

Agile Java Crafting Code With Test Driven Development Robert C Martin

The Pragmatic Programmers classic is back! Freshly updated for modern software development, Pragmatic Unit Testing in Java 8 With JUnit teaches you how to write and run easily maintained unit tests in JUnit with confidence. You'll learn mnemonics to help you know what tests to write, how to remember all the boundary conditions, and what the qualities of a good test are. You'll see how unit tests can pay off by allowing you to keep your system code clean, and you'll learn how to handle the stuff that seems too tough to test. Pragmatic Unit Testing in Java 8 With JUnit steps you through all the important unit testing topics. If you've never written a unit test, you'll see screen shots from Eclipse, IntelliJ IDEA, and NetBeans that will help you get past the hard part--getting set up and started. Once past the basics, you'll learn why you want to write unit tests and how to effectively use JUnit. But the meaty part of the book is its collected unit testing wisdom from people who've been there, done that on production systems for at least 15 years: veteran author and developer Jeff Langr, building on the wisdom of Pragmatic Programmers Andy Hunt and Dave Thomas. You'll learn: How to craft your unit tests to minimize your effort in maintaining them. How to use unit tests to help keep your system clean. How to test the tough stuff. Memorable mnemonics to help you remember what's important when writing unit tests. How to help your team reap and sustain the benefits of unit testing. You won't just learn about unit testing in theory--you'll work through numerous code examples. When it comes to programming, hands-on is the only way to learn!

Agile Values and Principles for a New Generation “In the journey to all things Agile, Uncle Bob has been there, done that, and has the both the t-shirt and the scars to show for it. This delightful book is part history, part personal stories, and all wisdom. If you want to understand what Agile is and how it came to be, this is the book for you.” –Grady Booch “Bob’s frustration colors every sentence of Clean Agile, but it’s a justified frustration. What is in the world of Agile development is nothing compared to what could be. This book is Bob’s perspective on what to focus on to get to that ‘what could be.’ And he’s been there, so it’s worth listening.” –Kent Beck “It’s good to read Uncle Bob’s take on Agile. Whether just beginning, or a seasoned Agilista, you would do well to read this book. I agree with almost all of it. It’s just some of the parts make me realize my own shortcomings, dammit. It made me double-check our code coverage (85.09%).” –Jon Kern Nearly twenty years after the Agile Manifesto was first presented, the legendary Robert C. Martin (“Uncle Bob”) reintroduces Agile values and principles for a new generation—programmers and nonprogrammers alike. Martin, author of Clean Code and other highly influential software development guides, was there at Agile’s founding. Now, in Clean Agile: Back to Basics, he strips away misunderstandings and distractions that over the years have made it harder to use Agile than was originally intended. Martin describes what Agile is in no uncertain terms: a small discipline that helps small teams manage small projects . . . with huge implications because every big project is comprised of many small projects. Drawing on his fifty years’ experience with projects of every conceivable type, he shows how Agile can help you bring true professionalism to software development. Get back to the basics—what Agile is, was, and should always be Understand the origins, and proper practice, of SCRUM Master essential business-facing Agile practices, from small releases and acceptance tests to whole-team communication Explore Agile team members’ relationships with each other, and with their product Rediscover indispensable Agile technical practices: TDD, refactoring, simple design, and pair programming Understand the central roles values and craftsmanship play in your Agile team’s success If you want Agile’s true benefits, there are no shortcuts: You need to do Agile right. Clean Agile: Back to Basics will show you how, whether you’re a developer, tester, manager, project manager, or customer. Register your book for convenient access to downloads, updates, and/or corrections as they become available. See inside book for details.

The Unified Modeling Language has become the industry standard for the expression of software designs. The Java programming language continues to grow in popularity as the language of choice for the serious application developer. Using UML and Java together would appear to be a natural marriage, one that can produce considerable benefit. However, there are nuances that the seasoned developer needs to keep in mind when using UML and Java together. Software expert Robert Martin presents a concise guide, with numerous examples, that will help the programmer leverage the power of both development concepts. The author ignores features of UML that do not apply to java programmers, saving the reader time and effort. He provides direct guidance and points the reader to real-world usage scenarios. The overall practical approach of this book brings key information related to Java to the many presentations. The result is an highly practical guide to using the UML with Java.

