

Fifth Edition

Java™ Foundations

Introduction to Program Design
& Data Structures



Fifth Edition

Java™ Foundations

Introduction to Program
Design & Data Structures

John Lewis | Peter DePasquale | Joseph Chase
Virginia Tech Apex Systems Radford University



Ljudmila Habrus/123RF

Senior Vice President Courseware Portfolio Management:
Director, Portfolio Management: Engineering,
Computer Science & Global Editions:
Portfolio Manager:
Portfolio Management Assistant:
Managing Content Producer:
Content Producer:
Rights and Permissions Manager:
Manufacturing Buyer, Higher Ed,
Lake Side Communications, Inc. (LSC):
Inventory Manager:
Product Marketing Manager:
Field Marketing Manager:
Marketing Assistant:
Cover Designer:
Cover Photo:
Printer/Binder:
Full-Service Project Management:

Marcia J. Horton
Julian Partridge
Matt Goldstein
Meghan Jacoby
Scott Disanno
Carole Snyder
Ben Ferrini
Maura Zaldivar-Garcia and Deidra Smith
Bruce Boundy
Yvonne Vannatta
Demetrius Hall
Jon Bryant
Integra Software Services, Inc.
Liudmila Habrus/123RF
LSC Communications, Inc.
Ramya Radhakrishnan, Integra Software Services Pvt. Ltd.,

Copyright © 2020 by Pearson Education, Inc. or its affiliates. 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 otherwise. For information regarding permissions, request forms and the appropriate contacts within the Pearson Education Global Rights & Permissions department, please visit www.pearsoned.com/permissions/.

Acknowledgements of third-party content appear on page xxix, which constitute an extension of this copyright page.

PEARSON, and ALWAYS LEARNING are exclusive trademarks in the U.S. and/or other countries owned by Pearson Education, Inc. or its affiliates.

Unless otherwise indicated herein, any third-party trademarks that may appear in this work are the property of their respective owners and any references to third-party trademarks, logos or other trade dress are for demonstrative or descriptive purposes only. Such references are not intended to imply any sponsorship, endorsement, authorization, or promotion of Pearson's products by the owners of such marks, or any relationship between the owner and Pearson Education, Inc. or its affiliates, authors, licensees or distributors.

The programs and applications presented in this book have been included for their instructional value. They have been tested with care, but are not guaranteed for any particular purpose. The publisher does not offer any warranties or representations, nor does it accept any liabilities with respect to the programs or applications.

Library of Congress Cataloging-in-Publication Data on File

Names: Lewis, John, author. | DePasquale, Peter J. (Peter Joseph), author. | Chase, Joseph, author.

Title: Java foundations : introduction to program design & data structures / John Lewis, Virginia Tech, Peter DePasquale, Sailthru, Inc., Joseph Chase, Radford University.

Description: Fifth edition. | Boston : Pearson, [2020] | Includes index.

Identifiers: LCCN 2018046357 | ISBN 9780135205976 | ISBN 0135205972

Subjects: LCSH: Java (Computer program language)

Classification: LCC QA76.73.J38 L48845 2020 | DDC 005.13/3—dc23

LC record available at <https://lcn.loc.gov/2018046357>

1 19



ISBN 10: 0-13-520597-2
ISBN 13: 978-0-13-520597-6

To my wife, Sharon, for everything.

– John

*To my wonderful wife Susan, and our children, Grace, Anthony, Adam, Lily, EJ, and Peter IV.
Your continued love and support keep me going as always.*

– Pete

To my loving wife, Melissa, for her support and encouragement.

– Joe



Preface

Welcome to *Java Foundations*. This book is designed to serve as the primary resource for a two- or three-term introductory course sequence, ranging from the most basic programming concepts to the design and implementation of complex data structures. This unified approach makes the important introductory sequence more cohesive and accessible for students.

We've borrowed the best elements from the industry-leading text *Java Software Solutions* for the introductory material, reworked to complement the design and vision of the overall text. For example, instead of having graphics sections spread throughout many chapters, the coverage of graphical user interfaces is accomplished in a well-organized chapter of its own.

In the later chapters, the exploration of collections and data structures is modeled after the coverage in *Java Software Structures*, but has been reworked to flow cleanly from the introductory material. The result is a comprehensive, cohesive, and seamless exploration of programming concepts.

New in the Fifth Edition

We appreciate the feedback we've received about this book and are pleased that it continues to serve so well as an introductory text. The changes made in this edition build on the strong pedagogy established by previous editions while updating crucial areas.

