Beginning C# 2008 Objects

From Concept to Code

Grant Palmer and Jacquie Barker

Apress®

Beginning C# 2008 Objects: From Concept to Code

ISBN-13 (pbk): 978-1-4302-1088-7

ISBN-13 (electronic): 978-1-4302-1087-0

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Lead Editors: Joohn Choe, Dominic Shakeshaft
Technical Reviewer: Andy Olsen
Editorial Board: Clay Andres, Steve Anglin, Mark Beckner, Ewan Buckingham, Tony Campbell, Gary Cornell, Jonathan Gennick, Michelle Lowman, Matthew Moodie, Jeffrey Pepper, Frank Pohlmann, Ben Renow-Clarke, Dominic Shakeshaft, Matt Wade, Tom Welsh
Project Manager: Beth Christmas
Copy Editor: Nancy Sixsmith
Associate Production Director: Kari Brooks-Copony
Production Editor: Elizabeth Berry
Compositor: Diana Van Winkle
Proofreader: Linda Seifert
Indexer: John Collin
Artist: April Milne
Cover Designer: Kurt Krames
Manufacturing Director: Tom Debolski

Distributed to the book trade worldwide by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax 201-348-4505, e-mail orders-ny@springer-sbm.com, or visit http://www.springeronline.com.

For information on translations, please contact Apress directly at 2855 Telegraph Avenue, Suite 600, Berkeley, CA 94705. Phone 510-549-5930, fax 510-549-5939, e-mail info@apress.com, or visit http://www.apress.com.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales–eBook Licensing web page at http://www.apress.com/info/bulksales.

The source code for this book is available to readers at http://www.apress.com.

In loving memory of my wonderful parents, Bill and Dorothy Jost
—Jacquie Barker

To Lisa, Zachary, and Jackson, oh my!
—Grant Palmer

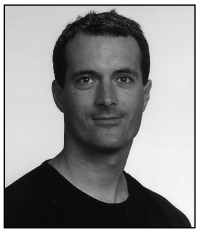
Contents at a Glance

PART THREE ■■■ Translating a UML "Blueprint" into C# Code

Contents

PART THREE Translating a UML "Blueprint" into C# Code

About the Authors



GRANT PALMER has worked in the Space Technology Division at NASA Ames Research Center in Moffett Field, CA for the past 23 years. Grant was a NASA engineer for 15 years and currently works as a scientific programmer with the ELORET Corporation, developing computer applications that help design the thermal protection systems of spacecraft reentering the Earth's atmosphere.

Grant earned a Bachelor of Science degree in mechanical engineering from the University of California, Berkeley. He later received a Master of Science degree in aerospace engineering from Stanford University. Grant is an expert in FORTRAN, C, C++, and Perl, but now does most of his programming in the more modern languages of Java and C#. He has authored or coauthored seven books on computer programming, including *Physics for Game Programmers* (Apress) and *C# Programmer's Reference* (Wrox).

Grant lives in Bothell, Washington, with his wife, Lisa; his two sons, Jackson and Zachary; and various members of the animal kingdom.

JACQUIE BARKER is a professional software engineer, author, and adjunct faculty member at George Mason University (GMU) in Fairfax, Virginia and The George Washington University (GWU) in Washington, D.C. With more than 25 years of experience as a hands-on software engineer and project manager, Jacquie has spent the past 12 years focusing on object technology, becoming proficient as an object modeler and Sun Microsystems–certified Java developer. She is currently employed as a senior member of the technical staff at Technology Associates, Inc. in Herndon, Virginia, and is also the founder of ObjectStart LLC, an object technology mentorship and training firm.

Jacquie earned a Bachelor of Science degree in computer engineering with highest honors from Case Western Reserve University in Cleveland, Ohio. She later received a Master of Science degree in computer science from UCLA, focusing on software systems engineering, and has subsequently pursued post-graduate studies in information technology at GMU.

The first edition of *Beginning C# Objects* was adapted from Jacquie's bestselling book, *Beginning Java Objects: From Concepts to Code*, published originally by the former Wrox Press, Ltd. and now by Apress. Jacquie's winning formula for teaching object fundamentals continues to receive praise from readers around the world, and *Beginning Java Objects: From Concepts to Code* has been adopted by many universities as a key textbook in their core IT curricula. Her latest book, *Taming the Technology Tidal Wave: Practical Career Advice for Technical Professionals*, is now available through ObjectStart Press.

Please visit Jacquie's web sites, `http://objectstart.com` and `http://techtidalwave.com`, for more information on her various publications and service offerings.

