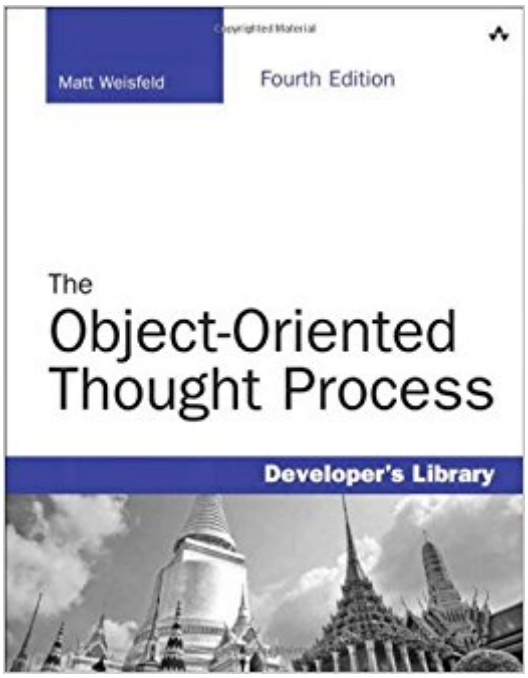
The Object-Oriented Thought Process (4th Edition) (Developer's Library)



Synopsis

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 reuseÃ¢â ¬â œbut 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 Ã¢â ¬Å“the object-oriented thought process.Ã¢â ¬Â• 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 constantÃ¢â ¬â œno 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. Ã¢â ¬Å“Programmers who aim to create high quality softwareÃ¢â ¬â œas all programmers shouldÃ¢â ¬â œmust 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 WeisfeldÃ¢â ¬â„¢s The Object-Oriented Thought Process.Ã¢â ¬Â• Ã¢â ¬â œBill McCarty, author of Java Distributed Objects, and Object-Oriented Design in Java Contents at a Glance 1 Introduction to Object-Oriented Concepts2 How to Think in Terms of Objects3 Advanced Object-Oriented Concepts4 The Anatomy of a Class5 Class Design Guidelines6 Designing with Objects7 Mastering Inheritance and Composition8 Frameworks and Reuse: Designing with Interfaces and Abstract Classes9 Building Objects and Object-Oriented Design10 Creating Object Models11 Objects and Portable Data: XML and JSON12 Persistent Objects: Serialization, Marshaling, and Relational Databases13 Objects in Web Services, Mobile Apps, and Hybrids14 Objects and Client/Server Applications15 Design Patterns

Book Information

Series: Developer’s Library

Paperback: 336 pages

Publisher: Addison-Wesley Professional; 4 edition (March 23, 2013)

Language: English

ISBN-10: 0321861272

ISBN-13: 978-0321861276

Product Dimensions: 7 x 0.9 x 8.9 inches

Shipping Weight: 1.2 pounds (View shipping rates and policies)

Average Customer Review: 3.9 out of 5 stars 66 customer reviews

Best Sellers Rank: #184,945 in Books (See Top 100 in Books) #76 inÃ Â Books > Humor & Entertainment > Movies > Guides & Reviews #78 inÃ Â Books > Textbooks > Computer Science > Object-Oriented Software Design #234 inÃ Â Books > Textbooks > Computer Science > Software Design & Engineering

Customer Reviews

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. DobbÃ¢â ¬â„¢s Journal, The C/C++ Users Journal, Software Development Magazine, Java Report, and the international journal Project Management.

The first chapter is profoundly enlightening. There is a reason I like to buy books on programming concepts rather than exclusively relying on the blogosphere. I spent a week at work trying to understand our legacy codebase and after reading the first chapter of this book I immediately starting to understand our codebase with much more clarity and fixed the issue we were dealing with very quickly. I also began to immediately see many ways in which the way things are currently implemented aren’t correct and create more problems than help.---- UpdateI finished reading this book and all I can say is great things about it. The author understands the subject AND the reader so well that he nearly 100% of the time know exactly when to explain things a little more or less. My code has become so much cleaner as a result of reading this and my understanding of design patterns is much much better now. This was the foundational knowledge I was looking for that would

tie together so many ideas like SOLID programming, separation of concerns, and as mentioned design patterns. I read this book in about 2 weeks because it was written in a language that makes it easy to absorb and fly through. I think I was ready for this knowledge as well after having written tons of programs without this solid background 'theory'.

As 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 halycon 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 segueway 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 tenatively 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.

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.

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 (4th Edition) (Developer's Library) Object-Oriented Analysis and Design with the Unified Process (Available Titles CengageNOW) Object-Oriented Programming in C++ (4th Edition) Programming in C (4th Edition) (Developer's Library) Introduction to Programming with Greenfoot: Object-Oriented Programming in Java with Games and Simulations (2nd Edition) Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd Edition) Object-Oriented Modeling and Design with UML (2nd Edition) Tools For Structured and Object-Oriented Design (7th Edition) Object-Oriented Programming in Java: A Graphical Approach, Preliminary Edition Object-Oriented Analysis and Design with Applications (3rd Edition) An Introduction to Object-Oriented Programming (3rd Edition) Object Oriented Software Development Using Java (2nd Edition) Object Lessons for a Year: 52 Talks for the Children's Sermon Time (Object Lesson Series) Systems Analysis and Design: An Object-Oriented Approach with UML C++ and Object-Oriented Numeric Computing for Scientists and Engineers Design Patterns: Elements of Reusable Object-Oriented Software Java Methods: An Introduction to Object Oriented Programming Object-Oriented Data Structures Using Java An Object-Oriented Approach to Programming Logic and Design Practical Object-Oriented Design in Ruby: An Agile Primer (Addison-Wesley Professional Ruby)