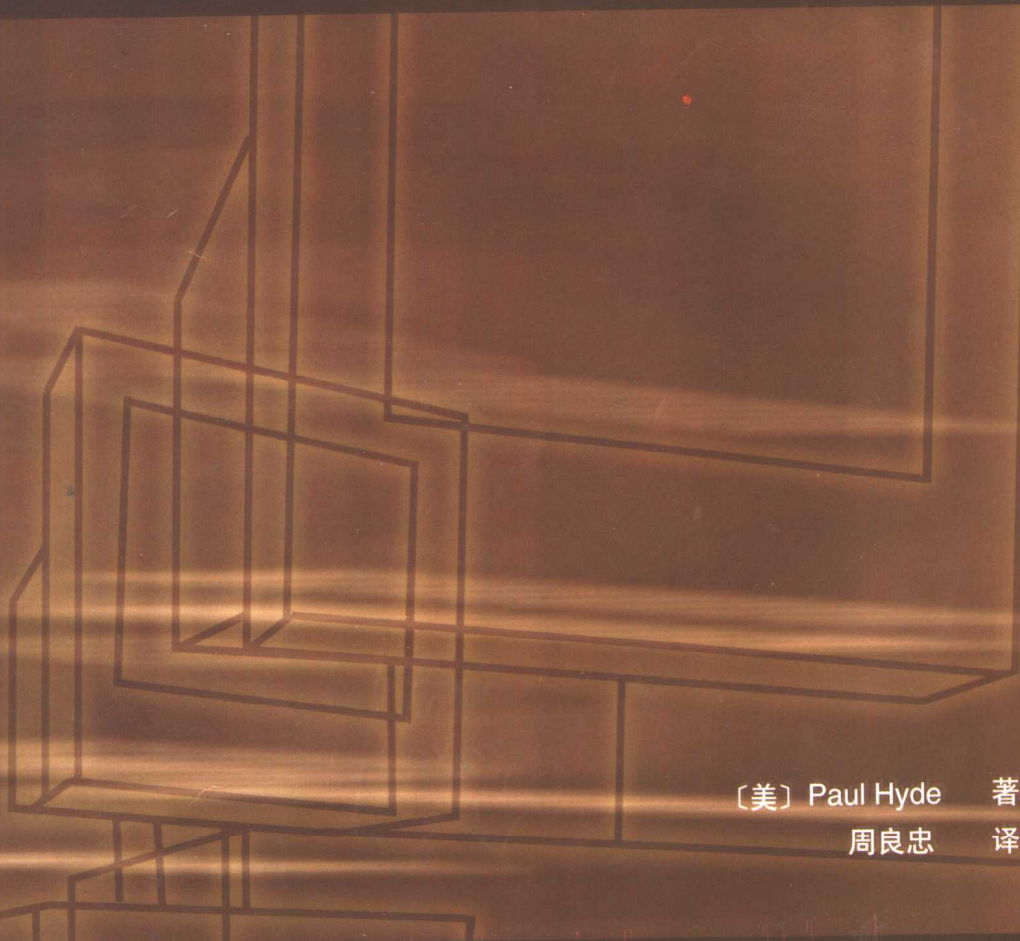


Java 线程编程

Java Thread Programming



〔美〕 Paul Hyde 著
周良忠 译



人民邮电出版社
POSTS & TELECOM PRESS

Java 线程编程

[美] Paul Hyde 著

周良忠 译

人民邮电出版社

图书在版编目 (CIP) 数据

Java 线程编程 / (美) 海德 (Hyde, p.) 著; 周良忠译. —北京: 人民邮电出版社, 2003.11
ISBN 7-115-11791-8

I. J... II. ①海...②周... III. JAVA 语言—程序设计 IV. TP312

中国版本图书馆 CIP 数据核字 (2003) 第 094544 号

版 权 声 明

Paul Hyde : Java Thread Programming

Copyright © 1999 by Sams Publishing

Authorized translation from the English language edition published by the Sams Publishing.

All rights reserved.

