

Growing Object Oriented Software Guided By Tests Addison Wesley Signature Series Beck

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.

Wenn Sie mit Programmiersprachen wie C++ oder Java vertraut sind, erscheint Ihnen JavaScript kaum objektorientiert. JavaScript kennt kein Klassenkonzept und Sie müssen nicht einmal Objekte definieren, um Programme zu erstellen. Lassen Sie sich aber nicht täuschen: JavaScript ist eine außerordentlich mächtige und ausdrucksstarke objektorientierte Sprache, die Ihnen selbst zahlreiche Designentscheidungen überlässt. In diesem kompakten Buch erforscht Nicholas Zakas mit Ihnen die objektorientierte Natur der Sprache und legt die einzigartige Implementierung der Vererbung und anderer Schlüsselkonzepte offen. Er erklärt Ihnen, wie Sie Kapselung, Aggregation, Vererbung und Polymorphismus umsetzen können, wenn sie JavaScript-Code schreiben. Im Einzelnen erfahren Sie, • welche Unterschiede es zwischen primitiven und Referenztypen gibt, • was JavaScript-Funktionen so einzigartig macht, • auf welche verschiedenen Arten Sie Objekte definieren, • wie Sie Ihren eigenen Konstruktor erzeugen und mit Prototypen arbeiten, • welche Vererbungsmuster für Typen und Objekte möglich sind. Das Buch führt auch erfahrene Programmierer zu einem tieferen Verständnis von JavaScript. Es ermöglicht Ihnen damit, besser strukturierten, flexibleren und effizienteren JavaScript-Code zu schreiben. Autor Nikolas Zakas arbeitet als Entwickler im JavaScript-Umfeld und ist durch zahlreiche Veröffentlichungen und Vorträge bekannt. Er ist bereits Autor mehrerer erfolgreicher Bücher zu JavaScript.

Scrum ist ein agiles Management-Framework, das keine Entwicklungspraktiken empfiehlt oder gar vorschreibt. Auswahl und Einsatz der richtigen Praktiken fallen unter die Selbstorganisation des Teams. Ohne den Einsatz geeigneter Entwicklungspraktiken und -tools ist der Einsatz von Scrum in der Softwareentwicklung jedoch nicht dauerhaft erfolgreich. Dieses Buch beschreibt praxisnah die wichtigsten Praktiken wie Architekturvision, inkrementeller Entwurf, Continuous Integration, testgetriebene Entwicklung, Refactoring, Akzeptanztests sowie modellgetriebene und verteilte Entwicklung mit Scrum.

Was lernen Sie mit diesem Buch? Haben Sie sich schon einmal gefragt, was es mit testgetriebener Entwicklung auf sich hat? Oder auf welcher Basis es die richtig guten Consultants schaffen, gewaltige Stundensätze zu kassieren? Vielleicht sind Sie auch gerade an dem Punkt, an dem Sie Ihre Builds automatisieren wollen, Ihren Code in eine Versionskontrolle füttern, einem Refactoring unterziehen oder mit ein paar Entwurfsmustern anreichern wollen. Egal: Wenn Sie mit diesem Buch fertig sind, werden Sie ganz selbstverständlich Ihre Burndown-Rate verfolgen, den Durchsatz Ihres Teams berücksichtigen und sich erfolgreich Ihren Weg durch Anforderungen, Entwurf, Entwicklung und Auslieferung iterieren. Wieso sieht dieses Buch so anders aus? Wir gehen davon aus, dass Ihre Zeit zu kostbar ist, um mit neuem Stoff zu kämpfen. Statt Sie mit Bleiwüstentexten langsam in den Schlaf zu wiegen, verwenden wir für Softwareentwicklung von Kopf bis Fuß ein visuell und inhaltlich abwechslungsreiches Format, das auf Grundlage neuester Forschungsergebnisse im Bereich der Kognitionswissenschaft und der Lerntheorie entwickelt wurde. Wir wissen nämlich, wie Ihr Gehirn arbeitet.

