<<Java 7程序设计>>

图书基本信息

书名：<<Java 7程序设计>>

13位ISBN编号：9787111390305

10位ISBN编号：711139030X

出版时间：2012-8

出版时间：机械工业出版社

作者：克尼亚万

页数：457

前言

欢迎阅读本书。
Java是一种很容易学会的成熟的编程语言，同时，它也汇集各种技术于一身，经常令初学者不知道从何入手。
如果你也有同感，那么本书就很适合你，因为这是一本特意为初学者量身定制的教程。
作为面向初学者的教程，本书不是要教会你每一种Java技术（因为薄薄一本书是不可能囊括所有Java技术的，这也是大部分Java类书籍都只专注于某一种技术的原因）。
应该说，本书涵盖了最重要的Java编程技术，使你能以此为基础进而学习其他技术。
本书的内容依然丰富，只要你真正理解所有章节的内容，并且完成练习，你将完全能够胜任一名中级Java程序员的日常工作。
本书共三大主题，这些是一名专业级Java程序员必须熟练掌握的内容： Java编程语言； Java面向对象编程（OOP）； Java核心类库。
要想把以上三大主题组织好一本高效的Java教材之所以困难，正是因为以下两方面的原因。
一方面，Java是一种面向对象编程（OOP）的语言，如果你已经了解OOP的话，就会知道它的语法是比较容易学的。
另一方面，OOP特性（如继承、多态以及数据封装），用现实的案例进行示范也很容易。
问题在于，理解真实的Java程序需要具备Java核心类库的知识。
由于它们之间的相辅相成关系，这三大主题并没有划分成三个独立的部分，而是讨论某一主题的章节会与讨论另一主题的章节相互交织在一起。
例如，在解释多态之前，我要先确保你了解某些Java类，之后才能讲解真实的案例。
此外，如泛型这样的语言特性，如果你事先不理解某些类，是无法解释清楚的，因此我就会在讲完那些基础类之后再讨论泛型。
还有这样的情况：某一个主题可能会在两个甚至更多个地方出现。
例如，for语句是一项基本的语言特性，它应该在较前面的章节中进行讨论。
但是，for还可以用来迭代一组对象，这种特性则又只能在讲完“集合框架”之后再做讨论。
因此，for最早在第3章中出现，之后在第11章中再度出现。
接下来，我要在此首先高度概述一下Java，粗略介绍一下面向对象编程（OOP）的概念，并简要地描述每一章的主要内容，以及讲述如何安装Java软件。
Java语言和技术 Java不仅是一种面向对象的编辑语言，它还是使软件开发变得更加快速并且获得更健壮、更安全的应用程序的一系列技术。
多年来，Java之所以会成为首选的编程技术，是因为它具有如下优势。
平台独立性 易用性 加速应用程序开发的完整类库 安全性 可扩展性 广泛的行业支持 Sun Microsystems公司于1995年开发出Java，随后Java（尽管它从一开始就是一种通用的语言）成为编写Applet（在Web浏览器上运行的小程序，为静态的网站增添交互性）的著名语言。
互联网的快速发展有很大一部分要归功于Java早期的成功。
然而，Applet并不是令Java魅力四射的唯一因素。
Java最吸引人的另一个特性是它的平台独立性承诺，也就是“一次编写，到处运行”（Write Once， Run Anywhere）的口号。
这意味着编写一个程序就可以在Windows、UNIX、Mac、Linux及其他操作系统上运行。
这是任何其他编程语言都无法实现的。
当时，C和C++是开发常规应用程序时最常用的两种语言。
Java自诞生之日起，似乎就抢尽了它们的风头。
这就是Java 1.0。
1997年，Java 1.1发布了，它增加了一些更加重要的特性，例如，更好的事件模型、Java Beans以及国际化（internationalization，i18n）。

