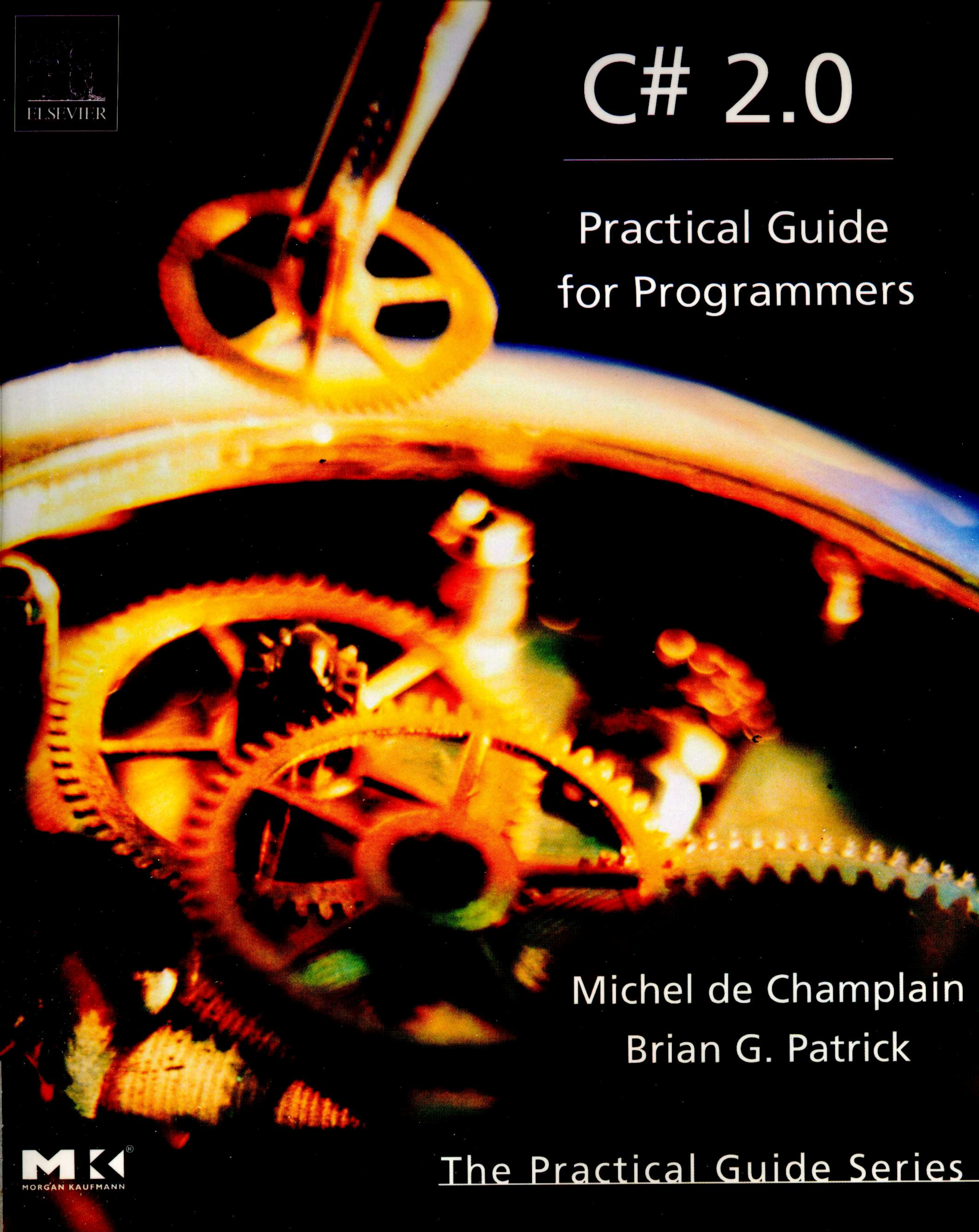
ELSEVIER

C# 2.0

Practical Guide for Programmers



Michel de Champlain
Brian G. Patrick

MORGAN KAUFMANN

The Practical Guide Series

C# 2.0: Practical Guide for Programmers

Michel de Champlain

DeepObjectKnowledge

Brian G. Patrick

Trent University

ELSEVIER

AMSTERDAM • BOSTON • HEIDELBERG
LONDON • NEW YORK • OXFORD
PARIS • SAN DIEGO • SAN FRANCISCO
SINGAPORE • SYDNEY • TOKYO

Morgan Kaufmann is an imprint of Elsevier

MORGAN KAUFMANN PUBLISHERS

Senior Editor	Rick Adams
Associate Editor	Karyn Johnson
Publishing Services Manager	Simon Crump
Project Manager	Brandy Lilly
Cover Design	Yvo Riezebos Design
Cover Image	Photo by Steve Cole, Photodisc Green, Getty Images
Composition	Cepha Imaging Pvt. Ltd.
Copyeditor	Kolam Inc.
Proofreader	Kolam Inc.
Indexer	Kolam Inc.
Interior printer	Maple Press
Cover printer	Phoenix Color

Morgan Kaufmann Publishers is an imprint of Elsevier.
500 Sansome Street, Suite 400, San Francisco, CA 94111

This book is printed on acid-free paper.

Library of Congress Cataloging-in-Publication Data
Application submitted.

ISBN: 0-12-167451-7

For information on all Morgan Kaufmann publications, visit our Web site at *www.mkp.com*

Printed in the United States of America
08 07 06 05 04   5 4 3 2 1

Praise for *C# 2.0: Practical Guide for Programmers*!

Great book for any C# developer! It describes the basic programming language with EBNF notation and provides a number of practical programming tips and best practices on program design that enable you to utilize the C# language features effectively.
– **Adarsh Khare**, Software Design Engineer, Microsoft

C# 2.0: A Practical Guide *provides an amazing breadth of information in a compact and efficient format, with clear and concise writing and useful code examples. It cuts right to the core of what you need to know, covering every aspect of the C# language, an introduction to the .NET API, and an overview of pertinent object-oriented concepts. This book tops my recommendation list for any developer learning C#.*
– **David Makofske**, Principal Consultant/Architect, Akamai Technologies

This book is essential for programmers who are considering system development using C#. The two authors have masterfully created a programming guide that is current, complete, and useful immediately. The writing style is crisp, concise, and engaging. This book is a valuable addition to a C# programmer's library.
