Liskov Substitution Principle C

Implementing Design Patterns in C# and .NET 5

Implement robust applications by applying efficient Design Patterns with .NET 5 and C# KEY FEATURES ? Detailed theoretical concepts covered, including the use of encapsulation, interfaces, and inheritance. ? Access to solutions applied for software strategy and final product output. ? Simplified demonstration of real applications implementing numerous design patterns. DESCRIPTION This book covers detailed aspects of Design Patterns and Object-Oriented Programming concepts using the most modern version of the C# language and .NET platform, including many real-world examples and good practice guidelines that help developers in building robust and extensible applications. The book begins with the essential concepts of C# programming and the .NET platform. You get your foundation strong by understanding SOLID Principles and the actual implementation of reliable applications. You will be working on most common Design Patterns such as Abstract Factory, Adapter, Composite, Proxy, Command, Strategy, Observer, Factory Method, Singleton, Builder, Interpreter, Mediator, and many other patterns that will help you to create solid enterprise applications. You will also witness the performance of these design patterns in a real software development environment with the help of practical examples. After learning the most common Design Patterns practiced in .NET enterprise applications, the reader will be able to understand and apply good practices of software development based on the object-oriented paradigm to develop complex enterprise applications efficiently and simply. WHAT YOU WILL LEARN ? Fine-tune your knowledge about interfaces, polymorphism, and encapsulation. ? Learn to practice implementing design patterns in enterprise applications. ? Implement rich design patterns: Observer, Strategy, Command, Proxy, and more. ? Get to learn the latest additional design patterns such as Builder, Bridge, and Decorator. ? Includes illustrations, examples, and real use-cases of .NET 5.0 applications. WHO THIS BOOK IS FOR This book is for .NET developers, application developers, and software engineers who want to develop .NET applications with proven techniques and build error-free applications. This book also attracts fresh graduates and entry-level developers as long as basic knowledge about .NET is known to them. TABLE OF CONTENTS 1. C# Fundamentals 2. Introduction to .NET 5 3. Basic Concepts of Object-Oriented Programming 4. Interfaces in C# 5. Encapsulation and Polymorphism in C# 6. SOLID Principles in C# 7. Abstract Factory 8. Abstract Factory 9. Prototype 10. Factory Method 11. Adapter 12. Composite 13. Proxy 14. Command 15. Strategy 16. Observer 17. Good Practices and Additional Design Patterns

Agile Principles, Patterns, and Practices in C#

With the award-winning book Agile Software Development: Principles, Patterns, and Practices, Robert C. Martin helped bring Agile principles to tens of thousands of Java and C++ programmers. Now .NET programmers have a definitive guide to agile methods with this completely updated volume from Robert C. Martin and Micah Martin, Agile Principles, Patterns, and Practices in C#. This book presents a series of case studies illustrating the fundamentals of Agile development and Agile design, and moves quickly from UML models to real C# code. The introductory chapters lay out the basics of the agile movement, while the later chapters show proven techniques in action. The book includes many source code examples that are also available for download from the authors’ Web site. Readers will come away from this book understanding Agile principles, and the fourteen practices of Extreme Programming Spiking, splitting, velocity, and planning iterations and releases Test-driven development, test-first design, and acceptance testing Refactoring with unit testing Pair programming Agile design and design smells The five types of UML diagrams and how to use them effectively Object-oriented package design and design patterns How to put all of it together for a real-world project Whether you are a C# programmer or a Visual Basic or Java programmer learning C#, a software development manager, or a business analyst, Agile Principles, Patterns, and Practices in C# is the first book you should read to understand agile software and how it applies to programming in the .NET Framework.

Design Patterns and Best Practices in Java

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.

Test-Driven Development for Embedded C

This collection of articles by well-known experts was originally published in 2000 and is intended for researchers in computer science, practitioners of formal methods, and computer programmers working in safety-critical applications or in the technology of component-based systems. The work brings together several elements of this area that were fast becoming the focus of much research and practice in computing. The introduction by Clemens Szyperski gives a snapshot of research in the field. About half the articles deal with theoretical frameworks, models, and systems of notation; the rest of the book concentrates on case studies by researchers who have built prototype systems and present findings on architectures verification. The emphasis is on advances in the technological infrastructure of component-based systems; how to design and specify reusable components; and how to reason about, verify, and validate systems from components. Thus the book shows how theory might move into practice.

Foundations of Component-Based Systems

Write code that can adapt to changes. By applying this book’s principles, you can create code that accommodates new requirements and unforeseen scenarios without significant rewrites. Gary McLean Hall describes Agile best practices, principles, and patterns for designing and writing code that can evolve more quickly and easily, with fewer errors, because it doesn’t impede change. Now revised, updated, and expanded, Adaptive Code, Second Edition adds indispensable practical insights on Kanban, dependency inversion, and creating reusable abstractions. Drawing on over a decade of Agile consulting and development experience, McLean Hall has updated his best-seller with deeper coverage of unit testing, refactoring, pure dependency injection, and more. Master powerful new ways to: • Write code that enables and complements Scrum, Kanban, or any other Agile framework • Develop code that can survive major changes in requirements • Plan for adaptability by using dependencies, layering, interfaces, and design patterns • Perform unit testing and refactoring in tandem, gaining more value from both • Use the “golden master” technique to make legacy code adaptive • Build SOLID code with single-responsibility, open/closed, and Liskov substitution principles • Create smaller interfaces to support more-diverse client and architectural needs • Leverage dependency injection best practices to improve code adaptability • Apply dependency inversion with the Stairway pattern, and avoid related anti-patterns About You This book is for programmers of all skill levels seeking more-practical insight into design patterns, SOLID principles, unit testing, refactoring, and related topics. Most readers will have programmed in C#, Java, C++, or similar object-oriented languages, and will be familiar with core procedural programming techniques.

