

Bookmark File Building Maintainable Software Java Edition Ten Guidelines For Future Proof Code Pdf File Free

Building Maintainable Software, Java Edition Building Maintainable Software, Java Edition Building Maintainable Software, Java Edition Entwurfsmuster von Kopf bis Fuß *Designing Hexagonal Architecture with Java Effektives Arbeiten mit Legacy Code Modellgetriebene Softwareentwicklung Clean Code - Refactoring, Patterns, Testen und Techniken für sauberen Code Building Maintainable Software, C# Edition Patterns für Enterprise-Application-Architekturen Datenintensive Anwendungen designen Code Complete - Deutsche Ausgabe Readable & Maintainable Code in Java Clean Coder Entwurfsmuster Refactoring Java 9 Modularity Real-World Software Development Effektiv Java programmieren Continuous Delivery in Java Einführung in die Programmierung mit Java Objektorientierte Analyse und Design von Kopf bis Fuß Sieben Wochen, sieben Sprachen (Prags) The Art of Unit Testing Python von Kopf bis Fuß Introducing Jakarta EE CDI Hands-On Design Patterns with Java Java von Kopf bis Fuß Building Maintainable Software, C# Edition Get Your Hands Dirty on Clean Architecture Web-Services mit REST Head First Java Scala High Performance Programming Mit Java programmieren lernen für Dummies JavaScript Patterns Effective Unit Testing C# von Kopf bis Fuß Head First Design Patterns Mehr Hacking mit Python Test-Driven Development with PHP 8*

A practical guide for software architects and Java developers to build cloud-native hexagonal applications using Java and Quarkus to create systems that are easier to refactor, scale, and maintain. Key Features: Learn techniques to decouple business and technology code in an application. Apply hexagonal architecture principles to produce more organized, coherent, and maintainable software. Minimize technical debts and tackle complexities derived from multiple teams dealing with the same code base. Book Description: Hexagonal architecture enhances developers' productivity by decoupling business code from technology code, making the software more change-tolerant, and allowing it to evolve and incorporate new technologies without the need for significant refactoring. By adhering to hexagonal principles, you can structure your software in a way that reduces the effort required to understand and maintain the code. This book starts with an in-depth analysis of hexagonal architecture's building blocks, such as entities, use cases, ports, and adapters. You'll learn how to assemble business code in the Domain hexagon, create features by using ports and use cases in the Application hexagon, and make your software compatible with different technologies by employing adapters in the Framework hexagon. Moving on, you'll get your hands dirty developing a system based on a real-world scenario applying all the hexagonal architecture's building blocks. By creating a hexagonal system, you'll also understand how you can use Java modules to reinforce dependency inversion and ensure the isolation of each hexagon in the architecture. Finally, you'll get to grips with using Quarkus to turn your hexagonal application into a cloud-native system. By the end of this hexagonal architecture book, you'll be able to bring order and sanity to the development of complex and long-lasting applications. What you will learn: Find out how to assemble business rules algorithms using the specification design pattern. Combine domain-driven design techniques with hexagonal principles to create powerful domain models. Employ adapters to make the system support different protocols such as REST, gRPC, and WebSocket. Create a module and package structure based on hexagonal principles. Use Java modules to enforce dependency inversion and ensure isolation between software components. Implement Quarkus DI to manage the life cycle of input and output ports. Who this book is for: This book is for software architects and Java developers who want to improve code maintainability and enhance productivity with an architecture that allows changes in technology without compromising business logic, which is precisely what hexagonal architecture does. Intermediate knowledge of the Java programming language and familiarity with Jakarta EE will help you to get the most out of this book. Write efficient, clean, and powerful Scala code and create high-performing applications that your users will love. About This Book: *This is the first book that explores Scala performance techniques in depth, including how to benchmark your performance so you can understand where to make gains. *It provides a first-principles examination of what performance means in a Scala context. *This book was written by industry experts Vincent Theron and Michael Diamant. Who This Book Is For: If you are a Scala developer with experience in programming Scala applications and know the basics in Scala, syntax, and frameworks such as Lift or Play, this book is for you. This book will also be useful if you are a Java developer who is interested in switching to Scala, but you don't want to give up the performance of Java code. No knowledge of anything outside Scala is required. What You Will Learn: *Find out about performance and how to evaluate the behavior of an application. *Analyze the performance of your application on JVM. *Use Scala features to achieve a high performance benchmark for your application. *Enhance the performance of your application with the Collection API. *Explore asynchronous programming to achieve concurrency and parallelism. *Achieve a deeper understanding of high performance using advanced tools. In Detail: Scala is a statically and strongly typed language that tries to elegantly blend both functional and object-oriented paradigms. It has experienced growing popularity in the past few years as both an appealing and pragmatic choice to write production-ready software in the functional paradigm. Scala lets you solve problems with less code than the alternatives. However, this programmatic gain can come at the cost of performance if you aren't careful. Scala High Performance Programming is written to arm you with the knowledge you need to create highly efficient, clean Scala applications. Starting with the basics of understanding what performance is in a Scala context, we'll look at how to benchmark your performance so you can see the results of your optimizations in action. We'll also take a deep dive into type specialization, concurrency, and parallel programming. By the end of the book, you'll be able to code efficient, optimized, solutions in Scala. Schnappen Sie sich einen Stift, werfen Sie Ihren Rechner an und begeben Sie sich auf eine Erlebnistour durch C#, bei der sowohl Ihre beiden Gehirnhälften als auch Ihre Lachsmuskeln stimuliert werden. C# von Kopf bis Fuß ist ein anregendes Arbeitsbuch für die C# 3.0-Programmierung mit Visual Studio 2008, das alle zentralen Themen von den Sprachgrundlagen bis zur Garbage Collection behandelt. Und Sie lernen auch LINQ, die neueste Syntax von C#, kennen. Wenn Sie dieses Buch durchgearbeitet haben, werden Sie kompetent auch umfangreiche C#-Anwendungen entwickeln können. C# von Kopf bis Fuß ist in einem visuell abwechslungsreichen Format gestaltet, das sich an den neuesten Forschungsergebnissen aus Kognitionswissenschaft und Lerntheorie orientiert und Ihnen das Lernen so einfach wie möglich machen soll. Das Buch bietet Ihnen ein unterhaltsames Lernerlebnis und spielt Ihnen C# direkt ins Hirn - und zwar so, dass es sitzt. Was lernen Sie in diesem Buch? Haben Sie sich schon einmal gewünscht, Sie könnten mit nur einem Buch Python richtig lernen? Mit Python von Kopf bis Fuß schaffen Sie es! Durch die ausgefeilte Von-Kopf-bis-Fuß-Didaktik, die viel mehr als die bloße Syntax und typische How-to-Erklärungen bietet, wird es sogar zum Vergnügen. Python-Grundlagen wie Datenstrukturen und Funktionen verstehen Sie hier schnell, und dann geht es auch schon weiter: Sie programmieren Ihre eigene Web-App, erkunden Datenbank-Management, Ausnahmebehandlung und die Verarbeitung von Daten. Da Python häufig im Data-Science-Umfeld eingesetzt wird, haben in der 2. Auflage diejenigen Techniken ein stärkeres Gewicht bekommen, die in der Welt der Big Data genutzt werden. Wieso sieht dieses Buch so anders aus? In diesem Buch sind die neuesten Erkenntnisse der Kognitionswissenschaft und der Lerntheorie eingeflossen, um Ihnen das Lernen so einfach wie möglich zu machen. Statt einschläfernder Bleiwüsten verwendet dieses Buch eine Vielzahl von Abbildungen und Textstilen, die Ihnen das Wissen direkt ins Hirn spielen - und zwar so, dass es sitzt. Discover the Jakarta EE Contexts and Dependency Injection (CDI 2.0) framework which helps you write better code through the use of well-defined enterprise Java-based components and beans (EJBs). If you have ever wanted to write clean Java EE code, this short book is your best guide for doing so: you will pick up valuable tips along the way from your author's years of experience teaching and coding. Introducing Jakarta EE CDI covers CDI 2.0 in detail and equips you with the theoretical underpinnings of Java EE, now Jakarta EE. This book is packed with so much that by the end of it, you will feel confident to use your new-found knowledge to help you write better, readable, maintainable, and long-lived mission-critical software. What You Will Learn: Write better code with the Jakarta EE Contexts and Dependency Injection (CDI) framework. Work with the powerful, extensible, and well-defined contextual life cycle for

