

你是否在求职时被鄙视“基本功不扎实”而万分惭愧？

本书能检验你的基本功，并让你弥补不足。

本书归纳了Java学习者、工作者在学习、工作过程中最欠缺的技术短板。

形成内存管理、常见的陷阱、数据结构、程序调试、软件测试等16堂必修课！

你是否深入研究过它们，是否深入掌握它们？

谨以此书献给打算以编程为职业，并愿意为之疯狂的人。

突破程序员 基本功的16课

Java

李刚 编著

 人民邮电出版社
POSTS & TELECOM PRESS

疯狂Java

突破程序员 基本功的16课

李刚 编著

人民邮电出版社
北京

图书在版编目(CIP)数据

疯狂Java:突破程序员基本功的16课 / 李刚编著
— 北京:人民邮电出版社, 2010.5
ISBN 978-7-115-22168-1

I. ①疯… II. ①李… III. ①JAVA语言—程序设计
IV. ①TP312

中国版本图书馆CIP数据核字(2010)第012230号

内 容 简 介

本书是著名Java领域研究专家、Java语言培训大师、“疯狂Java”创始人李刚老师的又一倾力力作。

在本书中,李刚老师一改枯燥的教学方式,专门面向Java初学者可能会遇到的各种学习问题,由点及面,详细讨论了Java内存管理、Java编程过程中常遇陷阱、常用数据结构的Java实现和Java程序开发的方法与经验等内容。

这些问题,看似“司空见惯”,实际上却是很多Java初学者在初学阶段都会遇到的问题和疑难。李刚老师在本书中,正是试图为读者们展现出这些疑点、难点的实质,让读者能在瞬息之间,彻底掌握住这门语言的“内功心法”。

这不是一本包容了所有技术细节的手册,而是一本Java前辈对于晚辈们的提点和教导。书中很多内容,是李刚老师和他的众多学子曾亲身体验过的桎梏,非常具有参考意义。本书承载了无数前辈的谆谆教导之言,向你展示着一个痛并快乐着的Java世界。

疯狂Java:突破程序员基本功的16课

- ◆ 编 著 李 刚
责任编辑 蒋 佳
- ◆ 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>
中国铁道出版社印刷厂印刷
- ◆ 开本: 800×1000 1/16
印张: 29.25
字数: 670千字 2010年5月第1版
印数: 1-4000册 2010年5月北京第1次印刷

ISBN 978-7-115-22168-1

定价: 59.00元

读者服务热线: (010)67132692 印装质量热线: (010)67129223
反盗版热线: (010)67171154



前言

Java 语言的开发人群越来越大了，大量程序员已经进入或正打算进入 Java 编程领域。这当然和 Java 语言本身的优秀不无关系，却也和 Java 语言入门简单有关。一个毫无编程基础的初学者，只要有点数据库和 SQL 的基础，大概花不到一个月时间就可以学会编写 JSP 页面，说不定这样就可以找到一份 Java 编程的工作了。如果他肯再多下点功夫，学习一下编写 Struts Action 类、配置 Action，编写 Spring Bean 类和配置 Bean，他甚至可以自我感觉很不错了。

问题是：这种“快餐式”、“突击式”的程序员真的能满足要求吗？如果仅仅满足于这些简单的、重复式的开发，他们也许没有太多的问题，但他们可能很少有突破的机会。究其根本原因，很大程度上是因为他们的基本功不够扎实。对他们而言，与其说 Java 是一种面向对象的语言，不如说更像一种脚本；他们从源代码层次来看程序运行（甚至只会从 Eclipse 等集成开发环境中看程序运行），完全无法从底层内存分配的角度来看程序运行；他们天天在用 Java 类库、用 Struts、用 Spring，但对这些东西的实现却知之甚少——这又如何突破自己、获得更好的提高呢？

