HZ BOOKS
华章科技

Oracle官方发布，Oracle资深专家撰写，国内资深Java技术专家翻译，Amazon畅销书

基于最新Java SE 7，完整且准确地阐述Java虚拟机规范，是深度了解Java虚拟机和Java语言实现细节的必读之作

Java 核心技术系列

PEARSON

Java 虚拟机规范

（Java SE 7版）

The Java Virtual Machine Specification

Java SE 7 Edition

（美）Tim Lindholm　Frank Yellin　Gilad Bracha　Alex Buckley　著

周志明　薛笛　吴璞渊　冶秀刚　译

机械工业出版社
China Machine Press

Java 核心技术系列

Java 虚拟机规范

（Java SE 7版）

The Java Virtual Machine Specification
Java SE 7 Edition

（美）Tim Lindholm　Frank Yellin　Gilad Bracha　Alex Buckley　著
周志明　薛笛　吴璞渊　冶秀刚　译

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

Java 虚拟机规范（Java SE 7 版）/（美）林德霍尔姆（Lindholm，T.）等著；周志明等译．—北京：机械工业出版社，2013.12
（Java 核心技术系列）
书名原文：The Java Virtual Machine Specification，Java SE 7 Edition

ISBN 978-7-111-44515-9

I. J…　II. ①林…　②周…　III. JAVA 语言—程序设计　IV. TP312

中国版本图书馆 CIP 数据核字（2013）第 251092 号

本书是 Java 领域最重要和最权威的著作之一，由 Oracle 官方发布，基于 Java SE 7，对最新的 Java 虚拟机规范进行了完整且详细的讲解，是深度了解 Java 虚拟机实现细节的必读之作。由国内几位资深的 Java 技术专家联袂翻译。

全书共 7 章，第 1 章从宏观的角度介绍了 Java 虚拟机与 Java 的关系与发展历程。第 2 章概览了 Java 虚拟机整体架构，包括 class 文件格式、数据类型、原始类型、引用类型、运行时数据区、栈帧、浮点算法、异常等，这对理解本书后面的内容有重要帮助。第 3 章详述如何将 Java 语言编写的程序转换为 Java 虚拟机指令集，涉及常量、局部变量、控制结构、算术运算、参数接收、方法调用、数组、操作数栈异常处理、同步与注解等。第 4 章深入分析了用来表示编译后的类和接口的 class 文件格式，主要包括 ClassFile 结构、描述符与签名、常量池、字段、方法、属性、代码约束与 class 文件校验等。第 5 章定义了 Java 虚拟机启动以及类和接口的加载、链接和初始化过程。第 6 章阐释并列举了 Java 虚拟机指令集。第 7 章提供了一张以操作码值为索引的 Java 虚拟机操作码助记符表。

机械工业出版社（北京市西城区百万庄大街 22 号　　邮政编码　100037）
责任编辑：秦　健
北京市荣盛彩色印刷有限公司印刷
2014 年 1 月第 1 版第 1 次印刷
186mm × 240mm · 19.75 印张
标准书号：ISBN 978-7-111-44515-9
定　　价：69.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991　88361066　　投稿热线：（010）88379604
购书热线：（010）68326294　88379649　68995259　　读者信箱：hzjsj@hzbook.com

译者序

从1999年4月出版《Java虚拟机规范（第2版）》至今，已经14年过去了，虽然此规范在JDK5发布的时候做了较大的更新，但却始终没有发布完整的规范。2011年6月28日，最新的《Java虚拟机规范》终于完成并在7月份正式发布。对于想了解Java虚拟机的程序员来说，本书是必须阅读的；想深入了解Java语言细节的程序员阅读本书也有极大好处，但是《Java虚拟机规范》、《Java语言规范》发布十余年，一直没有中文译本，这令国内不少对Java虚拟机感兴趣但英语能力较弱的程序员只能望书兴叹。

2011年年初，本书还处于草稿状态时，我就开始关注本书，并陆续对其中第1、2、6、7章进行了翻译，到2011年9月时完成了200余页的译稿。这时候又在国内著名Java社区ItEye中结识了另外三名译者薛笛、吴璞渊和冶秀刚，我们在随后的两个多月的时间里共同完成了其余章节的翻译和校对。在2013年，机械工业出版社获得了这本书的版权，并且采用了我们翻译的版本出版，因此有了这本中文版图书。

