Object Oriented Design Patterns

Learn everything you need to know about object-oriented programming with the latest features of Kotlin 1.3 Key Features A practical guide to understand objects and classes in Kotlin Learn to write asynchronous, non-blocking codes with Kotlin coroutines Explore Encapsulation, Inheritance, Polymorphism, and Abstraction in Kotlin Book Description Kotlin is an object-oriented programming language. The book is based on the latest version of Kotlin. The book provides you with a thorough understanding of programming concepts, object-oriented programming techniques, and design patterns. It includes numerous examples, explanation of concepts and keynotes. Where possible, examples and programming exercises are included. The main purpose of the book is to provide a comprehensive coverage of Kotlin features such as classes, data classes, and inheritance. It also provides a good understanding of design pattern and how Kotlin syntax works with object-oriented techniques. You will also gain familiarity with syntax in this book by writing labeled for loop and when as an expression. An introduction to the advanced concepts such as sealed classes and package level functions and coroutines is provided and we will also learn how these concepts can make the software development easy. Supported libraries for serialization, regular expression and testing are also covered in this book. By the end of the book, you would have learnt building robust and maintainable software with object oriented design patterns in Kotlin. What you will learn Get an overview of the Kotlin programming language Discover Object-oriented programming techniques in Kotlin Understand Object-oriented design patterns Uncover multithreading by Kotlin way Understand about arrays and collections Understand the importance of object-oriented design patterns Understand about exception handling and testing in OOP with Kotlin Who this book is for This book is for programmers and developers who wish to learn Object-oriented programming principles and apply them to build robust and scalable applications. Basic knowledge in Kotlin programming is assumed

With Pro JavaScript Design Patterns, you'll start with the basics of object-oriented programming in JavaScript applicable to design patterns, including making JavaScript more expressive, inheritance, encapsulation, information hiding, and more. The book then details how to implement and take advantage of several design patterns in JavaScript. Each chapter is packed with real-world examples of how the design patterns are best used and expert advice on writing better code, as well as what to watch out for. Along the way you'll discover how to create your own libraries and APIs for even more efficient coding.

Implement design patterns in .NET using the latest versions of the C# and F# languages. This book provides a comprehensive overview of the field of design patterns as they are used in today's developer toolbox. Using the C#

programming language, Design Patterns in .NET explores the classic design pattern implementation and discusses the applicability and relevance of specific language features for the purpose of implementing patterns. You will learn by example, reviewing scenarios where patterns are applicable. MVP and patterns expert Dmitri Nesteruk demonstrates possible implementations of patterns, discusses alternatives and pattern inter-relationships, and illustrates the way that a dedicated refactoring tool (ReSharper) can be used to implement design patterns with ease. What You'll Learn Know the latest pattern implementations available in C# and F# Refer to researched and proven variations of patterns Study complete, self-contained examples including many that cover advanced scenarios Use the latest implementations of C# and Visual Studio/ReSharper Who This Book Is For Developers who have some experience in the C# language and want to expand their comprehension of the art of programming by leveraging design approaches to solving modern problems
A catalog of solutions to commonly occurring design problems, presenting 23 patterns that allow designers to create flexible and reusable designs for object-oriented software. Describes the circumstances in which each pattern is applicable, and discusses the consequences and trade-offs of using the pattern within a larger design. Patterns are compiled from real systems, and include code for implementation in object-oriented programming languages like C++ and Smalltalk. Includes a bibliography. Annotation copyright by Book News, Inc., Portland, OR
Patterns, Domain-Driven Design (DDD), and Test-Driven Development (TDD) enable architects and developers to create systems that are powerful, robust, and maintainable. Now, there’s a comprehensive, practical guide to leveraging all these techniques primarily in Microsoft .NET environments, but the discussions are just as useful for Java developers. Drawing on seminal work by Martin Fowler (Patterns of Enterprise Application Architecture) and Eric Evans (Domain-Driven Design), Jimmy Nilsson shows how to create real-world architectures for any .NET application. Nilsson illuminates each principle with clear, well-annotated code examples based on C# 1.1 and 2.0. His examples and discussions will be valuable both to C# developers and those working with other .NET languages and any databases–even with other platforms, such as J2EE. Coverage includes · Quick primers on patterns, TDD, and refactoring · Using architectural techniques to improve software quality · Using domain models to support business rules and validation · Applying enterprise patterns to provide persistence support via NHibernate · Planning effectively for the presentation layer and UI testing · Designing for Dependency Injection, Aspect Orientation, and other new paradigms
"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 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

