



Preface

page xxiii

Algorithms+Data Structures = Programs	1
Overview	1
Learning Objectives	1
Goals	1
A Brief History of Algorithms	2
Why Do Data Structures and Algorithms Matter to Real-World Programmers?	4
Learning from This Book	5
Summary	6
Exercises	6
Notes and Further Reading	7
 1 Java Fundamentals	 8
Introduction	8
Learning Objectives	8
1.1 Hello, Java!	9
1.1.1 The First Program	9
1.1.2 How Java Programs Execute	11
1.2 Variables and Data Types	11
1.2.1 Declaring Variables	12
1.2.2 Variable Names	12
1.2.3 Primitive and Object Types	12
1.2.4 Arithmetic	14
1.2.5 Casting	16

1.3	Working with Objects	16
1.3.1	Reading User Input with <code>Scanner</code>	16
1.3.2	Constants and Formatted Printing	19
1.3.3	Run-Time Exceptions	20
1.3.4	Calculations with <code>Math</code>	20
1.4	Conditional Execution	22
1.4.1	Relational and Logical Operators	22
1.4.2	The <code>if-else</code> Statement	23
1.4.3	Cho-han	25
1.4.4	Comparing Strings and Other Objects	28
1.5	Loops	30
1.5.1	The <code>while</code> Loop	30
1.5.2	The <code>for</code> Loop	31
1.5.3	Monte Carlo Simulation	33
1.6	Static Methods	35
1.6.1	Writing Methods	35
1.6.2	Automatic Documentation with <code>JavaDoc</code>	37
1.7	Project: Substitution Ciphers	38
1.7.1	The Caesar Cipher	39
1.7.2	The Vigenère Cipher	42
1.7.3	The Polybius Square	44
1.7.4	The Russian Nihilist Cipher	47
	Summary	49
	Exercises	50
	Notes and Further Reading	55

2

Object-Oriented Programming

57

	Introduction	57
	Learning Objectives	57
2.1	The Concept of Object-Oriented Programming	57
2.1.1	Classes and Objects	58
2.1.2	Encapsulation	58
2.1.3	Advantages and Disadvantages	59
2.2	Developing a Class Step by Step	60
2.2.1	The Simplest Class: A Collection of Related Variables	60
2.2.2	Custom Constructors	62
2.2.3	The <code>toString</code> Method	64

2.2.4	Access Modifiers	65
2.2.5	Accessor and Mutator Methods	66
2.2.6	Throwing Exceptions	68
2.2.7	More on the <code>static</code> Keyword	69
2.2.8	Null Objects	71
2.3	The <code>ArrayList</code> Class	71
2.3.1	Creating an <code>ArrayList</code>	72
2.3.2	Useful Methods	72
2.3.3	Working with Primitive Types	74
2.4	Project: Deck of Cards	74
2.4.1	Enumerated Types	74
2.4.2	The <code>Card</code> Class	76
2.4.3	The <code>Deck</code> Class	78
2.4.4	Hi-Lo	82
	Summary	84
	Exercises	84
	Notes and Further Reading	90

3

Project: Mindstorms

91

	Introduction	91
	Learning Objectives	91
3.1	Introduction to Java Graphics	91
3.1.1	Drawing Shapes	92
3.1.2	Graphics Methods	93
3.1.3	Colors	95
3.2	Project: Do-It-Yourself Turtle Graphics	96
3.2.1	Basic Commands	96
3.2.2	Setup	97
3.2.3	Instance Variables and Constructor	101
3.2.4	Moving and Angles	102
3.2.5	Drawing	104
3.2.6	Adding New Commands	105
3.2.7	Complex Shapes	105
	Summary	109
	Extensions	110
	Notes and Further Reading	111

4	Arrays	112
	Introduction	112
	Learning Objectives	112
4.1	One-Dimensional Arrays	112
4.1.1	Creating and Accessing Arrays	113
4.1.2	Looping over Arrays	114
4.1.3	Modifying an Array in a Method	116
4.2	Multidimensional Arrays	118
4.2.1	Creating Multidimensional Arrays	119
4.2.2	The “Array of Arrays” Model and Ragged Arrays	120
4.2.3	Looping over Two-Dimensional Arrays	121
4.2.4	Ancient Algorithms: Magic Squares	123
4.3	Project: Cellular Automata	126
4.3.1	One-Dimensional Elementary Cellular Automata	126
4.3.2	Visualizing Elementary Cellular Automata	128
4.3.3	Conway’s Life	132
4.3.4	Animating Life	135
4.4	Example Interview Questions Using Arrays	143
4.4.1	Removing Elements	143
4.4.2	Rotate an Array	144
4.4.3	Maximum Subarray Sum	145
	Summary	146
	Exercises	147
	Notes and Further Reading	153
5	Searching and an Introduction to Algorithm Analysis	154
	Introduction	154
	Learning Objectives	154
5.1	Linear Search and Algorithm Performance	154
5.2	Binary Search	156
5.3	Project: Experimental Performance Analysis	159
5.3.1	Best, Worst, and Average Case Performance	159
5.3.2	Measuring Execution Time	159
5.4	The Growth of Functions	162
5.5	Big-O Notation	163
5.6	Applying Big-O Analysis to Programs	165

