HZ BOOKS 华章IT

Oracle官方发布，Java虚拟机技术创建人撰写，国内资深Java技术专家翻译，Amazon畅销书

基于最新Java SE 8，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 8版）

The Java Virtual Machine Specification

Java SE 8 Edition

[美] 蒂姆·林霍尔姆（Tim Lindholm）
弗兰克·耶林（Frank Yellin）
吉拉德·布拉查（Gilad Bracha）
亚历克斯·巴克利（Alex Buckley） 著

爱飞翔 周志明 等译

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 8版)

The Java Virtual Machine Specification
Java SE 8 Edition

[美] 蒂姆·林霍尔姆 (Tim Lindholm)
弗兰克·耶林 (Frank Yellin)
吉拉德·布拉查 (Gilad Bracha)
亚历克斯·巴克利 (Alex Buckley) 著

爱飞翔 周志明 等译

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 8 版）/（美）林霍尔姆（Lindholm, T.）等著；爱飞翔等译．
—北京：机械工业出版社，2015.5
（Java 核心技术系列）
书名原文：The Java Virtual Machine Specification, Java SE 8 Edition

ISBN 978-7-111-50159-6

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

中国版本图书馆 CIP 数据核字（2015）第 095083 号

本书版权登记号：图字：01-2014-5471

Java 虚拟机规范（Java SE 8 版）

出版发行：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：关 敏 责任校对：殷 虹
印 刷：三河市宏图印务有限公司 版 次：2015 年 6 月第 1 版第 1 次印刷
开 本：186mm × 240mm 1/16 印 张：21.25
书 号：ISBN 978-7-111-50159-6 定 价：79.00 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）88379426 88361066 投稿热线：（010）88379604
购书热线：（010）68326294 88379649 68995259 读者信箱：hzit@hzbook.com

推荐阅读

Java核心技术：卷I 基础知识（原书第9版）

作者：（美）Cay S. Horstmann　Gary Cornell
译者：周立新 等
ISBN：978-7-111-44514-2
定价：119.00元

Java核心技术：卷II 高级特性（原书第9版）

作者：（美）Cay S. Horstmann　Gary Cornell
译者：陈昊鹏 等
ISBN：978-7-111-44250-9
定价：139.00元

Java EE 7权威指南：卷1（原书第5版）

作者：（美）埃里克·珍兆科 等
译者：苏金国 等
ISBN：978-7-111-49760-8
定价：99.00元

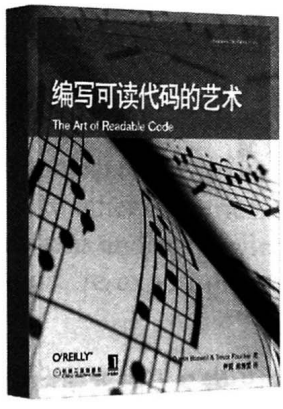
Java EE 7权威指南：卷2（原书第5版）

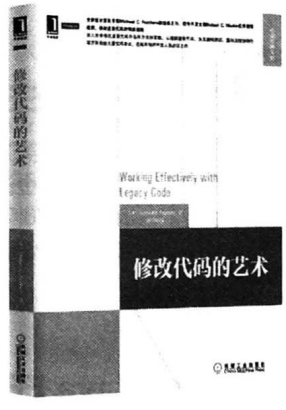
作者：（美）埃里克·珍兆科 等
译者：苏金国 等
ISBN：978-7-111-49711-0
定价：99.00元

Java应用架构设计：模块化模式与OSGi

作者：（美）Kirk Knoernschild
译者：张卫滨
ISBN：978-7-111-43768-0
定价：69.00元

推荐阅读





The Translator's Words 译者序

Java从诞生到现在历经20多年，如今已成为一门应用场合非常广泛的编程语言。而在它逐步发展的过程中，还有另一件事物也在不断发生变化，这就是Java虚拟机。

与某些语言相比，Java的特色之一就是通常需要把编译好的**class**文件放在虚拟机中执行，而不是直接放在硬件上执行。这种在硬件和二进制文件中加入虚拟机层的做法，自然有其优势与局限性，然而纵观Java语言与Java虚拟机的发展脉络就可看出，各种Java虚拟机的实现者依然在以他们自己的方式不断地优化虚拟机。

虚拟机的具体实现可以有差别，但它们都遵循一套抽象的规则，这就是Java虚拟机规范。这份规范不仅可以使Java虚拟机的实现变得更加协调，而且还阐明了Java虚拟机与Java语言之间的契合点，令实现者可以在保持程序语义不变的前提下获得充分的发挥空间。