– **Edward L. Lamie**, PhD, Director of Educational Services, Express Logic, Inc.

At last, a programming language book that provides complete coverage with a top-down approach and clear, simple examples! Another welcome feature of this book is that it is concise, in the tradition of classics such as Kernighan and Ritchie. The new book by De Champlain and Patrick is the best introduction to C# that I've seen so far.
– **Peter Grogono**, Professor and Associate Chair of Computer Science, Concordia University

The book covers the basic and the advanced features of a relatively new and well established programming language, C#. A truly Object Oriented style is used throughout the book in a consistent manner. C# and Object Oriented concepts are well illustrated through simple and concise examples to hold the reader's attention. A very well-written book.
– **Ferhat Khendek**, PhD, Research Chair in Telecommunications Software Engineering, Concordia University

C# 2.0: Practical Guide for Programmers

To Hélène, the air that I breathe
— Michel

With love to my parents, Lionel and Chrissie
— Brian

Preface

Writing a short book on a comprehensive programming language was most definitely a challenge. But such was our mandate and such is C#.

The C# programming language was first released in 2000 and has quickly established itself as the language *de rigueur* for application development at Microsoft Corporation and other software houses. It is a powerful language based on the paradigm of object-orientation and fully integrated with the Microsoft .NET Framework. Hence, C# is architecturally neutral and supported by a vast library of reusable software.

To describe all minutiae of the C# language or to indulge in all facets of the .NET Framework would require a tome or two. Yet the authors realize that experienced software programmers are not looking to plough through extraneous detail but are focused on extracting the essentials of a language, which allow them to commence development quickly and confidently. That is our primary objective.

To realize this objective, we followed the ABCs of writing: accuracy, brevity, and completeness. First and foremost, care has been taken to ensure that the terminology and the discussion on the syntax and semantics of C# are consistent with the latest language specifications, namely C# 2.0. For easy reference, those features that are new to C# 2.0 are identified in the margins.

