HZ BOOKS

PEARSON Addison Wesley

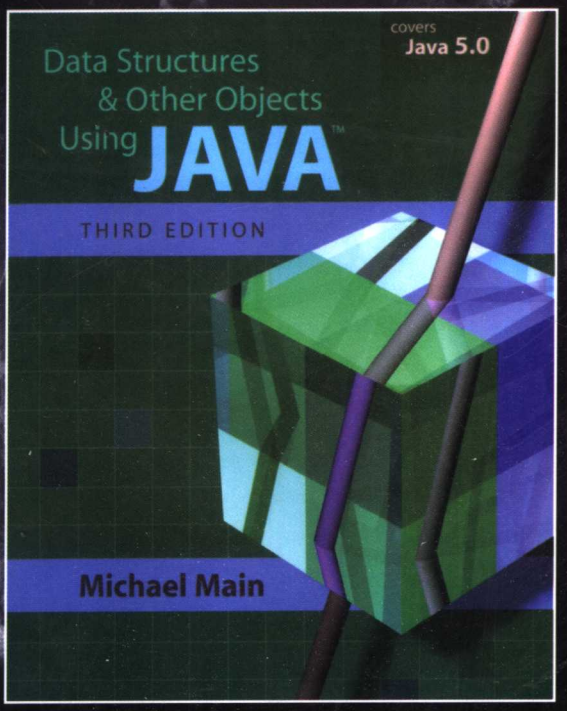
计 算 机 科 学 丛 书

原书第3版

数据结构

Java语言描述

(美) Michael Main 著 孔芳 韩月娟 译



Data Structures and Other Objects Using Java

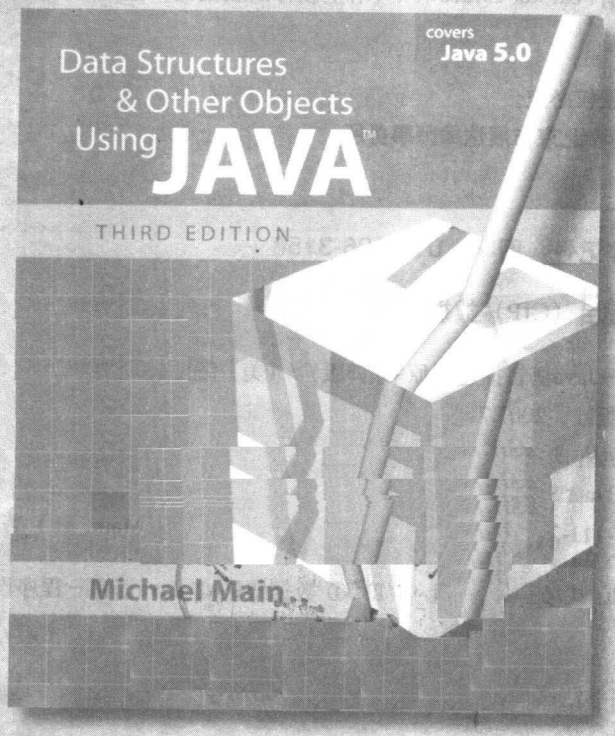
Third Edition

机械工业出版社
China Machine Press

计 算 机 科 学 丛

数据结构
Java语言描述

(美) Michael Main 著 孔芳 韩月娟 译



Data Structures and Other Objects Using Java
Third Edition

机械工业出版社
China Machine Press

本书系统地介绍各种常用的数据结构，对基本概念、基本原理和基本方法以及相关的算法设计做了深入浅出、详细和通俗的讲解。全书采用面向对象的Java语言对算法进行描述，将传统的数据结构的内容与面向对象的思想和技术完全融合，使得讲解更加贴近自然。

全书内容翔实，结构清晰合理，可作为大专院校计算机及其相关专业的有关数据结构的教材和参考书，也是计算机科学与工程领域的从业人员不可多得的一本参考书。

本书版权登记号：图字：01-2006-3158

图书在版编目（CIP）数据

数据结构——Java语言描述（原书第3版）/（美）缅因（Main, M.）著；孔芳等译. －北京：机械工业出版社，2007.7

（计算机科学丛书）

书名原文：Data Structures and Other Objects Using Java，Third Edition

ISBN 978-7-111-21553-0

Ⅰ.数… Ⅱ.①缅… ②孔… Ⅲ.①数据结构 ②Java语言－程序设计 Ⅳ.TP311.12 TP312

中国版本图书馆CIP数据核字（2007）第076516号

机械工业出版社（北京市西城区百万庄大街22号 邮政编码 100037）

责任编辑：王 玉

北京诚信伟业印刷有限公司印刷 · 新华书店北京发行所发行

2007年7月第1版第1次印刷

184mm × 260mm · 32.75印张

定价：56.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

本社购书热线：（010）68326294

译 者 序

用计算机解决任何实际问题都离不开数据表示和数据处理。数据表示和数据处理的核心问题之一就是确定数据结构并实现它。这正是数据结构课程的核心内容。

数据结构是计算机及其相关专业的主干课和专业基础课，主要介绍用计算机解决一系列问题，特别是非数值信息处理问题时所用的各种组织数据的方法、存储数据结构的方法以及对各种结构执行操作的算法。通过这门课程的学习，计算机专业人员能够掌握各种数据结构的特点、存储表示、运算方法以及在计算机科学中的具体应用。

本书系统地介绍了各种常用的数据结构，对基本概念、基本原理和基本方法做了深入浅出的介绍。同时，对相关的算法设计也做了详细和通俗的讲解。本书的特点包括：

- 使用目前非常流行的纯面向对象的程序设计语言——Java语言作为算法描述语言，将传统的数据结构的内容与面向对象的思想、技术完全融合，使得讲解更加自然贴切。
- 选择J2SE 5.0版进行算法描述。本书对J2SE 5.0中引入的类型参数等最新内容作了简明扼要的介绍，并着重介绍如何利用Java语言提供的这些最新特性进行一些通用算法的描述。通用算法正是数据结构与语言无关性的展现。
- 对各类问题的分析、设计和实现的描述非常规范，充分体现了用计算机解决实际问题的一般步骤，同时也强调了数据结构在问题解决过程中的重要角色。
- 实例丰富，自测习题较为充足，且习题涉及范围广泛，能起到很好的巩固、强化和进一步加深理解的作用。全书设计的某些实例在若干章节中反复出现，既介绍顺序方式的具体实现，又描述如何用链式方法实现。先在线性结构中作了说明，继而在非线性的树、图中再次强调。通过这种前后关联、横向比较的方式，能更好地帮助读者理解具体的数据结构，更好地体会逻辑结构、物理结构和算法描述之间的关系。

