Practical Object Oriented Design With Uml

Looks at the principles and clean code, includes case studies showcasing the practices of writing clean code, and contains a list of heuristics and "smells" accumulated from the process of writing clean code.

Take a step beyond syntax to discover the true art of software design, with Java as your paintbrush and objects on your palette. This in-depth discussion of how, when, and why to use objects enables you to create programs that not only work smoothly, but are easy to maintain and upgrade using Java of any other object-oriented language! Companion CD software Pc.zip (8.4MB) Unix.zip (541K)

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.

Object-Oriented Programming under Windows presents object-oriented programming (OOP) techniques that can be used in Windows programming. The book is comprised of 15 chapters that tackle an area in OOP. Chapter 1 provides an introductory discourse about OOP, and Chapter 2 covers the programming languages. Chapter 3 deals with the Windows environment, while Chapter 4 discusses the creation of application. Windows and dialogue boxes, as well as controls and standard controls, are tackled. The book then covers menus and event response. Graphics operation, clipboard, bitmaps, icons, and cursors are also dealt with. The book also tackles disk file access, and then discusses the help file system. The last chapter covers data transfer. The text will be of great use to individuals who want to write Windows based programs.

Introduction to Practical System Modeling

Design Patterns Explained

Practical Application of Object-Oriented Techniques to Relational Databases

Beginning C# Object-Oriented Programming

Design Patterns

A Handbook of Agile Software Craftsmanship

Learn the tools and techniques needed to design and implement moderate-sized software systems! Do you want to gain the necessary skills to effectively write moderate-sized (10,000 to 50,000 line) programs? Would you like to develop a more advanced understanding of object-oriented design and learn how to implement important design and style rules? Do you want to be able to take a project from the concept stage to completion? This is all possible with Steven Reiss's innovative text, A Pracical Introduction to Software Design with C++. Reiss provides you with all the tools and techniques to enable you to design and implement moderate-sized software systems alone or in a team. The book details the proper use of inheritance, design notations using a simplified form of OMT to describe designs, the use of object libraries such as STL, creating library classes, and the use of design patterns. You'll also find useful discussions on advanced language and programming features such as exception handling, interprocess communication, and debugging tools and techniques.

Praise for Design Patterns in Ruby " Design Patterns in Ruby documents smart ways to resolve many problems that Ruby developers commonly encounter. Russ Olsen has done a great job of selecting classic patterns and augmenting these with newer patterns that have special relevance for Ruby. He clearly explains each idea, making a wealth of experience available to Ruby developers for their own daily work." —Steve Metsker, Managing Consultant with Dominion Digital, Inc. "This book provides a great demonstration of the key 'Gang of Four' design patterns without resorting to overly technical explanations. Written in a precise, yet almost informal style, this book covers enough ground that even those without prior exposure to design patterns will soon feel confident applying them using Ruby. Olsen has done a great job to make a book about a classically 'dry' subject into such an engaging and even occasionally humorous read." —Peter Cooper "This book renewed my interest in understanding patterns after a decade of good intentions. Russ picked the most useful patterns for Ruby and introduced them in a straightforward and logical manner, going beyond the GoF's patterns. This book has improved my use of Ruby, and encouraged me to blow off the dust covering the GoF book." —Mike Stok " Design Patterns in Ruby is a great way for programmers from statically typed objectoriented languages to learn how design patterns appear in a more dynamic, flexible language like Ruby." —Rob Sanheim, Ruby Ninja, Relevance Most design pattern books are based on C++ and Java. But Ruby is different—and the language's unique qualities make design patterns easier to implement and use. In this book, Russ Olsen demonstrates how to combine Ruby's power and elegance with patterns, and write more sophisticated, effective software with far fewer lines of code. After reviewing the history, concepts, and goals of design patterns, Olsen offers a quick tour of the Ruby language—enough to allow any experienced software developer to immediately utilize patterns with Ruby. The book especially calls attention to Ruby features that simplify the use of patterns, including dynamic typing, code closures, and "mixins" for easier code reuse. Fourteen of the classic "Gang of Four" patterns are considered from the Ruby point of view, explaining what problems each pattern solves, discussing whether traditional implementations make sense in the Ruby environment, and introducing Ruby-specific improvements. You'll discover opportunities to implement patterns in just one or two lines of code, instead of the endlessly repeated boilerplate that conventional languages often require. Design Patterns in Ruby also identifies innovative new patterns that have emerged from the Ruby community. These include ways to create custom objects with metaprogramming, as well as the ambitious Rails-based "Convention Over Configuration" pattern, designed to help integrate entire applications and frameworks. Engaging, practical, and accessible, Design Patterns in Ruby will help you build better software while making your Ruby programming experience more rewarding.