Are you a beginner or an intermediate learner who has been struggling through the maze of OOP and Design Patterns concepts of PHP and you also want to learn the cool new features of PHP 7?PHP 7 - is kind of a revolution in the way that delivers enormous power to everything from websites and mobile to enterprises and the cloud. This is the most important change for PHP since the release of PHP 5 in 2004. It has brought explosive performance improvements, drastically reduced memory consumption, and a host of brand-new language features to make your applications soar. This book makes you ready to take that revolution under your wing.PHP 7 will be definitely faster, and speedier than before but that is not the thing we're going to discuss about in this book.The very first thing is of course speed. It's been found by the developers, around the world, PHP 7 is twice as fast as 5.6 and in some cases it's faster! Since it's tested on widely used php CMS like wordpress and drupal, you can bet on it.Php 7 has touched that dream-point.How we can use that speed?How we can use our object oriented programming concepts to exploit these cool features?How we can use beautiful design patterns to make our php7 application simply 'awesome'?Besides, for absolute beginners I have written a quick recapitulation that will guide you to the core concepts of OOP and Esign patterns.Are you a beginner or an intermediate learner who has been struggling through the maze of OOP and Design Patterns concepts?Then this book is for youContents: Authore(tm)s Note What makes PHP 7 so special? Who should read this book? How to upgrade to PHP 7 Day 1 OOP and PHP 7 A Quick Recapitulation Type Summery To Remember PHP Function More about Functions ... Summery of Simple Classes Namespace, TRAIT and JSON What is Namespace? How to Autoload Namespace? What is Trait? What is JSON? Composer Revolution SOLID Design Principle SOLID stands for What is Liskov Substitution Principle? Interfaces and Method Injection Day 2 Overview of Classes and Objects What you can do? Get, Set and Go... Hiding Information Inheritance, Encapsulation, Abstract Class and Interface Introducing Abstraction and Encapsulation Defining Abstraction Defining encapsulation Difference between Abstraction and Encapsulation Abstract Classes Basic Interfaces Day 3 What is Design Pattern What is Your Strategy? More about Architecture 12.1 -

Compose the Architecture Factory Patterns and Singleton Patterns Day 4 Decorating Applications Responsibility Unchained Adapt SMS into MAIL Day 5 The Template Pattern Relationships between Classes Static Variables, Static Functions and Singleton Pattern Day 6 PHP 7 is Twice Faster! Classes Without Names Know Your Type Can we make it more Strict? Return Type Declarations Day 7 Group 'Use' Declarations Our Spaceship Other few features Closure::call() Integer division with intdiv() Null coalescing operatorEpilogue

Larman covers how to investigate requirements, create solutions and then translate designs into code, showing developers how to make practical use of the most significant recent developments. A summary of UML notation is included

An industry insider explains why there is so much bad software—and why academia doesn't teach programmers what industry wants them to know. Why is software so prone to bugs? So vulnerable to viruses? Why are software products so often delayed, or even canceled? Is software development really hard, or are software developers just not that good at it? In The Problem with Software, Adam Barr examines the proliferation of bad software, explains what causes it, and offers some suggestions on how to improve the situation. For one thing, Barr points out, academia doesn't teach programmers what they actually need to know to do their jobs: how to work in a team to create code that works reliably and can be maintained by somebody other than the original authors. As the size and complexity of commercial software have grown, the gap between academic computer science and industry has widened. It's an open secret that there is little engineering in software engineering, which continues to rely not on codified scientific knowledge but on intuition and experience. Barr, who worked as a programmer for more than twenty years, describes how the industry has evolved, from the era of mainframes and Fortran to today's embrace of the cloud. He explains bugs and why software has so many of them, and why today's interconnected computers offer fertile ground for viruses and worms. The difference between good and bad software can be a single line of code, and Barr includes code to illustrate the consequences of seemingly inconsequential choices by programmers. Looking to the future, Barr writes that the best prospect for improving software engineering is the move to the cloud. When software is a service and not a product, companies will have more incentive to make it good rather than “good enough to ship."