鉴于此种现状，笔者在“疯狂 Java 实训营”的训练中，除了采用大量实际项目来驱动教学之外，往往会花很多时间、精力来培养学员的基本功。比如讲授 Spring 框架时，如果只关注如何编写 Bean 类、配置 Bean，那么一天的时间就足够了。而笔者往往会深入 Spring 框架的底层实现，带领学生从工厂模式、策略模式、门面模式、代理模式、命令模式的角度来深度分析 Spring 框架实现，然后进行对比，总结 Spring 框架的优势到底在哪里？不使用 Spring 框架是否有替代解决方案？从而感受设计模式对实际开发的帮助。

上面这些内容，看似“高深”，但其实质依然离不开 Java 编程的基本功。也可以这样说：一旦读者真正把基本功打扎实了，将可以看得更高、更透彻。

在这样的背景下，偶然之间我获得了这样一个想法：把这些容易被 Java 程序员所忽视的内容整理成一本书，也许可以帮助这些需要突破基本功的准程序员、初级程序员。本书的写法对笔者来说，是一种新的尝试：笔者以前所写之书，通常都会围绕某方面知识“画一个圆”——把这个方面的相关知识，全面、详细地向读者介绍出来。本书并没有采用这种写法，本书的每课围绕一个相对独立的专题进行讲解，形式上有点类似于“针对性地补课”。当读者感觉自己在某些方面的基本功不够扎实时，可以直接选择相应的内容进行学习，完全无须从头到尾地按顺序阅读。

本书创作过程

本书创作得有些困难，当初刚刚获得这个想法之时，我也曾感到有些力不从心，程序员的基

本书看似非常简单，但覆盖范围却非常广泛，因此似乎很难形成一个完整、系统的知识体系。

但随着写作过程的不断推进，此时笔者心中才渐渐明朗起来：该书即是一本无所不包的解惑之书，它即是将 Java 学习、实际开发中最容易被大家忽视、又非常重要的内容收集起来，便足够了。

期间，ACS（天津）公司邀请笔者为其公司员工进行了一次技术内训，内容主要围绕 Java 程序的内存管理问题。内训结束时该员工反应很好，这给笔者带来了很大的触动。Java 程序的内存管理既是 Java 程序员容易忽视的地方，但又是 Java 编程的重点。后来，笔者即将此次内训的知识进行整理，并最终形成了本书第 1 部分的内容。

本书第 3 部分所介绍的常见数据结构、排序算法的 Java 实现，则是笔者一直想介绍的内容。在笔者的“疯狂 Java 实训营”内，常见数据结构、排序算法是必须掌握的——也许你初涉编程时感受不到这些经典算法的用途，因为你可以直接利用别人的实现，但如果你希望突破自己，上升到另外一个高度时，你就不可避免地需要自己开发类库，而不是总使用别人的类库，那这些经典算法的作用就显现出来了。

本书第 2 部分和第 4 部分则主要来自于参加“疯狂 Java 实训营”的学生，正如每个动手编程的初学者，他们都曾经遭遇着各种各样的陷阱，因此笔者总是提醒他们应该将这些陷阱收集起来，以免再次陷进去。本书第 2 部分收集了 Java 编程中各种常见的陷阱，第 4 部分的内容则解决了他们进入实际开发之前的诸多困扰，包括程序开发的基本方法，有效进行程序调试的方法，如何看待、使用 IDE 工具，软件测试等相关内容。

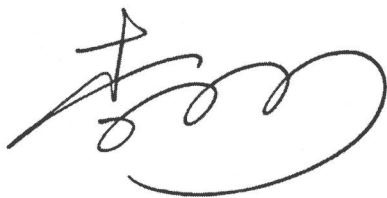
衷心感谢

本书得以成书，离不开人民邮电出版社诸位编辑的热心帮助，他们用心地整理出本书的知识结构，并为本书制定了初步大纲，这些工作给笔者提供很大的帮助。

感谢造物主和妻子在本书交稿之际赐给笔者一个非常可爱的宝宝，他的出现稍稍延迟了本书的部分工作，但却给笔者的生活添加了无限的乐趣。

本书写给谁看

如果你是一名 Java 语言的初学者，并已经学会了 Java 基本的语法，但却经常在动手编程时感到困难重重，或者你已经是一个 Java 程序员了，但在实际开发中却经常感觉力不从心，那么本书将非常适合你。本书会帮助你找出自己的技术短板，迅速突破 Java 编程的基本功，使你成为一名真正的 Java 达人。