Adaptive Code

\"Your process may be agile, but are you building agility directly into the code base? This book teaches .NET programmers how to give code the flexibility to adapt to changing requirements and customer demands by applying cutting-edge techniques, including SOLID principles, design patterns, and other industry best practices. Understand why composition is preferable to inheritance and how flexible the interface really can be; gain deep knowledge of key design patterns and anti-patterns, when to apply them, and how to give their code agility; bridge the gap between the theory behind SOLID principles, design patterns, and industry best practices by pragmatically solving real-world problems; get code samples written in upcoming version of Microsoft Visual C#. Topics include: Agile with Scrum process; dependencies and layering; the interface; patterns and anti-patterns; introduction to SOLID principles, including open/closed and dependency interjection; and using application templates\"--Publisher's description.

Adaptive Code Via C#

What is this Book About? At the beginning of the 21st century, computer systems—and especially so- ware—play an important role in our society. Software is contained in virtually every technical device that we use in everyday life (e.g., cellular phones and cars). Furthermore, computers and their software are used for leisure purposes at home (the Internet and computer games), at the office (e.g., writing letters and order processing), and for more complicated tasks such as controlling steel plants or insuring flight safety. Therefore, the quality of software (e.g., its correctness, re- ability, and efficiency) has become important not only in the context of critical systems (e.g., nuclear power plants) but also for our entire society, from business to leisure. Software engineering is the practical application of scientific knowledge for the economical production and use of high-quality software [Pomberger96]. The discipline aims at developing methods, techniques, tools, and standards to fulfill these aims. The number of methods and tools available to the software engineer nowadays is overwhelming; nevertheless, many software projects fail—that is, do not meet their schedules, are over budget, do not meet the user needs, or simply have considerable quality defects. The numerous possible explanations for this situation include poor project management, unsuitable methods and tools used in the project, and poorly developed skills of the participating software engineers.

Contracts, Scenarios and Prototypes

Learn the importance of architectural and design patterns in producing and sustaining next-generation IT and business-critical applications with this guide. About This Book Use patterns to tackle communication, integration, application structure, and more Implement modern design patterns such as microservices to build resilient and highly available applications Choose between the MVP, MVC, and MVVM patterns depending on the application being built Who This Book Is For This book will empower and enrich IT architects (such as enterprise architects, software product architects, and solution and system architects), technical consultants, evangelists, and experts. What You Will Learn Understand how several architectural and design patterns work to systematically develop multitier web, mobile, embedded, and cloud applications Learn object-oriented and component-based software engineering principles and patterns Explore the frameworks corresponding to various architectural patterns Implement domain-driven, test-driven, and behavior-driven methodologies Deploy key platforms and tools effectively to enable EA design and solutioning Implement various patterns designed for the cloud paradigm In Detail Enterprise Architecture (EA) is typically an aggregate of the business, application, data, and infrastructure architectures of any forward-looking enterprise. Due to constant changes and rising complexities in the business and technology landscapes, producing sophisticated architectures is on the rise. Architectural patterns are gaining a lot of attention these days. The book is divided in three modules. You'll learn about the patterns associated with object-oriented, component-based, client-server, and cloud architectures. The second module covers Enterprise Application Integration (EAI) patterns and how they are architected using various tools and patterns. You will come across patterns for Service-Oriented Architecture (SOA), Event-Driven Architecture (EDA), Resource-Oriented Architecture (ROA), big data analytics architecture, and Microservices Architecture (MSA). The final module talks about advanced topics such as Docker containers, high performance, and reliable application architectures. The key takeaways include understanding what architectures are, why they're used, and how and where architecture, design, and integration patterns are being leveraged to build better and bigger systems. Style and Approach This book adopts a hands-on approach with real-world examples and use cases.

Architectural Patterns

This book will teach the concepts of test driven development in Java so you can build clean, maintainable and robust code Key Features Explore the most popular TDD tools and frameworks and become more proficient in building applications Create applications with better code design, fewer bugs, and higher test coverage, enabling you to get them to market quickly Implement test-driven programming methods into your development workflows Book Description Test-driven development (TDD) is a development approach that relies on a test-first procedure that emphasizes writing a test before writing the necessary code, and then refactoring the code to optimize it.The value of performing TDD with Java, one of the longest established programming languages, is to improve the productivity of programmers and the maintainability and performance of code, and develop a deeper understanding of the language and how to employ it effectively. Starting with the basics of TDD and understanding why its adoption is beneficial, this book will take you from the first steps of TDD with Java until you are confident enough to embrace the practice in your day-to-day routine.You'll be guided through setting up tools, frameworks, and the environment you need, and we will dive right into hands-on exercises with the goal of mastering one practice, tool, or framework at a time. You'll learn about the Red-Green-Refactor procedure, how to write unit tests, and how to use them as executable documentation.With this book, you'll also discover how to design simple and easily maintainable code, work with mocks, utilize behavior-driven development, refactor old legacy code, and release a half-finished feature to production with feature toggles.You will finish this book with a deep understanding of the test-driven development methodology and the confidence to apply it to application programming with Java. What you will learn Explore the tools and frameworks required for effective TDD development Perform the Red-Green-Refactor process efficiently, the pillar around which all other TDD procedures are based Master effective unit testing in isolation from the rest of your code Design simple and easily maintainable code by implementing different techniques Use mocking frameworks and techniques to easily write and quickly execute tests Develop an application to implement behavior-driven development in conjunction with unit testing Enable and disable features using feature toggles Who this book is for If you're an experienced Java developer and want to implement more effective methods of programming systems and applications, then this book is for you.

