

GLOBAL  
EDITION



# Data Structures and Abstractions with Java™

FOURTH EDITION

Frank M. Carrano • Timothy M. Henry

ALWAYS LEARNING

PEARSON

## ONLINE ACCESS

Thank you for purchasing a new copy of *Data Structures and Abstractions with Java<sup>TM</sup>, Fourth Edition, Global Edition*. Your textbook includes twelve months of prepaid access to the book's Premium Content. This prepaid subscription provides you with full access to the following student support areas:

- VideoNotes are step-by-step video tutorials specifically designed to enhance the programming concepts presented in this textbook
- Premium Web Chapters

Use a coin to scratch off the coating and reveal your student access code.  
Do not use a knife or other sharp object as it may damage the code.

To access the *Data Structures and Abstractions with Java<sup>TM</sup>, Fourth Edition, Global Edition*, Premium Content for the first time, you will need to register online using a computer with an Internet connection and a web browser. The process takes just a couple of minutes and only needs to be completed once.

1. Go to **[www.pearsonglobaleditions.com/Carrano](http://www.pearsonglobaleditions.com/Carrano)**
2. Click on **Companion Website**.
3. Click on the **Register** button.
4. On the registration page, enter your student access code\* found beneath the scratch-off panel. Do not type the dashes. You can use lower- or uppercase.
5. Follow the on-screen instructions. If you need help at any time during the online registration process, simply click the **Need Help?** icon.
6. Once your personal Login Name and Password are confirmed, you can begin using the *Data Structures and Abstractions with Java<sup>TM</sup>* Companion Website!

### To log in after you have registered:

You only need to register for this Companion Website once. After that, you can log in any time at **[www.pearsonglobaleditions.com/Carrano](http://www.pearsonglobaleditions.com/Carrano)** by providing your Login Name and Password when prompted.

\*Important: The access code can only be used once. This subscription is valid for twelve months upon activation and is not transferable.

# Data Structures and Abstractions with Java™

*Fourth Edition*

*Global Edition*

Frank M. Carrano

*University of Rhode Island*

Timothy M. Henry

*New England Institute of Technology*

Global Edition contributions by

Mohit P. Tahiliani

*National Institute of Technology Karnataka*

**PEARSON**

Boston Columbus Indianapolis New York San Francisco Amsterdam  
Cape Town Dubai London Madrid Milan Munich Paris Montréal Toronto  
Delhi Mexico City São Paulo Sydney Hong Kong Seoul Singapore Taipei Tokyo

*Editorial Director, ECS:* Marcia Horton  
*Executive Editor:* Tracy Johnson (Dunkelberger)  
*Editorial Assistant:* Kelsey Loanes  
*Director of Marketing:* Christy Lesko  
*Product Marketing Manager:* Bram van Kempen  
*Field Marketing Manager:* Demetrius Hall  
*Marketing Assistant:* Jon Bryant  
*Director of Program Management:* Erin Gregg  
*Program Management-Team Lead:* Scott Disanno  
*Program Manager:* Carole Snyder  
*Project Manager:* Robert Engelhardt  
*Procurement Specialist:* Maura Zaldivar-Garcia  
*Assistant Acquisitions Editor, Global Edition:*  
Aditee Agarwal

*Project Editor, Global Edition:*  
Amrita Naskar  
*Manager, Media Production, Global Edition:*  
Vikram Kumar  
*Senior Manufacturing Controller, Production,  
Global Edition:* Trudy Kimber  
*Senior Art Director:* Kathryn Foot  
*Cover Designer:* Lumina Datamatics Ltd.  
*Cover Art:* Kris Leov/Shutterstock  
*Permissions Supervisor:* Rachel Youdelman  
*Permissions Administrator:* William Opaluch  
*Web Development, Senior Manager:*  
Steve Wright  
*Associate Web Developer:* Barry Offringa

Credits and acknowledgments borrowed from other sources and reproduced, with permission, in this textbook appear on the appropriate page within text.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Pearson Education Limited  
Edinburgh Gate  
Harlow  
Essex CM20 2JE  
England

and Associated Companies throughout the world

Visit us on the World Wide Web at:  
[www.pearsonglobaleditions.com](http://www.pearsonglobaleditions.com)

© Pearson Education Limited 2016

The rights of Frank M. Carrano and Timothy M. Henry to be identified as the authors of this work have been asserted by them in accordance with the Copyright, Designs and Patents Act 1988.

*Authorized adaptation from the United States edition, entitled Data Structures and Abstractions with Java™, 4th edition, ISBN 978-0-13-374405-7, by Frank M. Carrano and Timothy M. Henry, published by Pearson Education © 2015.*

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without either the prior written permission of the publisher or a license permitting restricted copying in the United Kingdom issued by the Copyright Licensing Agency Ltd, Saffron House, 6–10 Kirby Street, London EC 1N 8TS.

All trademarks used herein are the property of their respective owners. The use of any trademark in this text does not vest in the author or publisher any trademark ownership rights in such trademarks, nor does the use of such trademarks imply any affiliation with or endorsement of this book by such owners.

Many of the designations by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the publisher was aware of a trademark claim, the designations have been printed in initial caps or all caps.

ISBN 10: 1-292-07718-2  
ISBN 13: 978-1-292-07718-5

British Library Cataloguing-in-Publication Data  
A catalogue record for this book is available from the British Library

10 9 8 7 6 5 4 3 2 1

Typeset by GEX Publishing Services

Printed by Ashford Colour Press Ltd, Gosport

Welcome to the fourth edition of *Data Structures and Abstractions with Java*, a book for an introductory course in data structures, typically known as CS-2.

I wrote this book with you in mind—whether you are an instructor or a student—based upon my experiences during more than three decades of teaching undergraduate computer science. I wanted my book to be reader friendly so that students could learn more easily and instructors could teach more effectively. To this end, you will find the material covered in small pieces—I call them “segments”—that are easy to digest and facilitate learning. Numerous examples that mimic real-world situations provide a context for the new material and help to make it easier for students to learn and retain abstract concepts. Many simple figures illustrate and clarify complicated ideas. Included are over 60 video tutorials to supplement the instruction and help students when their instructor is unavailable.

I am pleased and excited to welcome my co-author and colleague, Dr. Timothy Henry, to this edition. Together we have given a fresh update to this work, while retaining the topics and order of the previous edition. You will find a greater emphasis on our design decisions for both specifications and implementations of the various data structures, as well as a new introduction to safe and secure programming practices. The new features in this edition are given on the next page.

We hope that you enjoy reading this book. Like many others before you, you can learn—or teach—data structures in an effective and sustainable way.

Warm regards,

*Frank M. Carrano*

## Organization and Structure

This book's organization, sequencing, and pace of topic coverage make learning and teaching easier by focusing your attention on one concept at a time, by providing flexibility in the order in which you can cover topics, and by clearly distinguishing between the specification and implementation of abstract data types, or ADTs. To accomplish these goals, we have organized the material into 29 chapters, composed of small, numbered segments that deal with one concept at a time. Each chapter focuses on either the specification and use of an ADT or its various implementations. You can choose to cover the specification of an ADT followed by its implementations, or you can treat the specification and use of several ADTs before you consider any implementation issues. The book's organization makes it easy for you to choose the topic order that you prefer.

