教育部高等教育司推荐
国外优秀信息科学与技术系列教学用书

数据结构与Java类集框架

（影印版）

DATA STRUCTURES AND THE JAVA COLLECTIONS FRAMEWORK

■ William J. Collins

高等教育出版社
Higher Education Press
McGraw-Hill Companies

教育部高等教育司推荐
国外优秀信息科学与技术系列教学用书

数据结构与 Java 类集框架

（影印版）

DATA STRUCTURES AND THE JAVA COLLECTIONS FRAMEWORK

William J. Collins

高等教育出版社

McGraw-Hill Companies

图字：01-2002-1177 号

Data Structures and the Java Collections Framework, First Edition

William J. Collins

图书在版编目(CIP)数据

数据结构与 Java 类集框架 /（美）柯林斯(Collins, W.J.）著. —影印本. —北京：高等教育出版社，2002.7

计算机系本科、研究生教材

ISBN 7-04-011257-4

Ⅰ.数... Ⅱ.柯... Ⅲ.①数据结构-高等学校-教材-英文②JAVA 语言-程序设计-高等学校-教材-英文 Ⅳ.TP311.12

中国版本图书馆 CIP 数据核字（2002）第 041607 号

数据结构与 Java 类集框架（影印版）

William J. Collins

出版发行	高等教育出版社	**购书热线**	010-64054588
社　　址	北京市东城区沙滩后街 55 号	**免费咨询**	800-810-0598
邮政编码	100009	**网　　址**	http://www.hep.edu.cn
传　　真	010-64014048		http://www.hep.com.cn
经　　销	新华书店北京发行所		
印　　刷	北京外文印刷厂		
开　　本	787×1092　1/16	**版　　次**	2002 年 7 月第 1 版
印　　张	46.25	**印　　次**	2002 年 7 月第 1 次印刷
字　　数	1000 000	**定　　价**	48.00 元

本书如有缺页、倒页、脱页等质量问题，请到所购图书销售部门联系调换。

前　言

20 世纪末，以计算机和通信技术为代表的信息科学和技术对世界经济、科技、军事、教育和文化等产生了深刻影响。信息科学技术的迅速普及和应用，带动了世界范围信息产业的蓬勃发展，为许多国家带来了丰厚的回报。

进入 21 世纪，尤其随着我国加入·WTO，信息产业的国际竞争将更加激烈。我国信息产业虽然在 20 世纪末取得了迅猛发展，但与发达国家相比，甚至与印度、爱尔兰等国家相比，还有很大差距。国家信息化的发展速度和信息产业的国际竞争能力，最终都将取决于信息科学技术人才的质量和数量。引进国外信息科学和技术优秀教材，在有条件的学校推动开展英语授课或双语教学，是教育部为加快培养大批高质量的信息技术人才采取的一项重要举措。

为此，教育部要求由高等教育出版社首先开展信息科学和技术教材的引进试点工作。同时提出了两点要求，一是要高水平，二是要低价格。在高等教育出版社和信息科学技术引进教材专家组的努力下，经过比较短的时间，第一批引进的 20 多种教材已经陆续出版。这套教材出版后受到了广泛的好评，其中有不少是世界信息科学技术领域著名专家、教授的经典之作和反映信息科学技术最新进展的优秀作品，代表了目前世界信息科学技术教育的一流水平，而且价格也是最优惠的，与国内同类自编教材相当。

这项教材引进工作是在教育部高等教育司和高教社的共同组织下，由国内信息科学技术领域的专家、教授广泛参与，在对大量国外教材进行多次遴选的基础上，参考了国内和国外著名大学相关专业的课程设置进行系统引进的。其中，John Wiley 公司出版的贝尔实验室信息科学研究中心副总裁 Silberschatz 教授的经典著作《操作系统概念》，是我们经过反复谈判，做了很多努力才得以引进的。William Stallings 先生曾编写了在美国深受欢迎的信息科学技术系列教材，其中有多种教材获得过美国教材和学术著作者协会颁发的计算机科学与工程教材奖，这批引进教材中就有他的两本著作。留美中国学者 Jiawei Han 先生的《数据挖掘》是该领域中具有里程碑意义的著作。由达特茅斯学院 Thomas Cormen 和麻省理工学院、哥伦比亚大学的几

