

1	Java Programming Basics	1
1.1	Getting Started: Classes, Types, and Objects	2
1.1.1	Base Types	5
1.1.2	Objects	7
1.1.3	Enum Types	14
1.2	Methods	15
1.3	Expressions	20
1.3.1	Literals	20
1.3.2	Operators	21
1.3.3	Casting and Autoboxing/Unboxing in Expressions	25
1.4	Control Flow	27
1.4.1	The If and Switch Statements	27
1.4.2	Loops	29
1.4.3	Explicit Control-Flow Statements	32
1.5	Arrays	34
1.5.1	Declaring Arrays	36
1.5.2	Arrays are Objects	37
1.6	Simple Input and Output	39
1.7	An Example Program	42
1.8	Nested Classes and Packages	45
1.9	Writing a Java Program	47
1.9.1	Design	47
1.9.2	Pseudo-Code	48
1.9.3	Coding	49
1.9.4	Testing and Debugging	53
1.10	Exercises	55
2	Object-Oriented Design	57
2.1	Goals, Principles, and Patterns	58
2.1.1	Object-Oriented Design Goals	58
2.1.2	Object-Oriented Design Principles	59
2.1.3	Design Patterns	62

2.2	Inheritance and Polymorphism	63
2.2.1	Inheritance	63
2.2.2	Polymorphism	65
2.2.3	Using Inheritance in Java	66
2.3	Exceptions	76
2.3.1	Throwing Exceptions	76
2.3.2	Catching Exceptions	78
2.4	Interfaces and Abstract Classes	80
2.4.1	Implementing Interfaces	80
2.4.2	Multiple Inheritance in Interfaces	83
2.4.3	Abstract Classes and Strong Typing	84
2.5	Casting and Generics	85
2.5.1	Casting	85
2.5.2	Generics	89
2.6	Exercises	91
3	Arrays, Linked Lists, and Recursion	95
3.1	Using Arrays	96
3.1.1	Storing Game Entries in an Array	96
3.1.2	Sorting an Array	104
3.1.3	java.util Methods for Arrays and Random Numbers	107
3.1.4	Simple Cryptography with Strings and Character Arrays	109
3.1.5	Two-Dimensional Arrays and Positional Games	112
3.2	Singly Linked Lists	116
3.2.1	Insertion in a Singly Linked List	118
3.2.2	Removing an Element in a Singly Linked List	120
3.3	Doubly Linked Lists	121
3.3.1	Insertion in the Middle of a Doubly Linked List	124
3.3.2	Removal in the Middle of a Doubly Linked List	125
3.3.3	An Implementation of a Doubly Linked List	126
3.4	Circularly Linked Lists and Linked-List Sorting	129
3.4.1	Circularly Linked Lists and Duck, Duck, Goose	129
3.4.2	Sorting a Linked List	134
3.5	Recursion	135
3.5.1	Linear Recursion	141
3.5.2	Binary Recursion	145
3.5.3	Multiple Recursion	148
3.6	Exercises	150

<i>Contents</i>	xvii
4 Analysis Tools	155
4.1 The Seven Functions Used in This Book	156
4.1.1 The Constant Function	156
4.1.2 The Logarithm Function	156
4.1.3 The Linear Function	158
4.1.4 The N-Log-N Function	158
4.1.5 The Quadratic Function	158
4.1.6 The Cubic Function and Other Polynomials	160
4.1.7 The Exponential Function	161
4.1.8 Comparing Growth Rates	163
4.2 Analysis of Algorithms	164
4.2.1 Experimental Studies	165
4.2.2 Primitive Operations	166
4.2.3 Asymptotic Notation	168
4.2.4 Asymptotic Analysis	172
4.2.5 Using the Big-Oh Notation	174
4.2.6 A Recursive Algorithm for Computing Powers	178
4.3 Simple Justification Techniques	179
4.3.1 By Example	179
4.3.2 The “Contra” Attack	179
4.3.3 Induction and Loop Invariants	180
4.4 Exercises	183
5 Stacks and Queues	189
5.1 Stacks	190
5.1.1 The Stack Abstract Data Type	191
5.1.2 A Simple Array-Based Stack Implementation	194
5.1.3 Implementing a Stack with a Generic Linked List	199
5.1.4 Reversing an Array Using a Stack	201
5.1.5 Matching Parentheses and HTML Tags	202
5.2 Queues	206
5.2.1 The Queue Abstract Data Type	206
5.2.2 A Simple Array-Based Queue Implementation	208
5.2.3 Implementing a Queue with a Generic Linked List	212
5.2.4 Round Robin Schedulers	213
5.3 Double-Ended Queues	215
5.3.1 The Deque Abstract Data Type	215
5.3.2 Implementing a Deque	216
5.4 Exercises	219

