MyLab Programming

Through the power of practice and immediate personalized feedback, MyLab™ Programming helps students master programming fundamentals and build computational thinking skills.

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 © 2018 Pearson Education, Inc. or its affiliate(s). All rights reserved. HEL088173 • 11/15

ALWAYS LEARNING
The newly revised fifth edition of our *Building Java Programs* textbook is designed for use in a two-course introduction to computer science. We have class-tested it with thousands of undergraduates, most of whom were not computer science majors, in our CS1-CS2 sequence at the University of Washington. These courses are experiencing record enrollments, and other schools that have adopted our textbook report that students are succeeding with our approach.

 Introductory computer science courses are often seen as “killer” courses with high failure rates. But as Douglas Adams says in *The Hitchhiker’s Guide to the Galaxy*, “Don’t panic.” Students can master this material if they can learn it gradually. Our textbook uses a layered approach to introduce new syntax and concepts over multiple chapters.

Our textbook uses an “objects later” approach where programming fundamentals and procedural decomposition are taught before diving into object-oriented programming. We have championed this approach, which we sometimes call “back to basics,” and have seen through years of experience that a broad range of scientists, engineers, and others can learn how to program in a procedural manner. Once we have built a solid foundation of procedural techniques, we turn to object-oriented programming. By the end of the course, students will have learned about both styles of programming.

The Java language is always evolving, and we have made it a point of focus in recent editions on newer features that have been added in Java 8 through 10. In the fourth edition we added a new Chapter 19 on Java’s functional programming features introduced in Java 8. In this edition we integrate the JShell tool introduced in Java 9.

**New to This Edition**

The following are the major changes for our fifth edition:

- **JShell integration.** Java 9 introduced JShell, a utility with an interactive read-eval-print loop (REPL) that makes it easy to type Java expressions and immediately see their results. We find JShell to be a valuable learning tool that allows students to explore Java concepts without the overhead of creating a complete program. We introduce JShell in Chapter 2 and integrate JShell examples in each chapter throughout the text.

- **Improved Chapter 2 loop coverage.** We have added new sections and figures in Chapter 2 to help students understand for loops and create tables to find patterns in nested loops. This new content is based on our interactions with our own students as they solve programming problems with loops early in our courses.
Preface

- **Revamped case studies, examples, and other content.** We have rewritten or revised sections of various chapters based on student and instructor feedback. We have also rewritten the Chapter 10 (ArrayLists) case study with a new program focusing on elections and ranked choice voting.

- **Updated collection syntax and idioms.** Recent releases of Java have introduced new syntax and features related to collections, such as the `<>` “diamond operator;” collection interfaces such as Lists, Sets, and Maps; and new collection methods. We have updated our collection Chapters 10 and 11 to discuss these new features, and we use the diamond operator syntax with collections in the rest of the text.

- **Expanded self-checks and programming exercises.** With each new edition we add new programming exercises to the end of each chapter. There are roughly fifty total problems and exercises per chapter, all of which have been class-tested with real students and have solutions provided for instructors on our web site.

- **New programming projects.** Some chapters have received new programming projects, such as the Chapter 10 ranked choice ballot project.

Features from Prior Editions

The following features have been retained from previous editions:

- **Focus on problem solving.** Many textbooks focus on language details when they introduce new constructs. We focus instead on problem solving. What new problems can be solved with each construct? What pitfalls are novices likely to encounter along the way? What are the most common ways to use a new construct?

- **Emphasis on algorithmic thinking.** Our procedural approach allows us to emphasize algorithmic problem solving: breaking a large problem into smaller problems, using pseudocode to refine an algorithm, and grappling with the challenge of expressing a large program algorithmically.

- **Layered approach.** Programming in Java involves many concepts that are difficult to learn all at once. Teaching Java to a novice is like trying to build a house of cards. Each new card has to be placed carefully. If the process is rushed and you try to place too many cards at once, the entire structure collapses. We teach new concepts gradually, layer by layer, allowing students to expand their understanding at a manageable pace.

- **Case studies.** We end most chapters with a significant case study that shows students how to develop a complex program in stages and how to test it as it is being developed. This structure allows us to demonstrate each new programming construct in a rich context that can’t be achieved with short code examples. Several of the case studies were expanded and improved in the second edition.

