国外计算机科学经典教材

Mc Graw Hill Education

Data Structures and the Java Collections Framework

数据结构和Java集合框架

(美) William J. Collins 著
陈曙晖 译

清华大学出版社

国外计算机科学经典教材

数据结构和Java集合框架

(美)William J. Collins 著

陈曙晖 译

清华大学出版社

北 京

William J.Collins
Data Structures and the Java Collections Framework
EISBN: 0-07-236964-7

图书在版编目(CIP)数据

数据结构和 Java 集合框架/(美)柯林斯(Collins, W.J.)著；陈曙晖译. —北京：清华大学出版社，2006.4
书名原文：Data Structures and the Java Collections Framework
(国外计算机科学经典教材)
ISBN 7-302-12134-6
Ⅰ. 数…　Ⅱ. ①柯…②陈…　Ⅲ. ①数据结构—教材②JAVA 语言—程序设计—教材
Ⅳ. ①TP311.12②TP312

中国版本图书馆 CIP 数据核字(2005)第 136277 号

出 版 者：清华大学出版社　　**地　　址**：北京清华大学学研大厦
http://www.tup.com.cn　**邮　　编**：100084
社 总 机：010-62770175　**客户服务**：010-62776969
组稿编辑：曹　康
文稿编辑：王　黎
封面设计：孔祥丰
版式设计：孔祥丰
印 刷 者：清华大学印刷厂
装 订 者：北京鑫海金澳胶印有限公司
发 行 者：新华书店总店北京发行所
开　　本：185×260　**印张**：37.75　**字数**：966 千字
版　　次：2006 年 4 月第 1 版　2006 年 4 月第1 次印刷
书　　号：ISBN 7-302-12134-6/TP・7834
印　　数：1～4000
定　　价：68.00 元

出 版 说 明

近年来，我国高等学校的计算机学科教育进行了较大的改革，急需一批门类齐全、具有国际水平的计算机经典教材，以适应当前的教学需要。引进国外经典教材，可以了解并吸收国际先进的教学思想和教学方法，使我国的计算机学科教育能够与国际接轨，从而培育更多具有国际水准的计算机专业人才，增强我国信息产业的核心竞争力。Pearson、Thomson、McGraw-Hill、Springer、John Wiley 等出版集团都是全球最有影响的图书出版机构，它们在高等教育领域也都有着不凡的表现，为全世界的高等学校计算机教学提供了大量的优秀教材。为了满足我国高等学校计算机学科的教学需要，我社计划从这些知名的国外出版集团引进计算机学科经典教材。

为了保证引进版教材的质量，我们在全国范围内组织并成立了“清华大学计算机外版教材编审委员会”(以下简称“编委会”)，旨在对引进教材进行审定、对教材翻译质量进行评审。“编委会”成员皆为全国各类重点院校教学与科研第一线的知名教授，其中许多教授为各校相关院、系的院长或系主任。“编委会”一致认为，引进版教材要能够满足国内各高校计算机教学与国际接轨的需要，要有特色风格，有创新性、先进性、示范性和一定的前瞻性，是真正的经典教材。为了保证外版教材的翻译质量，我们聘请了高校计算机相关专业教学与科研第一线的教师及相关领域的专家担纲译者，其中许多译者为海外留学回国人员。为了尽可能地保留与发扬教材原著的精华，在经过翻译和编辑加工之后，由“编委会”成员对文稿进行审定，以最大程度地弥补和修正在前面一系列加工过程中对教材造成的误差和瑕疵。

由于时间紧迫和能力所限，本套外版教材在出版过程中还可能存在一些不足和遗憾，欢迎广大师生批评指正。同时，也欢迎读者朋友积极向我们推荐各类优秀的国外计算机教材，共同为我国高等学校的计算机教育事业贡献力量。

清华大学出版社

国外计算机科学经典教材

编审委员会

前　言

本书包括了讲述数据结果和算法的面向对象课程。它采用 Java 语言实现，并假定学生在此之前已经学过该语言的基础课程，这些课程覆盖了基本语句和数据类型的用法，以及数组和文件的基本操作。

