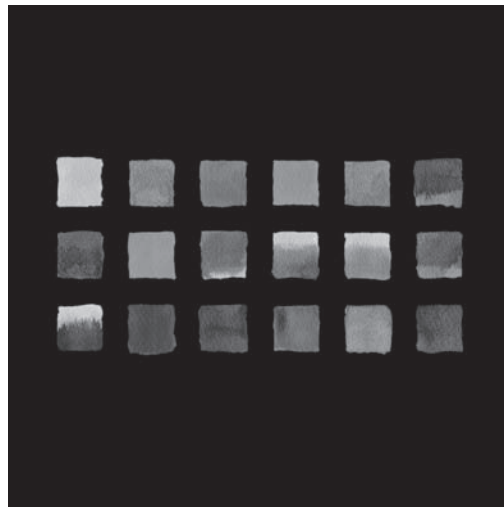


Data Structures and Abstractions with Java™

Fifth Edition



Frank M. Carrano
University of Rhode Island

Timothy M. Henry
New England Institute of Technology



330 Hudson Street, NY NY 10013

Senior Vice President Courseware Portfolio

Management: Marcia J. Horton

Director, Portfolio Management: Engineering,

Computer Science & Global Editions:

Julian Partridge

Executive Portfolio Manager: Tracy Johnson

Portfolio Management Assistant: Meghan Jacoby

Managing Content Producer: Scott Disanno

Content Producer: Lora Friedenthal

Rights and Permissions Manager: Ben Ferrini

Manufacturing Buyer, Higher Ed, Lake

Side Communications Inc (LSC):

Maura Zaldivar-Garcia

Inventory Manager: Bruce Bounty

Product Marketing Manager: Yvonne Vannatta

Field Marketing Manager: Demetrius Hall

Procurement Specialist: Maura Zaldivar-Garcia

Marketing Assistant: Jon Bryant

Cover Designer: Black Horse Designs

Cover Photo: Mustafahacalaki/E+/Getty Images

Printer/Binder: LSC Communications, Inc.

Full-Service Project Management: Billu Suresh,

SPi Global

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.

Copyright © 2019, 2015, 2012 and 2007 Pearson Education, Inc., All rights reserved. Printed in the United States of America. This publication is protected by Copyright, and permission should be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. To obtain permission(s) to use material from this work, please submit a written request to Pearson Education, Inc., Permissions Department, 221 River Street, Hoboken, NJ 07030, or you may fax your request to 201-236-3290.

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.

Library of Congress Cataloging-in-Publication Data

Names: Carrano, Frank M., author. | Henry, Timothy M., author.

Title: Data structures and abstractions with Java / Frank M. Carrano,

University of Rhode Island, Timothy M. Henry, New England Institute of Technology.

Description: Fifth edition. | Boston : Pearson Education, Inc., 2018. | Includes index.

Identifiers: LCCN 2018000065 | ISBN 9780134831695 (alk. paper) | ISBN 0134831691 (alk. paper)

Subjects: LCSH: Data structures (Computer science) | Java (Computer program language)

Classification: LCC QA76.9.D33 C37 2018 | DDC 005.7/3--dc23 LC record

available at <https://lccn.loc.gov/2018000065>

10 9 8 7 6 5 4 3 2 1



ISBN 10: 0-13-483169-1
ISBN 13: 978-0-13-483169-5

Welcome to the fifth 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 illustrations clarify complicated ideas. Included are over 60 video tutorials to supplement the instruction and help students when their instructor is unavailable.

I am happy to work again with my colleague and co-author of the fourth edition, Dr. Timothy Henry. Together we have continued to enhance the presentation with our focus on design decisions for both specifications and implementations of various data structures, as well as our emphasis on safe and secure programming practices.

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

Using This Flexible and Unique Textbook

This book’s organization, sequencing, and pace of topic coverage make teaching and learning easier by

- Focusing the reader’s attention on one concept at a time
- Providing flexibility in the order in which you can cover topics
- Clearly distinguishing between the specification and implementation of abstract data types (ADTs)
- Separating the relevant coverage of Java into Java Interludes, which you can use as needed

To accomplish this approach, we have organized the material into 30 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.

Our use of Java Interludes to treat the pertinent aspects of Java clearly separate our coverage of data structures from Java-specific issues. These interludes occur between chapters throughout the book as needed. Our focus, however, is on data structures not Java. You can see the titles of these interludes, as well as their placement between chapters, in the table of contents that follows.

Table of Contents at a Glance

The following brief table of contents shows the overall composition of the book. Notice the Introduction, Prelude, and nine Java Interludes. Further details—including a chapter-by-chapter description—are given later in this preface.