1998年12月，Java 1.2发布了。
在它发布3天之后，版本号由1变成了2，它标志着于1999年开始掀起的一场以Java作为“下一代”技术的销售战役拉开了序幕。
Java 2以4种风格出售：标准版（J2SE）、企业版（J2EE），移动版（J2ME），以及Java Card版本（从未在这个品牌名称中采用“2”）。
接下来于2000年发布了1.3版，也就是J2SE 1.3。
两年之后发布1.4版，即J2SE 1.4。
J2SE 1.5于2004年发布。
但是Java 2 的1.5版随后改成了Java 5。
2006年11月13日，即Java 6正式发布前的一个月，Sun Microsystems公司宣布Java变成开源了。
Java SE 6是Sun Microsystems公司诚邀外部开发者贡献代码和帮助修复bug后的第一个Java版本。
其实，Sun公司过去也曾接受过非本公司员工的参与，如Doug Lea在多线程方面的付出，但这是Sun公司第一次发出公开的邀请。
Sun公司承认他们的资源有限，在不久的将来，外界的参与者将会帮助他们画上完美的句号。
2007年5月，Sun公司把其Java源代码作为免费软件在OpenJDK社区发布。
IBM、Oracle和Apple公司随后也都纷纷加入了OpenJDK。
2010年，Oracle公司收购了Sun公司。
Java 7（代号Dolphin）于2011年7月发布，这也是通过OpenJDK进行开源合作的成果。
是什么使Java平台能够独立　　你一定听说过“平台独立”或者“跨平台”这类术语，它意味着程序可以在多种操作系统上运行。
这也是使Java深得人心的一大原因。
但是，到底是什么使Java得以实现平台独立呢？
在传统的编程中，源代码是要编译成可执行代码的。
这种可执行代码只能在所设计的平台上运行。
换句话说，为Windows而编写和编译的代码就只能在Windows上运行，在Linux中编写的代码就只能在Linux上运行，等等。
传统的编程范例　　而Java程序则编译成字节码（bytecode）。
你不能让它自己运行字节码，因为它不是本机代码（native code）。
字节码只能在Java虚拟机（JVM）上运行。
JVM是一种解释字节码的本机应用程序。
Sun公司使JVM在众多平台上都可用，从而把Java变成一种跨平台的语言。
如图I.2所示，同一个字节码就可以在任何操作系统的JVM上运行。
Java编程模式　　目前，JVM适用于Windows、UNIX、Linux、Free BSD，以及世界上在用的其他的所有主流操作系统。
JDK、JRE、JVM之间有何区别　　我说过，Java程序必须进行编译。
事实上，任何编程语言都需要一个编译器，经过编译后编程语言才真正有用。
编译器是一种将程序源代码转化成可执行格式（如字节码、本机代码或者其他）的程序。
在开始利用Java编程之前，必须先下载一个Java编译器。
Java编译器是一个名为javac的程序，它是Java compiler的缩写。
虽然javac可以将Java源代码编译成字节码，但要运行字节码，还需要一个Java虚拟机（JVM）。
此外，由于你常常需要用到Java核心类库中的类，因此还需要下载这些类库。
Java运行时环境（Java Runtime Environment，JRE）包含JVM和类库。
当然，Windows的JRE与Linux的JRE不同，也就是说，某一种操作系统的JRE与另一种操作系统的JRE是不同的。
Java软件有两个发行包：　　JRE，它包括JVM和核心类库，最适合用来运行字节码。
JDK，它包括JRE，外加一个编译器和其他工具。
它是编写和编译Java程序的必备软件。