5.7	Detailed Analysis of Binary Search	167
5.7.1	Correctness Using a Loop Invariant	167
5.7.2	Complexity	169
5.8	Common Complexities	171
5.9	Other Notations: Ω and Θ	171
5.9.1	Ω Notation for Lower Bounds	172
5.9.2	Θ Notation for Exact Bounds	172
5.10	Limitations of Asymptotic Analysis	173
	Summary	173
	Exercises	174
	Notes and Further Reading	177

6 Lists **178**

	Introduction	178
	Learning Objectives	178
6.1	Abstract Data Types	178
6.1.1	Separating Behaviors from Implementation	179
6.1.2	The List Abstract Data Type	179
6.2	Array Lists	180
6.2.1	Array List Structure	181
6.2.2	Implementing an <code>ArrayList</code> Class	182
6.2.3	Amortized Analysis of the Append Operation	185
6.2.4	Removing and Inserting	187
6.3	Linked Lists	189
6.3.1	Linked List Implementation	189
6.3.2	Adding to the Front	190
6.3.3	Getting an Item	191
6.3.4	Inserting at an Arbitrary Location	192
6.3.5	Removing	195
6.3.6	Doubly Linked and Circular Lists	197
6.4	Project: <i>Snake</i>	201
6.4.1	The <code>Snake</code> class	202
6.4.2	Class Members and Constructor	204
6.4.3	Keyboard Input with the <code>KeyListener</code> Interface	205
6.4.4	The <code>paint</code> Method	206
6.4.5	The Main Game Loop	207
6.5	Example Interview Questions Using Lists	210

6.5.1	Comparing Linked and Array Lists	210
6.5.2	Reversing a List	210
6.5.3	The Tortoise and Hare Algorithm	211
	Summary	212
	Exercises	213
	Notes and Further Reading	215

7 Project: Particle Effects 216

	Introduction	216
	Learning Objectives	216
7.1	Falling Sand	217
7.1.1	Listening for Mouse Input	219
7.1.2	Falling Physics	220
7.1.3	Better Performance with Double Buffering	223
7.1.4	Adding Colors	224
7.2	Boids	225
7.2.1	Flocking with Particles	226
7.2.2	Starting Code	227
7.2.3	The <code>Boid</code> Class	230
7.2.4	Implementing the Flocking Rules	231
	Summary	233
	Extensions	233
	Notes and Further Reading	239

8 Recursion 240

	Introduction	240
	Learning Objectives	240
8.1	Writing Recursive Methods	240
8.1.1	Mathematical Functions	241
8.1.2	Recursive Binary Search	243
8.1.3	How Recursive Methods Execute	244
8.1.4	Stack Overflow Errors	247
8.1.5	Recursion vs. Iteration	247
8.2	Analyzing Recursive Algorithms	249
8.2.1	Recurrence Relations	249

8.2.2	Proving Correctness Using Mathematical Induction	250
8.3	Example: Drawing Recursive Trees	252
8.3.1	Symmetric Trees	252
8.3.2	Adjusting Branch Thickness	254
8.3.3	Asymmetric and Randomized Trees	255
8.4	Example Interview Questions Using Recursion	258
8.4.1	Reverse a List	258
8.4.2	Decimal to Binary Conversion	259
8.4.3	Moving on a Chessboard	260
	Summary	261
	Exercises	262
	Notes and Further Reading	265

9 Project: Generative Art and Fractals 266

	Introduction	266
	Learning Objectives	266
9.1	Homage to <i>Homage to the Square</i>	267
9.2	In the Style of Piet Mondrian	271
9.3	Fractals	276
9.3.1	Sierpiński’s Carpet	277
9.3.2	Vicsek’s Cross	280
9.4	The Mandelbrot Set	283
9.4.1	Definition	283
9.4.2	Visualizing the Set	285
9.4.3	Zooming in on a Point	290
9.4.4	Adding Colors	292
	Extensions	297
	Notes and Further Reading	299

10 Sorting 300

	Introduction	300
	Learning Objectives	300
10.1	Sorting Preliminaries	300
10.1.1	Comparing and Ordering Items	300
10.1.2	Standard Libraries Exist, So Why Are You Reading this Book?	301

