OBJECT-ORIENTED DATA STRUCTURES USING Java™

FOURTH EDITION

NELL DALE
UNIVERSITY OF TEXAS, AUSTIN

DANIEL T. JOYCE
VILLANOVA UNIVERSITY

CHIP WEEMS
UNIVERSITY OF MASSACHUSETTS, AMHERST

JONES & BARTLETT LEARNING

World Headquarters
Jones & Bartlett Learning
5 Wall Street
Burlington, MA 01803
978-443-5000
info@jblearning.com
www.jblearning.com

Jones & Bartlett Learning books and products are available through most bookstores and online booksellers. To contact Jones & Bartlett Learning directly, call 800-832-0034, fax 978-443-8000, or visit our website, www.jblearning.com.

Substantial discounts on bulk quantities of Jones & Bartlett Learning publications are available to corporations, professional associations, and other qualified organizations. For details and specific discount information, contact the special sales department at Jones & Bartlett Learning via the above contact information or send an email to specialsales@jblearning.com.

09820-4

Production Credits
VP, Executive Publisher: David D. Cella
Acquisitions Editor: Laura Pagluica
Editorial Assistant: Taylor Ferracane
Director of Vendor Management: Amy Rose
Marketing Manager: Amy Langlais
VP, Manufacturing and Inventory Control: Therese Connell
Composition and Project Management: S4Carlisle Publishing Services
Cover Design: Kristin E. Parker
Text Design: Scott Moden
Rights & Media Specialist: Merideth Tumasz
Media Development Editor: Shannon Sheehan
Cover Image: © Ake13bk/Shutterstock
Printing and Binding: Edwards Brothers Malloy
Cover Printing: Edwards Brothers Malloy

Library of Congress Cataloging-in-Publication Data
Names: Dale, Nell (Nell B.), author. | Joyce, Daniel T., author. | Weems, Chip., author.
Title: Object-oriented data structures using Java / Nell Dale, Daniel T. Joyce, Chip Weems.
Description: Fourth edition. | Burlington, MA : Jones & Bartlett Learning, [2017]
Identifiers: LCCN 2016025145 | ISBN 9781284089097 (casebound)
Subjects: LCSH: Object-oriented programming (Computer science) | Data structures (Computer science) | Java (Computer program language)
Classification: LCC QA76.64 .D35 2017 | DDC 005.13/3--dc23 LC record available at https://lccn.loc.gov/2016025145

6048

Printed in the United States of America
20 19 18 17 16 10 9 8 7 6 5 4 3 2 1

To Alfred G. Dale
ND

To Kathy, Tom, and Julie, thanks for the love and support
DJ

To Lisa, Charlie, and Abby, thank you . . .
CW

Preface

Welcome to the fourth edition of *Object-Oriented Data Structures Using Java™*. This book presents the algorithmic, programming, and structuring techniques of a traditional data structures course in an object-oriented context. You'll find the familiar topics of linked lists, recursion, stacks, queues, collections, indexed lists, trees, maps, priority queues, graphs, sorting, searching, and complexity analysis, all covered from an object-oriented point of view using Java. We stress software engineering principles throughout, including modularization, information hiding, data abstraction, stepwise refinement, the use of visual aids, the analysis of algorithms, and software verification methods.

To the Student

You know that an algorithm is a sequence of unambiguous instructions for solving a problem. You can take a problem of moderate complexity, design a small set of classes/objects that work together to solve the problem, code the method algorithms needed to make the objects work, and demonstrate the correctness of your solution.

Algorithms describe actions. These actions manipulate data. For most interesting problems that are solved using computers, the structure of the data is just as important as the structure of the algorithms used to manipulate the data. Using this text you will discover that the way you structure data affects how efficiently you can use the data; you will see how the nature of the problem you are attempting to solve dictates your structuring decisions; and you will learn about the data structures that computer scientists have developed over the years to help solve problems.

Object-Oriented Programming with Java

Our primary goal is to present both the traditional and modern data structure topics with an emphasis on problem solving and software design. Using the Java programming language as a vehicle for problem solutions, however, presents an opportunity for students to expand their

familiarity with a modern programming language and the object-oriented paradigm. As our data structure coverage unfolds, we introduce and use the appropriate Java constructs that support our primary goal. Starting early and continuing throughout the text, we introduce and expand on the use of many Java features such as classes, objects, generics, polymorphism, packages, interfaces, library classes, inheritance, exceptions, and threads. We also use Universal Modeling Language (UML) class diagrams throughout to help model and visualize our objects, classes, interfaces, applications, and their interrelationships.

