

Design Patterns in C#

A Hands-on Guide
with Real-world Examples

—

Second Edition

—

Vaskaran Sarcar

Foreword by Priya Shimanthoor

Apress®

Design Patterns in C#

A Hands-on Guide with
Real-world Examples

Second Edition

Vaskaran Sarcar

Foreword by Priya Shimanthoor

Apress®

Design Patterns in C#: A Hands-on Guide with Real-world Examples

Vaskaran Sarcar
Garia, Kolkata, West Bengal, India

ISBN-13 (pbk): 978-1-4842-6061-6
<https://doi.org/10.1007/978-1-4842-6062-3>

ISBN-13 (electronic): 978-1-4842-6062-3

Copyright © 2020 by Vaskaran Sarcar

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

Trademarked names, logos, and images may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, logo, or image we use the names, logos, and images only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Smriti Srivastava
Development Editor: Laura Berendson
Coordinating Editor: Shrikant Vishwakarma

Cover designed by eStudioCalamar

Cover image designed by Freepik (www.freepik.com)

Distributed to the book trade worldwide by Springer Science+Business Media New York, 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 www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail booktranslations@springernature.com; for reprint, paperback, or audio rights, please e-mail bookpermissions@springernature.com.

Apress titles 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 Print and eBook Bulk Sales web page at <http://www.apress.com/bulk-sales>.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/978-1-4842-6061-6. For more detailed information, please visit <http://www.apress.com/source-code>.

Printed on acid-free paper

This book is dedicated to all the unsung heroes and volunteers who are continuously fighting at the frontlines of the COVID-19 battle to save humanity and this beautiful world.

Table of Contents

About the Author	xxi
About the Technical Reviewers	xxiii
Foreword	xxv
Acknowledgments	xxvii
Preface	xxix
Part I: Gang of Four Design Patterns	1
Part I.A: Creational Patterns	3
Chapter 1: Singleton Pattern	5
GoF Definition.....	5
Concept.....	5
Real-World Example.....	5
Computer-World Example	6
Implementation	6
Class Diagram	8
Solution Explorer View.....	8
Demonstration 1	9
Output.....	11
Analysis	12
Q&A Session.....	15
Alternative Implementation.....	22
Analysis	23
Q&A Session.....	25

TABLE OF CONTENTS

- Chapter 2: Prototype Pattern 27**
 - GoF Definition..... 27
 - Concept..... 27
 - Real-World Example..... 27
 - Computer-World Example 28
 - Implementation 29
 - Class Diagram 31
 - Solution Explorer View..... 32
 - Demonstration 1 33
 - Output..... 36
 - Modified Implementation 36
 - Class Diagram 36
 - Demonstration 2 37
 - Output..... 39
 - Analysis 39
 - Q&A Session..... 41
 - Shallow Copy vs. Deep Copy..... 42
 - Demonstration 3 44
 - Output from a Shallow Copy..... 47
 - Analysis 48
 - Output from Deep Copy 49
 - Analysis 50
 - Q&A Session..... 50
 - Demonstration 4 51
 - Output..... 54
 - Analysis 55
- Chapter 3: Builder Pattern 57**
 - GoF Definition..... 57
 - Concept..... 57
 - Real-World Example..... 58
 - Computer-World Example 58

Implementation	58
Class Diagram	60
Solution Explorer View	60
Demonstration 1	62
Output	67
Analysis	68
Q&A Session	68
An Alternative Implementation	70
Class Diagram	73
Solution Explorer View	73
Demonstration 2	74
Output	78
Analysis	79
Q&A Session	79
Chapter 4: Factory Method Pattern	81
GoF Definition	81
Concept	81
Real-World Example	81
Computer-World Example	82
Implementation	82
Class Diagram	83
Solution Explorer View	83
Demonstration 1	84
Output	89
Modified Implementation 1	89
Partial Demonstration 1	89
Output	91
Analysis	91
Q&A Session	91

