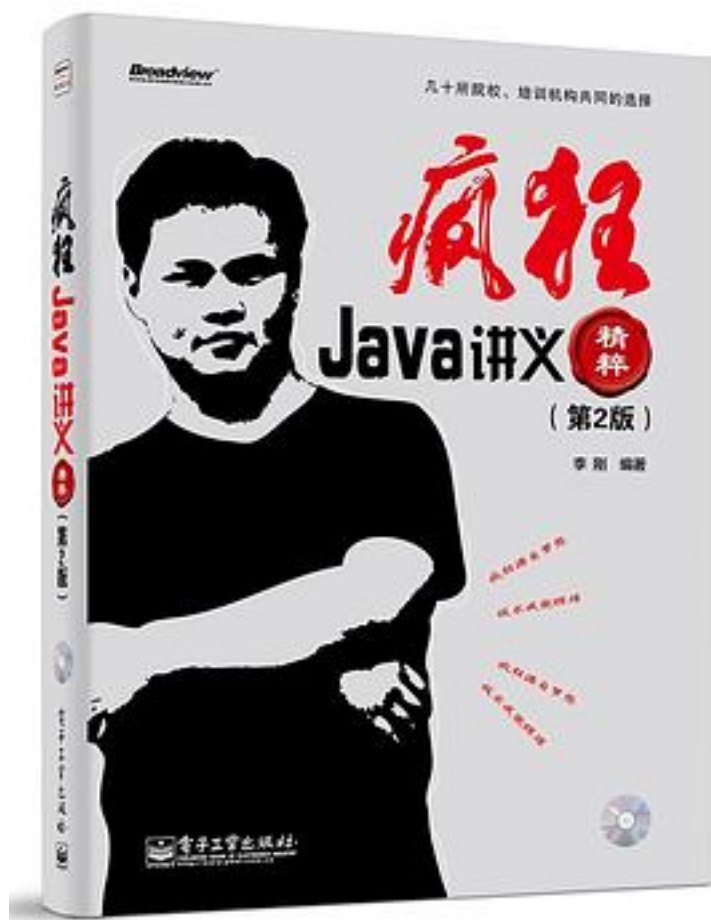


疯狂Java讲义精粹（第2版）



[疯狂Java讲义精粹（第2版）_下载链接1_](#)

著者:李刚

出版者:电子工业出版社

出版时间:2014-10

装帧:

isbn:9787121243462

《疯狂Java讲义精粹（第2版）》是《疯狂Java讲义精粹》的第2版，《疯狂Java讲义精粹（第2版）》相比《疯狂Java讲义》更浅显易懂，讲解更细致，本书同样介绍了Java

8的新特性，《疯狂Java讲义精粹（第2版）》大部分示例程序都采用Lambda表达式、流式API进行了改写，因此务必使用Java 8的JDK来编译、运行。

《疯狂Java讲义精粹（第2版）》尽量浅显、直白地介绍Java编程的相关方面，全书内容覆盖了Java的基本语法结构、Java的面向对象特征、Java集合框架体系、Java泛型、异常处理、Java注释、Java的IO流体系、Java多线程编程、Java网络通信编程。覆盖了java.lang、java.util、java.text、java.io和java.nio包下绝大部分类和接口。《疯狂Java讲义精粹（第2版）》全面介绍了Java 8的新的接口语法、Lambda表达式、方法引用、构造器引用、函数式编程、流式编程、新的日期、时间API、并行支持、改进的类型推断、重复注解、JDBC 4.2新特性等新特性。

《疯狂Java讲义精粹（第2版）》为打算认真掌握Java编程的读者而编写，适合各种层次的Java学习者和工作者阅读。《疯狂Java讲义精粹（第2版）》专门针对高校课程进行过调整，尤其适合作为高校教育、培训机构的Java教材。

作者介绍:

