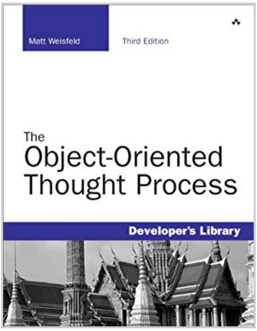

The Object-Oriented Thought Process (3rd Edition)

By Matt Weisfeld

The Object-Oriented Thought Process (3rd Edition) By Matt Weisfeld

The Object-Oriented Thought Process
Third Edition

Matt Weisfeld

An introduction to object-oriented concepts for developers looking to master modern application practices.

Object-oriented programming (OOP) is the foundation of modern programming languages, including C++, Java, C#, and Visual Basic .NET. By designing with objects rather than treating the code and data as separate entities, OOP allows objects to fully utilize other objects' services as well as inherit their functionality. OOP promotes code portability and reuse, but requires a shift in thinking to be fully understood. Before jumping into the world of object-oriented programming languages, you must first master *The Object-Oriented Thought Process*.

Written by a developer for developers who want to make the leap to object-oriented technologies as well as managers who simply want to understand what they are managing, *The Object-Oriented Thought Process* provides a solution-oriented approach to object-oriented programming. Readers will learn to understand object-oriented design with inheritance or composition, object aggregation and association, and the difference between interfaces and implementations. Readers will also become more efficient and better thinkers in terms of object-oriented development.

This revised edition focuses on interoperability across various technologies, primarily using XML as the communication mechanism. A more detailed focus is placed on how business objects operate over networks, including client/server architectures and web services.

"Programmers who aim to create high quality software–as all programmers should–must learn the varied subtleties of the familiar yet not so familiar beasts called objects and classes. Doing so entails careful study of books such as Matt

Weisfeld's The Object-Oriented Thought Process.*”*
–Bill McCarty, author of *Java Distributed Objects*, and *Object-Oriented Design in Java*

Matt Weisfeld is an associate professor in business and technology at Cuyahoga Community College in Cleveland, Ohio. He has more than 20 years of experience as a professional software developer, project manager, and corporate trainer using C++, Smalltalk, .NET, and Java. He holds a BS in systems analysis, an MS in computer science, and an MBA in project management. Weisfeld has published many articles in major computer trade magazines and professional journals.

The Object-Oriented Thought Process (3rd Edition)

By Matt Weisfeld

The Object-Oriented Thought Process (3rd Edition) By Matt Weisfeld

The Object-Oriented Thought Process
Third Edition

Matt Weisfeld

An introduction to object-oriented concepts for developers looking to master modern application practices.

Object-oriented programming (OOP) is the foundation of modern programming languages, including C++, Java, C#, and Visual Basic .NET. By designing with objects rather than treating the code and data as separate entities, OOP allows objects to fully utilize other objects' services as well as inherit their functionality. OOP promotes code portability and reuse, but requires a shift in thinking to be fully understood. Before jumping into the world of object-oriented programming languages, you must first master *The Object-Oriented Thought Process*.

Written by a developer for developers who want to make the leap to object-oriented technologies as well as managers who simply want to understand what they are managing, *The Object-Oriented Thought Process* provides a solution-oriented approach to object-oriented programming. Readers will learn to understand object-oriented design with inheritance or composition, object aggregation and association, and the difference between interfaces and implementations. Readers will also become more efficient and better thinkers in terms of object-oriented development.

This revised edition focuses on interoperability across various technologies, primarily using XML as the communication mechanism. A more detailed focus is placed on how business objects operate over networks, including client/server architectures and web services.

“Programmers who aim to create high quality software–as all programmers should–must learn the varied subtleties of the familiar yet not so familiar beasts called objects and classes. Doing so entails careful study of books such as Matt Weisfeld's The Object-Oriented Thought Process.*”*
–Bill McCarty, author of *Java Distributed Objects*, and *Object-Oriented Design in Java*

Matt Weisfeld is an associate professor in business and technology at Cuyahoga Community College in Cleveland, Ohio. He has more than 20 years of experience as a professional software developer, project manager, and corporate trainer using C++, Smalltalk, .NET, and Java. He holds a BS in systems analysis, an MS in computer science, and an MBA in project management. Weisfeld has published many articles in major computer trade magazines and professional journals.

The Object-Oriented Thought Process (3rd Edition) By Matt Weisfeld Bibliography

- Sales Rank: #415622 in Books
- Brand: Brand: Addison-Wesley Professional

- Published on: 2008-09-04
- Original language: English
- Number of items: 1
- Dimensions: 9.00" h x .71" w x 7.00" l, 1.25 pounds
- Binding: Paperback
- 360 pages

Introduction

This Book's Scope

As the title indicates, this book is about the object-oriented (OO) thought process. Obviously, choosing the theme and title of the book are important decisions; however, these decisions were not all that simple. Numerous books deal with various levels of object orientation. Several popular books deal with topics including OO analysis, OO design, OO programming, design patterns, OO data (XML), the Unified Modeling Language (UML), OO Internet development, various OO programming languages, and many other topics related to OO development.

However, while pouring over all of these books, many people forget that all of these topics are built on a single foundation: how you think in OO ways. It is unfortunate, but software professionals often dive into these books without taking the appropriate time and effort to really understand the concepts behind the content.

I contend that learning OO concepts is not accomplished by learning a specific development method or a set of tools. Doing things in an OO manner is, simply put, a way of thinking. This book is all about the OO thought process.

