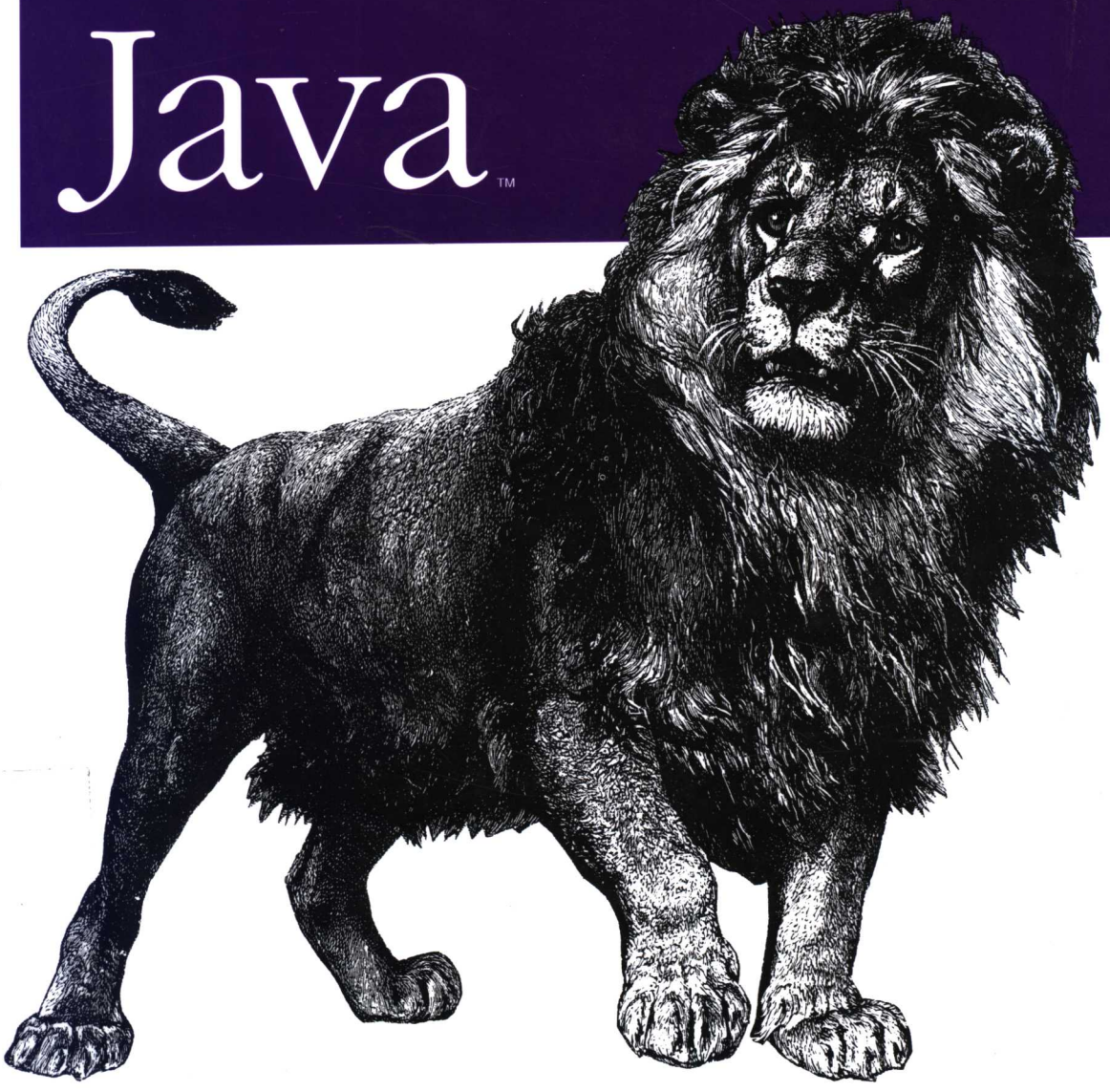
核心Java™（影印版）

Hardcore Java™



O'REILLY®

東南大學出版社

Robert Simmons, Jr. 著

核心Java™（影印版）

Hardcore Java™

Robert Simmons, Jr.

