



疯狂程

Java讲义

精粹

(第2版)

李刚 编著

疯狂源自梦想

技术成就辉煌

疯狂源自梦想

技术成就辉煌





疯狂

Java讲义

精粹

(第2版)

李刚 编著



电子工业出版社
Publishing House of Electronics Industry
北京·BEIJING

内 容 简 介

本书是《疯狂 Java 讲义精粹》的第 2 版，本书相比《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，本书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。

本书尽量浅显、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。本书全面介绍了 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。

本书为打算认真掌握 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。本书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。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有，侵权必究。

图书在版编目 (CIP) 数据

疯狂 Java 讲义精粹 / 李刚编著. — 2 版. — 北京: 电子工业出版社, 2014.10
ISBN 978-7-121-24346-2

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 (2014) 第 213893 号

策划编辑: 张月萍

责任编辑: 葛 娜

印 刷: 北京京科印刷有限公司

装 订: 三河市鹏成印业有限公司

出版发行: 电子工业出版社

北京市海淀区万寿路 173 信箱 邮编 100036

开 本: 850×1168 1/16

印张: 28

字数: 1045 千字 彩插: 1

版 次: 2012 年 1 月第 1 版

2014 年 10 月第 2 版

印 次: 2014 年 10 月第 1 次印刷

印 数: 4000 册 定价: 59.90 元 (含光盘 1 张)

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话: (010) 88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线: (010) 88258888。



如何学习 Java

——谨以此文献给打算以编程为职业、并愿意为之疯狂的人

经常看到有些学生、求职者捧着一本类似 JBuilder 入门、Eclipse 指南之类的图书学习 Java，当他们学会了在这些工具中拖出窗体、安装按钮之后，就觉得自己掌握、甚至精通了 Java；又或是找来一本类似 JSP 动态网站编程之类的图书，学会使用 JSP 脚本编写一些页面后，就自我感觉掌握了 Java 开发。

还有一些学生、求职者听说 J2EE、Spring 或 EJB 很有前途，于是立即跑到书店或图书馆找来一本相关图书。希望立即学会它们，然后进入软件开发业、大显身手。

还有一些学生、求职者非常希望找到一本既速成、又大而全的图书，比如突击 J2EE 开发、一本书精通 J2EE 之类的图书（包括笔者曾出版的《轻量级 J2EE 企业应用实战》一书，据说销量不错），希望这样一本图书就可以打通自己的“任督二脉”，一跃成为 J2EE 开发高手。

也有些学生、求职者非常喜欢 J2EE 项目实战、项目大全之类的图书，他们的想法很单纯：我按照书上介绍，按图索骥、依葫芦画瓢，应该很快就可学会 J2EE，很快就能成为一个受人羡慕的 J2EE 程序员了。

.....

凡此种种，不一而足。但最后的结果往往是失败，因为这种学习没有积累、没有根基，学习过程中困难重重，每天都被一些相同、类似的问题所困扰，起初热情十足，经常上论坛询问，按别人的说法解决问题之后很高兴，既不知道为什么错？也不知道为什么对？只是盲目地抄袭别人的说法。最后的结果有两种：

- ① 久而久之，热情丧失，最后放弃学习。
- ② 大部分常见问题都问遍了，最后也可以从事一些重复性开发，但一旦遇到新问题，又将束手无策。

第二种情形在普通程序员中占了极大的比例，笔者多次听到、看到（在网络上）有些程序员抱怨：我做了 2 年多 Java 程序员了，工资还是 3000 多点。偶尔笔者会与他们聊聊工作相关内容，他们会告诉笔者：我也用 Spring 了啊，我也用 EJB 了啊……他们感到非常不平衡，为什么我的工资这么低？其实笔者很想告诉他们：你们太浮躁了！你们确实是用了 Spring、Hibernate 又或是 EJB，但你们未想过为什么要用这些技术？用这些技术有什么好处？如果不用这些技术行不行？