6	Lists and Iterators	223
6.1	Array Lists	224
6.1.1	The Array List Abstract Data Type	224
6.1.2	The Adapter Pattern	225
6.1.3	A Simple Array-Based Implementation	226
6.1.4	A Simple Interface and the <code>java.util.ArrayList</code> Class	228
6.1.5	Implementing an Array List Using Extendable Arrays	229
6.2	Node Lists	233
6.2.1	Node-Based Operations	233
6.2.2	Positions	234
6.2.3	The Node List Abstract Data Type	234
6.2.4	Doubly Linked List Implementation	238
6.3	Iterators	244
6.3.1	The Iterator and Iterable Abstract Data Types	244
6.3.2	The Java For-Each Loop	246
6.3.3	Implementing Iterators	247
6.3.4	List Iterators in Java	249
6.4	List ADTs and the Collections Framework	251
6.4.1	The Java Collections Framework	251
6.4.2	The <code>java.util.LinkedList</code> Class	252
6.4.3	Sequences	253
6.5	Case Study: The Move-to-Front Heuristic	255
6.5.1	Using a Sorted List and a Nested Class	255
6.5.2	Using a List with the Move-to-Front Heuristic	258
6.5.3	Possible Uses of a Favorites List	259
6.6	Exercises	262
7	Trees	267
7.1	General Trees	268
7.1.1	Tree Definitions and Properties	269
7.1.2	The Tree Abstract Data Type	272
7.1.3	Implementing a Tree	273
7.2	Tree Traversal Algorithms	275
7.2.1	Depth and Height	275
7.2.2	Preorder Traversal	278
7.2.3	Postorder Traversal	281
7.3	Binary Trees	284
7.3.1	The Binary Tree ADT	286
7.3.2	A Binary Tree Interface in Java	286
7.3.3	Properties of Binary Trees	287
7.3.4	A Linked Structure for Binary Trees	289

7.3.5	An Array-List Representation of a Binary Tree	298
7.3.6	Traversals of Binary Trees	300
7.3.7	The Template Method Pattern	307
7.4	Exercises	311
8	Priority Queues	321
8.1	The Priority Queue Abstract Data Type	322
8.1.1	Keys, Priorities, and Total Order Relations	322
8.1.2	Entries and Comparators	324
8.1.3	The Priority Queue ADT	327
8.1.4	Sorting with a Priority Queue	329
8.2	Implementing a Priority Queue with a List	330
8.2.1	Implementation with an Unsorted List	330
8.2.2	Implementation with a Sorted List	330
8.2.3	Selection-Sort and Insertion-Sort	334
8.3	Heaps	336
8.3.1	The Heap Data Structure	336
8.3.2	Complete Binary Trees and Their Representation	339
8.3.3	Implementing a Priority Queue with a Heap	344
8.3.4	A Java Heap Implementation	349
8.3.5	Heap-Sort	352
8.3.6	Bottom-Up Heap Construction ★	354
8.4	Adaptable Priority Queues	358
8.4.1	Methods of the Adaptable Priority Queue ADT	358
8.4.2	Location-Aware Entries	359
8.4.3	Implementing an Adaptable Priority Queue	360
8.5	Exercises	363
9	Maps and Dictionaries	369
9.1	The Map Abstract Data Type	370
9.1.1	A Simple List-Based Map Implementation	373
9.2	Hash Tables	374
9.2.1	Bucket Arrays	374
9.2.2	Hash Functions	375
9.2.3	Hash Codes	376
9.2.4	Compression Functions	379
9.2.5	Collision-Handling Schemes	381
9.2.6	A Java Hash Table Implementation	385
9.2.7	Load Factors and Rehashing	389
9.2.8	Application: Counting Word Frequencies	390
9.3	The Dictionary Abstract Data Type	391
9.3.1	List-Based Dictionaries and Audit Trails	392