components Use CDI's mechanism for decoupling application components through a typesafe event API Build typesafe interceptors for altering the behaviour of components at runtime Harness the well-defined qualifier system for easy isolation of beans Convert almost any valid Java type to a CDI managed bean with CDI's producer mechanism Who This Book Is For Experienced enterprise Java, Java EE, or J2EE developers who may be new to CDI or dependency injection. Continuous delivery adds enormous value to the business and the entire software delivery lifecycle, but adopting this practice means mastering new skills typically outside of a developer's comfort zone. In this practical book, Daniel Bryant and Abraham Marín-Pérez provide guidance to help experienced Java developers master skills such as architectural design, automated quality assurance, and application packaging and deployment on a variety of platforms. Not only will you learn how to create a comprehensive build pipeline for continually delivering effective software, but you'll also explore how Java application architecture and deployment platforms have affected the way we rapidly and safely deliver new software to production environments. Get advice for beginning or completing your migration to continuous delivery Design architecture to enable the continuous delivery of Java applications Build application artifacts including fat JARs, virtual machine images, and operating system container (Docker) images Use continuous integration tooling like Jenkins, PMD, and find-sec-bugs to automate code quality checks Create a comprehensive build pipeline and design software to separate the deploy and release processes Explore why functional and system quality attribute testing is vital from development to delivery Learn how to effectively build and test applications locally and observe your system while it runs in production Lesbare, wartbare und zuverlässige Tests entwickeln Stubs, Mock-Objekte und automatisierte Frameworks Einsatz von .NET-Tools inkl. NUnit, Rhino Mocks und Typemock Isolator Unit Testing, richtig durchgeführt, kann den Unterschied ausmachen zwischen einem fehlgeschlagenen Projekt und einem erfolgreichen, zwischen einer wartbaren Code-Basis und einer, die niemand freiwillig anpackt, zwischen dem Nach-Hause-Kommen um 2 Uhr nachts oder zum Abendessen, selbst noch kurz vor dem Release-Termin. Roy Osherove führt Sie Schritt für Schritt von einfachen Tests zu Tests, die wartbar, lesbar und zuverlässig sind. Er geht danach auf die Grundlagen des Interaction Testings ein und stellt schließlich bewährte Vorgehensweisen für das Schreiben, das Verwalten und das Warten der Unit Tests in echten Projekten vor. Darüber hinaus werden auch fortgeschrittene Themen behandelt wie Mocks, Stubs und Frameworks wie etwa Typemock Isolator und Rhino Mocks. Sie werden eine Menge zu fortgeschrittenen Testmustern und zur Testorganisation, zum Arbeiten mit Legacy Code und auch zu untestbarem Code erfahren. Und Sie lernen Werkzeuge kennen, die Sie beim Testen von Datenbanken und anderen Technologien brauchen werden. Alle Beispiele sind mit Visual Studio in C# geschrieben, so dass die Beispiele insbesondere für .NET-Entwickler nützlich sind. Aber auch für Programmierer anderer Sprachen wird das Buch von großem Nutzen sein, da die Prinzipien des Unit Testings für andere Sprachen dieselben sind. Roys Blog finden Sie auf ISerializable.com. Aus dem Inhalt: Verwenden eines Test-Frameworks (NUnit) Grundlegende Testattribute Stubs zum Auflösen von Abhängigkeiten Interaction Testing mit Mock-Objekten Testhierarchie und Organisation Die Säulen guter Tests Integration von Unit Testing in das Unternehmen Umgang mit Legacy Code Have you ever felt frustrated working with someone else's code? Difficult-to-maintain source code is a big problem in software development today, leading to costly delays and defects. Be part of the solution. With this practical book, you'll learn 10 easy-to-follow guidelines for delivering Java software that's easy to maintain and adapt. These guidelines have been derived from analyzing hundreds of real-world systems. Written by consultants from the Software Improvement Group (SIG), this book provides clear and concise explanations, with advice for turning the guidelines into practice. Examples for this edition are written in Java, while our companion C# book provides workable examples in that language. Write short units of code: limit the length of methods and constructors Write simple units of code: limit the number of branch points per method Write code once, rather than risk copying buggy code Keep unit interfaces small by extracting parameters into objects Separate concerns to avoid building large classes Couple architecture components loosely Balance the number and size of top-level components in your code Keep your codebase as small as possible Automate tests for your codebase Write clean code, avoiding "code smells" that indicate deeper problems Wie entwickelt man eine gute JavaScript-Anwendung? Dieses Buch hilft Ihnen mit unzähligen Programmier-Mustern und Best Practices dabei, die Frage zu beantworten. Wenn Sie ein erfahrener Entwickler sind, der Probleme im Umfeld von Objekten, Funktionen und Vererbung lösen will, dann sind die Abstraktionen und Code-Vorlagen in diesem