本书并非某一款虚拟机实现的说明书，它是一份保证各个公司的Java虚拟机实现具备统一外部接口的契约文档，书中的概念和细节描述曾经与Sun公司早期虚拟机的实现高度吻合，但是随着技术的发展，高性能虚拟机真正的细节实现方式已经渐渐与虚拟机规范所描述的内容产生了越来越大的差距。作者也在书中不同地方反复强调：虚拟机规范中所提及的“Java虚拟机”皆为虚拟机的概念模型而非具体实现。实现只要保证与概念模型最终等效即可，而具体实现的方式无需受概念模型束缚。因此，通过《Java虚拟机规范》分析程序的执行语义问题（虚拟机会做什么）是十分合适且具权威性的，但分析程序的执行行为问题（虚拟机是怎样做的、性能如何）则意义不大，如果需要对具体虚拟机实现进行调优、性能分析等，我推荐在本书基础上继续阅读《Java Performance》和《Oracle JRockit：The Definitive Guide》等书。

在翻译过程中，我们尽最大努力保证作品的准确性和可读性，力求在保证语义准确的前提下，尽可能使用通俗易懂的方式向给各位读者介绍Java虚拟机的约束与运作原理。为实现此目标，我

们在翻译专有技术名词、偏僻词时用括号保留了原文，并专门在多处读者理解起来可能有困难的地方，添加了“译者注”加以解释。

囿于我们的水平和翻译时间，书中难免存在不妥之处，大家如有任何意见或建议都欢迎通过以下电子邮箱与我联系：**understandingjvm@gmail.com**。

周志明

前　言

本书整合了自1999年《Java虚拟机规范（第2版）》发布以来Java世界所出现的技术变化。另外，还修正了第2版中的许多错误，以及对目前主流Java虚拟机实现来说已经过时的内容。最后还处理了一些Java虚拟机和Java语言概念的模糊之处。

针对本书，读者有任何勘误或模糊之处，均可发邮件到jvms-comments-ww@oracle.com。

2004年发布的Java SE 5.0版为Java语言带来了翻天覆地的变化，但是对Java虚拟机设计的影响则相对较小。在Java SE 7这个版本中，我们扩充了class文件格式以便支持新的Java语言特性，譬如泛型和变长参数方法等。

2006年发布的Java SE 6.0版看起来并没有为Java语言带来什么新的变化，但是对Java虚拟机的影响就比较大。如新的字节码验证方式，它源于Eva Rose的一篇硕士论文，文中以Java Card平台为背景，展示了Java虚拟机字节码验证的另一种全新的实现思路。这促进了Java ME CLDC第1版实现的诞生，并最终成为Java SE平台class验证过程的理论基础。关于这部分内容将会在第4章中介绍。

Sheng Liang实现了Java ME CLDC的验证器。Gilad Bracha负责对该验证器做出详细说明，Antero Taivalsaari则是整个Java ME CLDC规范的负责人。Alessandro Coglio在字节码验证的分析方面的工作对本规范做出了很大的贡献。Wei Tao、Frank Yellin、TimLindholm与Gilad Bracha一起实现的Prolog验证器是Java ME和Java SE平台规范的共同基础。Wei Tao后续继续实现了实际运用于Hot Spot Java虚拟机的验证器。之后Mingyao Yang改进了规范和设计，形成了Java SE 6中的最终实现版本。该规范成文得益于以下JSR 202专家组成员：Peter Burka、Alessandro Coglio、Sanghoon Jin、Christian Kemper、Larry Rau、EvaRose以及Mark Stolz。

在2011年发布的Java SE 7平台终于兑现了在1997年《Java虚拟机规范》第1版中就已做出的承诺："在未来，我们会对Java虚拟机进行适当扩展，以便更好地支持其他语言运行于JVM之上。"Gilad Bracha的工作是开发Java虚拟机中的热替换（hotswapping）功能，以及在Java虚拟机静态类型系统上支持动态类型语言实现。invokedynamic指令以及支持这个指令的基础架构由

John Rose 以及 JSR 292 专家组成员：Ola Bini、Rémi Forax、Dan Heidinga、Fredrik Öhrström、JochenTheodorou 进行开发。还有 Charlie Nutter 和 Christian Thalinger 做出了特别贡献。

还有许多人的名字应当出现在这里，他们在不同时间段对 Java 虚拟机的设计和实现做出过贡献。我们今天所见的 Java 虚拟机拥有卓越的执行性能，这离不开 DavidUngar 和他的同事们在 Sun 实验室 Self 项目中所积累的技术基础。这些技术最初用于 Self 语言，后来形成了 Animorphic Smalltalk 虚拟机，经过长期而曲折的发展，最终成为今天 Oracle HotSpot JVM 的技术基础。Lars Bak 和 Urs Hölzle 经历了所有上述的技术发展阶段，对于今天的 Java 虚拟机能够拥有大家认为理所当然的高效执行性能，他们实在是居功至伟。