很多时候，我们的程序员把 Java 当成一种脚本，而不是一门面向对象的语言。他们习惯了在 JSP 脚本中使用 Java，但从不去想 JSP 如何运行，Web 服务器里的网络通信、多线程机制，为何一个 JSP 页面能同时向多个请求者提供服务？更不会想如何开发 Web 服务器；他们像代码机器一样编写 Spring Bean 代码，但从不去理解 Spring 容器的作用，更不会想如何开发 Spring 容器。

有时候，笔者的学生在编写五子棋、梭哈等作业感到困难时，会向他们的大学师兄、朋友求救，这些程序员告诉他：不用写了，网上有下载的！听到这样回答，笔者不禁感到哑然：网上还有 Windows 下载呢！网上下载和自己编写是两码事。偶尔，笔者会怀念以前黑色屏幕、绿荧荧字符时代，那时候程序员很单纯：当我们想偷懒时，习惯思维是写一个小工具；现在程序员很聪明：当他们想偷懒时，习惯思维是从网上下一个小工具。但是，谁更幸福？

当笔者的学生把他们完成的小作业放上互联网之后，然后就有许多人称他们为“高手”！这个称呼却让他们万分惭愧；惭愧之余，他们也感到万分欣喜，非常有成就感，这就是编程的快乐。编程的过程，与寻宝的过程完全一样：历经辛苦，终于找到心中的梦想，这是何等的快乐？

如果真的打算将编程当成职业，那就不应该如此浮躁，而是应该扎扎实实先学好 Java 语言，然后按 Java 本身的学习规律，踏踏实实一步一个脚印地学习，把基本功练扎实了才可获得更大的成功。

实际情况是，有多少程序员真正掌握了 Java 的面向对象？真正掌握了 Java 的多线程、网络通信、反射等内容？有多少 Java 程序员真正理解了类初始化时内存运行过程？又有多少程序员理解 Java 对象从创建到消失的全部细节？有几个程序员真正独立地编写过五子棋、梭哈、桌面弹球这种小游戏？又有几个 Java 程序员敢说：我可以开发 Struts？我可以开发 Spring？我可以开发 Tomcat？很多人又会说：这些都是许多人开发出来的！实际情况是：许多开源框架的核心最初完全是由一个人开发的。现在这些优秀程序已经出来了！你，是否深入研究过它们，是否深入掌握了它们？

如果要真正掌握 Java，包括后期的 Java EE 相关技术（例如 Struts、Spring、Hibernate 和 EJB 等），一定要记住笔者的话：绝不要从 IDE（如 JBuilder、Eclipse 和 NetBeans）工具开始学习！IDE 工具的功能很强大，初学者学起来也很容易上手，但也非常危险：因为 IDE 工具已经为我们做了许多事情，而软件开发者要全部了解软件开发的全部步骤。



2011 年 12 月 17 日



光盘说明

一、光盘内容



本光盘是《疯狂 Java 讲义精粹》（第 2 版）一书的配书光盘，书中的代码按章、节存放，即第 2 章第 2 节所使用的代码放在 codes 文件夹的 02\2.2 文件夹下，依此类推。

另：书中每份源代码也给出了与光盘源文件的对应关系，方便读者查找。



本光盘 codes 目录下有 13 个文件夹，其内容和含义说明如下：

(1) 01~13 文件夹名对应于《疯狂 Java 讲义精粹》（第 2 版）中的章名，即第 3 章所使用的代码放在 codes 文件夹的 03 文件夹下，依此类推。

(2) 本书所有代码都是 IDE 工具无关的程序，读者既可以在命令行窗口直接编译、运行这些代码，也可以导入 Eclipse、NetBeans 等 IDE 工具来运行它们。



本光盘根目录下提供了一个“Java 设计模式（疯狂 Java 联盟版）.chm”文件，这是一份关于设计模式的电子教材，由疯狂 Java 联盟的杨恩雄亲自编写、制作，他同意广大读者阅读、传播这份开源文档。



本光盘根目录下包含一个“project_codes”文件夹，该文件夹里包含了疯狂 Java 联盟的杨恩雄编写的《疯狂 Java 实战演义》一书的光盘内容，该光盘中包含了大量实战性很强的项目，这些项目基本覆盖了《疯狂 Java 讲义精粹》（第 2 版）课后习题的要求，读者可以参考相关案例来完成《疯狂 Java 讲义精粹》（第 2 版）的课后习题。



