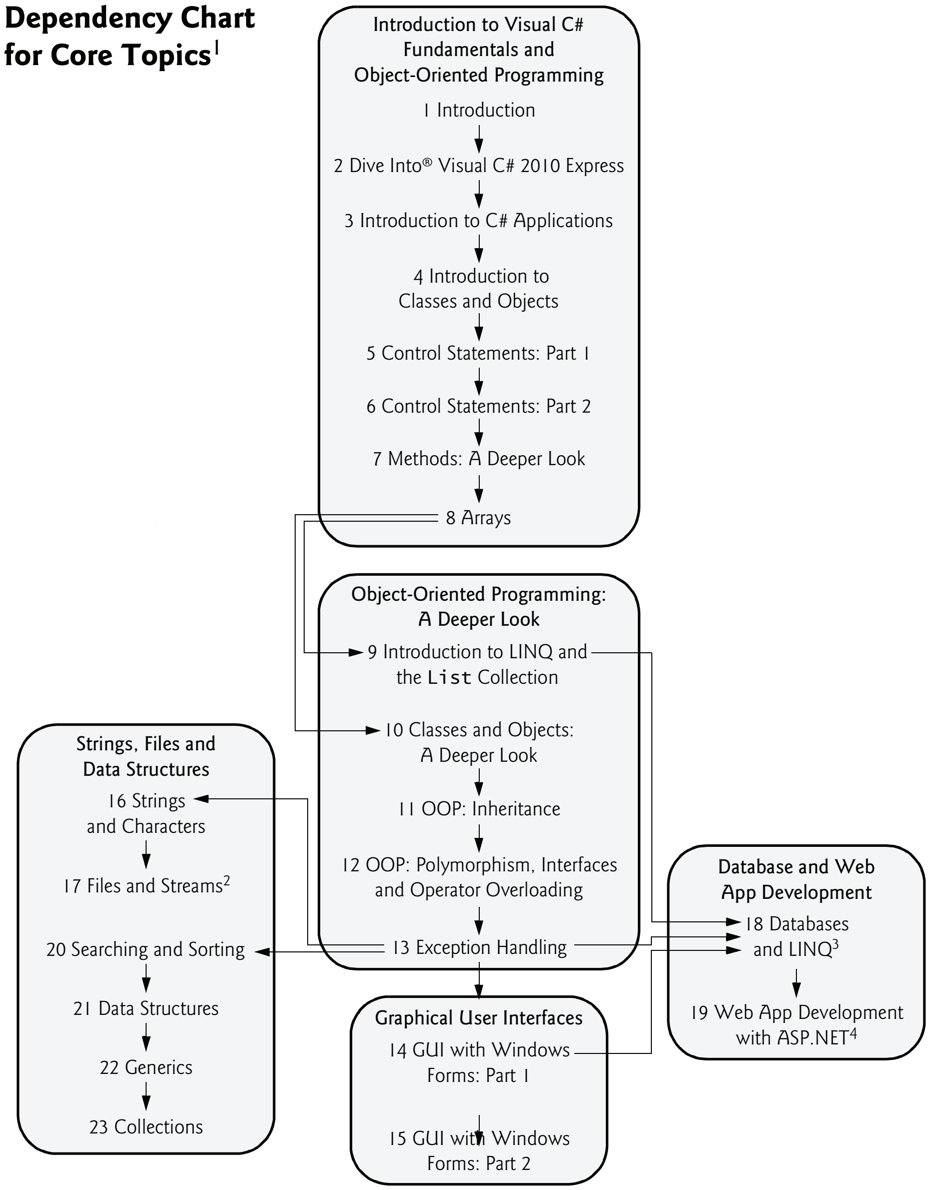
Section 31.4—ATM Case Study Implementation—The majority of the case study involves designing the model (i.e., the data and logic) of the ATM system. In this section, we fully implement that model in C#, working from the UML diagrams we created. We apply the concepts of object-oriented design with the UML and object-oriented programming in C# that you learned in the chapters. By the end of this case study, you'll have completed the design and implementation of a real-world system and should feel confident tackling larger systems.