另外，本书在章节的安排上还充分考虑了读者现有的知识背景，各章相对独立，读者可根据自己的知识结构有选择地阅读学习。

全书共14章，分几个层次不断深入。前3章首先对Java语言的基础知识以及面向对象的一些基本概念作简要说明。从第4章开始介绍几种常见的数据结构（链表、栈、树、散列和图等）。在介绍各类数据结构的过程中，全书从数据结构的视角又对通用程序设计的基本思路、软件重构的基本思想作了一定的说明。

全书内容翔实，结构清晰合理，是一本有关数据结构的优秀教材，可作为大专院校计算机及其相关专业的教材和参考书。对于计算机科学与工程领域的从业人员而言，本书也是一本不可多得的参考书。

本书由苏州大学的孔芳、韩月娟主译。在翻译过程中，译者尽可能广泛地参考了最新的数据结构教材和译者的教学经验。但由于译者知识的局限性，加之时间比较仓促，译文中难免会出现一些疏漏，诚恳地希望广大读者给予批评指正。

译 者

2007年4月

前　言

Java为程序员提供了引人入胜的论坛。有了Java和World Wide Web，程序员的作品可以包含动人的图片，并能快速地实现全球分发，而且在反馈、修改、软件重用以及功能完善方面具有卓越的能力。当然，Java的快速发展也得益于它的其他品质：Java能够实现清晰的、面向对象的设计；Java的语法结构是大家所熟悉的；Java类库中包含优秀的可用特性。而World Wide Web的迷人之处使得有经验的程序员和刚进入计算机科学领域的新手都选择了Java。

也就是说，令人满意的是，尽管在课程中已经开始使用Java语言，但第一学年的核心课程仍然没有变化。虽然课程变成了相应的面向对象方法，但数据结构课程中已经被证实的用于表示和使用数据结构的方法对刚起步的程序员来说，仍然是至关重要的。

本书正是为演进中的数据结构课程编写的，其中的程序设计采用Java技术进行。本书的重点是规格说明、设计、实现，以及通常在第二学期的面向对象课程中介绍的基本数据类型的使用。本书还广泛涵盖了一些重要的程序设计技术，例如递归、查找和排序。书中还介绍了算法的大O时间分析，在选读的附录中还介绍了如何编写交互式的applet来测试数据结构的实现，以及使用Javadoc指定前置条件和后置条件契约。

本书假设学生已经修完了计算机科学导论和程序设计课程，本书的内容还包含前导课程中没有完全覆盖的主题（例如Java的Object类型，并对参数传递的精确描述）。前言的其余部分将从第3版增加的新内容开始，逐一介绍本书的内容。

第3版：J2SE 5.0

这一版中的新素材来源于Sun Microsystems公司最新发布的Java 2标准版5.0（J2SE 5.0）。这一版中的许多特性对以前需要特别处理的数据结构概念提供了支持。要特别说明的是，在本书的新版中介绍了如下一些很有用的特性：

真正的通用（在第5章介绍）。在J2SE 5.0中，最大的变化是通过不指定基本的数据类型为通用方法和通用类提供支持。例如构建一个由E类型的对象构成的集合类时就可以使用通用形式，此时数据类型E并未指定。通用是在第5章中介绍的，而在本书的老版本的同样位置，我们使用的是Java的Object类作为元素的类型，实现了通用的一种简单形式。在介绍通用之前，本书已经介绍了两个基本的集合类，它们是由指定类型的元素使用数组或链表方式实现的。撇开通用，第一次介绍这两个基本实现时，学生主要学习的是像构建副本和操作链表这样的重要概念，而不被“通用”等这类问题干扰。第5章结束后，通用类就直接引入到相关内容中了。

非常有趣的是，J2SE 5.0中通用的运行时实现是使用Java 的Object作为元素类型的，这与J2SE 5.0推出之前，程序构建通用类的方法相同。不同的是Java通用的额外语法进行新的编译时类型检查，以防止多种类型混杂（例如，将一个字符串插入到了某个整数集合中）。

输入/输出（在第1章和附录B介绍）。全书使用格式化输出（使用System.out.printf方法）和基本类型值的输入（使用java.util.Scanner类）。本书老版本中使用的输入和输出类（EasyReader和FormatWriter）仍然可以从网址www.cs.colorado.edu/~main/edu/colorado/io处获得。

变量数量可变的方法（在第3章中介绍）。在J2SE 5.0中，一个方法可以包含可变数量的参数。尽管它在数据结构中并未起到很重要的作用，但在本书的第3章中仍对这一内容作了介绍。

增强型for循环（在第3章和第5章的相关部分介绍）。J2SE 5.0引入了一种新的for循环形式，可以非常容易地迭代遍历数组中的元素、由实现了Iterable接口的元素构成的集合中的元素，以及枚举类型的元素。不论这种for循环形式是否合适，本书通篇都使用了这一新形式。

基本类型值的自动包装和自动解包装（在第5章中介绍）。自动包装能将基本类型的值（例如一个int值）自动转换成一个Java对象（例如Integer类的对象）。在大多数情况下，自动解包装能将某一对象转换回基本类型的值。由于程序的语法比较清晰，所以运行时类型强制转换仍然存在。

共变的返回类型（在第13章中介绍）。J2SE 5.0允许共变返回类型。共变返回类型是指重写继承的方法时，返回类型现在可以是原返回类型的任意子类。全书使用这一思想编写clone方法，这样可以避免每次使用clone方法都需要的类型强制转换，而且增加了类型的安全性。我们将从继承章开始大量使用这一技术。