Buch ideal - egal, ob Sie eine Client-, Server- oder Desktop-Anwendung mit JavaScript erstellen. Dieses Buch wurde vom JavaScript-Experten Stoyan Stefanov geschrieben - Senior Yahoo! Technical und Architekt von YSlow 2.0, einem Tool zum Optimieren der Webseiten-Performance. Sie finden in JavaScript Patterns praktische Ratschläge für das Implementieren jedes beschriebenen Musters und ergänzend dazu viele nützliche Beispiele. Zudem lernen Sie Anti-Pattern kennen: häufig genutzte Programmier-Ansätze, die mehr Probleme verursachen, als sie lösen. Daten stehen heute im Mittelpunkt vieler Herausforderungen im Systemdesign. Dabei sind komplexe Fragen wie Skalierbarkeit, Konsistenz, Zuverlässigkeit, Effizienz und Wartbarkeit zu klären. Darüber hinaus verfügen wir über eine überwältigende Vielfalt an Tools, einschließlich relationaler Datenbanken, NoSQL-Datenspeicher, Stream- und Batchprocessing und Message Broker. Aber was verbirgt sich hinter diesen Schlagworten? Und was ist die richtige Wahl für Ihre Anwendung? In diesem praktischen und umfassenden Leitfaden unterstützt Sie der Autor Martin Kleppmann bei der Navigation durch dieses schwierige Terrain, indem er die Vor- und Nachteile verschiedener Technologien zur Verarbeitung und Speicherung von Daten aufzeigt. Software verändert sich ständig, die Grundprinzipien bleiben aber gleich. Mit diesem Buch lernen Softwareentwickler und -architekten, wie sie die Konzepte in der Praxis umsetzen und wie sie Daten in modernen Anwendungen optimal nutzen können. Inspizieren Sie die Systeme, die Sie bereits verwenden, und erfahren Sie, wie Sie sie effektiver nutzen können Treffen Sie fundierte Entscheidungen, indem Sie die Stärken und Schwächen verschiedener Tools kennenlernen Steuern Sie die notwendigen Kompromisse in Bezug auf Konsistenz, Skalierbarkeit, Fehlertoleranz und Komplexität Machen Sie sich vertraut mit dem Stand der Forschung zu verteilten Systemen, auf denen moderne Datenbanken aufbauen Werfen Sie einen Blick hinter die Kulissen der wichtigsten Onlinedienste und lernen Sie von deren Architekturen Verhaltensregeln für professionelle Programmierer Erfolgreiche Programmierer haben eines gemeinsam: Die Praxis der Software-Entwicklung ist ihnen eine Herzensangelegenheit. Auch wenn sie unter einem nicht nachlassenden Druck arbeiten, setzen sie sich engagiert ein. Software-Entwicklung ist für sie eine Handwerkskunst. In Clean Coder stellt der legendäre Software-Experte Robert C. Martin die Disziplinen, Techniken, Tools und Methoden vor, die Programmierer zu Profis machen. Dieses Buch steckt voller praktischer Ratschläge und behandelt alle wichtigen Themen vom professionellen Verhalten und Zeitmanagement über die Aufwandsschätzung bis zum Refactoring und Testen. Hier geht es um mehr als nur um Technik: Es geht um die innere Haltung. Martin zeigt, wie Sie sich als Software-Entwickler professionell verhalten, gut und sauber arbeiten und verlässlich kommunizieren und planen. Er beschreibt, wie Sie sich schwierigen Entscheidungen stellen und zeigt, dass das eigene Wissen zu verantwortungsvollem Handeln verpflichtet. In diesem Buch lernen Sie: Was es bedeutet, sich als echter Profi zu verhalten Wie Sie mit Konflikten, knappen Zeitplänen und unvernünftigen Managern umgehen Wie Sie beim Programmieren im Fluss bleiben und Schreibblockaden überwinden Wie Sie mit unerbittlichem Druck umgehen und Burnout vermeiden Wie Sie Ihr Zeitmanagement optimieren Wie Sie für Umgebungen sorgen, in denen Programmierer und Teams wachsen und sich wohlfühlen Wann Sie Nein sagen sollten - und wie Sie das anstellen Wann Sie Ja sagen sollten - und was ein Ja wirklich bedeutet Großartige Software ist etwas Bewundernswertes: Sie ist leistungsfähig, elegant, funktional und erfreut bei der Arbeit sowohl den Entwickler als auch den Anwender. Hervorragende Software wird nicht von Maschinen geschrieben, sondern von Profis, die sich dieser Handwerkskunst unerschütterlich verschrieben haben. Clean Coder hilft Ihnen, zu diesem Kreis zu gehören. Über den Autor: Robert C. Uncle Bob Martin ist seit 1970 Programmierer und bei Konferenzen in aller Welt ein begehrter Redner. Zu seinen Büchern gehören Clean Code - Refactoring, Patterns, Testen und Techniken für sauberen Code und Agile Software Development: Principles, Patterns, and Practices. Als überaus produktiver Autor hat Uncle Bob Hunderte von Artikeln, Abhandlungen und Blogbeiträgen verfasst. Er war Chefredakteur bei The C++ Report und der erste Vorsitzende der Agile Alliance. Martin gründete und leitet die Firma Object Mentor, Inc., die sich darauf spezialisiert hat, Unternehmen bei der Vollendung ihrer Projekte behilflich zu sein. Wenn es um die Entwicklung leistungsfähiger und effizienter Hacking-Tools geht, ist Python für die meisten Sicherheitsanalytiker die Sprache der Wahl. Doch wie genau funktioniert das? In dem neuesten Buch von Justin Seitz - dem Autor des Bestsellers "Hacking mit Python" - entdecken Sie Pythons dunkle Seite. Sie entwickeln Netzwerk-Sniffer, manipulieren Pakete, infizieren virtuelle Maschinen, schaffen unsichtbare Trojaner und vieles mehr. Sie lernen praktisch, wie man • einen "Command-and-Control"-Trojaner mittels GitHub schafft • Sandboxing erkennt und gängige Malware-Aufgaben wie Keylogging und Screenshotting automatisiert • Windows-Rechte mittels