目录: 第1章 Java语言概述与开发环境 1

1.1 Java语言的发展简史 2

1.2 Java程序运行机制 4

1.2.1 高级语言的运行机制 4

1.2.2 Java程序的运行机制和JVM 4

1.3 开发Java的准备 5

1.3.1 下载和安装Java 8的JDK 5

不是说JVM是运行Java程序的虚拟机吗？那JRE和JVM的关系是怎样的呢？ 6

1.3.2 设置PATH环境变量 8

为什么不安装公共JRE呢？ 7

为什么选择用户变量？用户变量与系统变量有什么区别？ 9

1.4 第一个Java程序 9

1.4.1 编辑Java源代码 9

1.4.2 编译Java程序 10

当编译C程序时，不仅需要指定存放目标文件的位置，也需要指定目标文件的文件名，这里使用javac编译Java程序时怎么不需要指定目标文件的文件名呢？ 10

1.4.3 运行Java程序 11

1.4.4 根据CLASSPATH环境变量定位类 11

1.5 Java程序的基本规则 12

1.5.1 Java程序的组织形式 12

1.5.2 Java源文件的命名规则 13

1.5.3 初学者容易犯的错误 13

1.6 何时开始使用IDE工具 15

我想学习Java编程，到底是学习Eclipse好，还是学习NetBeans好呢？ 16

1.7 本章小结 16

本章练习 16

第2章 数据类型和运算符 17

2.1 注释 18

2.1.1 单行注释和多行注释 18

2.1.2 文档注释 19

API文档是什么？ 19

为什么要学习查看API文档的方法？ 21

2.2 标识符和关键字 24

2.2.1 分隔符 24

- 2.2.2 标识符规则 25
- 2.2.3 Java关键字 26
- 2.3 数据类型分类 26
- 什么是变量？变量有什么用？ 26
- 2.4 基本数据类型 27
 - 2.4.1 整型 27
 - 2.4.2 字符型 29
 - 什么是字符集？ 29
 - 2.4.3 浮点型 31
 - 2.4.4 数值中使用下画线分隔 32
 - 2.4.5 布尔型 32
- 2.5 基本类型的类型转换 33
 - 2.5.1 自动类型转换 33
 - 2.5.2 强制类型转换 34
 - 2.5.3 表达式类型的自动提升 35
- 2.6 直接量 36
 - 2.6.1 直接量的类型 36
 - 2.6.2 直接量的赋值 37
- 2.7 运算符 38
 - 2.7.1 算术运算符 38
 - 2.7.2 赋值运算符 40
 - 2.7.3 位运算符 41
 - 2.7.4 扩展后的赋值运算符 43
 - 2.7.5 比较运算符 43
 - 2.7.6 逻辑运算符 44
 - 2.7.7 三目运算符 45
 - 2.7.8 运算符的结合性和优先级 46
- 2.8 本章小结 47
- 本章练习 47
- 第3章 流程控制与数组 48
 - 3.1 顺序结构 49
 - 3.2 分支结构 49
 - 3.2.1 if条件语句 49
 - 3.2.2 增强后的switch分支语句 53
 - 3.3 循环结构 54
 - 3.3.1 while循环语句 55
 - 3.3.2 do while循环语句 56
 - 3.3.3 for循环 57
 - 3.3.4 嵌套循环 59
 - 3.4 控制循环结构 60
 - 3.4.1 使用break结束循环 60
 - 3.4.2 使用continue忽略本次循环剩下语句 61
 - 3.4.3 使用return结束方法 62
 - 3.5 数组类型 63
 - 3.5.1 理解数组：数组也是一种类型 63
 - int[]是一种类型吗？怎么使用这种类型呢？ 63
 - 3.5.2 定义数组 63
 - 3.5.3 数组的初始化 64
 - 能不能只分配内存空间，不赋初始值呢？ 64
 - 3.5.4 使用数组 65
 - 为什么要我记住这些异常信息？ 66
 - 3.5.5 foreach循环 66
 - 3.6 深入数组 68
 - 3.6.1 没有多维数组 68