枚举类型（在第13章中介绍）。枚举类型提供一种定义新数据类型的便捷方法，其构成对象选取的是离散值的一个较小集合。

除了上述这些新语言特性，本书的第3版与前几版一样，介绍了基本数据结构的相关知识。特别需要说明的是，数据结构课程强调的是规格说明、设计和分析与特定语言无关的数据结构的能力，以及在现代语言中实现新的数据结构和使用数据结构的能力。前4章将采用这样的方法，学生首先复习Java语言，然后继续学习，使用很容易应用于各类高级语言的数组和链表技术来实现大多数基本数据结构（包和序列类）。

第5章的内容有点脱离主题，介绍了几种Java特有的技术，每种技术都与数据结构紧密相关：如何基于Java的新通用类型构建通用集合类；使用Java接口和API类；构建和使用Java的迭代器。尽管接下来的大部分内容与Java特性无关，但此时学习这些内容是非常自然的。

接下来的章节重点讨论了特定的数据结构（栈、队列、树、散列表和图）或者程序设计技术（递归、排序技术、继承）。直到开始学习树之前才介绍递归，这样可以使用一些简单但很有意义的例子来介绍递归，而这些例子不仅仅完成尾递归。这样就避免了一些学生在刚学习递归时总将它误认为是某种神奇的循环。在介绍完开始的这些简单例子后（包括这一版的一个新例子），我们将学习包含两个递归调用的大型例子，其中会涉及二叉树的常见状态。

本书特别强调了树，第10章给出使用平衡树数据结构的两个重要例子（堆项目和B树项目）。树以及其他方面的附加项目也已经添加到http://www.aw.com/cssupport中。

全书根据需要在第2版的基础上又添加了其他一些新的Java特性：使用断言检查前置条件，对继承进行更完整的介绍，构建新的抽象类的例子，并完成诸如Othello或Connect Four这样的双人策略游戏的类，对Java API中的新类（通用形式的ArrayList、Vector、HashMap和HashTable）以及Javadoc的新特性作了介绍。

这些Java新技术使用起来非常方便，学生们需要及时更新知识——但设计、规格说明、编制文档、实现、使用和分析数据结构的方法对学生而言是具有持久影响的。

每种数据类型的5个步骤

本书的核心内容是一些知名的数据类型，包括：集合、包（或多集合）、顺序表、栈、队列、表和图。此外，书中还另外补充了一些数据类型，例如优先队列。有些数据类型使用多种方法实现。例如包，首先通过将数据元素存储在数组中实现，随后又使用二叉查找树再次实现。每种数据类型都将按照下列

5个步骤的模式介绍。

第1步：抽象地理解数据类型。在这一层次，学生在概念和图形上对数据类型以及操作进行理解。例如，学生可以可视化栈，理解元素的出栈和入栈操作。理解简单的应用程序，并手工执行，例如使用栈来反转单词的字母次序。

第2步：以Java类的方式编写数据类型的规格说明。在这一步中，学生将会看到并学会如何为一个实现数据类型的Java类编写规格说明。使用Javadoc工具编写的规格说明包含构造函数的头部、公共方法和其他一些公共特性（例如受内存限制的约束）。每个方法的头部和前置条件/后置条件一起呈现，完整地指定方法的行为。在这一层次，让学生意识到规格说明与任何选定的特定实现技术无关是非常重要的。实际上，相同的规格说明可多次应用于同一数据类型的多种不同实现。

第3步：使用数据类型。有了规格说明，学生可以编写简单的应用程序或applet来演示数据类型的使用。这些应用程序仅仅建立在数据类型的规格说明之上，而规格说明还没有进一步实现。

第4步：选择相应的数据结构，继续设计和实现数据类型。在很好地理解抽象数据类型后，就可以选择相应的数据结构了，例如数组、由结点构成的链表，或者由结点构成的二叉树。对于许多数据类型，首先选择像数组这样的简单方法来设计和实现，然后再使用复杂的底层结构重新设计和实现相同的数据类型。

由于使用了Java类，数据类型的实现将会把选择的数据结构（数组、引向其他对象的引用，等等）作为类的私有实例变量。在教学过程中，让学生清楚地理解把私有实例变量与数据类型的抽象概念相关联的规则是十分必要的。要求每个学生用英文句子清晰地写出这些规则，这些规则称为*抽象数据类型的不变式*。一旦编写好了不变式，学生就可以继续实现各种方法。不变式有助于编写正确的方法，其原因有两个：(a) 在方法开始运行时，每个方法（除了构造函数）都知道不变式为true；(b) 在方法运行结束时，每个方法都负责确保不变式再次为true。

第5步：分析实现。可以分析每一实现的正确性、灵活性，并能对操作进行时间分析（使用大O表示法）。当相同数据类型以几种不同方式实现时，学生就有很好的机会能够进行分析。

本课程结束时，学生能学到什么？

在本课程结束时，学生能够彻底理解数据类型。知道如何使用数据类型以及如何通过多种方式来实现它们；明了选择不同实现的实际效果；能够通过大O分析推导算法效率，并能利用ADT不变式证明实现的正确性。

本书的重要目标之一是增强学生在规格说明、设计和实现方面的体验，帮助学生提高程序推理的能力。但最重要的可能是展示了许多在各类情况下都能使用的类。这样，学生无需从头开始编写所有的一切。我们告诉学生，可能某天他们正在考虑一个问题，突然之间，他们意识到大量工作都可以使用包、栈、队列等来实现。他们不再需要完成这其中的大量工作，而是可以直接利用本学期编写的包、栈或队列，甚至可以不做任何修改。更有可能的是，他们将使用标准数据类型库中熟悉的数据类型，例如推荐的Java类库。实际上，本书的一些数据类型是JCL的精简版本，所以，当学生开始真正学习JCL时，从如何使用类到构建类时需要考虑的因素，他们都会觉得非常熟悉。

其他基本主题

贯穿整个课程，我们还为“真实程序设计”的其他方面打下了基础，除了基本数据结构方面的内容之外，我们还介绍了下列主题。

面向对象程序设计。本书通过让学生深入理解Java类来为面向对象程序设计打下基础。首先介绍类的一些重要方面：方法的概念，私有和公共成员的区别，构造函数的用途，并展示了如何生成副本和如

