

# Contents

	<b>Introduction: Organizing Data</b>	<b>31</b>
	<b>Prelude: Designing Classes</b>	<b>35</b>
	Encapsulation	36
	Specifying Methods	38
	Comments	38
	Preconditions and Postconditions	39
	Assertions	40
	Java Interfaces	41
	Writing an Interface	42
	Implementing an Interface	43
	An Interface as a Data Type	45
	Extending an Interface	46
	Named Constants Within an Interface	47
	Choosing Classes	49
	Identifying Classes	50
	CRC Cards	51
	The Unified Modeling Language	51
	Reusing Classes	54
<b>Chapter 1</b>	<b>Bags</b>	<b>61</b>
	The Bag	62
	A Bag's Behaviors	62
	Specifying a Bag	63
	An Interface	69
	Using the ADT Bag	71
	Using an ADT Is Like Using a Vending Machine	75
	The ADT Set	77
	Java Class Library: The Interface set	77
<b>Java Interlude 1</b>	<b>Generics</b>	<b>83</b>
	Generic Data Types	83
	Generic Types Within an Interface	84
	Generic Classes	85
<b>Chapter 2</b>	<b>Bag Implementations That Use Arrays</b>	<b>89</b>
	Using a Fixed-Size Array to Implement the ADT Bag	90
	An Analogy	90
	A Group of Core Methods	91
	Implementing the Core Methods	92
	Making the Implementation Secure	99
	Testing the Core Methods	101
	Implementing More Methods	103
	Methods That Remove Entries	106
	Using Array Resizing to Implement the ADT Bag	114
	Resizing an Array	114
	A New Implementation of a Bag	117
	The Pros and Cons of Using an Array to Implement the ADT Bag	120

# Table of Contents

<b>Java Interlude 2</b>	<b>Exceptions</b>	<b>125</b>
	The Basics	126
	Handling an Exception	128
	Postpone Handling: The throws Clause	128
	Handle It Now: The try-catch Blocks	129
	Multiple catch Blocks	130
	Throwing an Exception	131
<b>Chapter 3</b>	<b>A Bag Implementation That Links Data</b>	<b>133</b>
	Linked Data	134
	Forming a Chain by Adding to Its Beginning	135
	A Linked Implementation of the ADT Bag	137
	The Private Class Node	137
	An Outline of the Class <code>LinkedBag</code>	138
	Defining Some Core Methods	139
	Testing the Core Methods	143
	The Method <code>getFrequencyOf</code>	144
	The Method <code>contains</code>	145
	Removing an Item from a Linked Chain	146
	The Methods <code>remove</code> and <code>clear</code>	147
	A Class Node That Has Set and Get Methods	151
	The Pros and Cons of Using a Chain to Implement the ADT Bag	154
<b>Chapter 4</b>	<b>The Efficiency of Algorithms</b>	<b>159</b>
	Motivation	160
	Measuring an Algorithm's Efficiency	161
	Counting Basic Operations	163
	Best, Worst, and Average Cases	165
	Big Oh Notation	166
	The Complexities of Program Constructs	168
	Picturing Efficiency	170
	The Efficiency of Implementations of the ADT Bag	173
	An Array-Based Implementation	173
	A Linked Implementation	175
	Comparing the Implementations	176
<b>Chapter 5</b>	<b>Stacks</b>	<b>183</b>
	Specifications of the ADT Stack	184
	Using a Stack to Process Algebraic Expressions	188
	A Problem Solved: Checking for Balanced Delimiters in an Infix Algebraic Expression	189
	A Problem Solved: Transforming an Infix Expression to a Postfix Expression	194
	A Problem Solved: Evaluating Postfix Expressions	199
	A Problem Solved: Evaluating Infix Expressions	201
	The Program Stack	203
	Java Class Library: The Class Stack	204
<b>Chapter 6</b>	<b>Stack Implementations</b>	<b>211</b>
	A Linked Implementation	211
	An Array-Based Implementation	215