Test-Driven Java Development, Second Edition

Getting the most out of Python to improve your codebase Key Features Save maintenance costs by learning to fix your legacy codebase Learn the principles and techniques of refactoring Apply microservices to your legacy systems by implementing practical techniques Book Description Python is currently used in many different areas such as software construction, systems administration, and data processing. In all of these areas, experienced professionals can find examples of inefficiency, problems, and other perils, as a result of bad code. After reading this book, readers will understand these problems, and more importantly, how to correct them. The book begins by describing the basic elements of writing clean code and how it plays an important role in Python programming. You will learn about writing efficient and readable code using the Python standard library and best practices for software design. You will learn to implement the SOLID principles in Python and use decorators to improve your code. The book delves more deeply into object oriented programming in Python and shows you how to use objects with descriptors and generators. It will also show you the design principles of software testing and how to resolve software problems by implementing design patterns in your code. In the final chapter we break down a monolithic application to a microservice one, starting from the code as the basis for a solid platform. By the end of the book, you will be proficient in applying industry approved coding practices to design clean, sustainable and readable Python code. What you will learn Set up tools to effectively work in a development environment Explore how the magic methods of Python can help us write better code Examine the traits of Python to create advanced object-oriented design Understand removal of duplicated code using decorators and descriptors Effectively refactor code with the help of unit tests Learn to implement the SOLID principles in Python Who this book is for This book will appeal to team leads, software architects and senior software engineers who would like to work on their legacy systems to save cost and improve efficiency. A strong understanding of Programming is assumed.

Clean Code in Python

Section 1 Agile development Section 2 Agile design Section 3 The payroll case study Section 4 Packaging the payroll system Section 5 The weather station case study Section 6 The ETS case study

Agile Software Development

”Demystifies object-oriented programming, and lays out how to use it to design truly secure and performant applications.” —Charles Soetan, Plum.io Key Features Dozens of techniques for writing object-oriented code that’s easy to read, reuse, and maintain Write code that other programmers will instantly understand Design rules for constructing objects, changing and exposing state, and more Examples written in an instantly familiar pseudocode that’s easy to apply to Java, Python, C#, and any object-oriented language Purchase of the print book includes a free eBook in PDF, Kindle, and ePub formats from Manning Publications. About The Book Well-written object-oriented code is easy to read, modify, and debug. Elevate your coding style by mastering the universal best practices for object design presented in this book. These clearly presented rules, which apply to any OO language, maximize the clarity and durability of your codebase and increase productivity for you and your team. In Object Design Style Guide, veteran developer Matthias Noback lays out design rules for constructing objects, defining methods, and much more. All examples use instantly familiar pseudocode, so you can follow along in the language you prefer. You’ll go case by case through important scenarios and challenges for object design and then walk through a simple web application that demonstrates how different types of objects can work together effectively. What You Will Learn Universal design rules for a wide range of objects Best practices for testing objects A catalog of common object types Changing and exposing state Test your object design skills with exercises This Book Is Written For For readers familiar with an object-oriented language and basic application architecture. About the Author Matthias Noback is a professional web developer with nearly two decades of experience. He runs his own web development, training, and consultancy company called “Noback’s Office.” Table of Contents: 1 ¦ Programming with objects: A primer 2 ¦ Creating services 3 ¦ Creating other objects 4 ¦ Manipulating objects 5 ¦ Using objects 6 ¦ Retrieving information 7 ¦ Performing tasks 8 ¦ Dividing responsibilities 9 ¦ Changing the behavior of services 10 ¦ A field guide to objects 11 ¦ Epilogue

Object Design Style Guide

Utilize the power of modular programming to improve code readability, maintainability, and testability About This Book This book demonstrates code reusability and distributed development to get high speed, maintainable, and fast applications It illustrates the development of a complete modular application developed using PHP7 in detail This book provides a high-level overview of the Symfony framework, a set of tools and a development methodology that are needed to build a modular web shop application Who This Book Is For This step-by-step guide is divided into two sections. The first section explores all the fundamentals of modular design technique with respect to PHP 7. The latter section demonstrates the practical development of individual modules of a web shop application. What You Will Learn Discover the new features of PHP 7 that are relevant to modular application development Write manageable code based on the GoF design patterns and SOLID principles Define the application requirements of a working modular application Explore the ins and outs of the Symfony framework Build a set of modules based on the Symfony framework that comprise a simple web shop app Use core modules to set the structure and dependencies for other modules to use Set up entities that are relevant to the module functionality and see how to manage these entities In Detail Modular design techniques help you build readable, manageable, reusable, and more efficient codes. PHP 7, which is a popular open source scripting language, is used to build modular functions for your software. With this book, you will gain a deep insight into the modular programming paradigm and how to achieve modularity in your PHP code. We start with a brief introduction to the new features of PHP 7, some of which open a door to new concepts used in modular development. With design patterns being at the heart of all modular PHP code, you will learn about the GoF design patterns and how to apply them. You will see how to write code that is easy to maintain and extend over time with the help of the SOLID design principles. Throughout the rest of the book, you will build different working modules of a modern web shop application using the Symfony framework, which will give you a deep understanding of modular application development using PHP 7. Style and approach This book is for intermediate-level PHP developers with little to no knowledge of modular programming who want to understand design patterns and principles in order to better utilize the existing frameworks for modular application development.

Modular Programming with PHP 7

Publisher's Note: Microsoft stops supporting .NET Core 3.1 in December 2022. The newer 7th edition of this book is available that covers .NET 7 (end-of-life May 2024) or .NET 6 (end-of-life November 2024), with C# 11 and EF Core 7. Key FeaturesBuild modern, cross-platform applications with .NET Core 3.0Get up to speed with C#, and up to date with all the latest features of C# 8.0Start creating professional web applications with ASP.NET Core 3.0Book Description In C# 8.0 and .NET Core 3.0 – Modern Cross-Platform Development, Fourth Edition, expert teacher Mark J. Price gives you everything you need to start programming C# applications. This latest edition uses the popular Visual Studio Code editor to work across all major operating systems. It is fully updated and expanded with new chapters on Content Management Systems (CMS) and machine learning with ML.NET. The book covers all the topics you need. Part 1 teaches the fundamentals of C#, including object-oriented programming, and new C# 8.0 features such as nullable reference types, simplified switch pattern matching, and default interface methods. Part 2 covers the .NET Standard APIs, such as managing and querying data, monitoring and improving performance, working with the filesystem, async streams, serialization, and encryption. Part 3 provides examples of cross-platform applications you can build and deploy, such as web apps using ASP.NET Core or mobile apps using Xamarin.Forms. The book introduces three technologies for building Windows desktop applications including Windows Forms, Windows Presentation Foundation (WPF), and Universal Windows Platform (UWP) apps, as well as web applications, web services, and mobile apps. What you will learnBuild cross-platform applications for Windows, macOS, Linux, iOS, and AndroidExplore application development with C# 8.0 and .NET Core 3.0Explore ASP.NET Core 3.0 and create professional web applicationsLearn object-oriented programming and C# multitaskingQuery and manipulate data using LINQUse Entity Framework Core and work with relational databasesDiscover Windows app development using the Universal Windows Platform and XAMLBuild mobile applications for iOS and Android using Xamarin.FormsWho this book is for Readers with some prior programming experience or with a science, technology, engineering, or mathematics (STEM) background, who want to gain a solid foundation with C# 8.0 and .NET Core 3.0.

