

Java

面向对象编程

(第2版)

Helping readers to master OOP(Object-Oriented Programming)
with Java and get deep understanding of how the language works

孙卫琴 编著



知名IT技术作家
孙卫琴老师
倾注六年心血
升级打造
业界经典畅销
10年图书
以行云流水般的
语言诠释Java
编程艺术
基于最新Java技术
理论紧密结合实际
可作为开发指南
高校教材和认证
辅导教材



中国工信出版集团



电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
<http://www.phei.com.cn>

Java

面向对象编程

(第2版)

孙卫琴

编著



电子工业出版社

Publishing House of Electronics Industry

北京·BEIJING

内 容 简 介

本书采用由浅入深、与实际应用紧密结合的方式,利用大量经典的实例,详细讲解 Java 面向对象的编程思想、编程语法和设计模式,介绍常见 Java 类库的用法,总结优化 Java 编程的各种宝贵经验,深入阐述 Java 虚拟机执行 Java 程序的原理。本书的实例均基于最新的 JDK8 版本。本书内容包括:面向对象开发方法概述、第一个 Java 应用、数据类型和变量、操作符、流程控制、继承、Java 语言中的修饰符、接口、异常处理、类的生命周期、对象的生命周期、内部类、多线程、数组、Java 集合、泛型、Lambda 表达式、输入与输出(I/O)、图形用户界面、常用 Swing 组件、Java 常用类和 Annotation 注解。本书的最大特色是以 6 条主线贯穿全书:面向对象编程思想、Java 语言的语法、Java 虚拟机执行 Java 程序的原理、在实际项目中的运用、设计模式和性能优化技巧。另外,本书还贯穿了 Oracle 公司的 OCJP(Oracle Certified Java Programmer)认证的考试要点。

书中实例源文件及思考题答案的下载网址为: www.javathinker.net/download.htm。

本书适用于所有 Java 编程人员,包括 Java 初学者及资深 Java 开发人员。本书还可作为高校的 Java 教材,以及企业 Java 培训教材,也可作为 Oracle 公司的 OCJP 认证的辅导教材。

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

Java 面向对象编程 / 孙卫琴编著. -- 2 版. -- 北京: 电子工业出版社, 2017.1

ISBN 978-7-121-30314-2

I. ①J… II. ①孙… III. ①JAVA 语言—程序设计 IV. ①TP312.8

中国版本图书馆 CIP 数据核字(2016)第 270355 号

责任编辑: 姜 伟

特约编辑: 刘红涛

印 刷: 三河市华成印务有限公司

装 订: 三河市华成印务有限公司

出版发行: 电子工业出版社

北京海淀区万寿路 173 信箱 邮编: 100036

经 销: 各地新华书店

开 本: 787×1092 1/16 印张: 47 字数: 1203.2 千字

版 次: 2017 年 1 月第 1 版

印 次: 2017 年 1 月第 1 次印刷

定 价: 89.00 元

参与本书编写的人员有: 孙卫琴、张雷、许亮思、张宇客、孟祥、曹文伟、李红军、王坤、曹雅洁、李洪成。

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, 88258888。

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

本书咨询联系方式: (010) 88254161~88254167 转 1897。

Java 自 1996 年正式发布以来,经历了初生、成长和壮大阶段,现在已经成为 IT 领域里的主流编程语言。Java 起源于 Sun 公司的一个名为“Green”的项目,目的是开发嵌入家用电器的分布式软件系统,使电器更加智能化。图 P-1 为参与 Green 项目的开发人员。Green 项目一开始准备采用 C++ 语言,但是考虑到 C++ 语言太复杂,而且安全性差,于是决定基于 C++ 语言开发一种新的 Oak 语言(即 Java 的前身)。



Java之父: James Gosling

图 P-1 参与 Green 项目的开发人员

Oak 是一种适用于网络编程的精巧而安全的语言,它保留了许多 C++ 语言的语法,但去除了明确的资源引用、指针算法与操作符重载等潜在的危险特性。并且 Oak 语言具有与硬件无关的特性,制造商只需更改芯片,就可以将烤面包机上的程序代码移植到微波炉或其他电器上,而不必改变软件,这就大大降低了开发成本。

当 Oak 语言成熟时,Internet 也在全球迅速发展。Sun 公司的开发小组认识到 Oak 非常适合 Internet 编程。1994 年,他们完成了一个用 Oak 语言编写的早期的 Web 浏览器,称为 WebRunner,后改名为 HotJava,展示了 Oak 作为 Internet 开发工具的能力。

1995 年,Oak 语言更名为 Java 语言(以下简称为 Java)。Java 的取名有一个趣闻:据说有一天,几位 Java 成员组的会员正在讨论给这个新的语言取什么名字,当时他们正在咖啡馆喝着 Java(爪哇)咖啡。有一个人灵机一动,说就叫 Java,并得到了其他人的赞赏。于是,Java 这个名字就这样传开了。

1996 年,Sun 公司发布 JDK1.0,计算机产业的各大公司(包括 IBM、Apple、DEC、Adobe、Silicon Graphics、HP、Oracel、Toshiba 和 Microsoft 等)相继从 Sun 公司购买了 Java 技术许可证,开发相应的产品。