Implement design patterns in .NET Core 3 using the latest versions of the C# and F# languages. This book provides a comprehensive overview of the field of design patterns as they are used in today’s developer toolbox. This new edition introduces topics such as Functional Builder, Asynchronous Factory Method, Generic Value Adapter, and new Composite Proxies, including one that attempts to solve the SoA/AoS problem. Using the C# and F# programming languages, Design Patterns in .NET Core 3 explores the classic design pattern implementations and discusses the applicability and relevance of specific language features for implementing patterns. You will learn by example, reviewing scenarios where patterns are applicable. MVP and patterns expert Dmitri Nesteruk demonstrates possible implementations of patterns, discusses alternatives and pattern inter-relationships, and illustrates the way that a dedicated refactoring tool (ReSharper) can be used to implement design patterns with ease. What You Will Learn Become familiar with the latest pattern implementations available in C# 8 and F# 5 Know how to better reason about software architecture Understand the process of refactoring code to patterns Refer to researched and proven variations of patterns Study complete, self-contained examples, including many that cover advanced scenarios Use the latest implementations of C# and Visual Studio/Rider/ReSharper Who This Book Is For Developers who have some experience in the C# language and want to expand their comprehension of the art of programming by leveraging design approaches to

solving modern problems

"Clojure programming ... This functional programming language not only lets you take advantage of Java libraries, services, and other JVM resources, it rivals other dynamic languages such as Ruby and Python. With this comprehensive guide, you'll learn Clojure fundamentals with examples that relate it to languages you already know"--P. [4] of cover.

A lucid statement of the philosophy of modular programming can be found in a 1970 textbook on the design of system programs by Gouthier and Pont [1, l Cfl0. 23], which we quote below: A well-defined segmentation of the project effort ensures system modularity. Each task fonos a separate, distinct program module. At implementation time each module and its inputs and outputs are well-defined, there is no confusion in the intended interface with other system modules. At checkout time the in tegrity of the module is tested independently; there are few sche duling problems in synchronizing the completion of several tasks before checkout can begin. Finally, the system is maintained in modular fashion; system errors and deficiencies can be traced to specific system modules, thus limiting the scope of detailed error searching. Usually nothing is said about the criteria to be used in dividing the system into modules. This paper will discuss that issue and, by means of examples, suggest some criteria which can be used in decomposing a system into modules. A Brief Status Report The major advancement in the area of modular programming has been the development of coding techniques and assemblers which (1) allow one modu1e to be written with little knowledge of the code in another module, and (2) alJow modules to be reas sembled and replaced without reassembly of the whole system.

Get hands-on experience with each Gang of Four design pattern using C#. For each of the patterns, you'll see at least one real-world scenario, a coding example, and a complete implementation including output. In the first part of Design Patterns in C#, you will cover the 23 Gang of Four (GoF) design patterns, before moving onto some alternative design patterns, including the Simple Factory Pattern, the Null Object Pattern, and the MVC Pattern. The final part winds up with a conclusion and criticisms of design patterns with chapters on anti-patterns and memory leaks. By working through easy-to-follow examples, you will understand the concepts in depth and have a collection of programs to port over to your own projects. Along the way, the author discusses the different creational, structural, and behavioral patterns and why such classifications are useful. In each of these chapters, there is a Q&A session that clears up any doubts and covers the pros and cons of each of these patterns.He finishes the book with FAQs that will help you consolidate your knowledge. This book presents the topic of design patterns in C# in such a way that anyone can grasp the idea. What You Will Learn Work with each of the design patterns Implement the design patterns in real-world applications Select an alternative to these patterns by comparing their pros and cons Use Visual Studio Community Edition 2017 to write code and generate output Who This Book Is For Software developers, software testers, and software architects.