kreativer Prozesskontrolle ausweitet • offensive Speicherforensik-Tricks nutzt, um Passwort-Hashes abzugreifen und Shellcode in virtuelle Maschinen einzuspeisen • das beliebte Web-Hacking-Tool Burp erweitert • die Windows COM-Automatisierung nutzt, um einen Man-in-the-Middle-Angriff durchzuführen • möglichst unbemerkt Daten aus einem Netzwerk abgreift Eine Reihe von Insider-Techniken und kreativen Aufgaben zeigen Ihnen, wie Sie die Hacks erweitern und eigene Exploits entwickeln können. What will you learn from this book? Head First Java is a complete learning experience in Java and object-oriented programming. With this book, you'll learn the Java language with a unique method that goes beyond how-to manuals and helps you become a great programmer. Through puzzles, mysteries, and soul-searching interviews with famous Java objects, you'll quickly get up to speed on Java's fundamentals and advanced topics including lambdas, streams, generics, threading, networking, and the dreaded desktop GUI. If you have experience with another programming language, Head First Java will engage your brain with more modern approaches to coding--the sleeker, faster, and easier to read, write, and maintain Java of today. What's so special about this book? If you've read a Head First book, you know what to expect--a visually rich format designed for the way your brain works. If you haven't, you're in for a treat. With Head First Java, you'll learn Java through a multisensory experience that engages your mind, rather than by means of a text-heavy approach that puts you to sleep. Explore the latest Java-based software development techniques and methodologies through the project-based approach in this practical guide. Unlike books that use abstract examples and lots of theory, Real-World Software Development shows you how to develop several relevant projects while learning best practices along the way. With this engaging approach, junior developers capable of writing basic Java code will learn about state-of-the-art software development practices for building modern, robust and maintainable Java software. You'll work with many different software development topics that are often excluded from software develop how-to references. Featuring real-world examples, this book teaches you techniques and methodologies for functional programming, automated testing, security, architecture, and distributed systems. Jetzt aktuell zu Java 8: Dieses Buch ist ein moderner Klassiker zum Thema Entwurfsmuster. Mit dem einzigartigen Von Kopf bis Fuß-Lernkonzept gelingt es den Autoren, die anspruchsvolle Materie witzig, leicht verständlich und dennoch gründlich darzustellen. Jede Seite ist ein Kunstwerk für sich, mit vielen visuellen Überraschungen, originellen Comic-Zeichnungen, humorvollen Dialogen und geistreichen Selbstlernkontrollen. Spätestens, wenn es mal wieder heißt "Spitzen Sie Ihren Bleistift", wird dem Leser klar, dass bei diesem Buch sein Mitmachen gefragt ist. Das ist nicht nur unterhaltsam, sondern auch effektiv: Komplexe Sachverhalte lassen sich nach Erkenntnis der modernen Lernwissenschaft am gründlichsten über mehrere verschiedene Kanäle verstehen. Das Buch verspricht dem Leser daher nicht nur Spaß beim Lernen, er wird nach der Lektüre auch die Herausforderungen des Software-Designs meistern können. Java von Kopf bis Fuß ist eine umfassende Erlebnistour durch die Objektorientierte Programmierung und Java. Ganz nach modernen Lernprinzipien entworfen, führt Sie dieses Buch von den Grundlagen der Sprache zu fortgeschrittenen Themen wie Threads, Netzwerk-Sockets und verteilter Programmierung. Und das Beste ist: Sie werden dabei lernen, wie ein echter objektorientierter Entwickler zu denken. Auf Ihrem Weg dorthin werden Sie nicht nur lesen. Sie knacken Rätsel, spielen, lösen Krimis und kommen mit Java auf eine Weise zusammen, die Sie sich nicht hätten träumen lassen. Aber natürlich werden Sie. Dieses Buch ist die deutsche Übersetzung eines Klassikers der Programmierliteratur von Steve McConnell. Seine mit vielen Preisen ausgezeichneten Bücher helfen Programmierern seit Jahren, besseren und effizienteren Code zu schreiben. Das Geheimnis dieses Buches liegt in der Art, wie der Autor das vorhandene Wissen über Programmier-Techniken aus wissenschaftlichen Quellen mit den Erfahrungen aus der taglichen praktischen Arbeit am Code zusammenführt und daraus die wesentlichen Grundvoraussetzungen der Softwareentwicklung und die effektivsten Arbeitstechniken ableitet. Verständliche Beispiele und klare Anleitungen vermitteln dem Leser dieses Wissen auf unkomplizierte Weise. Dieses Buch informiert und stimuliert, ganz gleich, wie viel sie bereits über Programmierung wissen, welche Entwicklungsumgebung und Sprache sie bevorzugen und welche Arten von Anwendungen sie normalerweise programmieren. • Umfassend überarbeitete und aktualisierte Neuauflage des Standardwerks in vollständig neuer Übersetzung • Verbesserungsmöglichkeiten von bestehender Software anhand von Code-Smells erkennen und Code effizient überarbeiten • Umfassender Katalog von Refactoring-Methoden mit Code-Beispielen in JavaScript Seit mehr als zwanzig Jahren greifen erfahrene Programmierer rund um den Globus auf dieses Buch zurück, um bestehenden Code zu verbessern und leichter lesbar zu machen sowie Software besser