

Matt Weisfeld

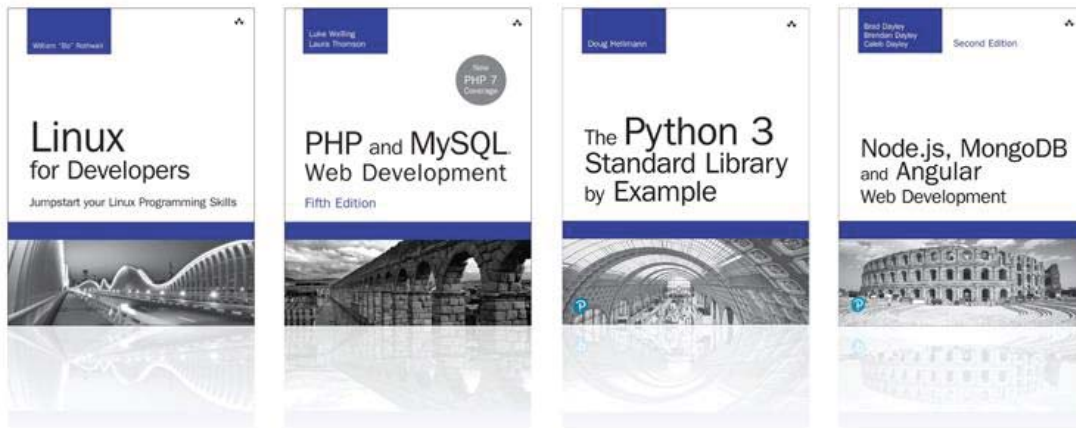
Fifth Edition



# The Object-Oriented Thought Process



# The Pearson Addison-Wesley Developer's Library



Visit [informit.com/devlibrary](http://informit.com/devlibrary) for a complete list of available publications.

**T**he **Developer's Library** series from Pearson Addison-Wesley provides practicing programmers with unique, high-quality references and tutorials on the latest programming languages and technologies they use in their daily work. All books in the Developer's Library are written by expert technology practitioners who are exceptionally skilled at organizing and presenting information in a way that is useful for other programmers.

Developer's Library titles cover a wide range of topics, from open source programming languages and technologies, mobile application developments, and web development to Java programming and more.



Make sure to connect with us!  
[informit.com/socialconnect](http://informit.com/socialconnect)

# **The Object-Oriented Thought Process**


**Fifth Edition**

# The Object-Oriented Thought Process

---

**Fifth Edition**

**Matt Weisfeld**

 **Addison-Wesley**

Boston • Columbus • New York • San Francisco • Amsterdam • Cape Town  
Dubai • London • Madrid • Milan • Munich • Paris • Montreal • Toronto • Delhi  
Mexico City • São Paulo • Sydney • Hong Kong • Seoul • Singapore • Taipei • Tokyo

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the publisher was aware of a trademark claim, the designations have been printed with initial capital letters or in all capitals.

The author and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

For information about buying this title in bulk quantities, or for special sales opportunities (which may include electronic versions; custom cover designs; and content particular to your business, training goals, marketing focus, or branding interests), please contact our corporate sales department at [corpsales@pearsoned.com](mailto:corpsales@pearsoned.com) or (800) 382-3419.

For government sales inquiries, please contact [governmentsales@pearsoned.com](mailto:governmentsales@pearsoned.com).

For questions about sales outside the U.S., please contact [intlcs@pearson.com](mailto:intlcs@pearson.com).

Visit us on the Web: [informit.com/aw](http://informit.com/aw)

*Library of Congress Control Number:* 2019930825

Copyright © 2019 Pearson Education, Inc.

All rights reserved. Printed in the United States of America. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. For information regarding permissions, request forms and the appropriate contacts within the Pearson Education Global Rights & Permissions Department, please visit [www.pearsoned.com/permissions/](http://www.pearsoned.com/permissions/).