C# 8.0 and .NET Core 3.0 – Modern Cross-Platform Development

Liskov (engineering, Massachusetts Institute of Technology) and Guttag (computer science and engineering, also at MIT) present a component- based methodology for software program development. The book focuses on modular program construction: how to get the modules right and how to organize a program as a collection of modules. It explains the key types of abstractions, demonstrates how to develop specifications that define these abstractions, and illustrates how to implement them using numerous examples. An introduction to key Java concepts is included. Annotation copyrighted by Book News, Inc., Portland, OR.

Program Development in Java

With its support for dynamic programming, C# 4.0 continues to evolve as a versatile language on its own. But when C# is used with .NET Framework 4, the combination is incredibly powerful. This bestselling tutorial shows you how to build web, desktop, and rich Internet applications using C# 4.0 with .NET's database capabilities, UI framework (WPF), extensive communication services (WCF), and more. In this sixth edition, .NET experts Ian Griffiths, Matthew Adams, and Jesse Liberty cover the latest enhancements to C#, as well as the fundamentals of both the language and framework. You'll learn concurrent programming with C# 4.0, and how to use .NET tools such as the Entity Framework for easier data access, and the Silverlight platform for browser-based RIA development. Learn C# fundamentals, such as variables, flow control, loops, and methods Build complex programs with object-oriented and functional programming techniques Process large collections of data with the native query features in LINQ Communicate across networks with Windows Communication Foundation (WCF) Learn the advantages of C# 4.0's dynamic language features Build interactive Windows applications with Windows Presentation Foundation (WPF) Create rich web applications with Silverlight and ASP.NET

Programming C# 4.0

Develop your programming skills by exploring essential topics such as code reviews, implementing TDD and BDD, and designing APIs to overcome code inefficiency, redundancy, and other problems arising from bad code Key FeaturesWrite code that cleanly integrates with other systems while maintaining well-defined software boundariesUnderstand how coding principles and standards enhance software qualityLearn how to avoid common errors while implementing concurrency or threadingBook Description Traditionally associated with developing Windows desktop applications and games, C# is now used in a wide variety of domains, such as web and cloud apps, and has become increasingly popular for mobile development. Despite its extensive coding features, professionals experience problems related to efficiency, scalability, and maintainability because of bad code. Clean Code in C# will help you identify these problems and solve them using coding best practices. The book starts with a comparison of good and bad code, helping you understand the importance of coding standards, principles, and methodologies. You’ll then get to grips with code reviews and their role in improving your code while ensuring that you adhere to industry-recognized coding standards. This C# book covers unit testing, delves into test-driven development, and addresses cross-cutting concerns. You’ll explore good programming practices for objects, data structures, exception handling, and other aspects of writing C# computer programs. Once you’ve studied API design and discovered tools for improving code quality, you’ll look at examples of bad code and understand which coding practices you should avoid. By the end of this clean code book, you’ll have the developed skills you need in order to apply industry-approved coding practices to write clean, readable, extendable, and maintainable C# code. What you will learnWrite code that allows software to be modified and adapted over timeImplement the fail-pass-refactor methodology using a sample C# console applicationAddress cross-cutting concerns with the help of software design patternsWrite custom C# exceptions that provide meaningful informationIdentify poor quality C# code that needs to be refactoredSecure APIs with API keys and protect data using Azure Key VaultImprove your code’s performance by using tools for profiling and refactoringWho this book is for This coding book is for C# developers, team leads, senior software engineers, and software architects who want to improve the efficiency of their legacy systems. A strong understanding of C# programming is required.

Clean Code in C#

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.

Beginning C# Object-Oriented Programming

Apply design principles to your classes, preparing them for reuse. You will use package design principles to create packages that are just right in terms of cohesion and coupling, and are user- and maintainer-friendly at the same time. The first part of this book walks you through the five SOLID principles that will help you improve the design of your classes. The second part introduces you to the best practices of package design, and covers both package cohesion principles and package coupling principles. Cohesion principles show you which classes should be put together in a package, when to split packages, and if a combination of classes may be considered a \"package\" in the first place. Package coupling principles help you choose the right dependencies and prevent wrong directions in the dependency graph of your packages. What You'll Learn Apply the SOLID principles of class design Determine if classes belong in the same package Know whether it is safe for packages to depend on each other Who This Book Is For Software developers with a broad range of experience in the field, who are looking for ways to reuse,share, and distribute their code