本规范中很多意义深远的改进来自于 Martin Buchholz、Brian Goetz、Paul Hohensee、David Holmes、Karen Kinnear、Keith McGuigan、Jeff Nisewanger、Mark Reinhold、Naoto Sato、BillPugh、Uday Dhanikonda、Janet Koenig、AdamMessinger、John Pampuch、Georges Saab 和 Bernard Traversat 所作出的贡献。Jon Courtney 和 Roger Riggs 帮助我们保证此规范的内容可同时适用于 Java ME 和 Java SE 平台。Leonid Arbouzov、Stanislav Avzan、Yuri Gaevsky、Ilya Mukhin、Sergey Reznick 和 Kirill Shirokov 在 Java 技术兼容包（JavaCompatibility Kit，JCK）上作出了卓越贡献，以保证本规范中描述的内容是可测试并且已测试的。

Gilad Bracha

Los Altos，California

Alex Buckley

Santa Clara，California

第 2 版前言

《 Java 虚拟机规范（第 2 版）》将规范所描述内容的技术背景升级到了 Java 2 平台（JDK 1.2），它还包括了许多对第 1 版的修正，并且在不改变规范内容逻辑的情况下，使描述变得更加清晰。我们也尝试调整了规范中的字体样式、勘误（希望勘误不会产生新的错误）以及对规范中模糊的部分增加额外的描述。另外，我们还修正了许多《 Java 虚拟机规范》和《 Java 语言规范》之间不一致的内容。

我们很感谢所有为我们梳理过第一版规范并指出问题的读者，特别感谢以下个人和团体，他们指出了问题甚至直接提供了修改意见。

Carla Schroer 与她在加利福尼亚州古本蒂诺、俄罗斯新西伯利亚的兼容性测试团队（尤其感谢其中的 Leonid Arbouzov 和 AlexeiKaigorodov）。他们煞费苦心地为第 1 版中各处可测试的场景编写了兼容性测试用例。在这个过程中，他们还发现了许多处第 1 版规范中不清晰和不完整的内容。Jeroen Vermeulen、Janice Shepherd、Peter Bertelsen、Roly Perera、JoeDarcy 与 Sandra Loosemore 提交了许多有用的建议和反馈，这些建议和反馈对于第 2 版规范的改进工作有很大帮助。Addison Wesley Longman 出版社的编辑 Marilyn Rash 和 Hilary Selby Polk 帮助我们在第 2 版中合并技术变更的同时，改进了规范的可读性和内容的布局排版。

还要特别感谢 Gilad Bracha，他对本书出版进行了严格审查，另外他也是本书新增内容的主要贡献者，尤其是第 4、5 章。他对计算机理论的贡献以及他解决的《 Java 虚拟机规范》和《 Java 语言规范》之间的描述差异问题都极大地完善了本书。

Tim Lindholm

Palo Alto，California

Frank Yellin

Redwood City，California

1999 年 4 月

第 1 版前言

《Java 虚拟机规范》是描述 Java 虚拟机设计原理的一份完整的规范文档。这份文档对于任何一个希望实现 Java 虚拟机的编译器作者，或者希望实现一个与规范兼容的 Java 虚拟机的程序员来说都是必不可少的。

Java 虚拟机是一个抽象化的机器，整个规范中提及的 Java 虚拟机都是抽象化的概念，而不是特指 Oracle 或者其他某一家公司的 Java 虚拟机实现。本书与一个具体的虚拟机实现之间的关系就犹如一份建筑蓝图与一间具体的房屋之间的关系一样。Java 虚拟机具体实现（包括任何公司的 JVM 实现）必须包括本规范所描述的内容，但是除了少数绝对必要的地方外，本规范中的描述不应成为 Java 虚拟机具体实现的束缚。我们希望这个规范至少能作为一个“实验室”版本的虚拟机实现的完整描述。

Java 虚拟机源于由 James Gosling 在 1992 年设计，用于支持 Oak 程序语言的虚拟机。在 Java 虚拟机的发展历程中，Sun 公司的 Green 项目、FirstPerson 公司、LiveOak 项目、Java 产品组、JavaSoft 公司以及今天的 Oracle 公司的 Java 平台组中许多人都作出了直接或间接的贡献。

这本书最初是源于由 Kathy Walrath 编著的一份公司内部文档。Mary Campione 将本书转换为 HTML 版本，使大家可通过互联网访问到本书。

《Java 虚拟机规范》的诞生离不开 Java 产品团队的总经理 Ruth Hennigar 的大力支持，还有编辑 Lisa Friendly、Mike Hendrickson 以及他在 Addison-Wesley 出版社的团队所做出的编辑工作。在此特别感谢 Richard Tuck 对原稿的仔细审查，还有 Bill Joy 对本书的审查、评价和指导意见，这些宝贵信息对本书定稿都有很大的帮助。