何进行相等性测试。这些内容主要在第2章中介绍，在CS1课程中已经很好理解了Java类的学生可以跳过这部分内容。

当类中首次使用动态数组（第3章）时将会介绍类更深一层的内容。这里介绍了使用较复杂的clone方法的必要性。在讲述OOP方面的知识时第一次使用了动态内存，这让学生对如何通过引用将实例变量引向一个动态对象有了具体的印象，例如数组。

从概念上讲，OOP最大的革新在于通过继承实现了软件重用。当然可以在数据结构课程的一开始就介绍继承（例如将一个集合类以包类的子类的形式实现）。然而，过早介绍继承也会带来一定的麻烦，让学生一下子接受过多的新概念，会使他们没有精力去深刻理解基本数据结构。因此，在数据结构课程中总是最后介绍继承，让学生认识这是一种新事物。但是也可以在理解了类之后立即介绍继承（13.1节和13.2节）。出于这种想法，有些教师会提前讲解第13章，即在栈和队列之前讲解，这样就能从另一个类派生出栈和队列。

另一个方法是把那些早就掌握了类的基本知识的学生分离出来，让他们直接去完成继承项目（例如13.2节的生态系统），而其余的学生则首先学习类。

Java对象。Java中的Object类型是Java中其他所有（或者几乎所有）类型的基础。8种基本类型不是Java对象，而且对于许多学生来说，CS1课程中的大部分工作主要是使用这8种基本类型完成的。正因如此，最先介绍的数据结构是基本类型的值构成的集合，例如由整数构成的包，或者由double型数值构成的序列。

迭代器。迭代器是Java类库的一个重要组成部分，通过使用它，程序员能够非常方便地遍历集合类中的各个元素。本书在第5章中介绍了Iterator接口。虽然在程序设计项目中提供了很好的使用迭代器的机会，例如使用栈来实现二叉查找树的迭代器（参见第9章），但在本书的其余部分并没有直接使用迭代器。

递归。第一学期的课程通常会给学生介绍到递归。但是，第一学期课程中的许多例子都是尾递归，也就是在方法的最后一步进行递归调用。这可能会误导学生，让他们错误地认为递归仅仅是一个循环。正因如此，我会尽量避免在第二学期的课程中过早使用尾递归。

因此，在第二学期的课程中，我强调的是递归解决方案，而不是尾递归的调用。在“递归思想”这一章中连续给出了4个例子，都是基于这一思想的。学生对其中的两个例子——产生随机分形和走迷宫非常感兴趣。其中分形的例子是以图形化的applet的形式给出的，而迷宫的例子虽然是基于文本的，但富于创造精神的学生可以将它转换成图形方式的applet。这些递归示例（第8章）出现在树（第9章）之前，这是因为在递归树算法中，递归是至关重要的。然而，如果教师想要强调递归，可以将这一章的内容提前，甚至可以提前到第2章之前。

如果课程时间充足，还可以学习高级树项目（第10章），在这一章中，我们将分析递归树算法，解释保证树平衡的重要性——它能提高最差性能，并能避免潜在的执行栈溢出。

查找和排序。第11章和第12章将介绍查找和排序的基本知识。查找章节回顾了基于有序数组的二分查找，这许多同学可能以前已经看到过了。在查找章节中还介绍了散列表，实现了JCL中散列表的一个版本。另外又使用链表代替开型寻址实现了JCL hash表的另一个版本。排序一章介绍了简单的二次型排序方法，但此章的绝大部分内容都集中在快速算法上：递归实现的归并排序（最坏情况耗时为$O(n \log n)$）、Tony Hoare的递归快速排序（平均耗时为$O(n \log n)$），以及基于树的堆排序（最坏情况耗时为$O(n \log n)$）。

高级项目

本书提供了许多可选项目，这些项目可以由高级班或一些在大型类方面具有良好背景知识的学生来

完成。这些高级项目包括：

- 适用于所有数据结构的基于applet的交互式测试程序（参见附录I）。
- 为序列类实现Iterator接口（参见第5章的程序设计项目）。
- 编写集合类的深度clone方法（参见第5章的程序设计项目）。
- 编写某个应用程序的applet版本（例如8.1节的迷宫问题，或者13.2节的生态系统）。
- 使用栈为二叉查找树构建迭代器（参见第9章的程序设计项目）。
- 使用普通队列数组（7.4节）或堆（10.1节）实现优先队列。
- 使用B树实现集合类（10.2节）。我特别重视这个项目，在实现这个项目时，应该为学生提供实现类的充足信息，这样学生就不需要参阅其他资料了。某些优秀的学生能够独立完成该项目。
- 像13.2节的生态系统这样的继承项目。
- 第14章的图形类，以及相关的图形算法。优秀的学生也可以独立完成这些项目。

Java语言版本

本书的所有源代码，包括诸如通用等新特性在内，都已在Java 2标准版5.0中测试过，并能正常工作。有关Sun Microsystems公司的所有Java产品信息，请访问http://java.sun.com/products/index.html。

主题次序的灵活性

本书在编写上给教师留了一定的回旋余地来调整教学内容的次序，以满足特定知识背景的学生的需求，或者对选中的主题提前强调。后面给出的“章节关系图”表明了各章节间的相互依赖关系。两个方框间的连线表明，上面的方框应该在下面的方框之前介绍。

下面对内容的次序给出一些建议：

典型课程。首先对于第1～9章，如果学生以前已经具有Java类的知识，则可跳过第2章。大多数章节只需要一周的时间介绍，但第4章（链表）、第8章（递归）或者第9章（树）可能需要更多的时间。通常，介绍第1～9章的内容我需要花13周的时间，包括考试时间以及介绍链表和树所需的额外时间。剩余的几周可用于介绍第10章中的树项目或二分查找（11.1节）以及排序（第12章）。

