

Java开发必读书籍清单

hollischuang

Published
with GitBook



目錄

简介	0
Effective Java	1
Java编程思想	2
深入理解Java虚拟机	3

简介

The book is the ladder of the progress of mankind ——Gorky

互联网时代，生活节奏加快，工作和学习的压力很大，人们逐渐被眼前的纷繁芜杂迷乱了双眼，渐渐忘记了书存在的价值。互联网的出现的的确给我们获取信息带来了便利，但是互联网毕竟只是一种单纯的阅读工具，任何事物都无法完全替代书的作用以及书的价值，因为书承载了太多的历史文明，我们需要通过书本探索与思考的东西还有很多很多。

对于 Java 语言开发人员来说，其实真正的问题并不是我们没有书可以读，如今的我们随便在淘宝上搜索一下关于 Java 的书籍，就会有成千上万本书可以选择，信息过量是一个真正的问题。到底哪些书应该专心的研读，哪些书可以在需要的时候拿来翻阅一下，哪些书不值得花钱去购买成了一个很难抉择的问题。作为一个一线开发，我们需要有丰富的行业知识。我们需要时刻补充自己在专业领域的深度和广度，读书是一个最好的方式。

本文总结了 Java 开发人员应该读的书籍的列表。供大家参考，同时欢迎补充。

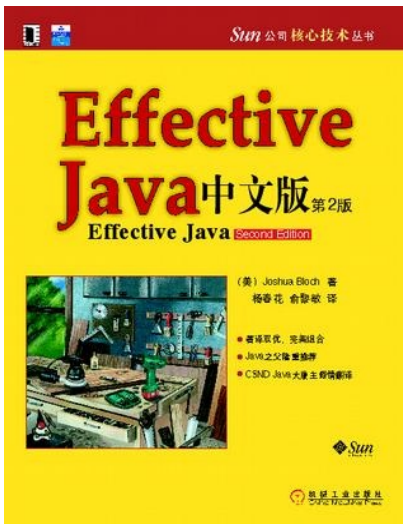
作者简介

Hollis

QQ : 907607222

Mail : hollischuang@qq.com

Site : <http://www.hollischuang.com>



原名: Effective Java Second Edition

中文版: Effective java 中文版 (第2版)

作者: Joshua Bloch

出版社: 机械工业出版社

译者: 俞黎敏

出版年: 2009-1-1

豆瓣评分: 9.1

推荐理由

该书 Joshua Bloch 是 Google 公司的首席 Java 架构师。是 Jolt 大奖的获得者。他曾是 Sun 公司的杰出工程师，和 Transarc 公司的高级系统设计师。

本书介绍了在 Java 编程中 78 条极具实用价值的经验规则，这些经验规则涵盖了大多数开发人员每天所面临的问题的解决方案。通过对 Java 平台设计专家所使用的技术的全面描述，揭示了应该做什么，不应该做什么才能产生清晰、健壮和高效的代码。

本书中的每条规则都以简短、独立的小文章形式出现，并通过例子代码加以进一步说明。本书内容全面，结构清晰，讲解详细。可作为技术人员的参考用书。

本书既适合 Java 学习的入门者阅读，也适合有一定编程经验的人。对于入门者，阅读本书可以养成良好的编程习惯。对于有经验者，也可以不断帮助自己提高代码质量。应该经常拿出来翻阅。

简介

“我很希望 10 年前就拥有这本书。可能有人认为我不需要任何 Java 方面的书籍，但是我需要这本书。”
——Java 之父 James Gosling

本书为我们带来了共78条程序员必备的经验法则，针对你每天都会遇到的编程问题提出了最有效、最实用的解决方案。作者对新版进行了彻底的更新。涵盖了自第一版之后所引入的Java SE5和Java SE 6的特性，同时开发了新的设计模式和语言习惯用法，介绍了如何利用从泛型到枚举、从注解到自动装箱的各种特性。

书中的每一章都包含几个“条目”，以简洁的形式呈现，自成独立的短文，它们提出了具体的建议，对于Java平台精妙之处的独到见解，以及优秀的代码范例。每个条目的综合描述和解释都阐明了应该怎么做，不应该怎么做，以及为什么。

本书的内容包括：

全新的泛型、枚举、注解、自动装箱、for-each循环、可变参数、并发机制，等等。

经典主题的全新技术和最佳实践，包括对象、类、类库、方法和序列化。

如何避免Java编程语言中常被误解的细微之处：陷阱和缺陷。

重点关注Java语言本身和*基本的类库及其扩展 `java.lang` `java.util`
`java.util.concurrent` 和 `java.io`。

作者简介

Joshua Bloch是Google公司的首席Java架构师。是Jolt大奖的获得者。他曾是Sun公司的杰出工程师，和Transarc公司的高级系统设计师。Bloch曾带领团队设计和实现过无数的Java平台特性，包括JDK 5.0语言增强版和获奖的Java Collections Framework。他的著作还包括：《Java Puzzlers》、《Java Concurrency in Practice》等。

