Google Java Code Style Guide

Kotlin is a statically typed programming language designed to interoperate with Java and fully supported by Google on the Android operating system. Based on Big Nerd Ranch’s popular Kotlin Essentials course, this guide shows you how to work effectively with the Kotlin programming language through hands-on examples and clear explanations of key Kotlin concepts and foundational APIs. Written for Kotlin 1.2, this book will also introduce you to JetBrains’ IntelliJ IDEA development environment. Whether you are an experienced Android developer looking for modern features beyond what Java offers or a new developer ready to learn your first programming language, the authors will guide you from first principles to advanced usage of Kotlin. By the end of this book, you will be empowered to create reliable, concise applications in Kotlin.

It takes a week to travel the 8,000 miles overland from Java to Kotlin. If you're an experienced Java developer who has tried the Kotlin language, you were probably productive in about the same time. You'll have found that they do things differently in Kotlin, though. Nullability is important, collections are different, and classes are final by default. Kotlin is more functional, but what does that mean, and how should it change the way that you program? And what about all that Java code that you still have to support? Your tour guides Duncan and Nat first made the trip in 2015, and they've since helped many teams and individuals follow in their footsteps. Travel with them as they break the route down into legs like Optional to Nullable, Beans to Values, and Open to Sealed Classes. Each explains a key concept and then shows how to refactor production Java to idiomatic Kotlin, gradually and safely, while maintaining interoperability. The resulting code is simpler, more expressive, and easier to change. By the end of the journey, you'll be confident in refactoring Java to Kotlin, writing Kotlin from scratch, and managing a mixed language codebase as it evolves over time.

?????? ?????? ????? ???????????????? ??????????? ??? ???????????????? ?????????? ???????????? ??????????. ????? ??? ??????????? – ??????, ???????, ??????? ???????? ????????????. ???????? ?????????? ? ???????????? ??? ??????? ????????????? ???????????? ??????????? ? ?????? ????????????-????????????? ????????????. ???????? ??? ?? ???????, ???????? ??????????? ???????? ???? ???????????? ? ????????. ?? "?????" – ??????????, ???????? ? ???????????, ???????????? ??? ???????? ????????????? ???????????? ????????.

A classic reference book on user interface design and graphic design for web sites, updated to reflect a rapidly changing market Consistently praised as the best volume on classic elements of web site design, Web Style Guide has sold many thousands of copies and has been published around the world. This new revised edition confirms Web Style Guide as the go-to authority in a rapidly changing market. As web designers move from building sites from scratch to using content management and aggregation tools, the book's focus shifts away from code samples and toward best practices, especially those involving mobile experience, social media, and accessibility. An ideal reference for web site designers in corporations, government, nonprofit organizations, and academic institutions, the book explains established design principles and covers all aspects of web design--from planning to

production to maintenance. The guide also shows how these principles apply in web design projects whose primary concerns are information design, interface design, and efficient search and navigation.

You may have definite ideas about writing code when working alone, but team development requires that everyone use the same approach. With the JavaScript practices in this book—including code style, programming tips, and automation—you will learn how to write maintainable code that other team members can easily understand, adapt, and extend. Author Nicholas Zakas assembled this collection of best practices as a front-end tech leader at Yahoo!, after completing his own journey from solo hacker to team player. He also includes rules recommended by other industry authorities. Use these tips and techniques to help your team set aside individual preferences and function at a higher level. Establish specific code conventions for your team Use tools such as JSLint and JSHint to keep your team on track Adopt style guidelines, such as basic formatting, to help your team produce uniform code Apply several programming practices to solve problems and improve code quality Create an automated JavaScript build system using a variety of utilities Integrate browser-based JavaScript testing with tools such as the YUI Test Selenium Driver Document the architecture of your software easily with this highly practical, open-source template. Key Features Get to grips with leveraging the features of arc42 to create insightful documents Learn the concepts of software architecture documentation through real-world examples Discover techniques to create compact, helpful, and easy-to-read documentation Book Description When developers document the architecture of their systems, they often invent their own specific ways of articulating structures, designs, concepts, and decisions. What they need is a template that enables simple and efficient software architecture documentation. arc42 by Example shows how it's done through several real-world examples. Each example in the book, whether it is a chess engine, a huge CRM system, or a cool web system, starts with a brief description of the problem domain and the quality requirements. Then, you'll discover the system context with all the external interfaces. You'll dive into an overview of the solution strategy to implement the building blocks and runtime scenarios. The later chapters also explain various cross-cutting concerns and how they affect other aspects of a program. What you will learn Utilize arc42 to document a system's physical infrastructure Learn how to identify a system's scope and boundaries Break a system down into building blocks and illustrate the relationships between them Discover how to describe the runtime behavior of a system Know how to document design decisions and their reasons Explore the risks and technical debt of your system Who this book is for This book is for software developers and solutions architects who are looking for an easy, open-source tool to document their systems. It is a useful reference for those who are already using arc42. If you are new to arc42, this book is a great learning resource. For those of you who want to write better technical documentation will benefit from the general concepts covered in this book.

