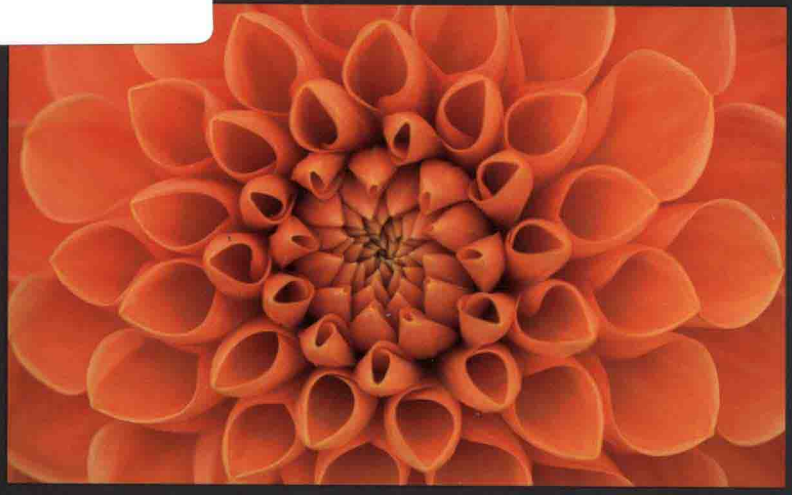
PEARSON



Java编码指南

编写安全可靠程序的75条建议

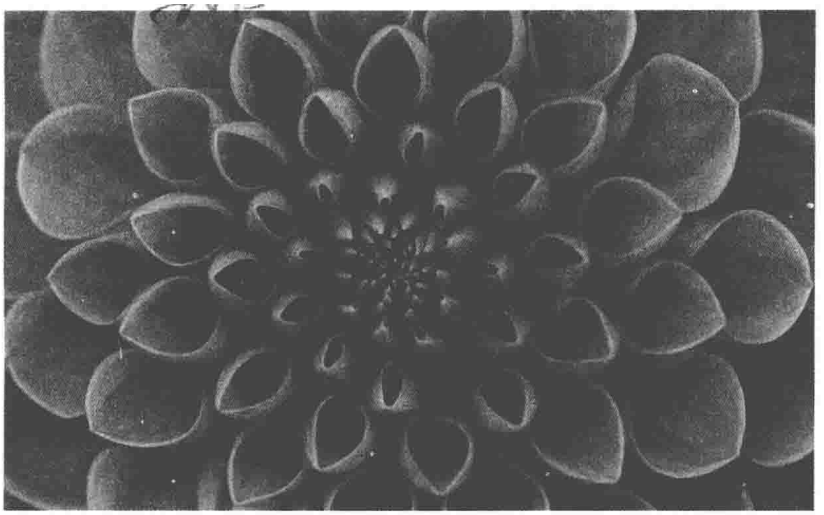
（英文版）

[美] Fred Long Dhruv Mohindra Robert C. Seacord
Dean F. Sutherland David Svoboda 著

Java Coding Guidelines

75 Recommendations for Reliable and Secure Programs

中国工信出版集团 人民邮电出版社
POSTS & TELECOM PRESS



Java编码指南

编写安全可靠程序的75条建议

（英文版）

[美] Fred Long Dhruv Mohindra Robert C. Seacord
Dean F. Sutherland David Svoboda 著

Java Coding Guidelines

75 Recommendations for Reliable and Secure Programs

人民邮电出版社
北京

图书在版编目（CIP）数据

Java编码指南 : 编写安全可靠程序的75条建议 = Java Coding Guidelines: 75 Recommendations for Reliable and Secure Programs : 英文 / (美) 朗 (Long,F.) 等著. -- 北京 : 人民邮电出版社, 2015.10
ISBN 978-7-115-40401-5