ISBN-13: 978-0-13-518196-6

ISBN-10: 0-13-518196-8

1 19

Microsoft and/or its respective suppliers make no representations about the suitability of the information contained in the documents and related graphics published as part of the services for any purpose. All such documents and related graphics are provided “as is” without warranty of any kind. Microsoft and/or its respective suppliers hereby disclaim all warranties and conditions with regard to this information, including all warranties and conditions of merchantability, whether express, implied or statutory, fitness for a particular purpose, title and non-infringement. In no event shall Microsoft and/or its respective suppliers be liable for any special, indirect or consequential damages or any damages whatsoever resulting from loss of use, data or profits, whether in an action of contract, negligence or other tortious action, arising out of or in connection with the use or performance of information available from the services.

The documents and related graphics contained herein could include technical inaccuracies or typographical errors. Changes are periodically added to the information herein. Microsoft and/or its respective suppliers may make improvements and/or changes in the product(s)

and/or the program(s) described herein at any time. Partial screenshots may be viewed in full within the software version specified.

Microsoft® and Windows® are registered trademarks of the Microsoft Corporation in the U.S.A. and other countries. Screenshots and icons reprinted with permission from the Microsoft Corporation. This book is not sponsored or endorsed by or affiliated with the Microsoft Corporation.

**Editor-in-Chief**

Mark Taub

**Development Editor**

Mark Taber

**Managing Editor**

Sandra Schroeder

**Senior Project Editor**

Tonya Simpson

**Indexer**

Erika Millen

**Proofreader**

Abigail Manheim

**Technical Reviewer**

John Upchurch

**Editorial Assistant**

Cindy Teeters

**Cover Designer**

Chuti Prasertsith

**Compositor**

codeMantra

# Acknowledgments

As with the first four editions, this book required the combined efforts of many people. I would like to take the time to acknowledge as many of these people as possible, for without them, this book would never have happened.

First and foremost, I would like to thank my wife Sharon for all her help. Not only did she provide support and encouragement throughout this lengthy process, she is also the first line editor for all my writing.

I would also like to thank my mom and the rest of my family for their continued support.

It is hard to believe that the work on the first edition of this book began in 1998. For all these years, I have thoroughly enjoyed working with everyone at Pearson—on all five editions. Working with editors Mark Taber and Tonya Simpson on this edition has been a pleasure.

A special thanks goes to Jon Upchurch for his expertise with much of the code as well as the technical editing of the manuscript. Jon's insights into an amazing range of technical topics have been of great help to me.

Finally, thanks to my daughters, Stacy and Stephanie, and my cat, Paulo, for always keeping me on my toes.

# About the Author

**Matt Weisfeld** is a college professor, software developer, and author based in Cleveland, Ohio. Prior to teaching college full time, he spent 20 years in the information technology industry as a software developer, entrepreneur, and adjunct professor. Weisfeld holds an MS in computer science and an MBA. Besides several editions of *The Object-Oriented Thought Process*, Matt has authored two other software development books and published many articles in magazines and journals, such as *informit.com*, *developer.com*, *Dr. Dobb's Journal*, *The C/C++ Users Journal*, *Software Development Magazine*, *Java Report*, and the international journal *Project Management*.

# We Want to Hear from You!

As the reader of this book, *you* are our most important critic and commentator. We value your opinion and want to know what we're doing right, what we could do better, what areas you'd like to see us publish in, and any other words of wisdom you're willing to pass our way.

We welcome your comments. You can email or write to let us know what you did or didn't like about this book—as well as what we can do to make our books better.

*Please note that we cannot help you with technical problems related to the topic of this book.*

When you write, please be sure to include this book's title and author as well as your name and email address. We will carefully review your comments and share them with the author and editors who worked on the book.

Email: [community@informit.com](mailto:community@informit.com)

## Reader Services

Visit our website and register this book at [www.informit.com/register](http://www.informit.com/register) for convenient access to any updates, downloads, or errata that might be available for this book.