- **Utility as a CS1+CS2 textbook.** In recent editions, we added chapters that extend the coverage of the book to cover all of the topics from our second course in computer science, making the book usable for a two-course sequence. Chapters 12–19
explore recursion, searching and sorting, stacks and queues, collection implementation, linked lists, binary trees, hash tables, heaps, and more. Chapter 12 also received a section on recursive backtracking, a powerful technique for exploring a set of possibilities for solving problems such as 8 Queens and Sudoku.

This year also marks the release of our new *Building Python Programs* textbook, which brings our “back to basics” approach to the Python language. In recent years Python has seen a surge in popularity in introductory computer science classrooms. We have found that our materials and approach work as well in Python as they do in Java, and we are pleased to offer the choice of two languages to instructors and students.

### Layers and Dependencies

Many introductory computer science books are language-oriented, but the early chapters of our book are layered. For example, Java has many control structures (including for-loops, while-loops, and if/else-statements), and many books include all of these control structures in a single chapter. While that might make sense to someone who already knows how to program, it can be overwhelming for a novice who is learning how to program. We find that it is much more effective to spread these control structures into different chapters so that students learn one structure at a time rather than trying to learn them all at once.

The following table shows how the layered approach works in the first six chapters:

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Control Flow</th>
<th>Data</th>
<th>Programming Techniques</th>
<th>Input/Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>methods</td>
<td>String literals</td>
<td>procedural decomposition</td>
<td>println, print</td>
</tr>
<tr>
<td>2</td>
<td>definite loops (for)</td>
<td>variables, expressions, int, double</td>
<td>local variables, class constants, pseudocode</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>return values</td>
<td>using objects</td>
<td>parameters</td>
<td>console input, 2D graphics (optional) printf</td>
</tr>
<tr>
<td>4</td>
<td>conditional (if/else)</td>
<td>char</td>
<td>pre/post conditions, throwing exceptions</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>indefinite loops (while)</td>
<td>boolean</td>
<td>assertions, robust programs</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>Scanner</td>
<td>token/line-based file processing</td>
<td>file I/O</td>
</tr>
</tbody>
</table>

Chapters 1–6 are designed to be worked through in order, with greater flexibility of study then beginning in Chapter 7. Chapter 6 may be skipped, although the case study in Chapter 7 involves reading from a file, a topic that is covered in Chapter 6.
Preface

The following is a dependency chart for the book:

Supplements

http://www.buildingjavaprograms.com/

Answers to all self-check problems appear on our web site and are accessible to anyone. Our web site has the following additional resources for students:

- Online-only supplemental chapters, such as a chapter on creating Graphical User Interfaces
Preface

- **Source code and data files** for all case studies and other complete program examples
- The **DrawingPanel class** used in the optional graphics Supplement 3G

Our web site has the following additional resources for teachers:

- **PowerPoint slides** suitable for lectures
- **Solutions** to exercises and programming projects, along with homework specification documents for many projects
- **Sample exams** and solution keys
- **Additional lab exercises** and **programming exercises** with solution keys
- **Closed lab creation tools** to produce lab handouts with the instructor's choice of problems integrated with the textbook

To access instructor resources, contact us at authors@buildingjavaprograms.com. The same materials are also available at http://www.pearsonhighered.com/cs-resources. To ask other questions related to resources, contact your Pearson sales representative.

**MyLab Programming**

MyLab Programming is an online practice and assessment tool that helps students fully grasp the logic, semantics, and syntax of programming. Through practice exercises and immediate, personalized feedback, MyLab Programming improves the programming competence of beginning students who often struggle with basic concepts and paradigms of popular high-level programming languages. A self-study and homework tool, the MyLab Programming course consists of hundreds of small practice exercises organized around the structure of this textbook. For students, the system automatically detects errors in the logic and syntax of code submissions and offers targeted hints that enable students to figure out what went wrong, and why. For instructors, a comprehensive grade book tracks correct and incorrect answers and stores the code inputted by students for review.

For a full demonstration, to see feedback from instructors and students, or to adopt MyLab Programming for your course, visit the following web site: www.pearson.com/mylab/programming

**VideoNotes**

We have recorded a series of instructional videos to accompany the textbook. They are available at the following web site: http://www.pearsonhighered.com/cs-resources
**Preface**

Roughly 3–4 videos are posted for each chapter. An icon in the margin of the page indicates when a VideoNote is available for a given topic. In each video, we spend 5–15 minutes walking through a particular concept or problem, talking about the challenges and methods necessary to solve it. These videos make a good supplement to the instruction given in lecture classes and in the textbook. Your new copy of the textbook has an access code that will allow you to view the videos.

