

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

© Pearson Education Limited 2014

The rights of John Lewis and Joseph Chase to be identified as authors of this work have been asserted by them in accordance with the Copyright, Designs and Patents Act 1988.

Authorized adaptation from the United States edition, entitled Java Software Structures, 4th edition, ISBN 978-0-13-325012-1, by John Lewis and Joseph Chase, published by Pearson Education © 2014.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without either the prior written permission of the publisher or a license permitting restricted copying in the United Kingdom issued by the Copyright Licensing Agency Ltd, Saffron House, 6–10 Kirby Street, London EC1N 8TS.

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

Microsoft and/or its respective suppliers make no representations about the suitability of the information contained in the documents and related graphics published as part of the services for any purpose. All such documents and related graphics are provided “as is” without warranty of any kind. Microsoft and/or its respective suppliers hereby disclaim all warranties and conditions with regard to this information, including all warranties and conditions of merchantability, whether express, implied or statutory, fitness for a particular purpose, title and non-infringement. In no event shall Microsoft and/or its respective suppliers be liable for any special, indirect or consequential damages or any damages whatsoever resulting from loss of use, data or profits, whether in an action of contract, negligence or other tortious action, arising out of or in connection with the use or performance of information available from the services.

The documents and related graphics contained herein could include technical inaccuracies or typographical errors. Changes are periodically added to the information herein. Microsoft and/or its respective suppliers may make improvements and/or changes in the product(s) and/or the program(s) described herein at any time. Partial screen shots may be viewed in full within the software version specified.

Microsoft® and Windows® are registered trademarks of the Microsoft Corporation in the U.S.A. and other countries. This book is not sponsored or endorsed by or affiliated with the Microsoft Corporation.

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.

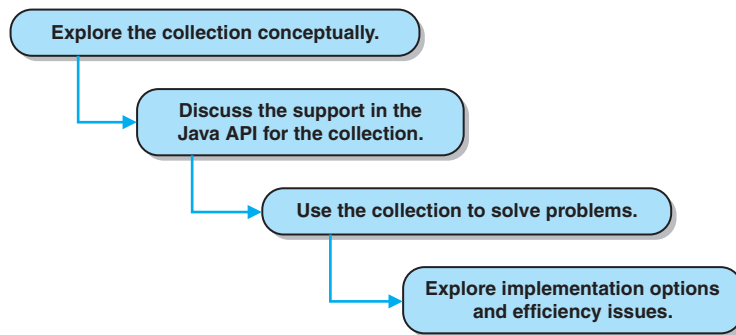
New in the Fourth Edition

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 its 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:



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

Preface		7
Credits		25
Chapter 1	Introduction	27
	1.1 Software Quality	28
	Correctness	29
	Reliability	29
	Robustness	30
	Usability	30
	Maintainability	31
	Reusability	31
	Portability	32
	Efficiency	32
	Quality Issues	32
	1.2 Data Structures	33
	A Physical Example	33
	Containers as Objects	36
Chapter 2	Analysis of Algorithms	41
	2.1 Algorithm Efficiency	42
	2.2 Growth Functions and Big-Oh Notation	43
	2.3 Comparing Growth Functions	45
	2.4 Determining Time Complexity	48
	Analyzing Loop Execution	48
	Nested Loops	48
	Method Calls	49

Chapter 3	Introduction to Collections – Stacks	55
3.1	Collections	56
	Abstract Data Types	57
	The Java Collections API	59
3.2	A Stack Collection	59
3.3	Crucial OO Concepts	61
	Inheritance and Polymorphism	62
	Generics	63
3.4	Using Stacks: Evaluating Postfix Expressions	64
	Javadoc	71
3.5	Exceptions	72
3.6	A Stack ADT	74
3.7	Implementing a Stack: With Arrays	77
	Managing Capacity	78
3.8	The ArrayStack Class	79
	The Constructors	80
	The push Operation	82
	The pop Operation	83
	The peek Operation	85
	Other Operations	85
	The EmptyCollectionException Class	85
	Other Implementations	86
Chapter 4	Linked Structures – Stacks	93
4.1	References as Links	94
4.2	Managing Linked Lists	96
	Accessing Elements	96
	Inserting Nodes	97
	Deleting Nodes	98
4.3	Elements without Links	99
	Doubly Linked Lists	99
4.4	Stacks in the Java API	100

4.5	Using Stacks: Traversing a Maze	101
4.6	Implementing a Stack: With Links	110
	The <code>LinkedList</code> Class	110
	The <code>push</code> Operation	114
	The <code>pop</code> Operation	116
	Other Operations	117
Chapter 5	Queues	123
5.1	A Conceptual Queue	124
5.2	Queues in the Java API	125
5.3	Using Queues: Code Keys	126
5.4	Using Queues: Ticket Counter Simulation	130
5.5	A Queue ADT	135
5.6	A Linked Implementation of a Queue	137
	The <code>enqueue</code> Operation	139
	The <code>dequeue</code> Operation	141
	Other Operations	142
5.7	Implementing Queues: With Arrays	143
	The <code>enqueue</code> Operation	147
	The <code>dequeue</code> Operation	149
	Other Operations	150
5.8	Double-Ended Queues (Deque)	150
Chapter 6	Lists	155
6.1	A List Collection	156
6.2	Lists in the Java Collections API	158
6.3	Using Unordered Lists: Program of Study	159
6.4	Using Indexed Lists: Josephus	170
6.5	A List ADT	172
	Adding Elements to a List	173