Ⅰ. ①J… Ⅱ. ①朗… Ⅲ. ①JAVA语言－程序设计－英文 Ⅳ. ①TP312

中国版本图书馆CIP数据核字(2015)第222143号

内 容 提 要

本书是《Java 安全编码标准》一书的扩展，书中把那些不必列入 Java 安全编码标准但是同样会导致系统不可靠或不安全的 Java 编码实践整理出来，并为这些糟糕的实践提供了相应的文档和警告，以及合规的解决方案。读者可以将本书作为 Java 安全方面的工具书，根据自己的需要，找到自己感兴趣的规则进行阅读和理解，或者在实际开发中遇到安全问题时，根据书中列出的大致分类对规则进行索引和阅读，也可以通读全书的所有规则，系统地了解 Java 安全规则，增强对 Java 安全特性、语言使用、运行环境特性的理解。

本书给出了帮助 Java 软件工程师设计出高质量的、安全的、可靠的、强大的、有弹性的、可用性和可维护性高的软件系统的 75 条编码指南，适合所有 Java 开发人员阅读，也适合高等院校教师和学生学习和参考。

◆ 著　[美] Fred Long　Dhruv Mohindra　Robert C. Seacord
Dean F. Sutherland　David Svoboda

责任编辑　杨海玲

责任印制　张佳莹　焦志炜

◆ 人民邮电出版社出版发行　北京市丰台区成寿寺路 11 号

邮编　100164　电子邮件　315@ptpress.com.cn

网址　http://www.ptpress.com.cn

固安县铭成印刷有限公司印刷

◆ 开本：720×960　1/16

印张：18

字数：298 千字　2015 年 10 月第 1 版

印数：1－2 000 册　2015 年 10 月河北第 1 次印刷

著作权合同登记号　图字：01-2015-6171 号

定价：59.00 元

读者服务热线：(010)81055410　印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京崇工商广字第 0021 号

版权声明

To my late wife, Ann, for all her love, help, and support over the years.
—Fred Long

To my parents, Deepak and Eta Mohindra, my grandmother Shashi Mohindra, and our very peppy, spotted Dalmatian, Google.
—Dhruv Mohindra

To my wife, Alfie, for making this book worthwhile, and to my parents, Bill and Lois, for making it possible.
—David Svoboda

To my wife, Rhonda, and our children, Chelsea and Jordan.
—Robert C. Seacord

For Libby, who makes everything worthwhile.
—Dean Sutherland

序[1]

——*James A. Gosling*

作为《The CERT® Oracle® Secure Coding Standard for Java™》的一个延伸，本书是非常有价值的，它甚至可以命名为《可靠的 Java 编码指南》（Reliable Java™ Coding Guidelines）。这些年来，可靠性和安全性之间的相互影响深深地触动了我。现如今，虽然有各种各样的显式的安全手段（如加密、身份验证等）用以确保系统安全，但是大多数的漏洞都来自于开发中的失误：编码太差或防御不足。当我们说要构建一个可靠的系统时，在很大程度上等同于构建一个安全的系统。系统的安全性将会得益于你对系统可靠性所做的一切，反之亦然。

本书强调了这样一个事实：我们所谓的安全性其实不是一个特性，而是一种针对所有的潜在的不安全因素都予以充分考虑的态度。安全性应该被持续贯穿在每一位软件工程师的设计思考过程中。它的基础是一系列的编码指南。本书最精彩的地方就是这些编码指南背后的微妙之处。例如，“用散列函数存储密码”，这看上去是一件很基础很明显的事情，然而经常有新闻报道说，由于程序员没有考虑到密码的加密而导致重要数据泄露。大量的细节之处成为了安全隐患的藏身之所，这让系统安全变得很棘手。本书充满了处理这些细节的出色指导。

1. 本序由刘先宁翻译。

前言[1]

本书为 Java 程序员提供了具体的建议。这些 Java 编码指南的应用将会带来更健壮、更能抵御攻击的系统。这些编码指南覆盖范围广泛，适用于大多数基于 Java 编写的运行在不同设备上的产品，这些设备包括电脑、游戏机、手机、平板电脑、家用电器和汽车电子设备。

不管是哪一门编程语言，开发人员在控制程序结构时都应遵守一系列基于该语言特定规则的指南。Java 程序员也理应如此。

为了编写安全可靠的 Java 程序，Java 程序员需要很多的帮助，单凭 Java 语言规范（Java Language Specification，JLS）[JLS 2013] 是远远不够的。由于 Java 包含的许多语言特性和 API 很容易被误用，因此需要一些必要的避免这些陷阱的指导。