O'REILLY®

Beijing • Cambridge • Farnham • Köln • Paris • Sebastopol • Taipei • Tokyo

O'Reilly Media, Inc.授权东南大学出版社出版

东南大学出版社

图书在版编目（CIP）数据

核心 Java™: /（美）西蒙斯（Simmons, R.）著. —影印版
—南京：东南大学出版社，2005.6
书名原文：Hardcore Java™
ISBN 7-5641-0040-0

Ⅰ.核... Ⅱ.西... Ⅲ.JAVA 语言－程序设计－英文 Ⅳ.TP312

中国版本图书馆 CIP 数据核字（2005）第 059126 号

江苏省版权局著作权合同登记
图字：10-2005-074 号

书　　名/ 核心 Java™（影印版）
书　　号/ ISBN 7-5641-0040-0
责任编辑/ 张烨
封面设计/ Ellie Volckhausen，张健
出版发行/ 东南大学出版社
地　　址/ 南京四牌楼 2 号 邮编 210096
印　　刷/ 扬中市印刷有限公司
开　　本/ 787 毫米 × 980 毫米　16 开本　21.5 印张
版　　次/ 2005 年 10 月第一版　2005 年 10 月第一次印刷
印　　数/ 0001-2000 册
定　　价/ 66.00 元（册）

O'Reilly Media, Inc.介绍

O'Reilly Media, Inc.是世界上在 UNIX、X、Internet 和其他开放系统图书领域具有领导地位的出版公司，同时是联机出版的先锋。

从最畅销的《The Whole Internet User's Guide & Catalog》（被纽约公共图书馆评为二十世纪最重要的50本书之一）到GNN（最早的Internet门户和商业网站），再到WebSite（第一个桌面PC的Web服务器软件），O'Reilly Media, Inc.一直处于Internet发展的最前沿。

许多书店的反馈表明，O'Reilly Media, Inc.是最稳定的计算机图书出版商 —— 每一本书都一版再版。与大多数计算机图书出版商相比，O'Reilly Media, Inc.具有深厚的计算机专业背景，这使得O'Reilly Media, Inc.形成了一个非常不同于其他出版商的出版方针。O'Reilly Media, Inc.所有的编辑人员以前都是程序员，或者是顶尖级的技术专家。O'Reilly Media, Inc.还有许多固定的作者群体 —— 他们本身是相关领域的技术专家、咨询专家，而现在编写著作，O'Reilly Media, Inc.依靠他们及时地推出图书。因为O'Reilly Media, Inc.紧密地与计算机业界联系着，所以O'Reilly Media, Inc.知道市场上真正需要什么图书。

出版说明

随着计算机技术的成熟和广泛应用，人类正在步入一个技术迅猛发展的新时期。计算机技术的发展给人们的工业生产、商业活动和日常生活都带来了巨大的影响。然而，计算机领域的技术更新速度之快也是众所周知的，为了帮助国内技术人员在第一时间了解国外最新的技术，东南大学出版社和美国 O'Reilly Meida, Inc.达成协议，将陆续引进该公司的代表前沿技术或者在某专项领域享有盛名的著作，以影印版或者简体中文版的形式呈献给读者。其中，影印版书籍力求与国外图书“同步”出版，并且“原汁原味”展现给读者。

我们真诚地希望，所引进的书籍能对国内相关行业的技术人员、科研机构的研究人员和高校师生的学习和工作有所帮助，对国内计算机技术的发展有所促进。也衷心期望读者提出宝贵的意见和建议。

第一批影印图书共 10 本，涉及 Java，Unix/Linux，Python 等方面:

- 《核心 Java》（影印版）
- 《Jakarta Commons 经典实例》（影印版）
- 《Weblogic 权威指南》（影印版）
- 《Java 网络编程 第三版》（影印版）
- 《Linux 设备驱动程序 第三版》（影印版）
- 《LPI Linux 认证权威指南》（影印版）
- 《GNU Make 项目管理》（影印版）
- 《游戏开发中的人工智能》（影印版）
- 《学习 Python 第二版》（影印版）
- 《精通正则表达式 第二版》（影印版）

Preface

Studying a computer language is a career-long process. Many developers make the mistake of thinking that they have learned enough. They get caught in the corporate cycle of build-and-deploy and don't seek to expand their knowledge. However, we can hardly blame them for that.

For one, the build-and-deploy cycle is intensive and carries with it a substantial amount of political pressure: managers don't want you to spend days reading a book or trying out code snippets when bugs and deadlines are looming. However, developers should take the time to experiment and learn more.

When you expand your skills as a developer, there is some initial time investment. However, this will rapidly pay off in increased productivity and quality. Instead of spending hundreds of hours debugging, you can implement coding standards that block bugs and spend a fraction of that time implementing new features. In the end, everyone wins; your company gets higher-quality code and quicker feature turnaround, and you get to spend more time playing Frisbee with your dog.