9.3.2	Hash Table Dictionary Implementation	395
9.3.3	Ordered Search Tables and Binary Search	396
9.4	Skip Lists	400
9.4.1	Search and Update Operations in a Skip List	402
9.4.2	A Probabilistic Analysis of Skip Lists ★	406
9.5	Extensions and Applications of Dictionaries	409
9.5.1	Supporting Location-Aware Dictionary Entries	409
9.5.2	The Ordered Dictionary ADT	410
9.5.3	Flight Databases and Maxima Sets	412
9.6	Exercises	415
10	Search Trees	419
10.1	Binary Search Trees	420
10.1.1	Searching	421
10.1.2	Update Operations	423
10.1.3	Java Implementation	427
10.2	AVL Trees	431
10.2.1	Update Operations	433
10.2.2	Java Implementation	439
10.3	Splay Trees	442
10.3.1	Splaying	442
10.3.2	When to Splay	446
10.3.3	Amortized Analysis of Splaying ★	448
10.4	(2,4) Trees	453
10.4.1	Multi-Way Search Trees	453
10.4.2	Update Operations for (2,4) Trees	459
10.5	Red-Black Trees	465
10.5.1	Update Operations	467
10.5.2	Java Implementation	480
10.6	Exercises	483
11	Sorting, Sets, and Selection	489
11.1	Merge-Sort	490
11.1.1	Divide-and-Conquer	490
11.1.2	Merging Arrays and Lists	495
11.1.3	The Running Time of Merge-Sort	498
11.1.4	Java Implementations of Merge-Sort	499
11.1.5	Merge-Sort and Recurrence Equations ★	502
11.2	Quick-Sort	503
11.2.1	Randomized Quick-Sort	510
11.2.2	In-Place Quick-Sort	512
11.3	A Lower Bound on Sorting	515

11.4 Bucket-Sort and Radix-Sort	517
11.4.1 Bucket-Sort	517
11.4.2 Radix-Sort	518
11.5 Comparison of Sorting Algorithms	520
11.6 The Set ADT and Union/Find Structures	522
11.6.1 A Simple Set Implementation	523
11.6.2 Partitions with Union-Find Operations	526
11.6.3 A Tree-Based Partition Implementation ★	528
11.7 Selection	531
11.7.1 Prune-and-Search	531
11.7.2 Randomized Quick-Select	532
11.7.3 Analyzing Randomized Quick-Select	533
11.8 Exercises	534
12 Text Processing	541
12.1 String Operations	542
12.1.1 The Java String Class	543
12.1.2 The Java StringBuffer Class	544
12.2 Pattern Matching Algorithms	545
12.2.1 Brute Force	545
12.2.2 The Boyer-Moore Algorithm	547
12.2.3 The Knuth-Morris-Pratt Algorithm	551
12.3 Tries	556
12.3.1 Standard Tries	556
12.3.2 Compressed Tries	560
12.3.3 Suffix Tries	562
12.3.4 Search Engines	566
12.4 Text Compression	567
12.4.1 The Huffman Coding Algorithm	568
12.4.2 The Greedy Method	569
12.5 Text Similarity Testing	570
12.5.1 The Longest Common Subsequence Problem	570
12.5.2 Dynamic Programming	571
12.5.3 Applying Dynamic Programming to the LCS Problem	571
12.6 Exercises	575
13 Graphs	581
13.1 The Graph Abstract Data Type	582
13.1.1 The Graph ADT	587
13.2 Data Structures for Graphs	588
13.2.1 The Edge List Structure	588
13.2.2 The Adjacency List Structure	591

13.2.3	The Adjacency Matrix Structure	593
13.3	Graph Traversals	595
13.3.1	Depth-First Search	595
13.3.2	Implementing Depth-First Search	599
13.3.3	Breadth-First Search	607
13.4	Directed Graphs	610
13.4.1	Traversing a Digraph	612
13.4.2	Transitive Closure	614
13.4.3	Directed Acyclic Graphs	617
13.5	Weighted Graphs	621
13.6	Shortest Paths	622
13.6.1	Dijkstra's Algorithm	623
13.7	Minimum Spanning Trees	632
13.7.1	Kruskal's Algorithm	634
13.7.2	The Prim-Jarník Algorithm	638
13.8	Exercises	641
14	Memory	651
14.1	Memory Management	652
14.1.1	Stacks in the Java Virtual Machine	652
14.1.2	Allocating Space in the Memory Heap	656
14.1.3	Garbage Collection	658
14.2	External Memory and Caching	660
14.2.1	The Memory Hierarchy	660
14.2.2	Caching Strategies	661
14.3	External Searching and B-Trees	666
14.3.1	(a, b) Trees	667
14.3.2	B-Trees	669
14.4	External-Memory Sorting	670
14.4.1	Multi-way Merging	671
14.5	Exercises	672
A	Useful Mathematical Facts	675
	Bibliography	683
	Index	689