"This book manages to convey the practical use of UML 2 in clear and understandable terms with many examples and guidelines. Even for people not working with the Unified Process, the book is still of great use. UML 2 and the Unified Process, Second Edition is a must-read for every UML 2 beginner and a helpful guide and reference for the experienced practitioner." --Roland Leibundgut, Technical Director, Zuehlke Engineering Ltd. "This book is a good starting point for organizations and individuals who are adopting UP and need to understand how to provide visualization of the different aspects needed to satisfy it. " --Eric Naiburg, Market Manager, Desktop Products, IBM Rational Software This thoroughly revised edition provides an indispensable and practical guide to the complex process of object-oriented analysis and design using UML 2. It describes how the process of OO analysis and design fits into the software development lifecycle as defined by the Unified Process (UP). UML 2 and the Unified Process contains a wealth of practical, powerful, and useful techniques that you can apply immediately. As you progress through the text, you will learn OO analysis and design techniques, UML syntax and semantics, and the relevant aspects of the UP. The book provides you with an accurate and succinct summary of both UML and UP from the point of view of the OO analyst and designer. This book provides Chapter roadmaps, detailed diagrams, and margin notes allowing you to focus on your needs Outline summaries for each chapter, making it ideal for revision, and a comprehensive index that can be used as a reference New to this edition: Completely revised and updated for UML 2 syntax Easy to understand explanations of the new UML 2 semantics More real-world examples A new section on the Object Constraint Language (OCL) Introductory material on the OMG's Model Driven Architecture (MDA) The accompanying website provides A complete example of a simple e-commerce system Open source tools for requirements engineering and use case modeling Industrial-strength UML course materials based on the book

This highly practical book shows systems professionals how to apply object-oriented techniques to relational databases immediately. Burleson demonstrates approaches that enable legacy databases--databases already in existence--to function within the scope of an object-oriented technology application.

A Practical Approach

Practical Object-Oriented Development in C++ and Java

A Student Guide to Object-Oriented Development

Confessions of a Java Framework Architect

Practical Object-oriented Design in Ruby : an Agile Primer

Object-Oriented Design And Patterns

Although the theory of object-oriented programming languages is far from complete, this book brings together the most important contributions to its development to date, focusing in particular on how advances in type systems and semantic models can contribute to new language designs.The fifteen chapters are divided into five parts: Objects and Subtypes, Type Inference, Coherence, Record Calculi, and Inheritance. The chapters are organized approximately in order of increasing complexity of the programming language constructs they consider - beginning with variations on Pascal- and Algol-like languages, developing the theory of illustrative record object models, and concluding with research directions for building a more comprehensive theory of object-oriented programming languages.Part I discusses the similarities and differences between "objects" and algebraic-style abstract data types, and the fundamental concept of a subtype. Parts II-IV are concerned with the "record model" of object-oriented languages. Specifically, these chapters discuss static and dynamic semantics of languages with simple object models that include a type or class hierarchy but do not explicitly provide what is often called dynamic binding. Part V considers extensions and modifications to record object models, moving closer to the full complexity of practical object-oriented languages.Carl A. Gunter is Professor in the Department of Computer and Information Science at the University of Pennsylvania. John C. Mitchell is Professor in the Department of Computer Science at Stanford University.