This tutorial teaches you how to use the statistical programming language R to develop a business case simulation and analysis. It presents a methodology for conducting business case analysis that minimizes decision delay by focusing stakeholders on what matters most and suggests pathways for minimizing the risk in strategic and capital allocation decisions. Business case analysis, often conducted in spreadsheets, exposes decision makers to additional risks that arise just from the use of the spreadsheet

environment. R has become one of the most widely used tools for reproducible quantitative analysis, and analysts fluent in this language are in high demand. The R language, traditionally used for statistical analysis, provides a more explicit, flexible, and extensible environment than spreadsheets for conducting business case analysis. The main tutorial follows the case in which a chemical manufacturing company considers constructing a chemical reactor and production facility to bring a new compound to market. There are numerous uncertainties and risks involved, including the possibility that a competitor brings a similar product online. The company must determine the value of making the decision to move forward and where they might prioritize their attention to make a more informed and robust decision. While the example used is a chemical company, the analysis structure it presents can be applied to just about any business decision, from IT projects to new product development to commercial real estate. The supporting tutorials include the perspective of the founder of a professional service firm who wants to grow his business and a member of a strategic planning group in a biomedical device company who wants to know how much to budget in order to refine the quality of information about critical uncertainties that might affect the value of a chosen product development pathway. What You'll Learn Set up a business case abstraction in an influence diagram to communicate the essence of the problem to other stakeholders Model the inherent uncertainties in the problem with Monte Carlo simulation using the R language Communicate the results graphically Draw appropriate insights from the results Develop creative decision strategies for thorough opportunity cost analysis Calculate the value of information on critical uncertainties between competing decision strategies to set the budget for deeper data analysis Construct appropriate information to satisfy the parameters for the Monte Carlo simulation when little or no empirical data are available Who This Book Is For Financial analysts, data practitioners, and risk/business professionals; also appropriate for graduate level finance, business, or data science students

Summary Serious developers know that code can always be improved. With each iteration, you make optimizations—small and large—that can have a huge impact on your application's speed, size, resilience, and maintainability. In Seriously Good Software: Code that Works, Survives, and Wins, author, teacher, and Java expert Marco Faella teaches you techniques for writing better code. You'll start with a simple application and follow it through seven careful refactorings, each designed to explore another dimension of quality. Purchase of the print book includes a free eBook in PDF, Kindle, and ePub formats from Manning Publications. About the technology Great code blends the skill of a programmer with the time-tested techniques and best practices embraced by the entire development community. Although each application has its own context and character, some dimensions of quality are always important. This book concentrates on seven pillars of seriously good software: speed, memory usage, reliability, readability, thread safety, generality, and elegance. The Java-based examples demonstrate techniques that apply to any OO language. About the book Seriously Good Software is a handbook for any professional developer serious about improving application quality. It explores fundamental dimensions of code quality by enhancing a simple implementation into a robust, professional-quality application. Questions, exercises, and Java-based examples ensure you'll get a firm grasp of the concepts as you go. When you finish the last version of the book's central project, you'll be able to confidently choose the right optimizations

for your code. What's inside Evaluating software qualities Assessing trade-offs and interactions Fulfilling different objectives in a single task Java-based exercises you can apply in any OO language About the reader For developers with basic object-oriented programming skills and intermediate Java skills. About the author Marco Faella teaches advanced programming at a major Italian university. His published work includes peer-reviewed research articles, a Java certification manual, and a video course. Table of Contents *Part 1: Preliminaries * 1 Software qualities and a problem to solve 2 Reference implementation *Part 2: Software Qualities* 3 Need for speed: Time efficiency 4 Precious memory: Space efficiency 5 Self-conscious code: Reliability through monitoring 6 Lie to me: Reliability through testing 7 Coding aloud: Readability 8 Many cooks in the kitchen: Thread safety 9 Please recycle: Reusability