## Table of Contents at a Glance

The following brief table of contents shows the overall composition of the book. Notice the new Prelude and nine Java Interludes. Further details—including a chapter-by-chapter description—are given later in this preface. Note that some of the appendixes and the glossary are available online.

Introduction	Organizing Data	Java Interlude 6	Mutable and Immutable Objects
Prelude	Designing Classes	Chapter 16	Sorted Lists
Chapter 1	Bags	Java Interlude 7	Inheritance
Java Interlude 1	Generics	Chapter 17	Inheritance and Lists
Chapter 2	Bag Implementations That Use Arrays	Chapter 18	Searching
Java Interlude 2	Exceptions	Java Interlude 8	Generics Once Again
Chapter 3	A Bag Implementation That Links Data	Chapter 19	Dictionaries
Chapter 4	The Efficiency of Algorithms	Chapter 20	Dictionary Implementations
Chapter 5	Stacks	Chapter 21	Introducing Hashing
Chapter 6	Stack Implementations	Chapter 22	Hashing as a Dictionary Implementation
Chapter 7	Recursion	Chapter 23	Trees
Java Interlude 3	More About Generics	Chapter 24	Tree Implementations
Chapter 8	An Introduction to Sorting	Java Interlude 9	Cloning
Chapter 9	Faster Sorting Methods	Chapter 25	A Binary Search Tree Implementation
Java Interlude 4	More About Exceptions	Chapter 26	A Heap Implementation
Chapter 10	Queues, Deques, and Priority Queues	Chapter 27	Balanced Search Trees
Chapter 11	Queue, Deque, and Priority Queue Implementations	Chapter 28	Graphs
Chapter 12	Lists	Chapter 29	Graph Implementations
Chapter 13	A List Implementation That Uses an Array	Appendix A	Documentation and Programming Style
Chapter 14	A List Implementation That Links Data	Appendix B	Java Basics (online)
Java Interlude 5	Iterators	Appendix C	Java Classes (online)
Chapter 15	Iterators for the ADT List	Appendix D	Creating Classes from Other Classes
		Appendix E	File Input and Output (online)
		Glossary	(online)

## What's New?

While the chapters are in the same order and cover the same topics as in the previous edition, reader feedback convinced us to move some material from the appendixes or online into the main portion of the book. Other changes are motivated by reader suggestions and our own desire to improve the presentation. Here are the significant changes in this edition:

- A new Prelude follows the Introduction and precedes Chapter 1 to discuss how to design classes. This material was in Appendix D of the previous edition.
- Relevant aspects of Java have been extracted from either the appendixes or the chapters themselves and placed into new Java Interludes that occur throughout the book and as needed. By doing so, we increase the distinction and separation between concepts and Java-specific issues. The titles of these interludes follow, and you can see their placement between chapters on the previous page:

Java Interlude 1 Generics

Java Interlude 2 Exceptions

Java Interlude 3 More About Generics

Java Interlude 4 More About Exceptions

Java Interlude 5 Iterators

Java Interlude 6 Mutable and Immutable Objects

Java Interlude 7 Inheritance

Java Interlude 8 Generics Once Again

Java Interlude 9 Cloning

- Safe and secure programming is a new topic that is introduced in Chapter 2, discussed in new Security Notes, and reflected in the Java code that implements the ADTs.
- Beginning with stacks in Chapter 5, most ADT methods now indicate failure by throwing an exception. Methods only return `null` when it cannot be a data value within a collection.
- Expanded coverage of generics treats generic methods and bounded types.
- Immutable, mutable, and cloneable objects are covered in Java Interludes instead of the online Chapter 30 of the previous edition.
- Additional Design Decisions continue to present the options one has when specifying and implementing particular ADTs and provide the rationale behind our choices.
- Illustrations have been revised to show objects specifically instead of as values within nodes or array elements.
- Vector-based implementations of the ADT list and queue are no longer covered, but are left as programming projects.
- Line numbers appear in program listings.
- Java code is Java 8 compliant.
- Supplements now include a test bank.

Here are the significant changes to specific chapters:

- Chapter 1 introduces the ADT set in addition to the bag.
- Chapter 2 introduces safe and secure programming. The code changes suggested here are integrated into all ADT implementations in subsequent chapters.
- Chapters 5 and 6 use exceptions in the specification and implementations of the ADT stack.
- Chapters 8 and 9 replace some Java code for sorting methods with pseudocode.

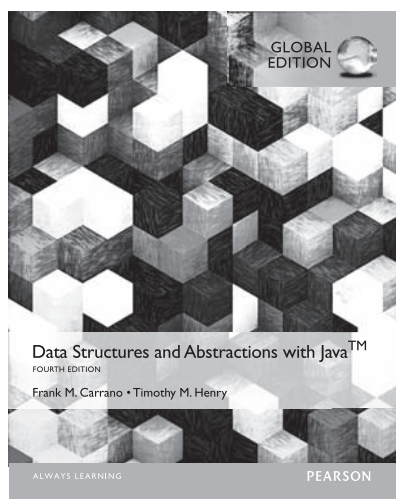
- Chapters 10 and 11 use exceptions in the specification and implementations of the ADTs queue, deque, and priority queue.
- Chapter 11 no longer covers the vector-based implementation of the ADT queue; it is left as a programming project.
- Chapters 12, 13, and 14 use exceptions in the specification and implementations of the ADT list.
- Chapter 13 changes the array-based implementation of the ADT list by ignoring the array element at index 0. The vector-based implementation of the ADT list is no longer covered, but is left as a programming project.
- Chapter 15 covers only iterators for the ADT list. The concepts of an iterator in Java are treated in the preceding Java Interlude 5 instead of in this chapter.
- Chapter 20 no longer covers the vector-based implementation of the ADT dictionary; it is left as a programming project.
- Chapter 23 defines balanced binary trees, which previously was in Chapter 25.
- Chapter 24 no longer defines an interface for a binary node, and the class `BinaryNode` no longer implements one.



The topics that we cover in this book deal with the various ways of organizing data so that a given application can access and manipulate data in an efficient way. These topics are fundamental to your future study of computer science, as they provide you with the foundation of knowledge required to create complex and reliable software. Whether you are interested in designing video games or software for robotic controlled surgery, the study of data structures is vital to your success. Even if you do not study all of the topics in this book now, you are likely to encounter them later. We hope that you will enjoy reading the book, and that it will serve as a useful reference tool for your future courses.

After looking over this preface, you should read the Introduction. There you will quickly see what this book is about and what you need to know about Java before you begin. The Prelude discusses class design and the use of Java interfaces. We use interfaces throughout the book. Appendixes A through E review javadoc comments, Java basics, classes, inheritance, and files. New Java Interludes occur throughout the book and cover advanced aspects of Java as they are needed. Note that at the end of the book you will find Java's reserved words, its primitive data types, the precedence of its operators, and a list of Unicode characters.

Please be sure to browse the rest of this preface to see the features that will help you in your studies.



## Features to Enhance Learning

**E**ach chapter begins with a table of contents, a list of prerequisite portions of the book that you should have read, and the learning objectives for the material to be covered. Other pedagogical elements appear throughout the book, as follows:



**Notes** Important ideas are presented or summarized in highlighted paragraphs and are meant to be read in line with the surrounding text.



**Security Notes** Aspects of safe and secure programming are introduced and highlighted in this new feature.



**A Problem Solved** Large examples are presented in the form of “A Problem Solved,” in which a problem is posed and its solution is discussed, designed, and implemented.