	6.6	Implementing Lists with Arrays	178
		The remove Operation	180
		The contains Operation	182
		The add Operation for an Ordered List	183
		Operations Particular to Unordered Lists	185
		The addAfter Operation for an Unordered List	185
	6.7	Implementing Lists with Links	186
		The remove Operation	187
Chapter 7	Iterators		195
	7.1	What's an Iterator?	196
		Other Iterator Issues	198
	7.2	Using Iterators: Program of Study Revisited	198
		Printing Certain Courses	202
		Removing Courses	203
	7.3	Implementing Iterators: With Arrays	205
	7.4	Implementing Iterators: With Links	207
Chapter 8	Recursion		213
	8.1	Recursive Thinking	214
		Infinite Recursion	214
		Recursion in Math	215
	8.2	Recursive Programming	216
		Recursion versus Iteration	219
		Direct versus Indirect Recursion	219
	8.3	Using Recursion	220
		Traversing a Maze	220
		The Towers of Hanoi	228
	8.4	Analyzing Recursive Algorithms	233
Chapter 9	Searching and Sorting		241
	9.1	Searching	242
		Static Methods	243
		Generic Methods	243

	Linear Search	244
	Binary Search	246
	Comparing Search Algorithms	248
9.2	Sorting	249
	Selection Sort	252
	Insertion Sort	254
	Bubble Sort	256
	Quick Sort	258
	Merge Sort	262
9.3	Radix Sort	265
Chapter 10	Trees	275
10.1	Trees	276
	Tree Classifications	277
10.2	Strategies for Implementing Trees	279
	Computational Strategy for Array Implementation of Trees	279
	Simulated Link Strategy for Array Implementation of Trees	279
	Analysis of Trees	281
10.3	Tree Traversals	282
	Preorder Traversal	282
	Inorder Traversal	283
	Postorder Traversal	283
	Level-Order Traversal	284
10.4	A Binary Tree ADT	285
10.5	Using Binary Trees: Expression Trees	289
10.6	A Back Pain Analyzer	301
10.7	Implementing Binary Trees with Links	305
	The <code>find</code> Method	310
	The <code>iteratorInOrder</code> Method	312
Chapter 11	Binary Search Trees	319
11.1	A Binary Search Tree	320

11.2	Implementing Binary Search Trees: With Links	322
	The addElement Operation	323
	The removeElement Operation	326
	The removeAllOccurrences Operation	329
	The removeMin Operation	330
	Implementing Binary Search Trees: With Arrays	332
11.3	Using Binary Search Trees: Implementing Ordered Lists	332
	Analysis of the BinarySearchTreeList Implementation	335
11.4	Balanced Binary Search Trees	336
	Right Rotation	337
	Left Rotation	338
	Rightleft Rotation	339
	Leftright Rotation	339
11.5	Implementing BSTs: AVL Trees	340
	Right Rotation in an AVL Tree	341
	Left Rotation in an AVL Tree	341
	Rightleft Rotation in an AVL Tree	341
	Leftright Rotation in an AVL Tree	343
11.6	Implementing BSTs: Red/Black Trees	343
	Insertion into a Red/Black Tree	344
	Element Removal from a Red/Black Tree	347
Chapter 12	Heaps and Priority Queues	357
12.1	A Heap	358
	The addElement Operation	360
	The removeMin Operation	361
	The findMin Operation	362
12.2	Using Heaps: Priority Queues	362
12.3	Implementing Heaps: With Links	366
	The addElement Operation	368
	The removeMin Operation	370
	The findMin Operation	373

12.4	Implementing Heaps: With Arrays	373
	The addElement Operation	375
	The removeMin Operation	376
	The findMin Operation	378
12.5	Using Heaps: Heap Sort	378
Chapter 13	Sets and Maps	385
13.1	Set and Map Collections	386
13.2	Sets and Maps in the Java API	386
13.3	Using Sets: Domain Blocker	389
13.4	Using Maps: Product Sales	392
13.5	Using Maps: User Management	396
13.6	Implementing Sets and Maps Using Trees	401
13.7	Implementing Sets and Maps Using Hashing	401
Chapter 14	Multi-Way Search Trees	409
14.1	Combining Tree Concepts	410
14.2	2-3 Trees	410
	Inserting Elements into a 2-3 Tree	411
	Removing Elements from a 2-3 Tree	413
14.3	2-4 Trees	416
14.4	B-Trees	418
	B*-Trees	419
	B+-Trees	419
	Analysis of B-Trees	420
14.5	Implementation Strategies for B-Trees	420
Chapter 15	Graphs	427
15.1	Undirected Graphs	428
15.2	Directed Graphs	429

15.3	Networks	431
15.4	Common Graph Algorithms	432
	Traversals	432
	Testing for Connectivity	436
	Minimum Spanning Trees	438
	Determining the Shortest Path	441
15.5	Strategies for Implementing Graphs	441
	Adjacency Lists	442
	Adjacency Matrices	442
15.6	Implementing Undirected Graphs with an Adjacency Matrix	443
	The <code>addEdge</code> Method	448
	The <code>addVertex</code> Method	448
	The <code>expandCapacity</code> Method	449
	Other Methods	450
Appendix A	UML	455
	The Unified Modeling Language (UML)	456
	UML Class Diagrams	456
	UML Relationships	458
Appendix B	Object-Oriented Design	463
B.1	Overview of Object-Oriented Design	464
B.2	Using Objects	464
	Abstraction	465
	Creating Objects	466
B.3	Class Libraries and Packages	468
	The <code>import</code> Declaration	468
B.4	State and Behavior	469
B.5	Classes	470
	Instance Data	473