**Acknowledgments**

First, we would like to thank the many colleagues, students, and teaching assistants who have used and commented on early drafts of this text. We could not have written this book without their input. Special thanks go to Hélène Martin, who pored over early versions of our first edition chapters to find errors and to identify rough patches that needed work. We would also like to thank instructor Benson Limketkai for spending many hours performing a technical proofread of the second edition.

Second, we would like to thank the talented pool of reviewers who guided us in the process of creating this textbook:

- Greg Anderson, Weber State University
- Delroy A. Brinkerhoff, Weber State University
- Ed Brunjes, Miramar Community College
- Tom Capaul, Eastern Washington University
- Tom Cortina, Carnegie Mellon University
- Charles Dierbach, Towson University
- H.E. Dunsmore, Purdue University
- Michael Eckmann, Skidmore College
- Mary Anne Egan, Siena College
- Leonard J. Garrett, Temple University
- Ahmad Ghafarian, North Georgia College & State University
- Raj Gill, Anne Arundel Community College
- Michael Hostetler, Park University
- David Hovemeyer, York College of Pennsylvania
- Chenglie Hu, Carroll College
- Philip Isenhour, Virginia Polytechnic Institute
- Andree Jacobson, University of New Mexico
- David C. Kamper, Sr., Northeastern Illinois University
- Simon G.M. Koo, University of San Diego
- Evan Korth, New York University
Preface

• Joan Krone, Denison University
• John H.E.F. Lasseter, Fairfield University
• Eric Matson, Wright State University
• Kathryn S. McKinley, University of Texas, Austin
• Jerry Mead, Bucknell University
• George Medelinskas, Northern Essex Community College
• John Neitzke, Truman State University
• Dale E. Parson, Kutztown University
• Richard E. Pattis, Carnegie Mellon University
• Frederick Pratter, Eastern Oregon University
• Roger Priebe, University of Texas, Austin
• Dehu Qi, Lamar University
• John Rager, Amherst College
• Amala V.S. Rajan, Middlesex University
• Craig Reinhart, California Lutheran University
• Mike Scott, University of Texas, Austin
• Alexa Sharp, Oberlin College
• Tom Stokke, University of North Dakota
• Leigh Ann Sudol, Fox Lane High School
• Ronald F. Taylor, Wright State University
• Andy Ray Terrel, University of Chicago
• Scott Thede, DePauw University
• Megan Thomas, California State University, Stanislaus
• Dwight Tuinstra, SUNY Potsdam
• Jeannie Turner, Sayre School
• Tammy VanDeGrift, University of Portland
• Thomas John VanDrunen, Wheaton College
• Neal R. Wagner, University of Texas, San Antonio
• Jiangping Wang, Webster University
• Yang Wang, Missouri State University
• Stephen Weiss, University of North Carolina at Chapel Hill
• Laurie Werner, Miami University
• Dianna Xu, Bryn Mawr College
• Carol Zander, University of Washington, Bothell
Finally, we would like to thank the great staff at Pearson who helped produce the book. Michelle Brown, Jeff Holcomb, Maurene Goo, Patty Mahtani, Nancy Kotary, and Kathleen Kenny did great work preparing the first edition. Our copy editors and the staff of Aptara Corp, including Heather Sisan, Brian Baker, Brendan Short, and Rachel Head, caught many errors and improved the quality of the writing. Marilyn Lloyd and Chelsea Bell served well as project manager and editorial assistant respectively on prior editions. For their help with the third edition we would like to thank Kayla Smith-Tarbox, Production Project Manager, and Jenah Blitz-Stoehr, Computer Science Editorial Assistant. Mohinder Singh and the staff at Aptara, Inc., were also very helpful in the final production of the third edition. For their great work on production of the fourth and fifth editions, we thank Louise Capulli and the staff of Lakeside Editorial Services, along with Carole Snyder at Pearson. Special thanks go to our lead editor at Pearson, Matt Goldstein, who has believed in the concept of our book from day one. We couldn’t have finished this job without all of their hard work and support.