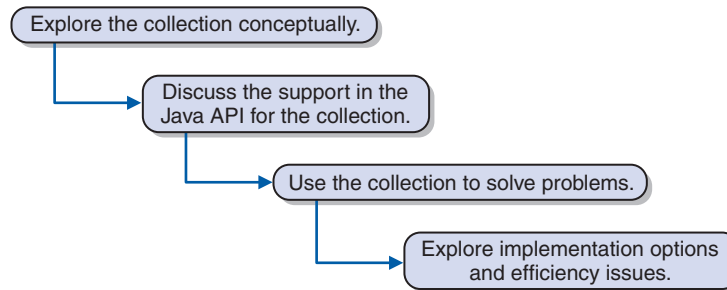
The biggest change in this edition is the overhaul of the graphical content to fully embrace the JavaFX platform, which has replaced Swing as the supported technology for graphics and Graphical User Interfaces (GUIs) in Java. The previous edition focused on Swing and had an introduction to JavaFX. The time has come to switch over completely to the new approach, which simplifies GUI development and provides better opportunities to discuss object-oriented programming.

The changes in this edition include:

- A brand new Chapter 6 on developing GUIs using JavaFX.
- A new Appendix F that discusses the rendering of graphics using JavaFX.
- A new Appendix G that explores the JavaFX Scene Builder, a drag-and-drop application for developing graphical front ends.

- Updated examples and discussions throughout the text.
- Updated end-of-chapter Programming Projects in several chapters.

In previous editions, we had established the following flow when discussing collections:



Your feedback has indicated that this approach is working well and we have continued and reinforced its use. It 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.

Chapter Breakdown

Chapter 1 (Introduction) introduces the Java programming language and the basics of program development. It contains an introduction to object-oriented development, including an overview of concepts and terminology. This chapter contains broad introductory material that can be covered while students become familiar with their development environment.

Chapter 2 (Data and Expressions) explores some of the basic types of data used in a Java program and the use of expressions to perform calculations. It discusses the conversion of data from one type to another, and how to read input interactively from the user with the help of the Scanner class.

Chapter 3 (Using Classes and Objects) explores the use of predefined classes and the objects that can be created from them. Classes and objects are used to manipulate character strings, produce random numbers, perform complex calculations, and format output. Packages, enumerated types, and wrapper classes are also discussed.

Chapter 4 (Conditionals and Loops) covers the use of Boolean expressions to make decisions. All related statements for conditionals and loops are discussed,

including the enhanced version of the `for` loop. The `Scanner` class is revisited for iterative input parsing and reading text files.

Chapter 5 (Writing Classes) explores the basic issues related to writing classes and methods. Topics include instance data, visibility, scope, method parameters, and return types. Constructors, method design, static data, and method overloading are covered as well. Testing and debugging are now covered in this chapter as well.

Chapter 6 (Graphical User Interfaces) is an exploration of GUI processing using the JavaFX platform, focusing on controls, events, and event handlers. Several types of controls are discussed using numerous GUI examples. Mouse events, keyboard events, and layout panes are also explored.

Chapter 7 (Arrays) contains extensive coverage of arrays and array processing. Topics include bounds checking, initializer lists, command-line arguments, variable-length parameter lists, and multidimensional arrays.

Chapter 8 (Inheritance) covers class derivations and associated concepts such as class hierarchies, overriding, and visibility. Strong emphasis is put on the proper use of inheritance and its role in software design.

Chapter 9 (Polymorphism) explores the concept of binding and how it relates to polymorphism. Then we examine how polymorphic references can be accomplished using either inheritance or interfaces. Design issues related to polymorphism are examined as well.

Chapter 10 (Exceptions) covers exception handling and the effects of uncaught exceptions. The `try-catch` statement is examined, as well as a discussion of exception propagation. The chapter also explores the use of exceptions when dealing with input and output, and examines an example that writes a text file.

Chapter 11 (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 12 (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 13 (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 12) using an underlying linked data structure.

Chapter 14 (Queues) explores the concept and implementation of a first-in, first-out queue. The Java API `Queue` interface is discussed, as are linked and circular array implementations with `Queue` in code font.

Chapter 15 (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 16 (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 17 (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 18 (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. An application uses animation to demonstrate the efficiency of sorting algorithms. The `comparator` interface is examined and demonstrated as well.

Chapter 19 (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 20 (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 21 (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 22 (Sets and Maps) explores these two types of collections and their importance to the Java Collections API.

Chapter 23 (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 24 (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.

Chapter 25 (Databases) explores the concept of databases and their management, and discusses the basics of SQL queries. It then explores the techniques for

establishing a connection between a Java program and a database, and the API used to interact with it.

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.pearson.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.pearsonhighered.com/irc or contact your local Pearson Education sales representative.

BREAK THROUGH
To improving results

MyLabProgramming

Through the power of practice and immediate personalized feedback, MyLab Programming™ helps improve your students' performance.

