Object Oriented Software Construction

TOMMY TESSA

C++ Prentice Hall
This text combines a practical, hands-on approach to programming with the introduction of sound theoretical support focused on teaching the construction of high-quality software. A major feature of the book is the use of Design by Contract.
Effective C++ Prentice Hall PTR
Software -- Software Engineering.
OOP - Learn Object Oriented Thinking & Programming Addison-Wesley Professional
This volume aims to study how practicing software developers, in industrial as well as academic environments, can use object technology to improve the quality of the software they produce. It includes topics on concurrency and Internet programming.
Object-oriented Software Construction McGraw-Hill College
You can find a whole range of programming textbooks intended for complete beginners. However, this one is exceptional to certain extent. The whole textbook is designed as a record of the dialogue of the author with his daughter who wants to learn programming. The author endeavors not to explain the Java programming language to the readers, but to teach them real programming. To teach them how to think and design the program as the experienced programmers do. Entire matter is explained in a very illustrative way which means even a current secondary school student can understand it quite simply.
Adaptive Object-oriented Software Springer Science & Business Media
Software -- Software Engineering.
Class Construction in C and C++ Pearson Deutschland GmbH
David A. Sykes is a member of Wofford College's faculty.
Object-oriented Software Construction Apress
A comprehensive, up-to-date, and resource-filled guide to Eiffel--the only "pure" object-oriented programming language. In addition to describing Eiffel, the book contains descriptions and comparisons of compilers and libraries available on the market as well as other resources for Eiffel programmers, ina ddition to plenty of compiler-independent examples and case studies.
Object Design Brooks/Cole
Design by Contract is a general approach to software design that dramatically improves the quality of the resulting products. This book provides an example-based approach to learning the powerful concept of Design by Contract.
Design by Contract, by Example CRC Press
Effective C++ has been updated to reflect the latest ANSI/ISO standards. The author, a recognised authority on C++, shows readers fifty ways to improve their programs and designs.
Object-oriented Compiler Construction Prentice Hall
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.
Software Construction with Object-oriented Pictures Addison-Wesley Professional
In the demanding world of software development, the object-oriented technique stands out in its potential for software reuse and in its potential to turn the analysis, design and implementation of general software systems into a truly seamless process. This book focuses on Business Object Notation approach and includes case studies, exercises and comprehensive appendices.
Growing Object-Oriented Software, Guided by Tests Elsevier
Covers four main areas: the re-use of software; tools and practices that software developers must use; GUI library utilization; and event-driven systems. Java applets are used to enhance the concept of conceptual material through animation and interaction.
Object-Oriented Analysis and Design Thought-Tools
Object-oriented programming (OOP) has been the leading paradigm for developing software applications for at least 20 years. Many different methodologies, approaches, and techniques have been created for OOP, such as UML, Unified Process, design patterns, and eXtreme Programming. Yet, the actual process of building good software, particularly large, interactive, and long-lived software, is still emerging. Software engineers familiar with the current crop of methodologies are left wondering, how does all of this fit together for designing and building software in real projects? This handbook from one of the world's leading software architects and his team of software engineers presents guidelines on how to develop high-quality software in an application-oriented way. It answers questions such as: * How do we analyze an application domain utilizing the knowledge and experience of the users? * What is the proper software architecture for large, distributed interactive systems that can utilize UML and design patterns? * Where and how should we utilize the techniques and methods of the Unified Process and eXtreme Programming? This book brings together the best of research, development, and day-to-day project work. "The strength of the book is that it focuses on the transition from design to implementation in addition to its overall vision about software development."--Bent Bruun Kristensen, University of Southern Denmark, Odense
Reusability and Software Construction Pearson Education
About the Cover: Although capacity may be a problem for a doghouse, other requirements are usually minimal. Unlike skyscrapers, doghouses are simple units. They do not require plumbing, electricity, fire alarms, elevators, or ventilation systems, and they do not need to be built to code or pass inspections. The range of complexity in software design is similar. Given available software tools and libraries—many of which are free—hobbyists can build small or short-lived computer apps. Yet, design for software longevity, security, and efficiency can be intricate—as is the design of large-scale systems. How can a software developer prepare to manage such complexity? By understanding the essential building blocks of software design and construction. About the Book: Software Essentials: Design and Construction explicitly defines and illustrates the basic elements of software design and construction, providing a solid understanding of control flow, abstract data types (ADTs), memory, type relationships, and dynamic behavior. This text evaluates the benefits and overhead of object-oriented design (OOD) and analyzes software design options. With a structured but hands-on approach, the book: Delineates malleable and stable characteristics of software design Explains how to evaluate the short- and long-term costs and benefits of design decisions Compares and contrasts design solutions, such as composition versus inheritance Includes supportive appendices and a glossary of over 200 common terms Covers key topics such as polymorphism, overloading, and more While extensive examples are given in C# and/or C++, often demonstrating alternative solutions, design—not syntax—remains the focal point of Software Essentials: Design and Construction.
Beginning C# Object-Oriented Programming Springer
This comprehensive volume describes the design and implementation of interpreters and compilers, with specific emphasis on the construction of a Pascal compiler. Author Jim Holmes uses object-oriented analysis and design methods to elucidate the specific compiler components and then gives actual C++ implementation details of these definitions.
Designing Object Systems Springer Science & Business Media
Venturing beyond C++ programming, this text shows how to engineer software products using object-oriented principles. It covers gathering requirements, specifying objects, object verification, defining relations between objects, translating object design into code, object testing, and software

maintenance.

Object-oriented Software Engineering Springer Science & Business Media

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.

Object-oriented System Development "O'Reilly Media, Inc."

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 Software Engineering Pearson Education

8676J-4 Learn to program the way commercial developers do! C++: Effective Object Oriented Software Construction, Second Edition is crafted to help you understand the C++ object-oriented paradigm in depth. It enables you to translate object concepts to practical solutions, no matter what software development environment you encounter. This edition is updated for the new ANSI C++ standard. The book introduces the fundamentals of object-oriented design/programming in the context of real world C++ software development, presenting proven strategies for using C++ to engineer elegant, high-quality software as quickly and efficiently as possible. You'll learn about: Classes, objects, and data abstraction 0bject design techniques and strategies for building efficient and stable architectures The C++ object model, and its cost/benefit implications C++ code style guidelines for projects Tips for writing multi-threaded object-oriented software Single and multiple inheritance, generic programming, and error management In this book, the author reveals the strategies professional developers have learned to maximize code and design reuse. You'll learn how to manage the extensive "housekeeping" that's associated with effective C++ software development. Then, you'll walk through detailed, real-world comparisons of the strengths and weaknesses of the major object-oriented languages. In addition, this book uses UML (Unified Modeling Language) to illustrate its design examples. Whether you're a new programmer, a programmer familiar with procedural languages, or a C++ programmer who isn't using object-oriented techniques to their full potential, C++: Effective Object Oriented Software Construction will help you achieve your most critical goals as a developer.

Object-Oriented Design Choices Tomáš Bruckner

Object-Oriented Software Engineering: An Agile Unified Methodology by David Kung presents a step-by-step methodology that integrates modeling and design, UML, patterns, test-driven development, quality assurance, configuration management, and agile principles throughout the life cycle. The overall approach is casual and easy to follow, with many practical examples that show the theory at work. The author uses his experiences as well as real-world stories to help the reader understand software design principles, patterns, and other software engineering concepts. The book also provides stimulating exercises that go far beyond the type of question that can be answered by simply copying portions of the text.