Sun公司核心技术丛书

HZ BOOKS

PH PTR

最新 Java 2

核心技术 卷II：高级性能 (V1.3) 5E

Core Java 2 Volume II Advanced Features

Sun microsystems

(美) Cay S. Horstmann
Gary Cornell 著

王建华 董志敏 杨保明 等译

附赠 CD-ROM

机械工业出版社
China Machine Press

Sun公司核心技术丛书

最新Java 2核心技术

卷II：高级性能 (v1.3)5E

（美） Cay S.Horstmann
Gary Cornell 著

王建华 董志敏 杨保明 等译

机械工业出版社
China Machine Press

本书是一本供Java编程人员使用的关于Java 2平台（包括JDK 1.3的完整更新版、JDK 1.4版的标准版）的高级参考书，是《最新Java 2核心技术 卷Ⅰ：原理》的姊妹篇。卷Ⅰ介绍Java技术的基本特性，而卷Ⅱ则主要介绍Java技术的高级特性。全书共分12章，各章内容互相独立，读者可以根据自己的需要，参阅你最感兴趣的技术信息。

本书内容丰富、覆盖面广，极具实用价值。书中大量的实例代码及新增和修改后的特性为Java程序设计员提供了极好的指导。

Simplified Chinese edition copyright © 2002 by PEARSON EDUCATION NORTH ASIA LIMITED and China Machine Press.

Original English language title: *Core Java 2, Volume Ⅱ, Advanced Features, 5E*, by Cay S.Horstmann and Gary Cornell, Copyright 2002.

All rights reserved.

Published by arrangement with the original publisher, Pearson Education, Inc., publishing as Prentice Hall.

This edition is authorized for sale only in the People's Republic of China (excluding the Special Administrative Region of Hong Kong and Macau).

本书封面贴有Pearson Education培生教育出版集团激光防伪标签，无标签者不得销售。

版权所有，侵权必究。

本书版权登记号：图字：01-2002-3652

图书在版编目（CIP）数据

最新Java 2核心技术，卷Ⅱ：高级性能（V1.3）5E/（美）霍斯特曼（Horstmann，C.S.），（美）科奈尔（Cornell，G.）著；王建华等译. -北京：机械工业出版社，2003.1

（Sun公司核心技术丛书）

书名原文：Core Java 2, Volume Ⅱ, Advanced Features, 5E,

ISBN 7-111-11381-5

Ⅰ.最… Ⅱ.①霍… ②科… ③王… Ⅲ.JAVA语言-程序设计 Ⅳ.TP312

中国版本图书馆CIP数据核字（2002）第108461号

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：周 睿

北京忠信诚胶印厂印刷·新华书店北京发行所发行

2003年1月第1版第1次印刷

787mm×1092mm 1/16·63.25印张

印数：0 001-5 000册

定价：108.00元（附光盘）

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

前　言

致读者

你手中拿着的这本书是《Core Java》第5版第Ⅱ卷的中文版。本书第1版于1996年初问世，第2版于1996年底推出，第3版于1997/1998年面市。前两个版本都是单卷版本，不过第2版的篇幅已经比第1版长了许多，因此它本身不是一本薄薄的书了。当我们着手开发第3版时，我们感到一卷的篇幅显然已经无法容纳编程人员需要了解的Java平台的所有特性了。为此，我们决定从第3版起分成两卷来编写。在此版中，我们再次将它编写成两卷。不过，我们对内容进行了创新性的安排，将数据流的特性放入第I卷，而将集合的特性放入了第Ⅱ卷。

第I卷主要讲述Java语言的基本特性，而第Ⅱ卷则主要介绍编程人员进行专业软件开发时需要了解的高级特性。因此，与第Ⅰ卷以及本书的以前版本一样，我们编写的这本书将仍然是面向想要将Java技术用于开发实际软件项目的编程人员的。

请注意，如果你是个经验丰富的软件开发人员，非常熟悉新的事件模型和高级语言特性，那么你就不需要阅读第I卷（不过在第Ⅱ卷中，我们仍然要根据情况适当地引用第I卷中有关章节的内容，并且当然希望你会购买或者已经购买了第I卷，这样你就可以在该书中找到你所需要的关于Java平台的背景材料）。