PROGRAMMING PRACTICE

With MyLab Programming, your students will gain first-hand programming experience in an interactive online environment.

IMMEDIATE, PERSONALIZED FEEDBACK

MyLab Programming automatically detects errors in the logic and syntax of their code submission and offers targeted hints that enables students to figure out what went wrong and why.

GRADUATED COMPLEXITY

MyLab Programming breaks down programming concepts into short, understandable sequences of exercises. Within each sequence the level and sophistication of the exercises increase gradually but steadily.

DYNAMIC ROSTER

Students' submissions are stored in a roster that indicates whether the submission is correct, how many attempts were made, and the actual code submissions from each attempt.

PEARSON eTEXT

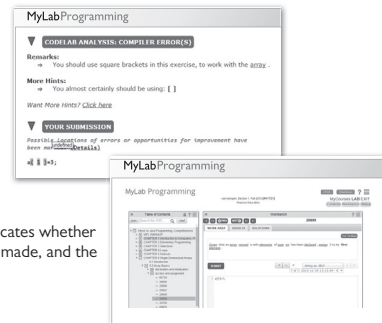
The Pearson eText gives students access to their textbook anytime, anywhere.

STEP-BY-STEP VIDEONOTE TUTORIALS

These step-by-step video tutorials enhance the programming concepts presented in select Pearson textbooks.

For more information and titles available with **MyLab Programming**, please visit www.pearson.com/mylab/programming

Copyright © 2020 Pearson Education, Inc. or its affiliate(s). All rights reserved. HELO88173 • 11/15



ALWAYS LEARNING



Contents

| | |
|---|--------------|
| Preface | vii |
| Credits | xxix |
| VideoNotes | xxxix |
| Chapter 1 Introduction | 1 |
| 1.1 The Java Programming Language | 2 |
| A Java Program | 3 |
| Comments | 5 |
| Identifiers and Reserved Words | 7 |
| White Space | 9 |
| 1.2 Program Development | 11 |
| Programming Language Levels | 11 |
| Editors, Compilers, and Interpreters | 13 |
| Development Environments | 15 |
| Syntax and Semantics | 16 |
| Errors | 17 |
| 1.3 Problem Solving | 18 |
| 1.4 Software Development Activities | 20 |
| 1.5 Object-Oriented Programming | 21 |
| Object-Oriented Software Principles | 22 |
| Chapter 2 Data and Expressions | 33 |
| 2.1 Character Strings | 34 |
| The <code>print</code> and <code>println</code> Methods | 34 |
| String Concatenation | 36 |
| Escape Sequences | 40 |
| 2.2 Variables and Assignment | 41 |
| Variables | 41 |
| The Assignment Statement | 44 |
| Constants | 46 |

| | | |
|--|--------------------------------------|------------|
| 2.3 | Primitive Data Types | 47 |
| | Integers and Floating Points | 47 |
| | Characters | 48 |
| | Booleans | 50 |
| 2.4 | Expressions | 51 |
| | Arithmetic Operators | 51 |
| | Operator Precedence | 52 |
| | Increment and Decrement Operators | 56 |
| | Assignment Operators | 57 |
| 2.5 | Data Conversion | 58 |
| | Conversion Techniques | 60 |
| 2.6 | Reading Input Data | 61 |
| | The <code>Scanner</code> Class | 61 |
| Chapter 3 Using Classes and Objects | | 75 |
| 3.1 | Creating Objects | 76 |
| | Aliases | 78 |
| 3.2 | The <code>String</code> Class | 80 |
| 3.3 | Packages | 83 |
| | The <code>import</code> Declaration | 84 |
| 3.4 | The <code>Random</code> Class | 86 |
| 3.5 | The <code>Math</code> Class | 89 |
| 3.6 | Formatting Output | 92 |
| | The <code>NumberFormat</code> Class | 92 |
| | The <code>DecimalFormat</code> Class | 94 |
| | The <code>printf</code> Method | 96 |
| 3.7 | Enumerated Types | 97 |
| 3.8 | Wrapper Classes | 100 |
| | Autoboxing | 102 |
| Chapter 4 Conditionals and Loops | | 111 |
| 4.1 | Boolean Expressions | 112 |
| | Equality and Relational Operators | 113 |
| | Logical Operators | 114 |