位学者共同编著的经典著作《算法导论》，在经历了 11 年的锤炼之后于 2001 年出版了第二版。目前任教于美国 Massachusetts 大学的 James Kurose 教授，曾在美国三所高校先后 10 次获得杰出教师或杰出教学奖，由他主编的《计算机网络》出版后，以其体系新颖、内容先进而倍受欢迎。在努力降低引进教材售价方面，高等教育出版社做了大量和细致的工作。这套引进的教材体现了权威性、系统性、先进性和经济性等特点。

教育部也希望国内和国外的出版商积极参与此项工作，共同促进中国信息技术教育和信息产业的发展。我们在与外商的谈判工作中，不仅要坚定不移地引进国外最优秀的教材，而且还要千方百计地将版权转让费降下来，要让引进教材的价格与国内自编教材相当，让广大教师和学生负担得起。中国的教育市场巨大，外国出版公司和国内出版社要通过扩大发行数量取得效益。

在引进教材的同时，我们还应做好消化吸收，注意学习国外先进的教学思想和教学方法，提高自编教材的水平，使我们的教学和教材在内容体系上，在理论与实践的结合上，在培养学生的动手能力上能有较大的突破和创新。

目前，教育部正在全国 35 所高校推动示范性软件学院的建设和实施，这也是加快培养信息科学技术人才的重要举措之一。示范性软件学院要立足于培养具有国际竞争力的实用性软件人才，与国外知名高校或著名企业合作办学，以国内外著名 IT 企业为实践教学基地，聘请国内外知名教授和软件专家授课，还要率先使用引进教材开展教学。

我们希望通过这些举措，能在较短的时间，为我国培养一大批高质量的信息技术人才，提高我国软件人才的国际竞争力，促进我国信息产业的快速发展，加快推动国家信息化进程，进而带动整个国民经济的跨越式发展。

教育部高等教育司

二〇〇二年三月

To Karen, my wife of 35 years, for giving me 20 of the happiest years of my life.

W.J.C.

PREFACE

This book is intended for an object-oriented course in data structures and algorithms. The implementation language is Java, and it is assumed that students have taken an introductory course in which that language was used. That course should have covered the fundamental statements and data types, as well as arrays and the basics of file processing.

THE JAVA COLLECTIONS FRAMEWORK

One of the distinctive features of this text is its emphasis on the Java Collections Framework, part of the java.util package. Basically, the framework is a hierarchy with interfaces at each level except the lowest, and collection classes that implement those interfaces at the lowest level. The collection classes implement most of the data structures studied in a second computer science course, such as a resizable array class, a linked-list class, a balanced binary-search-tree class, and a hash-set class.

There are several advantages to using the Java Collections Framework. First, students will be working with code that has been extensively tested; they need not depend on modules created by the instructor or textbook author. Second, students will have the opportunity to study professionals' code, which is substantially more efficient—and succinct—than what they have seen before. Third, the framework is available for later courses in the curriculum, and beyond!

OTHER IMPLEMENTATIONS CONSIDERED

As important as the Java Collections Framework is, it cannot be the exclusive focus of such a fundamental course in data structures and algorithms. Approaches that differ from those in the Java Collections Framework also deserve consideration. For example, the HashSet and HashMap classes utilize chaining, so there is a separate section on open addressing, and a discussion of the trade-offs of one design over the other. Also, there is coverage of data structures (such as graphs) and algorithms (such as the heap sort) that are not yet included in the Java Collections Framework.

This text also satisfies another essential need of a data structures and algorithms course: the need for students to practice developing their own data structures. There are programming projects in which data structures are either created "from the ground up" or extended from examples in the chapters. And there are other projects to develop or extend applications that *use* the Java Collections Framework.