最后要说明的是，编写任何一本书都会不可避免地存在一些错误和不准确的地方。我们非常愿意听到你对本书的意见。当然，我们希望这样的意见只出现一次。为此，我们建立了一个Web网站，网址是http://www.horstmann.com/corejava.html，上面设有FAQ（常见问题解答）、错误的修改、替代方案等栏目。我们在软件错误报告的Web页的结尾（目的是鼓励你阅读以前的报告）放置了一个窗体，你可以用它来报告软件中存在的错误和问题，提出你的建议，以便改进将来的版本。

本书内容的编排

本书中的大部分章节都是独立的。你可以查阅最感兴趣的问题，并可按照任意顺序阅读这些内容。

第1章介绍多线程，运用多线程技术，你就能够对各个任务进行编程，以便并行执行这些任务（所谓线程，就是程序中的一个控制流）。在本章中我们将要介绍如何建立线程，以及如何确保这些线程不会因为错误而停止运行。我们将用示例代码来说明线程的实际用法，告诉你建立定时器和动画时使用的技术。

第2章介绍Java 2平台的集合框架。每当你想要收集多个对象，并且在以后检索这些对象的时候，就需要使用最适合环境的集合，而不是将各个元素纳入一个向量（Vector）中。该章将介绍如何利用预先已为你建好的标准集合。

第3章将讲述Java平台中最令人感兴趣的一个API，即网络API。Java使它能够非常容易地进行复杂的网络编程。在该章中我们不仅深入介绍了这个API，而且还讲述了applet安全模型在用于网络编程时的重要作用。

第4章介绍JDBC，这是个Java数据库连接API。我们将要向你介绍如何使用JDBC API的核心子集来编写非常有用的程序，以便处理实际中的数据库日常操作任务。请注意，这一章并不是全面介绍功能丰富的JDBC API能够执行的所有任务（如果要全面介绍JDBC API的功能，可能需要写一本像本书一样厚的专著才行）。

第5章讲述远程对象和远程方法调用（RMI）。这个API使你能够对分布在多个机器上的Java对象进行各种操作。我们还要向你介绍何处才能够切实地使用“无处不在的对象”的强大功能。

第6章包含了没有纳入核心Java第I卷的所有Swing内容，尤其是重要而复杂的树结构和表格组件。我们将要说明编辑器窗格的基本用途，以及建立“多个文档”界面时使用的Java技术。在该章中我们将把重点放在实际编程中可能遇到的最有用的构件上。全面介绍整个Swing库将需要编写若干卷才行，并且只有专门的编程人员才对它们感兴趣。

第7章介绍你在建立实际图形时可以使用的Java 2D API。该章还要介绍抽象窗口操作工具包（Abstract Windowing Toolkit，AWT）的一些高级特性，这些特性如果要纳入第I卷的话，显得太专业了一些，但是这些特性应该成为每个编程人员工具包的组成部分。这些特性包括打印和用于剪切和粘贴以及拖放的API。实际上，我们使用的用于剪切和拖放的API比Sun公司本身开发的API向前推进了一步，也就是说，我们向你展示了如何运用系统剪贴板对用Java编程语言编写的程序之间的串行Java对象进行剪切和粘贴。

第8章介绍了用于Java平台的构件API—JavaBeans。你将学会如何编写自己的JavaBeans，使其他编程人员能够在集成式生成器环境中进行各种操作（不过我们没有介绍用于对JavaBeans进行操作的各种生成器环境）。如果Java技术要想取得最后的成功，JavaBeans组件技术是一项非常重要的技术，因为它能够像ActiveX控件一样为千百万Visual Basic编程人员在操作上提供很大的方便，使用户能够同样方便地使用各种用户界面编程环境。当然，由于这些组件是用Java编程语言编写的，因此它们的特性要优于ActiveX控件，因为它们可以直接用于其他的平台，并且能够被纳入到Java平台的复杂安全模型中。

第9章实际上就是介绍Java平台的这些安全模型的。从根本上讲，Java平台的设计是绝对安全的，该章将深入地介绍它的设计是如何进行的。我们将向你说明如何编写用于特定目的的应用程序的类加载器和安全管理器。然后介绍一个新的安全API，它可以用于实现像签名的类那样重要的特性。