本光盘根目录下包含一个“课件”文件夹，该文件夹里包含了《疯狂 Java 讲义精粹》（第 2 版）各章配套的授课 PPT 教案，各高校教师、学生可在此基础上自由修改、传播，但请保留署名。



本光盘根目录下包含一个“视频”文件夹，该文件夹里包含了 14 个小时的基础授课视频。



本光盘根目录下包含一个“疯狂 Java 面试题（疯狂 Java 讲义精粹附赠）.pdf”文件，该文件涵盖了大量常见的 Java 面试题及解答。

二、运行环境



本书中的程序在以下环境调试通过：

(1) 安装 jdk-8u5-windows-x64.exe，安装完成后，添加 CLASSPATH 环境变量，该环境变量的值为：;%JAVA_HOME%/lib/tools.jar;%JAVA_HOME%/lib/dt.jar。如果为了可以编译和运行 Java 程序，还应该在 PATH 环境变量中增加 %JAVA_HOME%/bin。其中 JAVA_HOME 代表 JDK（不是 JRE）的安装路径。

(2) 安装上面工具的详细步骤，请参考本书的第 1 章。

三、注意事项

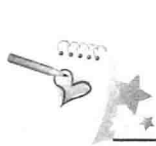
在使用本光盘中的程序时，请将程序拷贝到硬盘上，并去除文件的只读属性。

四、技术支持

如果您使用本光盘中遇到不懂的技术问题，可以登录如下网站与作者联系：

<http://www.crazyit.org>

我在 Java 编程教学中把《疯狂 Java 讲义》列为重要的中文参考资料。它覆盖了“够用”的 Java 语言和技术，作者有实际的编程和教学经验，也尽力把相关问题讲解明白、分析清楚，这在同类书籍中是比较难得的。



前 言

2014 年 3 月 18 日，Oracle 发布了 Java 8 正式版。Java 8 是自 Java 5 以来最重要的版本更新，Java 8 引入了大量新特性——重新设计的接口语法、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 等，这些新特性进一步增强了 Java 语言的功能。

为了向广大工作者、学习者介绍最新、最前沿的 Java 知识，在 Java 8 正式发布之前，笔者已经深入研究过 Java 8 绝大部分可能新增的功能；当 Java 8 正式发布之后，笔者在第一时间开始了《疯狂 Java 讲义精粹版》的升级：使用 Java 8 改写了全书所有程序，全面介绍了 Java 8 的各种新特性。

在以“疯狂 Java 体系”图书为教材的疯狂软件教育中心 (www.fkjava.org)，经常有学生询问：为什么叫疯狂 Java 这个名字？也有一些读者通过网络、邮件来询问这个问题。其实这个问题的答案可以在本书第 1 版的前言中找到。疯狂的本质是一种“享受编程”的状态。在一些不了解编程的人看来：编程的人总面对着电脑，在键盘上敲打，这种生活实在太枯燥了，但实际上是因为他们并未真正了解编程，并未真正走进编程。在外人眼中：程序员不过是在敲打键盘；但在程序员心中：程序员敲出的每个字符，都是程序的一部分。

程序是什么呢？程序是对现实世界的数字化模拟。开发一个程序，实际是创造一个或大或小的“模拟世界”。在这个过程中，程序员享受着“创造”的乐趣，程序员沉醉在他所创造的“模拟世界”里：疯狂地设计、疯狂地编码实现。实现过程不断地遇到问题，然后解决它；不断地发现程序的缺陷，然后重新设计、修复它——这个过程本身就是一种享受。一旦完全沉浸到编程世界里，程序员是“物我两忘”的，眼中看到的、心中想到的，只有他正在创造的“模拟世界”。

在学会享受编程之前，编程学习者都应该采用“案例驱动”的方式，学习者需要明白程序的作用是：解决问题——如果你的程序不能解决你自己的问题，如何期望你的程序去解决别人的问题呢？那你的程序的价值何在？——知道一个知识点能解决什么问题，才去学这个知识点，而不是盲目学习！因此本书强调编程实战，强调以项目激发编程兴趣。