TABLE OF CONTENTS

- Modified Implementation 2 93
 - Partial Demonstration 2..... 94
 - Output..... 96
- Chapter 5: Abstract Factory Pattern..... 97**
 - GoF Definition..... 97
 - Concept..... 97
 - Real-World Example..... 98
 - Computer-World Example 98
 - Implementation 98
 - Class Diagram 101
 - Solution Explorer View..... 101
 - Demonstration 1 102
 - Output..... 106
 - Q&A Session..... 107
- Part I.B: Structural Patterns 111**
- Chapter 6: Proxy Pattern 113**
 - GoF Definition..... 113
 - Concept..... 113
 - Real-World Example..... 113
 - Computer-World Example 114
 - Implementation 114
 - Class Diagram 116
 - Solution Explorer View..... 116
 - Demonstration 1 117
 - Output..... 119
 - Q&A Session..... 119
 - Demonstration 2 122
 - Output..... 125

Chapter 7: Decorator Pattern	127
GoF Definition.....	127
Concept.....	127
Real-World Example.....	128
Computer-World Example	129
Implementation	129
Class Diagram	132
Solution Explorer View.....	133
Demonstration	134
Output.....	138
Q&A Session.....	139
Chapter 8: Adapter Pattern	143
GoF Definition.....	143
Concept.....	143
Real-World Example.....	143
Computer-World Example	145
Implementation	145
Class Diagram	148
Solution Explorer View.....	149
Demonstration 1	150
Output.....	153
Analysis	153
Types of Adapters.....	154
Object Adapters	154
Class Adapters.....	155
Q&A Session.....	155
Demonstration 2	155
Output.....	159
Analysis	159
Q&A Session.....	159

TABLE OF CONTENTS

- Chapter 9: Facade Pattern 163**
 - GoF Definition..... 163
 - Concept..... 163
 - Real-World Example..... 163
 - Computer-World Example 164
 - Implementation 164
 - Class Diagram 168
 - Solution Explorer View..... 169
 - Demonstration 170
 - Output..... 174
 - Q&A Session..... 175

- Chapter 10: Flyweight Pattern 177**
 - GoF Definition..... 177
 - Concept..... 177
 - Real-World Example..... 178
 - Computer-World Example 178
 - Implementation 179
 - Class Diagram 183
 - Solution Explorer View..... 184
 - Demonstration 1 184
 - Output..... 190
 - Analysis 192
 - Q&A Session..... 192
 - Demonstration 2 195
 - Output..... 196
 - Analysis 197

- Chapter 11: Composite Pattern..... 199**
 - GoF Definition..... 199
 - Concept..... 199
 - Real-World Example..... 200

Computer-World Example	200
Implementation	200
Class Diagram	201
Solution Explorer View	202
Demonstration	203
Output	208
Q&A Session	209
Chapter 12: Bridge Pattern	211
GoF Definition	211
Concept	211
Real-World Example	211
Computer-World Example	212
Implementation	212
Class Diagram	215
Solution Explorer View	215
Demonstration 1	216
Output	220
Additional Implementation	220
Class Diagram	224
Demonstration 2	225
Output	229
Q&A Session	229
Part I.C: Behavioral Patterns	233
Chapter 13: Visitor Pattern	235
GoF Definition	235
Concept	235
Real-World Example	238
Computer-World Example	238

TABLE OF CONTENTS

- Implementation 238
 - Class Diagram 242
 - Solution Explorer View 243
 - Demonstration 1 244
 - Output 249
 - Q&A Session 249
- Using Visitor Pattern and Composite Pattern Together 254
 - Step 1 257
 - Step 2 258
 - Step 3 259
 - Step 4 259
 - Step 5 260
 - Demonstration 2 260
 - Output 267
- Chapter 14: Observer Pattern 269**
 - GoF Definition 269
 - Concept 269
 - Real-World Example 273
 - Computer-World Example 273
 - Implementation 273
 - Class Diagram 277
 - Solution Explorer View 278
 - Demonstration 279
 - Output 283
 - Q&A Session 284
- Chapter 15: Strategy Pattern 287**
 - GoF Definition 287
 - Concept 287
 - Real-World Example 287
 - Computer-World Example 288