目录

译者序 序

前言

致谢

第1章 引言

第2章 创建和销毁对象

第1条：考虑用静态工厂方法代替构造器

第2条：遇到多个构造器参数时要考虑用构建器

第3条：用私有构造器或者枚举类型强化Singleton属性

第4条：通过私有构造器强化不可实例化的能力

第5条：避免创建不必要的对象

第6条：消除过期的对象引用

第7条：避免使用终结函数

第3章 对于所有对象都通用的方法

第8条：改写equals时请遵守通用约定

第9条：改写equals时总要改写hashCode

第10条：始终要改写toString

第11条：谨慎地改写clone

第12条：考虑实现Comparable接口

第4章 类和接口

第13条：使类和成员的可访问性最小化

第14条：在公有类中使用访问方法而非公有域

第15条：使非可变性最小化

第16条：复合优先于继承

第17条：要么为继承而设计，并提供文档说明，要么就禁止继承

第18条：接口优于抽象类

第19条：接口只用于定义类型

第20条：类层次优于标签类

第21条：用函数对象表示策略

第22条：优先考虑静态成员类

第5章 泛型

第23条：请不要在新代码中使用原生态类型

第24条：消除非受检警告

第25条：列表优先于数组

第26条：优先考虑泛型

第27条：优先考虑泛型方法

第28条：利用有限制通配符来提升API的灵活性

第29条：优先考虑类型安全的异构容器

第6章 枚举和注解

第30条：用enum代替int常量

第31条：用实例域代替序数

第32条：用EnumSet代替位域

第33条：用EnumMap代替序数索引

第34条：用接口模拟可伸缩的枚举

第35条：注解优先于命名模式

第36条：坚持使用Override注解

第37条：用标记接口定义类型

第7章 方法

第38条：检查参数的有效性

第39条：必要时进行保护性拷贝

第40条：谨慎设计方法签名

第41条：慎用重载

第42条：慎用可变参数 (varargs)

第43条：返回零长度的数组或者集合，而不是null

第44条：为所有导出的API元素编写文档注释

第8章 通用程序设计

第45条：将局部变量的作用域最小化

第46条：for-each循环优先于传统的for循环

第47条：了解和使用类库

第48条：如果需要精确的答案，请避免使用float和double

第49条：原语类型优先于装箱的原语类型

第50条：如果其他类型更适合，则尽量避免使用字符串

第51条：了解字符串连接的性能

第52条：通过接口引用对象

第53条：接口优先于反射机制

第54条：谨慎地使用本地方法

第55条：谨慎地进行优化

第56条：遵守普遍接受的命名惯例

第9章 异常

第57条：只针对异常的条件才使用异常

第58条：对可恢复的条件使用受检异常，对编程错误使用运行时异常

第59条：避免不必要地使用受检的异常

第60条：尽量使用标准的异常

第61条：抛出与抽象相对应的异常

第62条：每个方法抛出的所有异常都要有文档

第63条：在细节消息中包含失败-捕获信息

第64条：努力使失败保持原子性

第65条：不要忽略异常

第10章 并发

第66条：同步访问共享的可变数据

第67条：避免过多同步

第68条：executor和task优先于线程

第69条：并发工具优先于wait和notify

第70条：线程安全性的文档化

第71条：慎用延迟初始化

第72条：不要依赖于线程调度器

第73条：避免使用线程组

第11章 序列化

第74条：谨慎地实现Serializable

第75条：考虑使用自定义的序列化形式

第76条：保护性地编写readObject方法

第77条：对于实例控制，枚举类型优先于readResolve

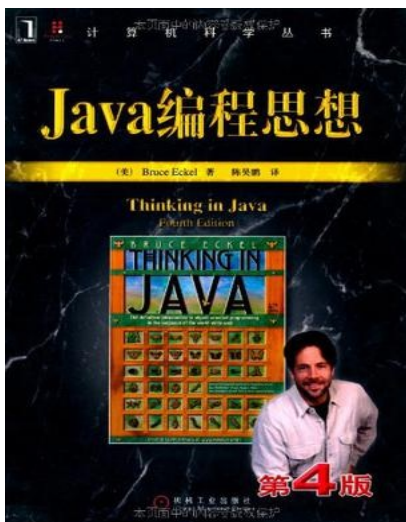
第78条：考虑用序列化代理代替序列化实例

购买链接

[淘宝购买](#)