**Design Decisions** To give readers insight into the design choices that one could make when formulating a solution, “Design Decision” elements lay out such options, along with the rationale behind the choice made for a particular example. These discussions are often in the context of one of the “A Problem Solved” examples.



**Examples** Numerous examples illuminate new concepts.



**Programming Tips** Suggestions to improve or facilitate programming are presented as soon as they become relevant.



**Self-Test Questions** Questions are posed throughout each chapter, integrated within the text, that reinforce the concept just presented. These “self-test” questions help readers to understand the material, since answering them requires pause and reflection. Solutions to these questions are provided at the end of each chapter.



VideoNote

**VideoNotes** Online tutorials are a Pearson feature that provides visual and audio support to the presentation given throughout the book. They offer students another way to recap and reinforce key concepts. VideoNotes allow for self-paced instruction with easy navigation, including the ability to select, play, rewind, fast-forward, and stop within each video. Unique VideoNote icons appear throughout this book whenever a video is available for a particular concept or problem. A detailed list of the VideoNotes for this text and their associated locations in the book can be found on page 26. VideoNotes are free with the purchase of a new textbook. To purchase access to VideoNotes, please go to

[www.pearsonglobaleditions.com/Carrano](http://www.pearsonglobaleditions.com/Carrano)

**Exercises and Programming Projects** Further practice is available by solving the exercises and programming projects at the end of each chapter. Unfortunately, we cannot give readers the answers to these exercises and programming projects, even if they are not enrolled in a class. Only instructors who adopt the book can receive selected answers from the publisher. For help with these exercises and projects, you will have to contact your instructor.

## Accessing Instructor and Student Resource Materials

The following items are available on the publisher's website at [www.pearsonglobaleditions.com/Carrano](http://www.pearsonglobaleditions.com/Carrano):

- Java code as it appears in the book
- A link to any misprints that have been discovered since the book was published
- Links to additional online content, which is described next

---

### Instructor Resources

The following protected material is available to instructors who adopt this book by logging onto Pearson's Instructor Resource Center, accessible from [www.pearsonglobaleditions.com/Carrano](http://www.pearsonglobaleditions.com/Carrano):

- PowerPoint lecture slides
- Instructor solutions manual
- Figures from the book
- Test bank

Additionally, instructors can access the book's Companion Website for the following online premium content, also accessible from [www.pearsonglobaleditions.com/Carrano](http://www.pearsonglobaleditions.com/Carrano):

- Instructional VideoNotes
- Appendixes B, C, and E
- A glossary of terms

---

### Student Resources

The following material is available to students by logging onto the Companion Website accessible from [www.pearsonglobaleditions.com/Carrano](http://www.pearsonglobaleditions.com/Carrano):

- Instructional VideoNotes
- Appendixes B, C, and E
- A glossary of terms

Students must use the access card located in the front of the book to register for and then enter the Companion Website.

Note that the Java Class Library is available at [docs.oracle.com/javase/8/docs/api/](http://docs.oracle.com/javase/8/docs/api/).

## Content Overview

Readers of this book should have completed a programming course, preferably in Java. The appendixes cover the essentials of Java that we assume readers will know. You can use these appendixes as a review or as the basis for making the transition to Java from another programming language. The book itself begins with the Introduction, which sets the stage for the data organizations that we will study.

- **Prelude:** At the request of readers of the previous edition, we have moved the introduction to class design from the appendix to the beginning of the book. Most of the material that was in Appendix D of the third edition is now in the Prelude, which follows the Introduction.
- **Chapters 1 through 3:** We introduce the bag as an abstract data type (ADT). By dividing the material across several chapters, we clearly separate the specification, use, and implementation of the bag. For example, Chapter 1 specifies the bag and provides several examples of its use. This chapter also introduces the ADT set. Chapter 2 covers implementations that use arrays, while Chapter 3 introduces chains of linked nodes and uses one in the definition of a class of bags.

In a similar fashion, we separate specification from implementation throughout the book when we discuss various other ADTs. You can choose to cover the chapters that specify and use the ADTs and then later cover the chapters that implement them. Or you can cover the chapters as they appear, implementing each ADT right after studying its specification and use. A list of chapter prerequisites appears later in this preface to help you plan your path through the book.

Chapter 2 does more than simply implement the ADT bag. It shows how to approach the implementation of a class by initially focusing on core methods. When defining a class, it is often useful to implement and test these core methods first and to leave definitions of the other methods for later. Chapter 2 also introduces the concept of safe and secure programming, and shows how to add this protection to your code.

- **Java Interludes 1 and 2:** The first Java interlude introduces generics, so that we can use it with our first ADT, the bag. This interlude immediately follows Chapter 1. Java Interlude 2 introduces exceptions and follows Chapter 2. We apply this material, which was formerly in an appendix, to the implementations of the ADT bag.
- **Chapter 4:** Here we introduce the complexity of algorithms, a topic that we integrate into future chapters.
- **Chapters 5 and 6:** Chapter 5 discusses stacks, giving examples of their use, and Chapter 6 implements the stack using an array, a vector, and a chain.
- **Chapter 7:** Next, we present recursion as a problem-solving tool and its relationship to stacks. Recursion, along with algorithm efficiency, is a topic that is revisited throughout the book.
- **Java Interlude 3:** This interlude provides the Java concepts needed for the sorting methods that we are about to present. It introduces the standard interface `Comparable`, generic methods, bounded type parameters, and wildcards.
- **Chapters 8 and 9:** The next two chapters discuss various sorting techniques and their relative complexities. We consider both iterative and recursive versions of these algorithms.
- **Java Interlude 4:** This Java interlude shows how the programmer can write new exception classes. In doing so, it shows how to extend an existing class of exceptions. It also introduces the `finally` block.
- **Chapters 10 and 11:** Chapter 10 discusses queues, dequeues, and priority queues, and Chapter 11 considers their implementations. It is in this latter chapter that we introduce circularly linked and doubly linked chains. Chapter 11 also uses the programmer-defined class `EmptyQueueException`.
- **Chapters 12, 13, and 14:** The next three chapters introduce the ADT list. We discuss this collection abstractly and then implement it by using an array and a chain of linked nodes.
- **Java Interlude 5 and Chapter 15:** The coverage of Java iterators that was formerly in Chapter 15 now appears before the chapter in Java Interlude 5. Included are the standard interfaces `Iterator`,

Iterable, and `ListIterator`. Chapter 15 then shows ways to implement an iterator for the ADT list. It considers and implements Java's iterator interfaces `Iterator` and `ListIterator`.