我是否可以让图3.3中灰色覆盖的数组元素再次指向另一个数组？这样不就可以扩展成三维数组，甚至扩展成更多维的数组吗？ 69

3.6.2 Java 8增强的工具类：Arrays 70

3.7 本章小结 73

本章练习 73

第4章 面向对象（上） 74

4.1 类和对象 75

4.1.1 定义类 75

构造器不是没有返回值吗？为什么不能用void声明呢？ 77

4.1.2 对象的产生和使用 77

4.1.3 对象、引用和指针 78

4.1.4 对象的this引用 79

4.2 方法详解 83

4.2.1 方法的所属性 83

4.2.2 方法的参数传递机制 83

4.2.3 形参个数可变的方法 87

4.2.4 递归方法 88

4.2.5 方法重载 89

为什么方法的返回值类型不能用于区分重载的方法？ 90

4.3 成员变量和局部变量 90

4.3.1 成员变量和局部变量 90

4.3.2 成员变量的初始化和内存中的运行机制 94

4.3.3 局部变量的初始化和内存中的运行机制 95

4.3.4 变量的使用规则 96

4.4 隐藏和封装 97

4.4.1 理解封装 97

4.4.2 使用访问控制符 97

4.4.3 package、import和import static 100

4.4.4 Java的常用包 104

4.5 深入构造器 105

4.5.1 使用构造器执行初始化 105

构造器是创建Java对象的途径，是不是说构造器完全负责创建Java对象？ 106

4.5.2 构造器重载 106

为什么要用this来调用另一个重载的构造器？我把另一个构造器里的代码复制、粘贴到这个构造器里不就可以了么？ 107

4.6 类的继承 108

4.6.1 继承的特点 108

4.6.2 重写父类的方法 109

4.6.3 super限定 110

4.6.4 调用父类构造器 112

为什么我创建Java对象时从未感觉到java.lang. Object类的构造器被调用过？ 114

4.7 多态 115

4.7.1 多态性 115

4.7.2 引用变量的强制类型转换 116

4.7.3 instanceof运算符 117

4.8 初始化块 118

4.8.1 使用初始化块 118

4.8.2 初始化块和构造器 120

4.8.3 静态初始化块 120

4.9 本章小结 123

本章练习 123

第5章 面向对象（下） 124

5.1 Java 8增强的包装类 125

Java为什么要对这些数据进行缓存呢？ 128

- 5.2 处理对象 129
 - 5.2.1 打印对象和toString方法 129
 - 5.2.2 ==和equals方法 130
 - 上面程序中判断obj是否为Person类的实例时，为何不用obj instanceof Person来判断呢？ 134
- 5.3 类成员 134
 - 5.3.1 理解类成员 134
 - 5.3.2 单例（Singleton）类 135
- 5.4 final修饰符 136
 - 5.4.1 final成员变量 136
 - 5.4.2 final局部变量 138
 - 5.4.3 final修饰基本类型变量和引用类型变量的区别 139
 - 5.4.4 可执行“宏替换”的final变量 139
 - 5.4.5 final方法 141
 - 5.4.6 final类 142
- 5.5 抽象类 142
 - 5.5.1 抽象方法和抽象类 142
 - 5.5.2 抽象类的作用 145
- 5.6 Java 8改进的接口 146
 - 5.6.1 接口的概念 146
 - 5.6.2 Java 8中接口的定义 147
 - 5.6.3 接口的继承 149
 - 5.6.4 使用接口 149
 - 5.6.5 接口和抽象类 151
- 5.7 内部类 152
 - 5.7.1 非静态内部类 152
 - 非静态内部类对象和外部类对象的关系是怎样的？ 155
 - 5.7.2 静态内部类 156
 - 为什么静态内部类的实例方法也不能访问外部类的实例属性呢？ 157
 - 接口里是否能定义内部接口？ 158
 - 5.7.3 使用内部类 158
 - 既然内部类是外部类的成员，那么是否可以为外部类定义子类，在子类中再定义一个内部类来重写其父类中的内部类呢？ 160
 - 5.7.4 局部内部类 160
 - 5.7.5 Java 8改进的匿名内部类 161
- 5.8 Java 8新增的Lambda表达式 164
 - 5.8.1 Lambda表达式入门 164
 - 5.8.2 Lambda表达式与函数式接口 166
 - 5.8.3 方法引用与构造器引用 168
 - 5.8.4 Lambda表达式与匿名内部类的联系和区别 171
 - 5.8.5 使用Lambda表达式调用Arrays的类方法 172
- 5.9 枚举类 172
 - 5.9.1 手动实现枚举类 173
 - 5.9.2 枚举类入门 173
 - 5.9.3 枚举类的成员变量、方法和构造器 175
 - 5.9.4 实现接口的枚举类 177
 - 枚举类不是用final修饰了吗？怎么还能派生子类呢？ 178
 - 5.9.5 包含抽象方法的枚举类 178
- 5.10 修饰符的适用范围 179
- 5.11 本章小结 180
- 本章练习 180
- 第6章 Java基础类库 181
 - 6.1 与用户互动 182
 - 6.1.1 运行Java程序的参数 182