仅仅是看完这本书，你不会成为高手！在编程领域里，没有所谓的“武林秘笈”，再好的书一定要配合大量练习，否则书里的知识依然属于作者，而读者则仿佛身入宝山而一无所获的笨汉。本书配合了大量高强度的练习，希望读者强迫自己去完成这些项目。这些习题的答案可以参考本书所附光盘中《疯

狂 Java 实战演义》的配套代码。如果需要获得编程思路和交流，可以登录 <http://www.crazyit.org> 与广大读者和笔者交流。

本书前两版面市的近 6 年时间里，无数读者已经通过《疯狂 Java 讲义》步入了 Java 编程世界，而且第 2 版的年销量比第 1 版的年销量大幅提升，这说明“青山遮不住”，优秀的作品，经过时间的沉淀，往往历久弥新。

广大读者对疯狂 Java 的肯定、赞誉既让笔者十分欣慰，也鞭策笔者以更高的热情、更严谨的方式创作图书。时至今日，每次笔者创作或升级图书时，总有一种诚惶诚恐、如履薄冰的感觉，惟恐辜负广大读者的厚爱。

笔者非常欢迎所有热爱编程、愿意推动中国软件业的学习者、工作者对本书提出宝贵的意见，非常乐意与大家交流。中国软件业还处于发展阶段，所有热爱编程、愿意推动中国软件业的人应该联合起来，共同为中国软件行业贡献自己的绵薄之力。

本书有什么特点



本书是《疯狂 Java 讲义》的精粹版，本书的内容完全取自《疯狂 Java 讲义》，只是对书中部分内容进行了简化，力求用更通俗、更浅显的方式进行讲解，也删减了原书部分过于高级的知识。

因此，本书具有如下三个特点。

1. 阐述透彻、原理清晰

本书并不是简单地罗列 Java 语法规则，而是尽量从语法设计者的角度向读者解释每个语法规则的作用、缘由；本书力求从运行机制来解释代码的运行过程，从内存分配的细节上剖析程序的运行细节。阅读本书不仅要求读者知道怎么做，而且要求读者能理解“为什么这么做”。

2. 再现李刚老师课堂氛围

本书的内容是笔者 8 年多授课经历的总结，知识体系取自疯狂 Java 实战的课程体系。本书力求再现笔者的课堂氛围：以浅显比喻代替乏味的讲解，以疯狂实战代替空洞的理论。书中包含了大量“注意”、“学生提问”部分，这些正是几千个 Java 学员所犯错误的汇总。

3. 注释详细，轻松上手

为了降低读者阅读的难度，书中代码的注释非常详细，几乎每两行代码就有一行注释。不仅如此，本书甚至还把一些简单理论作为注释穿插到代码中，力求让读者能轻松上手。

本书所有程序中的关键代码以粗体字标出，也是为了帮助读者能迅速找到这些程序的关键点。

本书写给谁看














本书为打算认真掌握 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。本书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。






