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$^{TM}$*, 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$^{TM}$*, 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**
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$^{TM}$* 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** 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

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

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.

## 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:

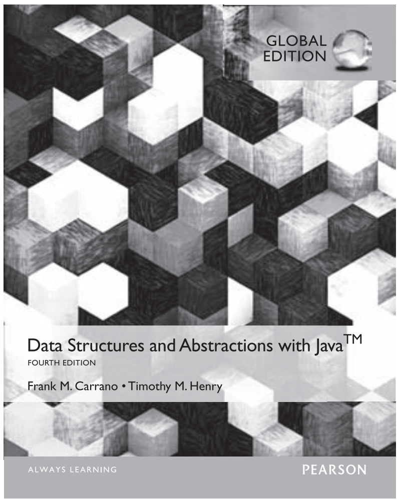
- Chapter 1 introduces the ADT set in addition to the bag.
- Chapter 2 introduces safe and secure programming. The code changes suggested here are integrated into all ADT implementations in subsequent chapters.
- Chapters 5 and 6 use exceptions in the specification and implementations of the ADT stack.
- 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.


GLOBAL EDITION
Data Structures and Abstractions with Java™
FOURTH EDITION
Frank M. Carrano • Timothy M. Henry
ALWAYS LEARNING
PEARSON

## 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.

SELF-TEST

**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`

**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`:

- 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`:

- 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`:

- 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`:

- 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/`.

## 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, deques, 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 McGlinn—*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

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

# 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 | A, B, C, D |
| **Chapter 1** | Bags | Prelude, D |
| **Java Interlude 1** | Generics | Prelude |
| **Chapter 2** | Bag Implementations That Use Arrays | Prelude, 1 |
| **Java Interlude 2** | Exceptions | B, C, D |
| **Chapter 3** | A Bag Implementation That Links Data | 1, 2, JI2 |
| **Chapter 4** | The Efficiency of Algorithms | 2, 3, C |
| **Chapter 5** | Stacks | Prelude, 1, JI2 |
| **Chapter 6** | Stack Implementations | 2, 3, 4, 5 |
| **Chapter 7** | Recursion | 2, 3, 4, 5, C |
| **Java Interlude 3** | More About Generics | JI1 |
| **Chapter 8** | An Introduction to Sorting | 3, 4, 7, JI3 |
| **Chapter 9** | Faster Sorting Methods | 4, 7, 8, JI3 |
| **Java Interlude 4** | More About Exception | D, JI2 |
| **Chapter 10** | Queues, Deques, and Priority Queues | Prelude, 5, 8 |
| **Chapter 11** | Queue, Deque, and Priority Queue Implementations | 2, 3, 6, 10 |
| **Chapter 12** | Lists | Prelude, 6, C, JI2, JI3 |
| **Chapter 13** | List Implementations That Use Arrays | Prelude, 2, 4, 12 |
| **Chapter 14** | A List Implementation That Links Data | 3, 11, 12, 13 |
| **Java Interlude 5** | Iterators | 12, JI2 |
| **Chapter 15** | Iterators | 13, 14, JI5 |
| **Java Interlude 6** | Mutable and Immutable Objects | 12, D |
| **Chapter 16** | Sorted Lists | 4, 7, 12, 14 |
| **Java Interlude 7** | Inheritance and Polymorphism | Prelude, 6, D |
| **Chapter 17** | Inheritance and Lists | 12, 13, 14, 16, D, JI7 |
| **Chapter 18** | Searching | 4, 7, 12, 13, 14, 16 |
| **Java Interlude 8** | Generics Once Again | C, JI3 |
| **Chapter 19** | Dictionaries | 12, 15, 18, JI5, JI8 |
| **Chapter 20** | Dictionary Implementations | 3, 4, 12, 13, 14, 18, 19, JI5 |
| **Chapter 21** | Introducing Hashing | 19, 20 |

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