With this practical book, architects, CTOs, and CIOs will learn a set of patterns for the practice of architecture, including analysis, documentation, and communication. Author Eben Hewitt shows you how to create holistic and thoughtful technology plans, communicate them clearly, lead people toward the vision, and become a great architect or Chief Architect. This book covers each key aspect of architecture comprehensively, including how to incorporate business architecture, information architecture, data architecture, application (software) architecture together to have the best chance for the system’s success. Get a practical set of proven architecture practices focused on shipping great products using architecture Learn how architecture works effectively with development teams, management, and product management teams through the value chain Find updated special coverage on machine learning architecture Get usable templates to start incorporating into your teams immediately Incorporate business architecture, information architecture, data architecture, and application (software) architecture together

Java Concepts: Late Objects, 3rd Edition focuses on the essentials of effective learning and is suitable for a two-semester introduction to programming sequence. This text requires no prior programming experience and only a modest amount of high school algebra. It provides an approachable introduction to fundamental programming techniques and design skills, helping students master basic concepts and become competent coders. Each important concept is introduced in easy-to understand terms before more complicated examples are discussed. Choosing the enhanced eText format allows students to develop their coding skills using targeted, progressive interactivities designed to integrate with the eText. All sections include built-in activities, open-ended review exercises, programming exercises, and projects to help students practice programming and build confidence. These activities go far beyond simplistic multiple-choice questions and animations. They have been designed to guide students along a learning path for mastering the complexities of programming. Students demonstrate comprehension of programming structures, then practice programming with simple steps in scaffolded settings, and finally write complete, automatically graded programs. The perpetual access VitalSource Enhanced eText, when integrated with your school's learning management system, provides the capability to monitor student progress in VitalSource SCORECenter and track grades for homework or participation. *Enhanced eText and interactive functionality available through select vendors and may require LMS integration approval for SCORECenter.

Straight from IBM: complete, proven guidelines for writing consistent, clear, concise, consumable, reusable, and easy to- translate

content Brings together everything IBM has learned about writing outstanding technical and business content.

I am not a recruiter. I am a software engineer. And as such, I know what it's like to be asked to whip up brilliant algorithms on the spot, and then write flawless code on a whiteboard. I know because I've been asked to do the same thing--in interviews at Google, Microsoft, Apple, and Amazon, among other companies. According to the Last year and this year Data that we have collected from different sources, More than 5,67,000 students and IT professionals gone through this book and Successfully secured their jobs in IT industry and Other industries as well. I also know because I've been on the other side of the table, asking candidates to do this. I've combed through stacks of resumes to find the engineers who I thought might be able to actually pass these interviews. And I've debated in Google's Hiring Committee whether or not a candidate did well enough to merit an offer. I understand and have experienced the full hiring circle. And you, reader, are probably preparing for an interview, perhaps tomorrow, next week, or next year. You likely have or are working towards a Computer Science or related degree. I am not here to re-teach you the basics of what a binary search tree is, or how to traverse a linked list. You already know such things, and if not, there are plenty of other resources to learn them. This book is here to help you take your understanding of Computer Science fundamentals to the next level, to help you apply those fundamentals to crack the coding interview. Because while the fundamentals are necessary to land one of the top jobs, they aren't always enough. For countless readers, this book has been just what they needed. Cracking The Java Coding Interview 2014 Edition: Total +1000 Java Programming Questions and Solutions (Java/J2EE Including +1000 Questions & Answers 4 Every step of Interview Process) The full list of topics are as follows: ========== The Interview Process This section offers an overview on questions are selected and how you will be evaluated. What happens when you get a question wrong? When should you start preparing, and how? What language should you use? Behind the Scenes Learn what happens behind the scenes during your interview, how decisions really get made, who you interview with, and what they ask you. Companies covered include Google, Amazon, Yahoo, Microsoft, Apple and Facebook. Special Situations This section explains the process for experience candidates, Program Managers, Dev Managers, Testers / SDETs, and more. Learn what your interviewers are looking for and how much code you need to know. Before the Interview In order to ace the interview, you first need to get an interview. This section describes what a software engineer's resume should look like and what you should be doing well before your interview. Behavioral Preparation Although most of a software engineering interview will be technical, behavioral questions matter too. This section covers how to prepare for behavioral questions and how to give strong, structured responses. 5The Apple Interview. 6The Google Interview. 7The Microsoft Interview 8The Yahoo Interview 9The Facebook Interview 10Before The Interview 11Interview Frequently Asked Questions 12How To Prepare for Technical Questions 13Handling Technical Questions 14Top Ten Mistakes Candidates Make 15Special Advice for Software Design Engineers 16The Sixteen Most Revealing Interview Questions 17Before The Danger Java Interview 18Java Interview Questions & Answers +250 Q/A (PART-1) (B)AWT.(C)Swing.(D)RMI.(E)JSP.(F)EJB.(G)JDBC.(H)Servlets. (I)Threads. (J)Java util.(K)JMS. (L)Networking. (M)Java Coding Standards. 19Java Interview Questions & Answers +250 Q/A (PART-2) 20Java Interview Questions & Answers +250 Q/A