Introduction	Organizing Data	Chapter 17	Sorted Lists
Prelude	Designing Classes	Java Interlude 7	Inheritance and Polymorphism
Chapter 1	Bags	Chapter 18	Inheritance and Lists
Java Interlude 1	Generics	Chapter 19	Searching
Chapter 2	Bag Implementations That Use Arrays	Java Interlude 8	Generics Once Again
Java Interlude 2	Exceptions	Chapter 20	Dictionaries
Chapter 3	A Bag Implementation That Links Data	Chapter 21	Dictionary Implementations
Chapter 4	The Efficiency of Algorithms	Chapter 22	Introducing Hashing
Chapter 5	Stacks	Chapter 23	Hashing as a Dictionary Implementation
Chapter 6	Stack Implementations	Chapter 24	Trees
Java Interlude 3	More About Exceptions	Chapter 25	Tree Implementations
Chapter 7	Queues, Deques, and Priority Queues	Java Interlude 9	Cloning
Chapter 8	Queue, Deque, and Priority Queue Implementations	Chapter 26	A Binary Search Tree Implementation
Chapter 9	Recursion	Chapter 27	A Heap Implementation
Chapter 10	Lists	Chapter 28	Balanced Search Trees
Chapter 11	A List Implementation That Uses an Array	Chapter 29	Graphs
Chapter 12	A List Implementation That Links Data	Chapter 30	Graph Implementations
Java Interlude 4	Iterators	Appendix A	Documentation and Programming Style
Chapter 13	Iterators for the ADT List	Appendix B	Java Classes
Chapter 14	Problem Solving With Recursion	Appendix C	Creating Classes from Other Classes
Java Interlude 5	More About Generics	Supplement 1	Java Basics (online)
Chapter 15	An Introduction to Sorting	Supplement 2	File Input and Output (online)
Chapter 16	Faster Sorting Methods	Supplement 3	Glossary (online)
Java Interlude 6	Mutable and Immutable Objects	Supplement 4	Answers to Study Questions (online)

What's New?

This new edition of *Data Structures and Abstractions with Java* enhances the previous edition and continues its pedagogical approach to make the material accessible to students at the introductory level. The coverage that you enjoyed in previous editions is still here. As is usual for us, we have read every word of the previous edition and made changes to improve clarity and correctness. No chapter or interlude appears exactly as it did before. Our changes are motivated by reader suggestions and our own desire to improve the presentation.

In this new edition, we

- Added coverage of recursion in a new chapter that introduces grammars, languages, and backtracking.
- Continued our introduction to safe and secure programming practices.
- Added additional Design Decisions, Notes, Security Notes, and Programming Tips throughout the book.
- Added new exercises and programming projects, with an emphasis in areas of gaming, e-commerce, and finance to most chapters.
- Adjusted the order of some topics.
- Refined our terminology, presentation, and word choices to ease understanding.
- Revised the illustrations to make them easier to read and to understand.
- Renamed Self-Test Questions as Study Questions and moved their answers to online. We encourage our students to discuss their own answers with a study partner or group.
- Included the appendix about Java classes within the book instead of leaving it online.
- Reduced the amount of Java code given in the book.
- Ensured that all Java code is Java 9 compliant.

Connect with Us

We are always available to instructors and students who use our books. Your comments, suggestions, and corrections will be greatly appreciated. Please e-mail us at carrano@acm.org or timhenry@acm.org

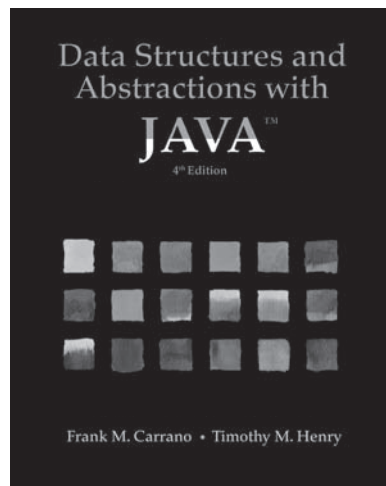
You can also find us on Twitter, our websites, or Facebook:

- Twitter: twitter.com/makingCSreal
- Websites: www.makingCSreal.com and timothyhenry.net
- Facebook: www.facebook.com/makingCSreal

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, B, and C review javadoc comments, Java classes, and inheritance. Java Interludes occur throughout the book and cover advanced aspects of Java as they are needed. Supplemental material is available online to review the basics of Java and its file input and output, to define important terms, and to answer the study questions. Note that inside the front and back covers 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

Each 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.



Study Questions Questions are posed throughout each chapter, integrated within the text, that reinforce the concept just presented. These “study” questions help readers to understand the material, since answering them requires pause and reflection. We suggest that you discuss these questions and their answers with others before consulting our solutions, which are available to you online.



VideoNote

VideoNotes Online tutorials are a Pearson feature that provides video 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 xxiv of this preface. VideoNotes are free with the purchase of a new textbook. To purchase access to VideoNotes, please go to

pearsonhighered.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 pearsonhighered.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 (IRC), accessible from pearsonhighered.com/carrano:

- Instructor's Guide
- PowerPoint lecture slides that have ADA compatible descriptive text for all images
- Instructor solutions manual
- Lab manual and solutions
- Java source code for instructors
- Figures from the book
- Test bank

Additionally, the resources available to students and described next, are available in the IRC to instructors.

Please contact your Pearson sales representative for an instructor access code. Contact information is available at pearsonhighered.com/relocator.

Student Resources

The following material is available to students by logging onto the Companion Website accessible from pearsonhighered.com/carrano:

- Instructional VideoNotes
- A survey of basic Java (Supplement 1)
- An overview of file I/O (Supplement 2)
- A glossary of terms (Supplement 3)
- Answers to Study Questions (Supplement 4)

Students must use the access card located in the front of the book to register for and then enter the Companion Website. Students without an access code can purchase access from the Companion Website by following the instructions listed there.