1998 年,Sun 公司发布了 JDK1.2(从这个版本开始的 Java 技术都称为 Java2)。Java2 不仅兼容于智能卡和小型消费类设备,还兼容于大型服务器系统,它使软件开发商、服务提供商和设备制造商更加容易抢占市场。这一开发工具极大地简化了编程人员编制企业级 Web 应用的工作,把“一次编程,到处使用”的诺言应用到服务器领域。

1999 年,Sun 公司把 Java2 技术分成 J2SE、J2EE 和 J2ME。其中 J2SE 就是指从 1.2 版本开始的 JDK,它为创建和运行 Java 程序提供了最基本的环境。J2EE 和 J2ME 建立在 J2SE 的基础上,J2EE 为分布式的企业应用提供开发和运行环境,J2ME 为嵌入式应用(比如运行在手机里的 Java 程序)提供开发和运行环境。

Java 的公用规范(Publicly Available Specification, PAS)在 1997 年被国际标准化组织(ISO)认定,这是 ISO 第一次破例接受一个具有商业色彩的公司作为公用规范 PAS 的提交者。

Foreword

2000 年和 2002 年, Sun 公司分别发布了 JDK1.3 和 JDK1.4, 它们在原先版本的基础上做了一些改进, 扩展了标准雷库, 提高了系统性能, 还修正了一些 Bug。

2004 年, Sun 公司发布了 JDK5.0, 这一版本做了重大的改进, 增加了泛型机制、自动装箱/拆箱和 foreach 语句等。这一版本原称为 JDK1.5, 在 2004 年的 SunOne 会议后, 更名为 JDK5.0。

2006 年, Sun 公司发布了 JDK6, 该版本没有在语言方面进行大的改进, 主要是改进了运行性能, 并增强了类库功能。

随着以操纵海量数据为主的软件系统越来越依赖商业硬件, 而不是专用服务器, 以 Solaris 操作系统为主打产品的 Sun 公司变得越来越不景气, 最终于 2009 年被 Oracle 公司收购。Java 的发展停滞了很长一段时间。直到 2011 年, Oracle 公司发布了 JDK7, 做了一些简单的改进。

2014 年, Oracle 公司发布了 JDK8, 这一版本也做出了重大的改进, 其中比较显著的改进包括:

- 引入能够简化编程的 Lambda 表达式。
- 接口中允许包含提供了实现的默认方法和静态方法。
- 引入功能强大的 Stream API。
- 引入方便实用的处理日期和时间的新的 Date/Time API。
- 引入避免空指针异常的 Optional 类。

总之, 面向对象的 Java 语言具备“一次编程, 任何地方均可运行”的能力, 使其成为服务提供商和系统集成商用以支持多种操作系统和硬件平台的首选解决方案。Java 作为软件开发的一种革命性的技术, 其地位已被确定。如今, Java 技术已被列为当今世界信息技术的主流之一。表 P-1 对 Java 的发展历史做了总结。

表 P-1 Java 发展历史

年份	Java 发展历史
1995	Java 语言诞生
1996	JDK1.0 发布, 10 个最主要的操作系统供应商申明将在其产品中支持 Java 技术
1997	JDK1.1 发布
1998	JDK1.1 下载量超过 200 万次, JDK1.2 (称 Java2) 发布, JFC/Swing 技术发布, JFC/Swing 被下载了 50 多万次
1999	Java 被分成 J2SE、J2EE 和 J2ME, JSP/Servlet 技术诞生
2000	JDK1.3 发布, JDK1.4 发布
2001	Nokia 公司宣布到 2003 年将出售 1 亿部支持 Java 的手机, J2EE1.3 发布
2002	JDK1.4 发布, 自此 Java 的计算能力有了大幅度提升。J2EE SDK 的下下载量达到 200 万次
2003	5.5 亿台桌面机上运行 Java 程序, 75% 的开发人员将 Java 作为首要开发工具
2004	JDK1.5 发布, 这是 Java 语言发展史上的又一里程碑事件。为了表示这个版本的重要性, JDK1.5 更名为 JDK5.0
2005	JavaOne 大会召开, Java 的各种版本被更名, 取消其中的数字“2”: J2EE 更名为 Java EE, J2SE 更名为 Java SE, J2ME 更名为 Java ME
2006	JDK6 发布
2009	Sun 公司被 Oracle 公司收购
2011	JDK7 发布
2014	JDK8 发布, 对 Java 语言特性做了重大改进, 增加了 Lambda 表达式

Java 语言的特点

Java 应用如此广泛是因为 Java 具有多方面的优势。其特点如下：

(1) 面向对象。Java 自诞生之时就被设计成面向对象的语言，而 C++ 语言是一种强制面向对象的语言。面向对象可以说是 Java 最重要的特性，它不支持类似 C 语言那样的面向过程的程序设计技术。Java 支持静态和动态风格的代码重用。

(2) 跨平台。对于 Java 程序，不管是 Windows 平台还是 UNIX 平台或是其他平台，它都适用。Java 编辑器把 Java 源程序编译成与体系结构无关的字节码指令，只要安装了 Java 运行系统，Java 程序就可在任意的处理器上运行。这些字节码指令由 Java 虚拟机来执行，Java 虚拟机的解释器得到字节码后，对它进行转换，使之能够在不同的平台运行。