Java编程思想

Java编程思想（第4版）



原名: Thinking in Java

中文版: Java编程思想（第4版）

作者: Bruce Eckel

译者: 陈昊鹏

出版社: 机械工业出版社

出版时间: 2007-6

豆瓣评分: 9.1 分

推荐理由

本书赢得了全球程序员的广泛赞誉，并斩多项奖项：

- 2003年《Software Development》杂志最佳书籍Jolt大奖
- 2003年《Java Developer's Journal》读者选择最佳书籍奖
- 2001年《Java World》编辑选择最佳书籍奖
- 2000年《Java World》读者选择最佳书籍奖
- 1999年《Software Development》杂志最佳产品奖
- 1998年《Java Developer's Journal》编辑选择最佳书籍奖

从本书获得的各项大奖中，不难看出这是一本经典之作。本书的作者拥有多年教学经验，对C、C++以及Java语言都有独到、深入的见解，以通俗易懂及小而直接的示例解释了一个个晦涩抽象的概念。

有人说这本书是Java开发入门的经典书籍，也有人说这本书并不适合初学者阅读。本人以为，该书绝对是一本十分经典的Java书籍，但是适合于有一定Java基础并且对面向对象有一定理解的人阅读。通过本书把自己零散的知识连贯成一个体系。

简介

本书共22章，包括操作符、控制执行流程、访问权限控制、复用类、多态、接口、通过异常处理错误、字符串、泛型、数组、容器深入研究、Java/I/O系统、枚举类型、并发以及图形化用户界面等内容。这些丰富的内容，包含了Java语言基础语法以及高级特性，适合各个层次的Java程序员阅读，同时也是高等院校讲授面向对象程序设计语言以及Java语言的绝佳教材和参考书。

作者简介

Bruce Eckel是MindView公司（www.MindView.net）的总裁，该公司向客户提供软件咨询和培训。他是C++标准委员会拥有表决权的成员之一，拥有应用物理学学士和计算机工程硕士学位。除本书外，他还是《C++编程思想》的作者，并与人合著了《C++编程思想 第2卷》（这两本书的英文影印版及中文版均已由机械工业出版社引进出版）及其他著作。他已经发表了150多篇论文，还经常参加世界各地的研讨会并进行演讲。

目录

出版者的话	
专家指导委员会	
读者评论	
关于《Thinking in C++》	
译者序	
译者简介	
前言	
绪论	
第1章 对象导论	1
1.1 抽象过程	1
1.2 每个对象都有一个接口	2
1.3 每个对象都提供服务	4

- 1.4 被隐藏的具体实现 4
- 1.5 复用具体实现 5
- 1.6 继承 6
 - 1.6.1 “是一个”与“像是一个”关系 8
- 1.7 伴随多态的可互换对象 8
- 1.8 单根继承结构 11
- 1.9 容器 11
 - 1.9.1 参数化类型（范型） 12
- 1.10 对象的创建和生命期 13
- 1.11 异常处理：处理错误 14
- 1.12 并发编程 14
- 1.13 Java与Internet 15
 - 1.13.1 Web是什么 15
 - 1.13.2 客户端编程 16
 - 1.13.3 服务器端编程 19
- 1.14 总结 20
- 第2章 一切都是对象 21
 - 2.1 用引用操纵对象 21
 - 2.2 必须由你创建所有对象 22
 - 2.2.1 存储到什么地方 22
 - 2.2.2 特例：基本类型 23
 - 2.2.3 Java中的数组 24
 - 2.3 永远不需要销毁对象 24
 - 2.3.1 作用域 24
 - 2.3.2 对象的作用域 25
 - 2.4 创建新的数据类型：类 25
 - 2.4.1 字段和方法 26
 - 2.5 方法、参数和返回值 27
 - 2.5.1 参数列表 27
 - 2.6 构建一个Java程序 28
 - 2.6.1 名字可见性 28
 - 2.6.2 运用其他构件 28
 - 2.6.3 static 关键字 29
 - 2.7 你的第一个Java程序 30
 - 2.7.1 编译和运行 31
 - 2.8 注释和嵌入式文档 32
 - 2.8.1 注释文档 32
 - 2.8.2 语法 33
 - 2.8.3 嵌入式HTML 33
 - 2.8.4 一些标签示例 34

- 2.8.5 文档示例 35
- 2.9 编码风格 36
- 2.10 总结 36
- 2.11 练习 37
- 第3章 操作符 38
 - 3.1 更简单的打印语句 38
 - 3.2 使用Java操作符 39
 - 3.3 优先级 39
 - 3.4 赋值 39
 - 3.4.1 方法调用中的别名问题 40
 - 3.5 算术操作符 41
 - 3.5.1 一元加、减操作符 43
 - 3.6 自动递增和递减 43
 - 3.7 关系操作符 44
 - 3.7.1 测试对象的等价性 44
 - 3.8 逻辑操作符 45
 - 3.8.1 短路 46
 - 3.9 直接常量 47
 - 3.9.1 指数记数法 48
 - 3.10 按位操作符 49
 - 3.11 移位操作符 49
 - 3.12 三元操作符 if-else 52
 - 3.13 字符串操作符 + 和 += 53
 - 3.14 使用操作符时常犯的错误 54
 - 3.15 类型转换操作符 54
 - 3.15.1 截尾和舍入 55
 - 3.15.2 提升 56
 - 3.16 Java没有sizeof 56
 - 3.17 操作符小结 56
 - 3.18 总结 63
- 第4章 控制执行流程 64
 - 4.1 true和false 64
 - 4.2 if-else 64
 - 4.3 迭代 65
 - 4.3.1 do-while 65
 - 4.3.2 for 66
 - 4.3.3 逗号操作符 67
 - 4.4 Foreach语法 67
 - 4.5 return 69
 - 4.6 break和 continue 69