2009-12-20



目 录

第 1 课 数组与内存控制	1
1.1 数组初始化.....	2
1.1.1 Java 数组是静态的.....	2
1.1.2 数组一定要初始化吗.....	5
1.1.3 基本类型数组的初始化.....	6
1.1.4 引用类型数组的初始化.....	8
1.2 使用数组.....	11
1.2.1 数组元素就是变量.....	11
1.2.2 没有多维数组.....	13
1.3 小结.....	18
第 2 课 对象与内存控制	19
2.1 实例变量和类变量.....	20
2.1.1 实例变量和类变量的属性.....	21
2.1.2 实例变量的初始化时机.....	24
2.1.3 类变量的初始化时机.....	27
2.2 父类构造器.....	29
2.2.1 隐式调用和显式调用.....	29
2.2.2 访问子类对象的实例变量.....	32
2.2.3 调用被子类重写的方法.....	34
2.3 父子实例的内存控制.....	36
2.3.1 继承成员变量和继承方法的区别.....	36
2.3.2 内存中子类实例.....	39
2.3.3 父、子类的类变量.....	43

2.4	final 修饰符	44
2.4.1	final 修饰的变量	44
2.4.2	执行“宏替换”的变量	49
2.4.3	final 方法不能被重写	53
2.4.4	内部类中的局部变量	55
2.5	小结	58
第 3 课	常见 Java 集合的实现细节	59
3.1	Set 和 Map	60
3.1.1	Set 和 Map 的关系	60
3.1.2	HashMap 和 HashSet	65
3.1.3	TreeMap 和 TreeSet	75
3.2	Map 和 List	80
3.2.1	Map 的 values()方法	81
3.2.2	Map 和 List 的关系	87
3.3	ArrayList 和 LinkedList	88
3.3.1	Vector 和 ArrayList 的区别	89
3.3.2	ArrayList 和 LinkedList 的实现差异	92
3.3.3	ArrayList 和 LinkedList 的性能分析和适用场景	96
3.4	Iterator 迭代器	96
3.5	小结	100
第 4 课	Java 的内存回收	101
4.1	Java 引用的种类	102
4.1.1	对象在内存中状态	102
4.1.2	强引用	105
4.1.3	软引用	105
4.1.4	弱引用	108
4.1.5	虚引用	111
4.2	Java 的内存泄漏	112
4.3	垃圾回收机制	116
4.3.1	垃圾回收的基本算法	116
4.3.2	堆内存的分代回收	118

4.3.3	与垃圾回收的附加选项	119
4.3.4	常见垃圾回收器	120
4.4	内存管理的小技巧	123
4.4.1	尽量使用直接量	123
4.4.2	使用 StringBuilder 和 StringBuffer 进行字符串连接	123
4.4.3	尽早释放无用对象的引用	124
4.4.4	尽量少用静态变量	124
4.4.5	避免在经常调用的方法、循环中创建 Java 对象	125
4.4.6	缓存经常使用的对象	125
4.4.7	尽量不要使用 finalize 方法	126
4.4.8	考虑使用 SoftReference	126
4.5	小结	126
第 5 课	表达式中的陷阱	127
5.1	关于字符串的陷阱	128
5.1.1	JVM 对字符串的处理	128
5.1.2	不可变的字符串	131
5.1.3	字符串比较	133
5.2	表达式类型的陷阱	135
5.2.1	表达式类型的自动提升	135
5.2.2	复合赋值运算符的陷阱	136
5.3	输入法导致的陷阱	138
5.4	注释的字符必须合法	138
5.5	转义字符的陷阱	139
5.5.1	慎用字符的 Unicode 转义形式	139
5.5.2	中止行注释的转义字符	140
5.6	泛型可能引起的错误	141
5.6.1	原始类型变量的赋值	141
5.6.2	原始类型带来的擦除	143
5.6.3	创建泛型数组的陷阱	145
5.7	正则表达式的陷阱	147
5.8	多线程的陷阱	148
5.8.1	不要调用 run 方法	148