(3) 直接支持分布式的网络应用。除了支持基本的语言功能，Java 核心类库还包括一个支持 HTTP、SMTP 和 FTP 等基于 TCP/IP 协议的类库。因此，Java 应用程序可凭借 URL 打开并访问网络上的对象，其访问方式与访问本地文件系统几乎完全相同。在 Java 出现以前，为分布式环境尤其是 Internet 提供动态的内容无疑是一项非常宏伟、难以想象的任务，但 Java 的语言特性却使我们很容易地达到了这个目标。

(4) 安全性和健壮性。Java 致力于检查程序在编译和运行时的错误，类型检查帮助检查出许多开发早期出现的错误。Java 支持自动内存管理，这不但让程序员减轻了许多负担，也减少了程序员犯错的机会。Java 自己操纵内存减少了内存出错的可能性。Java 还能够检测数组边界，避免了覆盖数据的可能。在 Java 语言里，指针和释放内存等功能均被抛弃，从而避免了非法内存操作的危险。

以上特点，是 C++ 语言及其他语言无法比拟的（C++ 语言尽管也是面向对象的，但并不是严格意义上的面向对象的语言）。单从面向对象的特性来看，Java 类似于 SmallTalk，但其他特性，尤其是适用于分布式计算环境的特性远远超越了 SmallTalk。Java 发展到现在，已经不仅仅是一种语言，可以说是一种技术，这个技术涉及网络和编程等领域。另外，Java 是非常简单、高效的，有调查数据发现：用 C++ 和 Java 来做一个相同功能的项目，用 Java 写的程序要比用 C++ 写的程序节省 60% 的代码和 66% 的时间。可以说，用 Java 语言编程时间短、功能强，编程人员接手起来更容易、更简便。

本书的组织结构和主要内容

本书以 6 条主线贯穿全书：面向对象编程思想、Java 语言的语法、Java 虚拟机执行 Java 程序的原理、在实际项目中的运用、设计模式和性能优化技巧。书的每一章都会围绕若干条主线来展开内容，并且根据全书的布局，合理安排每一章内容的深度。本书主要内容包括：面向对象开发方法概述、第一个 Java 应用、数据类型和变量、操作符、流程控制、继承、Java 语言中的修饰符、接口、异常处理、类的生命周期、对象的生命周期、内部类、多线程、数组、Java 集合、泛型、Lambda 表达式、输入与输出（I/O）、图形用户界面、常用 Swing 组件、Java 常用类和 Annotation 注解。

这本书是否适合您

在如今的 Java 领域,各种新技术、新工具层出不穷,一方面,每一种技术都会不停地升级换代,另一方面,还会不断涌现出新的技术和工具。Java 世界就像小时候玩的万花筒,尽管实质上只是由几个普通的玻璃碎片组成的,但只要轻轻一摇,就会变化出千万种缤纷的图案。Java 世界如此变化多端,很容易让初学 Java 的人有无从下手的感觉。常常会有读者问我这样的问题:

我学了 Java 已经一年多了,现在就只能用 JSP 写点东西,其他的东西实在太多了,我整天学都学不完,很迷茫,不知道该如何有针对性地去学,以找到一份 Java 工作,现在是困死在 Java 里了。

撰写本书,目的之一是为了帮助读者看清 Java 万花筒的本质,从复杂的表象中寻找普遍的规律,深刻理解 Java 的核心思想,只有掌握了普遍的规律与核心思想,才能以不变应万变,轻轻松松地把握 Java 技术发展的新趋势,迅速地领略并且会熟练运用一门新的技术,而不成为被动的追随者,知其然而不知其所以然。

阅读本书,读者对 Java 的领悟将逐步达到以下境界:

- 熟悉 Java 语法,熟练地编译和调试程序。
- 按照面向对象的思想来快速理解 JDK 类库,以及其他第三方提供的类库,通过阅读 JavaDoc 和相关文档,知道如何正确地使用这些类库。
- 按照面向对象的思想来分析问题领域,设计对象模型。
- 在开发过程中会运用现有的一些优秀设计模式,提高开发效率。
- 当一个方法有多种实现方式时,能够从可维护、可重用及性能优化的角度选择最佳的实现方式。
- 理解 Java 虚拟机执行 Java 程序的原理,从而更深入地理解 Java 语言的各种特性和语法规则。

本书的技术支持网站

本书的技术支持网站为: www.javathinker.net。读者可以在该网站交流 Java 技术,提出本书的勘误信息,作者会在本网站为读者答疑。本书中实例源文件及思考题答案的下载网址为: www.javathinker.net/download.htm。

致谢

本书在编写过程中得到了 Oracle 公司在技术上的大力支持,电子工业出版社少儿与艺术分社负责编辑审核。此外,复旦软件学院的戴开宇等老师为本书的编写提供了有益的帮助,JavaThinker.net 网站的网友们为本书的升级提供了许多宝贵建议,在此表示衷心的感谢!尽管我们尽了最大努力,但本书难免会有不妥之处,欢迎各界专家和读者朋友批评指正。