(PART-3) 21Java Interview Questions & Answers +250 Q/A (PART-4) 22Java Coding Standards/Code Clarity/Maintainability/DBMS Issues 23Dress/Body Appropriately Guidelines By Pictures &Grap

This book grew out of a lot of angst. Well, and wine. Put enough angst in me, and I'll start ranting. Pour in some wine, and the rants get mean—and funny. I still go back and read these posts now and then, and I always laugh. I was so mean. My angst grew out of traveling different roads than most programmers. Those roads forced me to see the world differently. Now I see all sorts of patterns that many experienced programmers fail to see—because, well, to put it bluntly, they're stuck in ruts. Over the past 25 years I've done a bunch of dramatically different types of programming, and I've also written far more code than any programmer ever should. The long roads I've traveled have basically given me a sixth sense. I see dead people. And it sucks. If you're ever unlucky enough to acquire a dreadful sixth sense, there are really only two choices: you can be angry and depressed about it, or you can laugh about it. So I try to laugh. It's hard, but I'm getting better at it. The wine helps. Practice helps, too. You need to get in the habit of laughing—at yourself, at others, at the crazy world we live in—or in time you'll just stop laughing altogether. When I first started ranting, I was the ugly American, stomping around in my posts, and essentially yelling "What the hell is wrong with all you people?" But over the next ten years or so, I like to think I've grown into more of an amateur software anthropologist. I now take cultural relativism seriously, and I try hard not to judge people who think differently from me. Of course I don't mind poking fun at them, because I don't mind people poking fun at me. And ultimately I would like to convince undecided programmers to share my view of the programming world, because programming works best if everyone nearby does it the same way. So I'll continue to argue that my view, which I've recently taken to calling "software liberalism," is a perfectly valid and perhaps even preferable way to do a lot of software development. Converting everyone to be more liberal is doomed to fail, of course. But even so, I hope I can still help people in radically different software cultures to understand each other better. I'm going to keep ranting, because it appears to be the only way to make a message sink in to a very large audience. Some people still tell me that my blog posts are too long. They tell me I could have made my "point" in under a hundred words. I have noticed that this complaint comes most often from people who disagree with me. They're really just saying they want less work to voice their disagreement. But even some folks who agree with me find the posts too long to carry their attention, and they complain too. They're missing the point, though. The posts aren't too long. You need a certain minimum "heft" to penetrate. Through years of trial and error, I've found that the best way to get a lot of people to listen to you is to tell them a story. And you can't spin a good yarn without settling in and enjoying the ride. So that's what this book is. It's really a bunch of stories. Each might take the form of an article, essay, guide, rant, or occasionally a fiction tale. But behind the structure, each one of them is sharing a story. Even if you don't always agree, I'm hoping you'll at least find the stories entertaining and, with luck, sometimes even eye-opening. The guys at Hyperink chose which of my posts to include, by and large, and they also came up with the overall chapter organization. I made a couple of tweaks, but what you're looking at is largely their vision of how to curate this stuff into a cohesive book. I think they did an admirable job. I hope you enjoy the journey as much as I did. Steve Yegge August 2012

This book takes a humorous slant on the programming practice manual by reversing the usual approach: under the pretence of teaching you how to become the world's worst programmer who generally causes chaos, the book teaches you how to avoid the kind of bad habits that introduce bugs or cause code contributions to be rejected. Why be a code monkey when you can be a chaos monkey? OK, so you want to become a terrible programmer. You want to write code that gets vigorously rejected in review. You look forward to reading feedback plastered in comments like "WTF???". Even better, you fantasize about your bug-ridden changes sneaking through and causing untold chaos in the codebase. You want to build a reputation as someone who writes creaky, messy, error-prone garbage that frustrates your colleagues. Bad Programming Practices 101 will help you achieve that goal a whole lot quicker by teaching you an array of bad habits that will allow you to cause maximum chaos. Alternatively, you could use this book to identify those bad habits and learn to avoid them. The bad practices are organized into topics that form the basis of programming (layout, variables, loops, modules, and so on). It's been remarked that to become a good programmer, you must first write 10,000 lines of bad code to get it all out of your system. This book is aimed at programmers who have so far written only a small portion of that. By learning about poor programming habits, you will learn good practices. In addition, you will find out the motivation behind each practice, so you can learn why it is considered good and not simply get a list of rules. What You'll Learn Become a better coder by learning how (not) to program Choose your tools wisely Think of programming as problem solving Discover the consequences of a program's appearance and overall structure Explain poor use of variables in programs Avoid bad habits and common mistakes when using conditionals and loops See how poor error-handling makes for unstable programs Sidestep bad practices related specifically to object-oriented programming Mitigate the effects of ineffectual and inadequate bug location and testing Who This Book Is For Those who have some practical programming knowledge (can program in at least one programming language), but little or no professional experience, which they would like to quickly build up. They are either still undergoing training in software development, or are at the beginning of their programming career. They have at most 1-2 years of professional experience.