Implementation	288
Class Diagram	290
Solution Explorer View	291
Demonstration	292
Output	295
Q&A Session	296
Chapter 16: Template Method Pattern	299
GoF Definition	299
Concept	299
Real-World Example	299
Computer-World Example	300
Implementation	300
Class Diagram	302
Solution Explorer View	303
Demonstration 1	303
Output	306
Q&A Session	306
Demonstration 2	309
Output	312
Chapter 17: Command Pattern	315
GoF Definition	315
Concept	315
Real-World Example	316
Computer-World Example	316
Implementation	316
Class Diagram	319
Solution Explorer View	319
Demonstration 1	321
Output	325

TABLE OF CONTENTS

- Q&A Session..... 326
- Modified Implementation 329
 - Demonstration 2 331
 - Output..... 335
- Chapter 18: Iterator Pattern 337**
- GoF Definition..... 337
- Concept..... 337
- Real-World Example..... 338
- Computer-World Example 339
- Implementation 339
 - Class Diagram 340
 - Solution Explorer View..... 341
 - Demonstration 1 342
 - Output..... 347
 - Demonstration 2 348
 - Output..... 352
- Q&A Session..... 352
- Chapter 19: Memento Pattern..... 355**
- GoF Definition..... 355
- Concept..... 355
- Real-World Example..... 355
- Computer-World Example 356
- Implementation 356
 - Class Diagram 358
 - Solution Explorer View..... 359
 - Demonstration 1 359
 - Output..... 364
 - Analysis 365

Q&A Session.....	365
Modified Implementation	367
Class Diagram	368
Solution Explorer View.....	369
Demonstration 2	369
Output.....	374
Chapter 20: State Pattern	377
GoF Definition.....	377
Concept.....	377
Real-World Example.....	378
Computer-World Example	378
Implementation	378
Class Diagram	381
Solution Explorer View.....	381
Demonstration	383
Output.....	388
Q&A Session.....	389
Chapter 21: Mediator Pattern	393
GoF Definition.....	393
Concept.....	393
Real-World Example.....	393
Computer-World Example	394
Implementation	394
Class Diagram	400
Solution Explorer View.....	401
Demonstration 1	401
Output.....	407
Analysis	408

TABLE OF CONTENTS

- Q&A Session..... 408
- Modified Implementation 409
 - Demonstration 2 410
 - Output..... 415
- Chapter 22: Chain of Responsibility Pattern..... 419**
- GoF Definition..... 419
- Concept..... 419
- Real-World Example..... 420
- Computer-World Example 420
- Implementation 421
 - Class Diagram 426
 - Solution Explorer View..... 427
 - Demonstration 428
 - Output..... 433
- Q&A Session..... 433
- Chapter 23: Interpreter Pattern 437**
- GoF Definition..... 437
- Concept..... 437
- Real-World Example..... 440
- Computer-World Example 440
- Implementation 441
 - Class Diagram 443
 - Solution Explorer View..... 444
 - Demonstration 1 445
 - Output..... 449
- Another Implementation 449
 - Class Diagram 454
 - Solution Explorer View..... 454
 - Demonstration 2 455
 - Output..... 461
- Q&A Session..... 462

Part II: Additional Design Patterns	463
Chapter 24: Simple Factory Pattern	465
Definition.....	465
Concept.....	465
Real-World Example.....	465
Computer-World Example	466
Implementation	466
Class Diagram	468
Solution Explorer View.....	469
Demonstration	470
Output.....	473
Q&A Session.....	474
Chapter 25: Null Object Pattern	477
Definition.....	477
Concept.....	477
A Faulty Program	478
Output with Valid Input	481
Analysis with an Unwanted Input	481
A Potential Fix.....	482
Analysis	482
Real-World Example.....	483
Computer-World Example	483
Implementation	483
Class Diagram	485
Solution Explorer View.....	486
Demonstration	487
Output.....	490
Analysis	491
Q&A Session.....	491

TABLE OF CONTENTS

- Chapter 26: MVC Pattern 495**
 - Definition..... 495
 - Concept..... 496
 - Key Points to Remember 496
 - Variation 1 497
 - Variation 2 498
 - Variation 3 498
 - Real-World Example..... 499
 - Computer-World Example 500
 - Implementation 501
 - Class Diagram 503
 - Solution Explorer View..... 505
 - Demonstration 1 506
 - Output..... 513
 - Q&A Session..... 515
 - Modified Output..... 518