Stuart Reges
Marty Stepp

Preface
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 1</td>
<td>Pages 31, 40</td>
</tr>
<tr>
<td>Chapter 2</td>
<td>Pages 65, 76, 92, 100, 115</td>
</tr>
<tr>
<td>Chapter 3</td>
<td>Pages 146, 161, 166, 173, 178</td>
</tr>
<tr>
<td>Chapter 3G</td>
<td>Pages 202, 220</td>
</tr>
<tr>
<td>Chapter 4</td>
<td>Pages 248, 256, 283</td>
</tr>
<tr>
<td>Chapter 5</td>
<td>Pages 329, 333, 337, 339, 362</td>
</tr>
<tr>
<td>Chapter 6</td>
<td>Pages 401, 413, 427</td>
</tr>
<tr>
<td>Chapter 7</td>
<td>Pages 464, 470, 488, 510</td>
</tr>
<tr>
<td>Chapter 8</td>
<td>Pages 540, 552, 560, 573</td>
</tr>
<tr>
<td>Chapter 9</td>
<td>Pages 602, 615, 631</td>
</tr>
<tr>
<td>Chapter 10</td>
<td>Pages 679, 686, 694</td>
</tr>
<tr>
<td>Chapter 11</td>
<td>Pages 723, 737, 745</td>
</tr>
<tr>
<td>Chapter 12</td>
<td>Pages 773, 781, 818</td>
</tr>
<tr>
<td>Chapter 13</td>
<td>Pages 842, 845, 852</td>
</tr>
<tr>
<td>Chapter 14</td>
<td>Pages 897, 904</td>
</tr>
<tr>
<td>Chapter 15</td>
<td>Pages 939, 945, 949</td>
</tr>
<tr>
<td>Chapter 16</td>
<td>Pages 982, 989, 1002</td>
</tr>
<tr>
<td>Chapter 17</td>
<td>Pages 1048, 1049, 1059</td>
</tr>
<tr>
<td>Chapter 18</td>
<td>Pages 1085, 1104</td>
</tr>
</tbody>
</table>
Brief Contents

Chapter 1  Introduction to Java Programming  1
Chapter 2  Primitive Data and Definite Loops  63
Chapter 3  Introduction to Parameters and Objects  142
Supplement 3G Graphics (Optional)  201
Chapter 4  Conditional Execution  243
Chapter 5  Program Logic and Indefinite Loops  320
Chapter 6  File Processing  392
Chapter 7  Arrays  447
Chapter 8  Classes  535
Chapter 9  Inheritance and Interfaces  592
Chapter 10  ArrayLists  667
Chapter 11  Java Collections Framework  722
Chapter 12  Recursion  763
Chapter 13  Searching and Sorting  840
Chapter 14  Stacks and Queues  892
Chapter 15  Implementing a Collection Class  931
Chapter 16  Linked Lists  975
Chapter 17  Binary Trees  1028
Chapter 18  Advanced Data Structures  1083
Chapter 19  Functional Programming with Java 8  1119
Appendix A  Java Summary  1161
Appendix B  The Java API Specification and Javadoc Comments  1176
Appendix C  Additional Java Syntax  1182
Index  1191
# Chapter 1 Introduction to Java Programming

## 1.1 Basic Computing Concepts
- Why Programming? 2
- Hardware and Software 3
- The Digital Realm 4
- The Process of Programming 6
- Why Java? 7
- The Java Programming Environment 8

## 1.2 And Now—Java 10
- String Literals (Strings) 14
  - `System.out.println` 15
  - Escape Sequences 15
  - `print` versus `println` 17
  - Identifiers and Keywords 18
  - A Complex Example: `DrawFigures1` 20
  - Comments and Readability 21

## 1.3 Program Errors 24
- Syntax Errors 24
- Logic Errors (Bugs) 28

## 1.4 Procedural Decomposition 28
- Static Methods 31
- Flow of Control 34
- Methods That Call Other Methods 36
- An Example Runtime Error 39

## 1.5 Case Study: `DrawFigures` 40
- Structured Version 41
- Final Version without Redundancy 43
- Analysis of Flow of Execution 44

# Chapter 2 Primitive Data and Definite Loops 63

## 2.1 Basic Data Concepts 64
- Primitive Types 64
Contents

Expressions 65
JShell 67
Literals 68
Arithmetic Operators 69
Precedence 72
Mixing Types and Casting 74