On a personal note, Jacquie's passions include her husband, Steve; their four pet cats, Kwiddie, Tiffie, Walter, and Wynxie; and her work as an animal rescue volunteer (please visit `http://petsbringjoy.org`). When not engaged in computer-related pursuits, Jacquie and Steve enjoy motorcycle road trips through the Virginia countryside on their Gold Wing, tandem bicycling, and spending quality time with family and friends.

About the Technical Reviewer

ANDY OLSEN is a freelance developer and consultant based in the UK. He has been working with .NET since the Beta 1 days and has authored and reviewed several books for Apress, covering C#, Visual Basic, ASP.NET, and other topics. Andy is a keen football and rugby fan, and enjoys running and skiing (badly). He lives by the seaside in Swansea with his wife, Jayne, and children, Emily and Thomas, who have just discovered the thrills of surfing and look much cooler than he ever will!

Acknowledgments

We'd like to offer sincere, heartfelt thanks to everyone who helped us produce this book:

- To Andy Olsen, who served as our primary technical reviewer. He's like an all-knowing kung fu master, except he knows everything about .NET instead of kung fu. Thanks also to Damien Foggon who gave the book an extra set of expert eyes during the reviews of the chapters in Part 3 of the book.
- To Dominic Shakeshaft, our editor, for his dedication to ensuring the clarity of our book's message.
- To Gary Cornell, Apress publisher, for suggesting a Java-to-C# "port" of *Beginning Java Objects*.
- To *all* the folks at Apress—especially Beth Christmas, Joohn Choe, Nancy Sixsmith, Laura Cheu, and Elizabeth Berry—for their superb editorial/production/marketing support.
- To our spouses, children, and assorted other family members and friends, for once again being patient as we became temporarily consumed with the "writing biz."

Grant Palmer and Jacquie Barker

Preface

As a Java developer and instructor, Jacquie Barker wrote her first book, *Beginning Java Objects,* to communicate her passionate belief that learning objects thoroughly is an essential first step in mastering an object-oriented programming language (OOPL). Since *Beginning Java Objects* was first published in November 2000, we've heard from countless readers who agree wholeheartedly!

We were therefore delighted when Gary Cornell, the publisher of Apress, and Dominic Shakeshaft, Apress editorial director, approached us about producing a C# version of *Beginning Java Objects*. It's indeed true that basic object concepts are "language neutral." What you'll learn conceptually about objects in Part One of this book, and about object modeling in Part Two, could apply equally well to any OOPL.

But our goal for this book is twofold: not only do we want to teach you about objects and object modeling, but we also want to get you properly jump-started with the C# programming language by showing you how such concepts translate into C# syntax specifically. Hence, *Beginning C# Objects* was born!

The first edition of *Beginning C# Objects* came out in the spring of 2004. Since that time, C# and the .NET platform have grown by leaps and bounds. Several new releases of the .NET Framework have come out, and many exciting new features have been added to the C# programming language. This second edition of the book captures the "latest and greatest" of .NET and C# for beginning programmers and should get you well on your way to becoming an expert C# programmer.

Grant Palmer and Jacquie Barker

Introduction

First and foremost, *Beginning C# 2008 Objects: From Concept to Code* is a book about software objects: what they are, why they are so "magical" and yet so straightforward, and how one goes about structuring a software application to use objects appropriately.

This is also a book about C#: not a hard-core, "everything-there-is-to-know-about-C#" book; it's a gentle, yet comprehensive, introduction to the language, with special emphasis on how to transition from an object model to a fully functional C# application (which few, if any, other books provide).

Goals for this Book

Our goals in writing this book (and, hopefully, yours for buying it) are the following:

- Make you comfortable with fundamental object-oriented (OO) terminology and concepts
- Give you hands-on, practical experience with object modeling; that is, with developing a "blueprint" that can be used as the basis for subsequently building an OO software system
- Illustrate the basics of how such an object model is translated into a working software application—a C# application, to be specific, although the techniques that you'll learn for object modeling apply equally well to any object-oriented programming language (OOPL)

If you're already experienced with the C# language (but not with object fundamentals), this book will provide you with critical knowledge about the language's OO roots. On the other hand, if you're a newcomer to C#, this book will get you properly "jump-started." Either way, this book is a "must-read" for anyone who wants to become proficient with an OOPL like C#.