warten und erweitern zu können. In diesem umfassenden Standardwerk zeigt Ihnen Martin Fowler, was die Vorteile von Refactoring sind, wie Sie verbesserungsbedürftigen Code erkennen und wie Sie ein Refactoring - unabhängig von der verwendeten Programmiersprache - erfolgreich durchführen. In einem umfangreichen Katalog gibt Fowler Ihnen verschiedene Refactoring-Methoden mit ausführlicher Erläuterung, Motivation, Vorgehensweise und einfachen Beispielen in JavaScript an die Hand. Darüber hinaus behandelt er insbesondere folgende Schwerpunkte: • Allgemeine Prinzipien und Durchführung des Refactorings • Refactoring anwenden, um die Lesbarkeit, Wartbarkeit und Erweiterbarkeit von Programmen zu verbessern • Code-Smells erkennen, die auf Verbesserungsmöglichkeiten durch Refactoring hinweisen • Entwicklung zuverlässiger Tests für das Refactoring • Erkennen von Fallstricken und notwendigen Kompromissen bei der Durchführung eines Refactorings Diese vollständig neu übersetzte Ausgabe wurde von Grund auf überarbeitet, um den maßgeblichen Veränderungen der modernen Programmierung Rechnung zu tragen. Sie enthält einen aktualisierten Katalog von Refactoring-Methoden sowie neue Beispiele für einen funktionalen Programmieransatz. Have you ever felt frustrated working with someone else's code? Difficult-to-maintain source code is a big problem in software development today, leading to costly delays and defects. Be part of the solution. With this practical book, you'll learn 10 easy-to-follow guidelines for delivering Java software that's easy to maintain and adapt. These guidelines have been derived from analyzing hundreds of real-world systems. Written by consultants from the Software Improvement Group (SIG), this book provides clear and concise explanations, with advice for turning the guidelines into practice. Examples for this edition are written in Java, while our companion C# book provides workable examples in that language. Write short units of code: limit the length of methods and constructors Write simple units of code: limit the number of branch points per method Write code once, rather than risk copying buggy code Keep unit interfaces small by extracting parameters into objects Separate concerns to avoid building large classes Couple architecture components loosely Balance the number and size of top-level components in your code Keep your codebase as small as possible Automate tests for your codebase Write clean code, avoiding "code smells" that indicate deeper problems Modellgetriebene Entwicklung befasst sich mit der Erstellung kompletter Softwaresysteme aus Modellen. Das Buch stellt einen praxisorientierten Leitfaden für modellgetriebene Entwicklung dar und richtet sich dabei an Architekten, Entwickler sowie technische Projektleiter. Obwohl die Model-Driven Architecture (MDA) der OMG einen hohen Stellenwert bei den Betrachtungen einnimmt, betrachtet das Buch auch allgemeine Aspekte modellgetriebener Entwicklung. Das Buch ist dreigeteilt in eine Einführung, einen praktischen Leitfaden mit einem ausführlichen Fallbeispiel sowie zusätzliche Kapitel, die bestimmte Aspekte der Thematik genauer beleuchten. Mit diesen sieben Sprachen erkunden Sie die wichtigsten Programmiermodelle unserer Zeit. Lernen Sie die dynamische Typisierung kennen, die Ruby, Python und Perl so flexibel und verlockend macht. Lernen Sie das Prototyp-System verstehen, das das Herzstück von JavaScript bildet. Erfahren Sie, wie das Pattern Matching in Prolog die Entwicklung von Scala und Erlang beeinflusst hat. Entdecken Sie, wie sich die rein funktionale Programmierung in Haskell von der Lisp-Sprachfamilie, inklusive Clojure, unterscheidet. Erkunden Sie die parallelen Techniken, die das Rückgrat der nächsten Generation von Internet-Anwendungen bilden werden. Finden Sie heraus, wie man Erlangs "Lass es abstürzen"-Philosophie zum Aufbau fehlertoleranter Systeme nutzt. Lernen Sie das Akteur-Modell kennen, das das parallele Design bei Io und Scala bestimmt. Entdecken Sie, wie Clojure die Versionierung nutzt, um einige der schwierigsten Probleme der Nebenläufigkeit zu lösen. Hier finden Sie alles in einem Buch. Nutzen Sie die Konzepte einer Sprache, um kreative Lösungen in einer anderen Programmiersprache zu finden - oder entdecken Sie einfach eine Sprache, die Sie bisher nicht kannten. Man kann nie wissen - vielleicht wird sie sogar eines ihrer neuen Lieblingswerkzeuge. A PHP developer's practical guide to planning, writing, and deploying maintainable and extensible software solutions with automated tests in a well-structured way Key Features Learn to organize and translate business requirements into maintainable software Minimize regressions and bugs while releasing new features or bug fixes Build maintainable applications using SOLID principles Book Description PHP web developers end up building complex enterprise projects without prior experience in test-driven and behavior-driven development which results in software that's complex and difficult to maintain. This step-by-step guide helps you manage the complexities of large-scale web applications. It takes you through the processes of working on a project, starting from understanding business requirements and translating them into actual maintainable software, to automated deployments. You'll learn how to break down business