Practical OO development tips for the C++ and Java programmer Practical Object-Oriented Development in C++ and Java offers advice on real-world ways to use these powerful programming languages and techniques. Using the Unified Modeling Language (UML) methodology, expert Cay S. Horstmann gives you clear, concise explanations of object-oriented design, C++, and Java in a way that makes these potentially daunting operations more accessible than they've ever been before. Horstmann compares and contrasts features of C++ and Java to give you a deeper understanding of OO design. He separates the genuinely useful C++, Java, and UML features from the less effective and potentially harmful ones. Horstmann shows you how to determine the best programming practice for whatever application you're in; provides the kind of eye-opening design tips and style rules that can only come from experience; and demystifies advanced topics like frameworks and object persistence. Dozens of illuminating programming examples are readily accessible through the accompanying Web site. Useful code is available for smart pointers, easy output formatting in C++ and Java, a set of classes that makes STL safe to use, and a nifty utility that automatically extracts header files. This unique book: * Offers over 100 practical design hints for good class design * Covers the essential OO features of Java 1.1-like serialization and reflection * Uses the C++ Standard Template Library (STL) throughout * Covers CRC cards in addition to UML

OOAD Cookbook: Introduction to Practical System Modeling is a modern, practical, and approachable guide to help students design and develop code that is modular, maintainable, and extensible. Whether you are a developer, devops, QA tester, systems analyst, or IT, this book will introduce the concepts to build a strong foundation in object-oriented methodologies. Step-by-Step instructions along with vivid examples and illustrations offer a fresh, practical, and approachable plan to learn object-oriented design. Students will learn and be exposed to efficient design through methodical analysis, UML diagrams, system architectures, and essential design principles so that they can design software pragmatically.

You might think more than enough design books exist in the programming world already. In fact, there are so many that it makes sense to ask why you would read yet another. Is there really a need for yet another design book? In fact, there is a greater need than ever before, and Practical API Design: Confessions of a Java Framework Architect fills that need! Teaches you how to write an API that will stand the test of time Written by the designer of the NetBeans API at Sun Technologies Based on best practices, scalability, and API design patterns

Mastering Object-Oriented Design in C++

Object-Oriented Analysis and Design

Build robust and maintainable object-oriented Python applications and libraries, 4th Edition

Practical Object-Oriented Design

Practical Object-oriented Design in Ruby

Object Thinking

Summary The Well-Grounded Rubyist, Third Edition is a beautifully written tutorial that begins with your first Ruby program and takes you all the way to sophisticated topics like reflection, threading, and recursion. Ruby masters David A. Black and Joe Leo distill their years of knowledge for you, concentrating on the language and its uses so you can use Ruby in any way you choose. Updated for Ruby 2.5. Purchase of the print book includes a free eBook in PDF, Kindle, and ePub formats from Manning Publications. About the Technology Designed for developer productivity, Ruby is an easy-to-learn dynamic language perfect for creating virtually any kind of software. Its famously friendly development community, countless libraries, and amazing tools, like the Rails framework, have established it as the language of choice for high-profile companies, including GitHub, SlideShare, and Shopify. The future is bright for the well-grounded Rubyist! About the Book In The Well-Grounded Rubyist, Third Edition, expert authors David A. Black and Joseph Leo deliver Ruby mastery in an easy-to-read, casual style. You'll lock in core principles as you write your first Ruby programs. Then, you'll progressively build up to topics like reflection, threading, and recursion, cementing your knowledge with high-value exercises to practice your skills along the way. What's Inside Basic Ruby syntax Running Ruby extensions FP concepts like currying, side-effect-free code, and recursion Ruby 2.5 updates About the Reader For readers with beginner-level programming skills. About the Authors David A. Black is an internationally known Ruby developer and author, and a cofounder of Ruby Central. Ruby teacher and advocate Joseph Leo III is the founder of Def Method and lead organizer of the Gotham Ruby Conference. Table of Contents PART 1 RUBY FOUNDATIONS Bootstrapping your Ruby literacy Objects, methods, and local variables Organizing objects with classes Modules and program organization The default object (self), scope, and visibility Control-flow techniques PART 2 BUILT-IN CLASSES AND MODULES Built-in essentials Strings, symbols, and other scalar objects Collection and container objects Collections central: Enumerable and Enumerator Regular expressions and regexp-based string operations File and I/O operations PART 3 RUBY DYNAMICS Object individuation Callable and runnable objects Callbacks, hooks, and runtime introspection Ruby and functional programming