The second problem that the corporate developer has to deal with is that the majority of computer books are often not appropriate for the intermediate to advanced developer. When looking at my rather impressive computer book library, much of it from O'Reilly, I notice that my books tend to fall into two categories: many are introductions to concepts and most of the others are references to concepts. Although these books are very useful, there is a distinct lack of books that target the intermediate to advanced programmer. However, there is one shining exception in my library.

In a dusty corner of my desk is a book I bought several years ago. *Secrets of the C++ Masters* by Jeff Alger (Academic Press Limited) is absolutely essential for an intermediate C++ developer. It begins with the assumption that you know the language and then expands from there. The result is a book that can really transform a developer from the intermediate level to a true guru.

That is the goal of this book with regards to the Java™ language. Most of the material is meant to help you avoid many common mistakes made by Java developers. We will also cover nuances of Java, idiosyncrasies of the JDK, and advanced techniques. With luck, this book will increase your productivity and your enjoyment of Java development.

Audience

This book is for the intermediate to advanced Java programmer. With that in mind, we can concentrate on the knowledge and techniques that go into some of the most advanced Java software available.

Prerequisites and Assumptions

Functional proficiency with Java
: I will largely gloss over entire areas of Java. I assume that you understand JavaBeans™, bound properties, JDBC, and other basics.

Familiarity with basic computer science
: I generally won't spend a lot of time on concepts such as scoping, logic operations, inheritance, and algorithm construction. These and similar concepts will be the basis for more detailed discussions.

Familiarity with UML
: The Unified Modeling Language is the best way to express object-oriented engineering concepts in a manner that is familiar to all programmers, regardless of what language they speak. Most of the code diagrams in this book incorporate UML.

Familiarity with the JDK and the virtual machine
: You should be familiar with the JDK and with how to compile a program and use its various tools in the JDK. However, expertise in all packages isn't necessary.

Typographical Conventions

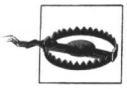
This book uses the following font conventions:

Italic
: Used for filenames, file extensions, URLs, application names, emphasis, and new terms when they are first introduced

`Constant width`
: Used for Java class names, functions, variables, components, properties, data types, events, and snippets of code that appear in the text

`Constant width bold`
: Used for commands you enter at the command line and to highlight new code inserted in a running example

This icon designates a note, which is an important aside to the nearby text.



This icon designates a warning.

Code Samples

The code sample set for this book is massive. Almost every snippet of code from the book can be found in the downloadable source code (*http://www.oreilly.com/catalog/hardcorejv*). However, without a guide, you could get lost quickly when surfing through the examples.

Regarding the code itself, I will frequently snip out pieces you would need to get the code to compile and run. Copying this infrastructure code in the book would add unnecessary bulk and potentially cloud the issue being discussed. Since I assume you are experienced in Java, I will also assume you know the housekeeping procedures used to implement pertinent concepts.

One other tactic that I commonly use is to append a number to the name of a class or method. This is designed to show successive versions of the same class or method. The goal is to emphasize the development while allowing the user to look up the old version and play with it if he chooses. For example, you should read `Country4` as `Country`.

Finally, the code samples are very well-documented. However, for brevity's sake, I will usually slice out this documentation when presenting code examples. Although I firmly believe that good Javadoc documentation is important to good development, in this book such documentation would needlessly increase the page count without adding to the discussion.

One other thing to note about the examples is that you will often see the comment `//$NON-NLS-1$` imbedded within the code. This is merely a flagging comment that tells Eclipse not to internationalize a particular String. I have snipped these comments from the book, as they aren't relevant to the discussions.

Locating an Example in the Downloadable Code

Each example cited in the book is formatted as:

```
package oreilly.hcj.review;
public class PointersAndReferences {
  public static void someMethod(Vector source) {
```

```
        Vector target = source;
        target.add("Swing");
    }
}
```

The emphasized lines show that you can find this code in the package `oreilly.hcj.review` and the class `PointersAndReferences`. However, be aware that the code example cited may be embedded with other examples that are not relevant to that particular topic. In fact, I frequently combine several examples from a single subject into one class file to reduce the housekeeping code needed to run the sample. Doing a search on the method name will quickly locate the cited example.

Categories of Examples

The examples themselves can be divided into three categories. Each of these categories has a different usage paradigm that you should be aware of.