2.2 Variables 76
Assignment/Declaration Variations 81
String Concatenation 84
Increment/Decrement Operators 87
Variables and Mixing Types 90

2.3 The for Loop 92
Tracing for Loops 94
for Loop Patterns 98
Nested for Loops 100

2.4 Managing Complexity 103
Scope 103
Pseudocode 108
The Table Technique 110
Class Constants 113

2.5 Case Study: Hourglass Figure 115
Problem Decomposition and Pseudocode 115
Initial Structured Version 117
Adding a Class Constant 119
Further Variations 122

Chapter 3 Introduction to Parameters and Objects 142

3.1 Parameters 143
The Mechanics of Parameters 146
Limitations of Parameters 150
Multiple Parameters 153
Parameters versus Constants 156
Overloading of Methods 156

3.2 Methods That Return Values 157
The Math Class 158
Defining Methods That Return Values 161

3.3 Using Objects 165
String Objects 166
Interactive Programs and Scanner Objects 173
Sample Interactive Program 176
## Contents

### Chapter 3.4 Case Study: Projectile Trajectory
- Unstructured Solution 178
- Structured Solution 184

### Supplement 3G Graphics (Optional)
- **3G.1 Introduction to Graphics** 201
  - DrawingPanel 202
  - Drawing Lines and Shapes 203
  - Colors 208
  - Drawing with Loops 211
  - Text and Fonts 215
  - Images 218
- **3G.2 Procedural Decomposition with Graphics** 220
  - A Larger Example: DrawDiamonds 220
- **3G.3 Case Study: Pyramids** 224
  - Unstructured Partial Solution 224
  - Generalizing the Drawing of Pyramids 226
  - Complete Structured Solution 228

### Chapter 4 Conditional Execution
- **4.1 if/else Statements** 243
  - Relational Operators 246
  - Nested if/else Statements 248
  - Object Equality 255
  - Factoring if/else Statements 256
  - Testing Multiple Conditions 258
- **4.2 Cumulative Algorithms** 259
  - Cumulative Sum 259
  - Min/Max Loops 261
  - Cumulative Sum with if 265
  - Roundoff Errors 267
- **4.3 Text Processing** 270
  - The char Type 270
  - char versus int 271
  - Cumulative Text Algorithms 272
  - System.out.printf 274
- **4.4 Methods with Conditional Execution** 279
  - Preconditions and Postconditions 279
  - Throwing Exceptions 279
## Contents

6.2 **Details of Token-Based Processing** 401  
Structure of Files and Consuming Input 403  
Scanner Parameters 407  
Paths and Directories 409  
A More Complex Input File 412  

6.3 **Line-Based Processing** 413  
String Scanners and Line/Token Combinations 415  

6.4 **Advanced File Processing** 420  
Output Files with PrintStream 420  
Guaranteeing That Files Can Be Read 424  

6.5 **Case Study: Zip Code Lookup** 427  

### Chapter 7 Arrays 447

7.1 **Array Basics** 448  
Constructing and Traversing an Array 448  
Accessing an Array 452  
Initializing Arrays 455  
A Complete Array Program 456  
Random Access 461  
Arrays and Methods 464  
The For-Each Loop 467  
The Arrays Class 468  

7.2 **Array-Traversal Algorithms** 470  
Printing an Array 471  
Searching and Replacing 473  
Testing for Equality 475  
Reversing an Array 477  
String Traversal Algorithms 481  
Functional Approach 482  

7.3 **Reference Semantics** 484  
Multiple Objects 486  

7.4 **Advanced Array Techniques** 488  
Shifting Values in an Array 488  
Arrays of Objects 493  
Command-Line Arguments 494  
Nested Loop Algorithms 495  

7.5 **Multidimensional Arrays** 497  
Rectangular Two-Dimensional Arrays 497  
Jagged Arrays 499
### Contents

7.6 **Arrays of Pixels** 504

7.7 **Case Study: Benford’s Law** 509
  - Tallying Values 510
  - Completing the Program 514

**Chapter 8 Classes** 535

8.1 **Object-Oriented Programming** 536
  - Classes and Objects 537
  - Point Objects 539

8.2 **Object State and Behavior** 540
  - Object State: Fields 541
  - Object Behavior: Methods 543
  - The Implicit Parameter 546
  - Mutators and Accessors 548
  - The `toString` Method 550