- 4.7 臭名昭著的goto 70
- 4.8 switch 73
- 4.9 总结 75
- 第5章 初始化与清理 76
 - 5.1 用构造器确保初始化 76
 - 5.2 方法重载 77
 - 5.2.1 区分重载方法 79
 - 5.2.2 涉及基本类型的重载 79
 - 5.2.3 以返回值区分重载方法 82
 - 5.3 默认构造器 83
 - 5.4 this关键字 84
 - 5.4.1 在构造器中调用构造器 85
 - 5.4.2 static的含义 86
 - 5.5 清理：终结处理和垃圾回收 87
 - 5.5.1 finalize()的用途何在 87
 - 5.5.2 你必须实施清理 88
 - 5.5.3 终结条件 88
 - 5.5.4 垃圾回收器如何工作 89
 - 5.6 成员初始化 91
 - 5.6.1 指定初始化 93
 - 5.7 构造器初始化 94
 - 5.7.1 初始化顺序 94
 - 5.7.2 静态数据的初始化 95
 - 5.7.3 显式的静态初始化 96
 - 5.7.4 非静态实例初始化 97
 - 5.8 数组初始化 98
 - 5.8.1 可变参数列表 102
 - 5.9 枚举类型 105
 - 5.10 总结 107
- 第6章 访问权限控制 109
 - 6.1 包：库单元 110
 - 6.1.1 代码组织 110
 - 6.1.2 创建独一无二的包名 111
 - 6.1.3 定制工具库 114
 - 6.1.4 用 import改变行为 115
 - 6.1.5 对使用包的忠告 115
 - 6.2 Java访问权限修饰词 116
 - 6.2.1 包访问权限 116
 - 6.2.2 public:接口访问权限 116
 - 6.2.3 private: 你无法访问 118

- 6.2.4 protected: 继承访问权限 118
- 6.3 接口和实现 120
- 6.4 类的访问权限 121
- 6.5 总结 123
- 第7章 复用类 125
 - 7.1 组合语法 125
 - 7.2 继承语法 127
 - 7.2.1 初始化基类 129
 - 7.3 代理 130
 - 7.4 结合使用组合和继承 132
 - 7.4.1 确保正确清理 133
 - 7.4.2 名称屏蔽 135
 - 7.5 在组合与继承之间选择 137
 - 7.6 protected关键字 138
 - 7.7 向上转型 139
 - 7.7.1 为什么称为向上转型 139
 - 7.7.2 再论组合与继承 140
 - 7.8 final关键字 140
 - 7.8.1 final 数据 140
 - 7.8.2 final 方法 143
 - 7.8.3 final 类 144
 - 7.8.4 有关final的忠告 145
 - 7.9 初始化及类的加载 145
 - 7.9.1 继承与初始化 146
 - 7.10 总结 147
- 第8章 多态 148
 - 8.1 再论向上转型 148
 - 8.1.1 忘记对象类型 149
 - 8.2 转机 150
 - 8.2.1 方法调用绑定 150
 - 8.2.2 产生正确的行为 151
 - 8.2.3 可扩展性 153
 - 8.2.4 缺陷：“覆盖”私有方法 156
 - 8.2.5 缺陷：域与静态方法 156
 - 8.3 构造器和多态 157
 - 8.3.1 构造器的调用顺序 157
 - 8.3.2 继承与清理 159
 - 8.3.3 构造器内部的多态方法的行为 162
 - 8.4 协变返回类型 164
 - 8.5 用继承进行设计 165