第 1 章 面向对象开发方法概述	1
1.1 结构化的软件开发方法简介	3
1.2 面向对象的软件开发方法简介	6
1.2.1 对象模型	6
1.2.2 UML：可视化建模语言	7
1.2.3 Rational Rose：可视化建模工具	7
1.3 面向对象开发中的核心思想和概念	8
1.3.1 问题领域、对象、属性、状态、行为、方法、实现	8
1.3.2 类、类型	10
1.3.3 消息、服务	12
1.3.4 接口	13
1.3.5 封装、透明	14
1.3.6 抽象	18
1.3.7 继承、扩展、覆盖	20
1.3.8 组合	21
1.3.9 多态、动态绑定	24
1.4 UML 语言简介	26
1.4.1 用例图	27
1.4.2 类框图	28
1.4.3 时序图	29
1.4.4 协作图	30
1.4.5 状态转换图	30
1.4.6 组件图	31
1.4.7 部署图	32
1.5 类之间的关系	32
1.5.1 关联 (Association)	33
1.5.2 依赖 (Dependency)	34
1.5.3 聚集 (Aggregation)	35
1.5.4 泛化 (Generalization)	36
1.5.5 实现 (Realization)	36
1.5.6 区分依赖、关联和聚集关系	36
1.6 实现 Panel 系统	39
1.6.1 扩展 Panel 系统	42
1.6.2 用配置文件进一步提高 Panel 系统的可维护性	43
1.6.3 运行 Panel 系统	45

1.7 小结	45
1.8 思考题	46
第 2 章 第一个 Java 应用	47
2.1 创建 Java 源文件	47
2.1.1 Java 源文件结构	49
2.1.2 包声明语句	49
2.1.3 包引入语句	51
2.1.4 方法的声明	53
2.1.5 程序入口 main()方法的声明	54
2.1.6 给 main()方法传递参数	55
2.1.7 注释语句	55
2.1.8 关键字	56
2.1.9 标识符	56
2.1.10 编程规范	57
2.2 用 JDK 管理 Java 应用	57
2.2.1 JDK 简介以及安装方法	58
2.2.2 编译 Java 源文件	60
2.2.3 运行 Java 程序	62
2.2.4 给 Java 应用打包	65
2.3 使用和创建 JavaDoc 文档	66
2.3.1 JavaDoc 标记	68
2.3.2 javadoc 命令的用法	73
2.4 Java 虚拟机运行 Java 程序的基本原理	75
2.5 小结	77
2.6 思考题	78
第 3 章 数据类型和变量	81
3.1 基本数据类型	82
3.1.1 boolean 类型	82
3.1.2 byte、short、int 和 long 类型	83
3.1.3 char 类型与字符编码	85
3.1.4 float 和 double 类型	87
3.2 引用类型	91
3.2.1 基本类型与引用类型的区别	92
3.2.2 用 new 关键字创建对象	94
3.3 变量的作用域	95
3.3.1 实例变量和静态变量的生命周期	97

Contents

3.3.2	局部变量的生命周期	100	5.2.1	while 语句	154
3.3.3	成员变量和局部变量同名	101	5.2.2	do while 语句	156
3.3.4	将局部变量的作用域 最小化	102	5.2.3	for 语句	158
3.4	对象的默认引用: this	103	5.2.4	foreach 语句	161
3.5	参数传递	105	5.2.5	多重循环	162
3.6	变量的初始化以及默认值	107	5.3	流程跳转语句	162
3.6.1	成员变量的初始化	107	5.4	综合例子: 八皇后问题	165
3.6.2	局部变量的初始化	108	5.5	小结	168
3.7	直接数	109	5.6	思考题	169
3.7.1	直接数的类型	110	第 6 章	继承	173
3.7.2	直接数的赋值	111	6.1	继承的基本语法	173
3.8	小结	112	6.2	方法重载 (Overload)	175
3.9	思考题	113	6.3	方法覆盖 (Override)	177
第 4 章	操作符	115	6.4	方法覆盖与方法重载的 异同	183
4.1	操作符简介	115	6.5	super 关键字	183
4.2	整型操作符	116	6.6	多态	185
4.2.1	一元整型操作符	117	6.7	继承的利弊和使用原则	189
4.2.2	二元整型操作符	118	6.7.1	继承树的层次不可太多	190
4.3	浮点型操作符	123	6.7.2	继承树的上层为抽象层	190
4.4	比较操作符和逻辑操作符	124	6.7.3	继承关系最大的弱点: 打破封装	191
4.5	特殊操作符 “?:”	127	6.7.4	精心设计专门用于被继承 的类	193
4.6	字符串连接操作符 “+”	127	6.7.5	区分对象的属性与继承	195
4.7	操作符 “==” 与对象的 equals() 方法	129	6.8	比较组合与继承	197
4.7.1	操作符 “==”	129	6.8.1	组合关系的分解过程对应 继承关系的抽象过程	197
4.7.2	对象的 equals() 方法	130	6.8.2	组合关系的组合过程对应 继承关系的扩展过程	200
4.8	instanceof 操作符	133	6.9	小结	203
4.9	变量的赋值和类型转换	135	6.10	思考题	204
4.9.1	基本数据类型转换	136	第 7 章	Java 语言中的修饰符	209
4.9.2	引用类型的类型转换	139	7.1	访问控制修饰符	210
4.10	小结	139	7.2	abstract 修饰符	212
4.11	思考题	142	7.3	final 修饰符	214
第 5 章	流程控制	145	7.3.1	final 类	215
5.1	分支语句	146	7.3.2	final 方法	215
5.1.1	if else 语句	146			
5.1.2	switch 语句	150			
5.2	循环语句	154			