从J2SE 5.0开始，Java有了较大改变，加入了泛型、枚举、变长参数、多异常**catch**语句等特性，到了Java SE 8，更是引入了与lambda表达式相关的许多新功能，使Java语言的写法变得更为灵活。与此同时，Java虚拟机也在针对这些特性而调整。无论读者是否从事虚拟机开发，都可以从研读规范的过程中更为深入地体会这些特性。大家还可以参考Bill Venners所著的《Inside the Java 2 Virtual Machine》(《深入Java虚拟机（原书第2版）》)，以了解Java虚拟机的原理及指令细节。

尽管Java虚拟机通常与Java语言配套使用，但除了Java语言之外，用Clojure、Scala等语言所写的程序也可以运行在Java虚拟机上。此外，还可以用Java语言实现出Python、Ruby等语言的解释器，从而将其放在Java虚拟机中执行。这些用法都表明：虚拟机规范不但对学习Java有帮助，而且还能促使我们以全新的手法来运用其他常见的语言。从某种意义上来看，Java虚拟机有其独特的地位，而且还是程序设计领域中的一种思维方式。

翻译本书的过程中，译者参考了由周志明、薛笛、吴璞渊、冶秀刚所翻译的《Java

虚拟机规范（Java SE 7 版）》，并保留了上一版的部分译者注，在此谨对四位译者深表感谢。同时感谢机械工业出版社华章公司诸位编辑与工作人员的帮助。

本书的风格和术语尽量与上一版相符，有时会酌情稍作调整。欢迎大家发邮件至eastarstormlee@gmail.com，或访问github.com/jeffreybaoshenlee/zh-translation-errata-jvmspec8/issues，给我以批评和指正。该网址还列有《中英文词汇对照表》，以供参考。

爱飞翔

Preface 前 言

本书涵盖了自2011年发布Java SE 7版之后所发生的全部变化。此外，为了与常见的Java虚拟机实现相匹配，本书还添加了大量修订及说明。

本版与前面各版一样，仅仅描述了抽象的Java虚拟机，而在实现具体的Java虚拟机时，本书指出了设计规划。Java虚拟机的实现必须体现出本书中的内容，但仅在确有必要时才应该受制于这些规范。

对于Java SE 8来说，Java编程语言里的一些重要变化在这本Java虚拟机规范中都有相应的体现。为了尽量保持二进制兼容性，我们应该直接在Java虚拟机里指定带有默认实现代码的`default`方法，而不应该依赖于编译器，因为那样做将无法在不同厂商、不同版本的产品之间移植，此外，那种做法也不可能适用于已有的`class`文件。在设计JSR 335，也就是《Lambda Expressions for the Java Programming Language》(Java编程语言的lambda表达式)时，Oracle公司的Dan Smith向虚拟机实现者咨询了将`default`方法集成到常量池和方法结构、方法与接口方法解析算法，以及字节码指令集中的最佳方式。JSR 335也允许在`class`文件级别的接口里出现`private`方法与`static`方法，而这些方法也同接口方法解析算法紧密地结合起来了。

Java SE 8的特点之一是：Java SE平台的程序库也伴随着Java虚拟机一起进化。有个小例子可以很好地说明这一特点：在运行程序的时候，Java SE 8可以获取方法的参数名，虚拟机会把这些名字存放在`class`文件结构中，而与此同时，`java.lang.reflect.Parameter`里也有个标准的API能够查询这些名字。另外，我们也可以通过`class`文件结构中一项有趣的统计数据来说明这个特点：本规范的第1版中定义了6个属性，其中有3个属性对Java虚拟机至关重要，而Java SE 8版的规范则定义了23个属性，其中只有5个属性对Java虚拟机很重要。换句话说，在新版规范中，属性主要是为了支持程序库而设计的，其次才是为了支持Java虚拟机本身。为了帮助读者理解`class`文件结构，本规范会更为清晰地描述出每项属性的角色及其使用限制。

在Oracle公司的Java Platform团队里，有多位同事都对这份规范提供了大力支持，他们包括：Mandy Chung、Joe Darcy、Joel Franck、Staffan Friberg、Yuri Gaevsky、Jon Gibbons、Jeannette Hung、Eric McCorkle、Matherey Nunez、Mark Reinhold、John Rose、Georges Saab、Steve Sides、Bernard Traversat、Michel Trudeau和Mikael Vidstedt。尤其感谢Dan Heidinga（IBM）、Karen Kinnear、Keith McGuigan及Harold Seigel，他们对常见的Java虚拟机实现中的兼容性及安全性贡献良多。