| | | |
|----------------------------------|--|------------|
| 4.2 | The <code>if</code> Statement | 116 |
| | The <code>if-else</code> Statement | 119 |
| | Using Block Statements | 121 |
| | The Conditional Operator | 124 |
| | Nested <code>if</code> Statements | 125 |
| 4.3 | Comparing Data | 127 |
| | Comparing Floats | 127 |
| | Comparing Characters | 127 |
| | Comparing Objects | 128 |
| 4.4 | The <code>switch</code> Statement | 130 |
| 4.5 | The <code>while</code> Statement | 134 |
| | Infinite Loops | 140 |
| | Nested Loops | 141 |
| | Other Loop Controls | 144 |
| 4.6 | Iterators | 145 |
| | Reading Text Files | 146 |
| 4.7 | The <code>do</code> Statement | 148 |
| 4.8 | The <code>for</code> Statement | 151 |
| | Iterators and <code>for</code> Loops | 156 |
| | Comparing Loops | 157 |
| Chapter 5 Writing Classes | | 169 |
| 5.1 | Classes and Objects Revisited | 170 |
| | Identifying Classes and Objects | 171 |
| | Assigning Responsibilities | 173 |
| 5.2 | Anatomy of a Class | 173 |
| | Instance Data | 178 |
| | UML Class Diagrams | 179 |
| 5.3 | Encapsulation | 181 |
| | Visibility Modifiers | 182 |
| | Accessors and Mutators | 183 |
| 5.4 | Anatomy of a Method | 188 |
| | The <code>return</code> Statement | 194 |
| | Parameters | 196 |
| | Local Data | 197 |
| | Constructors Revisited | 198 |

| | | |
|--|---|------------|
| 5.5 | Static Class Members | 199 |
| | Static Variables | 199 |
| | Static Methods | 200 |
| 5.6 | Class Relationships | 203 |
| | Dependency | 203 |
| | Dependencies among Objects of the Same Class | 204 |
| | Aggregation | 206 |
| | The <code>this</code> Reference | 211 |
| 5.7 | Method Design | 212 |
| | Method Decomposition | 213 |
| | Method Parameters Revisited | 218 |
| 5.8 | Method Overloading | 223 |
| 5.9 | Testing | 224 |
| | Reviews | 225 |
| | Defect Testing | 226 |
| | Unit Testing | 227 |
| | Integration Testing | 228 |
| | System Testing | 228 |
| | Test-Driven Development | 228 |
| 5.10 | Debugging | 229 |
| | Simple Debugging with <code>print</code> Statements | 230 |
| | Debugging Concepts | 230 |
| Chapter 6 Graphical User Interfaces | | 245 |
| 6.1 | Introduction to JavaFX | 246 |
| | GUI Elements | 249 |
| | Alternate Ways to Specify Event Handlers | 252 |
| | Determining Event Sources | 253 |
| 6.2 | Other GUI Controls | 256 |
| | Text Fields | 256 |
| | Check Boxes | 259 |
| | Radio Buttons | 263 |
| | Color and Date Pickers | 267 |
| 6.3 | Mouse and Key Events | 270 |
| | Mouse Events | 271 |
| | Key Events | 276 |

| | | |
|------------------------------|---|------------|
| 6.4 | Dialog Boxes | 279 |
| | File Choosers | 283 |
| 6.5 | JavaFX Properties | 286 |
| | Change Listeners | 289 |
| | Sliders | 292 |
| | Spinners | 295 |
| 6.6 | Tool Tips and Disabling Controls | 299 |
| Chapter 7 Arrays | | 313 |
| 7.1 | Array Elements | 314 |
| 7.2 | Declaring and Using Arrays | 315 |
| | Bounds Checking | 318 |
| | Alternative Array Syntax | 323 |
| | Initializer Lists | 324 |
| | Arrays as Parameters | 325 |
| 7.3 | Arrays of Objects | 325 |
| 7.4 | Command-Line Arguments | 335 |
| 7.5 | Variable-Length Parameter Lists | 337 |
| 7.6 | Two-Dimensional Arrays | 341 |
| | Multidimensional Arrays | 344 |
| 7.7 | Arrays and GUIs | 346 |
| | An Array of Color Objects | 346 |
| | Choice Boxes | 349 |
| Chapter 8 Inheritance | | 361 |
| 8.1 | Creating Subclasses | 362 |
| | The <code>protected</code> Modifier | 367 |
| | The <code>super</code> Reference | 368 |
| | Multiple Inheritance | 372 |
| 8.2 | Overriding Methods | 373 |
| | Shadowing Variables | 376 |
| 8.3 | Class Hierarchies | 376 |
| | The <code>Object</code> Class | 377 |
| | Abstract Classes | 379 |