Syntax checkers

These are methods and snippets that were written solely for the purpose of checking my syntax in the book. To check my syntax, I leverage the features of Eclipse 3.0M4. However, be aware that the syntax checker examples will often be mixed with other examples in the same class file.

Demonstrators

These examples demonstrate a specific concept but are not executed. They often take the form of methods, which take a certain input and produce a certain output. Mixed in with these samples, you will occasionally find little `main()` methods. I use these simply to test things. If you want to play with them, feel free to do so; however, I do not discuss them in the book.

Some of the demonstrators are also used to demonstrate compiler errors when using certain techniques. To use these examples, you can try changing the files and rebuilding to demonstrate the concept. To compile a single file, there is a special Ant target named `compile_example`. To use the target, simply pass the filename you want to compile in the property `example`:

```
>ant -Dexample=oreilly/hcj/review/RTTIDemo.java compile_example
Buildfile: build.xml

init:

compile_example:
    [javac] Compiling 1 source file to C:\dev\hcj\bin
    [javac] C:\dev\hcj\src\oreilly\hcj\review\RTTIDemo.java:54: incompatible types
```

```
[javac] found   : oreilly.hcj.review.RTTIDemo.A
[javac] required: oreilly.hcj.review.RTTIDemo.B
[javac]     b = (A)a1;  // compiler error: a1 is not a B.
[javac]         ^
[javac] C:\dev\hcj\src\oreilly\hcj\review\RTTIDemo.java:55: inconvertible types
[javac] found   : oreilly.hcj.review.RTTIDemo.C
[javac] required: oreilly.hcj.review.RTTIDemo.B
[javac]     b = (B)c;  // compiler error: c is not a B.
[javac]           ^
[javac] 2 errors
```

Executables

Unlike demonstrators, executables are intended to be run and the output examined to demonstrate a concept or prove a point to a skeptical audience. When one of these programs is introduced, I will show you how to run it using Ant:

```
>ant -Dexample=oreilly.hcj.review.ObjectIsGod run_example
run_example:
     [java] class oreilly.hcj.review.ObjectIsGod$SomeClass --|> class
java.lang.Object
     [java] class oreilly.hcj.review.ObjectIsGod$SomeOtherClass --|> class
java.lang.Object
```

The emphasized line gives you the command needed to run the example after the prompt (>). The command is identical in most cases. The only difference is the name of the property `example` that you pass to the `run_example` target. While we are on the subject of running the sample code, there is one thing to note about the output. Since all of the examples are run with Ant to get the classpath and other housekeeping done, the actual output from the command will be much longer:

```
>ant -Dexample=oreilly.hcj.review.ObjectIsGod run_example
Buildfile: build.xml

init:

run_example:
     [java] class oreilly.hcj.review.ObjectIsGod$SomeClass --|> class
java.lang.Object
     [java] class oreilly.hcj.review.ObjectIsGod$SomeOtherClass --|> class
java.lang.Object
BUILD SUCCESSFUL
Total time: 1 second
```

Although this is the actual output, most of it is trivial and common to every use of Ant. Therefore, I snip out all of this housekeeping for the sake of brevity. The emphasized lines will be taken out when the run is presented in the book. Therefore, when you run the examples, be aware that Ant is a bit more verbose than I am.

Tools

One of the most important skills in professional development is knowing how to use tools. There are a wide variety of tools available, from the standard text editor and compiler to full-blown IDEs that do everything for you. Selecting the best tools for the job will make you a more productive developer.

UML Diagramming

For creating diagrams in UML, I use a product called Magic Draw UML, by No Magic, Inc. (*http://www.magicdraw.com/*). This tool is, without a doubt, the best professional UML modeling tool on the market. Rational Rose and Together can't even touch the functionality and quality of Magic Draw. I like it so much that I bought a copy of the Enterprise edition for myself. I highly recommend this product. Although it isn't free like other tools I recommend, it is well worth the price.

IDE

The IDE I use is Eclipse 3.0M4, which happens to be the IDE I use professionally as well. Eclipse simply has the single best development tool on the market. I don't know how I could live without my refactoring tools and the other goodies that come with Eclipse. You can find Eclipse at *http://www.eclipse.org/*. Also, I use many Eclipse plug-ins to make my job easier. They can be found in the Community section of eclipse.org, or you can surf the best directory of Eclipse plug-ins at *http://eclipse-plugins.2y.net/eclipse/index.jsp*.