Alex Buckley

于加利福尼亚州圣克拉拉

Contents 目 录

译者序

前 言

第 1 章 引言……1

1.1 简史……1

1.2 Java 虚拟机……2

1.3 各章节摘要……2

1.4 说明……3

1.5 反馈……3

第 2 章 Java 虚拟机结构……4

2.1 **class** 文件格式……4

2.2 数据类型……5

2.3 原始类型与值……5

2.3.1 整数类型与整型值……6

2.3.2 浮点类型、取值集合及浮点值……6

2.3.3 **returnAddress** 类型和值…8

2.3.4 **boolean** 类型……8

2.4 引用类型与值……9

2.5 运行时数据区……9

2.5.1 **pc** 寄存器……9

2.5.2 Java 虚拟机栈……10

2.5.3 Java 堆……10

2.5.4 方法区……11

2.5.5 运行时常量池……11

2.5.6 本地方法栈……12

2.6 栈帧……12

2.6.1 局部变量表……13

2.6.2 操作数栈……14

2.6.3 动态链接……14

2.6.4 方法调用正常完成……15

2.6.5 方法调用异常完成……15

2.7 对象的表示……15

2.8 浮点算法……15

2.8.1 Java 虚拟机和 IEEE 754 中的浮点算法……15

2.8.2 浮点模式……16

2.8.3 数值集合转换……17

2.9 特殊方法……18

2.10 异常……19

2.11 字节码指令集简介……20

2.11.1 数据类型与 Java 虚拟机…21

2.11.2 加载和存储指令……23

2.11.3 算术指令……24

2.11.4 类型转换指令……25

2.11.5 对象的创建与操作……27
2.11.6 操作数栈管理指令……27
2.11.7 控制转移指令……27
2.11.8 方法调用和返回指令……28
2.11.9 抛出异常……28
2.11.10 同步……28
2.12 类库……29
2.13 公有设计、私有实现……30

第 3 章 Java 虚拟机编译器……31

3.1 示例的格式说明……31
3.2 常量、局部变量和控制结构的使用……32
3.3 算术运算……36
3.4 访问运行时常量池……36
3.5 与控制结构有关的更多示例……37
3.6 接收参数……40
3.7 方法调用……41
3.8 使用类实例……43
3.9 数组……44
3.10 编译 **switch** 语句……46
3.11 使用操作数栈……48
3.12 抛出异常和处理异常……48
3.13 编译 **finally** 语句块……51
3.14 同步……54
3.15 注解……55

第 4 章 class 文件格式……56

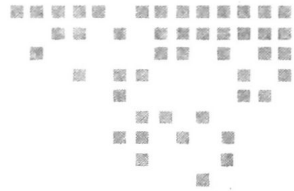
4.1 **ClassFile** 结构……57
4.2 各种名称的内部表示形式……61
4.2.1 类和接口的二进制名称……61
4.2.2 非限定名……61
4.3 描述符……62
4.3.1 语法符号……62
4.3.2 字段描述符……62
4.3.3 方法描述符……63
4.4 常量池……64
4.4.1 **CONSTANT_Class_info** 结构……65
4.4.2 **CONSTANT_Fieldref_info**、**CONSTANT_Methodref_info** 和 **CONSTANT_InterfaceMethodref_info** 结构……66
4.4.3 **CONSTANT_String_info** 结构……67
4.4.4 **CONSTANT_Integer_info** 和 **CONSTANT_Float_info** 结构……67
4.4.5 **CONSTANT_Long_info** 和 **CONSTANT_Double_info** 结构……68
4.4.6 **CONSTANT_NameAndType_info** 结构……69
4.4.7 **CONSTANT_Utf8_info** 结构……70
4.4.8 **CONSTANT_MethodHandle_info** 结构……72
4.4.9 **CONSTANT_MethodType_info** 结构……73