Provides information on analyzing, designing, and writing object-oriented software.

If a proven solution for a recurring problem already exists, why would you reinvent the wheel? This hands-on programming tutorial explains why and how you can use design patterns to help complete your ABAP tasks in less time with better code. Step-by-step, the author guides you through class and interface definitions, as well as the coding for all relevant methods. Plus, benefit immediately from extensively commented real-world code that shows how to implement MVC, Façade, Adapter, Decorator, and more in ABAP Objects.Implementation of Design PatternsFollow the implementation of Singleton, Adapter, Factory, MVC, Façade, Composite, and Decorator in ABAP.Hands-on ApproachWritten for practitioners, the book includes lots of code, detailed UML diagrams, and comprehensive explanations that guarantee

quick success.Real-World Demo ApplicationThe code in this book is not just theory - it's taken from a real-world application that implements all patterns shown in a production environment.Improving CodeThis book helps you improve the robustness and extendibility of your ABAP Objects code, while reducing maintenance efforts.New Coverage of Web Dynpro and the Factory PatternThis second edition has been thoroughly revised and expanded including a new chapter on the Factory pattern, and an extensive section on MVC implementation in multi-technology development for SAP GUI and Web Dynpro ABAP.

Upon completion of an object-oriented design, you are faced with a troubling question: "Is it good, bad, or somewhere in between?" Seasoned experts often answer this question by subjecting the design to a subconscious list of guidelines based on their years of experience. Experienced developer Arthur J. Riel has captured this elusive, subconscious list, and in doing so, has provided a set of metrics that help determine the quality of object-oriented models. Object-Oriented Design Heuristics offers insight into object-oriented design improvement. The more than sixty guidelines presented in this book are language-independent and allow you to rate the integrity of a software design. The heuristics are not written as hard and fast rules; they are meant to serve as warning mechanisms which allow the flexibility of ignoring the heuristic as necessary. This tutorial-based approach, born out of the author's extensive experience developing software, teaching thousands of students, and critiquing designs in a variety of domains, allows you to apply the guidelines in a personalized manner. The heuristics cover important topics ranging from classes and objects (with emphasis on their relationships including association, uses, containment, and both single and multiple inheritance) to physical object-oriented design. You will gain an understanding of the synergy that exists between design heuristics and the popular concept of design patterns; heuristics can highlight a problem in one facet of a design while patterns can provide the solution. Programmers of all levels will find value in this book. The newcomer will discover a fast track to understanding the concepts of object-oriented programming. At the same time, experienced programmers seeking to strengthen their object-oriented development efforts will appreciate the insightful analysis. In short, with Object-Oriented Design Heuristics as your guide, you have the tools to become a better software developer. 020163385XB04062001

Software -- Software Engineering.

A comprehensive guide with extensive coverage on concepts such as OOP, functional programming, generic programming, and STL along with the latest features of C++ Key Features Delve into the core patterns and components of C++ in order to master application design Learn tricks, techniques, and best practices to solve common design and architectural challenges Understand the limitation imposed by C++ and how to solve them using design patterns Book Description C++ is a general-purpose programming language designed with the goals of efficiency, performance, and flexibility in mind. Design patterns are commonly accepted solutions to well-recognized design problems. In essence,

they are a library of reusable components, only for software architecture, and not for a concrete implementation. The focus of this book is on the design patterns that naturally lend themselves to the needs of a C++ programmer, and on the patterns that uniquely benefit from the features of C++, in particular, the generic programming. Armed with the knowledge of these patterns, you will spend less time searching for a solution to a common problem and be familiar with the solutions developed from experience, as well as their advantages and drawbacks. The other use of design patterns is as a concise and an efficient way to communicate. A pattern is a familiar and instantly recognizable solution to specific problem; through its use, sometimes with a single line of code, we can convey a considerable amount of information. The code conveys: "This is the problem we are facing, these are additional considerations that are most important in our case; hence, the following well-known solution was chosen." By the end of this book, you will have gained a comprehensive understanding of design patterns to create robust, reusable, and maintainable code. What you will learn Recognize the most common design patterns used in C++ Understand how to use C++ generic programming to solve common design problems Explore the most powerful C++ idioms, their strengths, and drawbacks Rediscover how to use popular C++ idioms with generic programming Understand the impact of design patterns on the program’s performance Who this book is for This book is for experienced C++ developers and programmers who wish to learn about software design patterns and principles and apply them to create robust, reusable, and easily maintainable apps.