Note that documentation for the Java Class Library is available at docs.oracle.com/javase/9/docs/api/.

Content Overview

Readers of this book should have completed a programming course, preferably in Java. The appendixes and online supplements cover the essentials of Java that we assume readers will know. You can use this material as a review or as the basis for making the transition to Java from another programming language.

- **Introduction:** We begin by setting the stage for the data organizations that we will study by looking at some everyday examples.
- **Prelude:** The Prelude discusses the design of classes. Among the topics we cover are preconditions, postconditions, assertions, interfaces, and the Unified Modeling language (UML). Design is an important aspect of our presentation.
- **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 to the implementations of the ADT bag.
- **Chapter 4:** Here we introduce the efficiency and 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 chain of linked nodes, and a vector.
- **Java Interlude 3:** 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 7 and 8:** Chapter 7 discusses queues, dequeues, and priority queues, and Chapter 8 considers their implementations. It is in this latter chapter that we introduce circularly linked and doubly linked chains. Chapter 8 also uses the programmer-defined class `EmptyQueueException`.
- **Chapter 9:** 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.
- **Chapters 10, 11, and 12:** 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 4 and Chapter 13:** Included in our coverage of Java iterators are the standard interfaces `Iterator`, `Iterable`, and `ListIterator`. Chapter 13 then shows ways to implement an iterator for the ADT list. It considers and implements Java's iterator interfaces `Iterator` and `ListIterator`.
- **Chapter 14:** This new chapter offers more coverage of recursion, including languages, grammars, and backtracking.
- **Java Interlude 5:** 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 15 and 16:** The next two chapters discuss various sorting techniques and their relative complexities. We consider both iterative and recursive versions of these algorithms.
- **Java Interlude 6:** This interlude discusses mutable and immutable objects, a topic relevant to the preceding chapters about sorting and the following one about sorted lists.
- **Chapters 17 and 18 and Java Interlude 7:** Continuing the discussion of a list, Chapter 17 introduces the sorted list, looking at two possible implementations and their efficiencies. Chapter 18 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 C, the relevant particulars of inheritance—including protected access, abstract classes, and abstract methods—are presented in Java Interlude 7 just before Chapter 18.
- **Chapter 19:** 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 20 through 23:** Chapter 20 covers the specification and use of the ADT dictionary. Next, Chapter 21 presents implementations of the dictionary that are linked or that use arrays. Chapter 22 introduces hashing, and Chapter 23 uses hashing as a dictionary implementation.
- **Chapters 24 and 25:** Chapter 24 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 25 considers implementations of the binary tree and the general tree.
- **Java Interlude 9:** Java Interlude 9 discusses cloning. 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 26 through 28:** Chapter 26 focuses on the implementation of the binary search tree. Chapter 27 shows how to use an array to implement the heap. Chapter 28 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 29 and 30:** Finally, we discuss graphs and look at several applications and two implementations.
- **Appendixes A through C:** 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 discusses Java classes, and Appendix C expands this topic by looking at composition and inheritance.

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:

Prakash Duraisamy—*Miami University*
David Gries—*Cornell University*
Hakam Alomari—*Miami University*
Jack Pope—*North Hennepin Community College*
Anna Rafferty—*Carleton College*

Special thanks go to our support team at Pearson Education Computer Science during the lengthy process of revising this book: Executive Portfolio Manager Tracy Johnson, Portfolio Management Assistant Meghan Jacoby, Managing Content Producer Scott Disanno, and Content Producer Lora Friedenthal have always be 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. Tim Henry produced the VideoNotes and the lecture slides for this edition. Professor Charles Hoot of the Oklahoma City University created the lab manual, Professor Kathy Liszka from the University of Akron created the collection of test questions, and Joe Erickson 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 fourth edition:

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*

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, Steve Armstrong, James Blanding, Lianne Dunn, Bob Englehardt, 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, Bob, 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

Contents

	Introduction: Organizing Data	1
Prelude: Designing Classes		5
	Encapsulation	6
	Specifying Methods	8
	Comments	8
	Preconditions and Postconditions	9
	Assertions	10
	Java Interfaces	11
	Writing a Java Interface	12
	Implementing an Interface	13
	An Interface as a Data Type	15
	Extending an Interface	16
	Named Constants Within an Interface	17
	Choosing Classes	19
	Identifying Classes	20
	CRC Cards	21
	The Unified Modeling Language	21
	Reusing Classes	24
Chapter 1	Bags	29
	The Bag	30
	A Bag's Behaviors	30
	Specifying a Bag	31
	An Interface	37
	Using the ADT Bag	39
	Using an ADT Is Like Using a Vending Machine	43
	The ADT Set	45
	Java Class Library: The Interface Set	45
Java Interlude 1	Generics	51
	Generic Data Types	51
	Generic Types Within an Interface	52
	Generic Classes	53
Chapter 2	Bag Implementations That Use Arrays	57
	Using a Fixed-Size Array to Implement the ADT Bag	58
	An Analogy	58
	A Group of Core Methods	59
	Implementing the Core Methods	60
	Making the Implementation Secure	66
	Testing the Core Methods	69
	Implementing More Methods	71
	Methods That Remove Entries	74
	Using Array Resizing to Implement the ADT Bag	82
	Resizing an Array	82
	A New Implementation of a Bag	85
	The Pros and Cons of Using an Array to Implement the ADT Bag	87
Java Interlude 2	Exceptions	93
	The Basics	94
	Handling an Exception	96