5.8.2	静态的同步方法	150
5.8.3	静态初始化块启动新线程执行初始化	152
5.8.4	注意多线程执行环境	157
5.9	小结	161
第 6 课	流程控制的陷阱	163
6.1	switch 语句陷阱	164
6.1.1	default 分支永远会执行吗	164
6.1.2	break 的重要性	165
6.1.3	switch 表达式的类型	167
6.2	标签引起的陷阱	168
6.3	if 语句的陷阱	169
6.3.1	else 隐含的条件	169
6.3.2	小心空语句	171
6.4	循环体的花括号	173
6.4.1	什么时候可以省略花括号	173
6.4.2	省略花括号的危险	174
6.5	for 循环的陷阱	175
6.5.1	分号惹的祸	175
6.5.2	小心循环计数器的值	178
6.5.3	浮点数作循环计数器	179
6.6	foreach 循环的循环计数器	181
6.7	小结	182
第 7 课	面向对象的陷阱	183
7.1	instanceof 运算符的陷阱	184
7.2	构造器的陷阱	188
7.2.1	构造器之前的 void	188
7.2.2	构造器创建对象吗	189
7.2.3	无限递归的构造器	194
7.3	持有当前类的实例	195
7.4	到底调用哪个重载的方法	196
7.5	方法重写的陷阱	199

7.5.1	重写 private 方法	199
7.5.2	重写其他访问权限的方法	200
7.6	非静态内部类的陷阱	201
7.6.1	非静态内部类的构造器	201
7.6.2	非静态内部类不能拥有静态成员	203
7.6.3	非静态内部类的子类	204
7.7	static 关键字	206
7.7.1	静态方法属于类	206
7.7.2	静态内部类的限制	207
7.8	native 方法的陷阱	208
7.9	小结	209
第 8 课	异常捕捉的陷阱	211
8.1	正确关闭资源的方式	212
8.2	finally 块的陷阱	215
8.2.1	finally 的执行规则	215
8.2.2	finally 块和方法返回值	217
8.3	catch 块的用法	219
8.3.1	catch 块的顺序	219
8.3.2	不要用 catch 代替流程控制	221
8.3.3	只能 catch 可能抛出的异常	221
8.3.4	实际的修复	225
8.4	继承得到的异常	227
8.5	小结	228
第 9 课	线性表	229
9.1	线性表概述	230
9.1.1	线性表的定义及逻辑结构	230
9.1.2	线性表的基本操作	231
9.2	顺序存储结构	231
9.3	链式存储结构	236
9.3.1	单链表上的基本运算	237
9.3.2	循环链表	243

9.3.3	双向链表	244
9.4	线性表的分析	251
9.4.1	线性表的实现分析	251
9.4.2	线性表的功能	251
9.5	小结	252
第 10 课	栈和队列	253
10.1	栈	254
10.1.1	栈的基本定义	254
10.1.2	栈的常用操作	254
10.1.3	栈的顺序存储结构及实现	255
10.1.4	栈的链式存储结构及实现	259
10.1.5	Java 集合中的栈	263
10.2	队列	263
10.2.1	队列的基本定义	263
10.2.2	队列的常用操作	264
10.2.3	队列的顺序存储结构及实现	264
10.2.4	循环队列	268
10.2.5	队列的链式存储结构及实现	272
10.2.6	Java 集合中的队列	275
10.3	双向队列	276
10.4	小结	278
第 11 课	树和二叉树	279
11.1	树的概述	280
11.1.1	树的定义和基本术语	280
11.1.2	树的基本操作	281
11.1.3	父节点表示法	282
11.1.4	子节点链表示法	286
11.2	二叉树	290
11.2.1	二叉树的定义和基本概念	290
11.2.2	二叉树的基本操作	292
11.2.3	二叉树的顺序存储	292
11.2.4	二叉树的二叉链表存储	296