Become efficient in both frontend and backend web development with Spring and Vue Key Features Connect application's frontend and backend with Vue, Vuex, and Spring Boot Leverage the latest web standards to enhance code performance, readability, and cross-compatibility Build secure full-stack web applications with Spring Security Book Description Building Applications with Spring 5 and Vue.js 2, with its practical approach, helps you become a full-stack web developer. As well as knowing how to write frontend and backend code, a developer has to tackle all problems encountered in the application development life cycle – starting from the simple idea of an application, to the UI and technical designs, and all the way to implementation, testing, production deployment, and monitoring. With the help of this book, you'll get to grips with Spring 5 and Vue.js 2 as you learn how to develop a web application. From the initial structuring to full deployment, you'll be guided at every step of developing a web application from scratch with Vue.js 2 and Spring 5. You'll learn how to create different components of your application as you progress through each chapter, followed by exploring different tools in these frameworks to expedite your

development cycle. By the end of this book, you'll have gained a complete understanding of the key design patterns and best practices that underpin professional full-stack web development. What you will learn Analyze requirements and design data models Develop a single-page application using Vue.js 2 and Spring 5 Practice concept, logical, and physical data modeling Design, implement, secure, and test RESTful API Add test cases to improve reliability of an application Monitor and deploy your application to production Who this book is for Building Applications with Spring 5.0 and Vue.js 2.0 is for you if you are developer who is new to Vue.js or Spring. It is assumed that you have some knowledge of HTML, CSS, and Java.

Explains how to customize the Java integrated development environment, covering navigation, terminology, extension, the plug-in architecture, and frameworks.

This edited book presents scientific results of the 4th International Conference on Applied Computing and Information Technology (ACIT 2016) which was held on December 12–14, 2016 in Las Vegas, USA. The aim of this conference was to bring together researchers and scientists, businessmen and entrepreneurs, teachers, engineers, computer users, and students to discuss the numerous fields of computer science and to share their experiences and exchange new ideas and information in a meaningful way. The aim of this conference was also to bring out the research results about all aspects (theory, applications and tools) of computer and information science, and to discuss the practical challenges encountered along the way and the solutions adopted to solve them. The conference organizers selected the best papers from those papers accepted for presentation at the conference. The papers were chosen based on review scores submitted by members of the Program Committee, and underwent further rigorous rounds of review. This book captures 11 of the conference's most promising papers, and the readers impatiently await the important contributions that they know these authors are going to bring to the field of computer and information science.

What if you could condense Java down to its very best features and build better applications with that simpler version? In this book, veteran Sun Labs engineer Jim Waldo reveals which parts of Java are most useful, and why those features make Java among the best programming languages available. Every language eventually builds up crud, Java included. The core language has become increasingly large and complex, and the libraries associated with it have grown even more. Learn how to take advantage of Java's best features by working with an example application throughout the book. You may not like some of the features Jim Waldo considers good, but they'll actually help you write better code. Learn how the type system and packages help you build large-scale software Use exceptions to make code more reliable and easier to maintain Manage memory automatically with garbage collection Discover how the JVM provides portability, security, and nearly bug-free code Use Javadoc to embed documentation within the code Take advantage of reusable data structures in the collections library Use Java RMI to move code and data in a distributed network Learn how Java concurrency constructs let you exploit multicore processors

The fourth edition of the Official (ISC)2® Guide to the SSCP CBK® is a comprehensive resource providing an in-depth look at the seven domains of the SSCP Common Body of Knowledge (CBK). This latest edition provides an updated, detailed guide that is considered one of the best tools for candidates striving to become an SSCP. The book offers step-by-step guidance through each of SSCP's domains, including best practices and techniques used by the world's most experienced practitioners. Endorsed by (ISC)2 and compiled and reviewed by SSCPs and subject matter experts, this book brings together a global, thorough perspective to not only prepare for the SSCP exam, but it also provides a reference that will serve you well into your career.