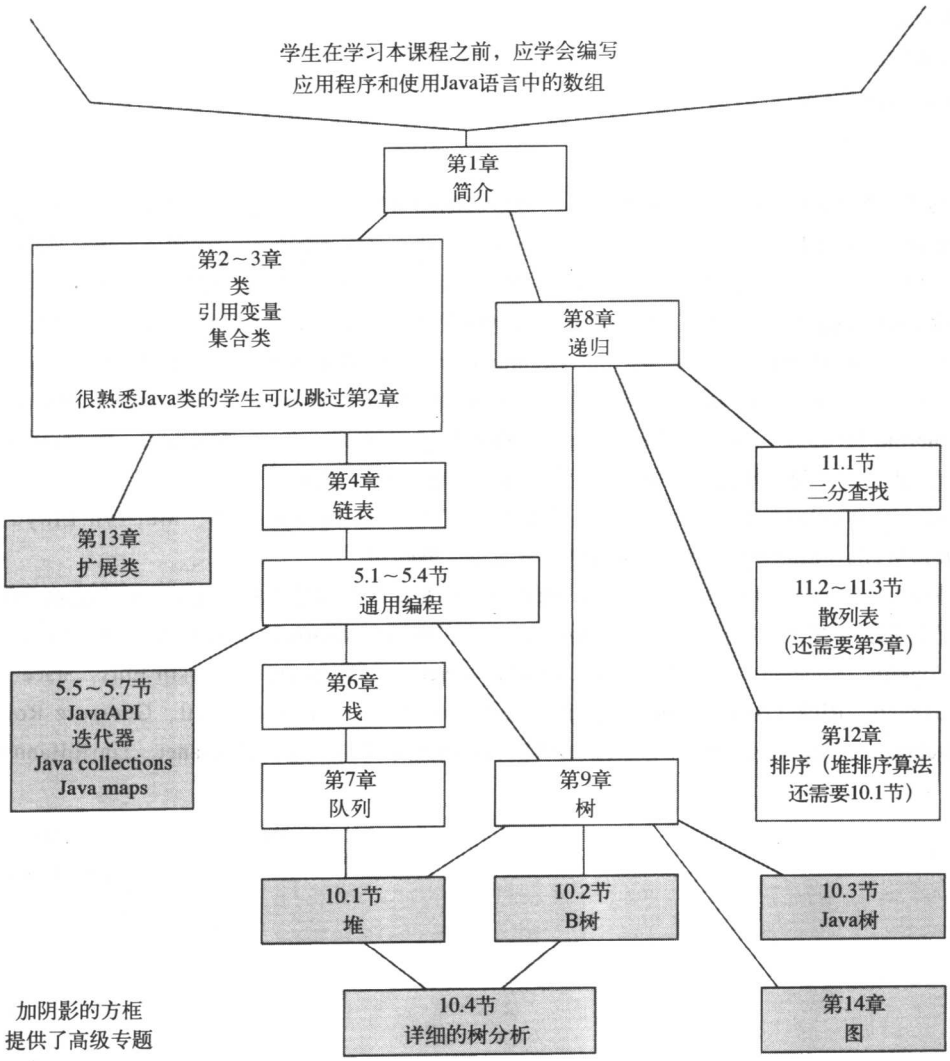
重点学习面向对象程序设计。如果学生在其他课程中已经学习过查找和排序的相关内容，那么就有时间重点学习面向对象程序设计了。前三章需要详细介绍，然后介绍继承类（13.1节）。到此为止，学生可以完成13.2节中介绍的生态系统这一有趣的OOP项目了。然后介绍基本数据结构（第4～7章），其中队列以继承类的形式实现（13.3节）。最后完成递归（第8章）和树（第9章）的内容，此时侧重于递归方法。

加速课程。将前三章作为第一周的自学内容，然后从第4章（链表）开始讲授。这样，将会在学期末空出两到三周的时间，以便学生可以在查找、排序以及其他高级主题（章节关系列表中的阴影部分）方面花费更多的时间。

也可以在讲课时省略栈和队列，以便进一步加快速度（让学生自学有关章节）。

提前讲授递归/提前讲授排序。课程开始的前一到三周介绍递归思想。然后首先阅读第1章和第8章的内容，也可以补充一些额外的递归项目。

如果提前介绍递归，那么在介绍集合类之前可以继续介绍二分查找（11.1节）和大多数排序算法（第12章）。



章节关系图

通过Internet进行补充

本书的读者可以访问网站www.cs.colorado.edu/~main/dsoj.html获得下列资源：

- 源代码
- 勘误表

如下资料是专门提供给采用本书作为教材的教师的教辅资源[⊖]：

- PowerPoint演示文稿
- 考试题及其答案
- 部分程序设计项目的解决方案

⊖ 请需要者与培生教育出版集团北京办事处联系，联系电话是：（8610）88819178或8008100855，也可发邮件到service@pearsoned.com.cn。——编者注

- 教学记录
- 示例作业
- 推荐的教学大纲

致谢

本书是我与Walter Savitch合作的结果，他是我的同事、朋友，一直以来都满腔热情地支持着我。在教学过程中，位于Boulder的Colorado大学的学生给了我很多启示和乐趣，特别是参加秋季研究班和周五下午课程的学生，他们学习的内容远远超出了数据结构。漂亮的封面图像是由一年级学生Benjamin Barrow设计的，他不时参加周五下午的课程，并利用空余时间编写了一个射线跟踪程序。在这一版的修改过程中，本书已由Philip Barry、Arthur Crummer、Herbert Dershem、Greg Dobbins、Zoran Duric、Dan Grecu、Scott Grissom、Bob Holloway、Rod Howell、Danny Krizanc、Ran Libeskind-Hadas、Meiliu Lu、Catherine Matthews、Robert Moll、Rebert Pastel、Don Slater、Ryan Stansifer、Deborah Trytten和John Wegis进行了广泛的审阅。非常感谢这些同事的出色评论和鼓励。

Addison-Wesley的编辑和员工给予了极大的支持。Michael Hirsch、Marilyn Lloyd、Joyce Consentino Wells和Lindsey Triebel每天都给予我建议、鼓励和支持。