2014-08-15

目 录

CONTENTS

第 1 章 Java 语言概述与开发环境	1	为什么要学习查看 API 文档的方法?	21
1.1 Java 语言的发展简史	2	2.2 标识符和关键字	24
1.2 Java 程序运行机制	4	2.2.1 分隔符	24
1.2.1 高级语言的运行机制	4	2.2.2 标识符规则	25
1.2.2 Java 程序的运行机制和 JVM	4	2.2.3 Java 关键字	26
1.3 开发 Java 的准备	5	2.3 数据类型分类	26
1.3.1 下载和安装 Java 8 的 JDK	5	什么是变量? 变量有什么用?	26
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	2.4 基本数据类型	27
1.3.2 设置 PATH 环境变量	8	2.4.1 整型	27
为什么不安装公共 JRE 呢?	7	2.4.2 字符型	29
为什么选择用户变量? 用户变量与系统变量有什么区别?	9	什么是字符集?	29
1.4 第一个 Java 程序	9	2.4.3 浮点型	31
1.4.1 编辑 Java 源代码	9	2.4.4 数值中使用下划线分隔	32
1.4.2 编译 Java 程序	10	2.4.5 布尔型	32
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	2.5 基本类型的类型转换	33
1.4.3 运行 Java 程序	11	2.5.1 自动类型转换	33
1.4.4 根据 CLASSPATH 环境变量定位类	11	2.5.2 强制类型转换	34
1.5 Java 程序的基本规则	12	2.5.3 表达式类型的自动提升	35
1.5.1 Java 程序的组织形式	12	2.6 直接量	36
1.5.2 Java 源文件的命名规则	13	2.6.1 直接量的类型	36
1.5.3 初学者容易犯的错误	13	2.6.2 直接量的赋值	37
1.6 何时开始使用 IDE 工具	15	2.7 运算符	38
我想学习 Java 编程, 到底是学习 Eclipse 好, 还是学习 NetBeans 好呢?	16	2.7.1 算术运算符	38
1.7 本章小结	16	2.7.2 赋值运算符	40
本章练习	16	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 2 章 数据类型和运算符	17	第 3 章 流程控制与数组	48
2.1 注释	18	3.1 顺序结构	49
2.1.1 单行注释和多行注释	18	3.2 分支结构	49
2.1.2 文档注释	19	3.2.1 if 条件语句	49
API 文档是什么?	19	3.2.2 增强后的 switch 分支语句	53

3.3	循环结构	54	4.3.1	成员变量和局部变量	90
3.3.1	while 循环语句	55	4.3.2	成员变量的初始化和内存中的运行 机制	94
3.3.2	do while 循环语句	56	4.3.3	局部变量的初始化和内存中的运行 机制	95
3.3.3	for 循环	57	4.3.4	变量的使用规则	96
3.3.4	嵌套循环	59	4.4	隐藏和封装	97
3.4	控制循环结构	60	4.4.1	理解封装	97
3.4.1	使用 break 结束循环	60	4.4.2	使用访问控制符	97
3.4.2	使用 continue 忽略本次循环剩下语句	61	4.4.3	package、import 和 import static	100
3.4.3	使用 return 结束方法	62	4.4.4	Java 的常用包	104
3.5	数组类型	63	4.5	深入构造器	105
3.5.1	理解数组：数组也是一种类型	63	4.5.1	使用构造器执行初始化	105
	int[] 是一种类型吗？怎么使用这种 类型呢？	63		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
3.5.2	定义数组	63	4.5.2	构造器重载	106
3.5.3	数组的初始化	64		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 不就可以了吗？	107
	能不能只分配内存空间，不赋初始 值呢？	64	4.6	类的继承	108
3.5.4	使用数组	65	4.6.1	继承的特点	108
	为什么要我记住这些异常信息？	66	4.6.2	重写父类的方法	109
3.5.5	foreach 循环	66	4.6.3	super 限定	110
3.6	深入数组	68	4.6.4	调用父类构造器	112
3.6.1	没有多维数组	68		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
	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	4.7	多态	115
3.6.2	Java 8 增强的工具类：Arrays	70	4.7.1	多态性	115
3.7	本章小结	73	4.7.2	引用变量的强制类型转换	116
	本章练习	73	4.7.3	instanceof 运算符	117
第 4 章	面向对象（上）	74	4.8	初始化块	118
4.1	类和对象	75	4.8.1	使用初始化块	118
4.1.1	定义类	75	4.8.2	初始化块和构造器	120
	构造器不是没有返回值吗？为什 么不能用 void 声明呢？	77	4.8.3	静态初始化块	120
4.1.2	对象的产生和使用	77	4.9	本章小结	123
4.1.3	对象、引用和指针	78		本章练习	123
4.1.4	对象的 this 引用	79	第 5 章	面向对象（下）	124
4.2	方法详解	83	5.1	Java 8 增强的包装类	125
4.2.1	方法的所属性	83		Java 为什么要对这些数据进行缓 存呢？	128
4.2.2	方法的参数传递机制	83	5.2	处理对象	129
4.2.3	形参个数可变的方法	87	5.2.1	打印对象和 toString 方法	129
4.2.4	递归方法	88	5.2.2	== 和 equals 方法	130
4.2.5	方法重载	89		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
	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			