This book is the "Hello, World" tutorial for building products, technologies, and teams in a startup environment. It's based on the experiences of the author, Yevgeniy (Jim) Brikman, as well as interviews with programmers from some of the most successful startups of the last decade, including Google, Facebook, LinkedIn, Twitter, GitHub, Stripe, Instagram, AdMob, Pinterest, and many others. Hello, Startup is a practical, how-to guide that consists of three parts: Products, Technologies, and Teams. Although at its core, this is a book for programmers, by programmers, only Part II (Technologies) is significantly technical, while the rest should be accessible to technical and non-technical audiences alike. If you’re at all interested in startups—whether you’re a programmer at the beginning of your career, a seasoned developer bored with large company politics, or a manager looking to motivate your engineers—this book is for you.

First and only book on the Java 5, including new Java EE 5, for SAP/ABAB programmers The author has given the first course of its kind in Belgium, and employs his experience and approach in this book More Java development or exposure to Java needed by SAP/ABAP programmers and developers as evidenced by NetWeaver, for example

This book, first published in 2000, Illustrates rules of Java code-writing with parallel examples of correct and incorrect usage.

“This is an incredibly wise and useful book. The authors have considerable real-world experience in delivering quality systems that matter, and their expertise shines through in these pages. Here you will learn what technical debt is, what is it not, how to manage it, and how to pay it down in responsible ways. This is a book I wish I had when I was just beginning my career. The authors present a myriad of case studies, born from years of experience, and offer a multitude of actionable insights for how to apply it to your project.” –Grady Booch, IBM Fellow Master Best Practices for Managing Technical Debt to Promote Software Quality and Productivity As software systems mature, earlier design or code decisions made in the context of budget or schedule constraints increasingly impede evolution and innovation. This phenomenon is called technical debt, and practical solutions exist. In Managing Technical Debt, three leading experts introduce integrated, empirically developed principles and practices that any software professional can use to gain control of technical debt in any software system. Using real-life examples, the authors explain the forms of technical debt that afflict software-intensive systems, their root causes, and their impacts. They introduce proven approaches for identifying and assessing specific sources of technical debt, limiting new debt, and “paying off” debt over time. They describe how to establish managing technical debt as a core software engineering practice in your organization. Discover how technical debt damages manageability, quality, productivity, and morale–and what you can do about it Clarify root causes of debt, including the linked roles of business goals, source code, architecture, testing, and infrastructure Identify technical debt items, and analyze their costs so you can prioritize action Choose the right solution for each technical debt item: eliminate, reduce, or mitigate Integrate software engineering practices that minimize new debt Managing Technical Debt will be a valuable resource for every software professional who wants to accelerate innovation in existing systems, or build new systems that will be easier to maintain and evolve.

JBuilder Developer's Guide provides comprehensive coverage of JBuilder from the practitioner's viewpoint. The authors develop a consolidated application throughout the chapters, allowing conceptual cohesion and illustrating the use of JBuilder to build 'real-world' applications. The examples can be compiled and run under JBuilder Personal edition, a free edition of JBuilder. JBuilder Developer's Guide is not version specific but explains the latest JBuilder 6, 7, and 8 features such as enterprise J2EE application development, CORBA, SOAP, XML tools, Enterprise JavaBeans, JavaServer Pages/Servlets, and JavaBeans technology. JBuilder repeatedly wins "developer's choice" awards as the best visual tool for developing Java applications.

The Complete Coding Interview Guide in Java is an all-inclusive solution guide with meticulously crafted questions and answers that will help

you crack any Java Developer job. This book will help you build a strong foundation and the skill-set required to confidently appear in the toughest coding interviews.

This book describes the landscape of cloud computing from first principles, leading the reader step-by-step through the process of building and configuring a cloud environment. The book not only considers the technologies for designing and creating cloud computing platforms, but also the business models and frameworks in real-world implementation of cloud platforms. Emphasis is placed on “learning by doing,” and readers are encouraged to experiment with a range of different tools and approaches. Topics and features: includes review questions, hands-on exercises, study activities and discussion topics throughout the text; demonstrates the approaches used to build cloud computing infrastructures; reviews the social, economic, and political aspects of the on-going growth in cloud computing use; discusses legal and security concerns in cloud computing; examines techniques for the appraisal of financial investment into cloud computing; identifies areas for further research within this rapidly-moving field.

If you are a programmer, you need this book. You've got a day to add a new feature in a 34,000-line program: Where do you start? Page 333 How can you understand and simplify an inscrutable piece of code? Page 39 Where do you start when disentangling a complicated build process? Page 167 How do you comprehend code that appears to be doing five things in parallel? Page 132 You may read code because you have to--to fix it, inspect it, or improve it. You may read code the way an engineer examines a machine--to discover what makes it tick. Or you may read code because you are scavenging--looking for material to reuse. Code-reading requires its own set of skills, and the ability to determine which technique you use when is crucial. In this indispensable book, Diomidis Spinellis uses more than 600 real-world examples to show you how to identify good (and bad) code: how to read it, what to look for, and how to use this knowledge to improve your own code. Fact: If you make a habit of reading good code, you will write better code yourself.