Second, for the sake of brevity, we strike at the heart of most features of C# with little digression, historical reflection, or comparative analysis. Although the book is not intended as a tutorial on object-oriented design, a few tips on good programming practice are scattered throughout the text and identified in the margins as well.

Finally, all principal features of the C# programming language are covered, from basic classes to attributes. The numerous examples throughout the text, however, focus on the most natural and most common applications of these features. It is simply not possible within the confines of two hundred pages to examine all permutations of C#.

This practical guide emerged from the experiences of the first author in teaching, training, and mentoring professional developers in industry and graduate students at university on the use of the C# language. Its organization is therefore rooted in several C# jump-start courses and one-day tutorials with an intended audience of experienced programmers. Although some background in object-oriented technology is ideal, all object-oriented features are reviewed in the broader context before they are described with respect to their implementation in C#.

In short, *C# 2.0: Practical Guide for Programmers* rests its hat on three hooks:

- Provide a concise yet comprehensive explanation of the basic, advanced, and latest features of the C# language. Each feature is illustrated with short, uncluttered examples. To ensure that code is error-free, the large majority of examples have been automatically and directly extracted from source code that has been verified and successfully compiled.
- Cover the essentials of the .NET Framework. Modern programming languages like Java and C# are supported by huge application programming interfaces (APIs) or frameworks in order to tackle the flexibility and complexity of today's applications. Although the focus of this book is on the C# language and *not* on the .NET Framework, we would be remiss to omit a basic discussion on the core functionalities of the .NET libraries. Any greater depth, however, would far exceed our mandate.
- Include a refresher on object-oriented concepts. The C# language is fully object-oriented, replete with a unified type system that encapsulates the full spectrum of types, from integers to interfaces. In addition to classes, the concepts of inheritance and polymorphism are given their share of proportional representation as two of the three tenets of object-oriented technology.

Organization of the Book

The book is organized into ten concise chapters and two appendices. Chapter 1 introduces the C# programming language and the .NET Framework. It also outlines a small project that is used as the basis for the exercises at the end of most chapters. This project is designed to gradually meld the features of the C# language into a comprehensive solution for a practical problem.

Unlike in books that present a programming language from the bottom up, Chapters 2, 3, and 4 immediately delve into what we consider the most fundamental, though higher-level, concepts of C#. Chapter 2 begins our discussion with classes and objects, the first of the three tenets of object-oriented technology. We demonstrate how classes are defined as an amalgam of behavior and state, how objects are created, and how access to classes and to class members is controlled. Namespaces are also described as an important aspect of "programming in the large" and how they are used to organize classes into logical groups, to control name conflicts, and to ease the integration and reuse of other classes within applications.

A fuller exposé on the basic class members of C# follows in Chapter 3: methods that define behavior and data members that define state. Constructors, destructors, and parameter passing by value and by reference are also covered. Chapter 3 concludes with an important discussion on class reuse and how classes derive, refine, and redefine their behavior and state via inheritance, the second tenet of object-oriented programming. We compare inheritance with aggregation (composition) and offer a few guidelines on their appropriate use.

The unified type system of C# is presented in Chapter 4, showing how value and reference types are derived from the same root class called `Object`. All value types, including nullable types, are fully described, along with a brief introduction to the basic notion of a reference type. The `Object` class itself provides an excellent vehicle to introduce polymorphism (the third tenet of object-oriented programming), virtual methods, and cloning using deep and shallow copying. The chapter ends with a presentation of two predefined but common classes for arrays and strings.

In Chapters 5 and 6, the rudiments of C# expressions and statements are reviewed with numerous short examples to illustrate their behavior. Expressions are built from arithmetic, logical, relational, and assignment operators and are largely inspired by the lexicon of C/C++. Because selection and iterative statements, too, are drawn from C/C++, our presentation is terse but comprehensive. However, whenever warranted, more time is devoted to those features, such as exceptions and the exception-handling mechanism of C#, that bolster its reliability and robustness.

