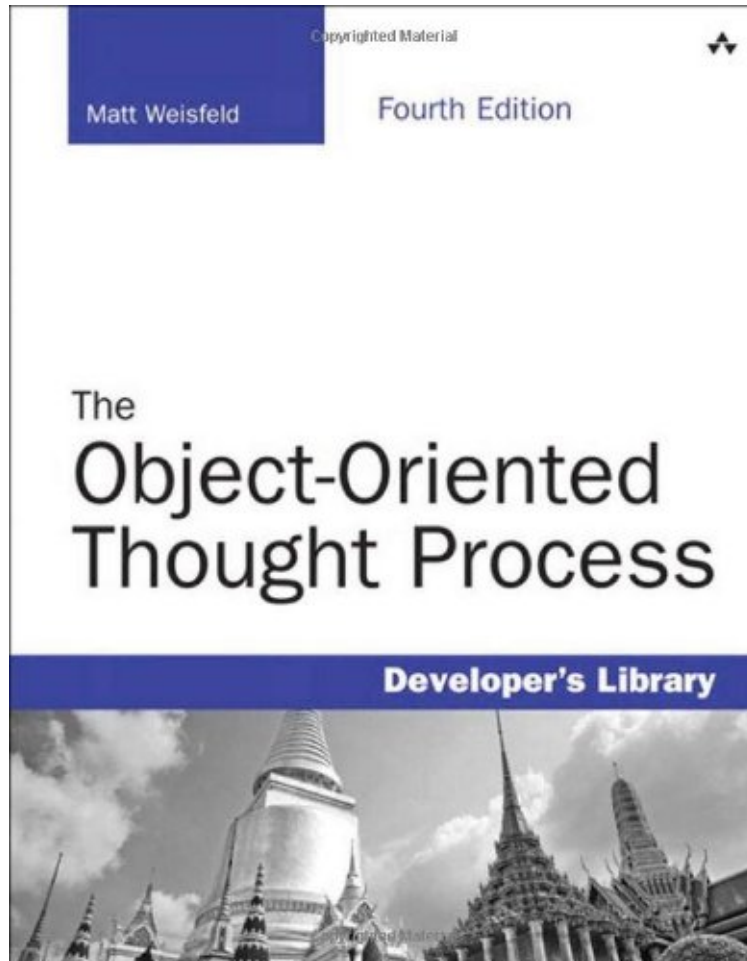


[Read ebook] The Object-Oriented Thought Process (4th Edition) (Developer's Library)

The Object-Oriented Thought Process (4th Edition) (Developer's Library)

Matt Weisfeld

**Download PDF | ePub | DOC | audiobook | ebooks*



DOWNLOAD



READ ONLINE

#408693 in Books 2013-03-23Original language:EnglishPDF # 1 8.90 x .90 x 7.00l, 1.15 #File Name: 0321861272336 pages | File size: 56.Mb

Matt Weisfeld : The Object-Oriented Thought Process (4th Edition) (Developer's Library) before purchasing it in order to gage whether or not it would be worth my time, and all praised The Object-Oriented Thought Process (4th Edition) (Developer's Library):

3 of 3 people found the following review helpful. A great introduction to OOP and to a new way of thinking...By ewomackAs this book's title, "The Object-Oriented Thought Process," suggests, object-oriented programming (OOP) involves a new way of thinking. Though the number of programmers who have not yet transitioned from procedural programming to OOP must be relatively small now (since the majority of current development jobs require at least decent knowledge of OOP), many beginning programming books, probably to save space, emphasize coding OOP rather than thinking or designing OOP. By contrast, this book not only solidifies OOP coding concepts, but it also initiates concepts of OOP design. Those looking for a way to energize their OOP power will find a lot to munch on