- 6.1.2 使用Scanner获取键盘输入 183
- 6.2 系统相关 185
 - 6.2.1 System类 185
 - 6.2.2 Runtime类 187
- 6.3 常用类 188
 - 6.3.1 Object类 188
 - 6.3.2 Objects类 189
 - 6.3.3 String、StringBuffer和StringBuilder类 190
 - 6.3.4 Math类 193
 - 6.3.5 ThreadLocalRandom与Random 195
 - 6.3.6 BigDecimal类 196
- 6.4 Java 8的日期、时间类 199
 - 6.4.1 Date类 199
 - 6.4.2 Calendar类 199
 - 6.4.3 Java 8新增的日期、时间包 202
- 6.5 Java 8新增的日期、时间格式器 204
 - 6.5.1 使用DateTimeFormatter完成格式化 205
 - 6.5.2 使用DateTimeFormatter解析字符串 206
- 6.6 本章小结 206
- 本章练习 206
- 第7章 Java集合 207
 - 7.1 Java集合概述 208
 - 7.2 Collection和Iterator接口 209
 - 7.2.1 使用Lambda表达式遍历集合 211
 - 7.2.2 使用Java 8增强的Iterator遍历集合元素 211
 - 7.2.3 使用Lambda表达式遍历Iterator 213
 - 7.2.4 使用foreach循环遍历集合元素 213
 - 7.2.5 使用Java 8新增的Predicate操作集合 214
 - 7.2.6 使用Java 8新增的Stream操作集合 215
 - 7.3 Set集合 217
 - 7.3.1 HashSet类 217
 - hashCode()方法对于HashSet是不是十分重要? 219
 - 7.3.2 LinkedHashSet类 221
 - 7.3.3 TreeSet类 222
 - 7.4 List集合 227
 - 7.4.1 Java 8改进的List接口和ListIterator接口 227
 - 7.4.2 ArrayList和Vector实现类 231
 - 7.4.3 固定长度的List 231
 - 7.5 Queue集合 232
 - 7.5.1 PriorityQueue实现类 232
 - 7.5.2 Deque接口与ArrayDeque实现类 233
 - 7.5.3 LinkedList实现类 235
 - 7.5.4 各种线性表的性能分析 236
 - 7.6 Java 8增强的Map集合 236
 - 7.6.1 Java 8为Map新增的方法 238
 - 7.6.2 Java 8改进的HashMap和Hashtable实现类 239
 - 7.6.3 LinkedHashMap实现类 242
 - 7.6.4 使用Properties读写属性文件 243
 - 7.6.5 SortedMap接口和TreeMap实现类 244
 - 7.6.6 各Map实现类的性能分析 246
 - 7.7 HashSet和HashMap的性能选项 246
 - 7.8 操作集合的工具类: Collections 247
 - 7.8.1 排序操作 247
 - 7.8.2 查找、替换操作 250