对于一个程序来说，可靠意味着在所有场景或所有可能输入条件下均能正常工作。不可避免的是，任何重要的程序都会遇到一些完全意想不到的输入或场景，从而发生错误。当此类错误发生时，最重要的是它产生的影响必须是有限的，而这可以通过快速定位错误并尽快处理它来实现。预期不寻常的输入或编程场景，并采用防御式编程方式，程序员会受益良多。

其中一些指南可能被认为是一种编码风格，但对于代码的可读性和可维护性来说，它们仍然很重要。针对 Java 语言，Oracle 公司提供了一组编码约定 [Conventions 2009] 来帮助程序员编写具有一致编程风格的代码，这些约定已经被 Java 程序员广泛采用。

1. 本前言由刘先宁翻译。

■《The CERT® Oracle® Secure Coding Standard for Java™》

本书由《The CERT® Oracle® Secure Coding Standard for Java™》[Long 2012]一书的作者编写。该编码标准提供了一组针对 Java 语言的安全编码规则，目的是消除那些可能导致安全隐患的不安全编码实践。该安全编码标准为软件系统建立了规范的需求，同时也可以用来评估软件系统的一致性，例如，使用《Source Code Analysis Laboratory (SCALe)》[Seacord 2012] 来检测软件系统的一致性。不过，有些不必列入 Java 安全编码标准的、糟糕的 Java 编码实践，也会导致不可靠或不安全的程序。本书针对这样的编码实践提供了相应的文档和警告。

虽然这些编码指南没有出现在《The CERT® Oracle® Secure Coding Standard for Java™》[Long 2012] 中，但是它们的重要性却不应该被忽视。当一个编码指南不能构成一个规范需求时，是不能被收录到编码标准里的。无法构成规范需求的原因有很多，可能最常见的原因是，规则取决于程序员的意图。这些规则不能自动应用，除非程序员有特定的意图，在这种情况下，代码和特定的意图是需要保持一致的。为了形成一个规范的需求，需要证明的是，如果不遵守这一需求将会导致代码缺陷。有些编码指南已经被排除在编码标准之外（但包括在本书当中），遵守这些指南进行编码始终是一个好主意，但违反这些指南也并不总是会导致错误的结果。之所以会有这样的区别，是因为如果该系统没有因此产生特定的缺陷，我们不能说它不合格。因此，编码规则必须有非常严格的定义。而编码指南往往会对安全性和可靠性产生更深远的影响，因为它们可以被更概括地定义。

许多指南都参考了《The CERT® Oracle® Secure Coding Standard for Java™》里面的规则。这些引用的形式类似于“IDS01-J. Normalize strings before validating them”，其中，这个引用的前三个字母表示的是《The CERT® Oracle® Secure Coding Standard for Java ™》一书的相关章节。例如，IDS 指的是第 2 章，“Input Validation and Data Sanitization (IDS)”。

这些安全编码标准针对 Java 的具体规则也可在 CERT 的安全编码百科网站（www.securecoding.cert.org）上找到，在那里它们有持续的更新。《The CERT® Oracle® Secure Coding Standard for Java ™》提供了针对一致性测试的定义，但安全编码百科上可能有该书没有涉及的、关于这些定义的扩展信息以及见解，这些可以帮助程序员理解这些规则的意义。

本书中对于其他编码指南的交叉引用都只给出了编码指南的编号。

■ 范围

本书侧重于 Java SE 7 平台环境，同时针对在安全编码过程中由于使用 Java SE 7 API 而产生的问题进行了一些指导。Java 语言规范 Java SE 7 版（The Java Language Specification: Java SE 7 Edition, JLS）[JLS 2013] 规定了 Java 编程语言的行为，本书的这些指南主要是参考它开发出来的。