Features

Data Abstraction In this text we view our data structures from three different perspectives: their specification, their application, and their implementation. The specification describes the logical or abstract level—*what* the logical relationships among the data elements are and *what* operations can be performed on the structure. The application level, sometimes called the client level, is concerned with how the data structure is used to solve a problem—*why* the operations do what they do. The implementation level involves the coding details—*how* the structures and operations are implemented. In other words we treat our data structures as abstract data types (ADTs).

Efficiency Analysis In Chapter 1 we introduce order of growth efficiency analysis using a unique approach involving the interaction of two students playing a game. Time and space analysis is consistently applied throughout the text, allowing us to compare and contrast data structure implementations and the applications that use them.

Recursion Treatment Recursion is introduced early (Chapter 3) and used throughout the remainder of the text. We present a design and analysis approach to recursion based on answering three simple questions. Answering the questions, which are based on formal inductive reasoning, leads the programmer to a solid recursive design and program.

Interesting Applications Eight primary data structures (stacks, queues, collections, indexed lists, trees, maps, priority queues, and graphs) are treated in separate chapters that include their definition, several implementations, and one or more interesting applications based on their use. Applications involve, for example, balanced expressions, postfix expressions, image generation (new!), fractals (new!), queue simulation, card decks and games (new!), text analysis (new!), tree and graph traversals, and big integers.

Robust Exercises We average more than 40 exercises per chapter. The exercises are organized by chapter sections to make them easier for you to manage. They vary in level of difficulty, including short and long programming problems (marked with "programming-required" icons—one icon to indicate short exercises and two icons for projects), the analysis of algorithms, and problems to test students' understanding of abstract concepts. In this edition we have streamlined the previous exercises, allowing us to add even more options for you to choose from. In particular we have added several larger programming exercises to many of the chapters.

Input/Output Options It is difficult to know what background the students using a data structures text will have in Java I/O. To allow all the students using our text to concentrate on the

primary topic of data structures, we use the simplest I/O approach we can, namely a command line interface. However, to support those teachers and students who prefer to work with graphical user interfaces (GUIs), we provide GUIs for many of our applications. Our modular approach to program design supports this approach—our applications separate the user interface code, problem solution code, and ADT implementation code into separate classes.

Concurrency Coverage We are pleased to be one of the only data structures texts to address the topics of concurrency and synchronization, which are growing in importance as computer systems move to using more cores and threads to obtain additional performance with each new generation. We introduce this topic in Section 4.9, "Concurrency, Interference, and Synchronization," where we start with the basics of Java threads, continue through examples of thread interference and synchronization, and culminate in a discussion of efficiency concerns.

New to the *Fourth Edition*

This edition represents a major revision of the text's material, although the philosophy and style that our loyal adopters have grown to appreciate remain unchanged. We removed material we felt was redundant or of lesser/outdated importance to the core topic of data structures, added new key material, and reworked much of the material that we kept. Although the length of the textbook was reduced by about 10%, the coverage of data structures has been expanded. We believe this new edition is a great improvement over previous editions and hope you do, too. Major changes include:

- Simplified Architecture: We continue to use the Java interface construct to define the abstract view of our ADTs, but we have reduced the number of levels of inheritance, simplifying the architecture and making it easier to understand and use.
- New Chapters: Chapter 5, "The Collection ADT," and Chapter 8, "The Map ADT," are brand new. The Collection ADT material introduces the idea of a data structure as a repository and concentrates on storage and retrieval of data based on key attributes. The Map ADT has become increasingly important with the rise in popularity of scripting languages with built-in associative arrays.
- New Section: Section 1.6, "Comparing Algorithms: Order of Growth Analysis," was completely rewritten and features an introduction to efficiency analysis driven by a game played between two students, plus analysis of sequential search, binary search, and sequential sort algorithms.
- New Sections: In response to reader's suggestions, Chapter 3, "Recursion," features two new sections: Section 3.3, "Recursive Processing of Arrays," is devoted to recursive processing of arrays and Section 3.4, "Recursive Processing of Linked Lists," is devoted to recursive processing of linked lists. These new sections provide practical examples of the use of recursion, before the reader moves on to the less practical but nevertheless popular Towers of Hanoi example covered in Section 3.5, "Towers."
- New Section: Fractals! A fun section related to recursively generating fractal-based images now wraps up the examples of Chapter 3, "Recursion."