	Postpone Handling: The throws Clause	96
	Handle It Now: The try-catch Blocks	97
	Multiple catch Blocks	98
	Throwing an Exception	99
Chapter 3	A Bag Implementation That Links Data	103
	Linked Data	104
	Forming a Chain by Adding to Its Beginning	105
	A Linked Implementation of the ADT Bag	107
	The Private Class Node	107
	An Outline of the Class <code>LinkedBag</code>	108
	Defining Some Core Methods	109
	Testing the Core Methods	113
	The Method <code>getFrequencyOf</code>	114
	The Method <code>contains</code>	115
	Removing an Item from a Linked Chain	116
	The Methods <code>remove</code> and <code>clear</code>	117
	A Class Node That Has Set and Get Methods	121
	The Pros and Cons of Using a Chain to Implement the ADT Bag	124
Chapter 4	The Efficiency of Algorithms	129
	Motivation	130
	Measuring an Algorithm's Efficiency	131
	Counting Basic Operations	133
	Best, Worst, and Average Cases	135
	Big Oh Notation	136
	The Complexities of Program Constructs	138
	Picturing Efficiency	140
	The Efficiency of Implementations of the ADT Bag	143
	An Array-Based Implementation	143
	A Linked Implementation	145
	Comparing the Implementations	146
Chapter 5	Stacks	153
	Specifications of the ADT Stack	154
	Using a Stack to Process Algebraic Expressions	158
	A Problem Solved: Checking for Balanced Delimiters in an Infix Algebraic Expression	159
	A Problem Solved: Transforming an Infix Expression to a Postfix Expression	164
	A Problem Solved: Evaluating Postfix Expressions	168
	A Problem Solved: Evaluating Infix Expressions	170
	The Program Stack	173
	Java Class Library: The Class Stack	174
Chapter 6	Stack Implementations	181
	A Linked Implementation	181
	An Array-Based Implementation	185
	A Vector-Based Implementation	189
	Java Class Library: The Class <code>Vector</code>	190
	Using a Vector to Implement the ADT Stack	190

Java Interlude 3	More About Exceptions	197
	Programmer-Defined Exception Classes	197
	Inheritance and Exceptions	201
	The finally Block	202
Chapter 7	Queues, Deques, and Priority Queues	205
	The ADT Queue	206
	A Problem Solved: Simulating a Waiting Line	210
	A Problem Solved: Computing the Capital Gain in a Sale of Stock	216
	Java Class Library: The Interface Queue	219
	The ADT Deque	220
	A Problem Solved: Computing the Capital Gain in a Sale of Stock	223
	Java Class Library: The Interface Deque	224
	Java Class Library: The Class ArrayDeque	225
	The ADT Priority Queue	225
	A Problem Solved: Tracking Your Assignments	227
	Java Class Library: The Class PriorityQueue	229
Chapter 8	Queue, Deque, and Priority Queue Implementations	237
	A Linked Implementation of a Queue	238
	An Array-Based Implementation of a Queue	242
	A Circular Array	242
	A Circular Array with One Unused Element	245
	Circular Linked Implementations of a Queue	251
	A Two-Part Circular Linked Chain	251
	Java Class Library: The Class AbstractQueue	257
	A Doubly Linked Implementation of a Deque	257
	Possible Implementations of a Priority Queue	261
Chapter 9	Recursion	267
	What Is Recursion?	268
	Tracing a Recursive Method	272
	Recursive Methods That Return a Value	275
	Recursively Processing an Array	277
	Recursively Processing a Linked Chain	280
	The Time Efficiency of Recursive Methods	282
	The Time Efficiency of countDown	282
	The Time Efficiency of Computing x^n	283
	Tail Recursion	284
	Using a Stack Instead of Recursion	285
Chapter 10	Lists	295
	Specifications for the ADT List	296
	Using the ADT List	303
	A Problem Solved: Working with Huge Integers	306
	Java Class Library: The Interface List	308
	Java Class Library: The Class ArrayList	309
Chapter 11	A List Implementation That Uses an Array	315
	Using an Array to Implement the ADT List	316
	An Analogy	316
	The Java Implementation	318
	The Efficiency of Using an Array to Implement the ADT List	326

