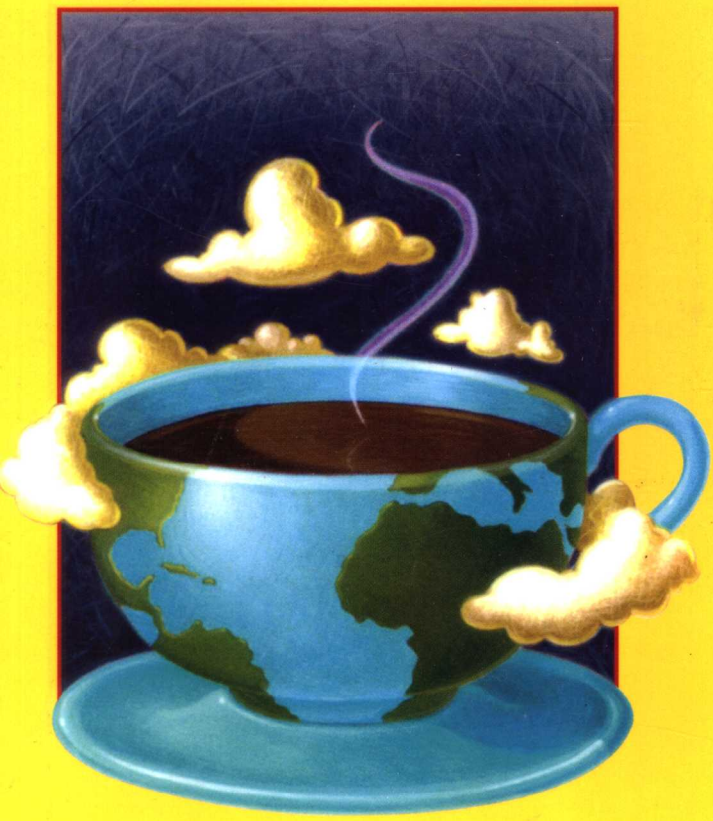
HZ BOOKS
PRENTICE HALL PTR

Sun公司核心技术丛书

JAVA 2

核心技术 卷Ⅱ：高级特性（原书第7版）

Core Java 2, Volume Ⅱ -Advanced Features Seventh Edition



（美） Cay S. Horstmann
Gary Cornell 著

陈昊鹏 王浩 姚建平 等译

- 所有代码示例都针对J2SE 5.0做了全面更新。
- 增加了有关注释和元数据的全新的一章。
- 订正和更新了多线程、集合、数据库编程、分布式计算和XML所涉及的内容。

Sun microsystems

机械工业出版社
China Machine Press

J2SE™ 5.0

Sun公司核心技术丛书

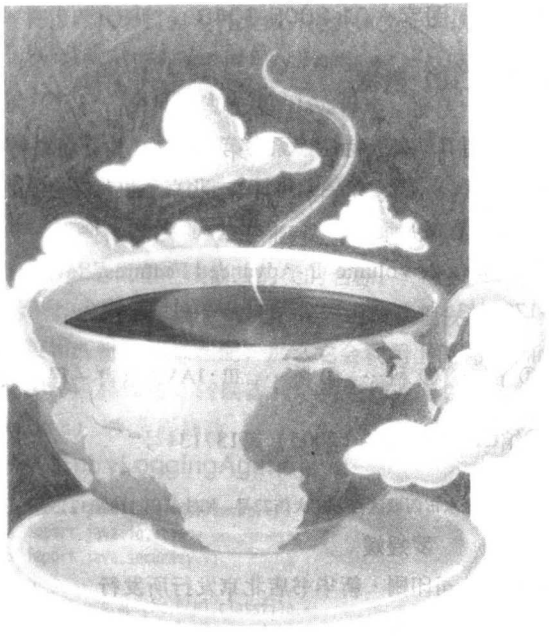
JAVA 2

核心技术 卷Ⅱ：高级特性（原书第7版）

Core Java 2, Volume Ⅱ -Advanced Features Seventh Edition

（美） Cay S. Horstmann
Gary Cornell 著

陈昊鹏 王浩 姚建平 等译



机械工业出版社
China Machine Press

J2SE™ 5.0

本书是Java 2技术权威指南，全面覆盖Java 2技术的高级主题，包括：多线程、集合框架、网络API、数据库编程、分布式对象等，深入探究了Swing、Java 2D API、JavaBean、Java安全模式、XML、注释、元数据等主题，同时涉及本地方法、国际化以及JDK 5.0的内容。本书适合软件开发人员、高等院校学生和教师参考。