Java 集合框架

本书的一个显著特点就是：它的重点放在 Java 集合框架(Java Collection Framwork)上，这个框架是 java.util 包的一部分。它基本上是一个层次结构，各层(除了最底层之外)由不同的接口和在最底层实现这些接口的集合类组成。这些集合类实现了计算机科学课程中所讲授的绝大多数数据结构，比如可变大小的数组类、链表类、平衡树类以及散列-集合类。

使用 Java 集合框架有几个好处。首先，学生使用的代码都已经经过广泛测试，而不需要由教员或者教科书作者另外创建一套模块。其次，学生们有机会学习专家们的代码，这些代码一定会比他们之前见过的代码高效，同时更加简洁。最后，这个框架可以用于教学大纲中的后续课程，甚至对课程的研究也同样有帮助！

其他实现

尽管 Java 集合类非常重要，但是在数据结构和算法基础课中，它并不是惟一的研究热点。那些不同于 Java 集合框架中的方法也值得考虑。例如，因为 HashSet 类和 HashMap 类使用链式结构，所以我们用单独的一节来讨论开放寻址，同时还讨论了不同设计的取舍。另外，本书还覆盖了 Java 集合框架中尚未包含的数据结构(例如图)和算法(例如堆排序)。

图形用户接口

我们使用了一个简单的图形用户接口(GUI)来取代控制台输入和输出，它的输入只有一行，输出可以有任意多行。第 1 章给出了这个 GUI 的大概轮廓，附录 B 给出了更为详细的描述。使用这个 GUI 带来 2 个问题，一个问题是无法循环输入，另一个问题是输出是滚动的。但是它的主要的目的是让学生更好地理解事件驱动模式编程的特征：面向现实世界环境。

教学方法

本书提供了几种方法来提高教员的教学环境和学生的学习环境。每一章开头处都列举了学习目标。每章在结束时提供至少一个较大的编程作业。我们对每一个数据结构都进行了详细描

述，同时还讲解使用每个方法的前置条件以及后置条件。另外，绝大多数方法均有例子演示如何调用这些方法以及调用的结果。

本书仔细研究了实现细节，特别是Java集合框架的实现细节，并在24个实验组成的套件中得到加强。我们将在后文介绍这些实验的组织。

支持材料

www.mhhe.com/collins网站上有所有的支持材料，它为学生们提供了以下信息：

- 实验概要及获取方法；
- 本书开发的所有项目的源代码；
- 一些小应用程序(applet)，用于那些具有强大可视化组件的项目。

另外，教员也可以从该网站上获取以下信息：

- 关于实验的教员选项；
- 每章的PowerPoint幻灯片(大约有1500张)；
- 所有习题的解答。

内容提要

第1章作为全书后续章节的基础，描述了Java的特性。大部分材料反映了面向对象的思想：继承、多态以及异常处理。本章有几个实验用于复习类以及继承和异常处理。查阅前言中“实验组织”小节，可以获取更多关于实验的信息。

第2章介绍了抽象类和接口，每个专题都有实验。我们将关注Collection接口，它是Java集合框架中许多类的根。我们创建了一个简单版本的单向链表类(LinkedCollection)，作为Collection接口的简单实现。创建这个类的目的主要是为了介绍Java集合框架几个重要特性(如迭代器和内置类)提供技术背景。迭代器实验可以帮助学生巩固对这个重要概念的理解。

第3章介绍了软件工程，概括了软件开发周期的4个阶段：分析、设计、实现以及维护。本章引入统一建模语言(UML)作为设计工具来描述继承、组合和集合。后面几章中用到的大O符号为对各种方法所需时间(独立于特定环境)进行估算提供了一种手段。我们将讨论运行时的效率和时间特性，对于每个主题都有相应的实验。

第4章的主题是递归，我们暂时将重点从数据结构转移到算法上来。本章介绍了回溯，目的不仅仅在于提供一种解决问题的通用方法，而且用来演示如何通过接口来创建多态引用。同时这个BackTrack类还提供如下用途：搜索迷宫；在棋盘上放置8个皇后，使得没有一个皇后会被另一个皇后攻击；同样还用来演示一个武士可以横跨所有棋盘中的所有方格，并且只在每个方格上横跨一次。而递归的其他应用(例如Hanoi塔游戏)进一步说明了递归方法的优点，特别是在与迭代方法比较时。Fibonacci数字实验、二分法搜索实验以及排列生成实验更进一步地说明了递归方法的优越性。递归方法在后面的章节中还将遇到，尤其是在讲解使用Java集合框架进行快速排序和归并排序时。递归虽然很少被用到，但是它是每个计算机专家必不可少的工具。