	A Vector-Based Implementation	219
	Java Class Library: The Class Vector	220
	Using a Vector to Implement the ADT Stack	220
<b>Chapter 7</b>	<b>Recursion</b>	<b>227</b>
	What Is Recursion?	228
	Tracing a Recursive Method	232
	Recursive Methods That Return a Value	235
	Recursively Processing an Array	237
	Recursively Processing a Linked Chain	240
	The Time Efficiency of Recursive Methods	241
	The Time Efficiency of countDown	242
	The Time Efficiency of Computing $x^n$	243
	A Simple Solution to a Difficult Problem	244
	A Poor Solution to a Simple Problem	249
	Tail Recursion	251
	Indirect Recursion	253
	Using a Stack Instead of Recursion	254
<b>Java Interlude 3</b>	<b>More About Generics</b>	<b>265</b>
	The Interface Comparable	265
	Generic Methods	267
	Bounded Type Parameters	268
	Wildcards	270
	Bounded Wildcards	271
<b>Chapter 8</b>	<b>An Introduction to Sorting</b>	<b>275</b>
	Organizing Java Methods That Sort an Array	276
	Selection Sort	277
	Iterative Selection Sort	278
	Recursive Selection Sort	280
	The Efficiency of Selection Sort	281
	Insertion Sort	281
	Iterative Insertion Sort	283
	Recursive Insertion Sort	285
	The Efficiency of Insertion Sort	287
	Insertion Sort of a Chain of Linked Nodes	287
	Shell Sort	290
	The Algorithm	292
	The Efficiency of Shell Sort	293
	Comparing the Algorithms	293
<b>Chapter 9</b>	<b>Faster Sorting Methods</b>	<b>301</b>
	Merge Sort	302
	Merging Arrays	302
	Recursive Merge Sort	303
	The Efficiency of Merge Sort	305
	Iterative Merge Sort	307
	Merge Sort in the Java Class Library	307
	Quick Sort	308
	The Efficiency of Quick Sort	308
	Creating the Partition	309

# Table of Contents

	Implementing Quick Sort	312
	Quick Sort in the Java Class Library	314
	Radix Sort	314
	Pseudocode for Radix Sort	315
	The Efficiency of Radix Sort	316
	Comparing the Algorithms	316
<b>Java Interlude 4</b>	<b>More About Exceptions</b>	<b>323</b>
	Programmer-Defined Exception Classes	323
	Inheritance and Exceptions	327
	The finally Block	328
<b>Chapter 10</b>	<b>Queues, Deques, and Priority Queues</b>	<b>331</b>
	The ADT Queue	332
	A Problem Solved: Simulating a Waiting Line	336
	A Problem Solved: Computing the Capital Gain in a Sale of Stock	342
	Java Class Library: The Interface Queue	345
	The ADT Deque	346
	A Problem Solved: Computing the Capital Gain in a Sale of Stock	349
	Java Class Library: The Interface Deque	350
	Java Class Library: The Class <i>ArrayDeque</i>	351
	The ADT Priority Queue	351
	A Problem Solved: Tracking Your Assignments	353
	Java Class Library: The Class <i>PriorityQueue</i>	355
<b>Chapter 11</b>	<b>Queue, Deque, and Priority Queue Implementations</b>	<b>361</b>
	A Linked Implementation of a Queue	362
	An Array-Based Implementation of a Queue	366
	A Circular Array	366
	A Circular Array with One Unused Location	369
	Circular Linked Implementations of a Queue	374
	A Two-Part Circular Linked Chain	375
	Java Class Library: The Class <i>AbstractQueue</i>	380
	A Doubly Linked Implementation of a Deque	381
	Possible Implementations of a Priority Queue	385
<b>Chapter 12</b>	<b>Lists</b>	<b>391</b>
	Specifications for the ADT List	392
	Using the ADT List	399
	Java Class Library: The Interface <i>List</i>	403
	Java Class Library: The Class <i>ArrayList</i>	403
<b>Chapter 13</b>	<b>A List Implementation That Uses an Array</b>	<b>409</b>
	Using an Array to Implement the ADT List	410
	An Analogy	410
	The Java Implementation	412
	The Efficiency of Using an Array to Implement the ADT List	420
<b>Chapter 14</b>	<b>A List Implementation That Links Data</b>	<b>427</b>
	Operations on a Chain of Linked Nodes	428
	Adding a Node at Various Positions	428
	Removing a Node from Various Positions	432
	The Private Method <i>getNodeAt</i>	433

