INTERNATIONAL EDITION

Java Software Structures

Designing and Using Data Structures

FOURTH EDITION

John Lewis • Joseph Chase

ALWAYS LEARNING

PEARSON

Java™ Software Structures

DESIGNING AND USING DATA STRUCTURES

4TH EDITION

This page is intentionally left blank.

Java™ Software Structures

DESIGNING AND USING DATA STRUCTURES

4TH EDITION

JOHN LEWIS
Virginia Tech

AND

JOSEPH CHASE
Radford University

International Edition contributions by
PIYALI SENGUPTA

PEARSON

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

Acquisitions Editor, US Edition: Matt Goldstein
Editorial Assistant: Jenah Blitz-Stoehr
Senior Managing Editor: Scott Disanno
Senior Production Supervisor: Marilyn Lloyd
Marketing Manager: Yes Alayan
Marketing Coordinator: Kathryn Ferranti
Publisher, International Edition: Angshuman Chakraborty
Publishing Administrator and Business Analyst, International Edition: Shokhi Shah Khandelwal
Associate Print & Media Editor, International Edition: Anuprova Dey Chowdhuri
Acquisitions Editor, International Edition: Sandhya Ghoshal
Publishing Administrator, International Edition: Hema Mehta
Project Editor, International Edition: Karthik Subramanian
Senior Manufacturing Controller, Production, International Edition: Trudy Kimber
Marketing Coordinator: Kathryn Ferranti
Manufacturing Buyer: Lisa McDowell
Cover Design: Jodi Notowitz
Project Management and Illustrations: Cenveo® Publisher Services
Project Manager, Cenveo® Publisher Services, Inc.: Rose Kernan
Text Design, Cenveo® Publisher Services, Inc.: Jerilyn Bockorick, Alisha Webber
Cover Image: Viachaslau Kraskouski/Shutterstock

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.pearsoninternationaleditions.com

ISBN 10: 0-273-79332-2
ISBN 13: 978-0-273-79332-8 (Print)
ISBN 13: 978-0-273-79368-7 (PDF)

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
14 13 12 11 10

Typeset in Sabon LT Std Roman by Cenveo® Publisher Services

Printed and bound by Courier Westford in The United States of America
The publisher's policy is to use paper manufactured from sustainable forests.

To my wife Sharon and my kids:
Justin, Kayla, Nathan, and Samantha
–J. L.

To my loving wife Melissa for her support and encouragement
and to our families, friends, colleagues, and students who have provided
so much support and inspiration through the years.
–J. C.

This page is intentionally left blank.

Preface

This book is designed to serve as a text for a course on data structures and algorithms. This course is typically referred to as the CS2 course because it is often taken as the second course in a computing curriculum.

Pedagogically, this book follows the style and approach of the leading CS1 book **Java Software Solutions: Foundations of Program Design,** by John Lewis and William Loftus. Our book uses many of the highly regarded features of that book, such as the Key Concept boxes and complete code examples. Together, these two books support a solid and consistent approach to either a two-course or three-course introductory sequence for computing students. That said, this book does not assume that students have used **Java Software Solutions** in a previous course.

Material that might be presented in either course (such as recursion or sorting) is presented in this book as well. We also include strong reference material providing an overview of object-oriented concepts and how they are realized in Java.

We understand the crucial role that the data structures and algorithms course plays in a curriculum and we think this book serves the needs of that course well.

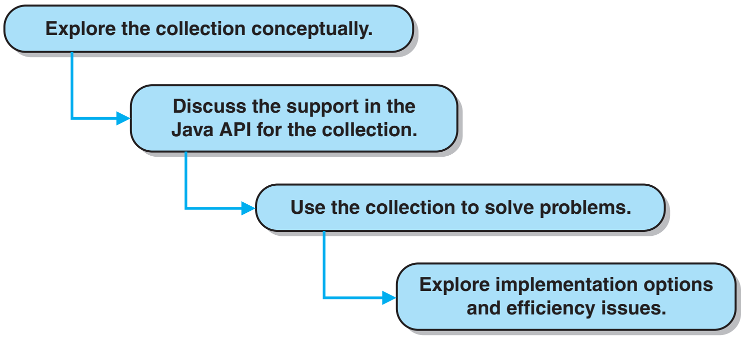
New in the Fourth Edition

We have made some key modifications in this fourth edition to enhance its pedagogy. They can be summarized as follows:

- Revised the collection chapters to provide a more complete explanation of how the Java API supports the collection.
- Added a summary of terms and definitions at the end of each chapter.
- Separated the coverage of Iterators into it's own chapter and expanded the discussion.
- Added a new Code Annotation feature, used to explore key statements with graphic annotations.
- Added a new Common Error callout feature.
- Added new Design Focus callouts.

- Added a new appendices covering graphical drawing, graphical user interface development, and regular expressions.
- Reviewed and updated the text throughout to improve discussions and address issues.

In particular, we've reworked the discussion of individual collections to match the following flow:


Explore the collection conceptually.
Discuss the support in the Java API for the collection.
Use the collection to solve problems.
Explore implementation options and efficiency issues.

This approach clarifies the distinction between the way the Java API supports a particular collection and the way it might be implemented from scratch. It makes it easier for instructors to point out limitations of the API implementations in a compare-and-contrast fashion. This approach also allows an instructor, on a case-by-case basis, to simply introduce a collection without exploring implementation details if desired.

The other modifications for this edition flesh out the presentation to a higher degree than previous editions did. The addition of a term list (with succinct definitions) at the end of each chapter provides a summary of core issues in ways that the other features don't. New Code Annotation and Common Error features highlight specific issues that might otherwise get lost in the body of the text, but without interrupting the flow of the topic.