Object-oriented analysis and design (OOAD) has over the years, become a vast field, encompassing such diverse topics as design process and principles, documentation tools, refactoring, and design and architectural patterns. For most students the learning experience is incomplete without implementation. This new textbook provides a comprehensive introduction to OOAD. The salient points of its coverage are: • A sound footing on object-oriented concepts such as classes, objects, interfaces, inheritance, polymorphism, dynamic linking, etc. • A good introduction to the stage of requirements analysis. • Use of UML to document user requirements and design. • An extensive treatment of the design process. • Coverage of implementation issues. • Appropriate use of design and architectural patterns. • Introduction to the art and craft of refactoring. • Pointers to resources that further the reader's knowledge. All the main case-studies used for this book have been implemented by the authors using Java. The text is liberally peppered with snippets of code, which are short and fairly self-explanatory and easy to read. Familiarity with a Java-like syntax and a broad understanding of the structure of Java would be helpful in using the book to its full potential.

It is a pleasure to present the proceedings of the 22nd European Conference on Object-Oriented Programming (ECOOP 2008) held in Paphos, Cyprus. The conference continues to serve a broad object-oriented community with a tech- cal program spanning theory and practice and a healthy mix of industrial and academic participants. This year a strong workshop and tutorial program c- plementedtheamaintechnicaltrack.Wehad13workshopsand8tutorials,aswell as the co-located Dynamic Language Symposium (DLS). Finally, the program was rounded out with a keynote by Rachid Guerraoui and a banquet speech by James Noble. As in previous years, two Dahl-Nygaard awards were selected by AITO, and for the ?rst time, the ECOOP Program Committee gave a best paper award. Theproceedingsinclude27papersselectedfrom138submissions.Thepapers werereviewed in a single-blind process with three to ?ve reviews per paper. P- liminaryversionsofthereviewsweremadeavailabletotheauthorsaweekbefore the PC meeting to allow for short (500 words or less) author responses. The - sponses were discussed at the PC meeting and were instrumental in reaching decisions. The PC discussions followed Oscar Nierstrasz'Champion pattern. PC papers had ?ve reviews and were held at a higher standard.

The Complete Guide to Writing More Maintainable, Manageable, Pleasing, and Powerful Ruby Applications Ruby's widely admired ease of use has a downside: Too many Ruby and Rails applications have been created without concern for their long-term maintenance or evolution. The Web is awash in Ruby code that is now virtually impossible to change or extend. This text helps you solve that problem by using powerful real-world object-oriented design techniques, which it thoroughly explains using simple and practical Ruby examples. Sandi Metz has distilled a lifetime of conversations and presentations about object-oriented design into a set of Ruby-focused practices for crafting manageable, extensible, and pleasing code. She shows you how to build new applications that can survive success and repair existing applications that have become impossible to change. Each technique is illustrated with extended examples, all downloadable from the companion Web site,poodr.info. The first title to focus squarely on object-oriented Ruby application design, Practical Object-Oriented Design in Ruby will guide you to superior outcomes, whatever your previous Ruby experience. Novice Ruby programmers will find specific rules to live by; intermediate Ruby programmers will find valuable principles they can flexibly interpret and apply; and advanced Ruby programmers will find a common language they can use to lead development and guide their colleagues. This guide will help you Understand how object-oriented programming can help you craft Ruby code that is easier to maintain and upgrade Decide what belongs in a single Ruby class Avoid entangling objects that should be kept separate Define flexible interfaces among objects Reduce programming overhead costs with duck typing Successfully apply inheritance Build objects via composition Design cost-effective tests Solve common problems associated with poorly designed Ruby code

A New Perspective on Object-Oriented Design

Practical Process Simulation Using Object-oriented Techniques and C++

Practical Object Oriented Design

Practical API Design

Object-Oriented Programming under Windows

A Practical Guide to Testing Object-oriented Software

Being familiar with object-oriented design is an essential part of programming in Python. This new edition includes all the topics that made Python Object-Oriented Programming an instant Packt classic. Moreover, it's packed with updated content to reflect more recent changes in the core Python libraries and cover modern third-party packages.