- **Java Interlude 6:** This interlude discusses mutable and immutable objects, material that previously was in the online Chapter 30.
- **Chapters 16 and 17 and Java Interlude 7:** Continuing the discussion of a list, Chapter 16 introduces the sorted list, looking at two possible implementations and their efficiencies. Chapter 17 shows how to use the list as a superclass for the sorted list and discusses the general design of a superclass. Although inheritance is reviewed in Appendix D, the relevant particulars of inheritance—including protected access, abstract classes, and abstract methods—are presented in Java Interlude 7 just before Chapter 17.
- **Chapter 18:** We then examine some strategies for searching an array or a chain in the context of a list or a sorted list. This discussion is a good basis for the sequence of chapters that follows.
- **Java Interlude 8:** Before we get to the next chapter, we quickly cover in this interlude situations where more than one generic data type is necessary.
- **Chapters 19 through 22:** Chapter 19 covers the specification and use of the ADT dictionary. Chapter 20 presents implementations of the dictionary that are linked or that use arrays. Chapter 21 introduces hashing, and Chapter 22 uses hashing as a dictionary implementation.
- **Chapters 23 and 24 and Java Interlude 9:** Chapter 23 discusses trees and their possible uses. Included among the several examples of trees is an introduction to the binary search tree and the heap. Chapter 24 considers implementations of the binary tree and the general tree. Java Interlude 9 discusses cloning, a topic that was previously online. We clone an array, a chain of linked nodes, and a binary node. We also investigate a sorted list of clones. Although this material is important, you can treat it as optional, as it is not required in the following chapters.
- **Chapters 25 through 27:** Chapter 25 focuses on the implementation of the binary search tree. Chapter 26 shows how to use an array to implement the heap. Chapter 27 introduces balanced search trees. Included in this chapter are the AVL, 2-3, 2-4, and red-black trees, as well as B-trees.
- **Chapters 28 and 29:** Finally, we discuss graphs and look at several applications and two implementations.
- **Appendixes A through E:** The appendixes provide supplemental coverage of Java. As we mentioned earlier, Appendix A considers programming style and comments. It introduces javadoc comments and defines the tags that we use in this book. Appendix B reviews Java up to but not including classes. However, this appendix also covers the `Scanner` class, enumerations, boxing and unboxing, and the `for-each` loop. Appendix C discusses Java classes, Appendix D expands this topic by looking at composition and inheritance, and Appendix E discusses files.

## Acknowledgments

Our sincere appreciation and thanks go to the following reviewers for carefully reading the previous edition and making candid comments and suggestions that greatly improved the work:

Tony Allevato—*Virginia Polytechnic Institute and State University*  
 Mary Boelk—*Marquette University*  
 Suzanne Buchele—*Southwestern University*  
 Kevin Buffardi—*Virginia Polytechnic Institute and State University*  
 Jose Cordova—*University of Louisiana at Monroe*  
 Greg Gagne—*Westminster College*  
 Victoria Hilford—*University of Houston*  
 Jim Huggins—*Kettering University*  
 Shamim Kahn—*Columbus State University*  
 Kathy Liszka—*University of Akron*  
 Eli Tilevich—*Virginia Polytechnic Institute and State University*  
 Jianhua Yang—*Columbus State University*  
 Michelle Zhu—*Southern Illinois University*

Special thanks go to our support team at Pearson Education Computer Science during the lengthy process of revising this book: Executive Editor Tracy Dunkelberger, Program Manager Carole Snyder, Program Management-Team Leader Scott Disanno, and Project Manager Bob Engelhardt have always been a great help to us in completing our projects. Our long-time copy editor, Rebecca Pepper, ensured that the presentation is clear, correct, and grammatical. Thank you so much!

Our gratitude for the previously mentioned people does not diminish our appreciation for the help provided by many others. Steve Armstrong produced the lecture slides for this edition and previous editions of the book. Professor Charles Hoot of the Oklahoma City University created the lab manual, Professor Kathy Liszka from the University of Akron created the new collection of test questions, and Jesse Grabowski provided the solutions to many of the programming projects. Thank you again to the reviewers of the previous editions of the book:

### Reviewers for the third edition:

Steven Andrianoff—*St. Bonaventure University*  
 Brent Baas—*LeTourneau University*  
 Timothy Henry—*New England Institute of Technology*  
 Ken Martin—*University of North Florida*  
 Bill Siever—*Northwest Missouri State University*  
 Lydia Sinapova—*Simpson College*  
 Lubomir Stanchev—*Indiana University*  
 Judy Walters—*North Central College*  
 Xiaohui Yuan—*University of North Texas*

### Reviewers for the second edition:

Harold Anderson—*Marist College*  
 Razvan Andonie—*Central Washington University*  
 Tom Blough—*Rensselaer Polytechnic Institute*  
 Chris Brooks—*University of San Francisco*  
 Adrienne Decker—*University at Buffalo, SUNY*

Henry Etlinger—*Rochester Institute of Technology*  
 Derek Harter—*Texas A&M University*  
 Timothy Henry—*New England Institute of Technology*  
 Robert Holloway—*University of Wisconsin, Madison*  
 Charles Hoot—*Oklahoma City University*  
 Teresa Leyk—*Texas A&M University*  
 Robert McGlenn—*Southern Illinois University, Carbondale*  
 Edward Medvid—*Marymount University*  
 Charles Metzler—*City College of San Francisco*  
 Daniel Zeng—*University of Arizona*

**Reviewers for the first edition:**

David Boyd—*Valdosta State University*  
 Dennis Brylow—*Purdue University*  
 Michael Croswell—*Industry trainer/consultant*  
 Matthew Dickerson—*Middlebury College*  
 Robert Holloway—*University of Wisconsin, Madison*  
 John Motil—*California State University, Northridge*  
 Bina Ramamurthy—*University at Buffalo, SUNY*  
 David Surma—*Valparaiso University*

We continue to appreciate the many others who helped during previous editions. They include Alan Apt, James Blanding, Lianne Dunn, Mike Giacobbe, Toni Holm, Charles Hoot, Brian Jepson, Rose Kernan, Christianna Lee, Patrick Lindner, John Lovell, Vince O'Brien, Patty Roy, Walt Savitch, Ben Schomp, Heather Scott, Carole Snyder, Chirag Thakkar, Camille Trentacoste, Nate Walker, and Xiaohong Zhu.

Finally, we thank our families and friends—Doug, Joanne, Tita, Bobby, Ted, Nancy, Sue, Tom, Maybeth, Marge, and Lorraine—for giving us lives away from computers.

Thank you, everyone, for your expertise and good cheer.

**Frank M. Carrano**  
**Timothy M. Henry**

The publishers would like to thank the following for reviewing the content of the Global Edition:

S. H. Chung, *Wawasan Open University*  
 Shaligram Prajapat, *Devi Ahilya University*  
 Holger Eichelberger, *University of Hildesheim*





# Contents

	<b>Introduction: Organizing Data</b>	<b>31</b>
	<b>Prelude: Designing Classes</b>	<b>35</b>
	Encapsulation	36
	Specifying Methods	38
	Comments	38
	Preconditions and Postconditions	39
	Assertions	40
	Java Interfaces	41
	Writing an Interface	42
	Implementing an Interface	43
	An Interface as a Data Type	45
	Extending an Interface	46
	Named Constants Within an Interface	47
	Choosing Classes	49
	Identifying Classes	50
	CRC Cards	51
	The Unified Modeling Language	51
	Reusing Classes	54
<b>Chapter 1</b>	<b>Bags</b>	<b>61</b>
	The Bag	62
	A Bag's Behaviors	62
	Specifying a Bag	63
	An Interface	69
	Using the ADT Bag	71
	Using an ADT Is Like Using a Vending Machine	75
	The ADT Set	77
	Java Class Library: The Interface set	77
<b>Java Interlude 1</b>	<b>Generics</b>	<b>83</b>
	Generic Data Types	83
	Generic Types Within an Interface	84
	Generic Classes	85
<b>Chapter 2</b>	<b>Bag Implementations That Use Arrays</b>	<b>89</b>
	Using a Fixed-Size Array to Implement the ADT Bag	90
	An Analogy	90
	A Group of Core Methods	91
	Implementing the Core Methods	92
	Making the Implementation Secure	99
	Testing the Core Methods	101
	Implementing More Methods	103
	Methods That Remove Entries	106
	Using Array Resizing to Implement the ADT Bag	114
	Resizing an Array	114
	A New Implementation of a Bag	117
	The Pros and Cons of Using an Array to Implement the ADT Bag	120