- 8.5.1 纯继承与扩展 166
- 8.5.2 向下转型与运行时类型识别 167
- 8.6 总结 168
- 第9章 接口 169
 - 9.1 抽象类和抽象方法 169
 - 9.2 接口 172
 - 9.3 完全解耦 174
 - 9.4 Java中的多重继承 178
 - 9.5 通过继承来扩展接口 180
 - 9.5.1 组合接口时的名字冲突 181
 - 9.6 适配接口 181
 - 9.7 接口中的域 183
 - 9.7.1 初始化接口中的域 184
 - 9.8 嵌套接口 185
 - 9.9 接口与工厂 186
 - 9.10 总结 188
- 第10章 内部类 190
 - 10.1 创建内部类 190
 - 10.2 链接到外部类 191
 - 10.3 使用.this与.new 193
 - 10.4 内部类与向上转型 194
 - 10.5 在方法和作用域内的内部类 195
 - 10.6 匿名内部类 196
 - 10.6.1 再访工厂方法 199
 - 10.7 嵌套类 201
 - 10.7.1 接口内部的类 202
 - 10.7.2 从多层嵌套类中访问外部类的成员 203
 - 10.8 为什么需要内部类 204
 - 10.8.1 闭包与回调 205
 - 10.8.2 内部类与控制框架 207
 - 10.9 内部类的继承 212
 - 10.10 内部类可以被覆盖吗 212
 - 10.11 局部内部类 214
 - 10.12 内部类标识符 215
 - 10.13 总结 215
- 第11章 持有对象 216
 - 11.1 泛型和类型安全的容器 216
 - 11.2 基本概念 219
 - 11.3 添加一组元素 220

- 11.4 容器的打印 221
- 11.5 List 223
- 11.6 迭代器 226
 - 11.6.1 ListIterator 227
- 11.7 LinkedList 228
- 11.8 Stack 229
- 11.9 Set 231
- 11.10 Map 233
- 11.11 Queue 236
 - 11.11.1 PriorityQueue 237
- 11.12 Collection和Iterator 238
- 11.13 Foreach与迭代器 241
 - 11.13.1 适配器方法惯用法 243
- 11.14 总结 248
- 第12章 通过异常处理错误 248
 - 12.1 概念 249
 - 12.2 基本异常 249
 - 12.2.1 异常参数 250
 - 12.3 捕获异常 250
 - 12.3.1 try块 250
 - 12.3.2 异常处理程序 250
 - 12.4 创建自定义异常 251
 - 12.4.1 异常与记录日志 253
 - 12.5 异常说明 256
 - 12.6 捕获所有异常 256
 - 12.6.1 栈轨迹 257
 - 12.6.2 重新抛出异常 258
 - 12.6.3 异常链 260
 - 12.7 Java标准异常 263
 - 12.7.1 特例: RuntimeException 263
 - 12.8 使用finally进行清理 264
 - 12.8.1 finally用来做什么 265
 - 12.8.2 在return中使用finally 267
 - 12.8.3 缺憾: 异常丢失 268
 - 12.9 异常的限制 269
 - 12.10 构造器 271
 - 12.11 异常匹配 275
 - 12.12 其他可选方式 276
 - 12.12.1 历史 277
 - 12.12.2 观点 278

- 12.12.3 把异常传递给控制台 279
- 12.12.4 把“被检查的异常”转换为“不受检查的异常” 279
- 12.13 异常使用指南 281
- 12.14 总结 281
- 第13章 字符串 283
- 13.1 不可变String 283
- 13.2 重载“+”与StringBuilder 283
- 13.3 无意识的递归 287
- 13.4 String上的操作 288
- 13.5 格式化输出 289
- 13.5.1 printf() 289
- 13.5.2 System.out.format() 289
- 13.5.3 Formatter类 290
- 13.5.4 格式化说明符 291
- 13.5.5 Formatter转换 292
- 13.5.6 String.format() 294
- 13.6 正则表达式 295
- 13.6.1 基础 295
- 13.6.2 创建正则表达式 297
- 13.6.3 量词 299
- 13.6.4 Pattern和Matcher 300
- 13.6.5 split() 305
- 13.6.6 替换操作 306
- 13.6.7 reset() 307
- 13.6.8 正则表达式与Java I/O 307
- 13.7 扫描输入 309
- 13.7.1 Scanner定界符 310
- 13.7.2 用正则表达式扫描 311
- 13.8 StringTokenizer 312
- 13.9 总结 312
- 第14章 类型信息 313
- 14.1 为什么需要RTTI 313
- 14.2 Class对象 314
- 14.2.1 类字面常量 318
- 14.2.2 泛化的Class引用 320
- 14.2.3 新的转型语法 322
- 14.3 类型转换前先做检查 322
- 14.3.1 使用类字面常量 327
- 14.3.2 动态的instanceof 329