本书版权登记号：图字：01-2005-3048

图书在版编目（CIP）数据

Java 2核心技术，卷II：高级特性（原书第7版）/（美）霍斯特曼（Horstmann, C. S.），（美）科奈尔（Cornell, G.）著；陈昊鹏等译. －北京：机械工业出版社，2006.3

（Sun公司核心技术丛书）

书名原文：Core Java 2, Volume II-Advanced Features, Seventh Edition

ISBN 7-111-17901-3

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

中国版本图书馆CIP数据核字（2005）第137134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：隋　曦　　罗媛媛

北京中兴印刷有限公司印刷·新华书店北京发行所发行

2006年3月第1版第1次印刷

787mm × 1092mm 1/16 · 53.5印张

定价：108.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

本社购书热线：（010）68326294

译 者 序

“Core Java” 已经连续推出了7版，在广大Java程序员和爱好者中的影响力越来越大。本书覆盖面广，几乎囊括了Java 2标准版的所有方面。以接近实战的实例来展开内容的书写方式，更是容易让读者理解和接受Java的精髓。

Java已经受到越来越多的程序员的青睐，越来越多的程序员开始从C++转战到Java的领域。但是Java语言的内容包罗万象，而且其自身发展的速度更是惊人，我们在Sun公司的网站上几乎每个月都会看到有新的基于Java的规范出台。JDK 5.0的出现使很多以前对JDK 1.4已经非常熟悉的Java程序员也产生了一些震惊，其新添加的特性将在很大程度上改变以前使用Java的方式。因此，本书在第6版的基础上，对JDK 5.0中的新特性进行了重点介绍，对原有部分章节进行了更新和调整，并且新增加了一些章节，以此来使程序员们能够更加透彻地理解和熟练地掌握这些新特性。

本卷面向的是已经熟读并掌握了本书卷I内容的读者，或者是已经对Java语言的基本特性相当熟悉的读者。本卷包含了多线程、集合、数据库编程、分布式对象、AWT高级特性、Swing高级特性、JavaBean、安全、国际化、本地方法、XML以及注释等内容，把读者引入了Java世界的更深处。

我们在翻译本书的过程中力求忠于原著。对于本书中出现的大量的专业术语尽量遵循标准的译法，并在有可能引起歧义之处注上了英文原文，以便读者对照理解。

全书的翻译由陈昊鹏、王浩、姚建平和龚斌合作完成，楼钢、李伟、郭嘉和方小丽也参与了全书的翻译和审校工作。由于我们水平有限，书中出现错误与不妥之处在所难免，恳请读者批评指正。

前　言

致读者

您手中的这本书是第7版的《Java 2 核心技术》（*Core Java 2*）的卷Ⅱ。卷Ⅰ主要介绍了Java语言的一些关键特性；而本卷主要介绍编程人员进行专业的软件开发时需要了解的高级主题。因此，与本书卷Ⅰ和它前面的一些版本一样，我们仍将本书定位于为那些将Java技术运用于实际项目的编程人员提供帮助。

请注意：如果你是一个经验丰富的开发人员，能够灵活运用像内部类和泛型这样的高级语言特性，那么你就不需要阅读完卷Ⅰ再学习本卷。（不过，本卷会根据适当情况去参考引用卷Ⅰ的有关内容，我们希望读者会购买或者已经购买了卷Ⅰ，当然，读者也可以在任何一本综合介绍Java平台的书中找到所需的背景知识。）

最后要说明的一点是，编写任何一本书籍都难免会有一些错误或不准确的地方。我们非常乐意听到这方面的内容。当然，我们更希望对这些问题的报告只听到一次。为此，我们创建了一个FAQ、bug修正以及应急方案的网站http://www.horstmann.com/corejava.html。你可以在bug报告网页（该网页的目的是鼓励读者阅读以前的报告）的末尾处添加bug报告来发布bug和问题、给出建议，以便改进本书将来的版本。