here. The book opens with a subject that likely doesn't affect many new programmers today: transitioning from procedural programming to Object-Oriented development. In the halcyon days before the dominance of .NET and Java, OOP was not a required application development skill. One could program in C, Visual Basic (VB) or in various other non-OOP languages without a thought of objects or classes. This book even argues that some early C++ developers were not using real OOP techniques, but simply writing C programs with C++ compilers. But once OOP Java broke into the mainstream and VB6 developers realized that they couldn't just dive into VB.NET as though it were just an upgrade to VB6, the divide between "real" OOP programmers and those who "just don't get it" arose. For the majority of beginning programmers within the past decade, OOP principles likely accompanied them from the start. So the "moving from procedural to OOP" problem will likely remain a largely generational one. The section nonetheless contains much useful information on the differences between procedural and OOP. The remainder of the book delivers a solid coverage of OOP principles. Many have accompanying code examples in Java, C# and VB.NET and UML diagrams. Some of the concepts will seem familiar, but the discussion goes far more in-depth than most beginner's books. Objects, classes, attributes, encapsulation, data-hiding, methods, object messaging, inheritance, polymorphism and other essential concepts get clearly elucidated with diagrams and code. Other topics usually not covered, at least in depth, in beginner OOP coding books also appear: composition (Has-a relationships), abstraction and interface vs. implementation. Some sections emphasize object thinking, such as thinking abstractly and in terms of object behavior. Later sections deal with constructors (a topic now vital to Dependency Injection), overloading, class modeling, the golden fleece of reuse, maintenance, object comparison and marshaling. One section provides a walk-through of a real-world example, a Black Jack game, of identifying the required classes, behaviors and attributes and how they should interact. An in-depth discussion of "is-a" versus "has-a" relationships helps elucidate the design principle to "favor composition over inheritance" or at least help decide when to most efficiently use either concept. Later chapters put objects into different contexts, such as XML, persistence with databases, the internet and client/server applications. These sections are less in-depth and provide cursory overviews to give one a taste of the technologies presented. The final chapter introduces design patterns, a crucial topic for OOP design. Though it too remains at a very high-level, it does cover a few patterns, though not deeply: Model/View/Controller (a now dominant pattern), Singleton, Adapter and Iterator. It also introduces the concept of an antipattern, or how not to do things. This book will not make anyone an expert in OOP design, though it's difficult to see how anyone could become an expert without first mastering the knowledge it contains. Most importantly, it emphasizes concepts that are not purely code-based and helps one think in new ways about how to structure an application. Anyone who has this feeling that they haven't quite absorbed OOP design principles should read this book end to end. The final chapter on design patterns also provides a perfect segue into further studies. Lastly, the book is currently in its 3rd edition, which appeared in 2008, so some of the information here may seem out of date. A planned forth edition will tentatively appear in March of 2013 that will cover unit testing, web services and mobile applications. Nonetheless, for those who can't wait, the 3rd edition still contains enough useful information about OOP to make it worth a read even now.

4 of 4 people found the following review helpful. Very Useful Case-Study Removed from 4th Edition! By G. L. Anderson For the 4th edition, the author removed a well-done, useful case-study - the BlackJack simulation - from Chapter 6. That has significantly reduced the usefulness of the book in my opinion. The 20(!) page example clearly showed how to move from a text description of the game to a collection of classes for the simulation program and explained the logic behind the various decisions made along the way. It was GREAT for people who are new to OO design. I cannot for the life of me understand why that case study was removed from the latest edition. There was no explanation in the forward. I am returning the 4th edition for a refund.

0 of 0 people found the following review helpful. Good info By DarthWI I am back in school working on another degree in IT, so have to do some programming. I've had classes in a few scripting and programming languages, but don't work as a developer, so my programming skills are still "novice". I bought this book after a recommendation from a fellow classmate, and read it to supplement and review textbook Java programming coursework. For an experienced object-oriented programmer this is probably too basic of a book, but for a student or someone new to programming, the information here is clear and explanatory. Even if someone were looking at programming as a possible career, reading this book to get an idea if you can even get some basic OO concepts could be useful.

The Object-Oriented Thought Process, Fourth Edition 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#, Visual Basic .NET, Ruby, and Objective-C. Objects also form the basis for many web technologies such as JavaScript, Python, and PHP. It is of vital importance to learn the fundamental concepts of object orientation before starting to use object-oriented development environments. OOP promotes good design practices, code portability, and reuse but it requires a shift in thinking to be fully understood. Programmers new to OOP should resist the temptation to jump directly into a particular programming language (such as Objective-C, VB .NET, C++, C# .NET, or Java) or a modeling language (such as UML), and instead first take the time to learn what author Matt Weisfeld calls the object-oriented thought process. Written by a developer for

developers who want to make the leap to object-oriented technologies, The Object-Oriented Thought Process provides a solutions-oriented approach to object-oriented programming. Readers will learn to understand the proper uses of inheritance and composition, the difference between aggregation and association, and the important distinction between interfaces and implementations. While programming technologies have been changing and evolving over the years, object-oriented concepts remain a constant no matter what the platform. This revised edition focuses on interoperability across programming technologies, whether you are using objects in traditional application design, in XML-based data transactions, in web page development, in mobile apps, or in any modern programming environment. 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 Contents at a Glance* 1 Introduction to Object-Oriented Concepts 2 How to Think in Terms of Objects 3 Advanced Object-Oriented Concepts 4 The Anatomy of a Class 5 Class Design Guidelines 6 Designing with Objects 7 Mastering Inheritance and Composition 8 Frameworks and Reuse: Designing with Interfaces and Abstract Classes 9 Building Objects and Object-Oriented Design 10 Creating Object Models 11 Objects and Portable Data: XML and JSON 12 Persistent Objects: Serialization, Marshaling, and Relational Databases 13 Objects in Web Services, Mobile Apps, and Hybrids 14 Objects and Client/Server Applications 15 Design Patterns

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 the first three editions of *The Object-Oriented Thought Process*, he has authored two other software development books and published many articles in magazines and journals, such as *developer.com*, *Dr. Dobbs Journal*, *The C/C++ Users Journal*, *Software Development Magazine*, *Java Report*, and the international journal *Project Management*.