An introduction to object-oriented design aimed particularly programmers with little or no design experience. The book looks at the computer programmes using the techniques of object-oriented design, object modelling - Rumbaugh Method, and also features code examples in C++. Emphasis is placed on connections between design and programme code. Design notations and how they provide a suitable vehicle for discussing software architecture are examined. Included are chapter exercises, a complete worked example with implementation and other case studies.

Beginning C# Object-Oriented Programming brings you into the modern world of development as you master the fundamentals of programming with C# and learn to develop efficient, reusable, elegant code through the object-oriented programming (OOP) methodology. Take your skills out of the 20th century and into this one with Dan Clark's accessible, quick-paced guide to C# and object-oriented programming, completely updated for .NET 4.0 and C# 4.0. As you develop techniques and best practices for coding in C#, one of the world's most popular contemporary languages, you'll experience modeling a "real world" application through a case study, allowing you to see how both C# and OOP (a methodology you can use with any number of languages) come together to make your code reusable, modern, and efficient. With more than 30 fully hands-on activities, you'll discover how to transform a simple model of an application into a fully-functional C# project, including designing the user interface, implementing the business logic, and integrating with a relational database for data storage. Along the way, you will explore the .NET Framework, the creation of a Windows-based user interface, a web-based user interface, and service-oriented programming, all using Microsoft's industry-leading Visual Studio 2010, C#, Silverlight, the Entity Framework, and more.

It's easy to write correct Ruby code, but to gain the fluency needed to write great Ruby code, you must go beyond syntax and absorb the "Ruby way" of thinking and problem solving. In Eloquent Ruby, Russ Olsen helps you write Ruby like true Rubyists do–so you can leverage its immense, surprising power. Olsen draws on years of experience internalizing the Ruby culture and teaching Ruby to other programmers. He guides you to the "Ah Ha!" moments when it suddenly becomes clear why Ruby works the way it does, and how you can take advantage of this language's elegance and expressiveness. Eloquent Ruby starts small, answering tactical questions focused on a single statement, method, test, or bug. You'll learn how to write code that actually looks like Ruby (not Java or C#); why Ruby has so many control structures; how to use strings, expressions, and symbols; and what dynamic typing is really good for. Next, the book addresses bigger questions related to building methods and classes. You'll discover why Ruby classes contain so many tiny methods, when to use operator overloading, and when to avoid it. Olsen explains how to write Ruby code that writes its own code–and why you'll want to. He concludes with powerful project-level features and techniques ranging from gems to Domain Specific Languages. A part of the renowned Addison-Wesley Professional Ruby Series, Eloquent Ruby will help you "put on your Ruby-colored glasses" and get results that make you a true believer.

The Art of Objects

The Object-Oriented Thought Process

Elements of Reusable Object-Oriented Software

Object Oriented Analysis and Design Cookbook

ECOOP 2008 - Object-Oriented Programming

Object-Oriented Design with ABAP

Cay Horstmann offers readers an effective means for mastering computing concepts and developing strong design skills. This book introduces object-oriented fundamentals critical to designing software and shows how to implement design techniques. The author's clear, hands-on presentation and outstanding writing style help readers to better understand the material. · A Crash Course in Java· The Object-Oriented Design Process· Guidelines for Class Design· Interface Types and Polymorphism· Patterns and GUI Programming· Inheritance and Abstract Classes· The Java Object Model· Frameworks· Multithreading· More Design Patterns

Object-Oriented Design with UML and Java provides an integrated introduction to object-oriented design with the Unified Modelling Language (UML) and the Java programming language. The book demonstrates how Java applications, no matter how small, can benefit from some design during their construction. Fully road-tested by students on the authors' own courses, the book shows how these complementary technologies can be used effectively to create quality software. It requires no prior knowledge of object orientation, though readers must have some experience of Java or other high level programming language. This book covers object technology; object-oriented analysis and design; and implementation of objects with Java. It includes two case studies dealing with library applications. The UML has been incorporated into a graphical design tool called ROME, which can be downloaded from the book's website. This object modelling environment allows readers to prepare and edit various UML diagrams. ROME can be used alongside a Java compiler to generate Java code from a UML class diagram then compile and run the resulting application for hands-on learning. This text would be a valuable resource for undergraduate students taking courses on O-O analysis and design, O-O modelling, Java programming, and modelling with UML. * Integrates design and implementation, using Java and UML * Includes case studies and exercises * Bridges the gap between programming texts and high level analysis books on design

David A. Sykes is a member of Wofford College's faculty.