5.3	类成员	134	5.9.4	实现接口的枚举类	177
5.3.1	理解类成员	134		枚举类不是用 final 修饰了吗? 怎 么还能派生子类呢?	178
5.3.2	单例 (Singleton) 类	135	5.9.5	包含抽象方法的枚举类	178
5.4	final 修饰符	136	5.10	修饰符的适用范围	179
5.4.1	final 成员变量	136	5.11	本章小结	180
5.4.2	final 局部变量	138		本章练习	180
5.4.3	final 修饰基本类型变量和引用类型 变量的区别	139	第 6 章 Java 基础类库		
5.4.4	可执行“宏替换”的 final 变量	139	181		
5.4.5	final 方法	141	6.1	与用户互动	182
5.4.6	final 类	142	6.1.1	运行 Java 程序的参数	182
5.5	抽象类	142	6.1.2	使用 Scanner 获取键盘输入	183
5.5.1	抽象方法和抽象类	142	6.2	系统相关	185
5.5.2	抽象类的作用	145	6.2.1	System 类	185
5.6	Java 8 改进的接口	146	6.2.2	Runtime 类	187
5.6.1	接口的概念	146	6.3	常用类	188
5.6.2	Java 8 中接口的定义	147	6.3.1	Object 类	188
5.6.3	接口的继承	149	6.3.2	Objects 类	189
5.6.4	使用接口	149	6.3.3	String、StringBuffer 和 StringBuilder 类	190
5.6.5	接口和抽象类	151	6.3.4	Math 类	193
5.7	内部类	152	6.3.5	ThreadLocalRandom 与 Random	195
5.7.1	非静态内部类	152	6.3.6	BigDecimal 类	196
	非静态内部类对象和外部类对象 的关系是怎样的?	155	6.4	Java 8 的日期、时间类	199
5.7.2	静态内部类	156	6.4.1	Date 类	199
	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	6.4.2	Calendar 类	199
	接口里是否能定义内部接口?	158	6.4.3	Java 8 新增的日期、时间包	202
5.7.3	使用内部类	158	6.5	Java 8 新增的日期、时间格式器	204
	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	6.5.1	使用 DateTimeFormatter 完成格式化	205
5.7.4	局部内部类	160	6.5.2	使用 DateTimeFormatter 解析字符串	206
5.7.5	Java 8 改进的匿名内部类	161	6.6	本章小结	206
5.8	Java 8 新增的 Lambda 表达式	164		本章练习	206
5.8.1	Lambda 表达式入门	164	第 7 章 Java 集合		
5.8.2	Lambda 表达式与函数式接口	166	207		
5.8.3	方法引用与构造器引用	168	7.1	Java 集合概述	208
5.8.4	Lambda 表达式与匿名内部类的联系 和区别	171	7.2	Collection 和 Iterator 接口	209
5.8.5	使用 Lambda 表达式调用 Arrays 的 类方法	172	7.2.1	使用 Lambda 表达式遍历集合	211
5.9	枚举类	172	7.2.2	使用 Java 8 增强的 Iterator 遍历集合 元素	211
5.9.1	手动实现枚举类	173	7.2.3	使用 Lambda 表达式遍历 Iterator	213
5.9.2	枚举类入门	173	7.2.4	使用 foreach 循环遍历集合元素	213
5.9.3	枚举类的成员变量、方法和构造器	175	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 LinkedHashMap 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	12.2.2	实现 Runnable 接口创建线程类	351
10.2.1	使用@Retention	304	12.2.3	使用 Callable 和 Future 创建线程	352
10.2.2	使用@Target	305	12.2.4	创建线程的三种方式对比	354
10.2.3	使用@Documented	305	12.3	线程的生命周期	354
10.2.4	使用@Inherited	306	12.3.1	新建和就绪状态	354
10.3	自定义 Annotation	307	12.3.2	运行和阻塞状态	356
10.3.1	定义 Annotation	307	12.3.3	线程死亡	357
10.3.2	提取 Annotation 信息	308	12.4	控制线程	358
10.3.3	使用 Annotation 的示例	310	12.4.1	join 线程	358
10.3.4	Java 8 新增的重复注解	314	12.4.2	后台线程	359
10.3.5	Java 8 新增的 Type Annotation	316	12.4.3	线程睡眠: sleep	360
10.4	编译时处理 Annotation	317	12.4.4	线程让步: yield	360
10.5	本章小结	320	12.4.5	改变线程优先级	362
第 11 章 输入/输出			321		
11.1	File 类	322	12.5	线程同步	363
11.1.1	访问文件和目录	322	12.5.1	线程安全问题	363
11.1.2	文件过滤器	324	12.5.2	同步代码块	365
11.2	理解 Java 的 IO 流	324	12.5.3	同步方法	366
11.2.1	流的分类	325	12.5.4	释放同步监视器的锁定	368
11.2.2	流的概念模型	326	12.5.5	同步锁 (Lock)	369
11.3	字节流和字符流	327	12.5.6	死锁	371
11.3.1	InputStream 和 Reader	327	12.6	线程通信	372
11.3.2	OutputStream 和 Writer	329	12.6.1	传统的线程通信	372
11.4	输入/输出流体系	330	12.6.2	使用 Condition 控制线程通信	376
11.4.1	处理流的用法	330	12.6.3	使用阻塞队列 (BlockingQueue) 控制线程通信	378
11.4.2	输入/输出流体系	331	12.7	线程池	380
11.4.3	转换流	333	12.7.1	Java 8 改进的线程池	381
 11.4.3	怎么没有把字符流转换成字节流 的转换流呢?	334	12.7.2	Java 8 增强的 ForkJoinPool	382
11.4.4	推回输入流	334	12.8	线程相关类	386
11.5	重定向标准输入/输出	336	12.8.1	ThreadLocal 类	386
11.6	RandomAccessFile	337	12.8.2	包装线程不安全的集合	387
11.7	NIO.2	340	12.8.3	线程安全的集合类	388
11.7.1	Path、Paths 和 Files 核心 API	341	12.9	本章小结	389
11.7.2	使用 FileVisitor 遍历文件和目录	342	本章练习		389
11.7.3	使用 WatchService 监控文件变化	343	第 13 章 网络编程		
11.7.4	访问文件属性	344	390		
11.8	本章小结	346	13.1	网络编程的基础知识	391
本章练习		346	13.1.1	网络基础知识	391
第 12 章 多线程			347		
12.1	线程概述	348	13.1.2	IP 地址和端口号	392
12.1.1	线程和进程	348	13.2	Java 的基本网络支持	393
12.1.2	多线程的优势	349	13.2.1	使用 InetAddress	393
12.2	线程的创建和启动	349	13.2.2	使用 URLDecoder 和 URLEncoder	393
12.2.1	继承 Thread 类创建线程类	350	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?