传统编程语言，如 C 和 C++，它们的语言标准里包括了一些未定义的、未指明的以及实现定义的（implementation-defined）行为，这些都容易使程序员对这些行为的可移植性作出不正确的假设，从而导致漏洞。相比之下，Java 语言规范更严格地定义了语言行为，因为 Java 是一种跨平台语言。即便如此，某些行为的自由裁量权还是留给了 Java 虚拟机（Java Virtual Machine，JVM）的实现者，或者 Java 编译器。这些指南确定出了这种语言的特点，提供了一些可以帮助开发者定位问题的解决方案，让程序员领会和理解语言的局限性并更好地利用它。

只关注语言本身并不能编写出可靠安全的软件。有时，Java API 中设计有问题的接口会被弃用。其他时候，API 或相关文档也有可能被编程社区不正确地解读。这些指南识别出了有可能被曲解的 API，并强调了它们的正确用法，常用的、有缺陷的设计模式和编程风格的示例也包括在内。

Java 语言、其核心 API 和扩展 API 以及 Java 虚拟机提供了一些安全特性，如安全管理器、访问控制器、加密解密、自动内存管理、强类型检查和字节码验证。这些特性可以为大多数应用程序提供足够的安全，但它们的正确使用却是至关重要的。这些指南不仅强调了那些与安全体系结构相关的陷阱和警告，同时强调了它的正确实现。遵守这些指南可以确保被信任程序不出现大量的可利用的安全漏洞，从而避免可能导致的拒绝服务、信息泄露、错误的计算和特权升级。

■ 包含的库

图 P-1 是 Oracle 公司 Java SE 产品的概念图。

这些编码指南主要适用于基于 `lang` 和 `util` 的库以及“其他基础库”，解决了其中的安全问题。这些编码指南没有包含那些已经标记为需要修复的公开 bug 或者那些没有负面影响的问题。某些功能性 bug 也被包含其中，它们发生频率高，可造成相当大的安全或可靠性问题，或者影响大多数依赖于核心平台的 Java 技术。

这些指南不仅包括特定于核心 API 的安全问题，还包括重要的有关标准扩展 API（javax 包）的可靠性和安全性问题。

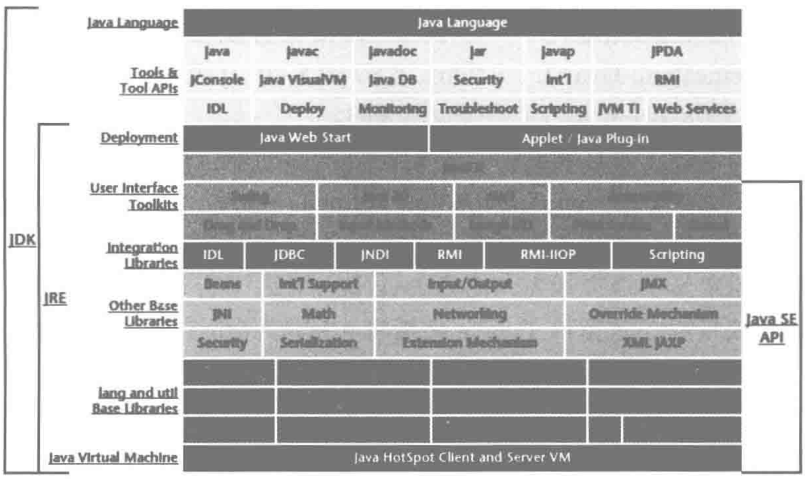


图 P-1 Oracle 公司 Java SE 产品的概念图（来自 Oracle 公司的 Java SE Documentation，http://docs.oracle.com/javase/7/docs/. Copyright © 1995, 2010, Oracle and/or its affiliates. All rights reserved.）

为了掌握 Java 提供的全方位的安全特性，程序员需要学习通过代码与其他组件和框架进行交互。偶尔，本书中的编码指南使用的示例来自于流行的 Web 应用程序框架（如 Spring 和 Struts）和流行的技术，如 Java 服务器页面（Java Server Page，JSP），通过这些示例突出安全漏洞并不是单独存在的。只有当标准 API 本身没有提供选项来避免漏洞时，第三方库和解决方案才应予以考虑。