Just as importantly, this book is *not* meant to do the following:

- *Turn you into an overnight "pro" in object modeling*: Like all advanced skills, becoming totally comfortable with object modeling takes two things: a good theoretical foundation and a lot of practice. We give you the foundation in this book, along with suggestions for projects and exercises that will enable you to apply and practice your newfound knowledge. But the only way you'll really get to be proficient with object modeling is by participating in OO modeling and development projects over time. This book will give you the skills—and hopefully the confidence—to begin to apply object techniques in a professional setting, which is where your real learning will take place, particularly if you have an OO-experienced mentor to guide you through your first "industrial-strength" project.

- *Make you an expert in any particular OO methodology*: There are dozens of different formal methods for OO software development; new variations continue to emerge, and no one methodology is necessarily better than another. For example, UML (Unified Modeling Language) notation is one of the newest, and OMT (Object Modeling Technique) notation is one of the oldest, yet the two are remarkably similar because UML is based to a great extent on OMT. By making sure that you understand the generic *process* of object modeling along with the specifics of the UML, you'll be armed with the knowledge you need to read about, evaluate, and select a specific methodology—or craft your own. (Who knows? Maybe someday you'll even write a book yourself on the methodology that you invent!)
- *Teach you everything you'll ever need to know about C#*: C# is a very rich language, consisting of dozens of core classes, hundreds of classes available from the Framework Class Library, and literally thousands of operations that can be performed with and by these classes. If C# provides a dozen alternative ways to do something in particular, we'll explain the one or two ways that we feel best suit the problem at hand, to give you an appreciation for how things are done. Nonetheless, you'll definitely see enough of the C# language in this book to be able to build a complete application.

Armed with the foundation you gain from this book, you'll be poised and ready to appreciate a more thorough treatment of C# such as that offered by one of the many other C# references that are presently on the market or an in-depth UML reference.

Why Is Understanding Objects So Critical to Being a Successful OO Programmer?

Time and again, we meet software developers—at our places of employment, at clients' offices, at professional conferences, on college campuses—who have attempted to master an OOPL like C# by taking a course in C#, reading a book about C#, or installing and using a C# integrated development environment (IDE) such as Visual Studio .NET. However, there is something fundamentally missing: a basic understanding of what objects are all about and, more importantly, knowledge of how to structure a software application from the ground up to make the most of objects.

Imagine that you know the basics of home construction and you're asked to build a house. In fact, you're a world-renowned home builder whose services are in high demand! Your client tells you that all the materials you'll need for building this home will be delivered to you. On the day construction is to begin, a truck pulls up at the building site and unloads a large pile of strange blue, star-shaped blocks with holes in the middle. You're totally baffled! You've built countless homes using materials like lumber, brick, and stone, and you know how to approach a building project using these familiar materials; but you haven't got a clue about how to assemble a house using blue stars.

Scratching your head, you pull out a hammer and some nails and try to nail the blue stars together as if you were working with lumber, but the stars don't fit together very well. You then try to fill in the gaps with the same mortar that you would use to make bricks adhere to one another, but the mortar doesn't stick to these blue stars very well. Because you're working

under tight cost and schedule constraints (and because you're too embarrassed to admit that you, as an "expert" builder, don't know how to work with these modern materials), you press on. Eventually, you wind up with something that looks (on the outside, at least) like a house.

Your client comes to inspect the work and is terribly disappointed. One of the reasons he had selected blue stars as a construction material was that they are extremely energy efficient, but because you have used nails and mortar to assemble the stars, they have lost a great deal of their inherent ability to insulate the home. To compensate, your client asks you to replace all the windows in the home with thermal glass windows so that they will allow less heat to escape. You're panicking at this point! Swapping out the windows will take as long, if not longer, than it has taken to build the house in the first place, not to mention the cost of replacing stars that will be damaged in the renovation process. When you tell your customer this, he goes ballistic! Another reason why he selected blue stars as the construction material was because of their recognized flexibility and ease of accommodating design changes, but because of the ineffective way in which you assembled these stars, you'll have to literally rip them apart and replace a great many of them.

Sad to say, this is the way many programmers wind up building an OO application when they don't have appropriate training in how to approach the project from the perspective of objects. Worse yet, the vast majority of would-be OO programmers are blissfully ignorant of the need to understand objects in order to program in an OO language. So they take off programming with a language such as C# and wind up with a far-from-ideal result: a program that lacks flexibility when an inevitable "mid-course correction" is required (for example, when new functionality needs to be introduced after an application has been deployed).

Who Is This Book Written For?