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

Java 语言基础与开发环境

本章要点

- ✎ Java 语言的发展简史
- ✎ 编译型语言和解释型语言
- ✎ Java 语言的编译、解释运行机制
- ✎ 通过 JVM 实现跨平台
- ✎ 安装 JDK
- ✎ 设置 PATH 环境变量
- ✎ 编写、运行 Java 程序
- ✎ Java 程序的组织形式
- ✎ Java 程序的命名规则
- ✎ 初学者易犯的错误

Java 语言历时近二十年,已发展成为人类计算机史上影响深远的编程语言,从某种程度上来看,它甚至超出了编程语言的范畴,成为一种开发平台,一种开发规范。更甚至于:Java 已成为一种信仰,Java 语言所崇尚的开源、自由等精神,吸引了全世界无数优秀的程序员。事实是,从人类有史以来,从来没有一门编程语言能吸引这么多的程序员,也没有一门编程语言能衍生出如此之多的开源框架。

Java 语言是一门非常纯粹的面向对象编程语言,它吸收了 C++ 语言的各种优点,又摒弃了 C++ 里难以理解的多继承、指针等概念,因此 Java 语言具有功能强大和简单易用两个特征。Java 语言作为静态面向对象编程语言的代表,极好地实现了面向对象理论,允许程序员以优雅的思维方式进行复杂的编程开发。