GRAPHICAL USER INTERFACE

Instead of console input-output, we employ a simple graphical user interface (GUI) with a single input line and any number of output lines. A brief outline of this GUI is given in Chapter 1, and a more extensive description is provided in Appendix 2. Two immediate consequences of using this GUI are that input loops are no longer applicable, and that the output is scrollable. But the main significance is that students gain a better understanding of event-driven programming: the real-world environment.

PEDAGOGICAL FEATURES

This text offers several features that may improve the teaching environment for instructors and the learning environment for students. Each chapter starts with a list of objectives, and concludes with at least one major programming assignment. Each data structure is carefully described, with a precondition and postcondition for each method. In addition, most of the methods include examples of how to call the method, and the results of that call.

The implementation details, especially of the Java Collections Framework classes, are carefully investigated in the text and reinforced in a suite of 24 labs. The organization of these labs is described later in this preface. Each chapter has a variety of exercises, and the answers to all of the exercises are available to the instructor.

SUPPORT MATERIAL

The website for all of the support material is www.mhhe.com/collins, and it has links to the following information for students:

- An overview of the labs and how to access them
- The source code for all projects developed in the text
- Applets for projects that have a strong visual component

Additionally, instructors can obtain the following from the website:

- Instructors' options with regard to the labs
- PowerPoint slides for each chapter (approximately 1500 slides)
- Answers to every exercise

SYNOPSES OF THE CHAPTERS

Chapter 1 presents those features of Java that serve as the foundation for subsequent chapters. Much of the material reflects an object orientation: inheritance, polymorphism, and exception handling. There are lab experiments to review classes, as well as on inheritance and exception handling. See the "Organization of the Labs" section in this preface for more information on labs.

Chapter 2 introduces abstract classes and interfaces, and there is a lab for both of these topics. Of special interest is the Collection interface, the root of many classes in the Java Collections Framework. As a simple implementation of the Collection interface, a primitive version of a singly linked list class is created. This LinkedCollection class serves mainly to provide a backdrop for several key features of the Java Collections Framework, such as iterators and embedded classes. A lab experiment on iterators helps to solidify a student's understanding of this vital concept.

Chapter 3, an introduction to software engineering, outlines the four stages of the software-development life cycle: analysis, design, implementation, and maintenance. The Unified Modeling Language is introduced as a design tool to depict inheritance, composition, and aggregation. Big-O notation, which pervades subsequent chapters, allows environment-independent estimates of the time requirements for methods. Both run-time validation and timing are discussed, and for each of those topics there is a follow-up lab.

Chapter 4, on recursion, represents a temporary shift in emphasis from data structures to algorithms. Backtracking is introduced, not only as a general technique for problem solving, but as another illustration of creating polymorphic references through interfaces. And the same BackTrack class is used for searching a maze; placing eight queens on a chess board, where none is under attack by another queen; and illustrating that a knight can traverse every square in a chess board without landing on any square more than once. Other applications of recursion, such as for the Towers of Hanoi game, further highlight the elegance of recursion, especially when compared to the corresponding iterative methods. This elegance is further illustrated in labs on Fibonacci numbers, the binary search, and generating permutations. Recursion is also encountered in later chapters, notably in the Java Collections Framework versions of the quick sort and merge sort. Moreover, recursion is an indispensable—even if seldom used—tool for every computing professional.

In Chapter 5, we begin our study of the Java Collections Framework with the ArrayList data structure and class. An ArrayList structure is a smart array: automatically resizable, and with methods to handle insertions and deletions at any index. The design starts with the method description—precondition, postcondition, and method heading—of the most widely used methods in the ArrayList class. There follows an outline of the implementation of the class, and further details are available in a lab. The application of the ArrayList class, high-precision arithmetic, is essential for public-key cryptography. This application is extended in a lab and in a programming project. There is another programming project to develop a Deque class.

Chapter 6 presents the LinkedList data structure and class, characterized by linear-time methods for inserting, removing, or retrieving at an arbitrary position. This property makes a compelling case for list iterators: objects that traverse a LinkedList object and have constant-time methods for inserting, removing, or retrieving at the "current" position. The Java Collections Framework's design is doubly linked and circular, but other approaches are also considered. The application is a small line-

editor, for which list iterators are well suited. This application is extended in a programming project.