7.3.3 final 变量.....	216	9.3.3 区分运行时异常和受 检查异常.....	283
7.4 static 修饰符.....	220	9.4 用户定义异常.....	285
7.4.1 static 变量.....	220	9.4.1 异常转译和异常链.....	285
7.4.2 static 方法.....	223	9.4.2 处理多样化异常.....	288
7.4.3 static 代码块.....	226	9.5 异常处理原则.....	289
7.4.4 用 static 进行静态导入.....	228	9.5.1 异常只能用于非正常情况.....	290
7.5 小结.....	228	9.5.2 为异常提供说明文档.....	290
7.6 思考题.....	230	9.5.3 尽可能地避免异常.....	291
第 8 章 接口.....	233	9.5.4 保持异常的原子性.....	292
8.1 接口的概念和基本特征.....	234	9.5.5 避免过于庞大的 try 代码块.....	294
8.2 比较抽象类与接口.....	237	9.5.6 在 catch 子句中指定具体的 异常类型.....	294
8.3 与接口相关的设计模式.....	241	9.5.7 不要在 catch 代码块中忽略 被捕获的异常.....	294
8.3.1 定制服务模式.....	241	9.6 记录日志.....	295
8.3.2 适配器模式.....	245	9.6.1 创建 Logger 对象及设置 日志级别.....	296
8.3.3 默认适配器模式.....	250	9.6.2 生成日志.....	297
8.3.4 代理模式.....	251	9.6.3 把日志输出到文件.....	297
8.3.5 标识类型模式.....	256	9.6.4 设置日志的输出格式.....	298
8.3.6 常量接口模式.....	257	9.7 使用断言.....	299
8.4 小结.....	258	9.8 小结.....	300
8.5 思考题.....	259	9.9 思考题.....	301
第 9 章 异常处理.....	261	第 10 章 类的生命周期.....	305
9.1 Java 异常处理机制概述.....	262	10.1 Java 虚拟机及程序的 生命周期.....	305
9.1.1 Java 异常处理机制的优点.....	262	10.2 类的加载、连接和初始化....	305
9.1.2 Java 虚拟机的方法调用栈.....	264	10.2.1 类的加载.....	306
9.1.3 异常处理对性能的影响.....	267	10.2.2 类的验证.....	307
9.2 运用 Java 异常处理机制.....	267	10.2.3 类的准备.....	307
9.2.1 try-catch 语句：捕获异常.....	267	10.2.4 类的解析.....	308
9.2.2 finally 语句：任何情况下 必须执行的代码.....	268	10.2.5 类的初始化.....	308
9.2.3 throws 子句：声明可能会 出现的异常.....	270	10.2.6 类的初始化的时机.....	310
9.2.4 throw 语句：抛出异常.....	271	10.3 类加载器.....	313
9.2.5 异常处理语句的语法规则.....	271	10.3.1 类加载的父亲委托机制.....	315
9.2.6 异常流程的运行过程.....	274	10.3.2 创建用户自定义的类 加载器.....	317
9.2.7 跟踪丢失的异常.....	278	10.3.3 URLClassLoader 类.....	323
9.3 Java 异常类.....	280		
9.3.1 运行时异常.....	282		
9.3.2 受检查异常 (Checked Exception).....	282		

10.4	类的卸载	324	12.3	子类与父类中的内部类	
10.5	小结	325	同名	380	
10.6	思考题	326	12.4	匿名类	381
第 11 章	对象的生命周期	327	12.5	内部接口以及接口中的	
11.1	创建对象的方式	327	内部类	384	
11.2	构造方法	330	12.6	内部类的用途	385
11.2.1	重载构造方法	331	12.6.1	封装类型	385
11.2.2	默认构造方法	332	12.6.2	直接访问外部类的成员	385
11.2.3	子类调用父类的构造方法	333	12.6.3	回调	386
11.2.4	构造方法的作用域	337	12.7	内部类的类文件	388
11.2.5	构造方法的访问级别	337	12.8	小结	389
11.3	静态工厂方法	338	12.9	思考题	389
11.3.1	单例类	340	第 13 章	多线程	393
11.3.2	枚举类	342	13.1	Java 线程的运行机制	393
11.3.3	不可变 (immutable) 类与		13.2	线程的创建和启动	395
可变类	344	13.2.1	扩展 java.lang.Thread 类	395	
11.3.4	具有实例缓存的		13.2.2	实现 Runnable 接口	400
不可变类	348	13.3	线程的状态转换	402	
11.3.5	松耦合的系统接口	350	13.3.1	新建状态	402
11.4	垃圾回收	351	13.3.2	就绪状态	402
11.4.1	对象的可触及性	352	13.3.3	运行状态	402
11.4.2	垃圾回收的时间	354	13.3.4	阻塞状态	403
11.4.3	对象的 finalize() 方法简介	354	13.3.5	死亡状态	404
11.4.4	对象的 finalize() 方法的		13.4	线程调度	405
特点	355	13.4.1	调整各个线程的优先级	406	
11.4.5	比较 finalize() 方法和 finally		13.4.2	线程睡眠: Thread.sleep()	
代码块	357	方法	408		
11.5	清除过期的对象引用	358	13.4.3	线程让步:	
11.6	对象的强、软、弱和		Thread.yield() 方法	409	
虚引用	360	13.4.4	等待其他线程结束: join()	410	
11.7	小结	366	13.5	获得当前线程对象的引用	411
11.8	思考题	367	13.6	后台线程	412
第 12 章	内部类	371	13.7	定时器	413
12.1	内部类的基本语法	371	13.8	线程的同步	415
12.1.1	实例内部类	373	13.8.1	同步代码块	418
12.1.2	静态内部类	376	13.8.2	线程同步的特征	422
12.1.3	局部内部类	377	13.8.3	同步与并发	425
12.2	内部类的继承	379	13.8.4	线程安全的类	426
			13.8.5	释放对象的锁	427