在第3版的编写过程中，同事John Gillett的建议和友谊是非常珍贵的。此外，还要感谢我的朋友和同事，他们每天都给予我支持和鼓励：Rebecca Bates、David Benson、Zachary Bergen、Cathy Bishop、Lynne Conklin、Vanessa Crittenden、Andrzej Ehrenfeucht、Paul Eisenbrey、Skip Ellis、Stace Johnson、James Garnett、Rick Lowell、Jim Martin、Tandy McKnight、Marga Powell、Grzegorz Rozenberg、Bobby Schnabel、Susan Hartman Sullivan、Ed Bryant和他的NCWW组，以及Janet、Tim和Hannah。

Michael Main(main@colorado.edu)
Boulder，Colorado

目　录

译者序
前言
第1章　软件开发周期 ……1
1.1　规格说明、设计和实现 ……2
1.2　运行时间分析 ……10
1.3　测试和调试 ……17
本章小结 ……22
自测练习参考答案 ……23
第2章　Java类和信息隐藏 ……24
2.1　类及其成员 ……24
2.2　类的使用 ……32
2.3　包 ……38
2.4　参数、equals方法和clone ……41
本章小结 ……58
自测练习参考答案 ……58
程序设计项目 ……59
第3章　集合类 ……64
3.1　Java数组简介 ……64
3.2　整数包的ADT ……70
3.3　程序设计项目：序列ADT ……90
3.4　程序设计项目：多项式 ……99
本章小结 ……100
自测练习参考答案 ……101
程序设计项目 ……103
第4章　链表 ……106
4.1　链表基础 ……106
4.2　操作结点的方法 ……108
4.3　操作整个链表 ……117
4.4　使用链表实现的包ADT ……132
4.5　程序设计项目：使用链表实现的序列ADT ……143
4.6　复杂链表 ……148
本章小结 ……150
自测练习参考答案 ……150
程序设计项目 ……152
第5章　通用程序设计 ……155
5.1　Java的Object类型 ……155
5.2　Object方法和通用方法 ……159
5.3　通用类 ……161
5.4　通用结点 ……174
5.5　接口和迭代器 ……176
5.6　实现Iterable接口的通用包类（选读） ……182
5.7　Java中Collection和Map接口简介（选读） ……184
本章小结 ……190
自测练习参考答案 ……191
程序设计项目 ……193
第6章　栈 ……194
6.1　栈简介 ……194
6.2　栈的应用 ……197
6.3　栈ADT的实现 ……206
6.4　更复杂的栈应用 ……213
本章小结 ……219
自测练习参考答案 ……219
程序设计项目 ……219
第7章　队列 ……222
7.1　队列简介 ……222
7.2　队列的应用 ……225
7.3　Queue类的实现 ……236
7.4　优先队列 ……247
本章小结 ……250
自测练习参考答案 ……250
程序设计项目 ……251
第8章　递归思想 ……254
8.1　递归方法 ……254
8.2　递归的研究：分形和迷宫 ……261
8.3　递归的推导 ……273
本章小结 ……278
自测练习参考答案 ……278
程序设计项目 ……279

第9章 树 ……283
9.1 树的简介 ……283
9.2 树的表示方法 ……286
9.3 二叉树结点类 ……289
9.4 树的遍历 ……302
9.5 二叉查找树 ……312
本章小结 ……322
自测练习参考答案 ……323
程序设计项目 ……324
第10章 树项目 ……327
10.1 堆 ……327
10.2 B树 ……331
10.3 Java对树的支持 ……346
10.4 树、日志和时间复杂度分析 ……350
本章小结 ……354
自测练习参考答案 ……354
程序设计项目 ……355
第11章 查找 ……356
11.1 顺序查找和二分查找 ……356
11.2 开型寻址散列 ……363
11.3 使用Java的Hashtable类 ……374
11.4 链式散列 ……375
11.5 散列的耗时分析 ……377
本章小结 ……379
自测练习参考答案 ……379
程序设计项目 ……380
第12章 排序 ……382
12.1 二次排序算法 ……382
12.2 递归排序算法 ……391
12.3 一个使用堆的$O(n \log n)$算法 ……403
本章小结 ……409
自测练习参考答案 ……410
程序设计项目 ……410
第13章 使用扩展类实现软件重用 ……412
13.1 扩展类 ……412
13.2 通用类型参数和继承 ……419
13.3 模拟生态系统 ……420
13.4 抽象类和Game类 ……434
本章小结 ……442
进阶阅读 ……442
自测练习参考答案 ……443
程序设计项目 ……444
第14章 图 ……445
14.1 图的定义 ……445
14.2 图的实现 ……449
14.3 图遍历 ……458
14.4 路径算法 ……464
本章小结 ……472
自测练习参考答案 ……472
程序设计项目 ……473
附录A Java的基本类型和算术溢出 ……474
附录B Java输入和输出 ……476
附录C 抛出和捕获Java异常 ……479
附录D ArrayList、Vector、Hashtable和HashMap类 ……482
附录E 用于链表的结点类 ……485
附录F 一个用于包对象的类 ……491
附录G 深入大O表示法 ……496
附录H Javadoc ……498
附录I 用于交互式测试的Applet ……503

第1章　软件开发周期

学习目标

- 使用javadoc编写某个方法的完整规格说明，包括前置条件/后置条件契约。
- 确定简单算法中的二次型、线性型，以及对数型运行时行为，并能使用大O表达式来描述这一行为。
- 创建并确定适用于某一问题的测试数据，包括测试边界条件和完全地测试代码所需的测试数据。

开篇第1章首先介绍了事情的来龙去脉，也就是如何、为何、何时以及何地曾经存在过问题。

Noel Langley

《The Land of Green Ginger》

本章介绍了软件开发的各个阶段。包括本章将要介绍的各个小程序在内，这些阶段在所有软件的开发中都会出现。在后续的章节中，在这些小应用程序的基础上，我们将会把软件开发周期应用于有组织的数据集合。这些有组织的数据集合称为**数据结构**（Data Structure），而本书的主题正是紧紧围绕着介绍用于表示和操作这些数据结构的实际技术展开。

数据结构

数据结构是数据的集合，一般均组织成能通过某些固定的技术来存储和检索的数据项形式。例如，Java数组就是一种简单的数据结构，它可以根据为每一数据项分配的下标（[0]，[1]，[2]……）来进行各个数据项的存储和检索。