■ 没有解决的问题

这个安全编码标准没有解决的若干问题。

内容

本书中采用的这些编码指南是广泛适用于几乎所有平台的，那些关注点在单一 Java 平台上的编码指南（例如，那些 Android、Java 微小版（ME）或 Java 企业版（EE）适用但 Java 标准版（SE）不适用的编码指南）被排除在了本书之外。另外，在 Java 标准版中，用于处理用户界面（用户界面工具包）或为 Web 界面提供特性（如声音、图像渲染、用户账户访问控制、会话管理、身份验证以及授权）的 API，也超出了本指南的范围。尽管如此，书中的这些指南讨论了网络化的 Java 系统因

不适当的输入验证而带来的相关风险，以及面临的注入式缺陷，并提供了适当的解决方案。另外，书中这些指南已假定产品的功能规范已被正确识别，同时没有来自高层设计和架构的缺陷。

编码风格

编码风格问题是主观性的，已被证明的是——在编码风格方面是不可能达成共识的。因此，本书通常尽量避免采用任何特定的编码风格。相反，我们建议用户通过这些指南定义自己的编码风格。让编码风格持续一致的最简单方法就是使用代码格式化工具。许多集成开发环境（IDE）都提供了这样的功能。

工具

这些指南并不能够自动检测或修正。在某些情况下，工具厂商可以选择实现检查器来确定代码是否违反了这些指南。作为美国联邦政府资助的研发中心（Federally Funded Research and Development Center，FFRDC），软件工程研究所（Software Engineering Institute，SEI）不适合为此推荐特定的供应商或工具。

有争议的指南

通常，本书会尽量避免包含具有争议的、缺乏广泛共识的指南。

目标读者

本书主要适用于 Java 语言程序开发人员。虽然这些指南的重点放在了 Java SE 7 平台环境上，但对于工作在 Java ME、Java EE 或其他 Java 版本平台环境上的程序员们，也具有一定的参考价值（虽然不完全）。

尽管这些指南的主要设计目的是构建安全可靠的系统，但这些指南同时也有助于提高系统的其他质量属性，如安全性、可靠性、健壮性、可用性和可维护性。

这些指南还适合于：

- 分析工具的开发者，用于诊断出不安全或不合规范的 Java 语言程序；
- 软件开发经理、软件购买者或其他软件开发专家，用于建立一套严格的安全编码标准；
- Java 编码课程的教育工作者，可作为主教材或辅助教材。

内容组织

本书的 75 条编码指南是围绕着以下原则组织的。

- 第 1 章介绍了用于确保 Java 应用程序安全性的编码指南。
- 第 2 章包含一些防御式编码指南，通过这些指南，程序员可以编写出防御性的程序。
- 第 3 章给出了提高 Java 应用程序可靠性和安全性的建议。
- 第 4 章给出了让程序更易读易懂的建议。
- 第 5 章展示一些 Java 语言和编程概念经常被误解的情形。

附录 A 描述了本书针对在 Android 平台上进行 Java 编程的适用性。本书还包含了常用术语表和参考文献。

本书中的指南均有一致的结构。标题和起始段落定义了该指南的本质。紧接着通常是由一个或多个违规代码示例及其相应的合规解决方案。每个指南的最后一个部分均给出了该指南的适用性和具体参考文献。

Acknowledgments

This book was only made possible through a broad community effort. First, we would like to thank those who contributed guidelines to this book, including Ron Bandes, Jose Sandoval Chaverri, Ryan Hall, Fei He, Ryan Hofler, Sam Kaplan, Michael Kross, Christopher Leonavicius, Bocong Liu, Bastian Marquis, Aniket Mokashi, Jonathan Paulson, Michael Rosenman, Tamir Sen, John Truelove, and Matthew Wiethoff.

The following people also contributed to this work, and their efforts are greatly appreciated: James Ahlborn, Neelkamal Gaharwar, Ankur Goyal, Tim Halloran, Sujay Jain, Pranjal Jumde, Justin Loo, Yitzhak Mandelbaum, Todd Nowacki, Vishal Patel, Justin Pincar, Abhishek Ramani, Brendon Saulsbury, Kirk Sayre, Glenn Stroz, Yozo Toda, and Shishir Kumar Yadav. We would also like to thank Hiroshi Kumagai and JPCERT for their work on the Android appendix.