13.8.6 死锁.....	429
13.9 线程通信.....	430
13.10 中断阻塞.....	435
13.11 线程控制.....	436
13.11.1 被废弃的 suspend()和 resume()方法.....	437
13.11.2 被废弃的 stop()方法.....	438
13.11.3 以编程的方式控制线程.....	438
13.12 线程组.....	440
13.13 处理线程未捕获的异常.....	441
13.14 ThreadLocal 类.....	443
13.15 concurrent 并发包.....	445
13.15.1 用于线程同步的 Lock 外部锁.....	446
13.15.2 用于线程通信的 Condition 条件接口.....	447
13.15.3 支持异步计算的 Callable 接口和 Future 接口.....	450
13.15.4 通过线程池来高效管理 多个线程.....	452
13.15.5 BlockingQueue 阻塞队列.....	454
13.16 小结.....	457
13.17 思考题.....	458
第 14 章 数组.....	461
14.1 数组变量的声明.....	461
14.2 创建数组对象.....	462
14.3 访问数组的元素和长度.....	463
14.4 数组的初始化.....	465
14.5 多维数组以及不规则数组.....	465
14.6 调用数组对象的方法.....	467
14.7 把数组作为方法参数或 返回值.....	467
14.8 数组排序.....	470
14.9 数组的二分查找算法.....	471
14.10 哈希表.....	472
14.11 数组实用类: Arrays.....	477
14.12 用符号“...”声明数目 可变参数.....	480
14.13 小结.....	481

14.14 思考题.....	481
第 15 章 Java 集合.....	485
15.1 Collection 和 Iterator 接口.....	486
15.2 集合中直接加入基本类型 数据.....	489
15.3 Set (集).....	490
15.3.1 Set 的一般用法.....	490
15.3.2 HashSet 类.....	491
15.3.3 TreeSet 类.....	493
15.4 List (列表).....	497
15.4.1 访问列表的元素.....	498
15.4.2 为列表排序.....	498
15.4.3 ListIterator 接口.....	499
15.4.4 获得固定长度的 List 对象.....	500
15.4.5 比较 Java 数组和各种 List 的性能.....	500
15.5 Queue (队列).....	503
15.5.1 Deque (双向队列).....	504
15.5.2 PriorityQueue (优先级队列).....	505
15.6 Map (映射).....	505
15.7 HashSet 和 HashMap 的 负载因子.....	507
15.8 集合实用类: Collections.....	508
15.9 线程安全的集合.....	510
15.10 集合与数组的互换.....	511
15.11 集合的批量操作.....	512
15.12 历史集合类.....	513
15.13 枚举类型.....	517
15.13.1 枚举类型的构造方法.....	519
15.13.2 EnumSet 类和 EnumMap 类.....	520
15.14 小结.....	521
15.15 思考题.....	521
第 16 章 泛型.....	523
16.1 Java 集合的泛型.....	523
16.2 定义泛型类和泛型接口.....	524

16.3	用 extends 关键字限定 类型参数.....	526
16.4	定义泛型数组.....	527
16.5	定义泛型方法.....	528
16.6	使用“?”通配符.....	529
16.7	使用泛型的注意事项.....	530
16.8	小结.....	531
16.9	思考题.....	531
第 17 章	Lambda 表达式.....	533
17.1	Lambda 表达式的基本用法..	533
17.2	用 Lambda 表达式代替 内部类.....	534
17.3	Lambda 表达式和集合的 forEach()方法.....	535
17.4	用 Lambda 表达式对集合 进行排序.....	536
17.5	Lambda 表达式与 Stream API 联合使用.....	537
17.6	Lambda 表达式可操纵的 变量作用域.....	539
17.7	Lambda 表达式中的方法 引用.....	540
17.8	函数式接口 (FunctionalInterface)	541
17.9	总结 Java 语法糖.....	541
17.10	小结.....	542
17.11	思考题.....	542
第 18 章	输入与输出 (I/O)	545
18.1	输入流和输出流概述.....	546
18.2	输入流.....	547
18.2.1	字节数组输入流： ByteArrayInputStream 类.....	548
18.2.2	文件输入流： FileInputStream 类.....	549
18.2.3	管道输入流： PipedInputStream	551
18.2.4	顺序输入流： SequenceInputStream 类.....	552
18.3	过滤输入流： FilterInputStream.....	552
18.3.1	装饰器设计模式.....	553
18.3.2	过滤输入流的种类.....	554
18.3.3	DataInputStream 类.....	555
18.3.4	BufferedInputStream 类.....	556
18.3.5	PushbackInputStream 类.....	557
18.4	输出流.....	557
18.4.1	字节数组输出流： ByteArrayOutputStream 类..	557
18.4.2	文件输出流： FileOutputStream.....	558
18.5	过滤输出流： FilterOutputStream	559
18.5.1	DataOutputStream	559
18.5.2	BufferedOutputStream.....	559
18.5.3	PrintStream 类.....	561
18.6	Reader/Writer 概述.....	563
18.7	Reader 类.....	565
18.7.1	字符数组输入流： CharArrayReader 类.....	566
18.7.2	字符串输入流： StringReader 类.....	566
18.7.3	InputStreamReader 类.....	567
18.7.4	FileReader 类.....	568
18.7.5	BufferedReader 类.....	568
18.8	Writer 类.....	568
18.8.1	字符数组输出流： CharArrayWriter 类.....	569
18.8.2	OutputStreamWriter 类.....	570
18.8.3	FileWriter 类.....	572
18.8.4	BufferedWriter 类.....	573
18.8.5	PrintWriter 类.....	573
18.9	标准 I/O	574
18.9.1	重新包装标准输入和输出...	574
18.9.2	标准 I/O 重定向.....	575
18.10	随机访问文件类： RandomAccessFile	576
18.11	新 I/O 类库.....	577
18.11.1	缓冲器 Buffer 概述.....	578