不仅如此,Java 语言相关的 Java EE 规范里包含了时下最流行的各种软件工程理念,各种先进的设计思想总能在 Java EE 规范、平台以及相关框架里找到相应实现。从某种程度上来看,学精了 Java 语言的相关方面,相当于系统地学习了软件开发相关知识,而不是仅仅学完了一门编程语言。

时至今日,大部分银行、电信、证券、电子商务、电子政务等系统或者已经采用 Java EE 平台构建,或者正在逐渐过渡到采用 Java EE 平台来构建,Java EE 规范是目前最成熟的,也是应用最广的企业级应用开发规范。

1.1 Java 语言的发展简史

Java 语言的诞生具有一定的戏剧性,它并不是经过精心策划、制作,最后产生的划时代产品,从某个角度来看,Java 语言的诞生完全是一种误会。

1990 年年末,Sun 公司预料嵌入式系统将在未来家用电器领域大显身手。于是 Sun 公司成立了一个由 James Gosling 领导的“Green 计划”,准备为下一代智能家电(如电视机、微波炉、电话)编写一个通用控制系统。

该团队最初考虑使用 C++ 语言,但是很多成员包括 Sun 的首席科学家 Bill Joy,发现 C++ 和可用的 API 在某些方面存在很大问题。而且工作小组使用的是嵌入式平台,可用的系统资源极其有限。并且很多成员都发现 C++ 太复杂,以致很多开发者经常错误使用。而且 C++ 缺少垃圾回收系统、可移植性、分布式和多线程等功能。

根据可用的资金,Bill Joy 决定开发一种新语言,他提议在 C++ 的基础上,开发一种面向对象的环境。于是,Gosling 试图通过修改和扩展 C++ 的功能来满足这个要求,但是后来他放弃了。他决定创造一种全新的语言:Oak。

到了 1992 年的夏天,Green 计划已经完成了新平台的部分功能,包括 Green 操作系统、Oak 的程序设计语言、类库等。同年 11 月,Green 计划被转化成“FirstPerson 有限公司”,一个 Sun 公司的全资子公司。

FirstPerson 团队致力于创建一种高度互动的设备。当时代华纳公司发布了一个关于电视机顶盒的征求提议书时,FirstPerson 改变了他们的目标,作为对征求提议书的响应,提出了一个机顶盒平台的提议。但有线电视业界觉得 FirstPerson 的平台给予用户过多的控制权,因此 FirstPerson 的投标败给了 SGI。同时,与 3DO 公司的另外一笔关于机顶盒的交易也没有成功。此时,可怜的 Green 项目几乎接近夭折,甚至 Green 项目组的一半成员也被调到了其他项目组。

正如中国古代的寓言所言:塞翁失马,焉知非福?如果 Green 项目在机顶盒平台投标成功,也许就不会诞生 Java 这门伟大的语言了。

1994 年夏天,互联网和浏览器的出现不仅给广大互联网的用户带来了福音,也给 Oak 语言带来了新的生机。Gosling 立即意识到,这是一个机会,于是对 Oak 进行了小规模改造,到了 1994 年秋,小组中的 Naughton 和 Jonathan Payne 完成了第一个 Java 语言的网页浏览器:WebRunner。Sun 公司实验室主任 Bert Sutherland 和技术总监 Eric Schmidt 观看了该浏览器的演示,对该浏览器的效果给予了高度评价。当时 Oak 这个商标已被别人注册,于是只得将 Oak 更名为 Java。

Sun 公司在 1995 年年初发布了 Java 语言,Sun 公司直接把 Java 放到互联网上,免费给大家使用。甚至连源代码也不保密,也放在互联网上向所有人公开。

几个月后,让所有人都大吃一惊的事情发生了:Java 成了互联网上最热门的宝贝。竟然有 10 万多人访问了 Sun 公司的网页,下载了 Java 语言。然后,互联网上立即就有数不清的 Java 小程序(也就是 Applet),演示着各种小动画、小游戏等。

Java 语言终于扬眉吐气了,成为了一种广为人知的编程语言。