Object technology pioneer Wirfs-Brock teams with expert McKean to present a thoroughly updated, modern, and proven method for the design of software. The book is packed with practical design techniques that enable the practitioner to get the job done.

Roles, Responsibilities, and Collaborations

An Agile Primer Using Ruby

An Agile Primer

Eloquent Ruby

Object-Oriented Design with UML and Java

Practical Object-oriented Design with UML

Practical Object Oriented Design deals with the designing of software systems in the 'solution space' using the Unified Modelling Language (UML 2.0). This book builds on the analysis models created in its precursor, Practical Object Oriented Analysis, and iteratively creates architectural and solution models.

Conquer your fear and anxiety learning how the concepts behind object-oriented design apply to the ABAP programming environment. Through simple examples and metaphors this book demystifies the object-oriented programming model. Object-Oriented Design with ABAP presents a bridge from the familiar procedural style of ABAP to the unfamiliar object-oriented style, taking you by the hand and leading you through the difficulties associated with learning these concepts, covering not only the nuances of using object-oriented principles in ABAP software design but also revealing the reasons why these concepts have become embraced throughout the software development industry. More than simply knowing how to use various object-oriented techniques, you'll also be able to determine whether a technique is applicable to the task the software addresses. This book: div Shows how object-oriented principles apply to ABAP program design Provides the basics for creating component design diagrams Teaches how to incorporate design patterns in ABAP programs What You'll Learn Write ABAP code using the object-oriented model as comfortably and easily as using the procedural model Create ABAP design diagrams based on the Unified Modeling Language Implement object-oriented design patterns into ABAP programs Reap the benefits of spending less time designing and maintaining ABAP programs Recognize those situations where design patterns can be most helpful Avoid long and exhausting searches for the cause of bugs in ABAP programs Who This Book Is For Experienced ABAP programmers who remain unfamiliar with the design potential presented by the object-oriented aspect of the language

A Student Guide to Object-Oriented Development is an introductory text that follows the software development process, from requirements capture to implementation, using an object-oriented approach. The book uses object-oriented techniques to present a practical viewpoint on developing software, providing the reader with a basic understanding of object-oriented concepts by developing the subject in an uncomplicated and easy-to-follow manner. It is based on a main worked case study for teaching purposes, plus others with password-protected answers on the web for use in coursework or exams. Readers can benefit from the authors' years of teaching experience. The book outlines standard object-oriented modelling techniques and illustrates them with a variety of examples and exercises, using UML as the modelling language and Java as the language of implementation. It adopts a simple, step by step approach to object-oriented development, and includes case studies, examples, and exercises with solutions to consolidate learning. There are 13 chapters covering a variety of topics such as sequence and collaboration diagrams; state diagrams; activity diagrams; and implementation diagrams. This book is an ideal reference for students taking undergraduate introductory/intermediate computing and information systems courses, as well as business studies courses and conversion masters' programmes. Adopts a simple, step by step approach to object-oriented development Includes case studies, examples, and exercises with solutions to consolidate learning Benefit from the authors' years of teaching experience

The Art of Objects offers an extensive overview of the long-standing principles of object technology, along with leading-edge developments in the field. It will give you a greater understanding of design patterns and the know-how to use them to find effective solutions to a wide range of design challenges. And because the book maintains an approach independent of specific programming languages, the concepts and techniques presented here can be applied to any object-oriented development environment. Using the Unified Modeling Language (UML), The Art of Objects examines numerous static and dynamic practical object design patterns, illustrated by real-life case studies that demonstrate how to put the patterns to work. You will also find discussion of basic concepts of database management and persistent objects, and an introduction to advanced topics in object modeling and interface design patterns. Moving beyond the design level, the book also covers important concepts in object-oriented architecture. Specific topics include: *Object creation and destruction, associations and links, aggregation, inheritance, and other object design fundamentals *UML notation basics for static and dyna