	Beginning the Implementation	434
	The Data Fields and Constructor	435
	Adding to the End of the List	437
	Adding at a Given Position Within the List	438
	The Methods isEmpty and toArray	439
	Testing the Core Methods	441
	Continuing the Implementation	442
	A Refined Implementation	445
	The Tail Reference	445
	The Efficiency of Using a Chain to Implement the ADT List	448
	Java Class Library: The Class LinkedList	450
<b>Java Interlude 5</b>	<b>Iterators</b>	<b>457</b>
	What Is an Iterator?	457
	The Interface Iterator	459
	The Interface Iterable	461
	Using the Interface Iterator	461
	Iterable and for-each Loops	465
	The Interface ListIterator	466
	The Interface List Revisited	469
	Using the Interface ListIterator	470
<b>Chapter 15</b>	<b>Iterators for the ADT List</b>	<b>473</b>
	Ways to Implement an Iterator	474
	A Separate Class Iterator	474
	An Inner Class Iterator	477
	A Linked Implementation	478
	An Array-Based Implementation	481
	Why Are Iterator Methods in Their Own Class?	484
	An Array-Based Implementation of the Interface ListIterator	486
	The Inner Class	487
<b>Java Interlude 6</b>	<b>Mutable and Immutable Objects</b>	<b>499</b>
	Mutable Objects	500
	Immutable Objects	502
	Creating a Read-Only Class	502
	Companion Classes	504
<b>Chapter 16</b>	<b>Sorted Lists</b>	<b>507</b>
	Specifications for the ADT Sorted List	508
	Using the ADT Sorted List	511
	A Linked Implementation	512
	The Method add	513
	The Efficiency of the Linked Implementation	520
	An Implementation That Uses the ADT List	520
	Efficiency Issues	523
<b>Java Interlude 7</b>	<b>Inheritance and Polymorphism</b>	<b>529</b>
	Further Aspects of Inheritance	529
	When to Use Inheritance	529
	Protected Access	530
	Abstract Classes and Methods	531
	Interfaces Versus Abstract Classes	533
	Polymorphism	534

# Table of Contents

<b>Chapter 17</b>	<b>Inheritance and Lists</b>	<b>541</b>
	Using Inheritance to Implement a Sorted List	542
	Designing a Base Class	544
	Creating an Abstract Base Class	549
	An Efficient Implementation of a Sorted List	551
	The Method add	551
<b>Chapter 18</b>	<b>Searching</b>	<b>557</b>
	The Problem	558
	Searching an Unsorted Array	558
	An Iterative Sequential Search of an Unsorted Array	559
	A Recursive Sequential Search of an Unsorted Array	560
	The Efficiency of a Sequential Search of an Array	562
	Searching a Sorted Array	562
	A Sequential Search of a Sorted Array	562
	A Binary Search of a Sorted Array	563
	Java Class Library: The Method <code>binarySearch</code>	568
	The Efficiency of a Binary Search of an Array	568
	Searching an Unsorted Chain	569
	An Iterative Sequential Search of an Unsorted Chain	570
	A Recursive Sequential Search of an Unsorted Chain	570
	The Efficiency of a Sequential Search of a Chain	571
	Searching a Sorted Chain	571
	A Sequential Search of a Sorted Chain	571
	A Binary Search of a Sorted Chain	572
	Choosing a Search Method	572
<b>Java Interlude 8</b>	<b>Generics Once Again</b>	<b>579</b>
	More Than One Generic Type	579
<b>Chapter 19</b>	<b>Dictionaries</b>	<b>581</b>
	Specifications for the ADT Dictionary	582
	A Java Interface	586
	Iterators	587
	Using the ADT Dictionary	588
	A Problem Solved: A Directory of Telephone Numbers	589
	A Problem Solved: The Frequency of Words	594
	A Problem Solved: A Concordance of Words	597
	Java Class Library: The Interface <code>Map</code>	600
<b>Chapter 20</b>	<b>Dictionary Implementations</b>	<b>605</b>
	Array-Based Implementations	606
	An Unsorted Array-Based Dictionary	606
	A Sorted Array-Based Dictionary	611
	Linked Implementations	616
	An Unsorted Linked Dictionary	617
	A Sorted Linked Dictionary	618
<b>Chapter 21</b>	<b>Introducing Hashing</b>	<b>625</b>
	What Is Hashing?	626
	Hash Functions	629
	Computing Hash Codes	629
	Compressing a Hash Code into an Index for the Hash Table	632