在第 5 章我们将从 ArrayList 数据结构和类开始学习 Java 集合框架。ArrayList 数据结构属于“智能”数组：能够自动调整数组大小，并且可以利用任何索引进行插入和删除。我们将从 ArrayList 类中使用最广泛的方法描述开始展开对类设计的讨论，包括方法的前置条件、后置条件以及方法头部。紧接着是这个类的实现概要，而进一步的实现细节留在实验中。应用 ArrayList 类可以实现高精度算术运算，这对于公钥加密是非常重要的。我们将在相应的实验和程序工程中进一步扩展这个应用。还将有另外一个编程项目用来开发 Deque 类。

第 6 章介绍了 LinkedList 数据结构和类，刻画了如何使用线性方法在任意位置进行插入、删除和检索。这个特性为链表迭代器提供了非常合适的使用场合：遍历 ArrayList 对象，在当前位置进行插入、删除或者检索具有常数时间的方法。Java 集合框架设计使用的是双向链表和环形结构，但是我们还考虑了其他方法。其应用是行编辑器，而链表迭代器非常适合于这类应用。这章后面的编程项目将进一步扩展这个应用。

第 7 章的主题是队列和堆栈。Java 集合框架目前并不包括 Queue 类(队列类)，但是使用一个 LinkedList 字段就可以非常容易和高效地实现它。另外我们还介绍了一个连续的实现。像计算排队洗车的平均等待时间这样的特殊应用属于计算机模拟通用类。java.util 包中 Stack 类的实现在 Java 集合框架之前就已经存在了。Stack 类有两个应用：实现递归以及中后缀符号转换。有个实验扩展了第二个应用，并形成了一个状态评估编程项目的基础。

第 8 章论述了一般的二叉树，然后特别地介绍二叉搜索树。本章描述了二叉树的一些关键特性，这些特性对于后面理解关于 AVL 树、red-black 树、堆以及决策树等的材料非常重要。学习二叉搜索树为第 9 章的主题(平衡二叉搜索树)做好了铺垫。二叉搜索树实际上是 Java 集合框架中 red-black 树的单色版本。

第 9 章我们将着重讨论平衡二叉搜索树，以及 AVL 树和 red-black 树。引用旋转机制可以达到树的重新平衡。通过 Fibonacci 树的帮助，可确定 AVL 树的高度与该树中的元素个数总是成对数关系。red-black 树也有类似的性质。我们实现了 AVLTree 类，但是没有实现 remove 方法(在编程项目 9.1 完成)。

第 10 章的重点是 red-black 树，利用 Java 集合框架中的 TreeMap 和 TreeSet 类来实现。在 Map 对象中，每个元素均有惟一的键-值对。TreeMap 对象存储在 red-black 树中，按照元素的键进行排序。有几个实验通过一些细节来指导学生在插入和删除之后如何重构树。相关的应用包括在字典中查找同义词。TreeSet 对象是作为 TreeMap 对象实现的，每个元素拥有相同的值。TreeSet 类的应用之一是一个简单的拼写检查器。有些编程项目就是要判断文本文件中每个词出现的频率，然后建立关键词索引。

第 11 章介绍了 PriorityQueue 接口，它目前还不属于 Java 集合框架。基于堆的实现允许以常数的平均时间插入元素，同时以最差对数量级的时间来删除最高优先级元素。其应用是数据压缩领域，特别是 Huffman 编码：给定文本文件，产生最小的无前缀(prefix-free)编码。本章的编程项目是将编码转换成原始的文本文件。而实验是将公平性整合到优先级队列中，这样那些在优先级队列中等待时间最长的元素将获得优先处理。

第 12 章的主题是排序。我们估计了基于比较方法的排序的最大下界。Java 集合框架提供了两种排序方法：针对基本数据类型数组的快速排序，以及针对由实现 List 接口的对象构成的数组的归并排序。同时还引进了其他两种很重要的排序方法：树排序和堆排序。本章的实验通过一组随机产生的整数对所有这些排序方法进行了比较。本章的编程项目是对一个名字和社会保险号码的文件进行排序。