4.4.10 **CONSTANT_Invoke-Dynamic_info** 结构……74
4.5 字段……74
4.6 方法……76
4.7 属性……78
4.7.1 自定义和命名新的属性……82
4.7.2 **ConstantValue** 属性……82
4.7.3 **Code** 属性……83
4.7.4 **StackMapTable** 属性……86
4.7.5 **Exceptions** 属性……92
4.7.6 **InnerClasses** 属性……93
4.7.7 **EnclosingMethod** 属性…95
4.7.8 **Synthetic** 属性……96
4.7.9 **Signature** 属性……96
4.7.10 **SourceFile** 属性……100
4.7.11 **SourceDebugExtension** 属性……101
4.7.12 **LineNumberTable** 属性……102
4.7.13 **LocalVariableTable** 属性……103
4.7.14 **LocalVariableTypeTable** 属性……104
4.7.15 **Deprecated** 属性……106
4.7.16 **RuntimeVisibleAnnotations** 属性……106
4.7.17 **RuntimeInvisible-Annotations** 属性……110
4.7.18 **RuntimeVisibleParameter Annotations** 属性……111
4.7.19 **RuntimeInvisiblePara-meterAnnotations** 属性……112
4.7.20 **RuntimeVisibleType Annotations** 属性……114
4.7.21 **RuntimeInvisibleType-Annotations** 属性……124
4.7.22 **AnnotationDefault** 属性……125
4.7.23 **BootstrapMethods** 属性……126
4.7.24 **MethodParameters** 属性……127
4.8 格式检查……129
4.9 Java 虚拟机代码约束……129
4.9.1 静态约束……130
4.9.2 结构化约束……132
4.10 **class** 文件校验……135
4.10.1 类型检查验证……136
4.10.2 类型推导验证……200
4.11 Java 虚拟机限制……206

第 5 章 加载、链接与初始化……208

5.1 运行时常量池……208
5.2 虚拟机启动……210
5.3 创建和加载……211
5.3.1 使用引导类加载器来加载类型……212
5.3.2 使用用户自定义类加载器来加载类型……212

5.3.3 创建数组类……213
5.3.4 加载限制……214
5.3.5 从 **class** 文件表示得到类…214
5.4 链接……215
5.4.1 验证……216
5.4.2 准备……216
5.4.3 解析……217
5.4.4 访问控制……225
5.4.5 方法覆盖……225
5.5 初始化……226
5.6 绑定本地方法实现……228
5.7 Java 虚拟机退出……228
第 6 章 Java 虚拟机指令集……229
6.1 设定："必须"的含义……229
6.2 保留操作码……229
6.3 虚拟机错误……230
6.4 指令描述格式……230
6.5 指令集描述……232
第 7 章 操作码助记符……320
附录 A Limited License Grant……327



第 1 章 Chapter 1

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，并能够运行在各种客户端上，而且还要使客户端确信这些代码是安全的。

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，用户总能看到并听到同样的内容。

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。显示 HTML 页面时，这些 Java 代码也会一并下载至浏览器中。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是相当常见的，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，它虽然不强求使用解释器来执行程序，但是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，甚至可以直接在 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，以及其他一些辅助信息。

基于安全方面的考虑，Java 虚拟机在 `class` 文件中施加了许多强制性的语法和结构化约束，凡是能用 `class` 文件正确表达出来的编程语言，都可以放在 Java 虚拟机里面执行。由于它是一个通用的、机器无关的执行平台，所以其他语言的实现者都可以考虑将 Java 虚拟机作为那些语言的交付媒介。

本书所说的 Java 虚拟机与 Java SE 8 平台相兼容，而且支持由本书所定义的 Java 编程语言。

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

⊖ 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

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。因此本规范里没有再包含此章节的内容，读者可以参考《 Java 语言规范》（Java SE 8 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之间的底层操作，它对应于《 Java 语言规范》（Java SE 8 版）的第 17 章，而那一章又对应于 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，读者可参考上述规范来获取关于线程与锁的信息。

1.4 说明

本书会用到 Java SE 平台 API 中的类和接口。如果单用一个未经修饰的标识符（比如 N）来指代类或接口，而那个标识符又不是范例中所定义的，那我们指的就是 `java.lang` 包中的类名或接口名（比如 `java.lang.N`）。如果要提到其他包中的类名或接口名，我们则会使用全限定名。

每当提及 `java` 或者它的子包（`java.*`）里的类和接口时，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 节）。

每当提及某个 `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。

非规范性的信息用于阐明规范中的某些内容，这部分信息以小字缩排的形式来印刷。

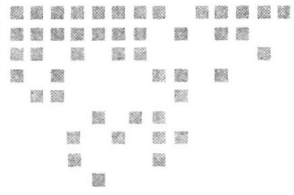
这些文本是 Java 虚拟机规范之外的信息。它们用来表示某些直观的内容、阐述某些原理、给出某种建议或演示某个范例等等。

1.5 反馈

读者如发现本书中有错误、遗漏或含义不明之处，可通过 jvms-comments_ww@oracle.com 发送反馈信息。

用 `javac`（Java 编程语言的参照编译器（reference compiler））来生成并操作 `class` 文件时，如果有问题，可与 compiler-dev@openjdk.java.net 联系。

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



Chapter 2

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。

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。

⊖ 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