Principles of Package Design

Apply business requirements to IT infrastructure and deliver a high-quality product by understanding architectures such as microservices, DevOps, and cloud-native using modern C++ standards and features Key FeaturesDesign scalable large-scale applications with the C++ programming languageArchitect software solutions in a cloud-based environment with continuous integration and continuous delivery (CI/CD)Achieve architectural goals by leveraging design patterns, language features, and useful toolsBook Description Software architecture refers to the high-level design of complex applications. It is evolving just like the languages we use, but there are architectural concepts and patterns that you can learn to write high-performance apps in a high-level language without sacrificing readability and maintainability. If you're working with modern C++, this practical guide will help you put your knowledge to work and design distributed, large-scale apps. You'll start by getting up to speed with architectural concepts, including established patterns and rising trends, then move on to understanding what software architecture actually is and start exploring its components. Next, you'll discover the design concepts involved in application architecture and the patterns in software development, before going on to learn how to build, package, integrate, and deploy your components. In the concluding chapters, you'll explore different architectural qualities, such as maintainability, reusability, testability, performance, scalability, and security. Finally, you will get an overview of distributed systems, such as service-oriented architecture, microservices, and cloud-native, and understand how to apply them in application development. By the end of this book, you'll be able to build distributed services using modern C++ and associated tools to deliver solutions as per your clients' requirements. What you will learnUnderstand how to apply the principles of software architectureApply design patterns and best practices to meet your architectural goalsWrite elegant, safe, and performant code using the latest C++ featuresBuild applications that are easy to maintain and deployExplore the different architectural approaches and learn to apply them as per your requirementSimplify development and operations using application containersDiscover various techniques to solve common problems in software design and developmentWho this book is for This software architecture C++ programming book is for experienced C++ developers looking to become software architects or develop enterprise-grade applications.

Software Architecture with C++

Practical Software Architecture Solutions from the Legendary Robert C. Martin (“Uncle Bob”) By applying universal rules of software architecture, you can dramatically improve developer productivity throughout the life of any software system. Now, building upon the success of his best-selling books Clean Code and The Clean Coder, legendary software craftsman Robert C. Martin (“Uncle Bob”) reveals those rules and helps you apply them. Martin’s Clean Architecture doesn’t merely present options. Drawing on over a half-century of experience in software environments of every imaginable type, Martin tells you what choices to make and why they are critical to your success. As you’ve come to expect from Uncle Bob, this book is packed with direct, no-nonsense solutions for the real challenges you’ll face–the ones that will make or break your projects. Learn what software architects need to achieve–and core disciplines and practices for achieving it Master essential software design principles for addressing function, component separation, and data management See how programming paradigms impose discipline by restricting what developers can do Understand what’s critically important and what’s merely a “detail” Implement optimal, high-level structures for web, database, thick-client, console, and embedded applications Define appropriate boundaries and layers, and organize components and services See why designs and architectures go wrong, and how to prevent (or fix) these failures Clean Architecture is essential reading for every current or aspiring software architect, systems analyst, system designer, and software manager–and for every programmer who must execute someone else’s designs. Register your product for convenient access to downloads, updates, and/or corrections as they become available.

Clean Architecture

'Downright revolutionary... the title is a major understatement... 'Quantum Programming' may ultimately change the way embedded software is designed.' -- Michael Barr, Editor-in-Chief, Embedded Systems Programming magazine (Click here

Practical Statecharts in C/C++

This title shows the process of cleaning code. Rather than just illustrating the end result, or just the starting and ending state, the author shows how several dozen seemingly small code changes can positively impact the performance and maintainability of an application code base.

Clean Code

How to Reduce Code Complexity and Develop Software More Sustainably \"Mark Seemann is well known for explaining complex concepts clearly and thoroughly. In this book he condenses his wide-ranging software development experience into a set of practical, pragmatic techniques for writing sustainable and human-friendly code. This book will be a must-read for every programmer.\" -- Scott Wlaschin, author of Domain Modeling Made Functional Code That Fits in Your Head offers indispensable, practical advice for writing code at a sustainable pace and controlling the complexity that causes projects to spin out of control. Reflecting decades of experience helping software teams succeed, Mark Seemann guides you from zero (no code) to deployed features and shows how to maintain a good cruising speed as you add functionality, address cross-cutting concerns, troubleshoot, and optimize. You'll find valuable ideas, practices, and processes for key issues ranging from checklists to teamwork, encapsulation to decomposition, API design to unit testing. Seemann illuminates his insights with code examples drawn from a complete sample project. Written in C#, they're designed to be clear and useful to anyone who uses any object-oriented language including Java , C++, and Python. To facilitate deeper exploration, all code and extensive commit messages are available for download. Choose mindsets and processes that work, and escape bad metaphors that don't Use checklists to liberate yourself, improving outcomes with the skills you already have Get past “analysis paralysis” by creating and deploying a vertical slice of your application Counteract forces that lead to code rot and unnecessary complexity Master better techniques for changing code behavior Discover ways to solve code problems more quickly and effectively Think more productively about performance and security If you've ever suffered through bad projects or had to cope with unmaintainable legacy code, this guide will help you make things better next time and every time. Register your book for convenient access to downloads, updates, and/or corrections as they become available. See inside book for details.

Code That Fits in Your Head

Exploit the features of TypeScript to develop and maintain captivating web applications with easeAbout This Book- Learn how to develop modular, scalable, maintainable, and adaptable web applications by taking advantage of TypeScript- Create object-oriented JavaScript that adheres to the solid principles efficiently- A comprehensive guide that explains the fundamentals of TypeScript with the help of practical examplesWho This Book Is ForIf you are a JavaScript developer aiming to learn TypeScript to build beautiful web applications, then this book is for you. No prior knowledge of TypeScript is required.What You Will Learn- Learn the key TypeScript language features and language runtime- Develop modular, scalable, maintainable, and adaptable web applications- Create object-oriented code that adheres to the solid principles- Save time using automation tools like Gulp and Karma- Develop robust applications with testing (Mocha, Chai and SinonJS)- Put your TypeScript skills in practice by developing a single-page web application framework from scratch- Use the JavaScript of tomorrow (ES6 and ES7) today with TypeScriptIn DetailTypeScript is an open source and cross-platform typed superset of JavaScript that compiles to plain JavaScript that runs in any browser or any host. It allows developers to use the future versions of JavaScript (ECMAScript 6 and 7) today. TypeScript adds optional static types, classes, and modules to JavaScript, to enable great tooling and better structuring of large JavaScript applications.This book is a step-by-step guide that will get you started with TypeScript with the help of practical examples. You start off by understanding the basics of TypeScript. Next, automation tools like Grunt are explained followed by a detailed description of function, generics, callbacks and promises. After this, object-oriented features and the memory management functionality of TypeScript are explained. At the end of this book, you will have learned enough to implement all the concepts and build a single page application from scratch.Style and approachThis is a step-by-step guide that covers the fundamentals of TypeScript with practical examples. Each chapter introduces a set of TypeScript language features and leads the readers toward the development of a real-world application.