本书中文简体字版由美国 Sams 出版公司授权人民邮电出版社出版。未经出版者书面许可, 对本书任何部分不得以任何方式复制或抄袭。

版权所有, 侵权必究。

Java 线程编程

- ◆ 著 [美] Paul Hyde
- 译 周良忠
- 责任编辑 陈冀康

- ◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街 14 号
- 邮编 100061 电子函件 315@ptpress.com.cn
- 网址 <http://www.ptpress.com.cn>
- 读者热线 010-67132705
- 北京汉魂图文设计有限公司制作
- 北京顺义振华印刷厂印刷
- 新华书店总店北京发行所经销

- ◆ 开本: 787×1092 1/16
- 印张: 29.5
- 字数: 716 千字 2003 年 11 月第 1 版
- 印数: 1-4 000 册 2003 年 11 月北京第 1 次印刷

著作权合同登记 图字: 01 - 2003 - 5508 号

ISBN 7-115-11791-8/TP • 3712

定价: 52.00 元

本书如有印装质量问题, 请与本社联系 电话: (010) 67129223

内 容 提 要

本书以丰富的实例、由浅至深的方式全面讲解如何用 Java 进行多线程编程。全书分三部分。第一部分（第 1 章~第 10 章）介绍基础概念，包括线程、线程组、Swing、线程同步、线程优先级、启动和终止线程、线程间通信等。第二部分（第 11 章~第 18 章）介绍一些重要的多线程编程技术，包括自运行对象、异常回调、线程池、完全超时、摆脱阻塞的 I/O 状态、先进先出（FIFO）队列以及运用类 `SureStop` 和 `BooleanLock` 等。第三部分为附录，提供了 Thread API 和 ThreadGroup API 的相关材料。

本书可以帮助具有 Java 基本编程知识的程序员学习多线程编程。

译者的话

随着 Java2 一系列新技术（如 Java2D、Java3D、Swing、Java Sound、EJB、Servlet、JSP、CORBA、XML、JNDI 等）的引入，以及 VM 安全策略的不断完善和效率不断提高，Java 阵营越来越庞大，Java 在电子商务以及金融、证券、电信等各个行业中的应用越来越广泛。

几乎每一位 Java 程序员都知道 Java 语言具有很多出色的特点：简单、面向对象、自动内存管理、分布式计算、稳定、安全、解释执行、结构中立、平滑移植以及异常处理等，让诸多开发人员尤其推崇的是它对多线程的天生支持。

本书正是一本讲解 Java 多线程编程的专著。本书内容几乎涉及了 Java 多线程编程的各个方面。第一部分着重介绍基本概念，第二部分详细地介绍了一些线程技术。第三部分附录提供了 Thread API 和 ThreadGroup API 的相关资料。

读者要注意，本书示例使用的开发包为 Java SDK 标准版 1.2。现在的主流版本是 1.4。所有例子在 1.4 版本下仍能正常运行。针对多线程编程，版本从 1.2 到 1.4 没有实质性的变化，主要是修正一些 Bug，提升线程的性能。在不同版本间开发、调试 Java 多线程应用程序（甚至要综合考虑未来版本的新特点）有许多地方仍然值得读者关注，如：

- 版本较低的 JDK 不支持非阻塞 I/O API。从 JDK 1.4 开始引入了非阻塞的 I/O 库（java.nio）。
- 从 1.2 版开始，JDK 中就加入了 java.lang.ThreadLocal 类。
- 从下一个版本 JDK1.5 开始，又要提供关于线程、并发等新性能的支持。读者可以从 Sun Microsystems 公司网站了解相关信息。
- Doug Lea 编写了一个极其优秀的免费并发实用程序包，它包括并发应用程序的锁、互斥、队列、线程池、轻量级任务、有效的并发集合、原子的算术操作和其他基本组件。该包将成为下一个版本 JDK 1.5 中 java.util.concurrent 包的基础。
- 为了提升 JDK 1.2 中 ThreadLocal 的性能，JDK 1.3 对它进行了重写，JDK 1.4 又重写了一次。

Java 对多线程的支持对程序员来说无疑是幸运的，但较难掌握、容易出错，使很多人终日苦恼。相信这本书能成为采用 Java 多线程编程的程序员们的良师益友。

由于译者水平有限，且时间仓促，错误在所难免，希望广大读者不吝指正。我的联系 E-mail: web_zhou@21cn.com。

译者

2003 年 8 月 16 日

前言

本书结构

本书面向的读者是那些已经开始使用 Java 语言，而且需要开发多线程应用程序和 applet（小应用程序）的人。读者可以没有任何线程编程方面的背景，因此本书开始使用的示例简单易懂，随着讲解的深入，以后章节逐渐切入高级主题，并且完全涵盖 Java 线程编程的所有方面。本书的第二部分重点演示不同的高级技巧，它们可以马上用于实战。第 1 章~第 10 章可以按顺序阅读，因为每一章的讲解均是以前一章介绍的概念为基础的。第 11 章~第 18 章讨论的技术可以按任意顺序跳跃式地学习。一些技术相当有用，演示其他技术时，也用到了这些重要的技术，因此，可以在碰到这些技术时，再研读它们。