Tim Lindholm
Palo Alto，California
Frank Yellin
Redwood City，California
1996 年 6 月

目 录

第 1 章
引　言

1.1　简史

Java 语言是一门通用的、面向对象的、支持并发的程序语言。它的语法与 C 和 C++ 语言非常相似，但隐藏了 C 和 C++ 中许多复杂、深奥及不安全的语言特性。Java 平台最初用于解决基于网络的消费类设备上的软件开发问题，它在设计上就考虑到要支持部署在不同架构的主机上，并且不同组件之间可以安全地交互。面对这些需求，编译出来的本地代码必须解决不同网络间的传输问题，可以操作各式各样的客户端，并且还要在这些客户端上能安全正确地运行。

伴随着万维网的盛行发生了一些十分有趣的事情：Web 浏览器允许数以百万计的用户共同在网上冲浪，以及通过很简单的方式访问丰富多样的内容。用户冲浪所使用的设备并不是其中的关键，它们仅仅是一种媒介，无论机器的性能如何，无论使用高速网络还是慢速的 modem，这些外界因素本质上与你所看到的、听到的内容没有任何关系。

Web 狂热者很快就发现网络信息的载体——HTML 文档格式对信息的表达有很多限制，HTML 的一些扩展应用，譬如网页表单，让这些限制显得更加明显。显而易见，没有任何浏览器能够承诺它可以提供给用户所需要的全部特性，扩展能力将是解决这个问题的唯一答案。

Sun 公司的 HotJava 浏览器是世界上第一款展现出 Java 语言某些有趣特性的浏览器，它允许把 Java 代码内嵌入 HTML 页面。在 HTML 页面呈现的时候，这些 Java 代码显式地下载至浏览器中。而在浏览器获取这些代码之前，它们已经过严谨地检查以保证它们是安全的。与 HTML 语言一样，这些 Java 代码与网络和主机是完全无关的，无论代码来自哪里，在哪台机器上执行，它们执行时都能表现出一致的行为。

带有 Java 技术支持的网页浏览器将不再受限于它本身所提供的功能。浏览网页的用户可以放心地假定在他们机器上运行的动态内容不会损害其机器。软件开发人员编写一次代码，程序就可以运行在所有支持 Java 运行时环境的机器之上。

1.2 Java 虚拟机

Java 虚拟机是整个 Java 平台的基石，是 Java 技术用以实现硬件无关与操作系统无关的关键部分，是 Java 语言生成出极小体积的编译代码的运行平台，是保障用户机器免于恶意代码损害的屏障。

Java 虚拟机可以看做一台抽象的计算机。如同真实的计算机那样，它有自己的指令集以及各种运行时内存区域。使用虚拟机来实现一门程序设计语言有许多合理的理由，业界中流传最为久远的虚拟机可能是 UCSD Pascal 的 P-Code 虚拟机[⊖]。

第一个 Java 虚拟机的原型机是由 Sun Microsystems 公司实现的，它用在一种类似 PDA（Personal Digital Assistant，俗称掌上电脑）的手持设备上，以仿真实现 Java 虚拟机指令集。时至今日，Oracle 已将许多 Java 虚拟机实现应用于移动设备、台式机、服务器等领域。但 Java 虚拟机并不局限于特定的实现技术、主机硬件和操作系统。Java 虚拟机也不局限于特定的代码执行方式，它不强求使用解释器来执行程序，也可以通过把自己的指令集编译为实际 CPU 的指令来实现。它可以通过微代码（microcode）来实现，或者甚至直接在 CPU 中实现。

Java 虚拟机与 Java 语言并没有必然的联系，它只与特定的二进制文件格式——class 文件格式所关联。class 文件包含了 Java 虚拟机指令集（或者称为字节码（bytecode））和符号表，以及一些其他辅助信息。

基于安全方面的考虑，Java 虚拟机要求在 class 文件中使用许多强制性的语法和结构化约束，但任意一门功能性语言都可以表示为一个能被 Java 虚拟机接收的有效的 class 文件。作为一个通用的、机器无关的执行平台，任何其他语言的实现者都可以将 Java 虚拟机作为他们语言的产品交付媒介。