requirements into workable and actionable lists using Jira. Using those organized lists of business requirements, you'll understand how to implement behavior-driven development (BDD) and test-driven development (TDD) to start writing maintainable PHP code. You'll explore how to use the automated tests to help you stop introducing regressions to an application each time you release code by using continuous integration. By the end of this book, you'll have learned how to start a PHP project, break down the requirements, build test scenarios and automated tests, and write more testable and maintainable PHP code. By learning these processes, you'll be able to develop more maintainable, and reliable enterprise PHP applications. What you will learn

Understand how to start a software project
Discover how to use Jira as a tool to organize your tasks
Explore when and how to write unit, integration, and functional tests using PHPUnit
Write behavior-driven tests using Behat
Apply SOLID principles to help you write more testable code
Get the most out of your automated tests by using continuous integration
Use continuous delivery to help you prepare your application for deployment

Who this book is for
This book is for PHP software developers looking to implement TDD and BDD in their projects. An understanding of object-oriented programming is required to get the most out of this book. Professional software development experience will also be beneficial to understand real-life business cases.

Können Sie Ihren Code leicht ändern? Können Sie fast unmittelbar Feedback bekommen, wenn Sie ihn ändern? Verstehen Sie ihn? Wenn Sie eine dieser Fragen mit nein beantworten, arbeiten Sie mit Legacy Code, der Geld und wertvolle Entwicklungszeit kostet. Michael Feathers erläutert in diesem Buch Strategien für den gesamten Entwicklungsprozess, um effizient mit großen, ungetesteten Code-Basen zu arbeiten. Dabei greift er auf erprobtes Material zurück, das er für seine angesehenen Object-Mentor-Seminare entwickelt hat. Damit hat er bereits zahlreichen Entwicklern, technischen Managern und Testern geholfen, ihre Legacy-Systeme unter Kontrolle zu bringen. Darüber hinaus finden Sie auch einen Katalog mit 24 Techniken zur Aufhebung von Dependencies, die Ihnen zeigen, wie Sie isoliert mit Programmelementen arbeiten und Code sicherer ändern können. You know you don't want to reinvent the wheel, so you look to design patterns--the lessons learned by those who've faced the same software design problems. With design patterns, you get to take advantage of the best practices and experience of others so that you can spend your time on something more challenging. Something more fun. This book shows you the patterns that matter, when to use them and why, how to apply them to your own designs, and the object-oriented design principles on which the patterns are based. Most importantly, you want to learn design patterns in a way that won't put you to sleep. If you've read a Head First book, you know what to expect--a visually rich format designed for the way your brain works. Newly updated and expanded, Head First Design Patterns, 2E, will load patterns into your brain in a way that sticks. In a way that makes you better at solving software design problems and better at speaking the language of patterns with others on your team. Join the hundreds of thousands of developers who have already improved their object-oriented design skills through Head First Design Patterns.

> Kommentare, Formatierung, Strukturierung Fehler-Handling und Unit-Tests

Zahlreiche Fallstudien, Best Practices, Heuristiken und Code Smells
Clean Code - Refactoring, Patterns, Testen und Techniken für sauberen Code

Aus dem Inhalt: Lernen Sie, guten Code von schlechtem zu unterscheiden
Sauberen Code schreiben und schlechten Code in guten umwandeln
Aussagekräftige Namen sowie gute Funktionen, Objekte und Klassen erstellen
Code so formatieren, strukturieren und kommentieren, dass er bestmöglich lesbar ist
Ein vollständiges Fehler-Handling implementieren, ohne die Logik des Codes zu verschleiern
Unit-Tests schreiben und Ihren Code testgesteuert entwickeln
Selbst schlechter Code kann funktionieren. Aber wenn der Code nicht sauber ist, kann er ein Entwicklungsunternehmen in die Knie zwingen. Jedes Jahr gehen unzählige Stunden und beträchtliche Ressourcen verloren, weil Code schlecht geschrieben ist. Aber das muss nicht sein. Mit Clean Code präsentiert Ihnen der bekannte Software-Experte Robert C. Martin ein revolutionäres Paradigma, mit dem er Ihnen aufzeigt, wie Sie guten Code schreiben und schlechten Code überarbeiten. Zusammen mit seinen Kollegen von Object Mentor destilliert er die besten Praktiken der agilen Entwicklung von sauberem Code zu einem einzigartigen Buch. So können Sie sich die Erfahrungswerte der Meister der Software-Entwicklung aneignen, die aus Ihnen einen besseren Programmierer machen werden - anhand konkreter Fallstudien, die im Buch detailliert durchgearbeitet werden. Sie werden in diesem Buch sehr viel Code lesen. Und Sie werden aufgefordert, darüber nachzudenken, was an diesem Code richtig und falsch ist. Noch wichtiger: Sie werden herausgefordert, Ihre professionellen Werte und Ihre Einstellung zu Ihrem Beruf zu überprüfen. Clean Code besteht aus drei Teilen: Der erste Teil beschreibt die Prinzipien, Patterns und Techniken, die zum Schreiben von sauberem Code benötigt werden. Der zweite Teil besteht aus mehreren, zunehmend komplexeren Fallstudien. An jeder Fallstudie wird aufgezeigt, wie Code gesäubert wird - wie eine mit Problemen behaftete Code-Basis in eine solide und effiziente Form umgewandelt wird. Der dritte Teil enthält den Ertrag und den Lohn der praktischen Arbeit: ein umfangreiches Kapitel mit Best Practices, Heuristiken und Code Smells, die bei der Erstellung der Fallstudien zusammengetragen wurden. Das Ergebnis ist eine Wissensbasis, die beschreibt, wie wir denken, wenn wir Code schreiben, lesen und säubern. Dieses Buch ist ein Muss für alle Entwickler, Software-Ingenieure, Projektmanager, Team-Leiter oder Systemanalytiker, die daran interessiert sind, besseren Code zu produzieren.