18.11.2 通道 Channel 概述.....	579	19.5 AWT 绘图.....	637
18.11.3 字符编码 Charset 类概述 ..	581	19.5.1 Graphics 类.....	639
18.11.4 用 FileChannel 读写文件...	581	19.5.2 Graphics2D 类.....	644
18.11.5 控制缓冲区.....	582	19.6 AWT 线程（事件分派 线程）.....	647
18.11.6 字符编码转换.....	583	19.7 小结.....	649
18.11.7 缓冲区视图.....	584	19.8 思考题.....	650
18.11.8 文件映射缓冲区： MappedByteBuffer.....	586	第 20 章 常用 Swing 组件.....	653
18.11.9 文件加锁.....	587	20.1 边框（Border）.....	653
18.12 对象的序列化与反序列化 ..	589	20.2 按钮组件（AbstractButton） 及子类.....	654
18.13 自动释放资源.....	595	20.3 文本框（JTextField）.....	657
18.14 用 File 类来查看、创建和 删除文件或目录.....	596	20.4 文本区域（JTextArea）与 滚动面板（JScrollPane）....	660
18.15 用 java.nio.file 类库来操作 文件系统.....	599	20.5 复选框（JCheckBox）与单选 按钮（JRadioButton）.....	661
18.15.1 复制、移动文件以及遍历、 过滤目录树.....	600	20.6 下拉列表（JComboBox）....	664
18.15.2 查看 ZIP 压缩文件.....	601	20.7 列表框（JList）.....	665
18.16 小结.....	602	20.8 页签面板（JTabbedPane）...	667
18.17 思考题.....	603	20.9 菜单（JMenu）.....	669
第 19 章 图形用户界面.....	605	20.10 对话框（JDialog）.....	674
19.1 AWT 组件和 Swing 组件.....	605	20.11 文件对话框 （JFileChooser）.....	676
19.2 创建图形用户界面的 基本步骤.....	608	20.12 消息框.....	679
19.3 布局管理器.....	610	20.13 制作动画.....	681
19.3.1 FlowLayout（流式布局 管理器）.....	611	20.14 播放音频文件.....	683
19.3.2 BorderLayout（边界布局 管理器）.....	613	20.15 BoxLayout 布局管理器.....	686
19.3.3 GridLayout（网格布局 管理器）.....	616	20.16 设置 Swing 界面的外观 和感觉.....	689
19.3.4 CardLayout（卡片布局 管理器）.....	619	20.17 小结.....	691
19.3.5 GridBagLayout（网格包 布局管理器）.....	620	20.18 思考题.....	692
19.4 事件处理.....	626	第 21 章 Java 常用类.....	693
19.4.1 事件处理的软件实现.....	626	21.1 Object 类.....	693
19.4.2 事件源、事件和监听器 的类层次和关系.....	632	21.2 String 类和 StringBuffer 类...	694
		21.2.1 String 类.....	694
		21.2.2 “hello”与 new String(“hello”) 的区别.....	697
		21.2.3 StringBuffer 类.....	698

Contents

21.2.4	比较 String 类与 StringBuffer 类	699
21.2.5	正则表达式	701
21.2.6	格式化字符串	703
21.3	包装类	707
21.3.1	包装类的构造方法	707
21.3.2	包装类的常用方法	708
21.3.3	包装类的自动装箱和拆箱 ..	709
21.4	Math 类	710
21.5	Random 类	712
21.6	传统的处理日期/时间的类 ...	712
21.6.1	Date 类	713
21.6.2	DateFormat 类	713
21.6.3	Calendar 类	715
21.7	新的处理日期/时间的类	716
21.7.1	LocalDate 类	717
21.7.2	LocalTime 类	718
21.7.3	LocalDateTime 类	718

21.8	BigInteger 类	719
21.9	BigDecimal 类	720
21.10	用 Optional 类避免空指针 异常	722
21.11	小结	724
21.12	思考题	725
第 22 章	Annotation 注解	727
22.1	自定义 Annotation 注解 类型	727
22.2	在类的源代码中引用 注解类型	730
22.3	在程序中运用反射机制读取 类的注解信息	732
22.4	基本内置注解	735
22.5	小结	736
22.6	思考题	736