- New Sections: We added "Variations" sections to the Stack, Queue, Collection, List, Tree, and Map chapters. In the primary exposition of each of these ADTs we record design decisions and specify the operations to be supported by the ADT. We also develop or at least discuss various implementation approaches, in most cases highlighting one array-based approach and one reference/linked-list-based approach. The "Variations" section discusses alternate approaches to defining/implementing the ADT and in most cases reviews the ADT counterparts available in the standard Java Library. Some of these sections also introduce related ADTs, for example, in the "Variations" section of the Collection chapter we define and discuss both the Set and Bag ADTs.
- Glossary: The text's glossary has always been available online. With this edition we make it available as Appendix E. Throughout the text we highlight important terms that might be unfamiliar to the student in **green**, the first time they are featured, to indicate that their definition can be found in the glossary.

Prerequisite Assumptions

In this book, we assume that readers are familiar with the following Java constructs:

- Built-in simple data types and the array type
- Control structures *while*, *do*, *for*, *if*, and *switch*
- Creating and instantiating objects
- Basic user-defined classes:
  - variables and methods
  - constructors, method parameters, and the *return* statement
  - visibility modifiers
- Commonly used Java Library Classes: *Integer, Math, Random, Scanner*, *String*, and *System*

Chapter Content

Chapter 1 is all about **Getting Organized**. An overview of object orientation stresses mechanisms for organizing objects and classes. The Java exception handling mechanisms, used to organize response to unusual situations, are introduced. Data structures are previewed and the two fundamental language constructs that are used to implement those structures, the array and the reference (link/pointer), are discussed. The chapter concludes with a look at efficiency analysis—how we evaluate and compare algorithms.

Chapter 2 presents **The Stack ADT**. The concept of abstract data type (ADT) is introduced. The stack is viewed from three different levels: the abstract, application, and implementation levels. The Java interface mechanism is used to support this three-tiered view. We also investigate using generics to support generally usable ADTs. The Stack ADT is implemented using both arrays and references. To support the reference-based approach we introduce the linked list structure. Sample applications include determining if a set of grouping symbols is well formed and the evaluation of postfix expressions.

Chapter 3 discusses **Recursion**, showing how recursion can be used to solve programming problems. A simple three-question technique is introduced for verifying the correctness of recursive methods. Sample applications include array processing, linked list processing, the classic Towers of Hanoi, and fractal generation. A detailed discussion of how recursion works shows how recursion can be replaced with iteration and stacks.

Chapter 4 presents **The Queue ADT**. It is also first considered from its abstract perspective, followed by a formal specification, and then implemented using both array-based and reference-based approaches. Example applications include an interactive test driver, a palindrome checker, and simulating a system of real-world queues. Finally, we look at Java's concurrency and synchronization mechanisms, explaining issues of interference and efficiency.

Chapter 5 defines **The Collection ADT**. A fundamental ADT, the Collection, supports storing information and then retrieving it later based on its content. Approaches for comparing objects for equality and order are reviewed. Collection implementations using an array, a sorted array, and a linked list are developed. A text processing application permits comparison of the implementation approaches for efficiency. The "Variations" section introduces two more well-known ADTs: the Bag and the Set.

Chapter 6 follows up with a more specific Collection ADT, **The List ADT**. In fact, the following two chapters also develop Collection ADTs. Iteration is introduced here and the use of anonymous inner classes to provide iterators is presented. As with the Collection ADT we develop array, sorted array, and linked-list–based implementations. The "Variations" section includes an example of how to "implement" a linked list within an array. Applications include a card deck model plus some card games, and a Big Integer class. This latter application demonstrates how we sometimes design specialized ADTs for specific problems.