Über den Autor: Robert C. »Uncle Bob« Martin entwickelt seit 1970 professionell Software. Seit 1990 arbeitet er international als Software-Berater. Er ist Gründer und Vorsitzender von Object Mentor, Inc., einem Team erfahrener Berater, die Kunden auf der ganzen Welt bei der Programmierung in und mit C++, Java, C#, Ruby, OO, Design Patterns, UML sowie Agilen Methoden und eXtreme Programming helfen. Kluge Bücher über Objektorientierte Analyse & Design gibt es viele. Leider versteht man die meisten erst, wenn man selbst schon Profi-Entwickler ist... Und was machen all die Normalsterblichen, die natürlich davon gehört haben, dass OOA&D dazu beiträgt, kontinuierlich tolle Software zu schreiben, Software, die Chef und Kunden glücklich macht - wenn sie aber nicht wissen, wie sie anfangen sollen? Sie könnten damit beginnen, dieses Buch zu lesen! Denn Objektorientierte Analyse & Design von Kopf bis Fuß zeigt Ihnen Schritt für Schritt, wie Sie richtige OO-Software analysieren, entwerfen und entwickeln. Software, die sich leicht wiederverwenden, warten und erweitern lässt. Software, die keine Kopfschmerzen bereitet. Software, der Sie neue Features spendieren können, ohne die existierende Funktionalität zu gefährden. Sie lernen, Ihre Anwendungen flexibel zu halten, indem Sie OO-Prinzipien wie Kapselung und Delegation anwenden. Sie lernen, die Wiederverwendung Ihrer Software dadurch zu begünstigen, dass Sie das OCP (das Open-Closed-Prinzip) und das SRP (das Single-Responsibility-Prinzip) befolgen. Sie lernen, wie sich verschiedene Entwurfsmuster, Entwicklungsansätze und Prinzipien zu einem echten OOA&D-Projektlebenszyklus ergänzen, UML, Anwendungsfälle und -diagramme zu verwenden, damit auch alle Beteiligten klar miteinander kommunizieren können, und Sie die Software abliefern, die gewünscht wird. Diesem Buch wurden die neuesten Erkenntnisse aus der Lerntheorie und der Kognitionswissenschaft zugrunde gelegt - Sie können davon ausgehen, dass Sie nicht nur schnell vorankommen, sondern dabei auch noch eine Menge Spaß haben! Have you ever felt frustrated working with someone else's code? Difficult-to-maintain source code is a big problem in software development today, leading to costly delays and defects. Be part of the solution. With this practical book, you'll learn 10 easy-to-follow guidelines for delivering C# software that's easy to maintain and adapt. These guidelines have been derived from analyzing hundreds of real-world systems. Written by consultants from the Software Improvement Group (SIG), this book provides clear and concise explanations, with advice for turning the guidelines into practice. Examples for this edition are written in C#, while our companion Java book provides clear examples in that language. Write short units of code: limit the length of methods and constructors
Write simple units of code: limit the number of branch points per method
Write code once, rather than risk copying buggy code
Keep unit interfaces small by extracting parameters into objects
Separate concerns to avoid building large classes
Couple architecture components loosely
Balance the number and size of top-level components in your code
Keep your codebase as small as possible
Automate tests for your codebase
Write clean code, avoiding "code smells" that indicate deeper problems
Gain insight into how hexagonal architecture can help to keep the cost of development low over the complete lifetime of an application
Key Features
Explore ways to make your software flexible, extensible, and adaptable
Learn new concepts that you can easily blend with your own software development style
Develop the mindset of building maintainable solutions instead of taking shortcuts

Book Description
We would all like to build software architecture that yields adaptable and flexible software with low development costs. But, unreasonable deadlines and shortcuts make it very hard to create such an architecture. Get Your Hands Dirty on Clean Architecture starts with a discussion about the conventional layered architecture style and its disadvantages. It also talks about the advantages of the domain-centric architecture styles of Robert C. Martin's Clean Architecture and Alistair Cockburn's Hexagonal Architecture. Then, the book dives into hands-on chapters that show you how to manifest a hexagonal architecture in actual code. You'll learn in detail about different mapping strategies between the layers of a hexagonal architecture and see how to assemble the architecture elements into an application. The later chapters

demonstrate how to enforce architecture boundaries. You'll also learn what shortcuts produce what types of technical debt and how, sometimes, it is a good idea to willingly take on those debts. After reading this book, you'll have all the knowledge you need to create applications using the hexagonal architecture style of web development. What you will learn

- Identify potential shortcomings of using a layered architecture
- Apply methods to enforce architecture boundaries
- Find out how potential shortcuts can affect the software architecture
- Produce arguments for when to use which style of architecture
- Structure your code according to the architecture
- Apply various types of tests that will cover each element of the architecture

Who this book is for This book is for you if you care about the architecture of the software you are building. To get the most out of this book, you must have some experience with web development. The code examples in this book are in Java. If you are not a Java programmer but can read object-oriented code in other languages, you will be fine. In the few places where Java or framework specifics are needed, they are thoroughly explained. Understand Gang of Four, architectural, functional, and reactive design patterns and how to implement them on modern Java platforms, such as Java 12 and beyond

Key Features

- Learn OOP, functional, and reactive patterns for creating readable and maintainable code
- Explore architectural patterns and practices for building scalable and reliable applications
- Tackle all kinds of performance-related issues and streamline development using design patterns

Book Description Java design patterns are reusable and proven solutions to software design problems. This book covers over 60 battle-tested design patterns used by developers to create functional, reusable, and flexible software. Hands-On Design Patterns with Java starts with an introduction to the Unified Modeling Language (UML), and delves into class and object diagrams with the help of detailed examples. You'll study concepts and approaches to object-oriented programming (OOP) and OOP design patterns to build robust applications. As you advance, you'll explore the categories of GOF design patterns, such as behavioral, creational, and structural, that help you improve code readability and enable large-scale reuse of software. You'll also discover how to work effectively with microservices and serverless architectures by using cloud design patterns, each of which is thoroughly explained and accompanied by real-world programming solutions. By the end of the book, you'll be able to speed up your software development process using the right design patterns, and you'll be comfortable working on scalable and maintainable projects of any size.

What you will learn

- Understand the significance of design patterns for software engineering
- Visualize software design with UML diagrams
- Strengthen your understanding of OOP to create reusable software systems
- Discover GOF design patterns to develop scalable applications
- Examine programming challenges and the design patterns that solve them
- Explore architectural patterns for microservices and cloud development

Who this book is for If you are a developer who wants to learn how to write clear, concise, and effective code for building production-ready applications, this book is for you. Familiarity with the fundamentals of Java is assumed. Most software is built to last a long time, and one of the key challenges is to keep technical debt at bay. Applying fundamental principles of clean code is crucial to the long-term success of any software project. In this book, *Readable and Maintainable Code*, in Java you will gain the ability to write high quality code that others will enjoy reading and working with. First, you will learn the importance of naming and how to choose great names for your classes, variables, and methods. Next, you will discover the pitfalls of Java constructors, methods, tests, comments, and exception handling before examining how you can either prevent or overcome them. Finally, you will learn some hands-on practical tips on how you can maintain the top quality of your code. By the end of this course, you will have the necessary skills to write clean, readable and maintainable code that human beings can read and understand clearly.