- 14.3.3 递归计数 330
- 14.4 注册工厂 331
- 14.5 instanceof 与 Class的等价性 333
- 14.6 反射：运行时的类信息 334
 - 14.6.1 类方法提取器 335
- 14.7 动态代理 337
- 14.8 空对象 341
 - 14.8.1 模拟对象与桩 346
- 14.9 接口与类型信息 346
- 14.10 总结 350
- 第15章 泛型 352
 - 15.1 与C++的比较 352
 - 15.2 简单泛型 353
 - 15.2.1 一个元组类库 354
 - 15.2.2 一个堆栈类 356
 - 15.2.3 RandomList 357
 - 15.3 泛型接口 358
 - 15.4 泛型方法 361
 - 15.4.1 杠杆利用类型参数推断 362
 - 15.4.2 可变参数与泛型方法 363
 - 15.4.3 用于Generator的泛型方法 364
 - 15.4.4 一个通用的Generator 364
 - 15.4.5 简化元组的使用 366
 - 15.4.6 一个Set实用工具 367
 - 15.5 匿名内部类 369
 - 15.6 构建复杂模型 371
 - 15.7 擦除的神秘之处 372
 - 15.7.1 C++的方式 373
 - 15.7.2 迁移兼容性 375
 - 15.7.3 擦除的问题 376
 - 15.7.4 边界处的动作 377
 - 15.8 擦除的补偿 380
 - 15.8.1 创建类型实例 381
 - 15.8.2 泛型数组 383
 - 15.9 边界 386
 - 15.10 通配符 389
 - 15.10.1 编译器有多聪明 391
 - 15.10.2 逆变 393
 - 15.10.3 无界通配符 395
 - 15.10.4 捕获转换 399

- 15.11 问题 400
 - 15.11.1 任何基本类型都不能作为类型参数 400
 - 15.11.2 实现参数化接口 401
 - 15.11.3 转型和警告 402
 - 15.11.4 重载 403
 - 15.11.5 基类劫持了接口 404
- 15.12 自限定的类型 404
 - 15.12.1 古怪的循环泛型 404
 - 15.12.2 自限定 405
 - 15.12.3 参数协变 407
- 15.13 动态类型安全 409
- 15.14 异常 410
- 15.15 混型 412
 - 15.15.1 C++中的混型 412
 - 15.15.2 与接口混合 413
 - 15.15.3 使用装饰器模式 414
 - 15.15.4 与动态代理混合 415
- 15.16 潜在类型机制 416
- 15.17 对缺乏潜在类型机制的补偿 420
 - 15.17.1 反射 420
 - 15.17.2 将一个方法应用于序列 421
 - 15.17.3 当你并未碰巧拥有正确的接口时 423
 - 15.17.4 用适配器仿真潜在类型机制 424
- 15.18 将函数对象用作策略 426
- 15.19 总结：转型真的如此之糟吗？ 430
 - 15.19.1 进阶读物 432
- 第16章 数组 433
 - 16.1 数组为什么特殊 433
 - 16.2 数组是第一级对象 434
 - 16.3 返回一个数组 436
 - 16.4 多维数组 437
 - 16.5 数组与泛型 440
 - 16.6 创建测试数据 442
 - 16.6.1 Arrays.fill() 442
 - 16.6.2 数据生成器 443
 - 16.6.3 从Generator中创建数组 447
 - 16.7 Arrays实用功能 450
 - 16.7.1 复制数组 450
 - 16.7.2 数组的比较 451

- 16.7.3 数组元素的比较 452
- 16.7.4 数组排序 454
- 16.7.5 在已排序的数组中查找 455
- 16.8 总结 457
- 第17章 容器深入研究 459
- 17.1 完整的容器分类法 459
- 17.2 填充容器 460
- 17.2.1 一种Generator解决方案 460
- 17.2.2 Map生成器 462
- 17.2.3 使用Abstract类 464
- 17.3 Collection的功能方法 470
- 17.4 可选操作 472
- 17.4.1 未获支持的操作 473
- 17.5 List的功能方法 474
- 17.6 Set和存储顺序 477
- 17.6.1 SortedSet 479
- 17.7 队列 480
- 17.7.1 优先级队列 481
- 17.7.2 双向队列 482
- 17.8 理解Map 483
- 17.8.1 性能 484
- 17.8.2 SortedMap 486
- 17.8.3 LinkedHashMap 487
- 17.9 散列与散列码 488
- 17.9.1 理解hashCode() 490
- 17.9.2 为速度而散列 492
- 17.9.3 覆盖hashCode() 495
- 17.10 选择接口的不同实现 499
- 17.10.1 性能测试框架 499
- 17.10.2 对List的选择 502
- 17.10.3 微基准测试的危险 507
- 17.10.4 对Set的选择 508
- 17.10.5 对Map的选择 509
- 17.11 实用方法 512
- 17.11.1 List的排序和查询 514
- 17.11.2 设定Collection或Map为不可修改 515
- 17.11.3 Collection或Map的同步控制 516
- 17.12 持有引用 518
- 17.12.1 WeakHashMap 519