第 13 章首先回顾了顺序检索和二叉树检索，然后研究了散列方法。Java 集合框架为由键-值对组成的元素提供了 HashMap 类。HashSet 类是退化的 HashMap 类，因为可以将 HashSet 对象看作是所有元素具有相同值的 HashMap 对象。插入、删除和搜索的平均时间基本上是常数。在比较 HashMap 和 TreeMap 对象的实验中将进一步探索平均速度。还有一个链式散列(它是 HashMap 类的基础)与开放地址散列的比较。在本章后面的编程项目中将深入探讨这个比较。

第 14 章介绍了最通用的数据类型，即图、树以及网络。首先简要介绍了几个关键算法：宽度优先遍历、深度优先遍历、连通性、寻找最小生成树以及寻找两点间最短路径。本章开发的惟一的一个类是基于邻接表实现的(有向)Network 类。其他的类(如 UndirectedGraph 类和 UndirectedNetwork 类)均可以直接定义为 Network 类的子类。本章后面有个实验研究了货郎问题，还有一个编程项目用来完成 Network 类的邻接矩阵版本。此外还介绍了另一个回溯应用，它用到了第 4 章引入的 BackTrack 类。

每一章均有一个网页包含那一章中开发的所有程序以及 applet，恰当地演示了所介绍的概念。

附录

附录 A 中所包含的背景知识让学生们能够领会各章的数学问题。求和符号和对数的基本特性是非常重要的，这些数学知识将使得我们对二叉树和开放寻址散列表的分析更加深入。

附录 B 是 GUI 类和 GUIListener 类的简明使用手册。这些类支持本书中以及实验中所有程序输入输出的事件模型。理解事件模型以及每个独立线程的作用要比理解控制台输入输出要复杂一些。但是实际上所有的应用程序都是事件驱动的，我们将来还要花时间研究这些主题。

附录 C 介绍了 Java 集合框架的用户视图。对于每个方法均提供了方法描述：前置条件、后置条件以及方法头。这 6 个类和 4 个接口的关系如下：

ArrayList 类实现了 List 接口，List 接口扩展了 Collection 接口；

LinkedList 类实现了 List 接口，List 接口扩展了 Collection 接口；

TreeMap 类实现了 Map 接口；

TreeSet 类实现了 Set 接口，List 接口扩展了 Collection 接口；

HashMap 类实现了 Map 接口；

HashSet 类实现了 Set 接口，List 接口扩展了 Collection 接口；

实验组织

与本书相关的有 24 个网站实验室。学生和教师均可以访问 URL：www.mhhe.com/collins。这些实验室不包括那些关键的材料，但是提供了增强的文字材料。比如，在研究完 ArrayList 类和 LinkedList 类之后，有两个实验针对这两个类进行与时间相关的实验。

这些实验是自包含的，因此教师在安排实验时，可以灵活掌握：

(1) 可将它们布置成封闭的实验室

(2) 也可以将它们布置成开放的实验室

(3) 还可以把它们布置成不计学分的家庭作业

除了可以促进学生主动学习这一明显好处之外，这些实验还鼓励学生使用科学的方法。每个实验室基本上都是一次实践。学生们**观察**一些现象，如 Java 集合框架中 LinkedList 类的组织，然后使用他们自己的代码制定并提交一个关于这种现象的**假设**。在**测试**之后，也许还要修改假设，提交从这次实验中得出的**结论**。

前面几章的实验要比后面几章多。这样，学生在课程开始时就可以开始实验，甚至是在布置编程项目之前。

目　录

第 1 章　Java 语言的重要特性 …… 1
1.1　类 …… 1
1.1.1　方法描述 …… 2
1.1.2　数据抽象 …… 4
1.1.3　Employee 类 …… 6
1.1.4　局部变量和字段 …… 8
1.1.5　构造函数 …… 8
1.1.6　实例变量和静态变量 …… 9
1.1.7　可见性修饰符 …… 10
1.1.8　图形用户接口 …… 10
1.1.9　Company 类 …… 11
1.1.10　继承 …… 12
1.1.11　可见性修饰符 protected …… 13
1.1.12　继承和构造函数 …… 15
1.1.13　多态性(Polymorphism) …… 19
1.1.14　信息隐藏 …… 21
1.1.15　异常处理 …… 22
1.1.16　异常传送 …… 24
1.2　小结 …… 26
1.3　练习 …… 27