"Taming CSS Complexity" is a collection of 11 CSS-packed chapters that are all about performance- and developer-friendly coding. In order to achieve a well-rounded coding experience, the Smashing Magazine authors have explored the complexity of CSS from different perspectives, balancing rather specific hands-on tips and more general coding best practices. Among other hot topics, this eBook covers how to design layouts with Flexbox, Atomic Design with Sass, and takes a look at the most common CSS issues. Experimental techniques such as the "Clown Car Technique" provide innovative approaches to new challenges, and an insight into the BEM methodology helps to improve the overall quality of front-end code. To simplify your daily coding routine, valuable tricks on how to structure and style your code have also been included in this eBook. TABLE OF CONTENTS - Semantic CSS With Intelligent Selectors - Absolute Horizontal And Vertical Centering In CSS - How To Benefit From CSS Generated Content - The Problem Of CSS Form Elements - Clown Car Technique: Solving Adaptive Images In Responsive Web Design - The "Other" Interface: Atomic Design With Sass - Simple Responsive Images With CSS Background Images - Designing CSS Layouts With Flexbox Is As Easy As Pie - The Evolution Of The BEM Methodology - Using White Space For Readability In HTML And CSS - Why Coding Style Matters

Quickly learn the ropes with the Rust programming language using this practical, step-by-step guide In Beginning Rust Programming, accomplished programmer and author Ric Messier delivers a highly practical, real-world guide to coding with Rust. Avoiding dry, theoretical content and “Hello, world”-type tutorials of questionable utility, the book dives immediately into functional Rust programming that takes advantage of the language’s blazing speed and memory efficiency. Designed from the ground up to give you a running start to using the multiparadigm system programming language, this book will teach you to: Solve real-world computer science problems of practical

importance Use Rust’s rich type system and ownership model to guarantee memory-safety and thread-safety Integrate Rust with other programming languages and use it for embedded devices Perfect for programmers with some experience in other languages, like C or C++, Beginning Rust Programming is also a great pick for students new to programming and seeking a user-friendly and robust language with which to start their coding career.

Learn how to successfully implement trustworthy computing tasks using aspect-oriented programming This landmark publication fills a gap in the literature by not only describing the basic concepts of trustworthy computing (TWC) and aspect-oriented programming (AOP), but also exploring their critical interrelationships. The author clearly demonstrates how typical TWC tasks such as security checks, in-and-out conditions, and multi-threaded safety can be implemented using AOP. Following an introduction, the book covers: Trustworthy computing, software engineering, and computer science Aspect-oriented programming and Aspect.NET Principles and case studies that apply AOP to TWC Coverage includes Aspect.NET, the AOP framework developed by the author for the Microsoft.NET platform, currently used in seventeen countries. The author discusses the basics of Aspect.NET architecture, its advantages compared to other AOP tools, and its functionality. The book has extensive practical examples and case studies of trustworthy software design and code using the Aspect.NET framework. In addition, the book explores other software technologies and tools for using AOP for trustworthy software development, including Java and AspectJ. This book also includes a valuable chapter dedicated to ERATO, the author's teaching method employed in this book, which has enabled thousands of students to quickly grasp and apply complex concepts in computing and software engineering, while the final chapter presents an overall perspective on the current state of AOP and TWC with a view toward the future. Software engineers, architects, developers, programmers, and students should all turn to this book to learn this tested and proven method to create more secure, private, and reliable computing.

The core of EPI is a collection of over 300 problems with detailed solutions, including 100 figures, 250 tested programs, and 150 variants. The problems are representative of questions asked at the leading software companies. The book begins with a summary of the nontechnical aspects of interviewing, such as common mistakes, strategies for a great interview, perspectives from the other side of the table, tips on negotiating the best offer, and a guide to the best ways to use EPI. The technical core of EPI is a sequence of chapters on basic and advanced data structures, searching, sorting, broad algorithmic principles, concurrency, and system design. Each chapter consists of a brief review, followed by a broad and thought-provoking series of problems. We include a summary of data structure, algorithm, and problem solving patterns.

"Organizations worldwide rely on Java code to perform mission-critical tasks, and therefore that code must be reliable, robust, fast, maintainable, and secure. JavaTM Coding Guidelines brings together expert guidelines, recommendations, and code examples to help you meet these demands."--Publisher description.