Domain-Driven Design (DDD) richtet den Fokus in der Softwareentwicklung auf das Wesentliche: die Domäne. Die Domäne wird als Modell in die Software übertragen. Damit entwickeln Sie Software in hoher Qualität, die lange hält, den Anwender zufriedenstellt und die Basis für Microservices bildet. Dieses Buch bietet einen kompakten Einstieg in DDD. Die wesentlichen Konzepte, wie die Entwicklung einer Ubiquitous Language, das Aufteilen der Domäne in Bounded Contexts und die Konstruktion innerhalb von Bounded Contexts, werden vermittelt. Außerdem wird die Anbindung von Legacy-Systemen behandelt. Die Themen im Einzelnen: - Strategisches Design mit Bounded Contexts und der Ubiquitous Language - Strategisches Design mit Subdomains - Strategisches Design mit Context Mapping - Taktisches Design mit Aggregates - Taktisches Design mit Domain Events Auch auf Techniken zur Beschleunigung von Design und das Management von Projekten wird eingegangen. Insbesondere wird erläutert, wie Event Storming, DDD in einem agilen Projekt und die Modellierung mit Timebox funktionieren. Der Leser findet in diesem Buch viele konkrete Handlungsvorschläge für die Praxis und wird so befähigt, die Zusammenarbeit von Entwicklern und Domain Experts sowie zwischen Teams zu fördern. Als Extra befindet sich ein Glossar mit den wichtigsten DDD-Begriffen auf den Umschlaginnenseiten.

Test-Driven Development (TDD) is now an established technique for delivering better software faster. TDD is based on a simple idea: Write tests for your code before you write the code itself. However, this "simple" idea takes skill and judgment to do well. Now there's a practical guide to TDD that takes you beyond the basic concepts. Drawing on a decade of experience building real-world systems, two TDD pioneers show how to let tests guide your development and "grow" software that is coherent, reliable, and maintainable. Steve Freeman and Nat Pryce describe the processes they use, the design principles they strive to achieve, and some of the tools that help them get the job done. Through an extended worked example, you'll learn how TDD works at multiple levels, using tests to drive the features and the object-oriented structure of the code, and using Mock Objects to discover and then describe relationships between objects. Along the way, the book systematically addresses challenges that development teams encounter with

TDD—from integrating TDD into your processes to testing your most difficult features. Coverage includes Implementing TDD effectively: getting started, and maintaining your momentum throughout the project Creating cleaner, more expressive, more sustainable code Using tests to stay relentlessly focused on sustaining quality Understanding how TDD, Mock Objects, and Object-Oriented Design come together in the context of a real software development project Using Mock Objects to guide object-oriented designs Succeeding where TDD is difficult: managing complex test data, and testing persistence and concurrency

Die objektorientierte Sprache Python eignet sich hervorragend zum Schreiben von Skripten, Programmen und Prototypen. Sie ist frei verfügbar, leicht zu lernen und zwischen allen wichtigen Plattformen portabel, einschließlich Linux, Unix, Windows und Mac OS. Damit Sie im Programmieralltag immer den Überblick behalten, sind die verschiedenen Sprachmerkmale und Elemente in Python – kurz & gut übersichtlich zusammengestellt. Für Auflage 5 wurde die Referenz komplett überarbeitet, erweitert und auf den neuesten Stand gebracht, so dass sie die beiden aktuellen Versionen 2.7 und 3.4 berücksichtigt. Python – kurz & gut behandelt unter anderem: Eingebaute Typen wie Zahlen, Listen, Dictionaries u.v.a.; nweisungen und Syntax für Entwicklung und Ausführung von Objekten; Die objektorientierten Entwicklungstools in Python; Eingebaute Funktionen, Ausnahmen und Attribute; pezielle Methoden zur Operatorenüberladung; Weithin benutzte Standardbibliotheksmodule und Erweiterungen; Kommandozeilenoptionen und Entwicklungswerkzeuge. Mark Lutz stieg 1992 in die Python-Szene ein und ist seitdem als aktiver Pythonista bekannt. Er gibt Kurse, hat zahlreiche Bücher geschrieben und mehrere Python-Systeme programmiert.