第1章 面向对象开发方法概述

一般说来，软件开发都会经历以下生命周期：

- 软件分析：分析问题领域，了解用户的需求。
- 软件设计：确定软件的总体架构，把整个软件系统划分成大大小小的多个子系统，设计每个子系统的结构。
- 软件编码：用选定的编程语言来编写程序代码，实现在设计阶段勾画出的软件蓝图。
- 软件测试：测试软件是否能实现特定的功能，以及测试软件的运行性能。
- 软件部署：为用户安装软件系统，帮助用户正确地使用软件。
- 软件维护：修复软件中存在的 Bug，当用户需求发生变化时（增加新的功能，或者修改已有功能的实现方式），修改相应的软件。

为了提高软件开发效率，降低软件开发成本，一个优良的软件系统应该具备以下特点：

- 可重用性：减少软件中的重复代码，避免重复编程。
- 可扩展性：当软件必须增加新的功能时，能够在现有系统结构的基础上，方便地创建新的子系统，不需要改变软件系统现有的结构，也不会影响已经存在的子系统。
- 可维护性：当用户需求发生变化时，只需要修改局部的子系统的少量程序代码，不会牵一发而动全身，修改软件系统中多个子系统的程序代码。

Tips

本章多次使用了系统结构的说法，这里的系统结构是指系统由多个子系统组成，以及子系统由多个更小的子系统组成的结构。

如何才能使软件系统具备以上特点呢？假如能把软件分解成多个小的子系统，每个子系统相对独立，把这些子系统像搭积木一样灵活地组装起来就构成了整个软件系统，这样的软件系统便能获得以上优良特性。软件中的子系统具有以下特点：

- 结构稳定性：软件在设计阶段，在把一个系统划分成更小的子系统时，设计合理，使得系统的结构比较健壮，能够适应用户变化的需求。
- 可扩展性：当软件必须增加新的功能时，可在现有子系统的基础上创建出新的子系统，该子系统继承了原有子系统的一些特性，并且还具有一些新的特性，从而提高软件的可重用性和可扩展性。
- 内聚性：每个子系统只完成特定的功能，不同子系统之间不会有功能的重叠。为了避免子系统之间功能的重叠，每个子系统的粒度在保持功能完整性的前提下都尽可能小，按这种方式构成的系统结构被称为精粒度系统结构。子系统的内聚性会提高软件的可重用性和可维护性。

- 可组合性：若干精粒度的子系统经过组合，就变成了大系统。子系统的可组合性会提高软件的可重用性和可维护性，并且能够简化软件的开发过程。
- 松耦合：子系统之间通过设计良好的接口进行通信，但是尽量保持相互独立，修改一个子系统，不会影响到其他的子系统。当用户需求发生变化时，只需要修改特定子系统的实现方式，从而提高软件的可维护性。

Tips

在精粒度系统结构中，大系统可以分解为多个松耦合的精粒度子系统。与此相反，在粗粒度系统结构中，粗粒度的大系统无法进一步分解，或者只能被分解为有限的几个粗粒度子系统。

图 1-1 显示了一个用积木搭建起来的建筑物系统，有凯旋门，还有小狗狗的家。这些系统中的最小子系统是各种形状的积木（精粒度的子系统），这些积木能够在多个子系统中重用，多个积木能组合成复杂的子系统。当大的系统被拆散或摧毁时，它所包含的小子系统依然有用。

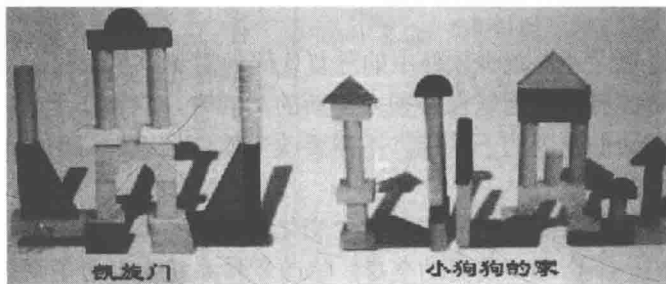


图 1-1 用积木搭建起来的建筑物系统

目前在软件开发领域有两种主流的开发方法：结构化开发和面向对象开发。结构化开发是一种比较传统的开发方法，早期的高级编程语言，如 Basic、C、Fortran 和 Pascal 等，都是支持结构化开发的编程语言。随着软件开发技术的逐步发展，为了进一步提高软件的可重用性、可扩展性和可维护性，面向对象的编程语言及面向对象设计理论应运而生，Java 语言就是一种纯面向对象的编程语言。

本章首先简要介绍结构化的软件开发过程，然后介绍了面向对象的软件开发过程，对面向对象的一些核心思想和概念进行了阐述。本章列举了不少形象的例子，来帮助读者理解面向对象的开发思想，并且以一个画板（Panel）软件系统的例子贯穿整个章节，这个例子分别按照结构化开发方式和面向对象开发方式实现，从而鲜明地对比这两种开发方式对软件的可维护性、可扩展性和可重用性的影响。

本章的思想性和理论性比较强，如果读者已经有一定的面向对象的开发经验，阅读本章时会很顺利。如果读者没有面向对象的开发经验，初次阅读时会感觉比较抽象，在这种情况下，初次阅读时只需了解一些基本概念与核心思想即可，等到阅读完全书后，再来回顾和领悟本章内容。

本章在介绍范例时，展示了部分 Java 程序代码，如果读者对 Java 编程没有任何基