

Contents

Preface	iii
Chapter I Object-Oriented Programming and Class Hierarchies	I
1.1 Abstract Data Types (ADTs), Interfaces, and the Java API	2
Interfaces	2
The <code>implements</code> Clause	5
Declaring a Variable of an Interface Type	6
Exercises for Section 1.1	6
1.2 Introduction to OOP	7
A Superclass and Subclass Example	8
Use of <code>this</code>	9
Initializing Data Fields in a Subclass	10
The No-Parameter Constructor	11
Protected Visibility for Superclass Data Fields	11
<i>Is-a</i> versus <i>Has-a</i> Relationships	12
Exercises for Section 1.2	12
1.3 Method Overriding, Method Overloading, and Polymorphism	13
Method Overriding	13
Method Overloading	15
Polymorphism	17
Methods with Class Parameters	17
Exercises for Section 1.3	18
1.4 Abstract Classes	19
Referencing Actual Objects	21
Initializing Data Fields in an Abstract Class	21
Abstract Class Number and the Java Wrapper Classes	21
Summary of Features of Actual Classes, Abstract Classes, and Interfaces	22
Implementing Multiple Interfaces	23
Extending an Interface	23
Exercises for Section 1.4	24
1.5 Class Object and Casting	24
The Method <code>toString</code>	24
Operations Determined by Type of Reference Variable	25
Casting in a Class Hierarchy	26
Using <code>instanceof</code> to Guard a Casting Operation	27
The Class <code>Class</code>	29
Exercises for Section 1.5	29
1.6 A Java Inheritance Example—The Exception Class Hierarchy	30
Division by Zero	30
Array Index Out of Bounds	30
Null Pointer	31
The Exception Class Hierarchy	31
The Class <code>Throwable</code>	31

- Checked and Unchecked Exceptions 32
- Handling Exceptions to Recover from Errors 34
- Using try-catch to Recover from an Error 35
- Throwing an Exception When Recovery Is Not Obvious 35
- Exercises for Section 1.6 36
- 1.7 Packages and Visibility**
 - Packages 37
 - The No-Package-Declared Environment 37
 - Package Visibility 38
 - Visibility Supports Encapsulation 38
 - Exercises for Section 1.7 39
- 1.8 A Shape Class Hierarchy**
 - Case Study*: Processing Geometric Figures 40
 - Exercises for Section 1.8 46
 - Chapter Review**
 - Java Constructs Introduced in This Chapter 47
 - Java API Classes Introduced in This Chapter 47
 - User-Defined Interfaces and Classes in This Chapter 47
 - Quick-Check Exercises 47
 - Review Questions 48
 - Programming Projects 49
 - Answers to Quick-Check Exercises 51

Chapter 2 Lists and the Collections Framework

- 2.1 Algorithm Efficiency and Big-O**
 - Big-O Notation 56
 - Formal Definition of Big-O 57
 - Summary of Notation 60
 - Comparing Performance 60
 - The Power of $O(\log n)$ Algorithms 62
 - Algorithms with Exponential and Factorial Growth Rates 62
 - Exercises for Section 2.1 63
- 2.2 The List Interface and ArrayList Class**
 - The ArrayList Class 65
 - Generic Collections 67
 - Exercises for Section 2.2 69
- 2.3 Applications of ArrayList**
 - A Phone Directory Application 71
 - Exercises for Section 2.3 72
- 2.4 Implementation of an ArrayList Class**
 - The Constructor for Class `KWArrayList<E>` 73
 - The `add(E anEntry)` Method 74
 - The `add(int index, E anEntry)` Method 75
 - The `set` and `get` Methods 76
 - The `remove` Method 76
 - The `reallocate` Method 77
 - Performance of the `KWArrayList` Algorithms 77
 - Exercises for Section 2.4 77