<b>Java Interlude 2</b>	<b>Exceptions</b>	<b>125</b>
	The Basics	126
	Handling an Exception	128
	Postpone Handling: The throws Clause	128
	Handle It Now: The try-catch Blocks	129
	Multiple catch Blocks	130
	Throwing an Exception	131
<b>Chapter 3</b>	<b>A Bag Implementation That Links Data</b>	<b>133</b>
	Linked Data	134
	Forming a Chain by Adding to Its Beginning	135
	A Linked Implementation of the ADT Bag	137
	The Private Class Node	137
	An Outline of the Class <code>LinkedList</code>	138
	Defining Some Core Methods	139
	Testing the Core Methods	143
	The Method <code>getFrequencyOf</code>	144
	The Method <code>contains</code>	145
	Removing an Item from a Linked Chain	146
	The Methods <code>remove</code> and <code>clear</code>	147
	A Class Node That Has Set and Get Methods	151
	The Pros and Cons of Using a Chain to Implement the ADT Bag	154
<b>Chapter 4</b>	<b>The Efficiency of Algorithms</b>	<b>159</b>
	Motivation	160
	Measuring an Algorithm's Efficiency	161
	Counting Basic Operations	163
	Best, Worst, and Average Cases	165
	Big Oh Notation	166
	The Complexities of Program Constructs	168
	Picturing Efficiency	170
	The Efficiency of Implementations of the ADT Bag	173
	An Array-Based Implementation	173
	A Linked Implementation	175
	Comparing the Implementations	176
<b>Chapter 5</b>	<b>Stacks</b>	<b>183</b>
	Specifications of the ADT Stack	184
	Using a Stack to Process Algebraic Expressions	188
	A Problem Solved: Checking for Balanced Delimiters in an Infix Algebraic Expression	189
	A Problem Solved: Transforming an Infix Expression to a Postfix Expression	194
	A Problem Solved: Evaluating Postfix Expressions	199
	A Problem Solved: Evaluating Infix Expressions	201
	The Program Stack	203
	Java Class Library: The Class Stack	204
<b>Chapter 6</b>	<b>Stack Implementations</b>	<b>211</b>
	A Linked Implementation	211
	An Array-Based Implementation	215

	A Vector-Based Implementation	219
	Java Class Library: The Class Vector	220
	Using a Vector to Implement the ADT Stack	220
<b>Chapter 7</b>	<b>Recursion</b>	<b>227</b>
	What Is Recursion?	228
	Tracing a Recursive Method	232
	Recursive Methods That Return a Value	235
	Recursively Processing an Array	237
	Recursively Processing a Linked Chain	240
	The Time Efficiency of Recursive Methods	241
	The Time Efficiency of <code>countDown</code>	242
	The Time Efficiency of Computing $x^n$	243
	A Simple Solution to a Difficult Problem	244
	A Poor Solution to a Simple Problem	249
	Tail Recursion	251
	Indirect Recursion	253
	Using a Stack Instead of Recursion	254
<b>Java Interlude 3</b>	<b>More About Generics</b>	<b>265</b>
	The Interface <code>Comparable</code>	265
	Generic Methods	267
	Bounded Type Parameters	268
	Wildcards	270
	Bounded Wildcards	271
<b>Chapter 8</b>	<b>An Introduction to Sorting</b>	<b>275</b>
	Organizing Java Methods That Sort an Array	276
	Selection Sort	277
	Iterative Selection Sort	278
	Recursive Selection Sort	280
	The Efficiency of Selection Sort	281
	Insertion Sort	281
	Iterative Insertion Sort	283
	Recursive Insertion Sort	285
	The Efficiency of Insertion Sort	287
	Insertion Sort of a Chain of Linked Nodes	287
	Shell Sort	290
	The Algorithm	292
	The Efficiency of Shell Sort	293
	Comparing the Algorithms	293
<b>Chapter 9</b>	<b>Faster Sorting Methods</b>	<b>301</b>
	Merge Sort	302
	Merging Arrays	302
	Recursive Merge Sort	303
	The Efficiency of Merge Sort	305
	Iterative Merge Sort	307
	Merge Sort in the Java Class Library	307
	Quick Sort	308
	The Efficiency of Quick Sort	308
	Creating the Partition	309

	Implementing Quick Sort	312
	Quick Sort in the Java Class Library	314
	Radix Sort	314
	Pseudocode for Radix Sort	315
	The Efficiency of Radix Sort	316
	Comparing the Algorithms	316
<b>Java Interlude 4</b>	<b>More About Exceptions</b>	<b>323</b>
	Programmer-Defined Exception Classes	323
	Inheritance and Exceptions	327
	The finally Block	328
<b>Chapter 10</b>	<b>Queues, Deques, and Priority Queues</b>	<b>331</b>
	The ADT Queue	332
	A Problem Solved: Simulating a Waiting Line	336
	A Problem Solved: Computing the Capital Gain in a Sale of Stock	342
	Java Class Library: The Interface Queue	345
	The ADT Deque	346
	A Problem Solved: Computing the Capital Gain in a Sale of Stock	349
	Java Class Library: The Interface Deque	350
	Java Class Library: The Class ArrayDeque	351
	The ADT Priority Queue	351
	A Problem Solved: Tracking Your Assignments	353
	Java Class Library: The Class PriorityQueue	355
<b>Chapter 11</b>	<b>Queue, Deque, and Priority Queue Implementations</b>	<b>361</b>
	A Linked Implementation of a Queue	362
	An Array-Based Implementation of a Queue	366
	A Circular Array	366
	A Circular Array with One Unused Location	369
	Circular Linked Implementations of a Queue	374
	A Two-Part Circular Linked Chain	375
	Java Class Library: The Class AbstractQueue	380
	A Doubly Linked Implementation of a Deque	381
	Possible Implementations of a Priority Queue	385
<b>Chapter 12</b>	<b>Lists</b>	<b>391</b>
	Specifications for the ADT List	392
	Using the ADT List	399
	Java Class Library: The Interface List	403
	Java Class Library: The Class ArrayList	403
<b>Chapter 13</b>	<b>A List Implementation That Uses an Array</b>	<b>409</b>
	Using an Array to Implement the ADT List	410
	An Analogy	410
	The Java Implementation	412
	The Efficiency of Using an Array to Implement the ADT List	420
<b>Chapter 14</b>	<b>A List Implementation That Links Data</b>	<b>427</b>
	Operations on a Chain of Linked Nodes	428
	Adding a Node at Various Positions	428
	Removing a Node from Various Positions	432
	The Private Method getNodeAt	433