If you program in C++ you've been neglected. Test-driven development (TDD) is a modern software development practice that can dramatically reduce the number of defects in systems, produce more maintainable code, and give you the confidence to change your software to meet changing needs. But C++ programmers have been ignored by those promoting TDD--until now. In this book, Jeff Langr gives you hands-on lessons in the challenges and rewards of doing TDD in C++. Modern C++ Programming With Test-Driven Development, the only comprehensive treatment on TDD in C++ provides you with everything you need to know about TDD, and the challenges and benefits of implementing it in your C++ systems. Its many detailed code examples take you step-by-step from TDD basics to advanced concepts. As a veteran C++ programmer, you're already writing high-quality code, and you work hard to maintain code quality. It doesn't have to be that hard. In this book, you'll learn: how to use TDD to improve legacy C++ systems how to identify and deal with troublesome system dependencies how to do dependency injection, which is particularly tricky in C++ how to use testing tools for C++ that aid TDD new C++11 features that facilitate TDD As you grow in TDD mastery, you'll discover how to keep a massive C++ system from becoming a design mess over time, as well as particular C++ trouble spots to avoid. You'll find out how to prevent your tests from being a maintenance burden and how to think in TDD without giving up your hard-won C++ skills. Finally, you'll see how to grow and sustain TDD in your team. Whether you're a complete unit-testing novice or an experienced tester, this book will lead you to mastery of test-driven development in C++. What You Need A C++ compiler running under Windows or Linux, preferably one that supports C++11. Examples presented in the book were built under gcc 4.7.2. Google Mock 1.6 (downloadable for free; it contains Google Test as well) or an alternate C++ unit testing tool. Most examples in the book are written for Google Mock, but it isn't difficult to translate them to your tool of choice. A good programmer's editor or IDE. cmake, preferably. Of course, you can use your own preferred make too. CMakeLists.txt files are provided for each project. Examples provided were built using cmake version 2.8.9. Various freely-available third-party libraries are used as the basis for examples in the book. These include: cURL JsonCpp Boost (filesystem, date_time/gregorian, algorithm, assign) Several examples use the boost headers/libraries. Only one example uses cURL and

JsonCpp.

More and more Agile projects are seeking architectural roots as they struggle with complexity and scale - and they're seeking lightweight ways to do it Still seeking? In this book the authors help you to find your own path Taking cues from Lean development, they can help steer your project toward practices with longstanding track records Up-front architecture? Sure. You can deliver an architecture as code that compiles and that concretely guides development without bogging it down in a mass of documents and guesses about the implementation Documentation? Even a whiteboard diagram, or a CRC card, is documentation: the goal isn't to avoid documentation, but to document just the right things in just the right amount Process? This all works within the frameworks of Scrum, XP, and other Agile approaches

A guide to writing computer code covers such topics as variable naming, presentation style, error handling, and security.

Tap into the wisdom of experts to learn what every programmer should know, no matter what language you use. With the 97 short and extremely useful tips for programmers in this book, you'll expand your skills by adopting new approaches to old problems, learning appropriate best practices, and honing your craft through sound advice. With contributions from some of the most experienced and respected practitioners in the industry--including Michael Feathers, Pete Goodliffe, Diomidis Spinellis, Cay Horstmann, Verity Stob, and many more--this book contains practical knowledge and principles that you can apply to all kinds of projects. A few of the 97 things you should know: "Code in the Language of the Domain" by Dan North "Write Tests for People" by Gerard Meszaros "Convenience Is Not an -ility" by Gregor Hohpe "Know Your IDE" by Heinz Kabutz "A Message to the Future" by Linda Rising "The Boy Scout Rule" by Robert C. Martin (Uncle Bob) "Beware the Share" by Udi Dahan

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.

Looks at the principles and clean code, includes case studies showcasing the practices of writing clean code, and contains a list of heuristics and "smells" accumulated from the process of writing clean code.

Peter Seibel interviews 15 of the most interesting computer programmers alive today in Coders at Work, offering a companion volume to Apress's highly acclaimed best-seller Founders at Work by Jessica Livingston. As the words "at work" suggest, Peter Seibel focuses on how his interviewees tackle the day-to-day work of programming, while revealing much more, like how they became great programmers, how they recognize programming talent in others, and what kinds of problems they find most interesting. Hundreds of people have suggested names of programmers to interview on the Coders at Work web site: www.codersatwork.com. The complete list was 284 names. Having digested everyone's feedback, we selected 15 folks who've been kind enough to agree to be interviewed: Frances Allen: Pioneer in optimizing compilers, first woman to win the Turing Award (2006) and first female IBM fellow Joe Armstrong: Inventor of Erlang Joshua Bloch: Author of the Java collections framework, now at Google Bernie Cosell: One of the main software guys behind the original ARPANET IMPs and a master debugger Douglas Crockford: JSON founder, JavaScript architect at Yahoo! L. Peter Deutsch: Author of Ghostscript, implementer of Smalltalk-80 at Xerox PARC and Lisp 1.5 on PDP-1 Brendan Eich: Inventor of JavaScript, CTO of the Mozilla Corporation Brad Fitzpatrick: Writer of LiveJournal, OpenID, memcached, and Perlbal Dan Ingalls: Smalltalk implementor and designer Simon Peyton Jones: Coinventor of Haskell and lead designer of Glasgow Haskell Compiler Donald Knuth: Author of The Art of Computer Programming and creator of TeX Peter Norvig: Director of Research at Google and author of the standard text on AI Guy Steele: Coinventor of Scheme and part of the Common Lisp Gang of Five, currently working on Fortress Ken Thompson: Inventor of UNIX Jamie Zawinski: Author of XEmacs and early Netscape/Mozilla hacker