简而言之，JVM就是一种运行字节码的本机应用程序。
JRE则是一种包含JVM和Java类库的环境。
JDK则包含JRE及一个Java编译器和其他工具。
JDK的第一个版本是1.0，之后的版本依次是1.1、1.2、1.3、1.4、1.5、1.6和1.7。
对于微小的更新发布，则是在版本号之后再添加一个数字。
例如，1.7.1就是对1.7版本的第一次微小的更新。
JDK 1.7（代号Dolphin）就是大家熟知的JDK 7。
JDK所包含的JRE版本与JDK的相同。
因此，JDK 1.7包含JRE 1.7。
JDK经常也称做SDK（Software Development Kit的缩写）。
除了JDK之外，Java程序员还需要下载说明核心类库中类的Java文档。
从提供JRE和JDK的同一个网址可以下载说明文档。
Java 2、J2SE、J2EE、J2ME、Java 7究竟为何物　　Sun公司为推广Java做了很多努力。
它的部分营销策略是冠以Java 2的名称，实际上就是JDK 1.2。
Java 2有三种版本：　　Java 2 Platform，标准版（J2SE）。
J2SE其实相当于JDK。
它也是J2EE中所定义的各种技术的基础。
Java 2 Platform，企业版（J2EE）。
它为开发基于组件的多层企业应用程序定义了标准。
功能包括Web Services特性支持和开发工具（SDK）。
Java 2 Platform，移动版（J2ME）。
它为在消费者设备上运行的应用程序提供了一种环境，如移动电话、掌上电脑（PDA），以及电视机顶盒等。
J2ME包括一个JVM和有限的几组类库。
在版本5中出现了名称的变化。
J2SE变成了Java Platform，Standard Edition 5（Java SE 5）。
而且J2EE和J2ME中的“2”也去掉了。
企业版的当前版本是Java Platform，Enterprise Edition 6（Java EE 6）。
J2ME现在称做Java Platform，Micro Edition（Java ME，没有版本号）。
在本书中，Java 7就是指Java SE 7。
与Sun公司出产的第一个Java版本不同，Java SE 7（包括J2SE 1.4、Java SE 5以及Java SE 6）是一系列规范，这些规范定义了在发布时需要实现的各种特性。
软件本身称做参考实现（Reference Implementation）。
Oracle、IBM和其他公司一起，通过OpenJDK提供了Java SE 7参考实现，以及Java后续版本的参考实现。
Java EE 6也属于规范，其中包含像Servlet、JavaServer Pages、JavaServer Faces、Java Messaging Service等技术。
为了开发和运行J2EE和Java EE 6应用程序，还需要一个J2EE／Java EE 6应用服务器（application server）。
任何人都可以实现一个J2EE／Java EE 6应用服务器。
这说明市场上各种应用服务器很容易获取到，其中包括一些开源的应用服务器。
下面列举了一些J2EE／Java EE 6应用服务器的例子：　　Oracle公司的WebLogic（之前BEA的WebLogic）　　IBM公司的WebSphere　　Sun公司的Sun Java Application Server　　Oracle公司的10g Application Server　　GlassFish　　JBoss　　Jonas　　Apache Geronimo　　JBoss、GlassFish、Jonas以及Geronimo都是开源的应用服务器。
不过它们有不同的许可要求，因此在决定使用这些产品之前，一定要确保先读过许可内容。
Java社区进程程序　　Java之所以能够持续成为技术首选，有很大一部分要归功于Sun公司的策略

，它允许其他行业的人士参与决定Java的未来。
这使得很多人都觉得Java也属于他们每一个人。
许多大公司（如IBM、Oracle、Nokia、Fujitsu等）都大力投资Java，因为它们也可以为某一种技术提出一种规范，并将它们想要的技术放到Java技术的下一个版本中。
这种协同工作正是采用了Java社区进程（Java Community Process，JCP）程序的形式。
它的网址是http：//www.jcp.org。
由JCP程序提出的规范，就是大家所熟悉的Java Specification Requests（JSR）。
例如JSR 336定义了Java SE 7。
面向对象编程概述　面向对象编程（Object-Oriented Programming，OOP）是根据真实的对象来构建应用程序模型而实现的。
OOP的三个原则是封装、继承和多态。
OOP的好处是实实在在的。
这些正是大多数现代编程语言（包括Java），均是面向对象（OO）的原因所在。
我甚至可以列举为了支持OOP而做出的两个著名的语言转变：C语言演变成C++，Visual Basic更新成Visual Basic.NET。
本节将阐明OOP的优势，并评估学习OOP的难易程度。
OOP的优势　OOP的优势包括代码易维护、可重用，以及易扩展。
下面就详细地介绍这些优势。
1. 易维护。
现代的软件应用规模往往都十分巨大。
以前，“大”系统包括几千行代码。
现在，即使有上百万行代码也算不上大。
C++之父Bjarne Stroustrup曾经说过，当一个系统变得越来越大时，就会开始给它的开发者带来很多问题。
一个小程序可以用任何语言以任何方式编写。
如果你不轻易放弃，最终还是可以使它运行起来。
但大型的程序就是另一回事了。
如果没有采用“好的编程”方法，一边修复错误的同时会不断出现新的错误。
其原因在于，大型程序的不同部分之间是相互依赖的。
当在程序的某个部分中修改某些东西时，可能不知道这个变化会对程序的其他部分造成什么样的影响。
OOP则可以很容易地使程序模块化，这种模块性大大减少了维护的问题。
在OOP中，模块性是可以继承的，因为类（对象的一个模板）本身就是一个模块。
好的设计应该允许类包含类似的功能性和有关的数据。
OOP中经常用到的一个重要的相关术语是耦合，它表示两个模块之间的交互程度。
不同部分之间的松耦合会使代码更容易实现重用，这就是OOP的另一个好处。
2. 可重用。
可重用意味着之前写好的代码可以被代码的创建者或需要该源代码所提供功能的其他人重用。
那么，OOP语言经常提供一些预先设计好的类库也就不足为奇了。
在Java中，它配套提供了几百个类库或API（应用编程接口），这些都是经过精心设计和测试过的。
编写和发布自己的类库时也非常容易。
支持编程平台中的可重用性，这是十分吸引人的，因为它可以大大缩短开发时间。
类可重用性的一大困难是，要为类库创建好的文档。
作为一名程序员，他要找到一个能够为其提供所需功能的类能有多快？
是找一个这样的类快呢，还是从头写一个新的类更快？
值得庆幸的是，Java核心和扩展的API都提供了详实的文档。
可重用性不仅适用于重用类和其他类型的编码阶段；在OO系统中设计应用程序时，针对OO设计