# Introduction

## THIS BOOK'S SCOPE

As the title suggests, this book is about the object-oriented (OO) thought process. Although choosing the theme and title of a book are important decisions, these decisions are not at all straightforward when dealing with a highly conceptual topic. Many books deal with one level or another of programming and object orientation. Several popular books cover topics including OO analysis, OO design, OO programming, design patterns, OO data (XML), the Unified Modeling Language (UML), OO web development, OO mobile development, various OO programming languages, and many other topics related to OO programming (OOP).

However, while poring over all these books, many people forget that all these topics are built on a single foundation: how you think in OO ways. Often, many software professionals, as well as students, dive into these books without taking the appropriate time and effort to really understand the design concepts behind the code.

I contend that learning OO concepts is not accomplished by learning a specific development method, a programming language, or a set of design tools. Object-oriented development is, simply put, a way of thinking. This book is all about the OO thought process.

Separating the languages, development practices, and tools from the OO thought process is not an easy task. Often, people are introduced to OO concepts by diving headfirst into a programming language. For example, many years ago a large number of C programmers were first introduced to object orientation by migrating directly to C++ before they were even remotely exposed to OO concepts.

It is important to understand the significant difference between learning OO concepts and programming in an OO language. This came into sharp focus for me well before I worked on the first edition of this book, when I read articles like Craig Larman's "What the UML Is—and Isn't," In this article he states,

Unfortunately, in the context of software engineering and the UML diagramming language, acquiring the skills to read and write UML notation seems to sometimes be equated with skill in object-oriented analysis and design. Of course, this is not so, and the latter is much more important than the former. Therefore, I recommend seeking education and educational materials in which intellectual skill in object-oriented analysis and design is paramount rather than UML notation or the use of a case tool.

Thus, although learning a modeling language is an important step, it is much more important to learn OO skills first. Learning UML before fully understanding OO concepts is similar to learning how to read an electrical diagram without first knowing anything about electricity.

The same problem occurs with programming languages. As stated earlier, many C programmers moved into the realm of object orientation by migrating to C++ before being directly exposed to OO concepts. This would always come out in an interview. Quite often developers who claim to be C++ programmers are simply C programmers using C++ compilers. Even now, with languages such as C# .NET, VB .NET, Objective-C, Swift, and Java well established, a few key questions in a job interview can quickly uncover a lack of OO understanding.

Early versions of Visual Basic are not OO. C is not OO, and C++ was developed to be backward compatible with C. Because of this, it is quite possible to use a C++ compiler while using only C syntax while forsaking all of C++'s OO features. Objective-C was designed as an extension to the standard ANSI C language. Even worse, a programmer can use just enough OO features to make a program incomprehensible to OO and non-OO programmers alike.

Thus, it is of vital importance that while you're learning to use OO development environments, you first learn the fundamental OO concepts. Resist the temptation to jump directly into a programming language, and instead take the time to learn the object-oriented thought process first.

## WHAT'S NEW IN THE FIFTH EDITION

As stated often in this introduction, my vision for the first edition was to stick to the concepts rather than focus on a specific emerging technology. Although I still adhere to this goal for the fifth edition, I also introduce more of the "counter-arguments" than were present in the earlier editions. By that I mean that although object-oriented development is, by far, the biggest game in town, it is not the only one.

Since the first edition of this book was completed in 1999, many technologies have emerged and some have faded. At the time, Java was just establishing itself and was the primary OO development language. Web pages would soon become a part of daily life and business. We all know how ubiquitous mobile devices have become. In the past 20 years software developers have encountered XML, JSON, CSS, XSLT, SOAP, and RESTful Web Services. Android devices use Java and now Kotlin, while iOS devices use Objective-C and Swift.

The point I am trying to make is that we have embraced a lot of technologies in the past 20 years (and four editions of the book). My primary goal for this edition is to condense all of this down to the original intent of the first edition, fundamental object-oriented concepts. I like to think that whatever success the first edition of the book had was because it focused on fundamental object-oriented concepts. In some ways we have gone full circle because this edition encapsulates all the technologies mentioned above.

Finally, the concepts that ultimately encapsulate these technologies into a design methodology are represented by SOLID, which is woven into all the chapters of this edition as well as two new chapters at the end of the book.