Learning Typescript

Learn the fundamentals of Object-Oriented design by investigating good—and bad—code! Well-designed applications run more efficiently, have fewer bugs, and are easier to revise and maintain. Using an engaging “before-and-after” approach, Object-Oriented Software Design in C++ shows you exactly what bad software looks like and how to fix it with good design principles and patterns. In Object-Oriented Software Design in C++, you’ll find: Design-code-test iterations that improve code with each revision Gathering requirements to make sure you’re developing the right application Design principles like encapsulation and delegation that solve programming problems Design patterns including Observer Design Pattern that fix architecture issues Using recursion and multithreading to simplify common solutions Object-Oriented Software Design in C++ is a vital guide to building the kind of high performance applications delivered by the pros—all using industry-proven design principles and patterns. You’ll learn how to gather and analyze requirements so you’re building exactly what your client is looking for, backtrack mistakes with iterative development, and build a toolbox of design patterns that troubleshoot common issues with application architecture. The book’s accessible examples are written in C++ 17, but its universal principles can be applied to any object-oriented language. Purchase of the print book includes a free eBook in PDF and ePub formats from Manning Publications. About the technology Good design is the foundation of great software. Mastering the principles of object-oriented design is the surest way to create applications that run fast, have few bugs, and last well into the future. Written especially for new C++ programmers, this easy-to-read book gently mentors you in the art of designing great software. About the book Object-Oriented Software Design in C++ introduces object-oriented design principles, practices, and patterns in clear, jargon-free language. The instantly-familiar before-and-after examples highlight the benefits of good design. Each chapter is full of friendly conversations that anticipate your questions and help point out the subtleties you might overlook. Along the way, you’ll pick up tips about idiomatic C++ style that will set your code apart. What's inside Design-code-test iterations Design principles for common programming problems Architecture design patterns in plain English Recursion and multithreading About the reader Examples are in C++ 17. About the author Ronald Mak is a former NASA senior scientist. Currently, he teaches computer science at San Jose State University. The technical editor on this book was Juan Rufes. Table of Contents PART 1 1 The path to well-designed software 2 Iterate to achieve good design PART 2 3 Get requirements to build the right application 4 Good class design to build the application right PART 3 5 Hide class implementations 6 Don’t surprise your users 7 Design subclasses right PART 4 8 The Template Method and Strategy Design Patterns 9 The Factory Method and Abstract Factory Design Patterns 10 The Adapter and Façade Design Patterns 11 The Iterator and Visitor Design Patterns 12 The Observer Design Pattern 13 The State Design Pattern 14 The Singleton, Composite, and Decorator Design Patterns PART 5 15 Designing solutions with recursion and backtracking 16 Designing multithreaded programs

Object-Oriented Software Design in C++

Get more out of your legacy systems: more performance, functionality, reliability, and manageability Is your code easy to change? Can you get nearly instantaneous feedback when you do change it? Do you understand it? If the answer to any of these questions is no, you have legacy code, and it is draining time and money away from your development efforts. In this book, Michael Feathers offers start-to-finish strategies for working more effectively with large, untested legacy code bases. This book draws on material Michael created for his renowned Object Mentor seminars: techniques Michael has used in mentoring to help hundreds of developers, technical managers, and testers bring their legacy systems under control. The topics covered include Understanding the mechanics of software change: adding features, fixing bugs, improving design, optimizing performance Getting legacy code into a test harness Writing tests that protect you against introducing new problems Techniques that can be used with any language or platform—with examples in Java, C++, C, and C# Accurately identifying where code changes need to be made Coping with legacy systems that aren't object-oriented Handling applications that don't seem to have any structure This book also includes a catalog of twenty-four dependency-breaking techniques that help you work with program elements in isolation and make safer changes.

Working Effectively with Legacy Code

Apply design patterns to solve problems in software architecture and programming using C# 7.x and .NET Core 2 Key FeaturesEnhance your programming skills by implementing efficient design patterns for C# and .NETExplore design patterns for functional and reactive programming to build robust and scalable applicationsDiscover how to work effectively with microservice and serverless architecturesBook Description Design patterns are essentially reusable solutions to common programming problems. When used correctly, they meet crucial software requirements with ease and reduce costs. This book will uncover effective ways to use design patterns and demonstrate their implementation with executable code specific to both C# and .NET Core. Hands-On Design Patterns with C# and .NET Core begins with an overview of object-oriented programming (OOP) and SOLID principles. It provides an in-depth explanation of the Gang of Four (GoF) design patterns such as creational, structural, and behavioral. The book then takes you through functional, reactive, and concurrent patterns, helping you write better code with streams, threads, and coroutines. Toward the end of the book, you’ll learn about the latest trends in architecture, exploring design patterns for microservices, serverless, and cloud native applications. You’ll even understand the considerations that need to be taken into account when choosing between different architectures such as microservices and MVC. By the end of the book, you will be able to write efficient and clear code and be comfortable working on scalable and maintainable projects of any size. What you will learnMake your code more flexible by applying SOLID principlesFollow the Test-driven development (TDD) approach in your .NET Core projectsGet to grips with efficient database migration, data persistence, and testing techniquesConvert a console application to a web application using the right MVPWrite asynchronous, multithreaded, and parallel codeImplement MVVM and work with RxJS and AngularJS to deal with changes in databasesExplore the features of microservices, serverless programming, and cloud computingWho this book is for If you have a basic understanding of C# and the .NET Core framework, this book will help you write code that is easy to reuse and maintain with the help of proven design patterns that you can implement in your code.