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

Dependency Charts

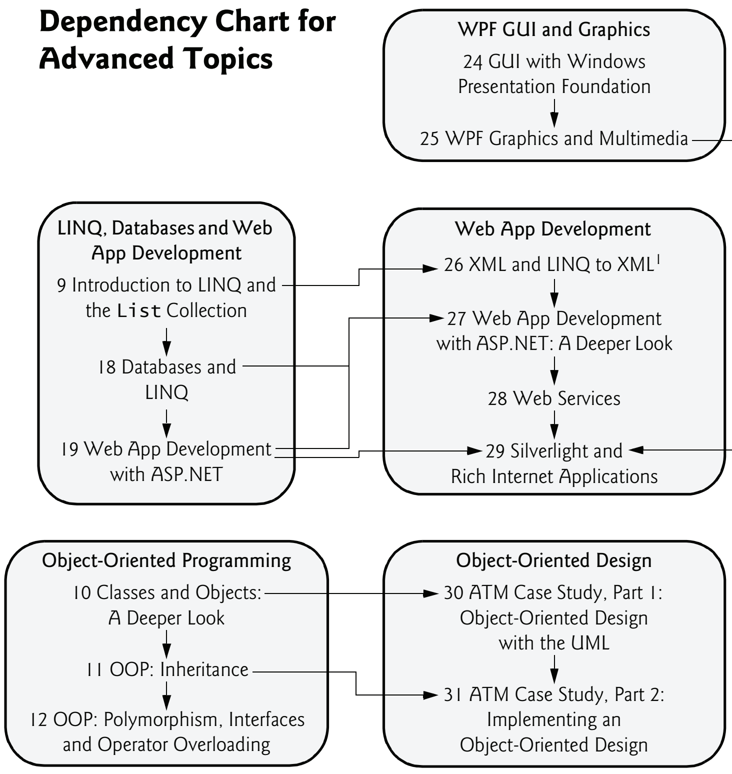
Figures 1–2 illustrate the dependencies among chapters in the book. An arrow pointing into a chapter indicates that it *depends on* the content of the chapter from which the arrow points. We've commented on some additional dependencies in the diagrams' footnotes.

Dependency Chart for Core Topics[1]


Introduction to Visual C# Fundamentals and Object-Oriented Programming
1 Introduction
2 Dive Into® Visual C# 2010 Express
3 Introduction to C# Applications
4 Introduction to Classes and Objects
5 Control Statements: Part 1
6 Control Statements: Part 2
7 Methods: A Deeper Look
8 Arrays
Object-Oriented Programming: A Deeper Look
9 Introduction to LINQ and the List Collection
10 Classes and Objects: A Deeper Look
11 OOP: Inheritance
12 OOP: Polymorphism, Interfaces and Operator Overloading
13 Exception Handling
Strings, Files and Data Structures
16 Strings and Characters
17 Files and Streams2
20 Searching and Sorting
21 Data Structures
22 Generics
23 Collections
Database and Web App Development
18 Databases and LINQ3
19 Web App Development with ASP.NET4
Graphical User Interfaces
14 GUI with Windows Forms: Part 1
15 GUI with Windows Forms: Part 2

1. See Fig. 2 for the advanced topics chapters.
2. Requires Sections 14.1–14.5.
3. Requires Sections 14.1–14.6 and 15.8.
4. Requires general GUI and event-handling knowledge (Sections 14.1–14.3).

Fig. 1 | Chapter dependency chart for the core-topic chapters.



Dependency Chart for Advanced Topics
WPF GUI and Graphics
24 GUI with Windows Presentation Foundation
25 WPF Graphics and Multimedia
LINQ, Databases and Web App Development
9 Introduction to LINQ and the List Collection
18 Databases and LINQ
19 Web App Development with ASP.NET
Web App Development
26 XML and LINQ to XML[1]
27 Web App Development with ASP.NET: A Deeper Look
28 Web Services
29 Silverlight and Rich Internet Applications
Object-Oriented Programming
10 Classes and Objects: A Deeper Look
11 OOP: Inheritance
12 OOP: Polymorphism, Interfaces and Operator Overloading
Object-Oriented Design
30 ATM Case Study, Part 1: Object-Oriented Design with the UML
31 ATM Case Study, Part 2: Implementing an Object-Oriented Design

1. Chapter 26 depends on the introduction to XML in Chapter 24.

Fig. 2 | Chapter dependency chart for the advanced-topic chapters.