第10章介绍一个我们认为其重要性将会不断提高的专门特性，那就是软件的本地化。Java编程语言是从一开始就能够处理Unicode的少数几种语言，不过Java平台中的本地化支持特性所具备的功能要强得多。因此，你可以对Java应用程序进行本地化，这样，它们不仅能够跨平台运行，而且它们的应用将不受国界的限制。例如，我们将向你展示如何编写一个退休金计算器applet，它根据浏览器所使用的语言，既可以使用英语，也可以使用德语和汉语。

第11章介绍各种本机方法（native method），使你能够调用为特定机器（比如Microsoft Windows API）编写的方法。显然，该特性的使用是有争议的，也就是说使用本机方法后，Java

试读结束：需要全本请在线购买： www.ertongbook.com

平台的跨平台性质就不再存在了。尽管如此，为特定平台编写Java应用程序的每个严谨的编程人员都必须了解这些方法。有的时候，当你编写重要的应用程序时，你必须转用你的目标平台操作系统的API。为此，我们将要向你介绍如何访问Windows中的注册表函数。

第12章介绍XML与Java之间的关系。XML并没有使Java过时，相反，它们之间配合得很好。该章将要讲述如何传递XML文档、文档类型的定义、名字空间等问题，说明应该如何使用SAX分析器，如何生成XML文档，以及如何进行XSL的转换。

本书中的一些约定

注意 提醒读者应当给予重视的地方。

提示 是对读者有所帮助的一些技巧。

警告 容易出错或者可能出现危险的情况。

C++注意 用于解释Java编程语言与C++之间的差别。如果你对C++不感兴趣，那么你可以跳过这些说明。

API Java平台配有大量的编程库或应用程序编程接口（API）。当初次使用一个API调用时，我们增加了一个简短的综述，并且加上了一个API图标。这些描述虽然不太正规，但是它们比正式在线的API文档提供的信息要稍微详细一些。

源代码在光盘上的程序是作为示例代码列出的程序，比如，“示例代码5-8：WarehouseServer.java”是指光盘上对应的这个代码。你也可以从Web上下载这些示例代码文件。

定义

“Java对象”是指使用java编程语言建立的对象。

“Java应用程序”是指使用java编程语言编写的程序，可以由Java虚拟机（即用于Java平台的虚拟机）运行。

参予本书翻译工作的人员还有：蒋小英、王卫峰、周茹、张晓佳、丁晓红、张明华、汪卫国、董明、董建平、李君英等。由于时间仓促、水平有限，疏漏之处，敬请读者批评指正。

目　录

第1章 多 线 程

本章内容提要：

- 什么是线程
- 线程的中断
- 线程的属性
- 线程的优先级
- 利己线程
- 同步
- 死锁
- 用线程进行用户界面编程
- 将管道用于线程间的通信

也许你已经非常熟悉多任务运行的概念了。使用多任务运行特性时，仿佛是在同一个时间内运行多个程序。例如，你可以在编辑或者发送传真的时候进行打印操作。当然，你必须拥有一台多处理器计算机，这实际上是由操作系统将资源分成许多小块，然后再分配给每个程序使用，这些程序看起来就像是在并行运行一样。之所以能够进行这样的资源分配，原因是虽然你认为你正在让计算机忙于接受（比如）数据的输入，但是实际上CPU在大部分时间内却是无所事事的。（当你进行高速数据键入时，每键入一个字符需要大约二十分之一秒的时间，但对于计算机来说，这是一个很长的时间间隔。）

多任务操作可以用两种不同的方式来进行，这取决于操作系统在中断程序运行之前是否首先与程序进行了咨询，还是只有当程序愿意产生控制时程序才被中断这些情况而定。前者称为抢占式多任务操作，后者称为共用式（或者可以简单的称为非抢占式）多任务操作。Windows 3.1和Mac OS 9都属于共用式多任务操作系统，而UNIX/Linux、Windows NT（和用于32位程序的Windows 95）和OS X则属于抢占式多任务操作系统（虽然抢占式多任务操作实现起来比较困难，但是它的效率要高得多。使用共用式多任务操作时，运行性能不良的程序会使其他所有程序无法正常运行）。