贯穿全书，你将会看到多种形式的数据结构，它们各自的组织形式分别是出于易用性，或者加快插入到结构中以及从相应的结构中删除数据项的速度的考虑。

几年后，你可能是在某个专门领域中编写大型系统的软件工程师，或许是计算机图像领域，或许是人工智能领域。未来的应用将会是非常刺激和令人激动的，而在工作中，你仍将会用到现在学习和实践的各种数据结构，仍将遵循在设计和实现第一个程序时学到的相同的软件开发周期。下面是软件开发周期所包含的各个典型阶段：

软件开发周期

- 任务的规格说明
- 解决方案的设计
- 解决方案的实现（编码）
- 解决方案的分析
- 测试和调试
- 系统的维护和演变
- 系统废弃

不需要牢记这一列表，贯穿全书，对这些阶段的实践会比纯粹的记忆更能让你熟悉它们。而且牢记这一形式化的列表也是一种误导，因为它暗示着软件开发总是由一个接一个出现的独立的步骤组成。实际上，这些阶段是彼此渗透的。例如，在开始编码之前，解决方案的效率分析可以结合设计同时进行，也可以将低级的设计决策推迟到实现阶段再进行。而且，这些阶段可能不是一个接一个地进行的。通常在各个阶段之间存在一个反复的过程。

大多数软件开发工作并不依赖于任何特定的程序设计语言。规格说明、设计和分析都可以在与特定程序设计语言没有或很少有联系的情况下进行。尽管如此，涉及实现细节时，我们确实需要选定某一特定的程序设计语言，本书使用的语言是Java™ 2 Standard Edition 5.0(J2SE 5.0)。

开始本课程之前，对Java语言，读者应该了解哪些

Java语言是由Sun Microsystems公司的一组程序员于1991年构思设计的。这个开发小组由James Gosling领导，最初的设计代号为Oak，目标是设计出一种语言，由它开发的程序可以很容易地在不同机器之间移植。接下来的4年中，Sun公司的许多其他程序员也加入了这一项目，Gosling的Oak演变成了Java语言，包括目前的Java 2 Standard Edition 5.0。Java的易移植性目标是通过引入一种**叫作字节码**(byte code)的中间形式实现的。要运行一个Java程序，该程序首先转换成字节码，然后将这些字节码交给一台运行了名为Java运行时环境（Java Runtime Environment，JRE）控制程序的机器解释执行。因为对各式各样的机器来说，JRE是可以免费获得的，所以Java程序可以在不同的机器间进行移植。由于JRE控制了所有Java程序的运行，因而它还增加了系统的安全级别，可以避免由于运行未知程序而产生的一些潜在问题。

除了最初设定的程序可移植性和安全性的目标外，Java的设计者还整合了其他一些现代程序设计语言的思想。其中最引人注目的是，Java以一种部分类似于C++程序设计语言的方式支**持面向对象的程序设计**（Object-Oriented Programming，OOP）。OOP是一种主张使用信息隐藏和组件重用策略的程序设计方法。本书将介绍这些重要的OOP原理，读者可以在自己的设计和实现中运用这些原理。

提示：本书将对OOP原理中的信息隐藏和组件重用进行介绍。

本书中的所有程序都已在Sun公司J2SDK（Java 2 Standard Development Kit）平台上运行通过，而且许多其他Java编程环境也可以成功地运行这些源代码。读者应该能非常熟练地在所使用的环境中编写、编译和运行一些短小的Java应用程序，还应该知道如何使用Java基本数据类型（数值型、字符型和布尔型），并且应该能够使用数组。

本章的剩余部分将会为读者解决Java中的数据结构问题做一些准备。1.1节将集中讨论用于确定程序行为的技术，同时还会给出一些关于设计和实现的提示。1.2节描述了一种特定类型的分析：程序的运行时间分析。1.3节介绍了一些测试和调试Java程序的相关技术。

1.1 规格说明、设计和实现

首先从一系列可能会变化的设计决定开始，然后设计各个模块，使得模块间能够隐藏这样的设计决定。

D.L. Parnas

《On the Criteria to Be Used in Decomposing Systems into Modules》

提示：读者应该已经知道如何在某些编程环境中编写、编译和运行短小的Java程序。

下面通过一个软件开发的实际例子来介绍某一特定问题的规格说明、设计和实现。**规格说明**是问题的准确描述；**设计**阶段确定解决问题的步骤（或**算法**）；**实现**是实施设计的具体Java代码。

要考虑的问题是显示一张表格，将摄氏温度转换成华氏温度，类似于下面给出的表格。对于简单问题，预期的输出示例就是一个合理的规格说明。这样的示例是准确的，对程序必须完成的任务没有任何疑问。下一步是设计一个解决方案。

温度转换表

摄氏温度	华氏温度
-50.00C	
-40.00C	
-30.00C	
-20.00C	此处显示与摄氏温度等价的华氏温度
-10.00C	
0.00C	
10.00C	
20.00C	
30.00C	
40.00C	
50.00C	

算法是解决问题的过程或指令序列。例如，对于上述温度转换问题，算法将说明如何产生这一转换表。一个算法可以有多种表示方法，例如使用自然语言表示，或者混合使用自然语言和数学符号表示，或者混合使用自然语言和某种程序设计语言表示等。自然语言和某一程序设计语言的混合体称为**伪代码**(pseudocode)。使用伪代码可以忽略程序设计语言的细节，而这些细节可能会混淆简单的解决方案。另一方面，当代码非常清楚时可以直接使用Java代码（或其他程序设计语言）。务必牢记，使用伪代码的原因是为了提高程序的清晰性。

算法

*算法是解决问题的过程或指令序列。所有算法都可以用多种方式进行表达：用自然语言表达、用某种特定的程序设计语言来表达，或者（最常见的情况）用自然语言和程序设计语言的混合体（称为**伪代码**）来表达。*