Hands-On Design Patterns with C# and .NET Core

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 And Patterns

Unleash the true power of JavaScript by mastering Object-Oriented programming principles and patterns About This Book Covering all the new Object-Oriented features introduced in ES6, this book shows you how to build large-scale web apps Build apps that promote scalability, maintainability, and reusability Learn popular Object-Oriented programming (OOP) principles and design patterns to build robust apps Implement Object-Oriented concepts in a wide range of front-end architectures Who This Book Is For This book is ideal for you if you are a JavaScript developers who wants to gain expertise in OOP with JavaScript to improve your web development skills and build professional quality web applications. What You Will Learn Master JavaScript's OOP features, including the one's provided by ES6 specification Identify and apply the most common design patterns such as Singleton, Factory, Observer, Model-View-Controller, and Mediator Patterns Understand the SOLID principles and their benefits Use the acquired OOP knowledge to build robust and maintainable code Design applications using a modular architecture based on SOLID principles In Detail ECMAScript 6 introduces several new Object-Oriented features that drastically change the way developers structure their projects. Web developers now have some advanced OOP functionality at their disposal to build large-scale applications in JavaScript. With this book, we'll provide you with a comprehensive overview of OOP principles in JavaScript and how they can be implemented to build sophisticated web applications. Kicking off with a subtle refresher on objects, we'll show you how easy it is to define objects with the new ES6 classes. From there, we'll fly you through some essential OOP principles, forming a base for you to get hands-on with encapsulation. You'll get to work with the different methods of inheritance and we'll show you how to avoid using inheritance with Duck Typing. From there, we'll move on to some advanced patterns for object creation and you'll get a strong idea of how to use interesting patterns to present data to users and to bind data. We'll use the famous promises to work with asynchronous processes and will give you some tips on how to organize your code effectively. You'll find out how to create robust code using SOLID principles and finally, we'll show you how to clearly define the goals of your application architecture to get better, smarter, and more effective coding. This book is your one-way ticket to becoming a JavaScript Jedi who can be counted on to deliver flexible and maintainable code. Style and approach This comprehensive guide on advanced OOP principles and patterns in JavaScript is packed with real-world use cases, and shows you how to implement advanced OOP features to build sophisticated web applications that promote scalability and reusability.

Mastering JavaScript Object-Oriented Programming

Become an accomplished Ruby programmer by understanding the design principles, best practices, and trade-offs involved in implementation approaches to keep your Ruby applications maintainable in the long term Key Features: Understand the design principles behind polished Ruby code and trade-offs between implementation approaches Use metaprogramming and DSLs to reduce the amount of code needed without decreasing maintainability Learn Ruby web application design principles and strategies for databases, security, and testing Book Description: Most successful Ruby applications become difficult to maintain over time as the codebase grows in size. Polished Ruby Programming provides you with recommendations and advice for designing Ruby programs that are easy to maintain in the long term. This book takes you through implementation approaches for many common programming situations, the trade-offs inherent in each approach, and why you may choose to use different approaches in different situations. You'll start by learning fundamental Ruby programming principles, such as correctly using core classes, class and method design, variable usage, error handling, and code formatting. Moving on, you'll learn higher-level programming principles, such as library design, use of metaprogramming and domain-specific languages, and refactoring. Finally, you'll learn principles specific to web application development, such as how to choose a database and web framework, and how to use advanced security features. By the end of this Ruby programming book, you'll have gained the skills you need to design robust, high-performance, scalable, and maintainable Ruby applications. While most code examples and principles discussed in the book apply to all Ruby versions, some examples and principles are specific to Ruby 3.0, the latest release at the time of publication. What You Will Learn: Use Ruby's core classes and design custom classes effectively Explore the principles behind variable usage and method argument choice Implement advanced error handling approaches such as exponential backoff Design extensible libraries and plugin systems in Ruby Use metaprogramming and DSLs to avoid code redundancy Implement different approaches to testing and understand their trade-offs Discover design patterns, refactoring, and optimization with Ruby Explore database design principles and advanced web app security Who this book is for: If you already know how to program in Ruby and want to learn more about the principles and best practices behind writing maintainable, scalable, optimized, and well-structured Ruby code, then this Ruby book is for you. Intermediate to advanced-level working knowledge of the Ruby programming language is expected to get the most out of this book.

Polished Ruby Programming

Applying Domain-Driven Design And Patterns Is The First Complete, Practical Guide To Leveraging Patterns, Domain-Driven Design, And Test-Driven Development In .Net Environments. Drawing On Seminal Work By Martin Fowler And Eric Evans, Jimmy Nilsson Shows How To Customize Real-World Architectures For Any .Net Application. You Ll Learn How To Prepare Domain Models For Application Infrastructure; Support Business Rules; Provide Persistence Support; Plan For The Presentation Layer And Ui Testing; And Design For Service Orientation Or Aspect Orientation. Nilsson Illuminates Each Principle With Clear, Well-Annotated Code Examples Based On C# 2.0, .Net 2.0, And Sql Server 2005. His Examples Will Be Valuable Both To C# Developers And Those Working With Other .Net Languages And Databases -- Or Even With Other Platforms, Such As J2Ee.