| | | |
|--|--------------------------------------|------------|
| 8.4 | Visibility | 381 |
| 8.5 | Designing for Inheritance | 383 |
| | Restricting Inheritance | 384 |
| 8.6 | Inheritance in JavaFX | 385 |
| Chapter 9 Polymorphism | | 395 |
| 9.1 | Dynamic Binding | 396 |
| 9.2 | Polymorphism via Inheritance | 397 |
| 9.3 | Interfaces | 409 |
| | Interface Hierarchies | 414 |
| | The Comparable Interface | 415 |
| | The Iterator Interface | 415 |
| 9.4 | Polymorphism via Interfaces | 416 |
| Chapter 10 Exceptions | | 425 |
| 10.1 | Exception Handling | 426 |
| 10.2 | Uncaught Exceptions | 427 |
| 10.3 | The try-catch Statement | 428 |
| | The finally Clause | 431 |
| 10.4 | Exception Propagation | 432 |
| 10.5 | The Exception Class Hierarchy | 435 |
| | Checked and Unchecked Exceptions | 439 |
| 10.6 | I/O Exceptions | 439 |
| Chapter 11 Analysis of Algorithms | | 449 |
| 11.1 | Algorithm Efficiency | 450 |
| 11.2 | Growth Functions and Big-Oh Notation | 451 |
| 11.3 | Comparing Growth Functions | 453 |
| 11.4 | Determining Time Complexity | 455 |
| | Analyzing Loop Execution | 455 |
| | Nested Loops | 456 |
| | Method Calls | 457 |

| | |
|--|------------|
| Chapter 12 Introduction to Collections—Stacks | 463 |
| 12.1 Collections | 464 |
| Abstract Data Types | 465 |
| The Java Collections API | 467 |
| 12.2 A Stack Collection | 467 |
| 12.3 Crucial OO Concepts | 469 |
| Inheritance and Polymorphism | 470 |
| Generics | 471 |
| 12.4 Using Stacks: Evaluating Postfix Expressions | 472 |
| Javadoc | 480 |
| 12.5 Exceptions | 481 |
| 12.6 A Stack ADT | 482 |
| 12.7 Implementing a Stack: With Arrays | 485 |
| Managing Capacity | 486 |
| 12.8 The <code>ArrayStack</code> Class | 487 |
| The Constructors | 488 |
| The <code>push</code> Operation | 490 |
| The <code>pop</code> Operation | 492 |
| The <code>peek</code> Operation | 493 |
| Other Operations | 493 |
| The <code>EmptyCollectionException</code> Class | 494 |
| Other Implementations | 495 |
| Chapter 13 Linked Structures—Stacks | 503 |
| 13.1 References as Links | 504 |
| 13.2 Managing Linked Lists | 506 |
| Accessing Elements | 506 |
| Inserting Nodes | 507 |
| Deleting Nodes | 508 |
| 13.3 Elements without Links | 509 |
| Doubly Linked Lists | 509 |
| 13.4 Stacks in the Java API | 510 |
| 13.5 Using Stacks: Traversing a Maze | 511 |

| | | |
|--------------------------|--|------------|
| 13.6 | Implementing a Stack: With Links | 520 |
| | The <code>LinkedList</code> Class | 520 |
| | The <code>push</code> Operation | 524 |
| | The <code>pop</code> Operation | 526 |
| | Other Operations | 527 |
| Chapter 14 Queues | | 533 |
| 14.1 | A Conceptual Queue | 534 |
| 14.2 | Queues in the Java API | 535 |
| 14.3 | Using Queues: Code Keys | 536 |
| 14.4 | Using Queues: Ticket Counter Simulation | 540 |
| 14.5 | A Queue ADT | 545 |
| 14.6 | A Linked Implementation of a Queue | 546 |
| | The <code>enqueue</code> Operation | 548 |
| | The <code>dequeue</code> Operation | 550 |
| | Other Operations | 551 |
| 14.7 | Implementing Queues: With Arrays | 552 |
| | The <code>enqueue</code> Operation | 556 |
| | The <code>dequeue</code> Operation | 558 |
| | Other Operations | 559 |
| 14.8 | Double-Ended Queues (Deque) | 559 |
| Chapter 15 Lists | | 565 |
| 15.1 | A List Collection | 566 |
| 15.2 | Lists in the Java Collections API | 568 |
| 15.3 | Using Unordered Lists: Program of Study | 569 |
| 15.4 | Using Indexed Lists: Josephus | 579 |
| 15.5 | A List ADT | 581 |
| | Adding Elements to a List | 582 |
| 15.6 | Implementing Lists with Arrays | 587 |
| | The <code>remove</code> Operation | 589 |
| | The <code>contains</code> Operation | 591 |
| | The <code>add</code> Operation for an Ordered List | 592 |