8.3 **Object Initialization: Constructors** 552
  - The Keyword `this` 557
  - Multiple Constructors 559

8.4 **Encapsulation** 560
  - Private Fields 561
  - Class Invariants 567
  - Changing Internal Implementations 571

8.5 **Case Study: Designing a Stock Class** 573
  - Object-Oriented Design Heuristics 574
  - Stock Fields and Method Headers 576
  - Stock Method and Constructor Implementation 578

**Chapter 9 Inheritance and Interfaces** 592

9.1 **Inheritance Basics** 593
  - Nonprogramming Hierarchies 594
  - Extending a Class 596
  - Overriding Methods 600

9.2 **Interacting with the Superclass** 602
  - Calling Overridden Methods 602
  - Accessing Inherited Fields 603
  - Calling a Superclass’s Constructor 605
  - `DividendStock` Behavior 607
  - The `Object` Class 609
  - The `equals` Method 610
  - The `instanceof` Keyword 613
## Contents

Abstract Data Types (ADTs)  
Linked List Case Study: Sieve  

### 11.2 Sets  
737  
Set Concepts  
TreeSet versus HashSet  
Set Operations  
Set Case Study: Lottery  

### 11.3 Maps  
745  
Basic Map Operations  
Map Views (keySet and values)  
TreeMap versus HashMap  
Map Case Study: WordCount  
Collection Overview  

### Chapter 12 Recursion  
763  

#### 12.1 Thinking Recursively  
764  
A Nonprogramming Example  
An Iterative Solution Converted to Recursion  
Structure of Recursive Solutions  

#### 12.2 A Better Example of Recursion  
771  
Mechanics of Recursion  

#### 12.3 Recursive Functions and Data  
781  
Integer Exponentiation  
Greatest Common Divisor  
Directory Crawler  
Helper Methods  

#### 12.4 Recursive Graphics  
797  

#### 12.5 Recursive Backtracking  
801  
A Simple Example: Traveling North/East  
8 Queens Puzzle  
Solving Sudoku Puzzles  

#### 12.6 Case Study: Prefix Evaluator  
818  
Infix, Prefix, and Postfix Notation  
Evaluating Prefix Expressions  
Complete Program  

### Chapter 13 Searching and Sorting  
840  

#### 13.1 Searching and Sorting in the Java Class Libraries  
841  
Binary Search
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sorting</td>
<td>845</td>
</tr>
<tr>
<td>Shuffling</td>
<td>846</td>
</tr>
<tr>
<td>Custom Ordering with Comparitors</td>
<td>848</td>
</tr>
<tr>
<td><strong>13.2 Program Complexity</strong></td>
<td>852</td>
</tr>
<tr>
<td>Empirical Analysis</td>
<td>855</td>
</tr>
<tr>
<td>Complexity Classes</td>
<td>858</td>
</tr>
<tr>
<td><strong>13.3 Implementing Searching and Sorting Algorithms</strong></td>
<td>861</td>
</tr>
<tr>
<td>Sequential Search</td>
<td>861</td>
</tr>
<tr>
<td>Binary Search</td>
<td>862</td>
</tr>
<tr>
<td>Recursive Binary Search</td>
<td>865</td>
</tr>
<tr>
<td>Searching Objects</td>
<td>868</td>
</tr>
<tr>
<td>Selection Sort</td>
<td>869</td>
</tr>
<tr>
<td><strong>13.4 Case Study: Implementing Merge Sort</strong></td>
<td>873</td>
</tr>
<tr>
<td>Splitting and Merging Arrays</td>
<td>873</td>
</tr>
<tr>
<td>Recursive Merge Sort</td>
<td>876</td>
</tr>
<tr>
<td>Complete Program</td>
<td>879</td>
</tr>
</tbody>
</table>