关于本书

本书中的章节大部分是相互独立的。你可以研究令你最感兴趣的任何主题，并可以按照任意顺序阅读这些章节。

第1章着重介绍多线程，它可以让你编程实现并行运行的多个任务。（一个线程就是程序中的一个控制流。）我们将介绍怎样创建线程，以及怎样对线程进行同步处理。多线程在JDK 5.0中变动很大，我们会告诉你有关的所有新机制。

第2章的主题是Java 2平台的集合框架。无论何时，只要你想要收集多个对象，并且以后还要对它们进行检索，你都会希望使用一个最适合你的运行环境的集合，而不是仅仅将这些元素随意置入一个向量中。

这一章将介绍如何利用预先为你构建的标准集合，同时对JDK 5.0泛型集合类做了彻底的修正。

第3章介绍网络API。Java使得复杂的网络编程工作变得很容易实现。我们将介绍怎样创建连接到服务器上的网络连接，怎样实现你自己的服务器，以及怎样生成HTTP连接。最后我们将讨论半关闭信道以及可中断信道这样的高级问题。

第4章介绍数据库编程。重点讲解JDBC，即Java数据库连接API，这是用于将Java程序与关系数据库进行连接的API。我们将介绍怎样通过使用JDBC API的核心子集，编写能够处理实际的数据库日常操作事务的实用程序。（如果要完整介绍JDBC API的功能，可能需要编写一本像本书一样厚的书才行。）最后我们简要介绍了层次数据库，探讨了一下JNDI（Java命名及目录接口）以及LDAP（轻量级目录访问协议）。

第5章介绍分布式对象。我们详细了介绍RMI（远程方法调用）。这个API可以让你运行分布在多台机器上的Java对象。然后简要讨论了CORBA（通用对象请求代理架构），并展示了那些用C++和Java编写的对象是怎样进行通信的。最后讨论了SOAP（简单对象访问协议），并给出了一个实现了Java程序和Amazon Web Service之间进行通信的示例。

第6章涵盖了没有纳入卷Ⅰ的所有Swing知识，尤其是重要但很复杂的树构件和表格构件。随后我们介绍了编辑面板的基本用法、“多文档”界面的Java实现以及在多线程程序中用到的进度指示器。我们仍着重介绍在实际编程中可能遇到的最为有用的构件，因为对Swing类库进行百科全书般的介绍可能会填满好几

卷书，并且只有专业编程人员才感兴趣。

第7章介绍Java 2D API，你可以用它来创建实际的图形。该章还介绍了抽象窗口操作工具包（AWT）的一些高级特性，这部分似乎应该在卷 I 中做专门介绍。虽然如此，这些技术还是应该成为每一个编程人员工具包的一部分。这些特性包括打印和用于剪切及拖放的API。

第8章介绍了用于Java平台的构件API——JavaBean。你将会看到怎样编写自己的bean，以及其他编程人员怎样在集成构建环境中对它们进行操作。最后我们展示怎样使用JavaBean的持久性，以某种与适用于长期存储的对象序列化不同的格式来存储自己的数据。

第9章继续介绍Java安全模式。Java平台一开始就是基于安全而设计的，该章会带你深入内部，查看这种设计是怎样实现的。我们将展示怎样编写用于特殊目的的应用的类加载器以及安全管理器。然后介绍允许使用消息、代码签名、授权以及认证和加密等重要特性的安全API。从JDK 5.0开始，这部分内容被彻底更新了，以便能够利用AES和RSA加密算法。

第10章讨论了一个我们认为重要性将会不断提升的特性——国际化。Java编程语言是几种一开始就被设计为可以处理Unicode的语言之一，不过Java平台的国际化支持则走得更加深远。因此，你可以对Java应用程序进行国际化，使得它们不仅可以跨平台，而且还可以跨越国界。例如，我们会展示怎样编写一个退休金计算器的applet，对它可以根据本地浏览器的情况使用英语、德语或者汉语进行浏览。