The five SOLID principles are

- **SRP**—Single Responsibility Principle
- **OCP**—Open/Close Principle
- **LSP**—Liskov Substitution Principle
- **IPS**—Interface Segregation Principle
- **DIP**—Dependency Inversion Principle

I often think of the first nine chapters as representing what I consider classical object-oriented principles. The last three chapters on design patterns, avoiding dependencies, and SOLID build on the classical principles and present a strong methodology.

## THE INTENDED AUDIENCE

This book is a general introduction to the concepts of object-oriented programming. The term *concepts* is important because, while code is certainly used to reinforce the topics covered, the primary focus of this book is to ground the reader in the object-oriented thought process. It is also important for programmers to understand that OOP does not represent a distinct paradigm (as many believe)—OOP is simply one part of a vast toolkit available to modern software developers.

When the material for the first edition of this book was initially created in 1995, OOP was in its infancy. I can say this because, other than pockets of OO languages such as Smalltalk, there really were no true object-oriented languages in play at the time. C++, which does not enforce object-oriented constructs, was the dominant C-based language. Java 1.0 was released in 1996 and C# 1.0 in 2002. In fact, when the first edition of this book was published in 1999, there was no certainty that OO would actually become the leading development paradigm. (Java 2 wasn't even released until December 1998.) Despite its current dominance, there are some interesting chinks in the OOP armor to be addressed.

Thus, the audience for the first edition differs from the audience today.

From 1995 until as late as 2010, I was basically retraining many structured programmers in the art of OOP. The vast majority of these students had grown up with COBOL, FORTRAN, C, and VB, both in college and on the job. Today, students graduating college, writing video games, creating websites, or producing mobile apps have almost certainly learned programming using an object-oriented language. Thus, the approach of the fifth edition of this book is significantly different from the first edition, or second, etc. Rather than teaching structured programmers to become OO developers, we are now teaching programmers who have grown up with OO languages.

The intended audience for this book includes business managers, designers, developers, programmers, and project managers: in short, anyone who wants to gain a general understanding of what object orientation is all about. My hope is that reading this book will provide a strong foundation for moving to other books covering more advanced topics.

## THE BOOK'S APPROACH

It should be obvious by now that I am a firm believer in becoming comfortable with the object-oriented thought process before jumping into a programming language or modeling language. This book is filled with examples of code and UML class diagrams; however, you do not need to know a specific programming language or UML to read it. After all I have said about learning the concepts first, why is there so much code and class diagrams?

First, code and class diagrams are great for illustrating OO concepts. Second, they are integral to the OO process and should be addressed at an introductory level. The key is not to focus on Java, C#, and so on but to use them as aids in the understanding of the underlying concepts.

Note that I really like using UML class diagrams as a visual aid to illustrate classes, and their attributes and methods. In fact, the class diagrams are the only component of UML used in this book. I believe that the UML class diagrams offer a great way to model the conceptual nature of object models. I continue to use object models as an educational tool to illustrate class design and how classes relate to one another.

The code examples in the book illustrate concepts such as loops and functions; however, understanding the code itself is not a prerequisite for understanding the concepts. It might be helpful to have a book at hand that covers specific languages' syntax if you want to get more detailed.

I cannot state too strongly that this book does *not* teach Java, C# .NET, VB .NET, Objective-C, Swift, or UML, all of which can command volumes unto themselves. It is also important to understand that this is a book of concepts, and the intent of the examples in this book is not, necessarily, to describe the *optimal* way to design your classes; they are an educational exercise meant to get you thinking about OO concepts. For example, it is obvious that you won't create many models using penguins and barkless dogs on the job—but using them is a fun way to demonstrate the concepts. With all of this in mind, it is my hope that this book will whet your appetite for other OO topics, such as OO analysis, object-oriented design, and OO programming.

## SOURCE CODE USED IN THIS BOOK

The sample code described throughout this book is available on the publisher's website. Go to [informit.com/register](http://informit.com/register) and register your book for access to downloads.