本书中的例子用 Sun Microsystems 的 Java 2 SDK，标准版 1.2（即 JDK 1.2）开发而成。开发包（Kit）的推荐运行环境：CPU 最低为 Intel 奔腾 166MHz，操作系统为微软 Windows 95 或更新版本。为了起强调作用，书中代码清单中的一些语句用黑体字排版。源代码文件可以从 www.sampublishing.com 下载。进入网站后，通过搜索引擎可以找到本书相关资源，方法是：在搜索引擎的左边文本框中填入本书的 ISBN 号（0672315858），在右边下拉框中选择“ISBN”，再点击“search”按钮即可以找到本书的资源链接。

下面为各章概要。

第 1 章：线程简介

第 1 章简介多线程编程以及如何在 Java 中使用线程。解释为什么需要线程，如何利用线程提高应用程序的性能。

第 2 章：一个简单的双线程实例

在第 2 章，可以看到最基本的 Java 多线程应用实例。在这些应用程序中，有两个线程同时运行并打印各自的信息。这一章将演示创建一个新线程所需的步骤。

第 3 章：创建和启动线程

第 3 章开始探讨类 Thread 的 API。在这一章将学习如何获得当前执行线程的句柄，如何命名线程，如何检查线程是否仍然激活以及如何暂时将线程设置成休眠状态。

第 4 章：实现 Runnable 接口与扩展 Thread 类

在第 4 章，使用接口 Runnable 来演示让对象中某个线程运行的第二种方式。对于缺少多重继承的 Java 环境，这是一个特别重要的特征。

第 5 章：完美终止线程

第 5 章演示如何终止线程的运行。这一章将介绍一些完美和安全的办法，以替代 JDK 1.2 中一些被淘汰的方法。

第 6 章：线程优先化

第 6 章演示如何在 Java 虚拟机上给线程分配优先级以及优先级对线程规划的影响。

第 7 章：并发访问对象和变量

当多个线程同时运行时，务必确保线程相互间能够安全交互。第 7 章将说明防止可能导致数据崩溃的竞争条件的具体步骤。这一章演示了关键字 synchronized 和 volatile 的正确使用。还演示如何以一种线程安全方式在 Collections API 中使用类。

第 8 章：线程间通信

让多线程与共享数据安全交互时，需要运用一种途径，让一个线程通知另一个线程数据已经更改。第 8 章演示如何使用 wait()、notify()和 notifyAll()在应用程序的线程间发送通知。还讨论了 join()、管道和变量 ThreadLocal 的使用。

第 9 章：线程和 Swing

第 9 章讲解如何通过图形用户界面在应用程序中使用多线程。Swing 工具包不能确保原有多线程安全，这一章将演示安全使用 Swing 的必需步骤。

第 10 章：线程组

第 10 章探讨 ThreadGroup API 以及给组分配线程的方式。

第 11 章：自运行对象

从第 11 章开始介绍技术部分。这一章首先演示如何创建一个类，在构建过程中能够自动

启动一个内部线程。运用这一技术，用户不必知道是否对象内存在运行的线程。

第 12 章：异常回调

第 12 章演示如何查找已经抛出异常的另一个线程。

第 13 章：线程池

第 13 章讨论如何在执行短小代码运行块时，共享线程池中的线程。这一章演示如何编写一个简单的 Web 页服务器，让它使用共享线程池为客户的页面请求服务。

第 14 章：等待完全超时

学习第 8 章时可以发现，判断线程是否受到另一个线程的通知，或者超时等待被通知，这一任务并非总是那么容易。第 14 章将讲解一种技术，可以用于确保线程等待完全超时值。

第 15 章：摆脱阻塞 I/O 状态的束缚

在完成数据的写或读之前，Java 中大部分 I/O 运算都处于阻塞状态。不方便的是，阻塞的 I/O 方法不会对中断做出反应。第 15 章将介绍一些处理这类情况的技术。