Anyone who wants to get the most out of an OOPL such as C#! It also was written for the following people:

- Anyone who has yet to tackle C#, but wants to get off on the right foot with the language
- Anyone who has ever purchased a book on C# and read it faithfully; who understands the "bits and bytes" of the language, but doesn't quite know how to structure an application to best take advantage of the OO features of the language
- Anyone who has purchased a C# IDE software tool, but really only knows how to drag and drop graphical user interface (GUI) components and add a little bit of logic behind buttons, menus, and so on without any real sense of how to properly structure the core of the application around objects
- Anyone who has built a C# application, but was disappointed with how difficult it was to maintain or modify it when new requirements were presented later in the application's life cycle
- Anyone who has previously learned something about object modeling, but is "fuzzy" on how to transition from an object model to real live code (C# or otherwise)

The bottom line is that anyone who really wants to master an OO language such as C# *must* become an expert in objects first!

To gain the most value from this book, you should have some programming experience under your belt; virtually any language will do. You should understand simple programming concepts such as the following:

- Simple data types (integer, floating point, and so on)
- Variables and their scope (including the notion of global data)
- Control flow (`if-then-else` statements, `for`/`do`/`while` loops, and so on)
- Arrays: what they are and how to use them
- The notion of a function/subroutine/method: how to pass data in and get results back out

You don't need any prior exposure to C# (we'll give you a taste of the language at the beginning of Part One and will go into the language in depth in Part Three). And you needn't have ever been exposed to objects, either—in the software sense, at least! As you'll learn in Chapter 2, human beings naturally view the entire world from the perspective of objects.

Even if you've already developed a full-fledged C# application, it's certainly not too late to read this book if you still feel "unclear" about the object aspects of structuring an application. Universities often offer separate courses in object modeling and C# programming. Although they would ideally take both courses in sequence, students often arrive at an object modeling course having already taken a stab at learning C#. Even for folks who will see some familiar landmarks (in the form of C# code examples) in this book, many new insights will be gained as they learn the rationale for why we do many of the things that we do when programming in C# (or any other OOPL, for that matter).

It ultimately makes someone a better C# programmer to know the "whys" of object orientation instead of just the mechanics of the language. If you have had prior experience with C#, you may find that you can quickly skim those chapters that provide an introduction to the language—namely, Chapter 1 in Part One and Chapter 13 in Part Three.

Because this book has its roots in courses that the authors teach, it's ideally suited for use as a textbook for a semester-long graduate or upper-division undergraduate course in either object modeling or C# programming.

What If You're Interested in Object Modeling, but Not Necessarily in C# Programming?

Will this book still be of value to you? Definitely! Even if you don't plan on making a career of programming (as is true of many of our object modeling students), we've found that being exposed to a smattering of code examples written in an OOL such as C# really helps to cement object concepts. So, you're encouraged to read Part Three—at least through Chapter 14—even if you never intend to set your hands to the keyboard for purposes of C# programming.

How This Book Is Organized

The book is structured around three major topics, as follows:

Part One: The ABCs of Objects

Before we dive into the how-to's of object modeling and the details of OO programming in C#, it's important that we all speak the same language with respect to objects. Part One, consisting of Chapters 1–7, starts out slowly by defining basic concepts that underlie all software development approaches, OO or otherwise. But the chapters quickly ramp up to a discussion of advanced object concepts so that, by the end of Part One, you should be "object-savvy."

Part Two: Object Modeling 101

In Part Two—Chapters 8–12 of the book—we focus on the underlying principles of how and (more importantly) why we do the things we do when we develop an object model of an application—principles that are common to all object modeling techniques. It's important to be conversant in UML notation because it is an industry standard and is most likely what the majority of your colleagues/clients will be using. So we'll teach you the basics of the UML and use the UML for all our concrete modeling examples. Using the modeling techniques presented in these chapters, we'll develop an object model "blueprint" for a Student Registration System (SRS), the requirements specification for which is presented at the end of this introduction.

Part Three: Translating an Object "Blueprint" into C# Code

In Part Three of the book—Chapters 13–17—we illustrate how to render the SRS object model that we've developed in Part Two into a fully functioning C# application, complete with a GUI and a way to persist data from one user logon to the next. All the code examples that we present in this section are available for download from the Apress web site (`www.apress.com`), and we strongly encourage you to download and experiment with this code. In fact, we provide exercises at the end of each chapter that encourage such experimentation. The requirements specification for the SRS is written in the narrative style with which software system requirements are often expressed. You might feel confident that you could build an application today to solve this problem, but by the end of this book you should feel much more confident in your ability to build it as an OO application.