	Resolving Collisions	633
	Open Addressing with Linear Probing	633
	Open Addressing with Quadratic Probing	638
	Open Addressing with Double Hashing	639
	A Potential Problem with Open Addressing	641
	Separate Chaining	642
<b>Chapter 22</b>	<b>Hashing as a Dictionary Implementation</b>	<b>649</b>
	The Efficiency of Hashing	650
	The Load Factor	650
	The Cost of Open Addressing	651
	The Cost of Separate Chaining	653
	Rehashing	654
	Comparing Schemes for Collision Resolution	655
	A Dictionary Implementation That Uses Hashing	656
	Entries in the Hash Table	656
	Data Fields and Constructors	657
	The Methods <code>getValue</code> , <code>remove</code> , and <code>add</code>	659
	Iterators	664
	Java Class Library: The Class <code>HashMap</code>	665
	Java Class Library: The Class <code>HashSet</code>	666
<b>Chapter 23</b>	<b>Trees</b>	<b>669</b>
	Tree Concepts	670
	Hierarchical Organizations	670
	Tree Terminology	672
	Traversals of a Tree	676
	Traversals of a Binary Tree	677
	Traversals of a General Tree	679
	Java Interfaces for Trees	680
	Interfaces for All Trees	680
	An Interface for Binary Trees	681
	Examples of Binary Trees	682
	Expression Trees	683
	Decision Trees	684
	Binary Search Trees	688
	Heaps	690
	Examples of General Trees	693
	Parse Trees	693
	Game Trees	693
<b>Chapter 24</b>	<b>Tree Implementations</b>	<b>703</b>
	The Nodes in a Binary Tree	704
	A Class of Binary Nodes	705
	An Implementation of the ADT Binary Tree	706
	Creating a Basic Binary Tree	707
	The Method <code>privateSetTree</code>	708
	Accessor and Mutator Methods	711
	Computing the Height and Counting Nodes	711
	Traversals	712
	An Implementation of an Expression Tree	717

# Table of Contents

	General Trees	718
	A Node for a General Tree	718
	Using a Binary Tree to Represent a General Tree	719
<b>Java Interlude 9</b>	<b>Cloning</b>	<b>727</b>
	Cloneable Objects	727
	Cloning an Array	733
	Cloning a Chain	736
	A Sorted List of Clones	739
	Cloning a Binary Node	741
<b>Chapter 25</b>	<b>A Binary Search Tree Implementation</b>	<b>743</b>
	Getting Started	744
	An Interface for the Binary Search Tree	745
	Duplicate Entries	747
	Beginning the Class Definition	748
	Searching and Retrieving	749
	Traversing	750
	Adding an Entry	751
	A Recursive Implementation	752
	An Iterative Implementation	755
	Removing an Entry	756
	Removing an Entry Whose Node Is a Leaf	757
	Removing an Entry Whose Node Has One Child	757
	Removing an Entry Whose Node Has Two Children	758
	Removing an Entry in the Root	761
	A Recursive Implementation	762
	An Iterative Implementation	765
	The Efficiency of Operations	769
	The Importance of Balance	770
	The Order in Which Nodes Are Added	770
	An Implementation of the ADT Dictionary	770
<b>Chapter 26</b>	<b>A Heap Implementation</b>	<b>783</b>
	Reprise: The ADT Heap	784
	Using an Array to Represent a Heap	784
	Adding an Entry	787
	Removing the Root	790
	Creating a Heap	793
	Heap Sort	796
<b>Chapter 27</b>	<b>Balanced Search Trees</b>	<b>805</b>
	AVL Trees	806
	Single Rotations	806
	Double Rotations	809
	Implementation Details	813
	2-3 Trees	817
	Searching a 2-3 Tree	818
	Adding Entries to a 2-3 Tree	819
	Splitting Nodes During Addition	821
	2-4 Trees	822
	Adding Entries to a 2-4 Tree	823
	Comparing AVL, 2-3, and 2-4 Trees	825