第 16 章：类 SureStop 的运用

第 16 章演示运用类 SureStop 来确保线程的最终消亡。

第 17 章：类 BooleanLock 的运用

第 17 章演示一种把等待/通知机制封装到一个紧凑、多线程安全类中的技术，这个类可以在多线程应用程序的多个地方重用。

第 18 章：先进先出（FIFO）队列

第 18 章演示如何构建一个在多线程环境中安全使用的 FIFO 队列。尤其是，这一章要讲解如何创建一个 FIFO 队列来保存对象引用以及如何创建一个 FIFO 队列来保存字节。

附录：线程 API 与线程组 API

本书末尾提供了两部分附录。附录 A 解释类 Thread 的 API；附录 B 解释类 ThreadGroup 的 API。

本书约定

本书使用不同的排版方式来匹配代码与常规文字，也有助于读者注意一些重要的概念。用户输入的文本（包括程序代码）以及应当出现在屏幕上的文本用如下字体：

`It will look like this to mimic the way text looks on your screen.`

代码行前面的箭头（↪）表示代码行太长，因此分排为 2 行。实际编译代码时，应该将此行与前一行当成完整代码行。

注意

“注意”提供一些与当前讨论内容相关的重要信息。

技巧

“技巧”提出建议，或者讲解更容易完成的途径。

警告

“警告”警告可能出现的问题，并帮助你避免灾难性错误。

目 录

第一部分 线 程

第 1 章 线程简介	3
1.1 什么是线程	4
1.2 为什么使用多线程	4
1.2.1 与用户的更佳交互	4
1.2.2 同步动作的模拟	5
1.2.3 利用多处理器	5
1.2.4 等待缓慢 I/O 操作时完成其他任务	5
1.2.5 简化对象模型	6
1.3 不宜采用多线程的场合	7
1.4 Java 的内置线程支持	7
1.5 易于起步，难以掌握	7
第 2 章 一个简单的双线程实例	9
2.1 扩展 java.lang.Thread 类	10
2.2 覆盖 run()方法	11
2.3 创建新线程	12
2.4 综合运用	13
2.5 小结	14
第 3 章 创建和启动线程	15
3.1 使用 Thread.currentThread()	16
3.2 线程命名：getName()和 setName()	18
3.2.1 使用 getName()	18
3.2.2 使用 setName()	21
3.3 线程构造函数	22
3.4 激活线程：start()和 isAlive()	24
3.5 使用 Thread.sleep()	26
3.6 小结	28
第 4 章 实现 Runnable 接口与扩展 Thread 类	30

4.1 可视定时器图形组件	31
4.2 能扩展 Thread 和 JComponent 吗	37
4.3 接口 java.lang.Runnable	38
4.4 把 Runnable 对象传递给 Thread 的构造函数	38
4.5 修改 SecondCounter 来使用 Runnable	39
4.6 检查 SecondCounter 的准确性	45
4.7 提高 SecondCounter 的准确性	49
4.8 小结	54
第 5 章 完美终止线程	55
5.1 中断线程: interrupt()	56
5.1.1 中断休眠线程	58
5.1.2 待决中断	56
5.1.3 使用 isInterrupted()	59
5.1.4 使用 Thread.interrupted()	60
5.1.5 使用 InterruptedException	61
5.2 挂起和恢复线程执行	63
5.2.1 使用淘汰的方法 suspend()和 resume()	63
5.2.2 在不恰当的时候挂起	67
5.2.3 不使用淘汰方法实现挂起和恢复	70
5.3 终止线程	74
5.3.1 使用淘汰的方法 stop()	74
5.3.2 取代 stop()	76
5.4 stop()、suspend()和 resume()的最佳替代	78
5.5 守护线程	84
5.6 小结	86
第 6 章 线程优先化	87
6.1 系统线程优先级	88
6.2 线程优先级常量	89
6.2.1 Thread.MAX_PRIORITY	89
6.2.2 Thread.MIN_PRIORITY	89
6.2.3 Thread.NORM_PRIORITY	89
6.3 判断当前优先级: getPriority()	89
6.4 更改线程的优先级: setPriority()	91
6.5 线程状态	94
6.6 优先级和规划	96
6.7 自愿放弃处理器: Thread.yield()	96
6.8 线程规划情形	102