1.3 各章节摘要

本书中其余章节的概述如下：

- 第 2 章概览 Java 虚拟机整体架构。
- 第 3 章介绍如何将 Java 语言编写的程序转换为 Java 虚拟机指令集。
- 第 4 章定义 class 文件格式。它是一种与硬件和操作系统无关的二进制格式，用来表示编译后的类和接口。
- 第 5 章定义了 Java 虚拟机启动以及类和接口的加载、链接和初始化过程。
- 第 6 章定义了 Java 虚拟机指令集，并按照这些指令的指令助记符的字母顺序来表示。
- 第 7 章提供了一张以操作码值为索引的 Java 虚拟机操作码助记符表。

在《Java 虚拟机规范（第 2 版）》中，第 2 章是 Java 语言概览，这可以使读者更好地理解 Java 虚拟机规范，但它本身并不属于规范的一部分。在本规范里没有再包含此章节的内

⊖ P-Code 虚拟机是由加州大学圣地亚哥分校（UCSD）于 1978 年发布的高度可移植、机器无关的、运行 Pascal 语言的虚拟机。——译者注

容，读者可以参考《Java 语言规范》(Java SE 7 版）来获取这部分信息，如在本书中有需要引用这些信息的地方，将使用类似于“(JLS §x.y)”的形式来表示。

在《Java 虚拟机规范（第 2 版）》中，第 8 章用于描述 Java 虚拟机线程和共享内存的底层操作，它对应于《Java 语言规范》第 1 版的第 17 章。而《Java 语言规范（第 2 版）》出版时，第 17 章则对应于 JSR-133 专家组所发布的《Java 内存模型和线程规范》[⊖]，本规范中不再包含这部分内容，读者参考上述规范来获取关于线程与锁的信息。

1.4 说明

在本书中，我们所使用到的类和接口来自于 Java SE 平台的 API。每当使用某个字母（譬如 N）来表示某个类或接口时，默认都是指 java.lang.N 所代表的那个类或接口，如果要描述其他包中的类或接口，我们将会使用全限定名。

每当提及某个类或接口的包是 java 或者它的子包（java.*）时，那就意味着这个类或接口是由启动类加载器进行加载的（见 5.3.1 小节）。每当提及某个包是 java 包的子包时，就意味着这个包是由类加载器所定义的。

在本书中，*斜体*用于描述 Java 虚拟机中的“汇编语言”，即操作码和操作数，也包括一些 Java 虚拟机运行时数据区中的项目，有时也用来说明一些新的条目和需要强调的内容。

⊖《Java Memory Model and Thread Specification》：http://www.jcp.org/en/jsr/summary?id=133。——译者注

第 2 章
Java 虚拟机结构

本规范描述的是一种抽象化的虚拟机的行为，而不是任何一种[⊖]广泛使用的虚拟机实现。

如果只是要去“正确地”实现一台 Java 虚拟机，其实并不像大多数人所想的那样高深和困难——只需要正确读取 class 文件中每一条字节码指令，并且能正确执行这些指令所蕴含的操作即可。所有在虚拟机规范之中没有明确描述的实现细节，都不应成为虚拟机设计者发挥创造性的牵绊，设计者可以完全自主决定所有规范中不曾描述的虚拟机内部细节，例如，运行时数据区的内存如何布局，选用哪种垃圾收集算法，是否要对虚拟机字节码指令进行一些内部优化操作（如使用即时编译器把字节码编译为机器码）。

在本规范之中所有关于 Unicode 的描述，都是基于 **Unicode 6.0.0 标准**，读者可以在 Unicode 的网站（http://www.unicode.org）中查找到相关资料。

2.1 class 文件格式

编译后被 Java 虚拟机所执行的代码使用了一种平台中立（不依赖于特定硬件及操作系统）的二进制格式来表示，并且经常（但并非绝对）以文件的形式存储，因此这种格式称为 class 文件格式。class 文件格式中精确地定义了类与接口的表示形式，包括在平台相关的目标文件格式中一些细节上的惯例[⊜]，例如字节序（byte ordering）等。

关于 class 文件格式细节的定义，请参见第 4 章的相关内容。

⊖ 包括 Oracle 公司自己的 HotSpot 和 JRockit 虚拟机。——译者注

⊜ 请勿误认为此处“平台相关的目标文件格式”是指在特定平台编译出的 class 文件无法在其他平台中使用。相反，正是因为强制、明确地定义了本来会跟平台相关的细节，所以才达到了平台无关的效果。例如在 SPARC 平台上数字以 Big-Endian（高位的字节存储在内存中的低地址处）形式存储，在 x86 平台上数字则是以 Little-Endian（高位的字节存储在内存中的高地址处）形式存储的，如果不强制统一字节序的话，同一个 class 文件的二进制形式放在不同平台上就可能以不同的方式解读。——译者注