第 2 章　接口和集合类 …… 31
2.1　抽象方法和抽象类 …… 31
2.2　接口 …… 33
2.3　数组 …… 36
2.4　集合类 …… 38
2.5　集合类的存储结构 …… 39
2.5.1　链接结构 …… 39
2.5.2　LinkedCollection 类 …… 39
2.5.3　LinkedCollection 类中的字段和方法定义 …… 42
2.5.4　迭代器 …… 45
2.5.5　数据结构和 Java Collections Framework …… 47
2.6　小结 …… 48
2.7　练习 …… 48

第 3 章　软件工程介绍 …… 51
3.1　软件开发生命期 …… 51
3.2　问题分析 …… 52
3.3　程序设计 …… 53
3.3.1　方法描述和字段 …… 54
3.3.2　依赖性图表 …… 55
3.4　程序实现 …… 57
3.4.1　方法验证 …… 57
3.4.2　修正是否可行 …… 58
3.4.3　评估方法的效率 …… 58
3.4.4　大 O 符号 …… 59
3.4.5　快速获取大 O 估计时间 …… 61
3.4.6　平衡 …… 64
3.4.7　运行时分析 …… 65
3.4.8　Random 类 …… 66
3.5　程序维护 …… 67
3.6　小结 …… 68
3.7　练习 …… 68

第 4 章　递归 …… 73
4.1　绪论 …… 73
4.2　阶乘 …… 74
4.3　十进制转换成二进制 …… 77
4.4　汉诺塔 …… 80
4.5　回溯 …… 88
4.6　二叉树搜索 …… 96
4.7　间接递归 …… 105
4.8　递归的开销 …… 105
4.9　小结 …… 106
4.10　练习 …… 107

第 5 章　数组列表 …… 119
5.1　List 接口 …… 119
5.2　ArrayList 类 …… 120

5.2.1 ArrayList 类的方法描述 …… 122
5.2.2 ArrayList 类标题 …… 127
5.2.3 ArrayList 类中的字段 …… 128
5.2.4 ArrayList 对象可串行化性 …… 128
5.2.5 ArrayList 对象的可克隆性 …… 129
5.3 实现 ArrayList 类 …… 130
5.3.1 定义 add 方法 …… 131
5.3.2 分摊时间 …… 133
5.3.3 clone 方法和 copy 构造函数 …… 134
5.3.4 Fail-Fast 迭代器 …… 135
5.4 高精度算法 …… 136
5.4.1 设计 VeryLongInt 类 …… 137
5.4.2 实现 VeryLongInt 类 …… 138
5.5 VECTOR 类 …… 141
5.6 小结 …… 141
5.7 练习 …… 141

第 6 章 链表 …… 149
6.1 LinkedList 类 …… 149
6.1.1 比较 LinkedList 类和 ArrayList 类 …… 151
6.1.2 LinkList 迭代器 …… 153
6.1.3 LinkedList 类的字段和实现方法 …… 158
6.1.4 ListItr 类的字段和实现 …… 164
6.1.5 LinkedList 类的其他设计和实现方法 …… 167
6.1.6 循环链表 …… 169
6.2 行编辑器 …… 171
6.2.1 设计 Editor 类 …… 174
6.2.2 实现 Editor 类 …… 176
6.2.3 Editor 类方法的大 O 分析 …… 179
6.2.4 EditorDirver 类 …… 179
6.3 小结 …… 181
6.4 练习 …… 181

第 7 章 队列和堆栈 …… 187
7.1 队列 …… 187
7.1.1 Queue 类的设计与实现 …… 188
7.1.2 Queue 类可选择的设计和实现 …… 190
7.2 计算机模拟 …… 194
7.3 应用：模拟洗车 …… 195
7.3.1 CarWash 类的设计 …… 196
7.3.2 CarWash 类的实现 …… 197
7.3.3 CarWash 方法分析 …… 200
7.3.4 随机到达时间 …… 200
7.4 堆栈 …… 201
7.4.1 Stack 类的设计和实现 …… 202
7.4.2 Java 集合框架中的 Stack 类 …… 202
7.4.3 Stack 类可选的设计和实现 …… 203
7.5 应用：如何编译实现递归 …… 203
7.6 应用：中缀表达式到后缀表达式的转换 …… 207
7.6.1 后缀表示 …… 208
7.6.2 转换矩阵 …… 210
7.6.3 标记 …… 211
7.6.4 前缀表达式 …… 212
7.7 小结 …… 214
7.8 练习 …… 215