Presentation Features

C# 2010 for Programmers, 4/e, contains a rich collection of examples. We concentrate on effective software engineering principles and stress program clarity in the context of hundreds of complete, working programs.

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

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

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

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

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

Error-Prevention Tip
These tips contain suggestions for exposing and removing bugs from your programs; many of the tips describe aspects of Visual C# that prevent bugs from getting into programs.

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

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

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

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

Using Fonts for Emphasis. We place the key terms and the index's page reference for each defining occurrence in **bold** text for easier reference. On-screen components are emphasized 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 *C# 2010 for Programmers, 4/e* are available for download from:

```
www.deitel.com/books/csharpfp4/
```

Site registration is quick and easy. Download all the examples, then run each program as you read the corresponding text discussions.

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've met the objectives.

Quotations. The learning objectives are accompanied by quotations. Some are humorous; some are philosophical; others offer interesting insights.

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

csfp4_preface.fm Page xxx Sunday, October 10, 2010 8:54 AM

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 OOD/UML 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.

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

Software for the Book

We use Microsoft Visual Studio 2010 development tools, including the free Visual C#® 2010 Express Edition, Visual Web Developer 2010 Express Edition and SQL Server 2008 Express Edition. The Express Editions provide rich functionality and can be used to build robust .NET applications. They are appropriate for professionals who do not have access to a complete version of Visual Studio 2010.

You may use the Express Editions to compile and execute *all* the example programs in the book. You may also use the full Visual Studio product to build and run the examples. All of the features supported by the Express Editions are also available in the complete Visual Studio 2010 editions.

You can download the latest versions of the Express Edition tools from:

```
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 Express. Microsoft provides a dedicated forum for help using the Express Editions at:

```
social.msdn.microsoft.com/forums/en-US/Vsexpressinstall/threads/
```

Windows 7, Windows Vista and Windows XP

You can use Windows 7, Windows Vista or Windows XP. We used Windows 7 while developing the book. We use the 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 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/csharpfp4/` with appropriate instructions. Write to us at `deitel@deitel.com` if you encounter any problems, and we'll respond promptly.

Other Software Requirements

For Chapters 18, 19, 27 and 28 you'll need the SQL Server 2008 Express Edition. Chapters 19, 27 and 28 require Visual Web Developer 2010 Express (or a full Visual Studio 2008 edition). For updates on the software used in this book, subscribe to our free e-mail newsletter at `www.deitel.com/newsletter/subscribe.html`, visit the book's website at `www.deitel.com/books/csharpfp4/`, and follow us on Twitter (`@deitel`) and Facebook (`www.deitel.com/deitelfan`).

C# 2010 Fundamentals: Parts I, II and III LiveLessons Video Product

Our *C# 2010 Fundamentals: Parts I, II and III* LiveLessons Camtasia-based video training product shows you what you need to know to start building robust, powerful software with C# 2010 and .NET 4. It includes 20+ hours of expert training synchronized to *C# 2010 for Programmers, 4/e*.

Check out our growing list of LiveLessons video products:

- *C# 2010 Fundamentals I, II, and III*
- *C# 2008 Fundamentals I and II*
- *Java Fundamentals I and II*
- *C++ Fundamentals I and II*
- *iPhone App-Development Fundamentals I and II*
- *JavaScript Fundamentals I and II*
- *Visual Basic 2010 Fundamentals I and II*
- *C Fundamentals I and II*
- *Android Fundamentals I and II*

For additional information about Deitel LiveLessons video products, visit:

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

Licensing Deitel Book and/or LiveLessons Video Content for *Your* Corporate Learning Management Systems

Corporations and organizations may purchase licenses for Deitel's best-selling book and LiveLessons video content to be placed on internal learning management systems. For more information, e-mail `deitel@deitel.com`.

The Deitel Online Resource Centers

We provide 100+ online Resource Centers on various topics of interest to our readers—see the list at `www.deitel.com/ResourceCenters.html`. We've found many exceptional resources online, including tutorials, documentation, software downloads, articles, blogs, podcasts, videos, code samples, books, e-books and more—most are free. Some of the Resource Centers you might find helpful while studying this book are Visual C#, ASP.NET, ASP.NET AJAX, LINQ, .NET, Silverlight, SQL Server, Web Services, Windows Communication Foundation, Windows Presentation Foundation, Windows 7, UML, Code Search Engines and Code Sites, Game Programming and Programming Projects.

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. Thanks especially to Abbey Deitel and Barbara Deitel.

We're fortunate to have worked on this project with the dedicated publishing professionals at Prentice Hall/Pearson. We appreciate the extraordinary efforts and 15-year mentorship of our friend and professional colleague Mark L. Taub, Editor-in-Chief of Pearson

Technology Group. Thanks to Sandra Schroeder and Chuti Prasertsith for their work on the cover, and to John Fuller for managing the production of the book.

We wish to acknowledge the efforts of our third and fourth edition reviewers. Adhering to tight schedules, they scrutinized the text and the programs and provided countless suggestions for improving the presentation:

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).

Other 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), Peter Bromberg (Senior Architect Merrill Lynch and C# MVP), Adam Calderon (C# MVP, InterKnowlogy), Stochio Goutsev (Independent Consultant, writer and developer and C# MVP), Octavio Hernandez (C# MVP, Advanced Bionics), Ged Mead (DevCity.Net, Microsoft VB MVP—Visual Developer) and José Antonio González Seco (Parliament of Andalusia).

Academic Reviewers
Mingsheng Hong (Cornell University), Stan Kurkovsky, Ph.D. (Central Connecticut State University), Markus Lumpe (Swinburne University of Technology), Gavin Osborne (Saskatchewan Institute of Applied Science and Technology) and Zijiang Yang (Western Michigan University).

Well, there you have it! Visual C# 2010 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 you build business-critical and mission-critical information systems. As you read the book, we'd 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/csharpfp4/
```