Software-Architekten müssen komplexe fachliche und technische Anforderungen an IT-Systeme umsetzen und diese Systeme durch nachvollziehbare Strukturen flexibel und erweiterbar gestalten. Dieser Praxisleitfaden zeigt Ihnen, wie Sie Software-Architekturen effektiv und systematisch entwickeln können. Der bekannte Software-Architekt Gernot Starke unterstützt Sie mit praktischen Tipps, Architekturmustern und seinen Erfahrungen. Er gibt Antworten auf zentrale Fragen: - Welche Aufgaben haben Software-Architekten? - Wie gehen Software-Architekten beim Entwurf vor? - Wie kommunizieren und dokumentieren Sie Software-Architekturen? - Wie helfen Architekturmuster und Architekturbausteine? - Wie bewerten Sie Software-Architekturen? - Wie behandeln Sie Persistenz, grafische Benutzeroberflächen, Geschäftsregeln, Integration, Verteilung, Sicherheit, Fehlerbehandlung, Workflow-Management und sonstige technische Konzepte? - Was müssen Software-Architekten über MDA/MDS, UML 2 und arc42 wissen? - Welche Aufgaben nehmen Enterprise-IT-Architekten wahr?

Dieses Buch beschreibt alle wichtigen praxistauglichen Methoden des Software-Tests für eingebettete Systeme und zeigt, wie sie sich in ein planungsgesteuertes bzw. agiles Projekt eingliedern lassen. Dabei werden die Teststufen in chronologischer Reihenfolge behandelt und die Erläuterungen mit zahlreichen Beispielen aus Projekten mit und ohne Sicherheitsrelevanz illustriert. Behandelt werden u. a. folgende Themen: • Statische Code-Analyse • Unit-Tests • Hardware/Software-Integrationstests • Software/Software-Integrationstests • Systemtests • Data Race Testing • Schedulability und Echtzeittests • Normen und Haftungsrisiko Persönliche Bewertungen von Testmethoden, Erfahrungsberichte und eine Diskussion von Testwerkzeugen am Ende jedes Kapitels geben dem Leser wichtige Orientierungshilfen bei der Umsetzung des Stoffs und beim Kauf von Werkzeugen. Verständnisfragen mit Antworten sowie Übungsbeispiele mit Lösungen erleichtern die Verwendung des Buches im Hochschulbereich und geben auch dem Leser aus dem industriellen Umfeld die Möglichkeit zur Selbstkontrolle. Die Buchkapitel sind weitgehend unabhängig voneinander aufgebaut. Sie erlauben das Lesen der Kapitel in einer beliebigen Reihenfolge und machen das Buch auch als Nachschlagewerk wertvoll. Neu in der 2. Auflage: • Aktualisierung auf die neuesten Tools und Normen – insbesondere ISO 29119 • Neues Kapitel zu Trace-Daten im Testumfeld • Zusätzliche Erfahrungsberichte

Testgetriebene Entwicklung (TDD) ist eine moderne Methode in der Softwareentwicklung, mit der Programmierer und Tester die Anzahl der Fehler im System erheblich verringern, wartungsfreundlicheren Code schreiben und die Software gefahrlos an geänderte Anforderungen anpassen können. Dieses Buch vermittelt praktische TDD-Kenntnisse und beschreibt die Probleme und Vorteile der Verwendung dieser Technik für C++-Systeme. Die vielen ausführlichen Codebeispiele führen schrittweise von den Grundlagen von TDD zu anspruchsvollen Themen: • TDD verwenden, um C++-Altsysteme zu verbessern • Problematische Systemabhängigkeiten erkennen und handhaben • Abhängigkeiten in C++ injizieren • Frameworks für C++ einsetzen, die TDD unterstützen • C++11-Features nutzen, die die Anwendung von TDD erleichtern Unabhängig davon, ob Sie viel Erfahrung mit Unit Tests haben oder ein absoluter Neuling auf diesem Gebiet sind, lernen Sie mit diesem Buch die testgetriebene Entwicklung in C++ erfolgreich anzuwenden.