2.5 Single-Linked Lists	78
A List Node	80
Connecting Nodes	81
A Single-Linked List Class	81
Inserting a Node in a List	82
Removing a Node	83
Completing the <code>KWSingleLinkedList</code> Class	84
The get and set Methods	85
The add Methods	85
Exercises for Section 2.5	86
2.6 Double-Linked Lists and Circular Lists	87
The Node Class	88
Inserting into a Double-Linked List	88
Removing from a Double-Linked List	89
A Double-Linked List Class	90
Circular Lists	90
Exercises for Section 2.6	91
2.7 The <code>LinkedList</code> Class and the <code>Iterator</code>, <code>ListIterator</code>, and <code>Iterable</code> Interfaces	91
The <code>LinkedList</code> Class	91
The <code>Iterator</code>	92
The <code>Iterator</code> Interface	93
The Enhanced for Loop	95
The <code>ListIterator</code> Interface	95
Comparison of <code>Iterator</code> and <code>ListIterator</code>	97
Conversion between a <code>ListIterator</code> and an Index	97
The <code>Iterable</code> Interface	97
Exercises for Section 2.7	98
2.8 Application of the <code>LinkedList</code> Class	98
<i>Case Study</i> : Maintaining an Ordered List	99
Exercises for Section 2.8	105
2.9 Implementation of a Double-Linked List Class	105
Implementing the <code>KWLinkedList</code> Methods	106
A Class That Implements the <code>ListIterator</code> Interface	107
The Constructor	108
The <code>hasNext</code> and <code>next</code> Methods	108
The <code>hasPrevious</code> and <code>previous</code> Methods	109
The <code>add</code> Method	110
Inner Classes: <code>Static</code> and <code>Nonstatic</code>	112
Exercises for Section 2.9	113
2.10 The Collections Framework Design	114
The <code>Collection</code> Interface	114
Common Features of Collections	114
The <code>AbstractCollection</code> , <code>AbstractList</code> , and <code>AbstractSequentialList</code> Classes	116
The <code>List</code> and <code>RandomAccess</code> Interfaces (Advanced)	116
Exercises for Section 2.10	117
Chapter Review	117
Java API Interfaces and Classes Introduced in This Chapter	118
User-Defined Interfaces and Classes in this Chapter	119
Quick-Check Exercises	119

Review Questions	119
Programming Projects	120
Answers to Quick-Check Exercises	122

Chapter 3 Testing and Debugging	123
3.1 Types of Testing	124
Preparations for Testing	126
Testing Tips for Program Systems	126
Exercises for Section 3.1	127
3.2 Specifying the Tests	127
Testing Boundary Conditions	127
Exercises for Section 3.2	128
3.3 Stubs and Drivers	129
Stubs	129
Preconditions and Postconditions	129
Drivers	130
Exercises for Section 3.3	130
3.4 The JUnit5 Platform	130
Exercises for Section 3.4	135
3.5 Test-Driven Development	136
Exercises for Section 3.5	140
3.6 Testing Interactive Programs in JUnit	140
ByteArrayInputStream	141
ByteArrayOutputStream	141
Exercises for Section 3.6	142
3.7 Debugging a Program	143
Using a Debugger	144
The IntelliJ and Eclipse Debuggers	144
Exercises for Section 3.7	147
Chapter Review	148
Java API Classes Introduced in This Chapter	149
User-Defined Interfaces and Classes in This Chapter	149
Quick-Check Exercises	149
Review Questions	149
Programming Projects	149
Answers to Quick-Check Exercises	151
Chapter 4 Stacks, Queues, and Deques	152
4.1 Stack Abstract Data Type	153
Specification of the Stack Abstract Data Type	153
Exercises for Section 4.1	155
4.2 Stack Applications	156
<i>Case Study:</i> Finding Palindromes	156
Exercises for Section 4.2	160
4.3 Implementing a Stack	160
Implementing a Stack with an ArrayList Component	160
Implementing a Stack as a Linked Data Structure	162

Comparison of Stack Implementations	163
Exercises for Section 4.3	164
4.4 Additional Stack Applications	164
<i>Case Study</i> : Evaluating Postfix Expressions	165
<i>Case Study</i> : Converting from Infix to Postfix	170
<i>Case Study</i> : Converting Expressions with Parentheses	178
Tying the Case Studies Together	181
Exercises for Section 4.4	181
4.5 Queue Abstract Data Type	182
A Print Queue	182
The Unsuitability of a “Print Stack”	183
A Queue of Customers	183
Using a Queue for Traversing a Multi-Branch Data Structure	183
Specification for a Queue Interface	184
Class <code>LinkedList</code> Implements the Queue Interface	184
Exercises for Section 4.5	185
4.6 Queue Applications	186
<i>Case Study</i> : Maintaining a Queue	186
Exercises for Section 4.6	191
4.7 Implementing the Queue Interface	192
Using a Double-Linked List to Implement the Queue Interface	192
Using a Single-Linked List to Implement the Queue Interface	192
Using a Circular Array to Implement the Queue Interface	194
Overview of the Design	194
Implementing <code>ArrayQueue<E></code>	196
Increasing Queue Capacity	198
Implementing Class <code>ArrayQueue<E>.Iter</code>	199
Comparing the Three Implementations	200
Exercises for Section 4.7	201
4.8 The Deque Interface	201
Classes that Implement Deque	202
Using a Deque as a Queue	203
Using a Deque as a Stack	203
Exercises for Section 4.8	204
Chapter Review	205
Java API Classes Introduced in This Chapter	205
User-Defined Interfaces and Classes in This Chapter	205
Quick-Check Exercises	206
Review Questions	207
Programming Projects	208
Answers to Quick-Check Exercises	211
Chapter 5 Recursion	213
5.1 Recursive Thinking	214
Steps to Design a Recursive Algorithm	216
Proving that a Recursive Method Is Correct	218
Tracing a Recursive Method	218
The Run-Time Stack and Activation Frames	219
Exercises for Section 5.1	220