6.8.1 情形一：一个高优先级线程独占处理器	102
6.8.2 情形二：所有高优先级线程独占处理器	102
6.8.3 情形三：所有线程均获得一定的处理器时间	102
6.9 小结	103
第 7 章 并发访问对象和变量	104
7.1 易变成员变量修饰符	105
7.2 同步方法修饰符	110
7.2.1 两个线程同时位于一个对象的同一个方法中	110
7.2.2 同一时刻一个线程	112
7.2.3 两个线程，两个对象	115
7.2.4 避免对象的意外崩溃	117
7.2.5 对象处于不一致状态时，推迟对它的访问	121
7.3 同步语句块	126
7.3.1 减少持有锁的时间	126
7.3.2 锁定任意对象，而非仅仅锁定当前对象	127
7.3.3 把向量内容安全地复制到数组	128
7.4 静态同步方法	130
7.5 在同步语句中使用类级别锁	134
7.6 同步化和集合 API	136
7.6.1 封装集合，使之同步化	136
7.6.2 安全地把列表中的内容复制到数组	137
7.6.3 安全遍历集合元素	139
7.7 死锁	141
7.8 加速并发访问	145
7.9 小结	145
第 8 章 线程间通信	147
8.1 线程间通信的必要性	148
8.2 等待/通知机制	148
8.2.1 最小规模的等待/通知	148
8.2.2 典型等待/通知	150
8.2.3 运用同步方法的等待/通知	151
8.3 用于等待/通知的对象 API	153
8.3.1 notify()	153
8.3.2 notifyAll()	153
8.3.3 wait()	153
8.3.4 wait(long)	154
8.3.5 wait(long, int)	154

8.4	何时使用 notifyAll()代替 notify()	154
8.5	遗漏通知	155
8.5.1	MissedNotify	155
8.5.2	MissedNotifyFix	159
8.6	早期通知	163
8.6.1	EarlyNotify	164
8.6.2	EarlyNotifyFix	167
8.7	示例 CubbyHole	171
8.8	使用 join()等待线程的消亡	177
8.8.1	join()	177
8.8.2	join(long)	177
8.8.3	join(long, int)	177
8.8.4	JoinDemo	178
8.9	使用管道在线程间流动数据	181
8.9.1	PipedBytes	182
8.9.2	PipedCharacters	185
8.10	使用 ThreadLocal 和 InheritableThreadLocal	188
8.10.1	ThreadLocal API	189
8.10.2	ThreadID	189
8.10.3	InheritableThreadLocal API	192
8.10.4	InheritableThreadID	192
8.11	小结	198
第 9 章	线程和 Swing	199
9.1	为什么 Swing 工具包不是多线程安全	200
9.2	使用 SwingUtilities.invokeLater()	201
9.3	使用 SwingUtilities.invokeLater()	204
9.4	使用 SwingUtilities.isEventDispatchThread()	207
9.5	何时不需要 invokeAndWait()和 invokeLater()	207
9.6	在 GUI 设置中使用工作线程的必需性	208
9.7	使用工作线程减轻事件线程的负担	213
9.8	在自定义组件中滚动文本	220
9.9	动画显示一系列图像	225
9.10	在 JLabel 上显示流逝的时间	229
9.11	在容器内浮动组件	233
9.12	小结	238
第 10 章	线程组	239
10.1	什么是线程组	240

10.2	使用 getParent()	241
10.3	查找线程组的子组	241
10.4	使用 Thread 的 getThreadGroup()方法	242
10.5	查找线程组中的所有线程	242
10.6	理解线程组的安全性	242
10.7	使用 setMaxPriority()和 getMaxPriority()	243
10.8	使用 interrupt()	243
10.9	淘汰的方法: stop()、suspend()和 resume()	243
10.10	类 ThreadViewer	243
10.11	小结	252