11.2.5	二叉树的三叉链表存储	299
11.3	遍历二叉树	303
11.3.1	先序遍历	303
11.3.2	中序遍历	304
11.3.3	后序遍历	304
11.3.4	广度优先(按层)遍历	305
11.4	森林、树和二叉树的转换	306
11.4.1	森林、树和二叉树的转换	306
11.4.2	树的链表存储	307
11.5	哈夫曼树	307
11.5.1	哈夫曼树的定义和基本概念	307
11.5.2	创建哈夫曼树	308
11.5.3	哈夫曼编码	311
11.6	排序二叉树	312
11.7	红黑树	319
11.7.1	插入操作	321
11.7.2	删除操作	322
11.8	小结	332
第 12 课	常用的内部排序	333
12.1	排序的基本概念	334
12.1.1	排序概述	334
12.1.2	内部排序的分类	335
12.2	选择排序法	335
12.2.1	直接选择排序	335
12.2.2	堆排序	339
12.3	交换排序	343
12.3.1	冒泡排序	343
12.3.2	快速排序	345
12.4	插入排序	347
12.4.1	直接插入排序	347
12.4.2	折半插入排序	349
12.4.3	Shell 排序	351

12.5	归并排序	353
12.6	桶式排序	357
12.7	基数排序	359
12.8	小结	362
第 13 课 程序开发		363
13.1	扎实的基本功	364
13.1.1	快速的输入能力	364
13.1.2	编程实现能力	365
13.1.3	快速排错	366
13.2	程序开发之前	366
13.2.1	分析软件的组件模型	366
13.2.2	建立软件的数据模型	369
13.3	弄清程序的具体实现	370
13.3.1	各组件如何通信	370
13.3.2	人机交互的实现	372
13.3.3	复杂算法的分析	374
13.4	编写开发文档	377
13.4.1	绘制建模图、流程图	377
13.4.2	提供简要说明	378
13.4.3	编写伪码实现	379
13.5	编码实现和开发心态	379
13.5.1	开发是复杂的	379
13.5.2	开发过程是漫长的	380
13.6	小结	380
第 14 课 程序调试		381
14.1	程序的可调试性	382
14.1.1	增加注释	382
14.1.2	使用 log	382
14.2	程序调试的基本方法	390
14.2.1	借助编译器的代码审查	390
14.2.2	跟踪程序执行流程	392

14.2.3	断点调试	394
14.2.4	隔离调试	395
14.2.5	错误重现	397
14.3	记录常见错误	398
14.3.1	常见异常可能的错误原因	399
14.3.2	常见运行时异常可能的错误原因	400
14.4	程序调试的整体思路	402
14.4.1	分段调试	402
14.4.2	分模块调试	403
14.5	调试心态	403
14.5.1	谁都会出错	403
14.5.2	调试比写程序更费时	404
14.6	小结	404
第 15 课	使用 IDE 工具	405
15.1	何时开始利用 IDE 工具	406
15.2	IDE 工具概述	407
15.2.1	IDE 工具的基本功能	407
15.2.2	常见的 Java IDE 工具	409
15.3	项目管理	412
15.3.1	建立项目	412
15.3.2	自动编译	416
15.3.3	自动部署、运行	417
15.4	代码管理	418
15.4.1	向导式的代码生成	418
15.4.2	代码生成器	420
15.4.3	代码提示	421
15.4.4	自动代码补齐	422
15.4.5	实时错误提示	422
15.5	项目调试	423
15.5.1	设置断点	424
15.5.2	单步调试	426
15.5.3	步入、步出	426