第11章介绍本地方法，它可以让你调用为微软Windows API这样的特殊机制而编写的各种调用方法。很显然，这种特性具有争议性：使用本地方法，那么Java平台的跨平台本质将会随之消失。虽然如此，每个为特定平台编写Java应用程序的严谨的编程人员都需要了解这些技术。当你编写重要的应用程序的时候，为了你的目标平台，你可能需要求助于操作系统API。我们将通过给出一个怎样从某个Java程序访问Windows注册表API的示例阐明这一点。

第12章介绍XML。介绍怎样解析XML文件，怎样生成XML以及怎样使用XSL转换。在一个实用示例中，我们将展示怎样在XML中指定Swing格式的布局。我们对该章进行了更新修正，将XPath API纳入其中，它使得“在XML的干草堆中发现绣花针”变得更加容易。

第13章是本版新增加的一部分。涉及注释、元数据以及在JDK 5.0中新添加的一些特性。可以使用注释向Java程序中添加任意信息（元数据）。我们将展示注释处理器怎样在源码级别或者在类文件级别上收集这些注释，以及怎样运用这些注释来影响运行时的类行为。注释只有在工具的支持下才有用，因此，我们希望我们的讨论能够帮助你根据需要选择有用的注释处理工具。

约定

我们使用等宽字体表示计算机代码，这种格式在众多的计算机书籍中极为常见。各种图标的含义如下：

注意：需要引起注意的地方。

提示：有用的提示。

警告：关于缺陷或危险情况的警告信息。

C++注意：本书中有一定的C++注释，用于解释Java程序设计语言和C++语言之间的不同。如果你对这部分不感兴趣，可以跳过。

应用编程接口

Java平台配备有大量的编程类库或者应用编程接口（API）。当第一次使用某个API时，我们添加了一

个简短的描述，并用一个API图标进行标识。这些描述可能有点不太规范，但是比起那些正式的在线API文档来说要更具指导性。

其源代码包含在与本书相伴的代码中的程序都被作为示例程序而将其代码列举了出来；例如

例5-12 WarehouseServer.java

可以从网站http://www.phptr.com/corejava[⊖]下载相关代码。

致谢

写书总是需要付出极大的努力，而重写也并不像看上去那么容易，特别是在Java技术方面，要跟上其飞快的发展速率，更是如此。一本书的面世需要众多有奉献精神的人共同努力，我非常荣幸地在此向整个《Java 2核心技术》团队致谢。

长期为我们辛勤劳动的Prentice Hall出版社的编辑Greg Doench再次出色地完成了工作，协调处理了这项复杂工程的方方面面。Mary Lou Nohr编辑誊写了手稿，对文字的一致性投入了极大的关注，并且总是能够发现我那日耳曼风格的句式，以及对Java注册商标规则的违反。Vanessa Moore再次出色地完成了本书的制作工作。Prentice Hall出版社和Sun Microsystems出版社的其他许多人也都提供了颇有价值的帮助，但是他们甘愿居于幕后。我希望他们都能够知道我是多么地感谢他们付出的努力。我的感谢还要送给本书以前版本的合著者Gary Cornell，他后来转向其他具有挑战性的领域了。

我非常感谢找到了很多令人尴尬的错误并提出了许多颇具创见性的建议的评审团队。这一版是由以下人员评审的：Chuck Allison (特约编辑，《*C/C++ Users Journal*》)、Cliff Berg (iSavvix 公司)、Frank Cohen (PushToTest)、Brian Goetz (首席顾问，Quiotix公司)、Rob Gordon、John Gray (哈特福大学)、Dan Harkey (圣何塞州立大学)、William Higgins (IBM)、Angelika Langer、Mark Lawrence、Bob Lynch (Lynch Associates)、Philip Milne (顾问)、Hao Pham、Stephen Stelting (Sun Microsystems)、Kim Topley (《*Core JFC*》的作者)和 Paul Tyma (顾问)。对你们的慷慨帮助，我表示万分的感谢！