- Chapter 27: Patterns in Asynchronous Programming 521**
 - Overview 521
 - Using Synchronous Approach 523
 - Demonstration 1 523
 - Using Thread Class 525
 - Demonstration 2 525
 - Q&A Session 527
 - Using ThreadPool Class..... 527
 - Demonstration 3 530
 - Q&A Session 534
 - Using Lambda Expression with the ThreadPool Class 534
 - Demonstration 4 535
 - Using IAsyncResult Pattern..... 538
 - Polling Using Asynchronous Delegates 538
 - Demonstration 5 538

Q&A Session	543
Using AsyncWaitHandle of IAsyncResult	544
Demonstration 6	545
Using Asynchronous Callback.....	548
Demonstration 7	549
Q&A Session	552
Using Event-based Asynchronous Pattern	555
Demonstration 8	556
Q&A Session	565
Understanding Tasks.....	565
Demonstration 9	568
Using Task-based Asynchronous Pattern (TAP).....	571
Demonstration 10	571
Demonstration 11	573
Q&A Session	578
Using the async and await Keywords.....	579
Demonstration 12	581
Demonstration 13.....	587
Part III: Final Thoughts on Design Patterns	593
Chapter 28: Criticisms of Design Patterns.....	595
Q&A Session.....	598
Chapter 29: AntiPatterns	601
Overview	601
A Brief History of AntiPatterns	602
Examples of AntiPatterns	603
Types of AntiPatterns	605
Q&A Session.....	606
Chapter 30: FAQ	609

TABLE OF CONTENTS

Appendix A: A Brief Overview of GoF Design Patterns..... 615
 Q&A Session..... 619

Appendix B: Useful Resources 621

Appendix C: The Road Ahead 623

Appendix D: Important Updates in the Second Edition 625

Index..... 629

About the Author



Vaskaran Sarcar obtained his master's degree in software engineering from Jadavpur University, Kolkata (India), and an MCA from Vidyasagar University, Midnapore (India). He was a National Gate Scholar (2007–2009), and he has more than 12 years of experience in education and the IT industry. Vaskaran devoted his early years (2005–2007) in teaching at various engineering colleges. Later, he joined HP India PPS R&D Hub Bangalore, where he worked until August 2019. At the time of his retirement from the IT industry, he was a Senior Software Engineer and Team Lead at HP. Following his passion, Vaskaran is now an independent full-time author. His books include the following:

- *Getting Started with Advanced C#* (Apress, 2020)
- *Interactive Object-Oriented Programming in Java Second Edition* (Apress, 2019)
- *Java Design Patterns Second Edition* (Apress, 2019)
- *Design Patterns in C#* (Apress, 2018)
- *Interactive C#* (Apress, 2017)
- *Interactive Object-Oriented Programming in Java* (Apress, 2016)
- *Java Design Patterns* (Apress, 2016)
- *C# Basics: Test Your Skills* (Createspace, 2015)
- *Operating System: Computer Science Interview Series* (Createspace, 2014)

About the Technical Reviewers



Carsten Thomsen is a back-end developer primarily but works with smaller front-end bits as well. He has authored and reviewed several books and has created numerous Microsoft Learning courses on software development. Carsten works as a freelancer/contractor in various countries in Europe; Azure, Visual Studio, Azure DevOps, and GitHub are some of his favorite tools. An exceptional troubleshooter, he asks the right questions, including the less logical ones, in the most logical to least logical fashion. He also enjoys working with architecture, research, analysis, development, testing, and bug fixing. Carsten is a communicator with skills in mentoring, team leadership, research, and presenting new material.



Shekhar Kumar Maravi is a lead engineer in design and development whose main interests are programming languages, algorithms, and data structures. He obtained his master's degree in computer science and engineering from the Indian Institute of Technology, Bombay. After graduation, he joined Hewlett-Packard's R&D Hub in India to work on printer firmware. Currently, he is a technical lead engineer at Siemens Healthcare's R&D division. He can be reached by email at shekhar.maravi@gmail.com or via LinkedIn at www.linkedin.com/in/shekharmaravi.