- 17.13 Java 1.0/1.1 的容器 520
 - 17.13.1 Vector 和 Enumeration 520
 - 17.13.2 Hashtable 521
 - 17.13.3 Stack 521
 - 17.13.4 BitSet 522
- 17.14 总结 524
- 第18章 Java I/O系统 525
 - 18.1 File类 525
 - 18.1.1 目录列表器 525
 - 18.1.2 目录实用工具 528
 - 18.1.3 目录的检查及创建 532
 - 18.2 输入和输出 533
 - 18.2.1 InputStream类型 534
 - 18.2.2 OutputStream类型 535
 - 18.3 添加属性和有用的接口 535
 - 18.3.1 通过FilterInputStream从InputStream读取数据 535
 - 18.3.2 通过FilterOutputStream向OutputStream写入 536
 - 18.4 Reader和Writer 537
 - 18.4.1 数据的来源和去处 537
 - 18.4.2 更改流的行为 538
 - 18.4.3 未发生变化的类 539
 - 18.5 自我独立的类：RandomAccessFile 539
 - 18.6 I/O流的典型使用方式 539
 - 18.6.1 缓冲输入文件 540
 - 18.6.2 从内存输入 540
 - 18.6.3 格式化的内存输入 541
 - 18.6.4 基本的文件输出 542
 - 18.6.5 存储和恢复数据 543
 - 18.6.6 读写随机访问文件 544
 - 18.6.7 管道流 545
 - 18.7 文件读写的实用工具 545
 - 18.7.1 读取二进制文件 548
 - 18.8 标准I/O 548
 - 18.8.1 从标准输入中读取 548
 - 18.8.2 将System.out转换成PrintWriter 549
 - 18.8.3 标准I/O重定向 549
 - 18.9 进程控制 550
 - 18.10 新I/O 551

- 18.10.1 转换数据 554
- 18.10.2 获取基本类型 556
- 18.10.3 视图缓冲器 557
- 18.10.4 用缓冲器操纵数据 560
- 18.10.5 缓冲器的细节 560
- 18.10.6 内存映射文件 563
- 18.10.7 文件加锁 566
- 18.11 压缩 568
 - 18.11.1 用GZIP进行简单压缩 568
 - 18.11.2 用Zip进行多文件保存 569
 - 18.11.3 Java档案文件 570
- 18.12 对象序列化 571
 - 18.12.1 寻找类 574
 - 18.12.2 序列化的控制 575
 - 18.12.3 使用“持久性” 581
- 18.13 XML 586
- 18.14 Preferences 588
- 18.15 总结 589
- 第19章 枚举类型 590
 - 19.1 基本enum特性 590
 - 19.1.1 将静态导入用于enum 591
 - 19.2 向enum中添加新方法 591
 - 19.2.1 覆盖enum的方法 592
 - 19.3 switch语句中的enum 593
 - 19.4 values()的神秘之处 594
 - 19.5 实现，而非继承 596
 - 19.6 随机选取 596
 - 19.7 使用接口组织枚举 597
 - 19.8 使用EnumSet替代标志 600
 - 19.9 使用EnumMap 602
 - 19.10 常量相关的方法 603
 - 19.10.1 使用enum的职责链 606
 - 19.10.2 使用enum的状态机 609
 - 19.11 多路分发 613
 - 19.11.1 使用enum分发 615
 - 19.11.2 使用常量相关的方法 616
 - 19.11.3 使用EnumMap分发 618
 - 19.11.4 使用二维数组 618
 - 19.12 总结 619
- 第20章 注解 620

- 20.1 基本语法 620
 - 20.1.1 定义注解 621
 - 20.1.2 元注解 622
- 20.2 编写注解处理器 622
 - 20.2.1 注解元素 623
 - 20.2.2 默认值限制 624
 - 20.2.3 生成外部文件 624
 - 20.2.4 注解不支持继承 627
 - 20.2.5 实现处理器 627
- 20.3 使用apt处理注解 629
- 20.4 将观察者模式用于apt 632
- 20.5 基于注解的单元测试 634
 - 20.5.1 将@Unit用于泛型 641
 - 20.5.2 不需要任何“套件” 642
 - 20.5.3 实现@Unit 642
 - 20.5.4 移除测试代码 647
- 20.6 总结 649
- 第21章 并发 650
 - 21.1 并发的多面性 651
 - 21.1.1 更快的执行 651
 - 21.1.2 改进代码设计 653
 - 21.2 基本的线程机制 653
 - 21.2.1 定义任务 654
 - 21.2.2 Thread类 655
 - 21.2.3 使用Executor 656
 - 21.2.4 从任务中产生返回值 658
 - 21.2.5 休眠 659
 - 21.2.6 优先级 660
 - 21.2.7 让步 661
 - 21.2.8 后台线程 662
 - 21.2.9 编码的变体 665
 - 21.2.10 术语 669
 - 21.2.11 加入一个线程 669
 - 21.2.12 创建有响应的用户界面 671
 - 21.2.13 线程组 672
 - 21.2.14 捕获异常 672
 - 21.3 共享受限资源 674
 - 21.3.1 不正确地访问资源 674
 - 21.3.2 解决共享资源竞争 676
 - 21.3.3 原子性与易变性 680