A thoroughly-revised and timely second edition to one of the most successful introductory design patterns books on the market.

The biggest challenge facing many game programmers is completing their game. Most game projects fizzle out, overwhelmed by the complexity of their own code. Game Programming Patterns tackles that exact problem. Based on years of experience in shipped AAA titles, this book collects proven patterns to untangle and optimize your game, organized as independent recipes so you can pick just the patterns you need. You will learn how to write a robust game loop, how to organize your entities using components, and take advantage of the CPUs cache to improve your performance. You'll dive deep into how scripting engines encode behavior, how quadtrees and other spatial partitions optimize your engine, and how other classic design patterns can be used in games.

In this new book, intended as a language companion to the classic Design Patterns , noted Smalltalk and design patterns experts implement the 23 design patterns using Smalltalk code. This approach has produced a language-specific companion that tailors the topic of design patterns to the Smalltalk programmer. The authors have worked closely with the authors of Design Patterns to ensure that this companion volume meets the same quality standards that made the original a bestseller and indispensable resource. The full source code will be available on the AWL web site.

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.

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

Java developers know that design patterns offer powerful productivity benefits but few books have been specific enough to address their programming challenges. With "Java Design Patterns", there's finally a hands-on guide focused specifically on real-world Java development. The book covers three main categories of design patterns--creational, structural, and behavioral--and the example programs and useful variations can be found on the accompanying CD-ROM.

Apply modern C++17 to the implementations of classic design patterns. As well as covering traditional design patterns, this book fleshes out new patterns and approaches that will be useful to C++ developers. The author presents concepts as a fun investigation of how problems can be solved in different ways, along the way using varying degrees of technical sophistication and explaining different sorts of trade-offs. Design Patterns in Modern C++ also provides a technology demo for modern C++, showcasing how some of its latest features (e.g., coroutines) make difficult problems a lot easier to solve. The examples in this book are all suitable for putting into production, with only a few simplifications made in order to aid readability. What You Will Learn Apply design patterns to modern C++ programming Use creational patterns of builder, factories, prototype and singleton Implement structural patterns such as adapter, bridge, decorator, facade and more Work with the behavioral patterns such as chain of responsibility, command, iterator, mediator and more Apply functional design patterns such as Monad and more Who This Book Is For Those with at least some prior programming experience, especially in C++.

Design PatternsElements of Reusable Object-Oriented SoftwarePearson Deutschland GmbH

About The Book: Bruno Preiss presents readers with a modern, object-oriented perspective for looking at data structures and algorithms, clearly showing how to use polymorphism and inheritance, and including fragments from working and tested programs. The book uses a single class hierarchy as a framework to present all of the data structures. This framework clearly shows the relationships between data structures and illustrates how polymorphism and inheritance can be used effectively.

Using research in neurobiology, cognitive science and learning theory, this text loads patterns into your brain in a way that lets you put them to work immediately, makes you better at solving software design problems, and improves your ability to speak the language of patterns with others on your team.