Foreword

Written programs need to be flexible, easily maintainable, and reusable. How do we know that a program is as elegant as it can be? The answer is that a successful programmer must use two primary tools: a good programming language (here it is C#) and design patterns.

When working on a problem, it is unusual to tackle it by inventing a new solution that is completely dissimilar from the existing ones. One often recalls a similar problem and reuses the essence of its solution to solve the new problem. This kind of thinking in problem-solving is common to many different domains, such as software engineering.

Design patterns are important building blocks for designing and modeling applications on all platforms. Design patterns help us understand, discuss, and reuse applications on a specific platform. The most common reasons for studying patterns are the reuse of solutions and the establishment of common terminology. By reusing established designs, a developer gets a headstart on the problem and avoids common mistakes. The benefit of learning from the experience of others' results is that the developer does not have to reinvent solutions for recurring problems. The other reason for using patterns is that common terminology brings a common base of vocabulary and viewpoint of the problem for developers. It provides a common point of reference during the analysis and design phase of a project.

Vaskaran Sarcar, who has worked with me for several years now, has been a Most Valuable Professional over the years in C#. He is enthusiastic, knowledgeable, talented, curious, analytical, and a teacher of others. He gets to the root of any problem he is trying to resolve in a well-defined and organized way. He is very committed and works hard until he gets to the solution. He gets involved and is deeply focused while working on any problem.

FOREWORD

And that is also why I am excited about this book. The book brings the frequently complex world of design patterns into sharp focus with the approach used: the definition, the core concept, a real-life example, a computer-world example, and a sample program with output. In this edition, Vaskaran has provided asynchronous programming patterns usage using C#.

I look forward to seeing where developers can go with this easy approach and language, and the useful patterns they can build into the infrastructure of other languages.

—Priya Shimanthoor
Test Architect
Managed Print Services Team
Bangalore, India
June 3, 2020

Acknowledgments

First, I thank the Almighty. I sincerely believe that I could complete this book only with His blessings. I extend my deepest gratitude and thanks to the following people.

Ratanlal Sarkar and Manikuntala Sarkar: My dear parents, only with your blessings could I complete this work.

Indrani, my wife; **Ambika**, my daughter; **Aryaman**, my son: Sweethearts, once again, without your love, I could not proceed at all. I know that we need to limit many social gatherings and invitations to complete my books on time, and each time I promise you that I'll take a long break and spend more time with you.

Sambaran, my brother: Thank you for your constant encouragement.

Carsten: I know that whenever I was in need, your support was there. Thank you once more.

Sekhar: I know this time you helped only in the incremented version of the book, but thank you once more.

Ankit, my technical advisor in the first edition of this book: I always acknowledge your contribution and help. I know that your valuable comments were some of the key foundations for this enhanced edition.

Priya, my ex-colleague cum senior: A special thanks to you for investing your time to write the forewords for both editions of this book. When experts like you agree to write for me, I get the additional motivation to enhance the quality of my work.

Celestin, Laura, Smriti: Thanks for giving me another opportunity to work with you and Apress.

Shrikant: Thank you for your exceptional support to beautify my work.

The production team—Krishnan Sathyamurthy, Sherly, Ramraj, Selvakumar, and MathaRajamohan: Thank you guys; your efforts are extraordinary.

Lastly, I extend my deepest gratitude to my publisher, the editorial board members, and everyone who directly or indirectly supports this book.

Preface

Welcome to your journey through *Design Patterns in C# Second Edition*.

This book is an introductory guide to the design patterns that you want to use in C#. You probably know that the concept of design patterns became extremely popular with the Gang of Four's famous book *Design Patterns: Elements of Reusable Object-Oriented Software* (Addison-Wesley, 1994). That book was primarily focused on C++, but these concepts still apply in today's programming world.

C# had its first major release (C# 2.0) in 2005. Since then, it has become rich with new features and is now a popular programming language. In 2015, I wrote the book *Design Patterns in C#: Computer Science Interview Series*. In 2018, *Design Patterns in C#: A Hands-on Guide with Real-World Examples* was born. In these books, my core intention was to implement each of the 23 Gang of Four (GoF) design patterns with C# implementations. I wanted to present each pattern with simple examples. One thing was always on my mind when writing: I wanted to use the most basic constructs of C# so that the code would be compatible with both the upcoming version and the legacy version of C#. I have found this method helpful in the world of programming.