The latest title in Addison Wesley's world-renowned Robert C. Martin Series on better software development, Code That Fits in Your Head offers indispensable practical advice for writing code at a sustainable pace, and controlling the complexity that causes too many software projects to spin out of control. Reflecting decades of experience consulting on software projects and helping development teams succeed, Mark Seemann shares proven practices and heuristics, supported by realistic advice. His guidance ranges from checklists to teamwork, encapsulation to decomposition, API design to unit testing and troubleshooting. Throughout, Seemann illuminates his insights with up-to-date code examples drawn from a start to finish sample project. Seemann's examples are written in C##, and designed to be clear and useful to every object-oriented enterprise developer, whether they use C#, Java, or another language. Code That Fits in Your Head is accompanied by the complete code base for this sample application, organized in a Git repository to facilitate further exploration of details that don't fit in the text.

Presents practical advice on the disciplines, techniques, tools, and practices of computer programming and how to approach software development with a sense of pride, honor, and self-respect.

This book introduces the author's collection of wisdom under one umbrella: Software Craftmanship. This approach is unique in that it spells out a programmer-centric way to build software. In other words, all the best computers, proven components, and most robust languages mean nothing if the programmer does not understand their craft.

In Clean Craftmanship , the legendary Robert C. Martin ("Uncle Bob") has written every programmer's definitive guide to working well. Martin brings together the disciplines, standards, and ethics you need to deliver robust, effective code quickly and productively, and be proud of all the software you write -- every single day. Martin, the best-selling author of The Clean Coder , begins with a pragmatic, technical, and prescriptive guide to five foundational disciplines of software craftsmanship: test-driven development, refactoring, simple design, collaborative programming (pairing), and acceptance tests. Next, he moves up to standards -- outlining the baseline expectations the world has of software developers, illuminating how those often differ from their own perspectives, and helping you repair the mismatch. Finally, he turns to the ethics of the programming profession, describing ten fundamental promises all software developers should make to their colleagues, their users, and above all, themselves . With Martin's guidance and advice, you can consistently write code that builds trust instead of undermining it -- trust among your users and throughout a society that depends on software for its very survival. 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.

Thousands of organizations are adopting Scrum to transform the way they execute complex projects, in software and beyond. This guide will give you the skills and confidence needed to deploy Scrum, resulting in high-performing teams and satisfied customers. Drawing on years of hands-on experience helping companies succeed, Certified Scrum Trainer (CST) Mitch Lacey helps you overcome the major challenges of Scrum adoption and the deeper issues that emerge later. Extensively revised to reflect improved Scrum practices and tools, this edition adds an all-new section of tips from the field. Lacey covers many new topics, including immersive interviewing, collaborative estimation, and deepening business alignment. In 35 engaging chapters, you'll learn how to build support and maximize value across your company. Now part of the renowned Mike Cohn Signature Series on agile development, this pragmatic guide addresses everything from establishing roles and priorities to determining team velocity, setting sprint length, and conducting customer reviews. Coverage includes Bringing teams and new team members on board Creating a workable definition of "done" Planning for short-term wins, and removing impediments to success Balancing predictability and adaptability in release planning Running productive daily scrums Fixing failing sprints Accurately costing projects, and measuring the value they deliver Managing risks in dynamic Scrum projects Prioritizing and estimating backlogs Working with distributed and offshore teams Institutionalizing improvements, and extending agility throughout the organization Packed with real-world examples straight from Lacey's experience, this book will be invaluable to anyone transitioning to Scrum, seeking to improve their early results, or trying to get back on track.