本书将使用伪代码设计上述温度转换问题的解决方案，其中还会用到非常重要的问题分解设计技术，下面就将讨论这一技术。

设计技术：分解问题

关键设计概念

将一个任务分解成多个子任务，然后再将每个子任务分解成多个更小的子任务。

设计算法的一种优秀技术是将问题划分成多个子任务，然后将每个子任务分解为较小的子任务，再将这些较小的子任务替换为更小的子任务，依此类推。最终每个子任务都非常之小，可以容易地使用Java或者你所使用的任何其他语言来实现它们。用Java代码实现算法后，每个子任务都是以独立的Java方法编写的。在其他一些程序设计语言中，方法又被称为“函数”或“过程”，但它们都可归结为相同的原理：大的问题可以分解成多个子任务，而子任务可以用独立的程序片段的形式实现。

例如，上述温度转换问题至少可以分解成两个子任务：将摄氏温度转换为华氏温度，以及按指定的精度（比如舍入到百分位）输出转换后的数值。针对这两个子问题可以设计得到如下所示的第一份伪代码草案：

1. 在表格的顶部显示标签。
2. 对于表格中的每一行（使用变量celsius和fahrenheit）
   2a. 将变量celsius设置成表格中的下一个摄氏温度。
   2b. fahrenheit= 根据celsius转换得到的华氏温度。
   2c. 输出结果表格中的一行，每个温度都精确到百分位，并给出相应的标识（分别用字符C或F标识）。
3. 输出表格底部的一行虚线。

其中带有下划线的步骤（2b和2c）是主要的子任务。但分解这一问题是否还有其他的办法呢？恰当的分解有些什么特点？一个主要的指导方针是，子任务应该有助于产生短小的伪代码，即用不超过一页的简洁描述来解决整个问题，理想情况下应该远远少于一页。在最初的设计中，选择恰当的子任务时还必须得考虑以下两个方面：代码重用的可能性，以及未来修改程序的可能性。下面看看我们的子任务是如何体现这些考虑的。

步骤2c是一个公共任务：以指定精度打印输出一些信息。该任务非常通用，以至于在最新版本的Java中已经包含了一个新方法——System.out.printf。任何程序都可以使用这一方法进行格式化输出。实现步骤2c时我们将会介绍并使用这一函数。

方法printf是一个**代码重用**的例子，它可以应用于需要完成类似任务的多个程序中。在Java中有许多包含了可重用方法的包，此外程序员还经常在一些包中创建自定义的Java方法，这些方法可以在许多

不同的应用程序中一遍遍地重复使用。

分解问题还可以产生易于理解的最终程序，而且会使得后续的维护和修改变得相对容易。例如：上述温度转换程序以后可能要修改成将摄氏温度转换成开氏温度，而不是转换成华氏温度。由于这一转换任务是由独立的Java方法完成的，所以大多数的修改都会限制在这一方法内。代码易于修改是至关重要的，因为现实世界的研究表明，程序员的大部分时间都花在维护和修改现有的程序上。

要使问题分解能产生易于修改的代码，所编写的Java方法必须名副其实地彼此分离。关于“名副其实地分离”这一概念，可以通过一个类比进行说明。假设需要将一袋金币移到一个安全又隐蔽的地方。如果袋子过重而无法搬动，就可能会将这袋金币分成三个小袋，然后一袋一袋地搬。除非是喜剧中的某个角色，否则不会试图一次搬运三个袋子。那样就违背了将一袋金币分成三份的初衷。这个策略仅适用于一次搬运一个袋子的情况。问题分解过程中发生的状况与此类似。如果把编程任务分解成三个子任务，然后编写三个不同的Java方法来分别解决这三个子任务，那么就将一个复杂的问题转换成了三个相对容易的问题。只要独立地设计各个方法，整个工作就会变得更加容易。编写某个方法时，无需关注其他方法是如何完成任务的。但方法之间确实会交互，因此在设计某一方法时，的确需要了解其他方法所完成的任务。其中的诀窍在于*只需知道需要知道的内容*，这就是**信息隐藏**。有一种信息隐藏技术，可以使用*前置条件*和*后置条件*来指定方法的行为，这将在下面部分进行讨论。

如何编写Java方法的规格说明

用Java语言实现某个方法时会给出该方法执行计算的完整指令。但在伪代码中*使用某一方法*，或在编写其他Java代码时，只需考虑该方法*完成什么工作*，而不必知道它是*如何*完成这些工作的。例如，假设需要编写一个温度转换程序，已知可以使用下列方法：

```
//将摄氏温度c转换成华氏温度
public static double celsiusToFahrenheit(double c)
```

该方法的上述信息称为**签名**（signature）。**签名**包括方法名（celsiusToFahrenheit）、方法的参数列表（double c）、方法的返回类型（double），以及修饰符（public、static）。

程序中有一个名为celsius的double型变量，它存储了一个摄氏温度值。知道了这点就可以自信地编写下列语句完成摄氏温度到华氏温度的转换，并将结果存储在一个名为fahrenheit的double变量中：

```
fahrenheit=celsiusToFahrenheit(celsius);
```

使用celsiusToFahrenheit方法时不必知道方法完成其工作的具体细节，只需知道方法完成了什么工作，而不必知道这些工作是*如何*完成的。

假设不知道方法是如何实现的，此时正在使用的信息隐藏形式叫作**过程抽象**。通过抽象不相关的细节（也就是隐藏它们）来简化推理。使用Java编程时，因为是在抽象某个方法如何工作的不相关细节，所以将这种技术称为“方法抽象”可能更有意义。然而，计算机科学家们使用术语“**过程**”（*procedure*）来表示各种指令序列，因此他们也习惯使用*过程抽象*这一术语。过程抽象可以是一种强大的工具，它允许你一次只考虑一个方法，而不是一次考虑所有的方法，从而简化推理。

要使用过程抽象，需要一些技术来编写文档，指明某个方法完成的工作，同时又不说明该方法是如何实现的。可以像处理celsiusToFahrenheit方法时那样，为方法编写简短的注释。然而这种简短的注释是不完整的。例如celsiusToFahrenheit方法的注释中并没有指明当参数c的值小于最低摄氏温度（-273.15℃，也称**绝对零度**）时会发生什么。为了具有更好的完整性和一致性，我们将采用固定的格式，以此确保能为编写的方法提供相同类型的信息。该格式由5部分构成，下面以celsiusToFahrenheit方法为例，给出了各构成部分的具体信息。