| | |
|---|------------|
| Operations Particular to Unordered Lists | 593 |
| The <code>addAfter</code> Operation for an Unordered List | 593 |
| 15.7 Implementing Lists with Links | 594 |
| The <code>remove</code> Operation | 595 |
| 15.8 Lists in JavaFX | 597 |
| Observable List | 597 |
| Sorted List | 597 |
| Chapter 16 Iterators | 605 |
| 16.1 What's an Iterator? | 606 |
| Other Iterator Issues | 608 |
| 16.2 Using Iterators: Program of Study Revisited | 609 |
| Printing Certain Courses | 613 |
| Removing Courses | 614 |
| 16.3 Implementing Iterators: With Arrays | 615 |
| 16.4 Implementing Iterators: With Links | 617 |
| Chapter 17 Recursion | 623 |
| 17.1 Recursive Thinking | 624 |
| Infinite Recursion | 624 |
| Recursion in Math | 625 |
| 17.2 Recursive Programming | 626 |
| Recursion versus Iteration | 629 |
| Direct versus Indirect Recursion | 629 |
| 17.3 Using Recursion | 630 |
| Traversing a Maze | 630 |
| The Towers of Hanoi | 638 |
| 17.4 Analyzing Recursive Algorithms | 643 |
| Chapter 18 Searching and Sorting | 651 |
| 18.1 Searching | 652 |
| Static Methods | 653 |
| Generic Methods | 653 |
| Linear Search | 654 |

| | | |
|---------------------------------------|--|------------|
| | Binary Search | 656 |
| | Comparing Search Algorithms | 658 |
| 18.2 | Sorting | 659 |
| | Selection Sort | 662 |
| | Insertion Sort | 664 |
| | Bubble Sort | 666 |
| | Quick Sort | 668 |
| | Merge Sort | 672 |
| 18.3 | Radix Sort | 675 |
| 18.4 | A Different Way to Sort—Comparator | 679 |
| Chapter 19 Trees | | 693 |
| 19.1 | Trees | 694 |
| | Tree Classifications | 695 |
| 19.2 | Strategies for Implementing Trees | 697 |
| | Computational Strategy for Array Implementation of Trees | 697 |
| | Simulated Link Strategy for Array Implementation of Trees | 697 |
| | Analysis of Trees | 699 |
| 19.3 | Tree Traversals | 700 |
| | Preorder Traversal | 700 |
| | Inorder Traversal | 701 |
| | Postorder Traversal | 701 |
| | Level-Order Traversal | 702 |
| 19.4 | A Binary Tree ADT | 703 |
| 19.5 | Using Binary Trees: Expression Trees | 707 |
| 19.6 | A Back Pain Analyzer | 719 |
| 19.7 | Implementing Binary Trees with Links | 724 |
| | The <code>find</code> Method | 728 |
| | The <code>iteratorInOrder</code> Method | 730 |
| Chapter 20 Binary Search Trees | | 737 |
| 20.1 | Binary Search Trees | 738 |
| | Adding an Element to a Binary Search Tree | 739 |

| | |
|---|----------------|
| Removing an Element from a Binary Search Tree | 741 |
| 20.2 Implementing a Binary Search Tree | 743 |
| 20.3 Implementing Binary Search Trees: With Links | 745 |
| The <code>addElement</code> Operation | 746 |
| The <code>removeElement</code> Operation | 748 |
| The <code>removeAllOccurrences</code> Operation | 752 |
| The <code>removeMin</code> Operation | 753 |
| Implementing Binary Search Trees: With Arrays | 755 |
| 20.4 Using Binary Search Trees: Implementing Ordered Lists | 755 |
| Analysis of the <code>BinarySearchTreeList</code> Implementation | 758 |
| 20.5 Balanced Binary Search Trees | 759 |
| Right Rotation | 760 |
| Left Rotation | 761 |
| Rightleft Rotation | 762 |
| Leftright Rotation | 762 |
| 20.6 Implementing Binary Search Trees: AVL Trees | 762 |
| Right Rotation in an AVL Tree | 763 |
| Left Rotation in an AVL Tree | 764 |
| Rightleft Rotation in an AVL Tree | 764 |
| Leftright Rotation in an AVL Tree | 765 |
| 20.7 Implementing Binary Search Trees: Red/Black Trees | 766 |
| Insertion into a Red/Black Tree | 766 |
| Element Removal from a Red/Black Tree | 770 |
| Chapter 21 Heaps and Priority Queues | 779 |
| 21.1 A Heap | 780 |
| The <code>addElement</code> Operation | 782 |
| The <code>removeMin</code> Operation | 783 |
| The <code>findMin</code> Operation | 784 |
| 21.2 Using Heaps: Priority Queues | 784 |