Agiles Produktmanagement mit Scrum hilft Ihnen, innovative Produkte mit Scrum zu entwickeln. Anhand zahlreicher Praxisbeispiele erklärt das Buch anschaulich und leicht verständlich den Einsatz agiler Produktmanagementkonzepte und -techniken. Hierzu zählen: •Die richtige Anwendung der Product-Owner-Rolle •Der effektive Einsatz einer agilen Produktvision und einer agilen Produkt-Roadmap •Der richtige Umgang mit dem Product Backlog inklusive Priorisierung, User Stories und nichtfunktionaler Anforderungen •Das Erstellen eines realistischen Releaseplans •Das richtige Verhalten des Product Owner in den Sprint-Besprechungen •Die Etablierung der Product-Owner-Rolle im Unternehmen Dieses Buch ist für alle Leser, die als Product Owner arbeiten oder dies vorhaben, sowie für Führungskräfte und Scrum Master, die sich für die Anwendung der Rolle und den Einsatz der Praktiken interessieren.

• 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.

Growing Object-Oriented Software, Guided by Tests Pearson Education

Diese Buch bietet eine praxisbezogene und anschauliche Einführung in die akzeptanztestgetriebene Entwicklung (Acceptance Test-driven Development, ATDD, auch bekannt als Behavior-driven Development oder Specification-by-Example). Anhand zweier ausführlicher Praxisbeispiele erfährt der Leser, wie sich Testautomatisierung innerhalb eines agilen Entwicklungsprozesses verwenden lässt. Anschließend werden die grundlegenden Prinzipien zusammengefasst und verdeutlicht. Dadurch erlebt der Leser praxisnah, was ATDD ist, und bekommt wertvolle Hinweise, wie er entsprechende Prozesse aufbauen kann.

Programmierenlernen ist wirklich nicht schwer. Du wirst erstaunt sein, wie einfach es ist, interaktive Welten und lustige Spiele zu entwickeln. Und du wirst garantiert viel Spaß dabei haben! Du lädst dir schnell den ICE Code Editor als Browsererweiterung herunter, mit dem du auch offline arbeiten kannst, und los geht's. Alles was du programmierst, siehst du direkt im Code Editor: animierte Figuren, deinen eigenen Avatar, der Radschlagen kann, oder dein eigenes Spiel mit Obstmonstern, Höhlenpuzzeln und Floßfahrten. Wenn du wissen willst, was es mit dem ganzen Code so auf sich hat, wird dir das ganz genau erklärt. Wenn du dich aber lieber auf die Praxis konzentrieren willst, kannst du die Theorie außen vor lassen. Warum 3D-Spiele? Weil das Programmierenlernen damit besonders großen Spaß macht, denn, Hand aufs Herz, wer spielt nicht gern Computerspiele? Und so macht das Lernen nicht nur riesigen Spaß, wir programmieren dabei auch jede Menge tolle Sachen: Du stellst coole Spielfiguren her und Welten, in denen du spielen kannst, du programmierst deinen eigenen Avatar, bastelst lila Monster und erzeugst Weltraumsimulationen. Warum JavaScript? JavaScript ist die Sprache des World Wide Web und die einzige Programmiersprache, die alle Webbrowser ohne zusätzliche Software verstehen. Wenn du gelernt hast, in JavaScript zu programmieren, kannst du nicht nur solche Spiele programmieren, wie du in diesem Buch kennenlernen wirst. Du kannst auch alle möglichen Websites programmieren. Außerdem zeigen wir dir, wie du anschließend deine Spiele mit deinen Freunden teilen und sie auf deiner eigenen Website einbinden kannst. Und dann erzählst du all deinen Freunden: „Das hab' ich gemacht!“ An wen richtet sich dieses Buch? Obwohl für Kinder geschrieben, können auch Erwachsene hiermit das Programmieren erlernen. Es richtet sich an Programmieranfänger von 11 bis 99.