The Complete Guide to Writing More Maintainable, Manageable, Pleasing, and Powerful Ruby Applications Ruby's widely admired ease of use has a downside: Too many Ruby and Rails applications have been created without concern for their long-term maintenance or evolution. The Web is awash in Ruby code that is now virtually impossible to change or extend. This text helps you solve that problem by using powerful real-world object-oriented design techniques, which it thoroughly explains using simple and practical Ruby examples. Sandi Metz has distilled a lifetime of conversations and presentations about object-oriented design into a set of Ruby-focused practices for crafting manageable, extensible, and pleasing code. She shows you how to build new applications that can survive success and repair existing applications that have become impossible to change. Each technique is illustrated with extended examples, all downloadable from the companion Web site, poodr.info. The first title to focus squarely on object-oriented Ruby application design, Practical Object-Oriented Design in Ruby will guide you to superior outcomes, whatever your previous Ruby experience. Novice Ruby programmers will find specific rules to live by; intermediate Ruby programmers will find valuable principles they can flexibly interpret and apply; and advanced Ruby programmers will find a common language they can use to lead development and guide their colleagues. This guide will help you Understand how object-oriented programming can help you craft Ruby code that is easier to maintain and upgrade Decide what belongs in a single Ruby class Avoid entangling objects that should be kept separate Define flexible interfaces among objects Reduce programming overhead costs with duck typing Successfully apply inheritance Build objects via composition Design cost-effective tests Solve common problems associated with poorly designed Ruby code

"One of the great things about the book is the way the authors explain concepts very simply using analogies rather than programming examples--this has been very inspiring for a product I'm

working on: an audio-only introduction to OOP and software development." –Bruce Eckel "...I would expect that readers with a basic understanding of object-oriented programming and design would find this book useful, before approaching design patterns completely. Design Patterns Explained complements the existing design patterns texts and may perform a very useful role, fitting between introductory texts such as UML Distilled and the more advanced patterns books." –James Noble Leverage the quality and productivity benefits of patterns—without the complexity! Design Patterns Explained, Second Edition is the field's simplest, clearest, most practical introduction to patterns. Using dozens of updated Java examples, it shows programmers and architects exactly how to use patterns to design, develop, and deliver software far more effectively. You'll start with a complete overview of the fundamental principles of patterns, and the role of object-oriented analysis and design in contemporary software development. Then, using easy-to-understand sample code, Alan Shalloway and James Trott illuminate dozens of today's most useful patterns: their underlying concepts, advantages, tradeoffs, implementation techniques, and pitfalls to avoid. Many patterns are accompanied by UML diagrams. Building on their best-selling First Edition, Shalloway and Trott have thoroughly updated this book to reflect new software design trends, patterns, and implementation techniques. Reflecting extensive reader feedback, they have deepened and clarified coverage throughout, and reorganized content for even greater ease of understanding. New and revamped coverage in this edition includes Better ways to start "thinking in patterns" How design patterns can facilitate agile development using eXtreme Programming and other methods How to use commonality and variability analysis to design application architectures The key role of testing into a patterns-driven development process How to use factories to instantiate and manage objects more effectively The Object-Pool Pattern—a new pattern not identified by the "Gang of Four" New study/practice questions at the end of every chapter Gentle yet thorough, this book assumes no patterns experience whatsoever. It's the ideal "first book" on patterns, and a perfect complement to Gamma's classic Design Patterns. If you're a programmer or architect who wants the clearest possible understanding of design patterns—or if you've struggled to make them work for you—read this book.

Learn Rails the way the Rails core team recommends it, along with the tens of thousands of developers who have used this broad, far-reaching tutorial and reference. If you're new to Rails, you'll get step-by-step guidance. If you're an experienced developer, get the comprehensive, insider information you need for the latest version of Ruby on Rails. The new edition of this award-winning classic is completely updated for Rails 6 and Ruby 2.6, with information on system testing, Webpack, and advanced JavaScript. Ruby on Rails helps you produce high-quality, beautiful-looking web applications quickly - you concentrate on creating the application, and Rails takes care of the details. Rails 6 brings many improvements, and this edition is updated to cover the new features and changes in best practices. We start with a step-by-step walkthrough of building a real application, and in-depth chapters look at the built-in Rails features. Follow along with an extended tutorial as you write a web-based store application. Eliminate tedious configuration and housekeeping, seamlessly incorporate Ajax and JavaScript, send and receive emails, manage background jobs with ActiveJob, and build real-time features using WebSockets and ActionCable. Test your applications as you write them using the built-in unit, integration, and system testing frameworks, internationalize your applications, and deploy your applications easily and securely. New in this edition is coverage of Action Mailer, which allows you to receive emails in your app as well as ActionText, a zero-configuration rich text editing feature. Rails 1.0 was released in December 2005. This book was there from the start, and didn't just evolve alongside Rails, it evolved with Rails. It has been developed in consultation with the Rails core team. In fact, Rails itself is tested against the code in this book. What You Need: All you need is a Windows, Mac OS X, or Linux machine to do development on. This book will take you through the steps to install Rails and its dependencies. If you aren't familiar with the Ruby programming language, this book contains a chapter that covers the basics necessary to understand the material in the book.