问题的解决方案也可以重用。
这些解决方案称做设计模式。
为了便于引用各解决方案，每种模式都有一个名字。
可重用设计模式的早期说明请见经典著作《Design Patterns：Elements of Reusable Object-Oriented Software》，由Erich Gamma、Richard Helm、Ralph Johnson和John Vlissides合著。

3. 可扩展性。

每一个应用程序都是独特的。
它有自己的需求和规范。
说到可重用性，有时候你可能找不到现成的类来为你提供应用程序所需的精确功能。
但是，你或许可以找到一两个能够提供部分功能的类。
可扩展性的意思是，你仍然可以使用那些类，方法是对它们进行扩展，使其满足你的需要。
这样做你依然节省了时间，因为不必从头编写代码。

在OOP中，可扩展性是通过继承来实现的。
可以扩展现成的类，对它添加一些方法或者数据，或者修改不喜欢方法的行为。
如果你知道某个基本功能将多次使用，但又不想让类提供太具体的功能，就可以提供一个泛型类，以后可以对它进行扩展，使它能够提供特定于某个应用程序的功能。

学习OOP很困难吗　　Java程序员必须掌握OOP。
事实上，如果你使用过程语言（如C或者Pascal）编过程序，就会发现它们之间有很大的区别。
这其中有好有坏。

先说坏的方面。

研究人员一直在探讨在学校里传授OOP的最佳方法；有人认为，最好在教OOP之前先教过程编程。
我们看到有许多地方，都是在学生临近大学毕业时才上OOP课程。

但是现在有越来越多的学生认为，具备过程编程能力的人，在模式中的思维，与OO程序员看待和解决问题的方式有很大的不同。
当需要学习OOP时，他所面临的最大困难是要克服模式转换。
据说，要经过6～18个月才能将一个人的思维从过程模式转换成面向对象的模式。
另有研究表明，没有学过过程编程的学生则不觉得OOP有多么难学。

接下来说好的方面。

Java堪称最容易学习的OOP语言之一。
例如，你不需要操心指针，不必浪费宝贵的时间去解决因没有销毁不用的对象而导致的内存泄漏等。
此外，在它们早期的版本中，还提供了非常全面的类库，并且bug相对要少得多。
一旦掌握了OOP，用Java编程就变得非常容易了。

关于本书　　下面概述一下各章节的主要内容：　　第1章旨在让读者体验使用Java的感觉。
其中包括编写一个简单的Java程序，利用javac工具对它进行编译，并利用java程序运行它。
此外，还提出了一些关于编码规范和集成开发环境的建议。

第2章讲解Java语言的语法。
该章将介绍字符集、基本类型、变量、运算符等话题。

第3章讲解Java语句for、while、do-while、if、if-else、switch、break和continue。

第4章是本书中第一节关于OOP的课程。
该章首先说明什么是Java对象，以及它在内存中是如何保存的。
随后继续讲解类、类成员，以及两个OOP概念（抽象和封装）。
另外还简单地介绍了一些有关的主题，如垃圾回收和对象比较。

第5章涵盖Java核心类库中重要的那些类：java.lang.Object、java.lang.String、java.lang.StringBuffer和java.lang.StringBuilder、包装类，以及java.util.Scanner。
这里面还介绍了装箱／拆箱和可变参数。
这是十分重要的一章，因为其中介绍的类是Java中最常用的一部分类。