Object Design

The Object-oriented Thought Process

The Well-Grounded Rubyist

Object-oriented Design and Architecture

Python Object-Oriented Programming

Practical Object-oriented Design

In OBJECT THINKING, esteemed object technologist David West contends that the mindset makes the programmer--not the tools and techniques. Delving into the history, philosophy, and even politics of object-oriented programming, West reveals how the best programmers rely on analysis and conceptualization--on thinking--rather than formal process and methods. Both provocative and pragmatic, this book gives form to what's primarily been an oral tradition among the field's revolutionary thinkers--and it illustrates specific object-behavior practices that you can adopt for true object design and superior results. Gain an in-depth understanding of: Prerequisites and principles of object thinking. Object knowledge implicit in eXtreme Programming (XP) and Agile software development. Object conceptualization and modeling. Metaphors, vocabulary, and design for object development. Learn viable techniques for: Decomposing complex domains in terms of objects. Identifying object relationships, interactions, and constraints. Relating object behavior to internal structure and implementation design. Incorporating object thinking into XP and Agile practice.

"One of the great things about the book is the way the authors explain concepts very simply using analogies rather than programming examples–this has been very inspiring for a product I'm working on: an audio-only introduction to OOP and software development." –Bruce Eckel "...I would expect that readers with a basic understanding of object-oriented programming and design would find this book useful, before approaching design patterns completely. Design Patterns Explained complements the existing design patterns texts and may perform a very useful role, fitting between introductory texts such as UML Distilled and the more advanced patterns books." –James Noble Leverage the quality and productivity benefits of patterns–without the complexity! Design Patterns Explained, Second Edition is the field's simplest, clearest, most practical introduction to patterns. Using dozens of updated Java examples, it shows programmers and architects exactly how to use patterns to design, develop, and deliver software far more effectively. You'll start with a complete overview of the fundamental principles of patterns, and the role of object-oriented analysis and design in contemporary software development. Then, using easy-to-understand sample code, Alan Shalloway and James Trott illuminate dozens of today's most useful patterns: their underlying concepts, advantages, tradeoffs, implementation techniques, and pitfalls to avoid. Many patterns are accompanied by UML diagrams. Building on their best-selling First Edition, Shalloway and Trott have thoroughly updated this book to reflect new software design trends, patterns, and implementation techniques. Reflecting extensive reader feedback, they have deepened and clarified coverage throughout, and reorganized content for even greater ease of understanding. New and revamped coverage in this edition includes Better ways to start "thinking in patterns" How design patterns can facilitate agile development using eXtreme Programming and other methods How to use commonality and variability analysis to design application architectures The key role of testing into a patterns-driven development process How to use factories to instantiate and manage objects more effectively The Object-Pool Pattern–a new pattern not identified by the "Gang of Four" New study/practice questions at the end of every chapter Gentle yet thorough, this book assumes no patterns experience whatsoever. It's the ideal "first book" on patterns, and a perfect complement to Gamma's classic Design Patterns. If you're a programmer or architect who wants the clearest possible understanding of design patterns–or if you've struggled to make them work for you–read this book.

The Complete Guide to Writing Maintainable, Manageable, Pleasing, and Powerful Object-Oriented Applications Object-oriented programming languages exist to help you create beautiful, straightforward applications that are easy to change and simple to extend. Unfortunately, the world is awash with object-oriented (OO) applications that are difficult to understand and expensive to change. Practical Object-Oriented Design, Second Edition, immerses you in an OO mindset and teaches you powerful, real-world, object-oriented design techniques with simple and practical examples. Sandi Metz demonstrates how to build new applications that can "survive success" and repair existing applications that have become impossible to change. Each technique is illustrated with extended examples in the easy-to-understand Ruby programming language, all downloadable from the companion website, poodr.com. Fully updated for Ruby 2.5, this guide shows how to Decide what belongs in a single class Avoid entangling objects that should be kept separate Define flexible interfaces among objects Reduce programming overhead costs with duck typing Successfully apply inheritance Build objects via composition Whatever your previous object-oriented experience, this concise guide will help you achieve the superior outcomes you're looking for. Register your book for convenient access to downloads, updates, and/or corrections as they become available. See inside book for details.