Queues and stacks are the subjects of Chapter 7. The Queue class is not currently included in the Java Collections Framework, but is easily and efficiently implemented with a LinkedList field. A contiguous implementation is also presented. The specific application of calculating the average waiting time at a car wash falls into the general category of *computer simulation.* The Stack class implementation, in the package java.util, predates the Java Collections Framework. There are two applications of the Stack class: the implementation of recursion, and the conversion from infix to postfix notation. This latter application is expanded in a lab, and forms the basis for a project on evaluating a condition.

Chapter 8 focuses on binary trees in general, and binary search trees in particular. The essential features of binary trees are presented; these are important for understanding later material on AVL trees, red-black trees, heaps, and decision trees. The study of binary search trees sets the stage for the subject of chapter 9, balanced binary search trees. In fact, the binary search tree class is a monochromatic version of the Java Collections Framework's implementation of red-black trees.

In Chapter 9, we look at balanced binary search trees, specifically, AVL trees and red-black trees. Rotations are introduced as the mechanism by which rebalancing is accomplished. With the help of Fibonacci trees, we establish that the height of an AVL tree is always logarithmic in the number of elements in the tree. Red-black trees are similarly well behaved. The AVLTree class is implemented, except for the remove method (Project 9.1).

Red-black trees are implemented in the Java Collections Framework in the TreeMap and TreeSet classes, the foci of Chapter 10. In a Map object, each element has a unique key part and also a value part. A TreeMap object is stored in a red-black tree, ordered by the elements' keys. There are labs to guide students through the details of restructuring after an insertion or removal. The application consists of searching a thesaurus for synonyms. A TreeSet object is implemented as a TreeMap object in which each element has the same dummy value part. The application of the TreeSet class is a simple spell-checker. There are project assignments to determine the frequency of each word in a text file, and to build a concordance.

Chapter 11 introduces the PriorityQueue interface, which is not yet part of the Java Collections Framework. A heap-based implementation allows insertions in constant average time, and removal of the highest-priority element in logarithmic worst time. The application is in the area of data compression, specifically, Huffman encodings: given a text file, generate a minimal, prefix-free encoding. The project assignment is to convert the encoding back to the original text file. The lab experiment incorporates fairness into a priority queue, so that ties for highest-priority element are always resolved in favor of the element that was on the priority queue for the longest time.

Sorting is the topic of Chapter 12. Estimates of the maximum lower bounds for comparison-based sorts are developed. The Java Collections Framework provides

two sort methods: the quick sort for arrays of primitive types, and the merge sort for arrays of objects and for implementations of the List interface. Two other important sort algorithms, the tree sort and the heap sort, are also included. The chapter's lab experiment compares all of these sort algorithms on randomly generated integers. The project assignment is to sort a file of names and social security numbers.

Chapter 13 starts with a review of sequential and binary searching, and then investigates hashing. The Java Collections Framework has a HashMap class for elements that consist of key-value pairs. The HashSet class is backed by the HashMap class; that is, a HashSet object is viewed as a HashMap object in which all the elements have the same dummy value parts. Basically, the average time for insertion, removal, and searching is constant! This average speed is further explored in a lab that compares HashMap objects to TreeMap objects. There is also a comparison of chained hashing (the basis for the HashMap class) and open-address hashing. This comparison is further explored in a programming project.

The most general data structures—graphs, trees, and networks—are presented in Chapter 14. There are, initially, outlines of the essential algorithms: breadth-first traversal, depth-first traversal, connectedness, finding a minimal spanning tree, and finding a shortest path between two vertices. The only class developed is the (directed) Network class, with an adjacency-list implementation. Other classes, such as UndirectedGraph and UndirectedNetwork, can be straightforwardly defined as subclasses of the Network class. The traveling salesperson problem is investigated in a lab, and there is a programming project to complete an adjacency-matrix version of the Network class. Another backtracking application is presented, with the same BackTrack class that was introduced in Chapter 4.