| | | |
|--|---|------------|
| 21.3 | Implementing Heaps: With Links | 788 |
| | The <code>addElement</code> Operation | 788 |
| | The <code>removeMin</code> Operation | 792 |
| | The <code>findMin</code> Operation | 795 |
| 21.4 | Implementing Heaps: With Arrays | 795 |
| | The <code>addElement</code> Operation | 797 |
| | The <code>removeMin</code> Operation | 798 |
| | The <code>findMin</code> Operation | 800 |
| 21.5 | Using Heaps: Heap Sort | 800 |
| Chapter 22 Sets and Maps | | 807 |
| 22.1 | Set and Map Collections | 808 |
| 22.2 | Sets and Maps in the Java API | 808 |
| 22.3 | Using Sets: Domain Blocker | 811 |
| 22.4 | Using Maps: Product Sales | 814 |
| 22.5 | Using Maps: User Management | 818 |
| 22.6 | Implementing Sets and Maps Using Trees | 823 |
| 22.7 | Implementing Sets and Maps Using Hashing | 823 |
| Chapter 23 Multi-way Search Trees | | 831 |
| 23.1 | Combining Tree Concepts | 832 |
| 23.2 | 2-3 Trees | 832 |
| | Inserting Elements into a 2-3 Tree | 833 |
| | Removing Elements from a 2-3 Tree | 835 |
| 23.3 | 2-4 Trees | 838 |
| 23.4 | B-Trees | 840 |
| | B*-Trees | 841 |
| | B ⁺ -Trees | 841 |
| | Analysis of B-Trees | 842 |
| 23.5 | Implementation Strategies for B-Trees | 842 |

| | |
|---|------------|
| Chapter 24 Graphs | 849 |
| 24.1 Undirected Graphs | 850 |
| 24.2 Directed Graphs | 851 |
| 24.3 Networks | 853 |
| 24.4 Common Graph Algorithms | 854 |
| Traversals | 854 |
| Testing for Connectivity | 858 |
| Minimum Spanning Trees | 860 |
| Determining the Shortest Path | 863 |
| 24.5 Strategies for Implementing Graphs | 863 |
| Adjacency Lists | 864 |
| Adjacency Matrices | 864 |
| 24.6 Implementing Undirected Graphs with an Adjacency Matrix | 865 |
| The addEdge Method | 870 |
| The addVertex Method | 870 |
| The expandCapacity Method | 871 |
| Other Methods | 872 |
| Chapter 25 Databases | 879 |
| 25.1 Introduction to Databases | 880 |
| 25.2 Establishing a Connection to a Database | 882 |
| Obtaining a Database Driver | 882 |
| 25.3 Creating and Altering Database Tables | 885 |
| Create Table | 885 |
| Alter Table | 886 |
| Drop Column | 887 |
| 25.4 Querying the Database | 887 |
| Show Columns | 888 |
| 25.5 Inserting, Viewing, and Updating Data | 890 |
| Insert | 891 |

| | |
|---|------------|
| SELECT ... FROM | 891 |
| Update | 896 |
| 25.6 Deleting Data and Database Tables | 897 |
| Deleting Data | 897 |
| Deleting Database Tables | 898 |
| | |
| Appendix A Glossary | 903 |
| | |
| Appendix B Number Systems | 937 |
| Place Value | 938 |
| Bases Higher Than 10 | 939 |
| Conversions | 940 |
| Shortcut Conversions | 943 |
| | |
| Appendix C The Unicode Character Set | 949 |
| | |
| Appendix D Java Operators | 953 |
| Java Bitwise Operators | 955 |
| | |
| Appendix E Java Modifiers | 959 |
| Java Visibility Modifiers | 960 |
| A Visibility Example | 960 |
| Other Java Modifiers | 961 |
| | |
| Appendix F JavaFX Graphics | 963 |
| Coordinate Systems | 964 |
| Representing Colors | 964 |
| Basic Shapes | 965 |
| Arcs | 970 |

| | |
|--|-------------|
| Images | 974 |
| Fonts | 976 |
| Graphic Transformations | 979 |
| Translation | 979 |
| Scaling | 980 |
| Rotation | 981 |
| Shearing | 982 |
| Polygons and Polylines | 982 |
| | |
| Appendix G JavaFX Scene Builder | 987 |
| Hello Moon | 988 |
| Handling Events in JavaFX Scene Builder | 993 |
| | |
| Appendix H Regular Expressions | 997 |
| | |
| Appendix I Hashing | 999 |
| I.1 A Hashing | 1000 |
| I.2 Hashing Functions | 1001 |
| The Division Method | 1002 |
| The Folding Method | 1002 |
| The Mid-Square Method | 1003 |
| The Radix Transformation Method | 1003 |
| The Digit Analysis Method | 1003 |
| The Length-Dependent Method | 1004 |
| Hashing Functions in the Java Language | 1004 |
| I.3 Resolving Collisions | 1004 |
| Chaining | 1005 |
| Open Addressing | 1006 |
| I.4 Deleting Elements from a Hash Table | 1009 |
| Deleting from a Chained Implementation | 1009 |
| Deleting from an Open Addressing Implementation | 1010 |