Separating the methods and tools from the OO thought process is not easy. Many people are introduced to OO concepts via one of these methods or tools. For example, years ago, most C programmers were first introduced to object orientation by migrating directly to C++—before they were even remotely exposed to OO concepts. Other software professionals were first introduced to object orientation by presentations that included object models using UML—again, before they were even exposed directly to OO concepts. It is not unusual to find that programming books and courses defer OO concepts until later in the learning process.

It is important to understand the significant difference between learning OO concepts and using the methods and tools that support the paradigm. This came into focus for me before I worked on the first edition of this book when I read articles such as 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.

Although learning a modeling language is an important step, it is much more important to learn OO skills first. Learning UML before 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. Many times developers who claim to be C++ programmers are simply C programmers using C++ compilers. Even now, with languages such as C# .NET, VB .NET, 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 (writing only C syntax) while forsaking all of C++'s OO features. 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 on the road to OO development, you first learn the fundamental OO concepts. Resist the temptation to jump directly into a programming language (such as VB .NET, C++, C# .NET or Java) or a modeling language (such as UML), and take the time to learn the object-oriented thought process.

In my first class in Smalltalk in the late 1980s, the instructor told the class that the new OO paradigm was a totally new way of thinking (*despite the fact that it has been around since the 60s*). He went on to say that although all of us were most likely very good programmers, about 10%–20% of us would never really grasp the OO way of doing things. If this statement is indeed true, it is most likely because some people never really take the time to make the paradigm shift and learn the underlying OO concepts.

What's New in the Third Edition

As stated often in this introduction, my vision for the first edition was primarily a conceptual book. Although I still adhere to this goal for the second and third editions, I have included several application topics that fit well with object-oriented concepts. For the third edition I expand on many of the topics of the second edition and well as include totally new chapters. These revised and updated concepts

- XML is used for object communication.
- Object persistence and serialization.
- XML integrated into the languages object definition.
- Adding properties to attributes.
- XML-based Internet applications.
- Client/Server technologies.
- Expanded code examples in Java, C# .NET and VB .NET.

The chapters that cover these topics are still conceptual in nature; however, many of the chapters include Java code that shows how these concepts are implemented. In this third edition, a code appendix is included that presents the chapter's examples in C# .NET and Visual Basic .NET.

The Intended Audience

This book is a general introduction to fundamental OO concepts with code examples to reinforce the concepts. One of the most difficult juggling acts was to keep the material conceptual while still providing a solid, technical code base. The goal of this book is to allow a reader to understand the concepts and technology without having a compiler at hand. However, if you do have a compiler available, then there is code to be investigated.

The intended audience includes business managers, designers, developers, programmers, project managers, and anyone who wants to gain a general understanding of what object orientation is all about. Reading this book should provide a strong foundation for moving to other books covering more advanced OO topics.

Of these more advanced books, one of my favorites remains *Object-Oriented Design in Java* by Stephen

Gilbert and Bill McCarty. I really like the approach of the book, and have used it as a textbook in classes I have taught on OO concepts. I cite *Object-Oriented Design in Java* often throughout this book, and I recommend that you graduate to it after you complete this one.

Other books that I have found very helpful include *Effective C++* by Scott Meyers, *Classical and Object-Oriented Software Engineering* by Stephen R. Schach, *Thinking in C++* by Bruce Eckel, *UML Distilled* by Martin Flower, and *Java Design* by Peter Coad and Mark Mayfield.

The conceptual nature of this book provides a unique perspective in regards to other computer technology books. While books that focus on specific technologies, such as programming languages, struggle with the pace of change, this book has the luxury of presenting established concepts that, while certainly being fine-tuned, do not experience radical changes. With this in mind, many of the books that were referenced several years ago, are still referenced because the concepts are still fundamentally the same.

This Book's Scope

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 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 Java, C# .NET, and VB .NET code and so many UML diagrams? First, they are all great for illustrating OO concepts. Second, both are vital to the OO process and should be addressed at an introductory level. The key is not to focus on Java, C# .NET, and VB .NET or UML, but to use them as aids in the understanding of the underlying concepts.

The Java, C# .NET and VB .NET 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, and VB .NET or UML, all of which can command volumes unto themselves. 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.

This Book's Conventions

The following conventions are used in this book:

- Code lines, commands, statements, and any other code-related terms appear in a `monospace` typeface.
- Placeholders that stand for what you should actually type appear in *`italic monospace`* . Text that you should type appears in **`bold monospace`** .
- Throughout the book, there are special sidebar elements, such as

---

Note - A Note presents interesting information related to the discussion—a little more insight or a pointer to some new technique.

Tip - A Tip offers advice or shows you an easier way of doing something.

Caution - A Caution alerts you to a possible problem and gives you advice on how to avoid it.

Source Code Used in This Book

You can download all the source code and examples discussed within this book from the publisher's website.

Read The Object-Oriented Thought Process (3rd Edition) By Matt Weisfeld for online ebook

The Object-Oriented Thought Process (3rd Edition) By Matt Weisfeld Free PDF d0wnl0ad, audio books, books to read, good books to read, cheap books, good books, online books, books online, book reviews epub, read books online, books to read online, online library, greatbooks to read, PDF best books to read, top books to read The Object-Oriented Thought Process (3rd Edition) By Matt Weisfeld books to read online.

Online The Object-Oriented Thought Process (3rd Edition) By Matt Weisfeld ebook PDF download

The Object-Oriented Thought Process (3rd Edition) By Matt Weisfeld Doc

The Object-Oriented Thought Process (3rd Edition) By Matt Weisfeld Mobipocket

The Object-Oriented Thought Process (3rd Edition) By Matt Weisfeld EPub

AFHN8QZV40W: The Object-Oriented Thought Process (3rd Edition) By Matt Weisfeld