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.

Dieses Lehrbuch des international bekannten Autors und Software-Entwicklers Craig Larman ist ein Standardwerk zur objektorientierten Analyse und Design unter Verwendung von UML 2.0 und Patterns. Das Buch zeichnet sich insbesondere durch die Fähigkeit des Autors aus, komplexe Sachverhalte anschaulich und praxisnah darzustellen. Es vermittelt grundlegende OOA/D-Fertigkeiten und bietet umfassende Erläuterungen zur iterativen Entwicklung und zum Unified Process (UP). Anschliessend werden zwei Fallstudien vorgestellt, anhand derer die einzelnen Analyse- und Designprozesse des UP in Form einer Inception-, Elaboration- und Construction-Phase durchgespielt werden

Jetzt aktualisiert und erweitert: das Standardwerk zu Entwurfsmustern »Dieses Buch vereint Spaß, herzhaftes Lachen, profundes technisches Verständnis und geniale Praxistipps zu einem denkwürdigen Lesegenuss.« Richard Helm, Gang of Four Softwareentwickler wollen das Rad nicht neu erfinden: Mit Entwurfsmustern können Sie von den Erfahrungen und Best Practices anderer profitieren Die Design Patterns werden detailliert und verständlich erklärt, inklusive Herleitung und geeigneten Anwendungsfällen 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 heißt "Spitzen Sie Ihren Bleistift", wird Leser:innen klar, dass bei diesem Buch 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 Ihnen daher nicht nur Spaß beim Lernen, Sie werden nach der Lektüre auch die Herausforderungen des Softwaredesigns meistern können.

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!

Bill ist IT-Manager bei Parts Unlimited. An einem Dienstagmorgen erhält er auf der Fahrt zur Arbeit einen Anruf seines CEO. Die neue IT-Initiative der Firma mit dem Codenamen Projekt Phoenix ist entscheidend für die Zukunft von Parts Unlimited, aber das Projekt hat Budget und Zeitplan massiv überzogen. Der CEO will, dass Bill direkt an ihn berichtet

und das ganze Chaos in neunzig Tagen aufräumt, denn sonst wird Bills gesamte Abteilung outgesourct. Mit der Hilfe eines Vorstandsmitglieds und dessen mysteriöser Philosophie der Drei Wege wird Bill klar, dass IT-Arbeit mehr mit dem Fertigungsbereich in einer Fabrik zu tun hat als er sich je vorstellen konnte. Die Zeit drängt: Bill muss dafür sorgen, dass der Arbeitsfluss auch zwischen den Abteilungen deutlich besser läuft und das Business-Funktionalität zuverlässig bereitgestellt wird. Drei Koryphäen der DevOps-Bewegung liefern hier die rasante und unterhaltsame Story, in der sich jeder, der im IT-Bereich arbeitet, wiederfinden wird. Sie erfahren nicht nur, wie Sie Ihre eigene IT-Organisation verbessern können - nach der Lektüre dieses Buchs werden Sie IT auch nie wieder so sehen wie zuvor.

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 Vervollständigung ihrer Projekte behilflich zu sein.