	Beginning the Implementation	434
	The Data Fields and Constructor	435
	Adding to the End of the List	437
	Adding at a Given Position Within the List	438
	The Methods isEmpty and toArray	439
	Testing the Core Methods	441
	Continuing the Implementation	442
	A Refined Implementation	445
	The Tail Reference	445
	The Efficiency of Using a Chain to Implement the ADT List	448
	Java Class Library: The Class LinkedList	450
<b>Java Interlude 5</b>	<b>Iterators</b>	<b>457</b>
	What Is an Iterator?	457
	The Interface Iterator	459
	The Interface Iterable	461
	Using the Interface Iterator	461
	Iterable and for-each Loops	465
	The Interface ListIterator	466
	The Interface List Revisited	469
	Using the Interface ListIterator	470
<b>Chapter 15</b>	<b>Iterators for the ADT List</b>	<b>473</b>
	Ways to Implement an Iterator	474
	A Separate Class Iterator	474
	An Inner Class Iterator	477
	A Linked Implementation	478
	An Array-Based Implementation	481
	Why Are Iterator Methods in Their Own Class?	484
	An Array-Based Implementation of the Interface ListIterator	486
	The Inner Class	487
<b>Java Interlude 6</b>	<b>Mutable and Immutable Objects</b>	<b>499</b>
	Mutable Objects	500
	Immutable Objects	502
	Creating a Read-Only Class	502
	Companion Classes	504
<b>Chapter 16</b>	<b>Sorted Lists</b>	<b>507</b>
	Specifications for the ADT Sorted List	508
	Using the ADT Sorted List	511
	A Linked Implementation	512
	The Method add	513
	The Efficiency of the Linked Implementation	520
	An Implementation That Uses the ADT List	520
	Efficiency Issues	523
<b>Java Interlude 7</b>	<b>Inheritance and Polymorphism</b>	<b>529</b>
	Further Aspects of Inheritance	529
	When to Use Inheritance	529
	Protected Access	530
	Abstract Classes and Methods	531
	Interfaces Versus Abstract Classes	533
	Polymorphism	534

<b>Chapter 17</b>	<b>Inheritance and Lists</b>	<b>541</b>
	Using Inheritance to Implement a Sorted List	542
	Designing a Base Class	544
	Creating an Abstract Base Class	549
	An Efficient Implementation of a Sorted List	551
	The Method add	551
<b>Chapter 18</b>	<b>Searching</b>	<b>557</b>
	The Problem	558
	Searching an Unsorted Array	558
	An Iterative Sequential Search of an Unsorted Array	559
	A Recursive Sequential Search of an Unsorted Array	560
	The Efficiency of a Sequential Search of an Array	562
	Searching a Sorted Array	562
	A Sequential Search of a Sorted Array	562
	A Binary Search of a Sorted Array	563
	Java Class Library: The Method <code>binarySearch</code>	568
	The Efficiency of a Binary Search of an Array	568
	Searching an Unsorted Chain	569
	An Iterative Sequential Search of an Unsorted Chain	570
	A Recursive Sequential Search of an Unsorted Chain	570
	The Efficiency of a Sequential Search of a Chain	571
	Searching a Sorted Chain	571
	A Sequential Search of a Sorted Chain	571
	A Binary Search of a Sorted Chain	572
	Choosing a Search Method	572
<b>Java Interlude 8</b>	<b>Generics Once Again</b>	<b>579</b>
	More Than One Generic Type	579
<b>Chapter 19</b>	<b>Dictionaries</b>	<b>581</b>
	Specifications for the ADT Dictionary	582
	A Java Interface	586
	Iterators	587
	Using the ADT Dictionary	588
	A Problem Solved: A Directory of Telephone Numbers	589
	A Problem Solved: The Frequency of Words	594
	A Problem Solved: A Concordance of Words	597
	Java Class Library: The Interface Map	600
<b>Chapter 20</b>	<b>Dictionary Implementations</b>	<b>605</b>
	Array-Based Implementations	606
	An Unsorted Array-Based Dictionary	606
	A Sorted Array-Based Dictionary	611
	Linked Implementations	616
	An Unsorted Linked Dictionary	617
	A Sorted Linked Dictionary	618
<b>Chapter 21</b>	<b>Introducing Hashing</b>	<b>625</b>
	What Is Hashing?	626
	Hash Functions	629
	Computing Hash Codes	629
	Compressing a Hash Code into an Index for the Hash Table	632

	Resolving Collisions	633
	Open Addressing with Linear Probing	633
	Open Addressing with Quadratic Probing	638
	Open Addressing with Double Hashing	639
	A Potential Problem with Open Addressing	641
	Separate Chaining	642
<b>Chapter 22</b>	<b>Hashing as a Dictionary Implementation</b>	<b>649</b>
	The Efficiency of Hashing	650
	The Load Factor	650
	The Cost of Open Addressing	651
	The Cost of Separate Chaining	653
	Rehashing	654
	Comparing Schemes for Collision Resolution	655
	A Dictionary Implementation That Uses Hashing	656
	Entries in the Hash Table	656
	Data Fields and Constructors	657
	The Methods <code>getValue</code> , <code>remove</code> , and <code>add</code>	659
	Iterators	664
	Java Class Library: The Class <code>HashMap</code>	665
	Java Class Library: The Class <code>HashSet</code>	666
<b>Chapter 23</b>	<b>Trees</b>	<b>669</b>
	Tree Concepts	670
	Hierarchical Organizations	670
	Tree Terminology	672
	Traversals of a Tree	676
	Traversals of a Binary Tree	677
	Traversals of a General Tree	679
	Java Interfaces for Trees	680
	Interfaces for All Trees	680
	An Interface for Binary Trees	681
	Examples of Binary Trees	682
	Expression Trees	683
	Decision Trees	684
	Binary Search Trees	688
	Heaps	690
	Examples of General Trees	693
	Parse Trees	693
	Game Trees	693
<b>Chapter 24</b>	<b>Tree Implementations</b>	<b>703</b>
	The Nodes in a Binary Tree	704
	A Class of Binary Nodes	705
	An Implementation of the ADT Binary Tree	706
	Creating a Basic Binary Tree	707
	The Method <code>privateSetTree</code>	708
	Accessor and Mutator Methods	711
	Computing the Height and Counting Nodes	711
	Traversals	712
	An Implementation of an Expression Tree	717

	General Trees	718
	A Node for a General Tree	718
	Using a Binary Tree to Represent a General Tree	719
<b>Java Interlude 9</b>	<b>Cloning</b>	<b>727</b>
	Cloneable Objects	727
	Cloning an Array	733
	Cloning a Chain	736
	A Sorted List of Clones	739
	Cloning a Binary Node	741
<b>Chapter 25</b>	<b>A Binary Search Tree Implementation</b>	<b>743</b>
	Getting Started	744
	An Interface for the Binary Search Tree	745
	Duplicate Entries	747
	Beginning the Class Definition	748
	Searching and Retrieving	749
	Traversing	750
	Adding an Entry	751
	A Recursive Implementation	752
	An Iterative Implementation	755
	Removing an Entry	756
	Removing an Entry Whose Node Is a Leaf	757
	Removing an Entry Whose Node Has One Child	757
	Removing an Entry Whose Node Has Two Children	758
	Removing an Entry in the Root	761
	A Recursive Implementation	762
	An Iterative Implementation	765
	The Efficiency of Operations	769
	The Importance of Balance	770
	The Order in Which Nodes Are Added	770
	An Implementation of the ADT Dictionary	770
<b>Chapter 26</b>	<b>A Heap Implementation</b>	<b>783</b>
	Reprise: The ADT Heap	784
	Using an Array to Represent a Heap	784
	Adding an Entry	787
	Removing the Root	790
	Creating a Heap	793
	Heap Sort	796
<b>Chapter 27</b>	<b>Balanced Search Trees</b>	<b>805</b>
	AVL Trees	806
	Single Rotations	806
	Double Rotations	809
	Implementation Details	813
	2-3 Trees	817
	Searching a 2-3 Tree	818
	Adding Entries to a 2-3 Tree	819
	Splitting Nodes During Addition	821
	2-4 Trees	822
	Adding Entries to a 2-4 Tree	823
	Comparing AVL, 2-3, and 2-4 Trees	825