第6章介绍一种能够使代码具有可扩展性的OOP特性。
该章教你如何扩展类，改变子类的可见性，覆盖方法等。
毫无疑问，错误处理是所有编程语言的一项重要特性。
作为一种成熟的语言，Java具备十分强健的错误处理机制，它可以有效地防止bug蔓延。
第7章详细介绍这一机制。
第8章讲解在使用数字和日期时要处理的三个问题：解析、格式化和操作。
该章介绍可以帮助你完成这些任务的Java类。
第9章说明接口不仅仅是一个没有实现的类，它还定义服务提供者和客户端之间的契约。
本章讲解如何使用接口和抽象类。
第10章主要讲解枚举，这是从Java 5开始添加的一个类型。
第11章说明如何利用java.util包中的成员，将对象进行分组，并对它们进行操作。
泛型是Java中非常重要的一项特性，第12章详细解释这一特性。
第13章介绍流的概念，并阐述如何利用java.io包中的4个流类型来执行输入／输出操作。
此外，还讲解对象序列化和反序列化。
第14章讲解如何在一个类的内部编写另一个类，并说明为什么这项OOP特性非常有用。
第15章是本书中关于Swing的两章之一，这是第一章。
它简单地介绍AWT组件，并详细介绍一些基础的Swing组件。
第16章是关于Swing的第二章。
它介绍一些更高级的技术，如布局管理、事件处理，以及Swing的外观。
多态是OOP的重要利器之一。
当我们在编译中不知道对象的类型时，多态的作用尤其出乎我们的想象。
第17章讲解这一特性，并给出很好的范例。
第18章讨论Java中的一项特性。
它介绍JDK中配套的标准注解、一般注解、元注解，以及定制注解。
如今，软件应用程序可以部署到不同的国家和地区，这是很平常的事。
这些应用程序切记要设计成国际化。
第19章探讨Java程序员可以使用的方法。
Applet是指可以在Web浏览器上运行的小程序。
第20章讲解Applet的生命周期、安全性限制，以及JApplet。
第21章介绍了网络编程中使用的类。
这里用一个简单的Web服务器应用程序来说明如何使用这些类。
访问数据库和操作数据是业务应用程序中最重要的部分任务。
数据库服务器种类繁多，访问不同的数据库需要不同的技能。
值得Java程序员感到庆幸的是，JDBC（Java Database Connectivity）技术为访问数据库提供了统一的方法。
JDBC将在第22章讨论。
线程是指操作系统分配处理器时间的一个基本处理单元，在一个进程里面可以有不止一个线程在执行代码。
第23章说明在Java中多线程编程并不是只有专家级程序员才能做到。
第24章是本书中关于多线程编程的第二章。
这里介绍了能使多线程编程变得更加容易的接口和类。
第25章讲解Java应用程序的用户可以如何限制运行Java应用程序，以及可以如何利用加密术来保护应用程序和数据的安全。
第26章探讨Servlet技术和Servlet API，并举了几个例子。
第27章介绍另一种Web开发技术，并讲解如何编写JSP页面。
第28章介绍javadoc工具，Java程序员可以用它为API产生文档。
第29章介绍Java Web Start，以及如何用它将Java应用程序部署到互联网、局域网，甚至如何从CD

进行部署。

附录A、附录B和附录C分别介绍javac、java和jar工具。

附录D和附录E分别简要地讲解NetBeans和Eclipse开发工具。

<<Java 7程序设计>>

内容概要

本书由全球资深Java技术专家、高级Java企业级应用架构师、《How Tomcat Works》作者亲自执笔，权威性毋庸置疑。
它以最新的Java 7为基础，全面讲解了Java编程语言、Java面向对象技术和Java核心类库三大主题，包含大量案例，是系统学习Java 7程序设计的Bible级著作。
本书是Java SE 7程序设计全面、系统的教程，涵盖Java编程语言、Java OOP和Java核心类库3大主题。
本书首先介绍了Java 7的语法、语句、对象和类、核心类、继承、错误处理等基础知识，帮助读者快速入门Java 7；然后深入解析了接口和抽象类、枚举、集合框架、泛型、输入输出、嵌套类和内部类等内容，掌握这部分内容有助于深入理解Java的底层原理；接着阐述Swing的基础和高级知识、多态、注解、国际化、Java网络、JDBC、Java线程、并发工具、安全、Java Web应用程序、JavaServer Pages、Javadoc，以及应用程序部署等内容，掌握这部分内容有助于提升编程技能。