Chapter 7 develops **The Binary Search Tree ADT**. It requires most of the chapter just to design and create our reference-based implementation of this relatively complex structure. The chapter also discusses trees in general (including breadth-first and depth-first searching) and the problem of balancing a binary search tree. A wide variety of special-purpose and self-balancing trees are introduced in the "Variations" section.

Chapter 8 presents **The Map ADT**, also known as a symbol table, dictionary, or associative array. Two implementations are developed, one that uses an *ArrayList* and the other that uses a hash table. A large part of the chapter is devoted to this latter implementation and the important concept of hashing, which provides a very efficient implementation of a Map. The "Variations" section discusses a map-based hybrid data structure plus Java's support for hashing.

Chapter 9 introduces **The Priority Queue ADT**, which is closely related to the Queue but with a different accessing protocol. This short chapter does present a sorted array-based implementation, but most of the chapter focuses on a clever, interesting, and very efficient implementation called a Heap.

Chapter 10 covers **The Graph ADT**, including implementation approaches and several important graph-related algorithms (depth-first search, breadth-first search, path existence, shortest paths, and connected components). The graph algorithms make use of stacks, queues, and priority queues, thus both reinforcing earlier material and demonstrating the general usability of these structures.

Chapter 11 presents/reviews a number of **Sorting and Searching Algorithms**. The sorting algorithms that are illustrated, implemented, and compared include straight selection sort, two versions of bubble sort, insertion sort, quick sort, heap sort, and merge sort. The sorting algorithms are compared using efficiency analysis. The discussion of algorithm analysis continues in the context of searching. Previously presented searching algorithms are reviewed and new ones are described.

Organization

Chapter Goals Sets of knowledge and skill goals are presented at the beginning of each chapter to help the students assess what they have learned.

Sample Programs Numerous sample programs and program segments illustrate the abstract concepts throughout the text.

Feature Sections Throughout the text these short sections highlight topics that are not directly part of the flow of material but nevertheless are related and important.

Boxed Notes These small boxes of information scattered throughout the text highlight, supplement, and reinforce the text material, perhaps from a slightly different point of view.

Chapter Summaries Each chapter concludes with a summary section that reviews the most important topics of the chapter and ties together related topics. Some chapter summaries include a UML diagram of the major interfaces and classes developed within the chapter.

Appendices The appendices summarize the Java reserved word set, operator precedence, primitive data types, the ASCII subset of Unicode, and provide a glossary of important terms used in the text.

Website http://go.jblearning.com/oods4e

This website provides access to the text's source code files for each chapter. Additionally, registered instructors are able to access selected answers to the text's exercises, a test item file, and presentation slides. Please contact the authors if you have material related to the text that you would like to share with others.

Acknowledgments

We would like to thank the following people who took the time to review this text: Mark Llewellyn at the University of Central Florida, Chenglie Hu at Carroll College, Val Tannen at the University of Pennsylvania, Chris Dovolis at the University of Minnesota, Mike Coe at Plano Senior High School, Mikel Petty at University of Alabama in Huntsville, Gene Sheppard at Georgia Perimeter College, Noni Bohonak at the University of South Carolina–Lancaster, Jose Cordova at the University of Louisiana–Monroe, Judy Gurka at the Metropolitan State College of Denver, Mikhail Brikman at Salem State University, Amitava Karmaker at University of Wisconsin–Stout, Guifeng Shao at Tennessee State University, Urska Cvek at Louisiana State University at Shreveport, Philip C. Doughty Jr. at Northern Virginia Community College, Jeff Kimball at Southwest Baptist University, Jeremy T. Lanman at Nova Southeastern University, Rao Li at University of South Carolina Aiken, Larry Thomas at University of Toledo, and Karen Works at Westfield State University. A special thanks to Christine Shannon at Centre College, to Phil LaMastra at Fairfield University, to Allan Gottlieb of New York University, and to J. William Cupp at Indiana Wesleyan University for specific comments leading to improvements in the text. A personal thanks to Kristen Obermyer, Tara Srihara, Sean Wilson, Christopher Lezny, and Naga Lakshmi, all of Villanova University, plus Kathy, Tom, and Julie Joyce for all of their help, support, and proofreading expertise.

A virtual bouquet of roses to the editorial and production teams who contributed so much, especially Laura Pagluica, Taylor Ferracane, Amy Rose, and Palaniappan Meyyappan.

ND
DJ
CW

Contents