	Red-Black Trees	826
	Properties of a Red-Black Tree	827
	Adding Entries to a Red-Black Tree	828
	Java Class Library: The Class TreeMap	834
	B-Trees	834
<b>Chapter 28</b>	<b>Graphs</b>	<b>841</b>
	Some Examples and Terminology	842
	Road Maps	842
	Airline Routes	845
	Mazes	845
	Course Prerequisites	846
	Trees	846
	Traversals	847
	Breadth-First Traversal	848
	Depth-First Traversal	849
	Topological Order	851
	Paths	854
	Finding a Path	854
	The Shortest Path in an Unweighted Graph	854
	The Shortest Path in a Weighted Graph	857
	Java Interfaces for the ADT Graph	860
<b>Chapter 29</b>	<b>Graph Implementations</b>	<b>871</b>
	An Overview of Two Implementations	872
	The Adjacency Matrix	872
	The Adjacency List	873
	Vertices and Edges	874
	Specifying the Class Vertex	875
	The Inner Class Edge	877
	Implementing the Class Vertex	878
	An Implementation of the ADT Graph	881
	Basic Operations	881
	Graph Algorithms	884
<b>Appendix A</b>	<b>Documentation and Programming Style</b>	<b>891</b>
	Naming Variables and Classes	891
	Indenting	892
	Comments	892
	Single-Line Comments	893
	Comment Blocks	893
	When to Write Comments	893
	Java Documentation Comments	893
<b>Appendix B</b>	<b>Java Basics (online)</b>	
	Introduction	
	Applications and Applets	
	Objects and Classes	
	A First Java Application Program	
	Elements of Java	
	Identifiers	
	Reserved Words	
	Variables	

- Primitive Types
- Constants
- Assignment Statements
- Assignment Compatibilities
- Type Casting
- Arithmetic Operators and Expressions
- Parentheses and Precedence Rules
- Increment and Decrement Operators
- Special Assignment Operators
- Named Constants
- The Class Math
- Simple Input and Output Using the Keyboard and Screen
  - Screen Output
  - Keyboard Input Using the Class Scanner
- The if-else Statement
  - Boolean Expressions
  - Nested Statements
  - Multiway if-else Statements
  - The Conditional Operator (*Optional*)
- The switch Statement
- Enumerations
- Scope
- Loops
  - The while Statement
  - The for Statement
  - The do-while Statement
  - Additional Loop Information
- The Class String
  - Characters Within Strings
  - Concatenation of Strings
  - String Methods
- The Class StringBuilder
- Using Scanner to Extract Pieces of a String
- Arrays
  - Array Parameters and Returned Values
  - Initializing Arrays
  - Array Index Out of Bounds
  - Use of = and == with Arrays
  - Arrays and the For-Each Loop
  - Multidimensional Arrays
- Wrapper Classes

## Appendix C

- Java Classes (online)**
- Objects and Classes
- Using the Methods in a Java Class
  - References and Aliases
- Defining a Java Class
  - Method Definitions
  - Arguments and Parameters
  - Passing Arguments
  - A Definition of the Class Name

	Constructors	
	The Method <code>toString</code>	
	Methods That Call Other Methods	
	Methods That Return an Instance of Their Class	
	Static Fields and Methods	
	Overloading Methods	
	Enumeration as a Class	
	Packages	
	The Java Class Library	
<b>Appendix D</b>	<b>Creating Classes from Other Classes</b>	<b>899</b>
	Composition	900
	Adapters	902
	Inheritance	903
	Invoking Constructors from Within Constructors	906
	Private Fields and Methods of the Superclass	907
	Overriding and Overloading Methods	908
	Multiple Inheritance	913
	Type Compatibility and Superclasses	913
	The Class Object	914
<b>Appendix E</b>	<b>File Input and Output (online)</b>	
	Preliminaries	
	Why Files?	
	Streams	
	The Kinds of Files	
	File Names	
	Text Files	
	Creating a Text File	
	Reading a Text File	
	Changing Existing Data in a Text File	
	Defining a Method to Open a Stream	
	Binary Files	
	Creating a Binary File of Primitive Data	
	Reading a Binary File of Primitive Data	
	Strings in a Binary File	
	Object Serialization	
	<b>Glossary (online)</b>	
	<b>Index</b>	<b>919</b>

# VideoNotes Directory



VideoNote

This table lists the VideoNotes that are available online. The page numbers indicate where in the book each VideoNote has relevance.

<b>Chapter 1</b>	<b>Bags</b>	<b>61</b>
	Designing an ADT	63
	Designing a test for an ADT	71
<b>Java Interlude 1</b>	<b>Generics</b>	<b>83</b>
	Generics	84
<b>Chapter 2</b>	<b>Bag Implementations That Use Arrays</b>	<b>89</b>
	An array-based bag	91
	A resizable bag	117
<b>Java Interlude 2</b>	<b>Exceptions</b>	<b>125</b>
	Exceptions	126
<b>Chapter 3</b>	<b>A Bag Implementation That Links Data</b>	<b>133</b>
	Linked data	134
	Beginning the class <code>LinkBag</code>	139
	Completing the class <code>LinkBag</code>	144
<b>Chapter 4</b>	<b>The Efficiency of Algorithms</b>	<b>159</b>
	Measuring efficiency	161
	Comparing ADT bag implementations	173
<b>Chapter 5</b>	<b>Stacks</b>	<b>183</b>
	The ADT stack	184
	Using the ADT stack	199
<b>Chapter 6</b>	<b>Stack Implementations</b>	<b>211</b>
	The class <code>LinkStack</code>	212
	The class <code>ArrayStack</code>	215
<b>Chapter 7</b>	<b>Recursion</b>	<b>227</b>
	Introducing recursion	228
	Using recursion to solve problems	237
<b>Java Interlude 3</b>	<b>More About Generics</b>	<b>265</b>
	Generic classes and methods	267
<b>Chapter 8</b>	<b>An Introduction to Sorting</b>	<b>275</b>
	Selection sort	277
	Insertion sort	282
<b>Chapter 9</b>	<b>Faster Sorting Methods</b>	<b>301</b>
	Merge sort	302
	Quick sort	308
<b>Java Interlude 4</b>	<b>More About Exceptions</b>	<b>323</b>
	Creating your own exceptions	323
<b>Chapter 10</b>	<b>Queues, Deques, and Priority Queues</b>	<b>331</b>
	The ADT queue	332
	The ADTs deque and priority queue	352