Chapter 7 extends our discussion on the reference types that were first introduced in Chapter 4. These advanced reference types include delegates, events, abstract classes, and interfaces. New features such as delegate inferences and anonymous methods are also covered. In this chapter, we carefully distinguish between the single inheritance of classes and the multiple implementation of interfaces. Polymorphism, first mentioned with respect to the `Object` root class, is illustrated once again with a comprehensive example based on a hierarchy of counter-classes and interfaces. The two accessors in C#, namely properties and indexers, are also presented, noting the latest specifications for property access modifiers.

The last three chapters (8, 9, and 10) shift their focus away from the programming language concepts of C# and examine some of the basic but indispensable features of the .NET Framework. Chapter 8 extends the notion of class reuse with a look at the different types of predefined collections and their constructors and iterators. Although not associated with the .NET Framework itself, one of the newest features of C# is generic classes (or templates) and is presented as a natural counterpart to collections.

Our discussion on resource disposal begun in Chapter 3 is rounded out in Chapter 9 along with input/output and threads. Input/output is a broad topic and is limited here to representative I/O for binary, bytes, and character streams. Threads, on the other hand, is a challenging topic, and the synchronization mechanisms required to support concurrent programming are carefully explained with several supporting examples. Finally, Chapter 10 examines the use and collection of metadata using reflection and attributes, both pre- and user-defined.

The first of the two appendices summarizes the grammatical rules of the C# language using EBNF notation. The second appendix provides an abridged list of the common XML tags used for the automatic generation of web documentation.

Source Code Availability

The code for most examples and all exercises of each chapter is available and maintained at the website `www.DeepObjectKnowledge.com`.

Acknowledgments

Any book goes through a number of incarnations, but none is more important than that based on the constructive and objective feedback of its reviewers. Much improvement on the organization and technical content of the book is due to their invaluable input, and our sincere thanks are extended to Gerald Baugartner (Ohio State University), Eric Gunnerson (Microsoft Corporation), Keith Hill (Agilent Technologies), Adarsh Khare (Microsoft Corporation), David Makofske (Akamai Technologies), and Mauro Ottaviani (Microsoft Corporation). Over the past year, we have also received timely advice and ongoing encouragement from the kind staff at Morgan Kaufmann and Kolam. We acknowledge their support with a special "tip of the cap" to Rick Adams, Mona Buehler, Karyn Johnson, and Cara Salvatore.

Finally, we warn all potential authors that writing a book is a wonderful way to while away the weeks and weekends. Unfortunately, these precious hours are spent apart from our families, and it is to them that we extend our deepest appreciation for their understanding, patience, and unconditional love.

We hope in the end that you enjoy the book. We hope that it reads well and provides a solid introduction to the C# language. Of course, full responsibility for its organization and content rests with the authors. And with that in mind, we defer to you, our reader, as our ultimate source for both improvement and encouragement.

Michel de Champlain
`mdec@DeepObjectKnowledge.com`

Brian G. Patrick
`bpatrick@trentu.ca`

About the Authors

Michel de Champlain is the President and Principal Architect of DeepObjectKnowledge Inc., a firm that provides industry with mentoring and training support in object technologies. Michel holds a Ph.D. in Software Engineering from the École Polytechnique de Montréal and has held university appointments at the Collège Militaire Royal de Saint-Jean, the University of Canterbury in New Zealand, and Concordia University in Montréal. He has also been a regular invited speaker at the Embedded Systems Conference for the last fourteen years. Working in close collaboration with industry as well as academia, Michel has trained thousands of people throughout Canada, the United States, Europe, and down under in object-oriented analysis, design, and implementation. His current research interests include object-oriented languages, frameworks, design patterns, compilers, virtual machines, and real-time microkernels.

Brian G. Patrick is an Associate Professor of Computer Science/Studies at Trent University in Peterborough, Ontario. He first met Michel as a colleague at the Collège Militaire Royal de Saint-Jean and has developed a close working relationship with Michel over the years. Brian earned his Ph.D. in Computer Science from McGill University in Montréal, where he later completed an M.B.A. in Finance and International Business. His research interests have included heuristic search, parallel algorithms, and software reuse. He is currently investigating job scheduling schemes for parallel applications.

Contents