第 8 章 二叉树和二叉搜索树 …… 225
8.1 二叉树的定义和属性 …… 226
8.1.1 二叉树定理 …… 232
8.1.2 外部路径长度 …… 234
8.1.3 对二叉树的遍历 …… 235
8.2 二叉搜索树 …… 240
8.2.1 BinSearchTree 类 …… 241
8.2.2 BinSearchTree 类的字段及内置类 …… 243
8.2.3 BinSearchTree 类的实现 …… 244
8.2.4 remove 方法 …… 249
8.2.5 TreeIterator 类 …… 256
8.3 小结 …… 258
8.4 练习 …… 259

第 9 章 平衡二叉搜索树 …… 265
9.1 二叉搜索树的一个问题 …… 265
9.2 旋转 …… 266
9.3 AVL 树 …… 270
9.3.1 AVL 树的高度 …… 271
9.3.2 AVLTree 类 …… 272
9.3.3 fixAfterInsertion 方法 …… 274

9.3.4 add 方法的正确性 ····· 282
9.4 RED-BLACK 树 ····· 284
9.5 小结 ····· 290
9.6 练习 ····· 290

第 10 章 TreeMap 和 TreeSet ····· 295
10.1 TreeMap 类 ····· 295
10.1.1 TreeMap 类的方法介绍 ····· 296
10.1.2 TreeMap 类的字段 ····· 298
10.1.3 Comparator 接口和 Comparable 接口 ····· 299
10.1.4 Entry 类 ····· 300
10.1.5 TreeMap 类的实现 ····· 300
10.1.6 fixAfterInsertion 方法 ····· 302
10.1.7 Insertion 的三种情况 ····· 303
10.1.8 TreeMap 类的其他方法 ····· 307
10.1.9 fixAfterDeletion 方法 ····· 311
10.1.10 entrySet 方法 ····· 318
10.2 TREEMAP 对象：一个简单的辞典 ····· 318
10.2.1 Thesaurus 类的设计和实现 ··· 319
10.2.2 ThesaurusDriver 类的设计和实现 ····· 320
10.3 TreeSet 类 ····· 322
10.4 一个简单的拼写检查器 ····· 326
10.4.1 SpellChecker 类的设计和实现 ····· 327
10.4.2 SpellCheckerDriver 类的设计与实现 ····· 328
10.5 小结 ····· 330
10.6 练习 ····· 331

第 11 章 优先级队列 ····· 337
11.1 简介 ····· 337
11.2 PriorityQueue 接口的定义 ····· 338
11.3 PriorityQueue 接口的实现 ····· 339
11.3.1 Heap 类 ····· 340
11.3.2 Heap 类中的字段 ····· 344
11.3.3 Heap 类的实现 ····· 344
11.3.4 percolateUp 方法 ····· 345
11.3.5 percolateDown 方法 ····· 349
11.4 应用：Huffman 编码 ····· 351
11.4.1 Huffman 树 ····· 353
11.4.2 贪婪算法 ····· 356
11.4.3 Huffman 类 ····· 356
11.5 小结 ····· 361
11.6 练习 ····· 362

第 12 章 排序 ····· 367
12.1 简介 ····· 367
12.2 插入排序 ····· 368
12.3 排序能有多快 ····· 370
12.4 快速排序法 ····· 371
12.4.1 归并排序 ····· 372
12.4.2 树排序 ····· 377
12.4.3 堆排序 ····· 379
12.4.4 快速排序 ····· 383
12.5 小结 ····· 391
12.6 练习 ····· 391