Chapter 12	A List Implementation That Links Data	333
	Operations on a Chain of Linked Nodes	334
	Adding a Node at Various Positions	334
	Removing a Node from Various Positions	338
	The Private Method <code>getNodeAt</code>	339
	Beginning the Implementation	340
	The Data Fields and Constructor	341
	Adding to the End of the List	343
	Adding at a Given Position Within the List	344
	The Methods <code>isEmpty</code> and <code>toArray</code>	345
	Testing the Core Methods	347
	Continuing the Implementation	348
	A Refined Implementation	351
	The Tail Reference	351
	The Efficiency of Using a Chain to Implement the ADT List	355
	Java Class Library: The Class <code>LinkedList</code>	357
Java Interlude 4	Iterators	361
	What Is an Iterator?	361
	The Interface <code>Iterator</code>	363
	The Interface <code>Iterable</code>	365
	Using the Interface <code>Iterator</code>	365
	<code>Iterable</code> and for-each loops	369
	The Interface <code>ListIterator</code>	370
	The Interface <code>List</code> Revisited	373
	Using the Interface <code>ListIterator</code>	374
Chapter 13	Iterators for the ADT List	377
	Ways to Implement an Iterator	378
	A Separate Class Iterator	378
	An Inner Class Iterator	381
	A Linked Implementation	382
	An Array-Based Implementation	385
	Why Are Iterator Methods in Their Own Class?	388
	An Array-Based Implementation of the Interface <code>ListIterator</code>	390
	The Inner Class	391
Chapter 14	Problem Solving with Recursion	403
	A Simple Solution to a Difficult Problem	404
	A Poor Solution to a Simple Problem	410
	Languages and Grammars	412
	The Language of Java Identifiers	412
	The Language of Prefix Expressions	413
	Evaluating Prefix Expressions	416
	Indirect Recursion	417
	Backtracking	418
	Walking Through a Maze	418
	The <i>n</i> -Queens Problem	420
Java Interlude 5	More About Generics	429
	The Interface <code>Comparable</code>	429
	Generic Methods	431
	Bounded Type Parameters	432

	Wildcards	434
	Bounded Wildcards	435
Chapter 15	An Introduction to Sorting	437
	Organizing Java Methods That Sort an Array	438
	Selection Sort	439
	Iterative Selection Sort	440
	Recursive Selection Sort	442
	The Efficiency of Selection Sort	443
	Insertion Sort	443
	Iterative Insertion Sort	445
	Recursive Insertion Sort	447
	The Efficiency of Insertion Sort	449
	Insertion Sort of a Chain of Linked Nodes	449
	Shell Sort	452
	The Algorithm	454
	The Efficiency of Shell Sort	455
	Comparing the Algorithms	455
Chapter 16	Faster Sorting Methods	461
	Merge Sort	462
	Merging Arrays	462
	Recursive Merge Sort	463
	The Efficiency of Merge Sort	465
	Iterative Merge Sort	467
	Merge Sort in the Java Class Library	467
	Quick Sort	468
	The Efficiency of Quick Sort	468
	Creating the Partition	469
	Implementing Quick Sort	472
	Quick Sort in the Java Class Library	474
	Radix Sort	474
	Pseudocode for Radix Sort	475
	The Efficiency of Radix Sort	476
	Comparing the Algorithms	476
Java Interlude 6	Mutable and Immutable Objects	483
	Mutable Objects	483
	Immutable Objects	485
	Creating a Read-Only Class	486
	Companion Classes	487
Chapter 17	Sorted Lists	491
	Specifications for the ADT Sorted List	492
	Using the ADT Sorted List	495
	A Linked Implementation	496
	The Method add	497
	The Efficiency of the Linked Implementation	504
	An Implementation That Uses the ADT List	504
	Efficiency Issues	507
Java Interlude 7	Inheritance and Polymorphism	513
	Further Aspects of Inheritance	513
	When to Use Inheritance	513

	Protected Access	514
	Abstract Classes and Methods	515
	Interfaces Versus Abstract Classes	517
	Polymorphism	518
Chapter 18	Inheritance and Lists	525
	Using Inheritance to Implement a Sorted List	526
	Designing a Base Class	528
	Creating an Abstract Base Class	533
	An Efficient Implementation of a Sorted List	535
	The Method <code>add</code>	536
Chapter 19	Searching	541
	The Problem	542
	Searching an Unsorted Array	542
	An Iterative Sequential Search of an Unsorted Array	543
	A Recursive Sequential Search of an Unsorted Array	544
	The Efficiency of a Sequential Search of an Array	546
	Searching a Sorted Array	546
	A Sequential Search of a Sorted Array	546
	A Binary Search of a Sorted Array	547
	Java Class Library: The Method <code>binarySearch</code>	552
	The Efficiency of a Binary Search of an Array	552
	Searching an Unsorted Chain	553
	An Iterative Sequential Search of an Unsorted Chain	554
	A Recursive Sequential Search of an Unsorted Chain	554
	The Efficiency of a Sequential Search of a Chain	555
	Searching a Sorted Chain	555
	A Sequential Search of a Sorted Chain	555
	A Binary Search of a Sorted Chain	556
	Choosing a Search Method	556
Java Interlude 8	Generics Once Again	563
	More Than One Generic Type	563
Chapter 20	Dictionaries	565
	Specifications for the ADT Dictionary	566
	A Java Interface	570
	Iterators	572
	Using the ADT Dictionary	573
	A Problem Solved: A Directory of Telephone Numbers	573
	A Problem Solved: The Frequency of Words	578
	A Problem Solved: A Concordance of Words	581
	Java Class Library: The Interface <code>Map</code>	584
Chapter 21	Dictionary Implementations	591
	Array-Based Implementations	592
	An Unsorted Array-Based Dictionary	592
	A Sorted Array-Based Dictionary	597
	Linked Implementations	602
	An Unsorted Linked Dictionary	603
	A Sorted Linked Dictionary	604