多线程程序扩展了多任务操作的概念，它将多任务操作降低一级来运行，那就是各个程序似乎是在同一个时间内执行多个任务。每个任务通常称为一个线程，它是控件的线程的简称。能够同时运行多个线程的程序称为多线程程序。可以将每个线程视为是在一个不同的环境中运行的，也就是说这些环境看上去好像每个线程都有它自己的CPU一样，该CPU配有寄存器、内存和它自己的代码。

那么，多个进程与多个线程之间有什么差别呢？基本的差别是，每个进程都有它自己的一组完整的变量，而线程则共享相同的数据。这听起来有些危险，并且它确实有些危险，这个问

试读结束：需要全本请在线购买： www.ertongbook.com

题你将在本章的后面部分中了解到。不过，这样一来，建立或者撤销各个线程所需要的开销将比调用新的进程少得多，正是由于这个原因，所以所有最新的操作系统都支持多线程的运行。此外，进程间的通信比线程之间的通信速度要慢得多，并且局限性更大。

多线程在实践中是极其有用的。例如，浏览器应该能够同时下载多个图像。电子邮件程序应该在它下载新邮件的同时，使你能够读取你的邮件。Java编程语言本身使用一个线程，以便在后台进行无用数据的回收，这样你就不会遇到内存管理的麻烦！图形用户界面（GUI）程序有一个专门的线程，用于收集来自主机操作环境的用户界面事件。本章将要向你介绍如何将多线程功能添加给你的Java应用程序和applet。

应该提醒你注意的是，多线程的编程可能会非常复杂。在本章中，我们将介绍Java编程语言提供给线程编程时所需的所有工具。我们将要说明它们的用途和局限性，并且提供一些简单而典型的编程例子。不过，对于比较复杂的编程，我们建议你阅读更高级的参考书，比如《Concurrent Programming in Java》（使用Java进行同时运行的程序编程），该书由Doug Lea撰写（Addison-Wesley出版社1999年出版）。

注意　在许多编程语言中，你必须使用外部线程编程工具包来进行多线程程序的编程。Java编程语言是围绕着多线程的编程建立的，因此它能够使你更加容易地进行编程。

1.1　什么是线程

让我们首先来看一下不使用多线程的程序的运行情况，用户很难用该程序来执行若干个任务。我们分析完该程序的运行情况后，我们再来向你介绍让该程序运行许多独立的线程是多么的容易。该程序使一个弹跳球动起来，方法是连续移动该球，再确定它是否从墙壁弹回来，然后对它进行刷新（见图1-1）。

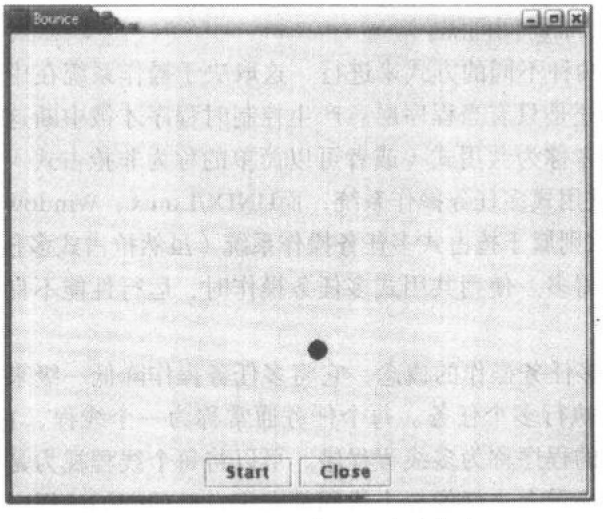

图1-1　使用线程使弹跳球动起来

当你单击“Start”按钮时，该程序便从屏幕的左上方发出一个球，然后该球便开始弹跳。“Start”按钮的处理程序将调用addBall方法。

```
public void addBall()
{
  try
   {
      Ball b = new Ball(canvas);
      canvas.add(b);
      for (int i = 1; i <= 1000; i++)
      {
         b.move();
         Thread.sleep(5);
      }
   }
   catch (InterruptedException exception)
   {
   }
}
```

该方法包含一个循环，它运行1000次移动。每次调用move时，便将球移动一点儿，如果遇到墙壁，便调整它的方向，然后刷新画布。Thread类的静态方法sleep用于暂停5毫秒。