第 13 章 检索和散列类 ····· 401
13.1 检索的分析框架 ····· 401
13.2 检索概述 ····· 402
13.2.1 顺序检索 ····· 402
13.2.2 二分法检索 ····· 403
13.2.3 red-black 树检索 ····· 403
13.3 HashMap 类 ····· 404
13.3.1 HashMap 类的方法描述 ····· 404
13.3.2 HashMap 类的字段 ····· 406
13.3.3 散列设计 ····· 406
13.3.4 hashCode 方法 ····· 409
13.3.5 均匀散列假设 ····· 410
13.3.6 链 ····· 411
13.3.7 HashMap 类的实现 ····· 412
13.3.8 链式散列分析 ····· 416
13.3.9 HashIterator 类 ····· 418
13.4 HashSet 类 ····· 419
13.5 开放地址散列 ····· 419
13.5.1 remove 方法 ····· 421
13.5.2 主簇 ····· 425
13.5.3 双散列 ····· 426
13.5.4 开放地址散列分析 ····· 429

13.6 小结……432
13.7 练习……432

第 14 章 图、树和网络……437
14.1 无向图……437
14.2 有向图……440
14.3 树……441
14.4 网络……442
14.5 图的算法……443
14.5.1 迭代器……444
14.5.2 连通性……449
14.5.3 产生最小生成树……450
14.5.4 在网络中寻找最短路径……454
14.6 开发 Network 类……457
14.6.1 Network 类的方法描述……458
14.6.2 Network 类的字段……460
14.6.3 实现 Network 类……462
14.6.4 Network 类的另一种设计和实现……469
14.7 突破网络……471
14.8 小结……473
14.9 练习……474

附录 A 数学背景知识……481
A.1 简介……481
A.2 函数和数列……481
A.3 求和与求积……482
A.4 对数……483
A.5 数学归纳法……485
A.6 练习……492

附录 B GUI 和 GUIListener 类……495
B.1 简介……495
B.2 线程……496
B.3 实现 Process 接口……497
B.4 GUI 类……499
B.4.1 GUI 构造函数……500
B.4.2 GUI 类中的其他方法……501
B.5 GUIListener 类……502
B.6 综合应用……503

附录 C Java 集合框架……505
C.1 简介……505
C.2 Collection 接口……505
C.3 List 接口……507
C.4 ListIterator 接口……509
C.5 Set 接口……511
C.6 Map 接口……513
C.7 ArrayList 类……516
C.8 LinkedList 类……527
C.9 TreeSet 类……543
C.10 TreeMap 类……555
C.11 HashSet 类……567
C.12 HashMap 类……575

第 1 章

Java 语言的重要特性

这是一本关于编程方面的书，本书特别关注于对数据结构和算法的理解、使用和实现。Java Collections Framework 是 java.util 程序包的一部分，其中已经实现了大量的数据结构和算法。本书后续的章节将重点讨论什么是框架(framework)，以及怎样在程序中使用框架。为了便于理解，读者首先需要熟悉本章中讲述的 Java 语言的一些重要特性。其中某些特性有些您可能已经比较熟悉；而有些可能是全新的。这些内容既是框架本身需要的，也是在编程中使用框架所必需的。

本章学习目标

(1) 复习类、对象和消息等基础知识
(2) 比较开发者眼中的类和用户眼中的类的区别
(3) 了解管理图形用户接口(GUI)中相关的事件的顺序
(4) 了解如何通过继承创建多态引用变量
(5) 能够通过创建 try 块和 catch 块来处理异常

1.1 类

由变量和作用于变量的方法组成类(class)。

类由字段以及对这些字段进行操作的方法组成。一个类在单个实体中封装了被动组件(字段)和主动组件(方法)。封装增加了程序的模块性：因为一个类和程序的其他部分隔离开来，所以程序更加容易理解、更便于修改。假设为了解决某个问题，我们需要处理日历中的日期。首先创建一个 CalendarDate 类(这个类和 java.util 程序包中的 Calendar 类或 Date 类没有任何关系)。这个 CalendarDate 类由一个或者多个保存数据的字段以及对这些字段进行操作的方法组成。开始并不需要关心怎样选择字段来表示日期。因为我们或者其他人将使用这个 CalendarDate 类，

所以需要确定 CalendarDate 类的职责。也就是说，这个类需要向用户提供什么？可能最初的职责仅仅是：

(1) 构造一个日期，给出年、月、日
(2) 确定给出的日期是否有效
(3) 返回给出日期的下一个日期
(4) 返回给出日期的上一个日期
(5) 返回给出日期是星期几
(6) 确定给出日期是否在另一日期之前