We hope you enjoy reading *C# 2010 for Programmers, 4/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., is a graduate of MIT, where he studied Information Technology. Through Deitel & Associates, Inc., he has delivered C#, Visual Basic, Java, C++, C and Internet programming courses to industry clients, including Cisco, IBM, Sun Microsystems, Dell, Siemens, Lucent Technologies, Fidelity, NASA at the Kennedy Space Center, the National Severe Storm Laboratory, White Sands Missile Range, Rogue Wave Software, Boeing, SunGard Higher Education, Stratus, Cambridge Technology Partners, One Wave, Hyperion Software, Adra Systems, Entergy, CableData Systems, 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 authors.

Dr. Harvey M. Deitel, Chairman and Chief Strategy Officer of Deitel & Associates, Inc., has 49 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 industry and academic 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. He and Paul are the co-authors of dozens of books and multimedia packages and they are writing many more. With translations published in Japanese, German, Russian, Chinese, Spanish, Koresan, French, Polish, Italian, Portuguese, Greek, Urdu and Turkish, the Deitels' texts have earned international recognition. 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., founded by Paul Deitel and Harvey Deitel, is an internationally recognized authoring, corporate training and software development organization specializing in computer programming languages, object technology, Android and iPhone app development, and Internet and web software technology. The company offers instructor-led training courses delivered at client sites worldwide on major programming languages and platforms, such as Visual C#®, Java™, C, C++, Visual Basic®, Objective-C and iPhone and iPad app development, Android app development, XML®, Python®, object technology, Internet and web programming, and a growing list of additional programming and software development courses. The company's clients include many of the world's largest companies, government agencies, branches of the military, and academic institutions.

Through its 34-year publishing partnership with Prentice Hall/Pearson, Deitel & Associates, Inc., publishes leading-edge programming professional books, college textbooks, and *LiveLessons* DVD- and web-based 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 on-site, instructor-led training at your company or organization, e-mail `deitel@deitel.com`.

Individuals wishing to purchase Deitel books and *LiveLessons* DVD training courses 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.pearsoned.com/professional/index.htm`.