以前版本的评审者为：Alec Beaton (PointBase, Inc.)、Joshua Bloch (Sun Microsystems)、David Brown、Dr. Nicholas J. De Lillo (曼哈顿学院)、Rakesh Dhoopar (Oracle)、David Geary (Sabreware Inc.)、Angela Gordon (Sun Microsystems)、Dan Gordon (Sun Microsystems)、Rob Gordon、Cameron Gregory (olabs.com)、Marty Hall (约翰斯·霍普金斯大学应用物理实验室)、Vincent Hardy (Sun Microsystems)、Vladimir Ivanovic (PointBase, Inc.)、Jerry Jackson (ChannelPoint Software)、Tim Kimmet (Preview Systems)、Chris Laffra、Charlie Lai (Sun Microsystems)、Doug Langston、Doug Lea (SUNY Oswego)、Gregory Longshore、Mark Morrissey (俄勒冈研究院)、Mahesh Neelakanta (佛罗里达大西洋大学)、Paul Philion、Blake Ragsdell、Stuart Reges (亚利桑那大学)、Peter Sanders (ESSI 大学, Nice, France)、Devang Shah (Sun Microsystems)、Christopher Taylor、Luke Taylor (Valtech)、George Thiruvathukal、Janet Traub、Peter van der Linden (Sun Microsystems)和Burt Walsh。

Cay Horstmann

旧金山2004年9月

⊖ 也可登录华章网站（http://www.hzbook.com）下载相关代码。——编辑注

目　录

第1章 多 线 程

- 什么是线程?
- 中断线程
- 线程状态
- 线程属性
- 同步
- 阻塞队列
- 线程安全的集合
- Callable和Future
- 执行器
- 同步器
- 线程和Swing

读者可能对操作系统中的多任务已经很熟悉了：在同一时刻似乎有多个程序在同时运行的能力。例如，你在编辑或发送一份传真的同时把它打印出来。当然，除非你有一台多处理器的机器，否则操作系统是将CPU时间划分成小的片段，并将其分配给不同的程序，从而造成一种并行处理的错觉。这种资源分配方法之所以可行，是因为虽然你可能认为当前的操作，如输入数据，会使计算机忙于处理，但实际上在这一过程中，CPU的大部分时间是空闲的。

有两种实现多任务的方法，这取决于操作系统在中断程序时的行为——直接中断而不需要事先和被中断程序协商，还是只有在被中断程序同意交出控制权之后才能执行中断。前者称为*抢占式多任务*；后者称为*协作*（即非抢占式）*多任务*。早期的操作系统，像Windows 3.x和Mac OS 9是协作多任务系统，一些简易设备（例如手机）上的操作系统也采用这种方式。UNIX/Linux、Windows NT/XP（以及针对32位程序的Windows 9x）和OS X则是抢占式的。抢占式多任务更加有效，但实现起来难度较大。而在协作多任务机制下，一个行为不当的程序可能会独占所有资源，导致其他所有程序无法正常工作。

多线程程序在更低的层次中引入多任务从而扩展了多任务的思想：单个程序看起来可以同时处理多个任务。通常将每个任务称为一个*线程*，它是控制线程的简称。可以一次运行多个线程的程序被称为是*多线程的*。

那么，*多线程*和*多进程*有什么区别呢？本质的区别在于每个进程有它自己的变量的完备集，线程则共享相同的数据。这听起来似乎有些危险，事实上也的确如此，你将会在本章后面的内容中看到这个问题。尽管如此，对程序来说，共享的变量使线程之间的通信比进程间的通信更加有效而简单。而且，对于某些操作系统而言，线程比进程更“轻量级”，创建和销毁单个线程比发起进程的开销要小得多。

在实际使用中，多线程非常有用。例如，一个浏览器必须能够同时下载多幅图片；一个Web服务器需要能够处理并发的请求；Java程序设计语言自身就使用了一个线程在后台进行垃圾回收，使你不用为内存管理而操心！图形用户界面（GUI）程序用一个单独的线程从主机操作环境中收集用户界面事件。本章将为你展示如何为你的Java应用程序添加多线程功能。

在JDK 5.0中，多线程发生了重大的变化，增加了大量的类和接口，它们为大部分程序员都需要的多线程机制提供了高质量的实现。在本章中，我们会解释JDK 5.0的新特性以及一些经典的同步机制，并告诉教你如何在这些机制中进行选择。