We would also like to thank the following reviewers: Thomas Hawtin, Dan Plakosh, and Steve Scholnick.

We would also like to thank SEI and CERT managers who encouraged and supported our efforts: Archie Andrews, Rich Pethia, Greg Shannon, and Bill Wilson.

Thanks also to our editor Peter Gordon and his team at Addison-Wesley: Kim Boedigheimer, Jennifer Bortel, John Fuller, Stephane Nakib, and Julie Nahil. Thanks also to project editor Anna Popick and copy editor Melinda Rankin.

We thank the remainder of the CERT team for their support and assistance, without which this book could not have been completed. And last but not least, we would like to thank our in-house editor, Carol J. Lallier, who helped make this work possible.

About the Authors

Fred Long is a senior lecturer in the Department of Computer Science at Aberystwyth University in the United Kingdom. He lectures on formal methods; Java, C++, and C programming; and programming-related security issues. He is chairman of the British Computer Society's mid-Wales branch. Fred has been a visiting scientist at the Software Engineering Institute since 1992. Recently, his research has involved the investigation of vulnerabilities in Java. Fred is a coauthor of *The CERT® Oracle® Secure Coding Standard for Java™* (Addison-Wesley, 2012).

Dhruv Mohindra is a technical lead in the security practices group that is part of the CTO's office at Persistent Systems Limited, India, where he provides information security consulting solutions across various technology verticals such as cloud, collaboration, banking and finance, telecommunications, enterprise, mobility, life sciences, and health care. He regularly consults for senior management and development teams of Fortune 500 companies, small and medium-sized enterprises, and start-ups on information security best practices and embedding security in the software-development life cycle.

Dhruv has worked for CERT at the Software Engineering Institute and continues to collaborate to improve the state of security awareness in the programming community. Dhruv obtained his M.S. in information security policy and management from Carnegie Mellon University. He holds an undergraduate degree in computer engineering from Pune University, India. Dhruv is also a coauthor of *The CERT® Oracle® Secure Coding Standard for Java™* (Addison-Wesley, 2012).

Robert C. Seacord is the secure coding technical manager in the CERT Division of Carnegie Mellon's Software Engineering Institute (SEI) in Pittsburgh, Pennsylvania. Robert is also a professor in the School of Computer Science and the Information Networking Institute at Carnegie Mellon University. He is the author of *The CERT® C Secure Coding Standard* (Addison-Wesley, 2009) and coauthor of *Building Systems from Commercial Components* (Addison-Wesley, 2002), *Modernizing Legacy Systems* (Addison-Wesley, 2003), *The CERT® Oracle® Secure Coding Standard for Java™* (Addison-Wesley, 2012), and *Secure Coding in C and C++, Second Edition* (Addison-Wesley, 2013). He has also published more than sixty papers on software security, component-based software engineering, web-based system design, legacy-system modernization, component repositories and search engines, and user interface design and development. Robert has been teaching *Secure Coding in C and C++* to private industry, academia, and government since 2005. He started programming professionally for IBM in 1982, working in communications and operating system software, processor development, and software engineering. Robert also has worked at the X Consortium, where he developed and maintained code for the Common Desktop Environment and the X Window System. He represents CMU at the ISO/IEC JTC1/SC22/WG14 international standardization working group for the C programming language.

Dean F. Sutherland is a senior software security engineer at CERT. Dean received his Ph.D. in software engineering from Carnegie Mellon in 2008. Before his return to academia, he spent 14 years working as a professional software engineer at Tartan, Inc. He spent the last six of those years as a senior member of the technical staff and a technical lead for compiler backend technology. He was the primary active member of the corporate R&D group, was a key instigator of the design and deployment of a new software-development process for Tartan, led R&D projects, and provided both technical and project leadership for the 12-person compiler backend group. Dean is a coauthor of *The CERT® Oracle® Secure Coding Standard for Java™* (Addison-Wesley, 2012).