5.2	Recursive Definitions of Mathematical Formulas	221
	Tail Recursion versus Iteration	225
	Efficiency of Recursion	225
	Exercises for Section 5.2	228
5.3	Recursive Array Search	229
	Design of a Recursive Linear Search Algorithm	229
	Implementation of Linear Search	230
	Design of a Binary Search Algorithm	231
	Efficiency of Binary Search	232
	The Comparable Interface	233
	Implementation of Binary Search	233
	Testing Binary Search	235
	Method Arrays.binarySearch	236
	Exercises for Section 5.3	236
5.4	Recursive Data Structures	236
	Recursive Definition of a Linked List	237
	Class <code>LinkedListRec</code>	237
	Method <code>size</code>	237
	Method <code>toString</code>	238
	Method <code>replace</code>	238
	Method <code>add</code>	239
	Removing a List Node	239
	Exercises for Section 5.4	240
5.5	Problem Solving with Recursion	241
	<i>Case Study:</i> Towers of Hanoi	241
	<i>Case Study:</i> Counting Cells in a Blob	246
	Exercises for Section 5.5	250
5.6	Backtracking	250
	<i>Case Study:</i> Finding a Path through a Maze	251
	Exercises for Section 5.6	255
	Chapter Review	255
	User-Defined Classes in This Chapter	256
	Quick-Check Exercises	256
	Review Questions	256
	Programming Projects	257
	Answers to Quick-Check Exercises	258
Chapter 6 Trees		259
6.1	Tree Terminology and Applications	260
	Tree Terminology	260
	Binary Trees	261
	Some Types of Binary Trees	262
	General Trees	265
	Exercises for Section 6.1	266
6.2	Tree Traversals	267
	Visualizing Tree Traversals	268
	Traversals of Binary Search Trees and Expression Trees	268
	Exercises for Section 6.2	269

6.3	Implementing a BinaryTree Class	270
	The Node<E> Class 270	
	The BinaryTree<E> Class 271	
	The Constructors 272	
	The getLeftSubtree and getRightSubtree Methods 273	
	The isLeaf Method 274	
	The toString Method 274	
	The Recursive toString Method 274	
	Exercises for Section 6.3 276	
6.4	Lambda Expressions and Functional Interfaces	277
	Functional Interfaces 278	
	A General Preorder Traversal Method 281	
	Using preOrderTraverse 282	
	Exercises for Section 6.4 282	
6.5	Binary Search Trees	283
	Overview of a Binary Search Tree 283	
	Performance 284	
	Interface SearchTree 284	
	The BinarySearchTree Class 285	
	Implementing the find Methods 286	
	Insertion into a Binary Search Tree 287	
	Implementing the add Methods 287	
	Removal from a Binary Search Tree 289	
	Implementing the delete Methods 291	
	Method findLargestChild 293	
	Testing a Binary Search Tree 294	
	<i>Case Study: Writing an Index for a Term Paper</i> 294	
	Exercises for Section 6.5 297	
6.6	Heaps and Priority Queues	298
	Inserting an Item into a Heap 298	
	Removing an Item from a Heap 299	
	Implementing a Heap 299	
	Performance of the Heap 302	
	Priority Queues 302	
	The PriorityQueue Class 303	
	The offer Method 305	
	The poll Method 306	
	The Other Methods 307	
	Exercises for Section 6.6 307	
6.7	Huffman Trees	308
	<i>Case Study: Building a Custom Huffman Tree</i> 309	
	Exercises for Section 6.7 315	
	Chapter Review	316
	Java API Interfaces and Classes Introduced in This Chapter 317	
	User-Defined Interfaces and Classes in This Chapter 317	
	Quick-Check Exercises 317	
	Review Questions 318	
	Programming Projects 318	
	Answers to Quick-Check Exercises 321	