- 21.3.4 原子类 684
- 21.3.5 临界区 685
- 21.3.6 在其他对象上同步 689
- 21.3.7 线程本地存储 690
- 21.4 终结任务 691
 - 21.4.1 装饰性花园 691
 - 21.4.2 在阻塞时终结 694
 - 21.4.3 中断 695
 - 21.4.4 检查中断 701
- 21.5 线程之间的协作 702
 - 21.5.1 wait()与notifyAll() 703
 - 21.5.2 notify()与notifyAll() 707
 - 21.5.3 生产者与消费者 709
 - 21.5.4 生产者-消费者与队列 713
 - 21.5.5 任务间使用管道进行输入/输出 717
- 21.6 死锁 718
- 21.7 新类库中的构件 722
 - 21.7.1 CountdownLatch 722
 - 21.7.2 CyclicBarrier 724
 - 21.7.3 DelayQueue 726
 - 21.7.4 PriorityBlockingQueue 728
 - 21.7.5 使用ScheduledExecutor的温室控制器 730
 - 21.7.6 Semaphore 733
 - 21.7.7 Exchanger 735
- 21.8 仿真 737
 - 21.8.1 银行出纳员仿真 737
 - 21.8.2 饭店仿真 741
 - 21.8.3 分发工作 744
- 21.9 性能调优 748
 - 21.9.1 比较各类互斥技术 748
 - 21.9.2 免锁容器 754
 - 21.9.3 乐观加锁 760
 - 21.9.4 ReadWriteLock 761
- 21.10 活动对象 763
- 21.11 总结 766
- 21.12 进阶读物 767
- 第22章 图形化用户界面 768
 - 22.1 applet 769
 - 22.2 Swing基础 769

- 22.2.1 一个显示框架 771
- 22.3 创建按钮 772
- 22.4 捕获事件 773
- 22.5 文本区域 774
- 22.6 控制布局 776
 - 22.6.1 BorderLayout 776
 - 22.6.2 FlowLayout 776
 - 22.6.3 GridLayout 777
 - 22.6.4 GridBagLayout 777
 - 22.6.5 绝对定位 778
 - 22.6.6 BoxLayout 778
 - 22.6.7 最好的方式是什么 778
- 22.7 Swing事件模型 778
 - 22.7.1 事件与监听器的类型 779
 - 22.7.2 跟踪多个事件 783
- 22.8 Swing组件一览 785
 - 22.8.1 按钮 785
 - 22.8.2 图标 787
 - 22.8.3 工具提示 788
 - 22.8.4 文本域 789
 - 22.8.5 边框 790
 - 22.8.6 一个迷你编辑器 791
 - 22.8.7 复选框 792
 - 22.8.8 单选按钮 793
 - 22.8.9 组合框 793
 - 22.8.10 列表框 794
 - 22.8.11 页签面板 796
 - 22.8.12 消息框 796
 - 22.8.13 菜单 798
 - 22.8.14 弹出式菜单 802
 - 22.8.15 绘图 803
 - 22.8.16 对话框 805
 - 22.8.17 文件对话框 808
 - 22.8.18 Swing组件上的HTML 809
 - 22.8.19 滑块与进度条 810
 - 22.8.20 选择外观 811
 - 22.8.21 树、表格和剪贴板 812
- 22.9 JNLP与Java Web Start 812
- 22.10 Swing与并发 816
 - 22.10.1 长期运行的任务 816

- 22.10.2 可视化线程机制 822
- 22.11 可视化编程与JavaBean 823
 - 22.11.1 JavaBean是什么 824
 - 22.11.2 使用Introspector抽取出BeanInfo 825
 - 22.11.3 一个更复杂的Bean 829
 - 22.11.4 JavaBean与同步 831
 - 22.11.5 把Bean打包 834
 - 22.11.6 对Bean更高级的支持 835
 - 22.11.7 有关Bean的其他读物 836
- 22.12 Swing的可替代选择 836
- 22.13 用Flex构建Flash Web客户端 836
 - 22.13.1 Hello, Flex 837
 - 22.13.2 编译MXML 838
 - 22.13.3 MXML与ActionScript 838
 - 22.13.4 容器与控制 839
 - 22.13.5 效果与样式 840
 - 22.13.6 事件 841
 - 22.13.7 连接到Java 841
 - 22.13.8 数据模型与数据绑定 843
 - 22.13.9 构建和部署 843
- 22.14 创建SWT应用 844
 - 22.14.1 安装SWT 845
 - 22.14.2 Hello, SWT 845
 - 22.14.3 根除冗余代码 847
 - 22.14.4 菜单 848
 - 22.14.5 页签面板、按钮和事件 849
 - 22.14.6 图形 852
 - 22.14.7 SWT中的并发 853
 - 22.14.8 SWT还是Swing 855
- 22.15 总结 855
 - 22.15.1 资源 855
- 附录A 补充材料 856
- 附录B 资源 859
- 索引 863

购买地址

[淘宝购买](#)

深入理解Java虚拟机