15.6	团队协作功能	427
	作为版本控制工具的客户端	428
15.7	小结	431
第 16 课	软件测试	433
16.1	软件测试概述	434
16.1.1	软件测试的概念和目的	434
16.1.2	软件测试的分类	436
16.1.3	开发活动和测试活动	436
16.1.4	常见 Bug 管理工具	437
16.2	单元测试	438
16.2.1	单元测试概述	438
16.2.2	单元测试的逻辑覆盖	439
16.2.3	JUnit 介绍	442
16.2.4	JUnit 的用法	443
16.3	系统测试和自动化测试	448
16.3.1	系统测试概述	448
16.3.2	自动化测试	449
16.3.3	常见自动化测试工具	450
16.4	性能测试	451
16.4.1	性能测试概述	451
16.4.2	性能测试的相关概念	452
16.4.3	常见性能测试工具	453
16.5	小结	453

第 1 课

数组与内存控制

一家国际著名软件企业的面试。

“你的简历我看了，你会使用 Java？”面试官面无表情地问道。

“是的。”参加面试的人，成竹在胸地回答。

“那好，你给我叙述一下，在 Java 中，声明一个数组的过程中，是如何分配内存的。”

“……”

“那，Java 数组的初始化一共哪有几种方式，你能说一说吗？”

“……”

“那你知道基本类型数组和应用类型数组之间，在初始化时的内存分配机制有什么区别吗？”

“……”

过了一会儿之后，房间的门打开了，可怜的面试者，狼狈地走了出来。

离开的时候，他喃喃自语：“原来，小小的数组，也有这么多的知识。”

本课要点：

- Java 数组的基本语法；
- Java 数组的静态特性；
- Java 数组的内存分配机制；
- 初始化 Java 数组的两种方式；
- 初始化基本类型数组的内存分配；
- 初始化应用类型数组的内存分配；
- 数组引用变量和数组对象；
- 何时是数组引用变量，何时是数组对象；
- 数组元素等同于变量；
- 多维数组的内存分配。

Java 数组并不是什么很难的知识，如果单从用法的角度来看，数组的用法并不难，只是很多程序员虽然一直使用 Java 数组，但他们往往对 Java 数组的内存分配把握并不准确。本章正是为了弥补程序员的这部分基本功而做的深入探讨。

本课将会深入探讨 Java 数组的静态特征。使用 Java 数组之前必须先对数组对象进行初始化。当数组的所有元素都被分配了合适的内存空间，并指定了初始值时，数组初始化完成。程序以后将不能重新改变数组对象在内存中的位置和大小。从用法角度来看，数组元素相当于普通变量，程序既可把数组元素的值赋给普通变量，也可把普通变量的值赋给数组元素。

本课还将深入分析多维数组的实质，深入讲解多维数组和一维数组之间的关联，并通过程序示范如何将一维数组扩展成多维数组。

1.1 数组初始化

数组是大多数编程语言都提供一种的复合结构，如果程序需要多个类型相同的变量时，就可以考虑定义一个数组。Java 语言的数组变量是引用类型的变量，因此具有 Java 独有的特性。

1.1.1 Java 数组是静态的

Java 语言是典型的静态语言，因此 Java 的数组是静态的，即当数组被初始化之后，该数组的长度是不可变的。Java 程序中的数组必须经初始化才可使用。所谓初始化，就是为数组对象的元素分配内存空间，并为每个数组元素指定初始值。

数组的初始化有以下两种方式。

- 静态初始化：初始化时由程序员显式指定每个数组元素的初始值，由系统决定数组长度。
- 动态初始化：初始化时程序员只指定数组长度，由系统为数组元素分配初始值。

不管采用哪种方式初始化 Java 数组，一旦初始化完成，该数组的长度就不可改变，Java 语言允许通过数组的 length 属性来访问数组的长度。示例如下。

程序清单：codes\01\1.1\ArrayTest.java

```
public class ArrayTest
{
    public static void main(String[] args)
    {
        //采用静态初始化方式初始化第 1 个数组
        String[] books = new String[]
        {
            "疯狂 Java 讲义",
            "轻量级 Java EE 企业应用实战",
            "疯狂 Ajax 讲义",
            "疯狂 XML 讲义"
        };
        //采用静态初始化的简化形式初始化第 2 个数组
        String[] names =
        {
            "孙悟空",
            "猪八戒",
            "白骨精"
        };
    }
}
```