This book is for developers who want to work smarter so they can focus their efforts on the details that will give them the advantage. This book is tailor-made for developers who want to move from NetBeans and Eclipse to experience the power and functionality of IntelliJ IDEA.

"After many decades - and even more methodologies - software projects are still failing. Why? Managers see software development as a production line. Companies don't know how to manage software projects and hire good developers. Many developers still behave like factory workers, providing terrible service to their employers and clients. Agile was a big step forward, but not enough. What's missing? The right mindset - for both developers and their employers. As developers worldwide are recognizing, the right mindset is craftsmanship ... Mancuso explains what craftsmanship means to the developer and his or her organization, and shows how to live it every day in your real-world development environment. Mancuso shows how software craftsmanship fits with and helps you improve upon best-practice technical disciplines such as agile and lean, taking all your development projects to the next level. You'll learn how to change the disastrous perception that software developers are the same as factory workers, and that software projects can be run like factories. By placing greater professionalism, technical excellence, and customer satisfaction at the heart of what you do, you won't just deliver more value to everyone involved: you'll be happier and more fulfilled doing it"--Publisher's description.

The Complete Guide to Writing Maintainable, Manageable, Pleasing, and Powerful Object-Oriented Applications Object-oriented programming languages exist to help you create beautiful, straightforward applications that are easy to change and simple to extend. Unfortunately, the world is awash with object-oriented (OO) applications that are difficult to understand and expensive to change. Practical Object-Oriented Design, Second Edition, immerses you in an OO mindset and teaches you powerful, real-world, object-oriented design techniques with simple and practical examples. Sandi Metz demonstrates how to build new applications that can "survive success" and repair existing applications that have become impossible to change. Each technique is illustrated with extended examples in the easy-to-understand Ruby programming language, all downloadable from the companion website, poodr.com. Fully updated for Ruby 2.5, this guide shows how to Decide what belongs in a single class Avoid entangling objects that should be kept separate Define flexible interfaces among objects Reduce programming overhead costs with duck typing Successfully apply inheritance Build objects via composition Whatever your previous object-oriented experience, this concise guide will help you achieve the superior outcomes you're looking for. Register your book for convenient access to downloads, updates, and/or corrections as they become available. See inside book for details.

Delve deep into the various technical practices, principles, and values of Agile. Key Features Discover the essence of Agile software development and the key principles of software design Explore the fundamental practices of Agile working, including test-driven development (TDD), refactoring, pair programming, and continuous integration Learn and apply the four elements of simple design Book Description The number of popular technical practices has grown exponentially in the last few years. Learning the common fundamental software development practices

can help you become a better programmer. This book uses the term Agile as a wide umbrella and covers Agile principles and practices, as well as most methodologies associated with it. You'll begin by discovering how driver-navigator, chess clock, and other techniques used in the pair programming approach introduce discipline while writing code. You'll then learn to safely change the design of your code using refactoring. While learning these techniques, you'll also explore various best practices to write efficient tests. The concluding chapters of the book delve deep into the SOLID principles - the five design principles that you can use to make your software more understandable, flexible and maintainable. By the end of the book, you will have discovered new ideas for improving your software design skills, the relationship within your team, and the way your business works. What you will learn Learn the red, green, refactor cycle of classic TDD and practice the best habits such as the rule of 3, triangulation, object calisthenics, and more Refactor using parallel change and improve legacy code with characterization tests, approval tests, and Golden Master Use code smells as feedback to improve your design Learn the double cycle of ATDD and the outside-in mindset using mocks and stubs correctly in your tests Understand how Coupling, Cohesion, Connascence, SOLID principles, and code smells are all related Improve the understanding of your business domain using BDD and other principles for "doing the right thing, not only the thing right" Who this book is for This book is designed for software developers looking to improve their technical practices. Software coaches may also find it helpful as a teaching reference manual. This is not a beginner's book on how to program. You must be comfortable with at least one programming language and must be able to write unit tests using any unit testing framework.