Applying Domain-Driven Design and Patterns

This book teaches you all the essential knowledge required to learn and apply time-proven SOLID principles of object-oriented design and important design patterns in ASP.NET Core 1.0 (formerly ASP.NET 5) applications. You will learn to write server-side as well as client-side code that makes use of proven practices and patterns. SOLID is an acronym popularized by Robert Martin used to describe five basic principles of good object-oriented design--Single Responsibility, Open/Closed, Liskov Substitution, Interface Segregation and Dependency Inversion. This book covers all five principles and illustrates how they can be used in

ASP.NET Core 1.0 applications. Design Patterns are time proven solutions to commonly occurring software design problems. The most well-known catalog of design patterns comes from Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides, the so-called as GoF patterns (Gang of Four patterns). This book contains detailed descriptions of how to apply Creational, Structural and Behavioral GoF design patterns along with some Patterns of Enterprise Application Architecture. Popular JavaScript patterns are covered, along with working examples of all these patterns in ASP.NET Core 1.0 and C# are included. What You Will Learn: How to apply SOLID principles to ASP.NET applications How to use Gang of Four (GoF) design patterns in ASP.NET applications Techniques for applying Patterns of Enterprise Application Architecture cataloged by Martin Fowler in ASP.NET applications How to organize code and apply design patterns in JavaScript Who This Book Is For:This book is for ASP.NET developers familiar with ASP.NET Core 1.0, C# and Visual Studio.

Beginning SOLID Principles and Design Patterns for ASP.NET Developers

Enhance your programming skills by learning the intricacies of object oriented programming in C# 8 Key FeaturesUnderstand the four pillars of OOP; encapsulation, inheritance, abstraction and polymorphismLeverage the latest features of C# 8 including nullable reference types and Async StreamsExplore various design patterns, principles, and best practices in OOPBook Description Object-oriented programming (OOP) is a programming paradigm organized around objects rather than actions, and data rather than logic. With the latest release of C#, you can look forward to new additions that improve object-oriented programming. This book will get you up to speed with OOP in C# in an engaging and interactive way. The book starts off by introducing you to C# language essentials and explaining OOP concepts through simple programs. You will then go on to learn how to use classes, interfacesm and properties to write pure OOP code in your applications. You will broaden your understanding of OOP further as you delve into some of the advanced features of the language, such as using events, delegates, and generics. Next, you will learn the secrets of writing good code by following design patterns and design principles. You'll also understand problem statements with their solutions and learn how to work with databases with the help of ADO.NET. Further on, you'll discover a chapter dedicated to the Git version control system. As you approach the conclusion, you'll be able to work through OOP-specific interview questions and understand how to tackle them. By the end of this book, you will have a good understanding of OOP with C# and be able to take your skills to the next level. What you will learnMaster OOP paradigm fundamentals Explore various types of exceptions Utilize C# language constructs efficiently Solve complex design problems by understanding OOP Understand how to work with databases using ADO.NET Understand the power of generics in C#Get insights into the popular version control system, Git Learn how to model and design your softwareWho this book is for This book is designed for people who are new to object-oriented programming. Basic C# skills are assumed, however, prior knowledge of OOP in any other language is not required.

Hands-On Object-Oriented Programming with C#

By developing object calculi in which objects are treated as primitives, the authors are able to explain both the semantics of objects and their typing rules, and also demonstrate how to develop all of the most important concepts of object-oriented programming languages: self, dynamic dispatch, classes, inheritance, protected and private methods, prototyping, subtyping, covariance and contravariance, and method specialization. An innovative and important approach to the subject for researchers and graduates.

A Theory of Objects

\"The puzzles and problems in Exceptional C++ not only entertain, they will help you hone your skills to become the sharpest C++ programmer you can be. - Many of these problems are culled from the famous Guru of the Week feature of the Internet newsgroup comp.lang.c++, moderated, expanded and updated to conform to the official ISO/ANSI C++ Standard.\"--BOOK JACKET. - \"Try your skills against the C++

masters and come away with the insight and experience to create more efficient, effective, robust, and portable C++ code.\"--Jacket.

Exceptional C++

A comprehensive undergraduate textbook covering both theory and practical design issues, with an emphasis on object-oriented languages.

Concepts in Programming Languages

Kerievsky lays the foundation for maximizing the use of design patterns by helping the reader view them in the context of refactorings. He ties together two of the most popular methods in software engineering today--refactoring and design patterns--as he helps the experienced developer create more robust software.

Refactoring to Patterns

Exploring C++ uses a series of self–directed lessons to divide C++ into bite–sized chunks that you can digest as rapidly as you can swallow them. The book assumes only a basic understanding of fundamental programming concepts (variables, functions, expressions, statements) and requires no prior knowledge of C or any other particular language. It reduces the usually considerable complexity of C++. The included lessons allow you to learn by doing, as a participant of an interactive education session. You'll master each step in one sitting before you proceed to the next. Author Ray Lischner has designed questions to promote learning new material. And by responding to questions throughout the text, you'll be engaged every step of the way.

Exploring C++

https://johnsonba.cs.grinnell.edu/+38899946/hsparkluy/vproparog/zparlishk/hitlers+bureaucrats+the+nazi+security+
https://johnsonba.cs.grinnell.edu/$40310187/urushtt/hshropgi/dinfluinciy/kawasaki+kz400+1974+workshop+repair+
https://johnsonba.cs.grinnell.edu/~48886319/wrushtu/dovorflowj/vcomplitiy/iso+dis+45001+bsi+group.pdf
https://johnsonba.cs.grinnell.edu/$72942378/irushta/lroturnm/hparlishr/a+look+over+my+shoulder+a+life+in+the+c
https://johnsonba.cs.grinnell.edu/@13648008/qsparklum/frojoicoz/xspetrik/embracing+sisterhood+class+identity+ar
https://johnsonba.cs.grinnell.edu/+63229826/wrushta/hrojoicok/spuykit/adomnan+at+birr+ad+697+essays+in+comm
https://johnsonba.cs.grinnell.edu/@67517033/prushtb/oroturnw/hpuykit/kootenai+electric+silverwood+tickets.pdf
https://johnsonba.cs.grinnell.edu/$25605686/psarckk/qproparon/tcomplitig/c+language+quiz+questions+with+answe
https://johnsonba.cs.grinnell.edu/^62546476/imatugo/crojoicod/rspetriy/beautiful+architecture+leading+thinkers+rev
https://johnsonba.cs.grinnell.edu/!49513718/scavnsistm/ppliynte/yspetriz/chrysler+3+speed+manual+transmission+i