第二部分 技 术

第 11 章	自运行对象	255
11.1	简单自运行类	256
11.2	使用内部类来隐藏 run()	259
11.3	要考虑的额外功能	261
11.4	小结	267
第 12 章	异常回调	268
12.1	ExceptionListener 接口	269
12.2	支持 ExceptionListener 的辅助方法	269
12.3	小结	276
第 13 章	线程池	277
13.1	线程池的好处	278
13.2	线程池的考虑与开销	278
13.3	泛型线程池: ThreadPool	279
13.4	专门工作线程池: HttpServer	288
13.4.1	类 HttpServer	289
13.4.2	类 HttpWorker	296
13.4.3	服务文件示例	305
13.4.4	用 3 个工作线程运行 HttpServer	307
13.4.5	用 10 个工作线程来运行 HttpServer	309
13.5	小结	310
第 14 章	等待完全超时	311
14.1	意外提前返回	312

14.2	判断是否应当再次调用 wait()	315
14.3	通用等待-直到模式	319
14.4	小结	325
第 15 章	摆脱阻塞 I/O 状态的束缚	326
15.1	read()方法忽略中断和终止请求	327
15.2	关闭流来摆脱阻塞状态	329
15.2.1	类 CalcServer 与摆脱阻塞的 accept()	330
15.2.2	类 CalcWorker 与摆脱阻塞的 read()	333
15.2.3	类 CalcClient	336
15.2.4	运行 CalcClient 的输出	337
15.3	被中断时抛出 InterruptedException	338
15.3.1	类 ThreadedInputStream	338
15.3.2	类 BufferedThreadedInputStream	347
15.4	针对可中断 I/O 使用 BufferedThreadedInputStream	349
15.5	小结	354
第 16 章	SureStop 的运用	356
16.1	使用 SureStop 的原则	357
16.2	SureStop 类	357
16.3	使用 SureStopVerbose 的分析	363
16.4	用 SureStopDemo 演示 SureStop 的工作方式	369
16.5	小结	374
第 17 章	类 BooleanLock 的运用	375
17.1	背景	376
17.2	类 BooleanLock	376
17.3	使用 BooleanLock 在线程间发送信号	380
17.4	避免阻塞于同步	382
17.4.1	SyncBlock	382
17.4.2	InterruptibleSyncBlock	385
17.5	使用 TransitionDetector 检测 Value 的短暂变化	388
17.6	小结	394
第 18 章	先进先出 (FIFO) 队列	395
18.1	FIFO 队列如何工作	396
18.2	用数组实现 FIFO	397
18.3	用 Java 的简单实现: SimpleObjectFIFO	399
18.4	对象引用的一个扩展 FIFO 队列: ObjectFIFO	405

18.5 字节的 FIFO 队列: ByteFIFO	419
18.6 小结.....	432

第三部分 附 录

附录 A Thread API	435
成员变量	436
Thread.MAX_PRIORITY	436
Thread.MIN_PRIORITY	436
Thread.NORM_PRIORITY	436
构造函数	437
Thread(ThreadGroup, Runnable, String)	437
Thread(ThreadGroup, Runnable)	437
Thread(ThreadGroup, String)	437
Thread(Runnable, String)	437
Thread(Runnable)	438
Thread(String)	438
Thread()	438
静态方法	438
Thread.activeCount()	438
Thread.currentThread()	438
Thread.dumpStack()	438
Thread.enumerate()	439
Thread.interrupted()	439
Thread.sleep(long)	439
Thread.sleep(long, int)	439
Thread.yield().....	439
实例方法	440
checkAccess()	440
destroy()	440
getContextClassLoader()	440
getName()	440
getPriority()	440
getThreadGroup()	440
interrupt()	441
isAlive()	441
isDaemon().....	441
isInterrupted()	441

join()	441
join(long)	441
join(long, int)	442
run()	442
setContextClassLoader(ClassLoader)	442
setDaemon(boolean)	442
setName(String)	442
setPriority(int)	443
start()	443
toString()	443
被淘汰的方法	443
countStackFrames()	443
resume()	443
stop()	444
stop(Throwable)	444
suspend()	444

附录 B ThreadGroup API 445

构造函数	446
ThreadGroup(ThreadGroup, String)	446
ThreadGroup(String)	446
实例方法	447
activeCount()	447
activeGroupCount()	447
checkAccess()	447
destroy()	447
enumerate(Thread[], boolean)	447
enumerate(Thread[])	448
enumerate(ThreadGroup[], boolean)	448
enumerate(ThreadGroup[])	448
getMaxPriority()	448
getName()	448
getParent()	448
interrupt()	449
isDaemon()	449
isDestroyed()	449
list()	449
parentOf(ThreadGroup)	449
setDaemon(boolean)	449