Chapter 7 Sets and Maps	322
7.1 Sets and the Set Interface	323
The Set Abstraction	323
The Set Interface and Methods	325
Using Method of to Initialize a Collection	327
Comparison of Lists and Sets	327
Exercises for Section 7.1	328
7.2 Maps and the Map Interface	329
The Map Hierarchy	330
The Map Interface	330
Creating a Map	331
Exercises for Section 7.2	333
7.3 Hash Tables	333
Hash Codes and Index Calculation	334
Methods for Generating Hash Codes	335
Open Addressing	335
Table Wraparound and Search Termination	336
Traversing a Hash Table	338
Deleting an Item Using Open Addressing	338
Reducing Collisions by Expanding the Table Size	338
Algorithm for rehashing	339
Reducing Collisions Using Quadratic Probing	339
Problems with Quadratic Probing	340
Chaining	340
Performance of Hash Tables	341
Performance of Open Addressing versus Chaining	341
Performance of Hash Tables versus Sorted Arrays and Binary Search Trees	342
Storage Requirements for Hash Tables, Sorted Arrays, and Trees	342
Storage requirements for Open Addressing and Chaining	343
Exercises for Section 7.3	343
7.4 Implementing the Hash Table	345
Interface KHashMap	345
Class Entry	345
Class HashtableOpen	346
Class HashtableChain	351
Testing the Hash Table Implementations	354
Exercises for Section 7.4	355
7.5 Implementation Considerations for Maps and Sets	355
Methods hashCode and equals	355
Implementing HashSetOpen	356
Writing HashSetOpen as an Adapter Class	356
Implementing the Java Map and Set Interfaces	357
Interface Map.Entry and Class AbstractMap.SimpleEntry	357
Creating a Set View of a Map	358
Method entrySet and Classes EntrySet and SetIterator	358
Classes TreeMap and TreeSet	359
Exercises for Section 7.5	360

7.6 Additional Applications of Maps	360
<i>Case Study</i> : Implementing a Cell Phone Contact list	360
<i>Case Study</i> : Completing the Huffman Coding Problem	362
Encoding the Huffman Tree	366
Exercises for Section 7.6	367
7.7 Navigable Sets and Maps	367
Application of a NavigableMap	369
Exercises for Section 7.7	371
7.8 Skip-Lists	371
Skip-List Structure	372
Searching a Skip-List	372
Performance of a Skip-List Search	373
Inserting into a Skip-List	373
Increasing the Height of a Skip-List	374
Implementing a Skip-List	374
SkipList Methods for Search and Retrieval	375
Method put for Inserting into a Skip-List	376
Constants and Methods for Computing Random Level	378
Performance of a Skip-List	378
Testing Class Skip-List	379
Exercises for Section 7.8	379
Chapter Review	380
Java API Interfaces and Classes Introduced in This Chapter	381
User-Defined Interfaces and Classes in This Chapter	381
Quick-Check Exercises	381
Review Questions	382
Programming Projects	383
Answers to Quick-Check Exercises	384

Chapter 8 Sorting **385**

8.1 Using Java Sorting Methods	386
Collections.sort Methods	389
Method List.sort	389
Exercises for Section 8.1	390
8.2 Selection Sort	390
Analysis of Selection Sort	391
Implementation of Selection Sort	392
Exercises for Section 8.2	393
8.3 Insertion Sort	393
Analysis of Insertion Sort	395
Implementation of Insertion Sort	395
Exercises for Section 8.3	396
8.4 Comparison of Quadratic Sorts	397
Comparisons versus Exchanges	398
Exercises for Section 8.4	398
8.5 Shell Sort: A Better Insertion Sort	398
Analysis of Shell Sort	400
Implementation of Shell Sort	400
Exercises for Section 8.5	401

8.6 Merge Sort	402
Analysis of Merge	403
Implementation of Merge	403
Design of Merge Sort	404
Trace of Merge Sort Algorithm	404
Analysis of Merge Sort	405
Implementation of Merge Sort	406
Exercises for Section 8.6	406
8.7 Timsort	407
Merging Adjacent Runs	410
Performance of Timsort	410
Implementation of Timsort	411
Exercises for Section 8.7	414
8.8 Heapsort	414
First Version of a Heapsort Algorithm	414
Analysis of Revised Heapsort Algorithm	416
Implementation of Heapsort	417
Exercises for Section 8.8	418
8.9 Quicksort	418
Algorithm for Quicksort	419
Analysis of Quicksort	420
Implementation of Quicksort	420
Algorithm for Partitioning	421
Implementation of partition	422
A Revised partition Algorithm	424
Implementation of Revised partition Method	425
Exercises for Section 8.9	426
8.10 Testing the Sort Algorithms	427
Exercises for Section 8.10	428
8.11 The Dutch National Flag Problem (Optional Topic)	428
<i>Case Study:</i> The Problem of the Dutch National Flag	429
Exercises for Section 8.11	431
Chapter Review	432
Java Classes Introduced in This Chapter	432
User-Defined Classes in This Chapter	432
Quick-Check Exercises	433
Review Questions	433
Programming Projects	433
Answers to Quick-Check Exercises	434
Chapter 9 Self-Balancing Search Trees	435
9.1 Tree Balance and Rotation	436
Why Balance Is Important	436
Rotation	436
Algorithm for Rotation	437
Implementing Rotation	438
Exercises for Section 9.1	440