Chapter 22	Introducing Hashing	611
	What Is Hashing?	612
	Hash Functions	615
	Computing Hash Codes	615
	Compressing a Hash Code into an Index for the Hash Table	618
	Resolving Collisions	619
	Open Addressing with Linear Probing	619
	Open Addressing with Quadratic Probing	626
	Open Addressing with Double Hashing	626
	A Potential Problem with Open Addressing	629
	Separate Chaining	629
Chapter 23	Hashing as a Dictionary Implementation	637
	The Efficiency of Hashing	638
	The Load Factor	638
	The Cost of Open Addressing	639
	The Cost of Separate Chaining	641
	Rehashing	642
	Comparing Schemes for Collision Resolution	643
	A Dictionary Implementation That Uses Hashing	644
	Entries in the Hash Table	644
	Data Fields and Constructors	645
	The Methods <code>getValue</code> , <code>remove</code> , and <code>add</code>	647
	Iterators	649
	Java Class Library: The Class <code>HashMap</code>	650
	Java Class Library: The Class <code>HashSet</code>	651
Chapter 24	Trees	655
	Tree Concepts	656
	Hierarchical Organizations	656
	Tree Terminology	658
	Traversals of a Tree	662
	Traversals of a Binary Tree	663
	Traversals of a General Tree	665
	Java Interfaces for Trees	666
	Interfaces for All Trees	666
	An Interface for Binary Trees	667
	Examples of Binary Trees	668
	Expression Trees	669
	Decision Trees	670
	Binary Search Trees	673
	Heaps	675
	Examples of General Trees	678
	Parse Trees	678
	Game Trees	678
Chapter 25	Tree Implementations	685
	The Nodes in a Binary Tree	686
	A Class of Binary Nodes	687
	An Implementation of the ADT Binary Tree	688
	Creating a Basic Binary Tree	689
	The Method <code>initializeTree</code>	690
	Accessor and Mutator Methods	692

	Computing the Height and Counting Nodes	693
	Traversals	694
	An Implementation of an Expression Tree	699
	General Trees	700
	A Node for a General Tree	700
	Using a Binary Tree to Represent a General Tree	701
Java Interlude 9	Cloning	709
	Cloneable Objects	709
	Cloning an Array	716
	Cloning a Chain	718
	A Sorted List of Clones	722
	Cloning a Binary Node	724
Chapter 26	A Binary Search Tree Implementation	727
	Getting Started	728
	An Interface for the Binary Search Tree	729
	Duplicate Entries	731
	Beginning the Class Definition	732
	Searching and Retrieving	733
	Traversing	734
	Adding an Entry	735
	A Recursive Implementation	736
	An Iterative Implementation	739
	Removing an Entry	740
	Removing an Entry Whose Node Is a Leaf	741
	Removing an Entry Whose Node Has One Child	741
	Removing an Entry Whose Node Has Two Children	742
	Removing an Entry in the Root	745
	A Recursive Implementation	746
	An Iterative Implementation	749
	The Efficiency of Operations	753
	The Importance of Balance	754
	The Order in Which Nodes Are Added	754
	An Implementation of the ADT Dictionary	754
Chapter 27	A Heap Implementation	765
	Reprise: The ADT Heap	766
	Using an Array to Represent a Heap	766
	Adding an Entry	769
	Removing the Root	772
	Creating a Heap	775
	Heap Sort	778
Chapter 28	Balanced Search Trees	785
	AVL Trees	786
	Single Rotations	786
	Double Rotations	789
	Implementation Details	793
	2-3 Trees	797
	Searching a 2-3 Tree	798
	Adding Entries to a 2-3 Tree	799
	Splitting Nodes During Addition	801

	2-4 Trees	802
	Adding Entries to a 2-4 Tree	803
	Comparing AVL, 2-3, and 2-4 Trees	805
	Red-Black Trees	806
	Properties of a Red-Black Tree	807
	Adding Entries to a Red-Black Tree	808
	Java Class Library: The Class TreeMap	814
	B-Trees	814
Chapter 29	Graphs	819
	Some Examples and Terminology	820
	Road Maps	820
	Airline Routes	823
	Mazes	823
	Course Prerequisites	824
	Trees	824
	Traversals	825
	Breadth-First Traversal	826
	Depth-First Traversal	827
	Topological Order	829
	Paths	832
	Finding a Path	832
	The Shortest Path in an Unweighted Graph	832
	The Shortest Path in a Weighted Graph	835
	Java Interfaces for the ADT Graph	838
Chapter 30	Graph Implementations	849
	An Overview of Two Implementations	850
	The Adjacency Matrix	850
	The Adjacency List	851
	Vertices and Edges	852
	Specifying the Class Vertex	853
	The Inner Class Edge	855
	Implementing the Class Vertex	856
	An Implementation of the ADT Graph	859
	Basic Operations	859
	Graph Algorithms	862
Appendix A	Documentation and Programming Style	869
	Naming Variables and Classes	869
	Indenting	870
	Comments	870
	Single-Line Comments	871
	Comment Blocks	871
	When to Write Comments	871
	Java Documentation Comments	871
Appendix B	Java Classes	875
	Objects and Classes	875
	Using the Methods in a Java Class	877
	References and Aliases	878
	Defining a Java Class	879
	Method Definitions	881