In the last few years, I got a lot of constructive feedback from my readers. This fully revised and updated version was created with that feedback in mind. I took the opportunity to update the formatting and correct some typos from the previous version of the book and add new content to this new edition. In this book, I focus on another important area; I call it the "doubt-clearing sessions." I knew that if I could add more information, such as alternative ways to write the implementations, the pros and cons of the patterns, when to choose one approach over another, and so on, readers would find this book even more helpful.

In this updated version of the original, the "Q&A Session" sections in each chapter are further enhanced. These sessions can help you learn about each pattern in more depth. In addition, you see more code explanations for all the programs, and in many cases, the programs are further simplified, and new programs are added for the patterns. *To learn about the most important enhancements in this edition, refer to Appendix D at the end of this book.*

How Is the Book Organized?

This book has three major parts.

Part I consists of the first 23 chapters, which discusses and implements all the GoF design patterns.

In the world of programming, there is no shortage of patterns, and each has its own significance. Part II discusses some additional design patterns (Simple Factory, Null Object, and MVC) that are equally important in today's world of programming. In this second edition, I dropped discussions of memory leaks, but I include several patterns from asynchronous programming. In modern applications, these patterns are very common.

Part III discusses the criticism of design patterns and overviews antipatterns, which are important when you implement the concepts of design patterns in your applications.

Each chapter is divided into six major parts: a definition (which is basically the intent in the GoF book), a core concept, a real-world example, a computer/coding-world example, a sample program with various output, and the "Q&A Session" section. These sections help you learn about each pattern in more depth.

Please remember that you have just started this journey. As you learn the concepts, try to write your own code; only then will you master an area.

You will be able to download all the book's source code from the Apress website. I plan to maintain the errata, and if necessary, I will also make updates and announcements there.

Prerequisite Knowledge

This book's target readers are those who are familiar with C# basic language constructs and pure object-oriented concepts, like polymorphism, inheritance, abstraction, encapsulation, and most importantly, how to compile or run a C# application in Visual Studio. This book does not invest time in easily available topics, such as how to install Visual Studio on your system, or how to write a "Hello World" program in C#, or how can you use an if-else statement or a while loop, and so forth. This book was written using the most basic features of C#, so for most of the programs in this book, you do not need to be familiar with C# advanced topics. The examples are simple and straightforward. I believe that the examples are written in such a way that even if you are familiar with another popular language, such as Java or C++, you can still easily grasp the concepts in this book.

Who Is This Book For?

In short, you want this book if your answer is “yes” to all of the following questions.

- Are you familiar with basic constructs in C# and object-oriented concepts like polymorphism, inheritance, abstraction, and encapsulation?
- Do you know how to set up your coding environment?
- Do you want to explore the design patterns in C# step by step?
- Do you want to explore GoF design patterns?
- Are you interested in learning about Simple Factory, Null Object, MVC, and asynchronous programming patterns?
- Do you want to know how the core constructs of C# work behind these patterns?

You probably don't want this book if the answer is “yes” to any of the following questions.

- Are you new to C#?
- Are you looking for advanced concepts in C#, excluding the topics mentioned previously?
- Are you interested in exploring a book where the focus is on GoF patterns (and the patterns listed in the previous section)?
- Do you dislike a book that uses Q&A sessions?
- “I do not like Windows, Visual Studio, and .NET Core. I want to learn and use C# without them.” Is this statement true for you?

Guidelines for Using This Book

Here are some suggestions to help you use this book more effectively.

- I assume that you have some knowledge of GoF design patterns. If you are new to design patterns, I suggest you quickly go through Appendix A, which helps you become familiar with the basic concepts of design patterns.