| | |
|---|-------------|
| I.5 Hash Tables in the Java Collections API | 1011 |
| The <code>Hashtable</code> Class | 1011 |
| The <code>HashSet</code> Class | 1013 |
| The <code>HashMap</code> Class | 1013 |
| The <code>IdentityHashMap</code> Class | 1014 |
| I.6 The <code>weakHashMap</code> Class | 1015 |
| <code>LinkedHashSet</code> and <code>LinkedHashMap</code> | 1016 |
| | |
| Appendix J Java Syntax | 1023 |
| | |
| Index | 1037 |

Credits

Cover: Liudmila Habrus/123RF

Chapter 1 page 2: Reference: Java is a relatively new programming language compared to many others. It was developed in the early 1990s by James Gosling at Sun Microsystems. Java was released to the public in 1995 and has gained tremendous popularity since “The History of Java Technology” Oracle Corporation. 1995. Accessed at <http://www.oracle.com/technetwork/java/javase/overview/javahistory-index-198355.html>

Chapter 1 page 15: Excerpt: A research group at Auburn University has developed jGRASP, a free Java IDE that is included on the CD that accompanies this book. It can also be downloaded from www.jgrasp.org. “jGRASP” is developed by the Department of Computer Science and Software Engineering in the Samuel Ginn College of Engineering at Auburn University.

Chapter 1 page 20: Reference: The programming language Simula, developed in the 1960s, had many characteristics that define the modern object-oriented approach to software development. Nygaard, Kristen, Myhrhaug, Bjørn, and Dahl, Ole-Johan. “Simula. Common Base Language.” Norwegian Computing Center. 1970. Accessed at <http://www.nr.no/>

Chapter 4: Excerpt: The Twelve Days of Christmas. “Twelve Days of Christmas.” Mirth Without Mischief. 1780.

Chapter 11: Text: Another way of looking at the effect of algorithm complexity was proposed by Aho, Hopcroft, and Ullman. Aho, A.V., J.E. Hopcroft, and J.D. Ullman. “The Design and Analysis of Computer Algorithms.” Addison-Wesley. 1974.

Chapter 20: Text: Adel’son-Vel’skii and Landis developed a method called AVL trees that is a variation on this theme. For each node in the tree, we will keep track of the height of the left and right subtrees. Adelson-Velskii, Georgii and Evengii Landis. “An Algorithm for the Organization of Information.” 1962.

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.



VideoNote

LOCATION OF VIDEONOTES IN THE TEXT

| | |
|------------|---|
| Chapter 1 | Overview of program elements, page 4 Comparison of Java IDEs, page 16 Examples of various error types, page 18 |
| Chapter 2 | Example using strings and escape sequences, page 40 Review of primitive data and expressions, page 52 Example using the <code>Scanner</code> class, page 63 |
| Chapter 3 | Creating objects, page 77 Example using the <code>Random</code> and <code>Math</code> classes, page 89 |
| Chapter 4 | Examples using conditionals, page 123 Examples using <code>while</code> loops, page 138 Examples using <code>for</code> loops, page 155 |
| Chapter 5 | Dissecting the <code>Die</code> class, page 178 Discussion of the <code>Account</code> class, page 194 |
| Chapter 7 | Overview of arrays, page 315 Discussion of the <code>LetterCount</code> example, page 323 |
| Chapter 8 | Overview of inheritance, page 363 Example using a class hierarchy, page 378 |
| Chapter 9 | Exploring the <code>Firm</code> program, page 404 |
| Chapter 10 | Proper exception handling, page 432 |
| Chapter 12 | An overview of the <code>ArrayStack</code> implementation, page 488 |
| Chapter 13 | Using a stack to solve a maze, page 512 |
| Chapter 14 | An array-based queue implementation, page 552 |
| Chapter 15 | List categories, page 566 |
| Chapter 17 | Analyzing recursive algorithms, page 644 |
| Chapter 18 | Demonstration of a binary search, page 657 |
| Chapter 19 | Demonstration of the four basic tree traversals, page 703 |
| Chapter 20 | Demonstration of the four basic tree rotations, page 763 |
| Chapter 21 | Demonstration of a heap sort on an array, page 801 |
| Chapter 22 | A comparison of sets and maps, page 808 |
| Chapter 23 | Inserting elements into, and removing elements from, a 2-3 tree, page 835 |
| Chapter 24 | Illustration of depth-first and breadth-first traversals of a graph, page 855 |