<b>Chapter 11</b>	<b>Queue, Deque, and Priority Queue Implementations</b>	<b>361</b>
	The class <code>LinkedList</code>	362
	The class <code>ArrayQueue</code>	369
	Other queue implementations	374
<b>Chapter 12</b>	<b>Lists</b>	<b>391</b>
	The ADT list	392
	Using the ADT list	399
<b>Chapter 13</b>	<b>A List Implementation That Uses Array</b>	<b>409</b>
	The class <code>ArrayList</code>	412
	Completing the class <code>ArrayList</code>	417
<b>Chapter 14</b>	<b>A List Implementation That Links Data</b>	<b>427</b>
	The class <code>LinkedList</code>	436
	Completing the class <code>LinkedList</code>	442
<b>Java Interlude 5</b>	<b>Iterators</b>	<b>457</b>
	Iterators and their use	458
<b>Chapter 15</b>	<b>Iterators for the ADT List</b>	<b>473</b>
	Alternative iterator implementations	477
<b>Java Interlude 6</b>	<b>Mutable and Immutable Objects</b>	<b>499</b>
	Mutable and immutable objects	500
<b>Chapter 16</b>	<b>Sorted Lists</b>	<b>507</b>
	The class <code>LinkedListSorted</code>	512
	An array-based sorted list	520
<b>Java Interlude 7</b>	<b>Inheritance and Polymorphism</b>	<b>529</b>
	Inheritance	530
<b>Chapter 17</b>	<b>Inheritance and Lists</b>	<b>541</b>
	Inheritance and ADT implementations	542
	Creating a base class	549
<b>Chapter 18</b>	<b>Searching</b>	<b>557</b>
	Searching an array	559
	Searching a linked chain	570
<b>Java Interlude 8</b>	<b>Generics Once Again</b>	<b>579</b>
	Multitype generics	579
<b>Chapter 19</b>	<b>Dictionaries</b>	<b>581</b>
	The ADT dictionary	582
	Using the ADT dictionary	588
<b>Chapter 20</b>	<b>Dictionary Implementations</b>	<b>605</b>
	Array-based dictionaries	606
	Linked-chain dictionaries	616
<b>Chapter 21</b>	<b>Introducing Hashing</b>	<b>625</b>
	Hashing	626
	Resolving collisions	633
<b>Chapter 22</b>	<b>Hashing as a Dictionary Implementation</b>	<b>649</b>
	Hashing efficiency	650
	Implementing a dictionary	656

<b>Chapter 23</b>	<b>Trees</b>	<b>669</b>
	The ADT Tree	676
	Using a binary tree	683
<b>Chapter 24</b>	<b>Tree Implementations</b>	<b>703</b>
	Creating a binary tree	707
	Binary tree operations	711
<b>Java Interlude 9</b>	<b>Cloning</b>	<b>727</b>
	Cloneable objects	727
<b>Chapter 25</b>	<b>A Binary Search Tree Implementation</b>	<b>743</b>
	Creating a binary search tree	748
	Binary search tree additions and removals	751
<b>Chapter 26</b>	<b>A Heap Implementation</b>	<b>783</b>
	Implementing the ADT heap	784
	The heap sort	796
<b>Chapter 27</b>	<b>Balanced Search Trees</b>	<b>805</b>
	AVL trees	806
	2-3 trees	817
	2-4 and red-black trees	823
<b>Chapter 28</b>	<b>Graphs</b>	<b>841</b>
	Graph concepts and terminology	842
	Graph operations	847
<b>Chapter 29</b>	<b>Graph Implementations</b>	<b>871</b>
	The adjacency matrix	872
	Implementing graph operations	881

# Chapter Prerequisites

Each chapter and appendix assumes that the reader has studied certain previous material. This list indicates those prerequisites. Numbers represent chapter numbers, letters reference appendixes, and “JI” precedes each interlude number. You can use this information to plan a path through the book.

		Prerequisites
<b>Prelude</b>	Designing Classes	A, B, C, D
<b>Chapter 1</b>	Bags	Prelude, D
<b>Java Interlude 1</b>	Generics	Prelude
<b>Chapter 2</b>	Bag Implementations That Use Arrays	Prelude, 1
<b>Java Interlude 2</b>	Exceptions	B, C, D
<b>Chapter 3</b>	A Bag Implementation That Links Data	1, 2, JI2
<b>Chapter 4</b>	The Efficiency of Algorithms	2, 3, C
<b>Chapter 5</b>	Stacks	Prelude, 1, JI2
<b>Chapter 6</b>	Stack Implementations	2, 3, 4, 5
<b>Chapter 7</b>	Recursion	2, 3, 4, 5, C
<b>Java Interlude 3</b>	More About Generics	JII
<b>Chapter 8</b>	An Introduction to Sorting	3, 4, 7, JI3
<b>Chapter 9</b>	Faster Sorting Methods	4, 7, 8, JI3
<b>Java Interlude 4</b>	More About Exception	D, JI2
<b>Chapter 10</b>	Queues, Deques, and Priority Queues	Prelude, 5, 8
<b>Chapter 11</b>	Queue, Deque, and Priority Queue Implementations	2, 3, 6, 10
<b>Chapter 12</b>	Lists	Prelude, 6, C, JI2, JI3
<b>Chapter 13</b>	List Implementations That Use Arrays	Prelude, 2, 4, 12
<b>Chapter 14</b>	A List Implementation That Links Data	3, 11, 12, 13
<b>Java Interlude 5</b>	Iterators	12, JI2
<b>Chapter 15</b>	Iterators	13, 14, JI5
<b>Java Interlude 6</b>	Mutable and Immutable Objects	12, D
<b>Chapter 16</b>	Sorted Lists	4, 7, 12, 14
<b>Java Interlude 7</b>	Inheritance and Polymorphism	Prelude, 6, D
<b>Chapter 17</b>	Inheritance and Lists	12, 13, 14, 16, D, JI7
<b>Chapter 18</b>	Searching	4, 7, 12, 13, 14, 16
<b>Java Interlude 8</b>	Generics Once Again	C, JI3
<b>Chapter 19</b>	Dictionaries	12, 15, 18, JI5, JI8
<b>Chapter 20</b>	Dictionary Implementations	3, 4, 12, 13, 14, 18, 19, JI5
<b>Chapter 21</b>	Introducing Hashing	19, 20

		<b>Prerequisites</b>
<b>Chapter 22</b>	Hashing as a Dictionary Implementation	4, 13, 14, 19, 20, 21, JI5
<b>Chapter 23</b>	Trees	5, 7, 14, 18, JI5
<b>Chapter 24</b>	Tree Implementations	5, 10, 14, 23, D, JI2
<b>Java Interlude 9</b>	Cloning	16, 24, C, D, JI3, JI6
<b>Chapter 25</b>	A Binary Search Tree Implementation	7, 19, 23, 24, D
<b>Chapter 26</b>	A Heap Implementation	2, 13, 23
<b>Chapter 27</b>	Balanced Search Trees	23, 24, 25
<b>Chapter 28</b>	Graphs	5, 10, 23
<b>Chapter 29</b>	Graph Implementations	5, 10, 12, 15, 19, 23, 28, JI5
<b>Appendix A</b>	Documentation and Programming Style	Some knowledge of Java
<b>Appendix B</b>	Java Essentials	Programming knowledge
<b>Appendix C</b>	Java Classes	B
<b>Appendix D</b>	Creating Classes from Other Classes	C
<b>Appendix E</b>	File Input and Output	Prelude, B, JI2