9.2	AVL Trees	440
	Balancing a Left-Left Tree	440
	Balancing a Left-Right Tree	441
	Four Kinds of Critically Unbalanced Trees	442
	Implementing an AVL Tree	444
	The AVLNode Class	445
	Inserting into an AVL Tree	446
	add Starter Method	447
	Recursive add Method	447
	Initial Algorithm for rebalanceLeft	448
	The Effect of Rotations on Balance	448
	Revised Algorithm for rebalanceLeft	449
	Method rebalanceLeft	449
	The decrementBalance Method	450
	Removal from an AVL Tree	451
	Performance of the AVL Tree	452
	Exercises for Section 9.2	452
9.3	Red-Black Trees	453
	Insertion into a Red-Black Tree	453
	Implementation of Red-Black Tree Class	458
	Algorithm for Red-Black Tree Insertion	458
	The add Starter Method	460
	The Recursive add Method	461
	Removal from a Red-Black Tree	462
	Performance of a Red-Black Tree	462
	The TreeMap and TreeSet Classes	463
	Exercises for Section 9.3	463
9.4	2-3 Trees	464
	Searching a 2-3 Tree	465
	Inserting an Item into a 2-3 Tree	465
	Inserting into a 2-Node Leaf	465
	Inserting into a 3-Node Leaf with a 2-Node Parent	466
	Inserting into a 3-Node Leaf with a 3-Node Parent	466
	Analysis of 2-3 Trees and Comparison with Balanced Binary Trees	468
	Removal from a 2-3 Tree	469
	Exercises for Section 9.4	470
9.5	B-Trees and 2-3-4 Trees	470
	B-Trees	471
	Implementing the B-Tree	472
	Code for the insert Method	474
	The insertIntoNode Method	475
	The splitNode Method	476
	Removal from a B-Tree	478
	B+ Trees	479
	2-3-4 Trees	480
	Relating 2-3-4 Trees to Red-Black Trees	481
	Exercises for Section 9.5	482
	Chapter Review	483
	Java Classes Introduced in This Chapter	484
	User-Defined Interfaces and Classes in This Chapter	484

Quick-Check Exercises	485
Review Questions	486
Programming Projects	486
Answers to Quick-Check Exercises	489

Chapter 10 Graphs	492
10.1 Graph Terminology	493
Visual Representation of Graphs	493
Directed and Undirected Graphs	494
Paths and Cycles	494
Relationship between Graphs and Trees	496
Graph Applications	496
Exercises for Section 10.1	497
10.2 The Graph ADT and Edge Class	497
Representing Vertices and Edges	498
Exercises for Section 10.2	499
10.3 Implementing the Graph ADT	499
Adjacency List	499
Adjacency Matrix	501
Overview of the Hierarchy	501
Declaring the Graph Interface	502
The ListGraph Class	503
Comparing Implementations	506
Exercises for Section 10.3	507
10.4 Traversals of Graphs	508
Breadth-First Search	508
Depth-First Search	513
Exercises for Section 10.4	519
10.5 Applications of Graph Traversals	519
<i>Case Study:</i> Shortest Path through a Maze	519
<i>Case Study:</i> Topological Sort of a Graph	523
Exercises for Section 10.5	526
10.6 Algorithms Using Weighted Graphs	526
Finding the Shortest Path from a Vertex to All Other Vertices	526
Analysis of Dijkstra's Algorithm	528
Minimum Spanning Trees	530
Exercises for Section 10.6	533
10.7 A Heuristic Algorithm A* to Find the Best Path	534
A* (A-Star) an Improvement of Dijkstra's Algorithm	535
Exercises for Section 10.7	541
Chapter Review	541
User-Defined Classes and Interfaces in This Chapter	542
Quick-Check Exercises	542
Review Questions	543
Programming Projects	543
Answers to Quick-Check Exercises	545