10.2	Quadratic Sorting Algorithms	302
10.2.1	Selection Sort	302
10.2.2	Insertion Sort	304
10.3	Quicksort	307
10.3.1	Sorting by Divide-and-Conquer	307
10.3.2	Partitioning	309
10.3.3	Correctness of partition	312
10.3.4	Intuitive Analysis	313
10.3.5	Worst-Case Analysis	315
10.3.6	Average-Case Analysis with Random Pivots	315
10.3.7	Improving Quicksort	317
10.3.8	Java’s Arrays.sort	318
10.4	Merge Sort	318
10.4.1	Implementation	318
10.4.2	Analysis	323
10.4.3	Merge Sort in Practice: Python’s Timsort	323
10.5	Radix Sorting	324
10.6	Example Interview Questions Using Sorting	327
10.6.1	Problems Where You Sort and then Do Something	327
10.6.2	Finding the Median with Quickselect	328
10.6.3	National Flag Problems	329
	Summary	330
	Exercises	331
	Notes and Further Reading	334

11 Stacks 336

	Introduction	336
	Learning Objectives	336
11.1	The Stack Abstract Data Type	336
11.1.1	Stack Methods	337
11.1.2	Example: Validating HTML	337
11.1.3	Implementations	342
11.2	Stack-Based Arithmetic	343
11.2.1	Prefix and Postfix Notations	344
11.2.2	Evaluating Postfix Expressions with a Stack	345
11.2.3	Converting Infix Expressions to Postfix	347
11.3	Project: Tiny Web Browser	349

11.3.1	Viewing an HTML Page	350
11.3.2	Listening for Link Events	354
11.3.3	Implementing the Address Bar and Back Button	354
11.4	Example Interview Questions Using Stacks	356
11.4.1	Removals to Make Matching Parentheses	356
11.4.2	Constant Time Minimum Operation	358
	Summary	359
	Exercises	359
	Notes and Further Reading	361

12 Project: Logic Puzzles 362

	Introduction	362
	Learning Objectives	362
12.1	Solving Puzzles by Backtracking	362
12.1.1	Latin Squares	363
12.1.2	The Backtracking Search Strategy	364
12.1.3	Implementation	367
12.1.4	Sudoku	369
12.2	Who Owns the Fish?	370
12.2.1	Representing the Solution	371
12.2.2	Search Procedure	374
12.2.3	Coding the Constraints	375
	Summary	377
	Extensions	378
	Notes and Further Reading	381

13 Queues and Buffers 382

	Introduction	382
	Learning Objectives	382
13.1	The Queue Abstract Data Type	382
13.1.1	Queue and Deque Methods	383
13.1.2	Java’s Queue and Deque Interfaces	383
13.2	Queue and Deque Implementations	384
13.2.1	Using <code>LinkedList</code>	384
13.2.2	Array-Based Circular Buffers	384
13.2.3	A <code>CircularBuffer</code> Class	386

13.3	Application: Flood-Filling	388
13.3.1	The Breadth-First Flooding Algorithm	389
13.3.2	Flood-Filling in a Matrix Using <code>ArrayDeque</code>	390
13.4	Example Interview Questions Using Queues	392
13.4.1	Count the Number of Islands	392
13.4.2	Fairy Chess Pieces	393
	Summary	395
	Exercises	395
	Notes and Further Reading	398

14 Hashing 399

	Introduction	399
	Learning Objectives	399
14.1	Hash Functions	399
14.1.1	Java’s <code>hashCode</code>	400
14.1.2	A Real-World Hash Function: <code>MurmurHash3</code>	401
14.1.3	Properties of Good Hash Functions	403
14.1.4	Cryptographic Hash Functions	404
14.1.5	Application: Message Authentication	405
14.2	Project: Password Cracking	406
14.2.1	Shadow Password Files	406
14.2.2	Cryptographic Hashing with <code>MessageDigest</code>	407
14.2.3	Brute-Force Cracking	408
14.2.4	Dictionary Attacks	411
14.3	Application: Proof-of-Work and Bitcoin	413
14.3.1	Fighting Email Spam with Hashing	413
14.3.2	Proof-of-Work in the Bitcoin Blockchain	414
	Summary	415
	Exercises	415
	Notes and Further Reading	417

15 Hash Tables 419

	Introduction	419
	Learning Objectives	419
15.1	The Map Abstract Data Type	419
15.1.1	Map Methods	420

15.1.2	Java’s HashMap	421
15.2	Implementing Hash Tables	423
15.2.1	Direct Addressing	423
15.2.2	Chained Hashing	425
15.2.3	Analyzing Chaining	428
15.2.4	Open Addressing with Probing	430
15.2.5	Cuckoo Hashing	432
15.3	Example Interview Questions Using Hash Tables	433
15.3.1	Representing a Set	433
15.3.2	Find the Pairs with a Given Sum	434
15.3.3	Finding Anagrams	434
	Summary	435
	Exercises	436
	Notes and Further Reading	438