“Mantle and Lichty have assembled a guide that will help you hire, motivate, and mentor a software development team that functions at the highest level. Their rules of thumb and coaching advice are great blueprints for new and experienced software engineering managers alike.” —Tom Conrad, CTO, Pandora “I wish I'd had this material available years ago. I see lots and lots of 'meat' in here that I'll use over and over again as I try to become a better manager. The writing style is right on, and I love the personal anecdotes.” —Steve Johnson, VP, Custom Solutions, DigitalFish All too often, software development is deemed unmanageable. The news is filled with stories of projects that have run catastrophically over schedule and budget. Although adding some formal discipline to the development process has improved the situation, it has by no means solved the problem. How can it be, with so much time and money spent to get software development under control, that it remains so unmanageable? In *Managing the Unmanageable: Rules, Tools, and Insights for Managing Software People and Teams*, Mickey W. Mantle and Ron Lichty answer that persistent question with a simple observation: You first must make programmers and software teams manageable. That is, you need to begin by understanding your people—how to hire them, motivate them, and lead them to develop and deliver great products. Drawing on their combined seventy years of software development and management experience, and highlighting the insights and wisdom of other successful managers, Mantle and Lichty provide the guidance you need to manage people and teams in order to deliver software successfully. Whether you are new to software management, or have already been working in that role, you will appreciate the real-world knowledge and practical tools packed into this guide.

This guide for programmers teaches how to practice Test Driven Development (TDD), also called Test First Development. Contrary to the accepted approach to testing, when you practice TDD you write tests for code before you write the code being tested. This text provides examples in Java.

The Practitioner's Guide to Implementing SOA with Java EE Technologies This book brings together all the practical insight you need to successfully architect enterprise solutions and implement them using SOA and Java EE technologies. Writing for senior IT developers, strategists, and enterprise architects, the authors cover everything from concepts to implementation, requirements to tools. The authors first review the Java EE platform's essential elements in the context of SOA and web services deployment, and demonstrate how Java EE has evolved into the world's best open source solution for enterprise SOA. After discussing standards such as SOAP, WSDL, and UDDI, they walk through implementing each key aspect of SOA with Java EE. Step by step, you'll learn how to integrate service-oriented web and business components of Java EE technologies with the help of process-oriented standards such as BPEL/CDL into a coherent, tiered enterprise architecture that can deliver a full spectrum of business services. *Implementing SOA Using Java™ EE* concludes with a section-length case study that walks through analyzing a company's requirements, creating an effective SOA architecture, and building a concise proof-of-concept prototype with NetBeans IDE. Coverage includes Using Java EE technologies to simplify SOA implementation Mastering messaging, service descriptions, registries, orchestration, choreography, and other essential SOA concepts Building an advanced web services infrastructure for implementing SOA Using Java Persistence API to provide for persistence Getting started with Java Business Integration (JBI), the new open specification for delivering SOA Implementing SOA at the web and business tiers Developing, configuring, and deploying SOA systems with NetBeans IDE Constructing SOA systems with NetBeans SOA Pack

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

Master Java 5.0 and TDD Together: Build More Robust, Professional Software Master Java 5.0, object-oriented design, and Test-Driven Development (TDD) by learning them together. Agile Java weaves all three into a single coherent approach to building professional, robust software systems. Jeff Langr shows exactly how Java and TDD integrate throughout the entire development lifecycle, helping you leverage today's fastest, most efficient development techniques from the very outset. Langr writes for every programmer, even those with little or no experience with Java, object-oriented development, or agile methods. He shows how to translate oral requirements into practical tests, and then how to use those tests to create reliable, high-performance Java code that solves real problems. Agile Java doesn't just teach the core features of the Java language: it presents coded test examples for each of them. This TDD-centered approach doesn't just lead to better code: it provides powerful feedback that will help you learn Java far more rapidly. The use of TDD as a learning mechanism is a landmark departure from conventional teaching techniques. Presents an expert overview of TDD and agile programming

techniques from the Java developer's perspective Brings together practical best practices for Java, TDD, and OO design Walks through setting up Java 5.0 and writing your first program Covers all the basics, including strings, packages, and more Simplifies object-oriented concepts, including classes, interfaces, polymorphism, and inheritance Contains detailed chapters on exceptions and logging, math, I/O, reflection, multithreading, and Swing Offers seamlessly-integrated explanations of Java 5.0's key innovations, from generics to annotations Shows how TDD impacts system design, and vice versa Complements any agile or traditional methodology, including Extreme Programming (XP)