调用Thread.sleep并不建立新线程，sleep是Thread类的一个静态方法，用于暂时停止当前线程的活动。

sleep方法能够抛出一个InterruptedException异常。我们将在后面介绍这个异常事件和对它的正确处理方法。现在，我们只是在出现该异常事件时终止球的弹跳。

如果你运行该程序，那么该球便能够很好地进行反复弹跳，不过，它会完成整个应用程序的运行。如果你在该球完成它的1000次弹跳之前，不想再让它弹跳了，并且单击“Close”按钮，该球还会继续弹跳下去。在该球完成弹跳之前，你无法与该程序进行互动操作。

注意 仔细观察本节结尾处的代码，你会发现在Ball类的move方法中有下面这样一个调用：

```
canvas.paint (canvas.getGraphics ())
```

这是非常奇怪的，通常来说，你应该调用repaint，并且让抽象窗口操作工具包（AWT）负责获取图形环境并且执行绘图操作。但是，如果你试图调用该程序中的canvas.repaint( )，你将发现自从addBall方法接管所有的处理操作以来，画布从来没有重画过。在下面的程序中，我们使用一个独立的线程来计算球的位置，我们将再次使用人们熟悉的repaint方法。

显然，该程序的运行是非常差劲的。当你要用程序执行一个很费时间的作业时，你不会希望你的程序会用这种方式来运行。当你通过网络连接来读取数据时，你的操作任务将被卡住，因此无法进行，这种情况太常见了，以至于你真想中断数据的读取工作。例如，假设你正在下载一个很大的图形，当你看到该图形的一部分时，你决定不需要或者不想看到该图形的其余部分，你当然希望能够单击“Stop”（停止）或者“Back”（退回）按钮，中断图形加载的过程。在下一节中，我们将要介绍如何通过一个独立线程中的部分关键代码，使用户能够控制程序运行的过程。

示例代码1-1是该程序的整个代码。

示例代码1-1 Bounce.java

```
1. import java.awt.*;
2. import java.awt.event.*;
3. import java.awt.geom.*;
4. import java.util.*;
5. import javax.swing.*;
6.
7. /**
8.    Shows an animated bouncing ball.
9. */
10. public class Bounce
11. {
12.    public static void main(String[] args)
13.    {
14.       JFrame frame = new BounceFrame();
15.       frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
16.       frame.show();
17.    }
18. }
19.
20. /**
21.    The frame with canvas and buttons.
22. */
23. class BounceFrame extends JFrame
24. {
25.    /**
26.       Constructs the frame with the canvas for showing the
27.       bouncing ball and Start and Close buttons
28.    */
29.    public BounceFrame()
30.    {
31.       setSize(WIDTH, HEIGHT);
32.       setTitle("Bounce");
33.
34.       Container contentPane = getContentPane();
35.       canvas = new BallCanvas();
36.       contentPane.add(canvas, BorderLayout.CENTER);
37.       JPanel buttonPanel = new JPanel();
38.       addButton(buttonPanel, "Start",
39.          new ActionListener()
40.             {
41.                public void actionPerformed(ActionEvent evt)
42.                {
43.                   addBall();
44.                }
45.             });
46.
47.       addButton(buttonPanel, "Close",
48.          new ActionListener()
49.             {
50.                public void actionPerformed(ActionEvent evt)
51.                {
52.                   System.exit(0);
53.                }
54.             });
55.       contentPane.add(buttonPanel, BorderLayout.SOUTH);
56.    }
57.
```

```
58.    /**
59.       Adds a button to a container.
60.       @param c the container
61.       @param title the button title
62.       @param listener the action listener for the button
63.    */
64.    public void addButton(Container c, String title,
65.       ActionListener listener)
66.    {
67.       JButton button = new JButton(title);
68.       c.add(button);
69.       button.addActionListener(listener);
70.    }
71.
72.    /**
73.       Adds a bouncing ball to the canvas and makes
74.       it bounce 1,000 times.
75.    */
76.    public void addBall()
77.    {
78.       try
79.       {
80.          Ball b = new Ball(canvas);
81.          canvas.add(b);
82.
83.          for (int i = 1; i <= 1000; i++)
84.          {
85.             b.move();
86.             Thread.sleep(5);
87.          }
88.       }
89.       catch (InterruptedException exception)
90.       {
91.       }
92.    }
93.
94.    private BallCanvas canvas;
95.    public static final int WIDTH = 450;
96.    public static final int HEIGHT = 350;
97. }
98.
99. /**
100.    The canvas that draws the balls.
101. */
102. class BallCanvas extends JPanel
103. {
104.    /**
105.       Add a ball to the canvas.
106.       @param b the ball to add
107.    */
108.    public void add(Ball b)
109.    {
110.       balls.add(b);
111.    }
112.
113.    public void paintComponent(Graphics g)
114.    {
115.       super.paintComponent(g);
116.       Graphics2D g2 = (Graphics2D)g;
```

```
117.       for (int i = 0; i < balls.size(); i++)
118.       {
119.          Ball b = (Ball)balls.get(i);
120.          b.draw(g2);
121.       }
122.    }
123.
124.    private ArrayList balls = new ArrayList();
125. }
126.
127. /**
128.    A ball that moves and bounces off the edges of a
129.    component
130. */
131. class Ball
132. {
133.    /**
134.       Constructs a ball in the upper left corner
135.       @c the component in which the ball bounces
136.    */
137.    public Ball(Component c) { canvas = c; }
138.
139.    /**
140.       Draws the ball at its current position
141.       @param g2 the graphics context
142.    */
143.    public void draw(Graphics2D g2)
144.    {
145.       g2.fill(new Ellipse2D.Double(x, y, XSIZE, YSIZE));
146.    }
147.
148.    /**
149.       Moves the ball to the next position, reversing direction
150.       if it hits one of the edges
151.    */
152.    public void move()
153.    {
154.       x += dx;
155.       y += dy;
156.       if (x < 0)
157.       {
158.          x = 0;
159.          dx = -dx;
160.       }
161.       if (x + XSIZE >= canvas.getWidth())
162.       {
163.          x = canvas.getWidth() - XSIZE;
164.          dx = -dx;
165.       }
166.       if (y < 0)
167.       {
168.          y = 0;
169.          dy = -dy;
170.       }
171.       if (y + YSIZE >= canvas.getHeight())
172.       {
173.          y = canvas.getHeight() - YSIZE;
174.          dy = -dy;
175.       }
```

```
176.
177.        canvas.paint(canvas.getGraphics());
178.     }
179.
180.     private Component canvas;
181.     private static final int XSIZE = 15;
182.     private static final int YSIZE = 15;
183.     private int x = 0;
184.     private int y = 0;
185.     private int dx = 2;
186.     private int dy = 2;
187. }
```