We think these modifications build upon the strong pedagogy established by previous editions and give instructors more opportunity and flexibility to cover topics as they choose.

Our Approach

Books of this type vary greatly in their overall approach. Our approach is founded on a few important principles that we fervently embraced. First, we present the various collections explored in the book in a consistent manner. Second, we emphasize the importance of sound software design techniques. Third, we organized the book to support and reinforce the big picture: the study of data structures and algorithms.

Throughout the book, we keep sound software engineering practices a high priority. Our design of collection implementations and the programs that use them follow consistent and appropriate standards.

Of primary importance is the separation of a collection's interface from its underlying implementation. The services that a collection provides are always formally defined in a Java interface. The interface name is used as the type designation of the collection whenever appropriate to reinforce the collection as an abstraction.

Chapter Breakdown

Chapter 1 (**Introduction**) discusses various aspects of software quality and provides an overview of software development issues. It is designed to establish the appropriate mindset before embarking on the details of data structure and algorithm design.

Chapter 2 (**Analysis of Algorithms**) lays the foundation for determining the efficiency of an algorithm and explains the important criteria that allow a developer to compare one algorithm to another in proper ways. Our emphasis in this chapter is understanding the important concepts more than getting mired in heavy math or formality.

Chapter 3 (**Introduction to Collections—Stacks**) establishes the concept of a collection, stressing the need to separate the interface from the implementation. It also conceptually introduces a stack, then explores an array-based implementation of a stack.

Chapter 4 (**Linked Structures—Stacks**) discusses the use of references to create linked data structures. It explores the basic issues regarding the management of linked lists, and then defines an alternative implementation of a stack (introduced in Chapter 3) using an underlying linked data structure.

Chapter 5 (**Queues**) explores the concept and implementation of a first-in, first-out queue. Radix sort is discussed as an example of using queues effectively. The implementation options covered include an underlying linked list as well as both fixed and circular arrays.

Chapter 6 (**Lists**) covers three types of lists: ordered, unordered, and indexed. These three types of lists are compared and contrasted, with discussion of the operations that they share and those that are unique to each type. Inheritance is used appropriately in the design of the various types of lists, which are implemented using both array-based and linked representations.

Chapter 7 (**Iterators**) is a new chapter that isolates the concepts and implementation of iterators, which are so important to collections. The expanded discussion drives home the need to separate the iterator functionality from the details of any particular collection.

Chapter 8 (**Recursion**) is a general introduction to the concept of recursion and how recursive solutions can be elegant. It explores the implementation details of recursion and discusses the basic idea of analyzing recursive algorithms.

Chapter 9 (**Searching and Sorting**) discusses the linear and binary search algorithms, as well as the algorithms for several sorts: selection sort, insertion sort, bubble sort, quick sort, and merge sort. Programming issues related to searching and sorting, such as using the Comparable interface as the basis of comparing objects, are stressed in this chapter. Searching and sorting that are based in particular data structures (such as heap sort) are covered in the appropriate chapter later in the book.

Chapter 10 (**Trees**) provides an overview of trees, establishing key terminology and concepts. It discusses various implementation approaches and uses a binary tree to represent and evaluate an arithmetic expression.

Chapter 11 (**Binary Search Trees**) builds off of the basic concepts established in Chapter 10 to define a classic binary search tree. A linked implementation of a binary search tree is examined, followed by a discussion of how the balance in the tree nodes is key to its performance. That leads to exploring AVL and red/black implementations of binary search trees.

Chapter 12 (**Heaps and Priority Queues**) explores the concept, use, and implementations of heaps and specifically their relationship to priority queues. A heap sort is used as an example of its usefulness as well. Both linked and array-based implementations are explored.

Chapter 13 (**Sets and Maps**) explores these two types of collections and their importance to the Java Collections API.

Chapter 14 (**Multi-way Search Trees**)is a natural extension of the discussion of the previous chapters. The concepts of 2-3 trees, 2-4 trees, and general B-trees are examined and implementation options are discussed.

Chapter 15 (**Graphs**) explores the concept of undirected and directed graphs and establishes important terminology. It examines several common graph algorithms and discusses implementation options, including adjacency matrices.

Appendix A (**UML**) provides an introduction to the Unified Modeling Language as a reference. UML is the de facto standard notation for representing object-oriented systems.

Appendix B (**Object-Oriented Concepts**) is a reference for anyone needing a review of fundamental object-oriented concepts and how they are accomplished in Java. Included are the concepts of abstraction, classes, encapsulation, inheritance, and polymorphism, as well as many related Java language constructs such as interfaces.

Appendix C (**Graphics**) covers the basics of drawing shapes using the Java API.

Appendix D (**Graphical User Interfaces**) provides a detailed overview of the elements needed to develop a Swing-based GUI. It includes many examples using a variety of interface components.

Appendix E (**Hashing**) covers the concept of hashing and related issues, such as hash functions and collisions. Various Java Collections API options for hashing are discussed.

Appendix F (**Regular Expressions**) provides an introduction to the use of regular expressions, which come into play in various Java API elements, such as the Scanner class.

Supplements

The following student resources are available for this book:

- **Source code** for all programs presented in the book
- **VideoNotes** that explore select topics from the book

Resources can be accessed at www.pearsoninternationaleditions.com/lewis

The following instructor resources can be found at Pearson Education's Instructor Resource Center:

- **Solutions** for select exercises and programming projects in the book
- **Powerpoint slides** for the presentation of the book content
- **Test bank**

To obtain access, please visit www.pearsoninternationaleditions.com/lewis or contact your local Pearson Education sales representative.

Pearson would also like to thank Mohit P. Tahiliani of NITK Surathkal for reviewing the content of the International Edition.

This page is intentionally left blank.

Contents