PREFACE

- If you are confident with what Appendix A covers, you can start with any part of the book. But I suggest you go through the chapters sequentially. The reason is that some fundamental design techniques may have been discussed in the Q&A Session of a previous chapter, and I do not repeat those techniques in later chapters.
- There is only one exception to the previous suggestion. There are three factory patterns: Simple Factory, Factory Method, and Abstract Factory. These three patterns are closely related, but the Simple Factory pattern does not directly fall into the GoF design catalog, so it appears in Part II of the book. Therefore, of the three patterns, I suggest that you begin with the Simple Factory.
- Except for a few programs in Chapter 27, all programs were executed and tested in .NET Core 3.1. The remaining programs were executed in .NET Framework 4.7.2 because .NET Core doesn't support certain functionalities. The specific reasons are discussed in Chapter 27.
- I used Visual Studio Community edition 2019 (version 16.3.9) in a Windows 10 environment. This Community edition is free. If you do not use the Windows operating system, you can use Visual Studio Code, which is a source code editor developed by Microsoft to support Windows, Linux, or macOS operating systems. This multiplatform IDE is free. When I started writing this book, I used the latest versions of C# available. In this context, it is useful to know that the C# language version is automatically selected based on your project's target framework(s) so that you always have the highest compatible version by default. In the most recent versions, Visual Studio doesn't support the UI to change the value, but you can change it by editing the .csproj file. The Visual Studio 2019 compiler and the .NET Core 3.0 SDK follow this rule. Therefore, you can simply say that when your target framework is .NET Core 3.x (or newer), you'll get C# 8.0 (and higher) by default. If you are interested in C# language versioning, go to <https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/configure-language-version>.

- Version updates are continuous, but I strongly believe that the versions should not matter much to you because I use the fundamental constructs of C# in this book. The code should execute smoothly in the upcoming versions of C#/Visual Studio as well. Although I believe that the results should not vary in other environments, you know the nature of a software-it can be naughty. So, I recommend that if you want to see the same output, it is best to mimic the same environment.
- You can download and install Visual Studio IDE from <https://visualstudio.microsoft.com/downloads/>. You should see what's shown in Figure P-1.

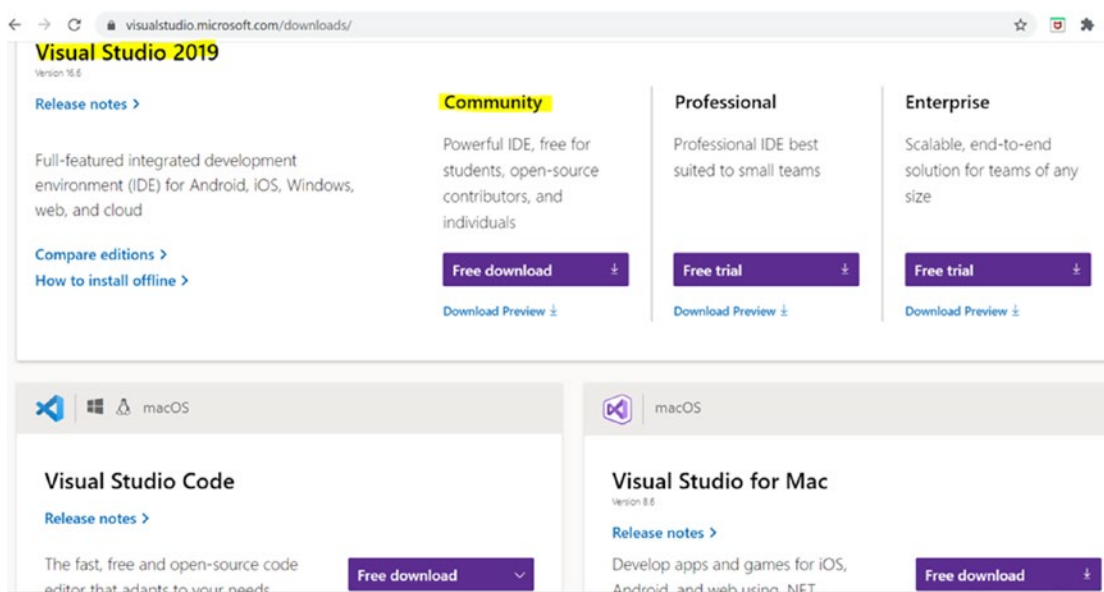


Figure P-1. Download link for Visual Studio 2019 and Visual Studio Code

Note At the time of this writing, this link works fine, and the information is correct. But the link and policies may change in the future.