作者简介

Budi Kurniawan
资深Java技术专家和Java企业级应用架构师，是软件咨询和技术出版公司Brainy Software的创始人，经验十分丰富。
他还是一位经验丰富的技术作家，撰写了深入揭示Tomcat工作原理和设计理念的名著《How Tomcat Works》（中文名称：《深入剖析Tomcat》，机械工业出版社出版）和《Struts Design and Programming》，并在多种权威出版物上发表过100多篇文章。

<<Java 7程序设计>>

书籍目录

章节摘录

第1章　初识Java　开发Java程序包括编写代码，将它编译成字节码，然后运行这些字节码。这是Java程序员在职业生涯中要不断重复的过程，因此，这个过程的舒适度就显得尤为重要。本章的主要目的就是让你体验一下用Java进行软件开发的过程。

编写出来的代码不仅需要能够运行，更重要的是还要易于阅读和维护，因此，本章将先介绍Java编码规范。此外，由于明智的开发者将会选择使用集成开发环境（IDE）来简化他们的工作，因此本章最后一节将会介绍Java IDE。

1.1　第一个Java程序　本节将重点介绍Java开发的步骤：编写程序，将它编译成字节码，然后运行字节码。

1.1.1　编写Java程序　你可以利用任何一种文本编辑器来编写Java程序。打开一个文本编辑器，并输入代码清单1.1中的代码。当然，如果你已经下载了本书配套的程序范例，那么只需将它复制到你的文本编辑器中即可。

代码清单1.1　一个简单的Java程序　就目前而言，我们只需要将Java代码放在一个类中即可，并确保将代码清单1.1中的代码另存为MyFirstJava.java文件。所有Java源文件都必须用java作为扩展名。

1.1.2　编译Java程序　我们使用javac工具来编译Java程序，它在JDK安装目录下的bin目录中。假如在电脑中已经设置好了PATH环境变量（如果还没有设置，请查阅“引言”中的“下载和安装Java”），那么就可以从任何目录下调用javac。现在用以下的操作来编译代码清单1.1中的MyFirstJava类：　1）打开一个命令提示符窗口，并将目录切换到MyFirstProgram.java文件所在的目录。

2）输入以下命令：javac MyFirstJava.java　如果一切顺利，javac会在当前工作目录下创建一个名为MyFirstJava.class的文件。

提示　通过指定选项，可以使用javac工具提供的更多的特性。例如，可以指定将新生成的类文件放置的在什么位置。关于javac工具使用方法的详细介绍，请参见附录A。

1.1.3　运行Java程序　我们使用JDK的java工具来运行Java程序。当然，只有添加了PATH环境变量，才能够从任何目录下调用java指令。在上述命令窗口的路径后输入以下代码，并按回车键：java MyFirstJava　注意，在运行Java程序时，不需要包含class扩展名。

这时，就可以在控制台中看到以下代码：Java rocks.　恭喜！你已经成功地编写了第一个Java程序！由于本章的唯一目的就是让你亲身体验编写和编译的过程，因此将不在此解释这个程序的工作原理。

你还可以给Java程序传递参数。例如，要传递给Calculator类两个参数，可以这样写：　java Calculator arg-1 arg-2　这里的arg-1是第一个参数，arg-2是第二个参数。你想传递多少个参数都可以，java工具会将这些参数作为一个字符串数组提供给要运行的Java程序。第5章将教你使用参数。

提示：java工具是一个高级程序，它也提供了很多配置选项。例如，可以设置要分配给它的内存大小。这些选项的解释请参见附录B。

提示：java工具还可以用来运行打包在jar文件中的Java类。详情请参见附录C中的C.4节。

……

<<Java 7程序设计>>

编辑推荐

·全球资深Jvava技术专家、《How Tomcat Works》作者最新力作！
·基于最新Java 7，全面讲解Java编程语言、Java面向对象技术和Java核心类库三大主题，为系统学习Java 7程序设计提供绝佳指导。

<<Java 7程序设计>>

版权说明