	Red-Black Trees	826
	Properties of a Red-Black Tree	827
	Adding Entries to a Red-Black Tree	828
	Java Class Library: The Class TreeMap	834
	B-Trees	834
<b>Chapter 28</b>	<b>Graphs</b>	<b>841</b>
	Some Examples and Terminology	842
	Road Maps	842
	Airline Routes	845
	Mazes	845
	Course Prerequisites	846
	Trees	846
	Traversals	847
	Breadth-First Traversal	848
	Depth-First Traversal	849
	Topological Order	851
	Paths	854
	Finding a Path	854
	The Shortest Path in an Unweighted Graph	854
	The Shortest Path in a Weighted Graph	857
	Java Interfaces for the ADT Graph	860
<b>Chapter 29</b>	<b>Graph Implementations</b>	<b>871</b>
	An Overview of Two Implementations	872
	The Adjacency Matrix	872
	The Adjacency List	873
	Vertices and Edges	874
	Specifying the Class Vertex	875
	The Inner Class Edge	877
	Implementing the Class Vertex	878
	An Implementation of the ADT Graph	881
	Basic Operations	881
	Graph Algorithms	884
<b>Appendix A</b>	<b>Documentation and Programming Style</b>	<b>891</b>
	Naming Variables and Classes	891
	Indenting	892
	Comments	892
	Single-Line Comments	893
	Comment Blocks	893
	When to Write Comments	893
	Java Documentation Comments	893
<b>Appendix B</b>	<b>Java Basics (online)</b>	
	Introduction	
	Applications and Applets	
	Objects and Classes	
	A First Java Application Program	
	Elements of Java	
	Identifiers	
	Reserved Words	
	Variables	

# Table of Contents

- Primitive Types
- Constants
- Assignment Statements
- Assignment Compatibilities
- Type Casting
- Arithmetic Operators and Expressions
- Parentheses and Precedence Rules
- Increment and Decrement Operators
- Special Assignment Operators
- Named Constants
- The Class Math
- Simple Input and Output Using the Keyboard and Screen
  - Screen Output
  - Keyboard Input Using the Class Scanner
- The `if-else` Statement
  - Boolean Expressions
  - Nested Statements
  - Multiway `if-else` Statements
  - The Conditional Operator (*Optional*)
- The `switch` Statement
- Enumerations
- Scope
- Loops
  - The `while` Statement
  - The `for` Statement
  - The `do-while` Statement
  - Additional Loop Information
- The Class `String`
  - Characters Within Strings
  - Concatenation of Strings
  - `String` Methods
- The Class `StringBuilder`
- Using `Scanner` to Extract Pieces of a String
- Arrays
  - Array Parameters and Returned Values
  - Initializing Arrays
  - Array Index Out of Bounds
  - Use of `=` and `==` with Arrays
  - Arrays and the For-Each Loop
  - Multidimensional Arrays
- Wrapper Classes

**Appendix C** **Java Classes (online)**

- Objects and Classes
- Using the Methods in a Java Class
  - References and Aliases
- Defining a Java Class
  - Method Definitions
  - Arguments and Parameters
  - Passing Arguments
  - A Definition of the Class Name

	Constructors	
	The Method <code>toString</code>	
	Methods That Call Other Methods	
	Methods That Return an Instance of Their Class	
	Static Fields and Methods	
	Overloading Methods	
	Enumeration as a Class	
	Packages	
	The Java Class Library	
<b>Appendix D</b>	<b>Creating Classes from Other Classes</b>	<b>899</b>
	Composition	900
	Adapters	902
	Inheritance	903
	Invoking Constructors from Within Constructors	906
	Private Fields and Methods of the Superclass	907
	Overriding and Overloading Methods	908
	Multiple Inheritance	913
	Type Compatibility and Superclasses	913
	The Class <code>Object</code>	914
<b>Appendix E</b>	<b>File Input and Output (online)</b>	
	Preliminaries	
	Why Files?	
	Streams	
	The Kinds of Files	
	File Names	
	Text Files	
	Creating a Text File	
	Reading a Text File	
	Changing Existing Data in a Text File	
	Defining a Method to Open a Stream	
	Binary Files	
	Creating a Binary File of Primitive Data	
	Reading a Binary File of Primitive Data	
	Strings in a Binary File	
	Object Serialization	
	<b>Glossary (online)</b>	
	<b>Index</b>	<b>919</b>