注意：多线程可能会变得非常复杂。在本章中，我们覆盖了程序员可能会需要的所有工具。尽管如此，对于更复杂的系统级编程，我们建议参考更高级的参考资料，例如《*Concurrent Programming in Java*》，该书由Doug Lea撰写，由Addison-Wesley在1999年出版。

1.1 什么是线程

让我们先来看一个不使用多线程的程序，用户很难让它去执行多个任务。对其进行剖析之后，我们将向你展示让这个程序运行彼此独立的多个线程是多么地简单。这个程序动画显示了一个跳动的球，方法是不断地移动它，确定它是否从墙壁弹回，然后刷新它，见图1-1。

一旦按下Start按钮，程序将从屏幕的左上角弹射出一个球，这个球便开始弹跳。Start按钮的处理程序调用addBall方法。这个方法包含一个运行1 000次move的循环。每次调用move时，球会被移动一点点，如果它从墙上被反弹回来就要调整移动的方向，然后刷新面板。

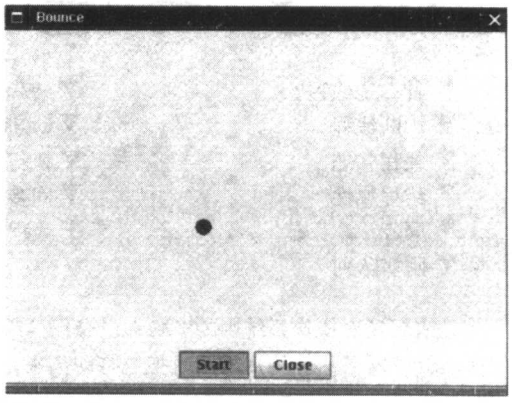


图1-1 使用线程演示跳动的球

```
Ball ball = new Ball();
panel.add(ball);

for (int i = 1; i <= STEPS; i++)
{
   ball.move(panel.getBounds());
   panel.paint(panel.getGraphics());
   Thread.sleep(DELAY);
}
```


Thread类的静态的sleep方法会使线程暂停指定的毫秒数。

调用Thread.sleep不会创建新的线程，sleep只是Thread类的一个静态方法，用于暂时停止当前线程的活动。

sleep方法可以抛出一个InterruptedException。我们会在稍后讨论这个异常和处理它的方法。而现在，如果这个异常发生了，我们只终止弹跳。

如果你运行这个程序，球能够很好地到处跳跃，但是它接管了整个应用。如果你在它结束1 000次移动前感到厌倦了，并点击关闭按钮，就会发现球仍旧在继续跳跃。在球自己结束跳跃之前你无法与程序交互。

注意：如果你仔细地看过了本节末尾的代码，你将会注意到在Ball类的move方法中有以下这样一个调用。

```
panel.paint(panel.getGraphics( ))
```

这一点很奇怪，一般来讲，你应该调用repaint，让AWT来获取图形上下文并负责绘制。但是如果在该程序中你试图调用panel.repaint ( )，你会发现面板永远不会被刷新。因为addBall方法已经完全接管了所有的处理。在后面一个使用独立线程来计算球位置的程序中，我们会重新使用大家所熟悉的repaint。

显然，这个程序的性能相当糟糕。你肯定不希望你所使用的程序在处理一件费时的工作时会以这种方式来运行。毕竟，当你通过网络连接读取数据时，其他任务被阻塞是经常发生的，有时候你的确想要中断读取操作。例如，假设你在下载一副很大的图片，当你看了一部分后，你决定不需要或不想看余下的部分了，你当然希望能够点击停止或后退按钮来中断加载过程。在下一节中，我们会演示如何通过在一个独立的线程中运行代码的关键部分，来保持用户对程序的控制。