- 7.8.3 同步控制 251
- 7.8.4 设置不可变集合 251
- 7.9 烦琐的接口：Enumeration 252
- 7.10 本章小结 253
- 本章练习 253
- 第8章 泛型 254
 - 8.1 泛型入门 255
 - 8.1.1 编译时不检查类型的异常 255
 - 8.1.2 使用泛型 255
 - 8.1.3 泛型的“菱形”语法 256
 - 8.2 深入泛型 257
 - 8.2.1 定义泛型接口、类 257
 - 8.2.2 从泛型类派生子类 259
 - 8.2.3 并不存在泛型类 260
 - 8.3 类型通配符 260
 - 8.3.1 使用类型通配符 262
 - 8.3.2 设定类型通配符的上限 262
 - 8.3.3 设定类型形参的上限 264
 - 8.4 泛型方法 264
 - 8.4.1 定义泛型方法 265
 - 8.4.2 泛型方法和类型通配符的区别 267
 - 8.4.3 “菱形”语法与泛型构造器 268
 - 8.4.4 设定通配符下限 269
 - 8.4.5 泛型方法与方法重载 271
 - 8.4.6 Java 8改进的类型推断 272
 - 8.5 擦除和转换 272
 - 8.6 泛型与数组 274
 - 8.7 本章小结 275
- 第9章 异常处理 276
 - 9.1 异常概述 277
 - 9.2 异常处理机制 278
 - 9.2.1 使用try...catch捕获异常 278
 - 9.2.2 异常类的继承体系 279
 - 9.2.3 多异常捕获 282
 - 9.2.4 访问异常信息 282
 - 9.2.5 使用finally回收资源 283
 - 9.2.6 异常处理的嵌套 285
 - 9.2.7 自动关闭资源的try语句 286
 - 9.3 Checked异常和Runtime异常体系 287
 - 9.3.1 使用throws声明抛出异常 287
 - 9.4 使用throw抛出异常 289
 - 9.4.1 抛出异常 289
 - 9.4.2 自定义异常类 290
 - 9.4.3 catch和throw同时使用 291
 - 9.4.4 增强的throw语句 292
 - 9.4.5 异常链 293
 - 9.5 Java的异常跟踪栈 294
 - 9.6 异常处理规则 296
 - 9.6.1 不要过度使用异常 296
 - 9.6.2 不要使用过于庞大的try块 297
 - 9.6.3 避免使用Catch All语句 298
 - 9.6.4 不要忽略捕获到的异常 298
 - 9.7 本章小结 298
- 本章练习 298

第10章 Annotation（注解）	299
10.1 基本Annotation	300
10.1.1 限定重写父类方法：@Override	300
10.1.2 标示已过时：@Deprecated	301
10.1.3 抑制编译器警告：@SuppressWarnings	302
10.1.4 “堆污染”警告与@SafeVarargs	302
10.1.5 Java 8的函数式接口与@FunctionalInterface	303
10.2 JDK的元Annotation	304
10.2.1 使用@Retention	304
10.2.2 使用@Target	305
10.2.3 使用@Documented	305
10.2.4 使用@Inherited	306
10.3 自定义Annotation	307
10.3.1 定义Annotation	307
10.3.2 提取Annotation信息	308
10.3.3 使用Annotation的示例	310
10.3.4 Java 8新增的重复注解	314
10.3.5 Java 8新增的Type Annotation	316
10.4 编译时处理Annotation	317
10.5 本章小结	320
第11章 输入/输出	321
11.1 File类	322
11.1.1 访问文件和目录	322
11.1.2 文件过滤器	324
11.2 理解Java的IO流	324
11.2.1 流的分类	325
11.2.2 流的概念模型	326
11.3 字节流和字符流	327
11.3.1 InputStream和Reader	327
11.3.2 OutputStream和Writer	329
11.4 输入/输出流体系	330
11.4.1 处理流的用法	330
11.4.2 输入/输出流体系	331
11.4.3 转换流	333
怎么没有把字符流转换成字节流的转换流呢？	334
11.4.4 推回输入流	334
11.5 重定向标准输入/输出	336
11.6 RandomAccessFile	337
11.7 NIO.2	340
11.7.1 Path、Paths和Files核心API	341
11.7.2 使用FileVisitor遍历文件和目录	342
11.7.3 使用WatchService监控文件变化	343
11.7.4 访问文件属性	344
11.8 本章小结	346
本章练习	346
第12章 多线程	347
12.1 线程概述	348
12.1.1 线程和进程	348
12.1.2 多线程的优势	349
12.2 线程的创建和启动	349
12.2.1 继承Thread类创建线程类	350
12.2.2 实现Runnable接口创建线程类	351
12.2.3 使用Callable和Future创建线程	352
12.2.4 创建线程的三种方式对比	354