API java.lang.Thread

- static void sleep (long millis)

睡眠经过了规定的毫秒数

参数：millis　　睡眠经过的毫秒数

在上一节中，我们已经了解了如何才能将一个程序分割成多个同时执行的任务。每个任务都需要被放到一个扩展Thread类的run方法中。但是，如果我们想要将run方法添加给已经扩展了另一个类的某个类，那将会出现什么样的情况呢？当我们想要将多线程添加给一个applet时，这种情况最易发生。applet类已经继承了JApplet的特性，并且我们不能继承两个父类的特性，因此我们必须使用一个接口。必要的接口内置在Java平台中。它称为Runnable。我们将在下面介绍这个重要的接口。

1.1.1 使用线程为其他任务提供机会

如果使弹跳球代码运行在不同的线程中，则能使弹跳球程序具有更好的响应性。

注意 由于大多数计算机没有配备多个处理器，因此Java虚拟机（JVM）使用了一种机制，使每个线程都能够得到一个机会，以便运行一会儿之后，可激活另一个线程。该虚拟机通常依靠主机操作系统来提供线程调度程序包。

我们的下一个程序要使用两个线程，一个用于使球弹跳，另一个用于事件调度，它负责处理用户接口事件。由于每个线程都能得到一个运行的机会，因此主线程有机会注意到当球在弹跳时你是何时点击“Close”按钮的。然后该主线程便负责处理该“关闭”操作。

有一个简单的过程可以用于运行在某个独立的线程中的代码，那就是将该代码放入到从Thread派生而来的这个类的run方法中。

为了将我们的弹跳球程序放在一个独立的线程中，我们只需要从Thread中派生一个类BallThread，并且将用于产生动画的代码放入run方法中，就像下面的代码那样：

```
class BallThread extends Thread
{
   . . .
```

试读结束：需要全本请在线购买： www.ertongbook.com