	Arguments and Parameters	883
	Passing Arguments	883
	A Definition of the Class Name	887
	Constructors	888
	The Method <code>toString</code>	891
	Methods That Call Other Methods	891
	Methods That Return an Instance of Their Class	892
	Static Fields and Methods	893
	Overloading Methods	894
	Enumeration as a Class	895
	Packages	898
	The Java Class Library	899
Appendix C	Creating Classes from Other Classes	901
	Composition	902
	Adapters	904
	Inheritance	905
	Invoking Constructors from Within Constructors	908
	Private Fields and Methods of the Superclass	909
	Overriding and Overloading Methods	910
	Multiple Inheritance	915
	Type Compatibility and Superclasses	915
	The Class <code>Object</code>	916
Supplement 1	Java Basics (online)	
	Introduction	2
	Applications and Applets	2
	Objects and Classes	2
	A First Java Application Program	3
	Elements of Java	5
	Identifiers	5
	Reserved Words	5
	Variables	6
	Primitive Types	7
	Constants	7
	Assignment Statements	7
	Assignment Compatibilities	8
	Type Casting	9
	Arithmetic Operators and Expressions	10
	Parentheses and Precedence Rules	11
	Increment and Decrement Operators	12
	Special Assignment Operators	13
	Named Constants	14
	The Class <code>Math</code>	15
	Simple Input and Output Using the Keyboard and Screen	15
	Screen Output	15
	Keyboard Input Using the Class <code>Scanner</code>	17
	The <code>if-else</code> Statement	19
	Boolean Expressions	20
	Nested Statements	23
	Multiway <code>if-else</code> Statements	24
	The Conditional Operator (<i>Optional</i>)	25

The switch Statement	26
Enumerations	28
Scope	30
Loops	30
The while Statement	31
The for Statement	32
The do-while Statement	34
Additional Loop Information	35
The Class String	36
Characters Within Strings	36
Concatenation of Strings	37
String Methods	38
The Class StringBuilder	40
Using Scanner to Extract Pieces of a String	42
Arrays	44
Array Parameters and Returned Values	46
Initializing Arrays	47
Array Index Out of Bounds	47
Use of = and == with Arrays	47
Arrays and the For-Each Loop	48
Multidimensional Arrays	49
Wrapper Classes	51
Supplement 2 File Input and Output (online)	
Preliminaries	2
Why Files?	2
Streams	2
The Kinds of Files	3
File Names	3
Text Files	3
Creating a Text File	3
Reading a Text File	8
Changing Existing Data in a Text File	11
Defining a Method to Open a Stream	12
Binary Files	13
Creating a Binary File of Primitive Data	13
Reading a Binary File of Primitive Data	17
Strings in a Binary File	19
Object Serialization	20
Supplement 3 Glossary (online)	
Supplement 4 Answers to Study Questions (online)	
Index	919

VideoNotes Directory



VideoNote

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

Chapter 1	Bags	29
	Designing an ADT	31
	Designing a test for an ADT	39
Java Interlude 1	Generics	51
	Generics	52
Chapter 2	Bag Implementations That Use Arrays	57
	An array-based bag	59
	A resizable bag	85
Java Interlude 2	Exceptions	93
	Exceptions	94
Chapter 3	A Bag Implementation That Links Data	103
	Linked data	104
	Beginning the class <code>LinkedBag</code>	109
	Completing the class <code>LinkedBag</code>	115
Chapter 4	The Efficiency of Algorithms	129
	Measuring efficiency	131
	Comparing ADT bag implementations	143
Chapter 5	Stacks	153
	The ADT stack	154
	Using the ADT stack	169
Chapter 6	Stack Implementations	181
	The Class <code>LinkedStack</code>	182
	The Class <code>ArrayStack</code>	185
Java Interlude 3	More About Exceptions	197
	Creating your own exceptions	197
Chapter 7	Queues, Deques, and Priority Queues	205
	The ADT queue	206
	The ADTs deque and priority queue	226
Chapter 8	Queue, Deque, and Priority Queue Implementations	237
	The Class <code>LinkedStack</code>	238
	The class <code>ArrayQueue</code>	245
	Other queue implementations	251
Chapter 9	Recursion	267
	Introducing recursion	268
	Recursively processing structures	277
Chapter 10	Lists	295
	The ADT list	296
	Using the ADT list	303