### Chapter 14 Stacks and Queues

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>14.1 Stack/Queue Basics</strong></td>
<td>893</td>
</tr>
<tr>
<td>Stack Concepts</td>
<td>893</td>
</tr>
<tr>
<td>Queue Concepts</td>
<td>896</td>
</tr>
<tr>
<td><strong>14.2 Common Stack/Queue Operations</strong></td>
<td>897</td>
</tr>
<tr>
<td>Transferring between Stacks and Queues</td>
<td>899</td>
</tr>
<tr>
<td>Sum of a Queue</td>
<td>900</td>
</tr>
<tr>
<td>Sum of a Stack</td>
<td>901</td>
</tr>
<tr>
<td><strong>14.3 Complex Stack/Queue Operations</strong></td>
<td>904</td>
</tr>
<tr>
<td>Removing Values from a Queue</td>
<td>904</td>
</tr>
<tr>
<td>Comparing Two Stacks for Similarity</td>
<td>906</td>
</tr>
<tr>
<td><strong>14.4 Case Study: Expression Evaluator</strong></td>
<td>908</td>
</tr>
<tr>
<td>Splitting into Tokens</td>
<td>909</td>
</tr>
<tr>
<td>The Evaluator</td>
<td>914</td>
</tr>
</tbody>
</table>

### Chapter 15 Implementing a Collection Class

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>15.1 Simple ArrayIntList</strong></td>
<td>932</td>
</tr>
<tr>
<td>Adding and Printing</td>
<td>932</td>
</tr>
<tr>
<td>Thinking about Encapsulation</td>
<td>938</td>
</tr>
<tr>
<td>Dealing with the Middle of the List</td>
<td>939</td>
</tr>
<tr>
<td>Another Constructor and a Constant</td>
<td>944</td>
</tr>
<tr>
<td>Preconditions and Postconditions</td>
<td>945</td>
</tr>
</tbody>
</table>
### Contents

**Chapter 15 Arrays**

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>15.2</td>
<td>A More Complete ArrayIntList</td>
<td>949</td>
</tr>
<tr>
<td>15.3</td>
<td>Advanced Features</td>
<td>955</td>
</tr>
<tr>
<td>15.4</td>
<td>ArrayList&lt;E&gt;</td>
<td>963</td>
</tr>
</tbody>
</table>

**Chapter 16 Linked Lists**

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>16.1</td>
<td>Working with Nodes</td>
<td>976</td>
</tr>
<tr>
<td>16.2</td>
<td>A Linked List Class</td>
<td>989</td>
</tr>
<tr>
<td>16.3</td>
<td>A Complex List Operation</td>
<td>1002</td>
</tr>
<tr>
<td>16.4</td>
<td>An IntList Interface</td>
<td>1008</td>
</tr>
<tr>
<td>16.5</td>
<td>LinkedList&lt;E&gt;</td>
<td>1011</td>
</tr>
</tbody>
</table>

**Chapter 17 Binary Trees**

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>17.1</td>
<td>Binary Tree Basics</td>
<td>1029</td>
</tr>
<tr>
<td>17.2</td>
<td>Tree Traversals</td>
<td>1033</td>
</tr>
<tr>
<td>17.3</td>
<td>Common Tree Operations</td>
<td>1048</td>
</tr>
<tr>
<td>17.4</td>
<td>Binary Search Trees</td>
<td>1052</td>
</tr>
</tbody>
</table>
Contents

The Pattern $x = \text{change}(x)$ 1059
Searching the Tree 1062
Binary Search Tree Complexity 1066

17.5 SearchTree\(<E>\) 1067

Chapter 18 Advanced Data Structures 1083

18.1 Hashing 1084
Array Set Implementations 1084
Hash Functions and Hash Tables 1085
Collisions 1087
Rehashing 1092
Hashing Non-Integer Data 1095
Hash Map Implementation 1098

18.2 Priority Queues and Heaps 1099
Priority Queues 1099
Introduction to Heaps 1101
Removing from a Heap 1103
Adding to a Heap 1104
Array Heap Implementation 1106
Heap Sort 1110

Chapter 19 Functional Programming with Java 8 1119

19.1 Effect-Free Programming 1120

19.2 First-Class Functions 1123
Lambda Expressions 1126

19.3 Streams 1129
Basic Idea 1129
Using Map 1131
Using Filter 1132
Using Reduce 1134
Optional Results 1135

19.4 Function Closures 1136

19.5 Higher-Order Operations on Collections 1139
Working with Arrays 1140
Working with Lists 1141
Working with Files 1145
Contents

19.6 Case Study: Perfect Numbers 1146
   Computing Sums 1147
   Incorporating Square Root 1150
   Just Five and Leveraging Concurrency 1153

Appendix A  Java Summary 1161
Appendix B  The Java API Specification and Javadoc Comments 1176
Appendix C  Additional Java Syntax 1182
Index 1191