Discover the untapped features of object-oriented programming and use it with other software tools to code fast, efficient applications. Key Features Explore the complexities of object-oriented programming (OOP) Discover what OOP can do for you Learn to use the key tools and software engineering practices to support your own programming needs Book Description Your experience and knowledge always influence the approach you take and the tools you use to write your programs. With a sound understanding of how to approach your goal and what software paradigms to use, you can create high-performing applications quickly and efficiently. In this two-part book, you'll discover the untapped features of object-oriented programming and use it with other software tools to code fast and efficient applications. The first part of the book begins with a discussion on how OOP is used today and moves on to analyze the ideas and problems that OOP doesn't address. It continues by deconstructing the complexity of OOP, showing you its fundamentally simple core. You'll see that, by using the distinctive elements of OOP, you can learn to build your applications more easily. The next part of this book talks about acquiring the skills to become a better programmer. You'll get an overview of how various tools, such as version control and build management, help make your life easier. This book also discusses the pros and cons of other programming paradigms, such as aspect-oriented programming and functional programming, and helps to select the correct approach for your projects. It ends by talking about the philosophy behind designing software and what it means to be a "good" developer. By the end of this two-part book, you will have learned that OOP is not always complex, and you will know how you can evolve into a better programmer by learning about ethics, teamwork, and documentation. What you will learn Untangle the complexity of object-oriented programming by breaking it down to its essential building blocks Realize the full potential of OOP to design efficient, maintainable programs Utilize coding best practices, including TDD, pair programming and code reviews, to improve your work Use tools, such as source control and IDEs, to work more efficiently Learn how to most productively work with other developers Build your own software development philosophy Who this book is for This book is ideal for programmers who want to understand the philosophy behind creating software and what it means to be "good" at designing software. Programmers who want to deconstruct the OOP paradigm and see how it can be reconstructed in a clear, straightforward way will also find this book useful. To understand the ideas expressed in this book, you must be an experienced programmer who wants to evolve their practice.

Lesbare, wartbare und zuverlässige Tests entwickelnStubs, Mock-Objekte und automatisierte FrameworksEinsatz 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 Software Testen und Analysieren: Prozesse, Prinzipien und Techniken ist das erste Buch, das eine grosse Bandbreite sich ergänzender Software-Test und -Analysetechniken in einer ganzheitlichen, schlussigen Form erklärt. Es behandelt das ganze Themenspektrum, angefangen bei den Grundlagen und Basistheorien bis hin zu Organisations- und Prozessfragen von Anwendersoftware. Der Schwerpunkt des Buches liegt darin, anwendungsorientierte Techniken auszuwählen, um beim Testen und Analysieren von Software eine akzeptable Qualität zu akzeptablen Preisen zu bekommen."

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 Aktor-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. Spring Boot hat seit 2014 das Spring-Ökosystem revolutioniert und verdrängt in zunehmendem Maße "klassische" Spring-Anwendungen. Spring Boot ist kein neues Framework, sondern basiert auf Spring und dem Spring-Ökosystem. Es vereinfacht die Verwaltung von Abhängigkeiten und die Konfiguration des Spring-Frameworks. Spring Boot löst dabei Probleme, die einer effektiven und effizienten Produktivsetzung im Weg stehen, und bietet vielfältige Möglichkeiten, testgetrieben zu entwickeln. Spring Boot sollte die erste Wahl sein, Springbasierte Anwendungen zu entwickeln, unabhängig davon, ob es sich um Microservices handelt oder nicht. Dieses Buch bietet eine umfassende Einführung in die von Spring Boot unterstützten Spring-Module und -Technologien: Webanwendungen Reaktive Anwendungen Security Datenbanktechnologien Caching Tests und Dokumentation Darüber hinaus stellt es verschiedene Möglichkeiten vor, Spring-Boot-Anwendungen zu deployen, sowohl in klassischen als auch in Cloud-Szenarien. Hinweise auf Best Practices sowie eine Übersicht der zahlreichen Änderungen von Spring Boot 1 auf Version 2 runden das Buch ab.

[Copyright: 75b1986ce3b4dddb1cb00474475f3ed8](https://www.addison-wesley.com/9781491986364)