In 1994, Design Patterns changed the landscape of object-oriented development by introducing classic solutions to recurring design problems. In 1999, Refactoring revolutionized design by introducing an effective process for improving code. With the highly anticipated Refactoring to Patterns , Joshua Kerievsky has changed our approach to design by forever uniting patterns with the evolutionary process of refactoring. This book introduces the theory and practice of pattern-directed refactorings: sequences of low-level refactorings that allow designers to safely move designs to, towards, or away from pattern implementations. Using code from real-world projects, Kerievsky documents the thinking and steps underlying over two dozen pattern-based design transformations. Along the way he offers insights into pattern differences and how to implement patterns in the simplest possible ways. Coverage includes: A catalog of twenty-seven pattern-directed refactorings, featuring real-world code examples Descriptions of twelve design smells that indicate the need for this book's refactorings General information and new insights about patterns and refactoring Detailed implementation mechanics: how low-level refactorings are combined to implement high-level patterns Multiple ways to implement the same pattern—and when to use each Practical ways to get started even if you have little experience with patterns or refactoring Refactoring to Patterns reflects three years of refinement and the insights of more than sixty software engineering thought leaders in the global patterns, refactoring, and agile development communities. Whether you're focused on legacy or "greenfield" development, this book will make you a better software designer by helping you learn how to make important design changes safely and effectively. Get past the myths of testing in agile environments - and implement agile testing the RIGHT way. * * For everyone concerned with agile testing: developers, testers, managers, customers, and other stakeholders. * Covers every key issue: Values, practices, organizational and cultural challenges, collaboration, metrics, infrastructure, documentation, tools, and more. * By two of the world's most experienced agile testing practitioners and consultants. Software testing has always been crucial, but it may be even more crucial in agile environments that rely heavily on repeated iterations of software capable of passing tests. There are, however, many myths associated with testing in agile environments. This book helps agile team members overcome those myths -- and implement testing that truly maximizes software quality and value. Long-time agile testers Lisa Crispin and Janet Gregory offer powerful insights for three large, diverse groups of readers: experienced testers who are new to agile; members of newly-created agile teams who aren't sure how to perform testing or work with testers; and test/QA managers whose development teams are implementing agile. Readers will learn specific agile testing practices and techniques that can mean the difference between success and failure; discover how to transition 'traditional' test teams to agile; and learn how to integrate testers smoothly into agile teams. Drawing on extensive experience, the authors illuminate topics ranging from culture to test planning to automated tools. They cover every form of testing: business-facing tests, technology-facing tests, exploratory tests, context-driven and scenario tests, load, stability, and endurance tests, and more. Using this book's techniques, readers can improve the effectiveness and reduce the risks of any agile project or initiative.

This deck of index cards is arranged in four sections: concepts, planning, teamwork and coding. The front of the card lists the things you need to know and the back provides further detail.

Summary The Well-Grounded Java Developer offers a fresh and practical look at new Java 7 features, new JVM languages, and the array of supporting technologies you need for the next generation of Java-based software. About the Book The Well-Grounded Java Developer starts with thorough coverage of Java 7 features like try-with-resources and NIO.2. You'll then explore a cross-section of emerging JVM-based languages, including Groovy, Scala, and Clojure. You will find clear examples that are practical and that help you dig into dozens of valuable development techniques showcasing modern approaches to the dev process, concurrency, performance, and much more. Written for readers familiar with Java. No experience with Java 7 or new JVM languages required. 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. What's Inside New Java 7 features Tutorials on Groovy, Scala, and Clojure Discovering multicore processing and concurrency Functional programming with new JVM languages Modern approaches to testing, build, and CI Table of Contents PART 1 DEVELOPING WITH JAVA 7 Introducing Java 7 New I/O PART 2 VITAL TECHNIQUES Dependency Injection Modern concurrency Class files and bytecode Understanding performance tuning PART 3 POLYGLOT PROGRAMMING ON THE JVM Alternative JVM languages Groovy: Java's dynamic friend Scala: powerful and concise Clojure: safer programming PART 4 CRAFTING THE POLYGLOT PROJECT Test-driven development Build and continuous integration Rapid web development Staying well-grounded

"Naked Objects is the embodiment of the Agile movement: lean, elegant, user-focused, and with testing built right in. Reduce a problem to its bare essentials, code it up with no extra fluff, then ship it out. Naked Objects brings programming back to its real purpose: expressing and solving business problems." Dave Thomas, co-author, The Agile Manifesto and The Pragmatic Programmer "I believe that this could be a landmark book. Naked Objects may well herald the next major evolution in the way systems are presented to end users, and how they're developed. Naked Objects adds near-instant prototyping to the business modeller's toolbox." Oliver Sims, co-author, Business Component Factory "A well-written description of a radical new approach to OO programming." James W Cooper, IBM T J Watson Research Center "Naked Objects is a bold approach. If you want to push the envelope and let end-users access their business objects without cluttered interfaces, read this book." Rebecca Wirfs-Brock, co-author, Object Design An object should completely model the behaviour of that which it represents. This principle of 'behaviourally complete' objects is the driving force behind this book. Naked Objects is a Java-based open source framework that exposes behaviourally complete business objects such as Customer, Product and Order, directly to the user - without the need for scripts, controllers or even dialog boxes in between. The resulting systems are empowering for the user and immensely agile. With Naked Objects the user presentation is generated automatically from the business object definitions, so you need never write another line of code for a user interface again! This book, written for business object modellers and Java developers, includes: an introduction to designing systems from naked objects a tutorial on programming with the Naked Objects framework a lightweight methodology case studies on business applications