Out-of-IDE Building

For building outside of my IDE and running examples, I use Apache Ant 1.5, which is available from *http://ant.apache.org/*. Ant is simply the best make program ever invented. I take my hat off to the folks at Apache.

Logging

I use Jakarta Log4J to do logging in my programs. Log4J is available from *http://jakarta.apache.org/log4j/docs/index.html*. In much of the sample code, there is little logging. However, in production systems, I am a logging fanatic.

For those of you that are curious, I don't use the JDK 1.4 logging mechanism because it is, in my opinion, vastly inferior to Log4J on many levels.

Jakarta Commons

Another set of libraries that I often use in my professional code is the Jakarta Commons Libraries. These libraries are available from *http://jakarta.apache.org/commons/index.html*. They extend the JDK to include things that Sun either forgot to include or decided not to include in the JDK. Many of the common tools you will learn about later in the book, such as ConstantObject, will be submitted to Jakarta Commons after this book is published to make them more reusable. For now, though, you will find these tools in the *Hardcore Java* source code itself. If you haven't checked out the Commons Libraries, I strongly advise you to do so.

Using Code Examples

This book is here to help you get your job done. In general, you may use the code in this book in your programs and documentation. You do not need to contact O'Reilly for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books *does* require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation *does* require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Hardcore Java*, by Robert Simmons, Jr. Copyright 2004 O'Reilly Media, Inc., 0-596-00568-7."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at *permissions@oreilly.com*.

Comments and Questions

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
(800) 998-9938 (in the United States or Canada)
(707) 829-0515 (international or local)
(707) 829-0104 (fax)

There's a web page for this book that lists errata, examples, and any additional information. You can access this page at:

http://www.oreilly.com/catalog/hardcorejv

To comment or ask technical questions about this book, send email to:

bookquestions@oreilly.com

For more information about books, conferences, Resource Centers, and the O'Reilly Network, see the O'Reilly web site at:

http://www.oreilly.com

Acknowledgments

I would like to thank all of the thousands of people who have answered my endless questions over the years in my quest to become a professional software engineer. Without these anonymous thousands on the Internet, the path would have been much more difficult.

I would also like to thank all of the junior programmers that have worked for me and kept my eyes open to new ideas. There is nothing like a fresh mind to suggest things that are radical, untried, and, in the end, brilliant. Certain people leap to mind, such as Bettina Linssen, now a senior developer. Even though I had considered myself something of a guru in Java GUI coding, her fresh insight and ideas taught me a thing or two. All you senior developers out there would be well advised to listen to the ideas of your junior developers. You never know what they might teach you.

I would also like to extend very special thanks to Marco Kukulies. Marco started his programming career as a junior developer working for me. He excelled beyond all expectations and demonstrated that he has that special gift that separates the normal programmers from the true gurus. Marco is now one of the best senior developers and architects I know. He has earned my respect, trust, and eternal friendship. Marco reviewed all of my book from a reader's perspective and gave me invaluable advice on what needed to be clarified, expanded on, or removed.

I would also like to thank one of my best friends who has little to do with programming but has provided endless moral support to keep me energized and working. Although my friend Saćir Husejnovic isn't a programmer, and generally doesn't understand much of my work, without his and Marco Kukulies' moral support, this book simply wouldn't have happened. My deepest respect goes out to these two gentlemen as well as to Saćir's wife Mirsada, and his three children Aida, Selma, and Amela. Thanks for being a second family to me!

I would also like to thank my new wife Alma for being the wonderful person she is and supporting me through all of my efforts to write and maintain this book.

In addition, I would like to thank all of my readers from all over the world who have submitted errata. Your vigilance has helped refine my work and kept me honest. I look forward to your feedback, concerns and questions.

On the publishing side of things, I would like to thank the artists at O'Reilly who did such a great job with my innumerable diagrams and drawings, not to mention the awesome cover. Also, I can't forget everyone who did such a good job publishing, distributing, and marketing this book. My compliments to you all.

Last, but definitely not least, I would like to thank my editor Brett McLaughlin, who can be found buried under piles of electronic chapters. Just look for the hand reaching up frantically for help. His editing prowess has increased the quality of my work to a level I never knew possible. When I proposed this book to O'Reilly, it was in a much rougher state than it is now. With many publishers, I would have been mostly on my own. However, O'Reilly has really worked hard to help me make this book one that I can be proud of. I have learned a lot, and Brett has truly converted me from a newbie author to a professional.

Table of Contents