Every major enterprise has a significant installed base of existing software systems that reflect the tangled IT architectures that result from decades of patches and failed replacements. Most of these systems were designed to support business architectures that have changed dramatically. At best, these systems hinder agility and competitiveness and, at worst, can bring critical business functions to a halt. Architecture-Driven Modernization (ADM) restores the value of entrenched systems by capturing and retooling various aspects of existing application environments, allowing old infrastructures to deliver renewed value and align effectively with

enterprise strategies and business architectures. Information Systems Transformation provides a practical guide to organizations seeking ways to understand and leverage existing systems as part of their information management strategies. It includes an introduction to ADM disciplines, tools, and standards as well as a series of scenarios outlining how ADM is applied to various initiatives. Drawing upon lessons learned from real modernization projects, it distills the theory and explains principles, processes, and best practices for every industry. Acts as a one-stop shopping reference and complete guide for implementing various modernization models in myriad industries and departments Every concept is illustrated with real-life examples from various modernization projects, allowing you to immediately apply tested solutions and see results Authored by the Co-chair of the Object Management Group (OMG) Architecture-Driven Modernization (ADM) Task Force, which sets definitive systems modernization standards for the entire IT industry A web site supports the book with up to date coverage of evolving ADM Specifications, Tutorials, and Whitepapers, allowing you to remain up to date on modernization topics as they develop

Today, software engineers need to know not only how to program effectively but also how to develop proper engineering practices to make their codebase sustainable and healthy. This book emphasizes this difference between programming and software engineering. How can software engineers manage a living codebase that evolves and responds to changing requirements and demands over the length of its life? Based on their experience at Google, software engineers Titus Winters and Hyrum Wright, along with technical writer Tom Manshreck, present a candid and insightful look at how some of the world’s leading practitioners construct and maintain software. This book covers Google’s unique engineering culture, processes, and tools and how these aspects contribute to the effectiveness of an engineering organization. You’ll explore three fundamental principles that software organizations should keep in mind when designing, architecting, writing, and maintaining code: How time affects the sustainability of software and how to make your code resilient over time How scale affects the viability of software practices within an engineering organization What trade-offs a typical engineer needs to make when evaluating design and development decisions

Software Development is the most thorough, realistic guide to "what works" in software development - and how to make it happen in your organization. Leading consultant Marc Hamilton tackles all three key elements of successful development: people, processes, and technology. From streamlining infrastructures to retraining programmers, choosing tools to implementing service level agreements, Hamilton unifies all of today's best practices - in management, architecture, and software engineering.

The Elements of Java Style, written by renowned author Scott Ambler, Alan Vermeulen, and a team of programmers from Rogue Wave Software, is directed at anyone who writes Java code. Many books explain the syntax and basic use of Java; however, this essential guide explains not only what you can do with the syntax, but what you ought to do. Just as Strunk and White's The Elements of Style provides rules of usage for the English language, this text furnishes a set of rules for Java practitioners. While illustrating these rules with parallel examples of correct and incorrect usage, the authors offer a collection of standards, conventions, and guidelines for writing solid Java code that will be easy to understand, maintain, and enhance. Java developers and programmers who read this book will write better Java code, and become more productive as well. Indeed, anyone who writes

Java code or plans to learn how to write Java code should have this book next to his/her computer.
This textbook offers undergraduate students an introduction to the main principles and some of the most popular techniques that constitute ‘software quality assurance’. The book seeks to engage students by placing an emphasis on the underlying foundations of modern quality-assurance techniques , using these to highlight why techniques work, as opposed to merely focussing on how they work. In doing so it provides readers with a comprehensive understanding of where software quality fits into the development lifecycle (spoiler: everywhere), and what the key quality assurance activities are. The book focuses on quality assurance in a way that typical, more generic software engineering reference books do not. It is structured so that it can (and should) be read from cover to cover throughout the course of a typical university module. Specifically, it is Concise: it is small enough to be readable in its entirety over the course of a typical software engineering module. Explanatory: topics are discussed not merely in terms of what they are, but also why they are the way they are – what events, technologies, and individuals or organisations helped to shape them into what they are now. Applied: topics are covered with a view to giving the reader a good idea of how they can be applied in practice, and by pointing, where possible, to evidence of their efficacy. The book starts from some of the most general notions (e.g. quality and development process), and gradually homes-in on the more specific activities, assuming knowledge of the basic notions established in prior chapters. Each chapter concludes with a “Key Points” section, summarising the main issues that have been covered in the chapter. Throughout the book there are exercises that serve to remind readers of relevant parts in the book that have been covered previously, and give them the opportunity to reflect on a particular topic and refer to related references.