Agile Estimating and Planning is the definitive, practical guide to estimating and planning agile projects. In this book, Agile Alliance cofounder Mike Cohn discusses the philosophy of agile estimating and planning and shows you exactly how to get the job done, with real-world examples and case studies. Concepts are clearly illustrated and

readers are guided, step by step, toward how to answer the following questions: What will we build? How big will it be? When must it be done? How much can I really complete by then? You will first learn what makes a good plan-and then what makes it agile. Using the techniques in Agile Estimating and Planning , you can stay agile from start to finish, saving time, conserving resources, and accomplishing more. Highlights include: Why conventional prescriptive planning fails and why agile planning works How to estimate feature size using story points and ideal days--and when to use each How and when to re-estimate How to prioritize features using both financial and nonfinancial approaches How to split large features into smaller, more manageable ones How to plan iterations and predict your team's initial rate of progress How to schedule projects that have unusually high uncertainty or schedule-related risk How to estimate projects that will be worked on by multiple teams Agile Estimating and Planning supports any agile, semiagile, or iterative process, including Scrum, XP, Feature-Driven Development, Crystal, Adaptive Software Development, DSDM, Unified Process, and many more. It will be an indispensable resource for every development manager, team leader, and team member.

"This set of books represents a detailed compendium of authoritative, research-based entries that define the contemporary state of knowledge on technology"--Provided by publisher.

If you're familiar with functional programming basics and want to gain a much deeper understanding, this in-depth guide takes you beyond syntax and demonstrates how you need to think in a new way. Software architect Neal Ford shows intermediate to advanced developers how functional coding allows you to step back a level of abstraction so you can see your programming problem with greater clarity. Each chapter shows you various examples of functional thinking, using numerous code examples from Java 8 and other JVM languages that include functional capabilities. This book may bend your mind, but you'll come away with a much better grasp of functional programming concepts.

Understand why many imperative languages are adding functional capabilities Compare functional and imperative solutions to common problems Examine ways to cede control of routine chores to the runtime Learn how memoization and laziness eliminate hand-crafted solutions Explore functional approaches to design patterns and code reuse View real-world examples of functional thinking with Java 8, and in functional architectures and web frameworks Learn the pros and cons of living in a paradigmatically richer world If you're new to functional programming, check out Josh Backfield's book Becoming Functional.

Learning a complex new language is no easy task especially when it s an object-oriented computer programming language like Java. You might think the problem is your brain. It seems to have a mind of its own, a mind that doesn't always want to take in the dry, technical stuff you're forced to study. The fact is your brain craves novelty. It's constantly searching, scanning, waiting for something unusual to happen. After all, that's the way it was built to help you stay alive. It takes all the routine, ordinary, dull stuff and filters it to the background so it won't interfere with your brain's real work--recording things that matter. How does your brain know what matters? It's like the creators of the Head First approach say, suppose you're out for a hike and a tiger jumps in front of you, what happens in your brain? Neurons fire. Emotions crank up. Chemicals surge. That's how your brain knows. And that's how your brain will learn Java. Head First Java combines puzzles, strong visuals, mysteries, and soul-searching interviews with famous Java objects to engage you in many different ways. It's fast, it's fun, and it's effective. And, despite its playful appearance, Head First Java is serious stuff: a complete introduction to object-oriented programming and Java. You'll learn everything from the fundamentals to advanced topics, including threads, network sockets, and distributed programming with RMI. And the new. second edition focuses on Java 5.0, the latest version of the Java language and development platform. Because Java 5.0 is a major update to the platform, with deep, code-level changes, even more careful study and implementation is required. So learning the Head First way is more important than ever. 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. You'll see why people say it's unlike any other Java book you've ever read. By exploiting how your brain works, Head First Java compresses the time it takes to learn and retain--complex information. Its unique approach not only shows you what you need to know about Java syntax, it teaches you to think like a Java programmer. If you want to be bored, buy some other book. But if you want to understand Java, this book's for you.

Agile Java™Crafting Code with Test-Driven DevelopmentPearson Education

[Copyright: d83f1e25858b88a0fc37a24fcfa8d1f8](https://www.pearson.com/9780132034237/agile-java-crafting-code-with-test-driven-development)