Software -- Software Engineering.

An Agile Primer Using Ruby, Second Edition

Theoretical Aspects of Object-oriented Programming

Design Patterns in Ruby (Adobe Reader)

Practical Software Development Using UML and Java

22nd European Conference Paphos, Cyprus, July 7-11, 2008, Proceedings

Object-oriented Software Engineering

A new edition of this title is available, ISBN-10: 0672330164 ISBN-13: 9780672330162 The Object-Oriented Thought Process, Second Edition will lay the foundation in object-oriented concepts and then explain how various object technologies are used. Author Matt Weisfeld introduces object-oriented concepts, then covers abstraction, public and private classes, reusing code, and devloping frameworks. Later chapters cover building objects that work with XML, databases, and distributed systems (including EJBs, .NET, Web Services and more).Throughout the book Matt uses UML, the standard language for modeling objects, to provide illustration and examples of each concept.

This book covers the essential knowledge and skills needed by a student who is specializing in software engineering. Readers will learn principles of object orientation, software development, software modeling, software design, requirements analysis, and testing. The use of the Unified Modelling Language to develop software is taught in depth. Many concepts are illustrated using complete examples, with code written in Java.

Offers a discussion of all the advanced and object-oriented features of C++. Hands-on examples show how features are used in real programming situations. Contains a coding style guide that shows users how to program more effectively and enables them to gain experience with professional style guides. Chapter two provides a crash course which is accessible to programmers in any procedural language.

This text looks at the design of computing programs using the techniques of object-oriented design and UML. It covers modelling techniques and notation in detail, supported by exercises throughout. It is suitable for both Java and C++ users.

The Practical Guide to Effective, Efficient Program Design

UML 2 and the Unified Process

Practical Object-Oriented Analysis and Design

Types, Semantics, and Language Design

A Practical Introduction to Object-Oriented Design with C++

Principles of Object-Oriented Programming in Java 1.1

Intended to help novices and seasoned pros better understand the construction and use of the process interaction approach to discrete-event simulation using object-oriented modeling and programming, this book details both the fundamentals and implementation aspects of simulation modeling using C++. Analysts, software engineers, and programmers faced with the challenge of developing medium to large complex systems will put this book to work in helping them more efficiently design and test systems and alternative concepts.

Do modern programming languages, IDEs, and libraries make coding easy? Maybe, but coding is not design. Large-scale or expensive apps clearly require evaluation of design choices. Still, software design directly impacts code reuse and longevity even for small-scale apps with limited overhead. This text evaluates and contrasts common object-oriented designs. A given problem may have many solutions. A developer may employ different design techniques – composition, inheritance, dependency injection, delegation, etc. – to solve a particular problem. A skilled developer can determine the costs and benefits of different design responses, even amid competing concerns. A responsible developer documents design choices as a contract with the client, delineating external and internal responsibilities. To promote effective software design, this book examines contractual, object-oriented designs for immediate and sustained use as well as code reuse. The intent of identifying design variants is to recognize and manage conflicting goals such as short versus long-term utility, stability versus flexibility, and storage versus computation. Many examples are given to evaluate and contrast different solutions and to compare C# and C++ effects. No one has a crystal ball; however, deliberate design promotes software longevity. With the prominence of legacy OO code, a clear understanding of different object-oriented designs is essential. Design questions abound. Is code reuse better with inheritance or composition? Should composition rely on complete encapsulation? Design choices impact flexibility, efficiency, stability, longevity, and reuse, yet compilers do not enforce design and syntax does not necessarily illustrate design. Through deliberate design, or redesign when refactoring, developers construct sustainable, efficient code.

Clean Code

Object-Oriented Design Choices