Create various design patterns to master the art of solving problems using Java Key Features This book demonstrates the shift from OOP to functional programming and covers reactive and functional patterns in a clear and step-by-step manner All the design patterns come with a practical use case as part of the explanation, which will improve your productivity Tackle all kinds of performance-related issues and streamline your development Book Description Having a knowledge of design patterns enables

you, as a developer, to improve your code base, promote code reuse, and make the architecture more robust. As languages evolve, new features take time to fully understand before they are adopted en masse. The mission of this book is to ease the adoption of the latest trends and provide good practices for programmers. We focus on showing you the practical aspects of smarter coding in Java. We'll start off by going over object-oriented (OOP) and functional programming (FP) paradigms, moving on to describe the most frequently used design patterns in their classical format and explain how Java's functional programming features are changing them. You will learn to enhance implementations by mixing OOP and FP, and finally get to know about the reactive programming model, where FP and OOP are used in conjunction with a view to writing better code. Gradually, the book will show you the latest trends in architecture, moving from MVC to microservices and serverless architecture. We will finish off by highlighting the new Java features and best practices. By the end of the book, you will be able to efficiently address common problems faced while developing applications and be comfortable working on scalable and maintainable projects of any size. What you will learn Understand the OOP and FP paradigms Explore the traditional Java design patterns Get to know the new functional features of Java See how design patterns are changed and affected by the new features Discover what reactive programming is and why is it the natural augmentation of FP Work with reactive design patterns and find the best ways to solve common problems using them See the latest trends in architecture and the shift from MVC to serverless applications Use best practices when working with the new features Who this book is for This book is for those who are familiar with Java development and want to be in the driver's seat when it comes to modern development techniques. Basic OOP Java programming experience and elementary familiarity with Java is expected.

Now that ActionScript is reengineered from top to bottom as a true object-oriented programming (OOP) language, reusable design patterns are an ideal way to solve common problems in Flash and Flex applications. If you're an experienced Flash or Flex developer ready to tackle sophisticated programming techniques with ActionScript 3.0, this hands-on introduction to design patterns is the book you need. ActionScript 3.0 Design Patterns takes you step by step through the process, first by explaining how design patterns provide a clear road map for structuring code that actually makes OOP languages easier to learn and use. You then learn about various types of design patterns and construct small abstract examples before trying your hand at building full-fledged working applications outlined in the book. Topics in ActionScript 3.0 Design Patterns include: Key features of ActionScript 3.0 and why it became an OOP language OOP characteristics, such as classes, abstraction, inheritance, and polymorphism The benefits of using design patterns Creational patterns, including Factory and Singleton patterns Structural patterns, including Decorator, Adapter, and Composite patterns Behavioral patterns, including Command, Observer, Strategy, and State patterns Multiple design patterns, including Model-View-Controller and Symmetric Proxy designs During the course of the book, you'll work with examples of increasing complexity, such as an e-business application with service options that users can select, an interface for selecting a class of products and individual products in each class, an action game application, a video record and playback application, and many more. Whether you're coming to Flash and Flex from Java or C++, or have experience

with ActionScript 2.0, ActionScript 3.0 Design Patterns will have you constructing truly elegant solutions for your Flash and Flex applications in no time.

Templates are among the most powerful features of C++, but they remain misunderstood and underutilized, even as the C++ language and development community have advanced. In C++ Templates, Second Edition, three pioneering C++ experts show why, when, and how to use modern templates to build software that's cleaner, faster, more efficient, and easier to maintain. Now extensively updated for the C++11, C++14, and C++17 standards, this new edition presents state-of-the-art techniques for a wider spectrum of applications. The authors provide authoritative explanations of all new language features that either improve templates or interact with them, including variadic templates, generic lambdas, class template argument deduction, compile-time if, forwarding references, and user-defined literals. They also deeply delve into fundamental language concepts (like value categories) and fully cover all standard type traits. The book starts with an insightful tutorial on basic concepts and relevant language features. The remainder of the book serves as a comprehensive reference, focusing first on language details and then on coding techniques, advanced applications, and sophisticated idioms. Throughout, examples clearly illustrate abstract concepts and demonstrate best practices for exploiting all that C++ templates can do. Understand exactly how templates behave, and avoid common pitfalls Use templates to write more efficient, flexible, and maintainable software Master today's most effective idioms and techniques Reuse source code without compromising performance or safety Benefit from utilities for generic programming in the C++ Standard Library Preview the upcoming concepts feature The companion website, tmplbook.com, contains sample code and additional updates.