- 12.3 线程的生命周期 354
 - 12.3.1 新建和就绪状态 354
 - 12.3.2 运行和阻塞状态 356
 - 12.3.3 线程死亡 357
- 12.4 控制线程 358
 - 12.4.1 join线程 358
 - 12.4.2 后台线程 359
 - 12.4.3 线程睡眠: sleep 360
 - 12.4.4 线程让步: yield 360
 - 12.4.5 改变线程优先级 362
- 12.5 线程同步 363
 - 12.5.1 线程安全问题 363
 - 12.5.2 同步代码块 365
 - 12.5.3 同步方法 366
 - 12.5.4 释放同步监视器的锁定 368
 - 12.5.5 同步锁 (Lock) 369
 - 12.5.6 死锁 371
- 12.6 线程通信 372
 - 12.6.1 传统的线程通信 372
 - 12.6.2 使用Condition控制线程通信 376
 - 12.6.3 使用阻塞队列 (BlockingQueue) 控制线程通信 378
- 12.7 线程池 380
 - 12.7.1 Java 8改进的线程池 381
 - 12.7.2 Java 8增强的ForkJoinPool 382
- 12.8 线程相关类 386
 - 12.8.1 ThreadLocal类 386
 - 12.8.2 包装线程不安全的集合 387
 - 12.8.3 线程安全的集合类 388
- 12.9 本章小结 389
- 本章练习 389
- 第13章 网络编程 390
 - 13.1 网络编程的基础知识 391
 - 13.1.1 网络基础知识 391
 - 13.1.2 IP地址和端口号 392
 - 13.2 Java的基本网络支持 393
 - 13.2.1 使用InetAddress 393
 - 13.2.2 使用URLDecoder和URLEncoder 393
 - 13.2.3 URL、URLConnection和URLPermission 395
 - 13.3 基于TCP协议的网络编程 401
 - 13.3.1 TCP协议基础 401
 - 13.3.2 使用ServerSocket创建TCP服务器端 401
 - 13.3.3 使用Socket进行通信 402
 - 13.3.4 加入多线程 404
 - 13.3.5 记录用户信息 407
 - 13.3.6 半关闭的Socket 414
 - 13.3.7 使用NIO实现非阻塞Socket通信 415
 - 13.3.8 使用AIO实现非阻塞通信 420
 - 上面程序中好像没用到④⑤号代码的get()方法的返回值, 这两个地方不调用get()方法行吗? 424
 - 13.4 使用代理服务器 427
 - 13.4.1 直接使用Proxy创建连接 427
 - 13.4.2 使用ProxySelector自动选择代理服务器 428
 - 13.5 本章小结 431
 - 本章练习 431

• • • • • ([收起](#))

[疯狂Java讲义精粹（第2版）_下载链接1](#)

标签

Java

编程

专业书

计算科学

程序设计

java

666666

评论

还行，可以对java有个全面的了解！

看了一半 还了 重新打好基础还是不错的

感觉有点难以读懂，不是一本入门的好书

[疯狂Java讲义精粹（第2版）_下载链接1](#)

书评

[疯狂Java讲义精粹（第2版）_下载链接1](#)