With each chapter, there is an associated web page that includes all programs developed in the chapter, and applets, where appropriate, to animate the concepts presented.

APPENDICES

Appendix 1 contains the background that will allow students to comprehend the mathematical aspects of the chapters. Summation notation and the rudimentary properties of logarithms are essential, and the material on mathematical induction will lead to a deeper appreciation of the analysis of binary trees and open-address hashing.

Appendix 2 is a brief tutorial on the GUI and GUIListener classes. These classes support the event model for input and output in all the programs in the text and labs. Understanding the event model and the role of separate threads is quite a bit more difficult than understanding console input and output. But virtually every application program is event driven, so the time spent on these topics is an investment in the future.

Appendix 3 presents a user's view of the Java Collections Framework. For each method, a method description is provided: precondition, postcondition, and method heading. The relationships between the six classes and the four interfaces are as follows:

The ArrayList class **implements** the List interface, which **extends** the Collection interface.

The LinkedList class **implements** the List interface, which **extends** the Collection interface.

The TreeMap class **implements** the Map interface.

The TreeSet class **implements** the Set interface, which **extends** the Collection interface.

The HashMap class **implements** the Map interface.

The HashSet class **implements** the Set interface, which **extends** the Collection interface.

ORGANIZATION OF THE LABS

There are 24 website labs associated with this text. For both students and instructors, the Uniform Resource Locator (URL) is www.mhhe.com/collins. The labs do not contain essential material, but provide reinforcement of the text material. For example, after the ArrayList and LinkedList classes have been investigated, there is a lab to perform some timing experiments on those two classes.

The labs are self-contained, so the instructor has considerable flexibility in assigning the labs:

1. They can be assigned as closed labs.
2. They can be assigned as open labs.
3. They can be assigned as ungraded homework.

In addition to the obvious benefit of promoting active learning, these labs also encourage use of the scientific method. Basically, each lab is set up as an experiment. Students *observe* some phenomenon, such as the organization of the Java Collections Framework's LinkedList class. They then formulate and submit a *hypothesis*—with their own code—about the phenomenon. After *testing* and, perhaps, revising their hypothesis, they submit the *conclusions* they drew from the experiment.

There are more labs related to earlier chapters than to later ones. This allows students to start working right from the beginning of the course, even before programming projects can be assigned.

ACKNOWLEDGMENTS

Chun Wai Liew developed the graphical user interface employed throughout the book and converted the Java classes to utilize Swing components. Joshua Bloch of Sun Microsystems provided valuable insights into the Java Collections Framework. And I am grateful to Sun Microsystems for permission to include code (mostly Joshua's!) from the Java Collections Framework.

The following reviewers made many helpful suggestions:

Joseph M. Clifton, *University of Wisconsin, Platteville*
Homer C. Gerber, *University of Central Florida*
James K. Huggins, *Kettering University*
Dean Kelley, *Minnesota State University, Mankato*
Teresa Leyk, *Texas A&M University*
Mitchell L. Neilsen, *Kansas State University*
Thaddeus F. Pawlicki, *University of Rochester*
Gregory J. E. Rawlins, *Indiana University*
Munindar P. Singh, *North Carolina State University*
Sebastian Thrun, *Carnegie Mellon University*
Kenneth R. Vollmar, *Southwest Missouri State University*
Ming Wang, *Embry-Riddle Aeronautical University*
Zhi-Li Zhang, *University of Minnesota*

I am thankful for the encouragement and support from McGraw-Hill. Emily Gray and Betsy Jones were cordially persistent in steering me away from unworkable ideas. Jane Matthews kept the production on schedule.

Several students from Lafayette College made important contributions. Eric Panchenko created all of the applets and many of the driver programs; Yi Sun constructed the initial template for the labs; Xenia Taoubina wrote many of the method descriptions and examples in Appendix 3. Finally, I am indebted to all of the students at Lafayette College who participated in the class testing of the book and endured earlier versions of the labs.

Bill Collins

责任编辑　康兆华
封面设计　张　楠
责任印制　陈伟光

BRIEF CONTENTS

CONTENTS