Object-oriented programming is the de facto programming paradigm for many programming languages. Object-Oriented Programming in C# Succinctly provides an introduction to OOP for C# developers. Author Sander Rossel provides overviews and numerous samples to guide readers towards OOP mastery.

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.

Drawing from his extensive experience as a programmer and teacher, author Cay Horstmann helps readers gain an appreciation for the

value of object-oriented design principles. He provides the context so that readers can apply these principles and techniques in their own designs.

With Learning JavaScript Design Patterns, you'll learn how to write beautiful, structured, and maintainable JavaScript by applying classical and modern design patterns to the language. If you want to keep your code efficient, more manageable, and up-to-date with the latest best practices, this book is for you. Explore many popular design patterns, including Modules, Observers, Facades, and Mediators. Learn how modern architectural patterns—such as MVC, MVP, and MVVM—are useful from the perspective of a modern web application developer. This book also walks experienced JavaScript developers through modern module formats, how to namespace code effectively, and other essential topics. Learn the structure of design patterns and how they are written Understand different pattern categories, including creational, structural, and behavioral Walk through more than 20 classical and modern design patterns in JavaScript Use several options for writing modular code—including the Module pattern, Asyncronous Module Definition (AMD), and CommonJS Discover design patterns implemented in the jQuery library Learn popular design patterns for writing maintainable jQuery plug-ins "This book should be in every JavaScript developer's hands. It's the go-to book on JavaScript patterns that will be read and referenced many times in the future."—Andrée Hansson, Lead Front-End Developer, presis!

Build server-side applications more efficiently—and improve your PHP programming skills in the process—by learning how to use design patterns in your code. This book shows you how to apply several object-oriented patterns through simple examples, and demonstrates many of them in full-fledged working applications. Learn how these reusable patterns help you solve complex problems, organize object-oriented code, and revise a big project by only changing small parts. With Learning PHP Design Patterns, you'll learn how to adopt a more sophisticated programming style and dramatically reduce development time. Learn design pattern concepts, including how to select patterns to handle specific problems Get an overview of object-oriented programming concepts such as composition, encapsulation, polymorphism, and inheritance Apply creational design patterns to create pages dynamically, using a factory method instead of direct instantiation Make changes to existing objects or structure without having to change the original code, using structural design patterns Use behavioral patterns to help objects work together to perform tasks Interact with MySQL, using behavioral patterns such as Proxy and Chain of Responsibility Explore ways to use PHP's built-in design pattern interfaces

What will you learn from this book? It's no secret the world around you is becoming more connected, more configurable, more programmable, more computational. You can remain a passive participant, or you can learn to code. With Head First Learn to Code you'll learn how to think computationally and how to write code to make your computer, mobile device, or anything with a CPU do things for you. Using the Python programming language, you'll learn step by step the core concepts of programming as well as many fundamental topics from computer science, such as data structures, storage, abstraction, recursion, and modularity. Why does this book look so different? Based on the latest research in cognitive science and learning theory, Head First Learn to Code uses a visually rich format to engage your mind, rather than a text-heavy approach that puts you to sleep. Why waste your time struggling with new concepts? This multi-sensory learning experience is designed for the way your brain really works.

Capturing a wealth of experience about the design of object-oriented software, four top-notch designers present a catalog of simple and succinct solutions to commonly occurring design problems. Previously undocumented, these 23 patterns allow designers to create more flexible, elegant, and ultimately reusable designs without having to rediscover the design solutions themselves. The authors begin by

describing what patterns are and how they can help you design object-oriented software. They then go on to systematically name, explain, evaluate, and catalog recurring designs in object-oriented systems. With Design Patterns as your guide, you will learn how these important patterns fit into the software development process, and how you can leverage them to solve your own design problems most efficiently. Each pattern describes the circumstances in which it is applicable, when it can be applied in view of other design constraints, and the consequences and trade-offs of using the pattern within a larger design. All patterns are compiled from real systems and are based on real-world examples. Each pattern also includes code that demonstrates how it may be implemented in object-oriented programming languages like C++ or Smalltalk.

Copyright: ef5d06d9c415081cf22fd318d0b4623b