16 Project: Ye Olde Shakespearean Search Engine 439

	Introduction	439
	Learning Objectives	439
16.1	Building a Search Index	439
16.2	Implementation	441
16.2.1	Getting the Text	442
16.2.2	The Location Class	443
16.2.3	The Driver Class	444
16.2.4	The Index Class	445
16.2.5	Adding a Script to the Index	448
16.2.6	Constructing the Index	450
16.2.7	Putting It All Together	451
16.3	Extending to Web Search	451
	Summary	452
	Extensions	452
	Notes and Further Reading	454

17 Binary Trees 455

	Introduction	455
	Learning Objectives	455

17.1	Representing Hierarchical Data	455
17.1.1	Dendrology	455
17.1.2	Binary Trees	457
17.1.3	Implementation	458
17.2	Tree Traversals	460
17.3	Application: Syntax Trees	464
17.4	Binary Search Trees	468
17.4.1	Properties	468
17.4.2	Searching	469
17.4.3	Insertion	471
17.4.4	Deletion	473
17.4.5	Analysis	479
17.5	Example Interview Questions Using Trees	480
17.5.1	Mirror a Binary Tree	480
17.5.2	Checking If a Tree Is Height Balanced	482
	Summary	483
	Exercises	483
	Notes and Further Reading	486

18 Self-Balancing Search Trees 488

	Introduction	488
	Learning Objectives	488
18.1	2-3-4 Trees	488
18.1.1	Nonbinary search trees	489
18.1.2	Insertion	490
18.1.3	Deletion	491
18.1.4	Going Wider: B-trees	495
18.2	Red-Black Trees	497
18.2.1	Red-Black Ordering Rules	497
18.2.2	Connection to 2-3-4 Trees	499
18.2.3	Bounds on the Height of the Tree	500
18.2.4	Insertion	500
18.2.5	Deletion	506
	Summary	513
	Exercises	513
	Notes and Further Reading	516

19 **Heaps and Priority Queues** **517**

Introduction	517
Learning Objectives	517
19.1 Tree-Based Heaps	517
19.1.1 Definition	518
19.1.2 Inserting into a Heap	518
19.1.3 Removing the Root Element	520
19.1.4 Initializing a Heap	521
19.1.5 Analysis of Bottom-Up Initialization	522
19.2 Implementing a Heap in an Array	523
19.2.1 Parent–Child Relationships	523
19.2.2 A Heap Class	524
19.3 Application: Heapsort	529
19.4 Example Interview Questions Using Heaps	532
19.4.1 Verify If a Binary Tree Is a Heap	532
19.4.2 Merging Sorted Lists	533
19.4.3 Find the Top- <i>k</i> Largest Elements in an Array	534
Summary	534
Exercises	535
Notes and Further Reading	536

20 **Graph Algorithms** **538**

Introduction	538
Learning Objectives	538
20.1 Graph Basics	538
20.1.1 Some Important Graph Terms	539
20.1.2 Representing Graphs	541
20.1.3 Graphs (or the Lack Thereof) in Java	543
20.2 Traversals	545
20.2.1 Breadth-First Traversal	545
20.2.2 Depth-First Traversal	547
20.2.3 Application: Garbage Collection	549
20.3 Shortest Paths	552
20.3.1 Dijkstra’s Algorithm	552
20.3.2 Analyzing Dijkstra’s Algorithm	555
20.4 Example Interview Questions Using Graphs	559

20.4.1 Snakes and Ladders	559
20.4.2 Finding Cycles	560
Summary	562
Exercises	562
Notes and Further Reading	565

21 Project: Graph-Based Recommendation Engine 566

Introduction	566
Learning Objectives	566
21.1 Bipartite Graphs	566
21.1.1 Definition	567
21.1.2 Testing If a Graph Is Bipartite	568
21.2 Building the Recommendation System	570
21.2.1 The Object Graph and Random Walks	570
21.2.2 Implementation	572
Summary	577
Extensions	577
Notes and Further Reading	580

22 Project: Twisty Little Passages 581

Introduction	581
Learning Objectives	581
22.1 Minimum Spanning Trees	581
22.1.1 Kruskal’s Algorithm	582
22.1.2 Prim’s Algorithm	586
22.1.3 Correctness	590
22.2 Maze Generation	591
22.2.1 Perfect Mazes and Spanning Trees	591
22.2.2 Implementation	592
Summary	600
Extensions	600
Notes and Further Reading	602

<i>References</i>	604
<i>Index</i>	610