1.1.1 方法描述

方法描述提供给用户了解方法的所有信息。

类的责任高度概括在方法描述中：即用户为了调用方法所需的直接信息。每个方法描述由三部分组成：前置条件、后置条件和方法标题。前置条件(precondition)是程序在执行方法之前的状态要求。后置条件(postcondition)是在前置条件为真的前提下，程序执行方法之后的状态要求。前置条件和后置条件由调用对象和形式参数确定。

例如，isValid()方法的方法描述如下所示：

```
// Postcondition:  true has been returned if the date is legal: the year must be
//                 an integer between 1800 and 2200, inclusive; the month
//                 must be an integer between 1 and 12, inclusive; the day
//                 must be an integer between 1 and the maximum number of
//                 days for the given month and year, inclusive. Otherwise,
//                 false has been returned.
public boolean isValid();
```

这里并没有给出前置条件，因为程序在调用方法之前对状态没有任何特殊的要求。从技术角度来讲，前置条件就是 true。但是我们在这种情况下省略了前置条件。每个方法都应当实现某些任务，因此后置条件总是显式地给出。在后置条件中，thisDate 指调用对象，也就是调用这一方法的对象。在方法标题中，返回值的类型是 boolean 型，方法标识符是 isValid()，没有任何形式参数。

就类而言，对象(有时也称作类的实例)是具有类的字段的变量，可以调用类的方法。在 Java 语言中，对象总是通用引用变量来间接访问。例如，假设有如下定义：

```
CalendarDate thisDate;
```

在这里，thisDate 是 CalendarDate 类型的一个对象的引用。如果以后使用这样的语句：

```
if (thisDate.isValid( ))
      System.out.println ("The date is valid.");
else
      System.out.println ("The date is not valid.");
```

那么 thisDate 所引用的对象调用 isValid()方法。输出结果由 isValid()方法返回的 boolean 值决定。一般来说，方法调用的语法由对象引用变量加上小圆点，加上方法标识符，再加上括号中的参数列表组成。在面向对象用语中，消息(message)是指对象对一个方法的调用。例如，下面的消息返回调用对象下一天的日期：

```
thisDate.next( )
```

在这个消息中，引用变量 thisDate 引用的对象调用了 CalendarDate 类的 next()方法。术语消息(message)意味着指一条信息从程序的一部分传递到另一部分。例如，消息 thisDate.next()可能从某个类的一个方法传递，而不是通过 CalendarDate 类传递。

如果改进 CalendarDate 类的方法，那么可以得到如下的方法描述：

```
// Postcondition: this CalendarDate has been constructed from year, month
//                and day.
public CalendarDate (int year, int month, int day);

// Postcondition: true has been returned if the date is legal: the year must be
//                an integer between 1800 and 2200, inclusive; the month
//                must be an integer between 1 and 12, inclusive; the day
//                must be an integer between 1 and the maximum number of
//                days for the given month and year, inclusive. Otherwise,
//                false has been returned.
public boolean isValid( );
```

示例 假设 currentDate 是 CalendarDate 类的一个对象的引用变量。如果这个对象的字段值是“February 29 , 2000”，那么 currentDate.isValid( )将返回 true。如果对象的字段值是“February 29 , 2001”，那么 currentDate.isValid( )将返回 false。

```
// Precondition: The date is valid.
// Postcondition: The next date has been returned.
public CalendarDate next( );
```

示例 假设 date 是 CalendarDate 类的一个对象的引用变量。如果这个对象的字段值是“February 29, 2000”，那么 date.next()将返回日期“March 1, 2000”的引用变量。如果 date 引用的对象是无效的，并且调用了 date.next()方法，那么将会发生什么？无效的日期不满足前置条件，结果将不确定。也就是说，可能不返回日期，可能返回一个错误的日期，甚至可能导致程序崩溃等等。类的用户必须在调用方法之前确保方法的前置条件已经得到满足。例如，用户可以这样使用：

```
if (date.isValid( ))
. . . date.next( ) . . .
else
System.out.println ("Invalid date");
```

另外，类的开发者要确保方法描述为类的用户提供了足够的信息。

```
// Precondition: The date is valid.
// Postcondition: The previous date has been returned.
public CalendarDate previous( );

// Precondition: The date is valid.
// Postcondition: The day of the week-"Sunday", "Monday", and so on-on
//                which the date falls has been returned.
```