例1-1展示了该程序的代码。

例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.setVisible(true);
17.    }
18. }
19.
20. /**
21.    A ball that moves and bounces off the edges of a
22.    rectangle
23. */
24. class Ball
25. {
26.    /**
27.       Moves the ball to the next position, reversing direction
28.       if it hits one of the edges
29.    */
30.    public void move(Rectangle2D bounds)
31.    {
32.       x += dx;
33.       y += dy;
34.       if (x < bounds.getMinX())
35.       {
36.          x = bounds.getMinX();
37.          dx = -dx;
38.       }
39.       if (x + XSIZE >= bounds.getMaxX())
40.       {
41.          x = bounds.getMaxX() - XSIZE;
42.          dx = -dx;
43.       }
44.       if (y < bounds.getMinY())
45.       {
46.          y = bounds.getMinY();
47.          dy = -dy;
48.       }
49.       if (y + YSIZE >= bounds.getMaxY())
50.       {
51.          y = bounds.getMaxY() - YSIZE;
52.          dy = -dy;
53.       }
54.    }
55.
56.    /**
57.       Gets the shape of the ball at its current position.
58.    */
59.    public Ellipse2D getShape()
60.    {
61.       return new Ellipse2D.Double(x, y, XSIZE, YSIZE);
62.    }
63.
64.    private static final int XSIZE = 15;
```

```
65.    private static final int YSIZE = 15;
66.    private double x = 0;
67.    private double y = 0;
68.    private double dx = 1;
69.    private double dy = 1;
70. }
71.
72. /**
73.    The panel that draws the balls.
74. */
75. class BallPanel extends JPanel
76. {
77.    /**
78.       Add a ball to the panel.
79.       @param b the ball to add
80.    */
81.    public void add(Ball b)
82.    {
83.       balls.add(b);
84.    }
85.
86.    public void paintComponent(Graphics g)
87.    {
88.       super.paintComponent(g);
89.       Graphics2D g2 = (Graphics2D) g;
90.       for (Ball b : balls)
91.       {
92.          g2.fill(b.getShape());
93.       }
94.    }
95.
96.    private ArrayList<Ball> balls = new ArrayList<Ball>();
97. }
98.
99. /**
100.   The frame with panel and buttons.
101. */
102. class BounceFrame extends JFrame
103. {
104.    /**
105.       Constructs the frame with the panel for showing the
106.       bouncing ball and Start and Close buttons
107.    */
108.    public BounceFrame()
109.    {
110.       setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);
111.       setTitle("Bounce");
112.
113.       panel = new BallPanel();
114.       add(panel, BorderLayout.CENTER);
115.       JPanel buttonPanel = new JPanel();
116.       addButton(buttonPanel, "Start",
117.          new ActionListener()
118.          {
119.             public void actionPerformed(ActionEvent event)
120.             {
121.                addBall();
122.             }
123.          });
124.
```

```
125.       addButton(buttonPanel, "Close",
126.          new ActionListener()
127.          {
128.             public void actionPerformed(ActionEvent event)
129.             {
130.                System.exit(0);
131.             }
132.          });
133.       add(buttonPanel, BorderLayout.SOUTH);
134.    }
135.
136.    /**
137.       Adds a button to a container.
138.       @param c the container
139.       @param title the button title
140.       @param listener the action listener for the button
141.    */
142.    public void addButton(Container c, String title, ActionListener listener)
143.    {
144.       JButton button = new JButton(title);
145.       c.add(button);
146.       button.addActionListener(listener);
147.    }
148.
149.    /**
150.       Adds a bouncing ball to the panel and makes
151.       it bounce 1,000 times.
152.    */
153.    public void addBall()
154.    {
155.       try
156.       {
157.          Ball ball = new Ball();
158.          panel.add(ball);
159.
160.          for (int i = 1; i <= STEPS; i++)
161.          {
162.             ball.move(panel.getBounds());
163.             panel.paint(panel.getGraphics());
164.             Thread.sleep(DELAY);
165.          }
166.       }
167.       catch (InterruptedException e)
168.       {
169.       }
170.    }
171.
172.    private BallPanel panel;
173.    public static final int DEFAULT_WIDTH = 450;
174.    public static final int DEFAULT_HEIGHT = 350;
175.    public static final int STEPS = 1000;
176.    public static final int DELAY = 3;
177. }
```

java.lang.Thread 1.0

- static void sleep(long millis)

休眠指定的毫秒数。

参数: millis　　**休眠的毫秒数**