Steigen Sie mit diesem Buch in die Welt des Programmierens ein und zwar mit der beliebten Programmiersprache Java! Schritt für Schritt werden Sie mit den Grundlagen, wie zum Beispiel Variablen, Schleifen und objektorientierter Programmierung, vertraut gemacht, probieren viele anschauliche Beispiele aus und schreiben Ihr erstes eigenes Programm. Dieses Buch steht Ihnen bei allen Herausforderungen jederzeit mit hilfreichen Tipps und Lösungsvorschlägen zur Seite, sodass Sie für Ihren Weg zum Programmierer optimal gerüstet sind!

Summary

Effective Unit Testing is written to show how to write good tests—tests that are concise and to the point, expressive, useful, and maintainable. Inspired by Roy Osherove's bestselling *The Art of Unit Testing*, this book focuses on tools and practices specific to the Java world. It introduces you to emerging techniques like behavior-driven development and specification by example, and shows you how to add robust practices into your toolkit. About Testing Test the components before you assemble them into a full application, and you'll get better software. For Java developers, there's now a decade of experience with well-crafted tests that anticipate problems, identify known and unknown dependencies in the code, and allow you to test components both in isolation and in the context of a full application. About this Book Effective Unit Testing teaches Java developers how to write unit tests that are concise, expressive, useful, and maintainable. Offering crisp explanations and easy-to-absorb examples, it introduces emerging techniques like behavior-driven development and specification by example. Programmers who are already unit testing will learn the current state of the art. Those who are new to the game will learn practices that will serve them well for the rest of their career. Purchase of the print book comes with an offer of a free PDF, ePub, and Kindle eBook from Manning. Also available is all code from the book. About the Author Lasse Koskela is a coach, trainer, consultant, and programmer. He hacks on open source projects, helps companies improve their productivity, and speaks frequently at conferences around the world. Lasse is the author of *Test Driven*, also published by Manning. What's Inside A thorough introduction to unit testing Choosing best-of-breed tools Writing tests using dynamic languages Efficient test automation Table of Contents

PART 1 FOUNDATIONS

- The promise of good tests
- In search of good Test doubles

PART 2 CATALOG

- Readability
- Maintainability
- Trustworthiness

PART 3 DIVERSIONS

- Testable design
- Writing tests in other JVM languages
- Speeding up test execution

Have you ever felt frustrated working with someone else's code? Difficult-to-maintain source code is a big problem in software development today, leading to costly delays and defects. Be part of the solution. With this practical book, you'll learn 10 easy-to-follow guidelines for delivering C# software that's easy to maintain and adapt. These guidelines have been derived from analyzing hundreds of real-world systems. Written by consultants from the Software Improvement Group (SIG), this book provides clear and concise explanations, with advice for turning the guidelines into practice. Examples for this edition are written in C#, while our companion Java book provides clear examples in that language. Write short units of code: limit the length of methods and constructors Write simple units of code: limit the number of branch points per method Write code once, rather than risk copying buggy code Keep unit interfaces small by extracting parameters into objects Separate concerns to avoid building large classes Couple architecture components loosely Balance the number and size of top-level components in your code Keep your codebase as small as possible Automate tests for your codebase Write clean code, avoiding "code smells" that indicate deeper problems

The upcoming Java 9 module system will affect existing applications and offer new ways of creating modular and maintainable applications. With this hands-on book, Java developers will learn not only about the joys of modularity, but also about the patterns needed to create truly modular and reliable applications. Authors Sander Mak and Paul Bakker teach you the concepts behind the Java 9 module system, along with the new tools it offers. You'll also gain learn how to modularize existing code and how to build new Java applications in a modular way. Understand Java 9 module system concepts Master the patterns and practices for building truly modular applications Migrate existing applications and libraries to Java 9 modules Use JDK 9 tools for modular development and migration

Getting the books **Building Maintainable Software Java Edition Ten Guidelines For Future Proof Code** now is not type of inspiring means. You could not deserted going once books deposit or library or borrowing from your connections to door them. This is an totally simple means to specifically acquire lead by on-line. This online notice **Building Maintainable Software Java Edition Ten Guidelines For Future Proof Code** can be one of the options to accompany you in imitation of having additional time.

It will not waste your time. bow to me, the e-book will very publicize you extra business to read. Just invest tiny mature to admittance this on-line broadcast **Building Maintainable Software Java Edition Ten Guidelines For Future Proof Code** as without difficulty as review them wherever you are now.

This is likewise one of the factors by obtaining the soft documents of this **Building Maintainable Software Java Edition Ten Guidelines For**

Future Proof Code by online. You might not require more get older to spend to go to the book initiation as with ease as search for them. In some cases, you likewise attain not discover the declaration Building Maintainable Software Java Edition Ten Guidelines For Future Proof Code that you are looking for. It will unconditionally squander the time.

However below, gone you visit this web page, it will be fittingly extremely simple to get as with ease as download lead Building Maintainable Software Java Edition Ten Guidelines For Future Proof Code

It will not receive many grow old as we accustom before. You can realize it even though feat something else at house and even in your workplace. fittingly easy! So, are you question? Just exercise just what we allow under as with ease as evaluation **Building Maintainable Software Java Edition Ten Guidelines For Future Proof Code** what you later than to read!

If you ally compulsion such a referred **Building Maintainable Software Java Edition Ten Guidelines For Future Proof Code** book that will give you worth, acquire the definitely best seller from us currently from several preferred authors. If you want to entertaining books, lots of novels, tale, jokes, and more fictions collections are also launched, from best seller to one of the most current released.

You may not be perplexed to enjoy all books collections Building Maintainable Software Java Edition Ten Guidelines For Future Proof Code that we will certainly offer. It is not on the costs. Its nearly what you obsession currently. This Building Maintainable Software Java Edition Ten Guidelines For Future Proof Code, as one of the most functional sellers here will entirely be along with the best options to review.

Yeah, reviewing a book **Building Maintainable Software Java Edition Ten Guidelines For Future Proof Code** could grow your close connections listings. This is just one of the solutions for you to be successful. As understood, expertise does not recommend that you have extraordinary points.

Comprehending as competently as deal even more than additional will come up with the money for each success. next to, the publication as skillfully as keeness of this Building Maintainable Software Java Edition Ten Guidelines For Future Proof Code can be taken as without difficulty as picked to act.

discourse.labfab.fr