Chapter 11	A List Implementation That Uses an Array	315
	The class AList	319
	Completing the class AList	323
Chapter 12	A List Implementation That Links Data	333
	The class LLi st	342
	Completing the class LLi st	348
Java Interlude 4	Iterators	361
	Iterators and their use	362
Chapter 13	Iterators for the ADT List	377
	Alternative iterator implementations	381
Chapter 14	Problem Solving with Recursion	403
	Processing expressions	412
	Backtracking	418
Java Interlude 5	More About Generics	429
	Generic classes and methods	431
Chapter 15	An Introduction to Sorting	437
	Selection sort	439
	Insertion sort	444
Chapter 16	Faster Sorting Methods	461
	Merge sort	462
	Quick sort	468
Java Interlude 6	Mutable and Immutable Objects	483
	Mutable and immutable objects	483
Chapter 17	Sorted Lists	491
	The class linkedsortedlist	496
	An array-based sorted list	504
Java Interlude 7	Inheritance and Polymorphism	513
	Inheritance	514
Chapter 18	Inheritance and Lists	525
	Inheritance and ADT implementations	526
	Creating a base class	533
Chapter 19	Searching	541
	Searching an array	542
	Searching a linked chain	554
Java Interlude 8	Generics Once Again	563
	Multitype generics	563
Chapter 20	Dictionaries	565
	The ADT dictionary	566
	Using the ADT dictionary	573
Chapter 21	Dictionary Implementations	591
	Array-based dictionaries	592
	Linked-chain dictionaries	602
Chapter 22	Introducing Hashing	611
	Hashing	612
	Resolving collisions	619

Chapter 23	Hashing as a Dictionary Implementation	637
	Hashing efficiency	638
	Implementing a dictionary	644
Chapter 24	Trees	655
	The ADT Tree	662
	Using a binary tree	669
Chapter 25	Tree Implementations	685
	Creating a binary tree	689
	Binary tree operations	693
Java Interlude 9	Cloning	709
	Cloneable objects	709
Chapter 26	A Binary Search Tree Implementation	727
	Creating a binary search tree	732
	Binary search tree additions and removals	735
Chapter 27	A Heap Implementation	765
	Implementing the ADT heap	766
	The heap sort	778
Chapter 28	Balanced Search Trees	785
	AVL trees	786
	2-3 tree	797
	2-4 and red-black trees	803
Chapter 29	Graphs	819
	Graph concepts and terminology	820
	Graph operations	825
Chapter 30	Graph Implementations	849
	The adjacency matrix	850
	Implementing graph operations	859

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
Prelude	Designing Classes	Supplement 1, A, B, C
Chapter 1	Bags	Prelude, C
Java Interlude 1	Generics	Prelude
Chapter 2	Bag Implementations That Use Arrays	Prelude, 1
Java Interlude 2	Exceptions	Supplement 1, B, C
Chapter 3	A Bag Implementation That Links Data	1, 2, JI2
Chapter 4	The Efficiency of Algorithms	B, 2, 3
Chapter 5	Stacks	Prelude, 1, JI2
Chapter 6	Stack Implementations	2, 3, 4, 5
Java Interlude 3	More About Exceptions	C, JI2
Chapter 7	Queues, Deques, and Priority Queues	Prelude, 5
Chapter 8	Queue, Deque, and Priority Queue Implementations	2, 3, 6, 7
Chapter 9	Recursion	B, 2, 3, 4, 5
Chapter 10	Lists	Introduction, B, Prelude, JI2, 6
Chapter 11	A List Implementation That Uses an Array	Prelude, 2, 4, 10
Chapter 12	A List Implementation That Links Data	3, 8, 10, 11
Java Interlude 4	Iterators	JI2, 10
Chapter 13	Iterators for the ADT List	11, 12, JI4
Chapter 14	Problem Solving with Recursion	5, 9
Java Interlude 5	More About Generics	JI1
Chapter 15	An Introduction to Sorting	3, 4, 9, JI5
Chapter 16	Faster Sorting Methods	4, 9, JI5, 15
Java Interlude 6	Mutable and Immutable Objects	C, 10
Chapter 17	Sorted Lists	4, 9, 10, 12, JI6
Java Interlude 7	Inheritance and Polymorphism	Prelude, C, 6
Chapter 18	Inheritance and Lists	C, 10, 11, 12, 17, JI7
Chapter 19	Searching	4, 9, 10, 11, 12, 17
Java Interlude 8	Generics Once Again	B, JI5
Chapter 20	Dictionaries	10, JI4, 13, 19, JI8
Chapter 21	Dictionary Implementations	3, 4, 10, 11, 12, JI4, 19, 20

		Prerequisites
Chapter 22	Introducing Hashing	20, 21
Chapter 23	Hashing as a Dictionary Implementation	4, 11, 12, JI4, 20, 21, 22
Chapter 24	Trees	5, 9, 12, JI4, 19
Chapter 25	Tree Implementations	C, JI2, 5, 7, 12, 24
Java Interlude 9	Cloning	JI5, JI6, 17, 25, B, C
Chapter 26	A Binary Search Tree Implementation	C, 9, 20, 24, 25
Chapter 27	A Heap Implementation	2, 11, 24
Chapter 28	Balanced Search Trees	24, 25, 26
Chapter 29	Graphs	5, 7, 24
Chapter 30	Graph Implementations	5, 7, 10, JI4, 13, 20, 24, 29
Appendix A	Documentation and Programming Style	Some knowledge of Java
Appendix B	Java Classes	Supplement 1
Appendix C	Creating Classes from Other Classes	B
Supplement 1	Java Basics	Knowledge of a programming language
Supplement 2	File Input and Output	Prelude, JI2, Supplement 1