To round out the book, we've included a final chapter, "Next Steps," which provides suggestions for how you might wish to continue your OO discovery process after finishing this book. We furnish you with a list of recommended books that will take you to the next level of proficiency, depending on what your intention is for applying what you've learned in this book.

Conventions

To help you get the most from the text and keep track of what's happening, we've used a number of conventions throughout the book.

For instance:

Note Note and Tip boxes reflect important background information and advice.

As for styles in the text:

- When we introduce important words, we *italicize* them.
- Terms that you should type in or significant new additions to code samples are shown in **boldface**.
- File names, URLs, and code within the text are in a `special code font`.

Example code is shown as follows:

```
// Bolding is used to call attention to new or significant code:
Student s = new Student();
// whereas unbolded code is code that's less important in the
// present context, or perhaps has been seen before.
int x = 3;
```

Which Version of C# Is This Book Based On?

As with any programming language, from time to time new versions of C# will be released by Microsoft. At the time the second edition of this book was published (fall 2008), the latest version of .NET was 3.5, and we do use some of the "latest and greatest" C# language features in this book such as auto-implemented properties and object initializers. However, the insights into proper OO programming that you will gain after reading this book aren't tied to any version of .NET and will serve you equally well when new versions of C# appear.

A Final Thought Before We Get Started

A lot of the material in this book—particularly at the beginning of Part One—might seem overly simplistic to experienced programmers. This is because much of object technology is founded on basic software engineering principles that have been in practice for many years and—in many cases—just repackaged slightly differently. There are indeed a few new tricks that make OOPLs extremely powerful and virtually impossible to achieve with non-OO languages—inheritance and polymorphism, for example, which you'll learn more about in Chapters 5 and 7, respectively. (Such techniques can be simulated by hand in a non-OOPL, just as programmers could program their own database management system [DBMS] from scratch instead of using a commercial product such as Oracle, Sybase, or MS SQL Server—but who'd want to?)

The biggest challenge for experienced programmers in becoming proficient with objects is to reorient the way they think about the problem they will be automating:

- Software engineers/programmers who have developed applications using non-OO methods often have to "unlearn" certain approaches used in the traditional methods of software analysis and design.
- Paradoxically, people just starting out as programmers (or as OO modelers) sometimes have an easier time when learning the OO approach to software development as their only approach.

Fortunately, the way we need to think about objects when developing software turns out to be the natural way that people think about the world in general. So, learning to "think" objects—and to program them in C#—is as easy as 1, 2, 3!

Tell Us What You Think

We've worked hard to make this book as useful to you as possible, so we want to know what you think. We're always keen to know what it is you want and need to know.

We appreciate feedback on our efforts and take both criticism and praise to heart in our future editorial efforts. If you have anything to say, please let us know at `info@apress.com` or `www.apress.com`, or contact the authors at `grantepalmer@gmail.com`, `jacquie@objectstart.com`, or `http://objectstart.com`.

STUDENT REGISTRATION SYSTEM (SRS) CASE STUDY: STUDENT REGISTRATION SYSTEM REQUIREMENTS SPECIFICATION

We have been asked to develop an automated Student Registration System (SRS) that will enable students to register online for courses each semester, as well as track a student's progress toward completion of his or her degree.

When a student first enrolls at the university, he or she uses the SRS to set forth a plan of study about which courses he or she plans to take to satisfy a particular degree program, and chooses a faculty advisor. The SRS will verify whether the proposed plan of study satisfies the requirements of the degree the student is seeking. Once a plan of study is established, students are can view the schedule of classes online during the registration period preceding each semester and then choose whichever classes they wish to attend, indicating the preferred section (day of the week and time of day) if the class is offered by more than one professor. The SRS will verify whether the student has satisfied the necessary prerequisites for each requested course by referring to the student's online transcript of courses completed and grades received (the student can review his or her transcript online at any time).

Assuming that (a) the prerequisites for the requested course(s) are satisfied, (b) the course(s) meets one of the student's plan-of-study requirements, and (c) there is room available in each of the class(es), the student is enrolled in the class(es).

If (a) and (b) are satisfied, but (c) is not, the student is placed on a first-come, first-served waiting list. If a class/section that he or she was previously waitlisted for becomes available (either because some other student has dropped the class or because the seating capacity for the class has been increased), the student is automatically enrolled in the waitlisted class, and an email message to that effect is sent to the student. It is his or her responsibility to drop the class if it is no longer desired; otherwise, he or she will be billed for the course.

Students can drop a class up to the end of the first week of the semester in which the class is being taught.