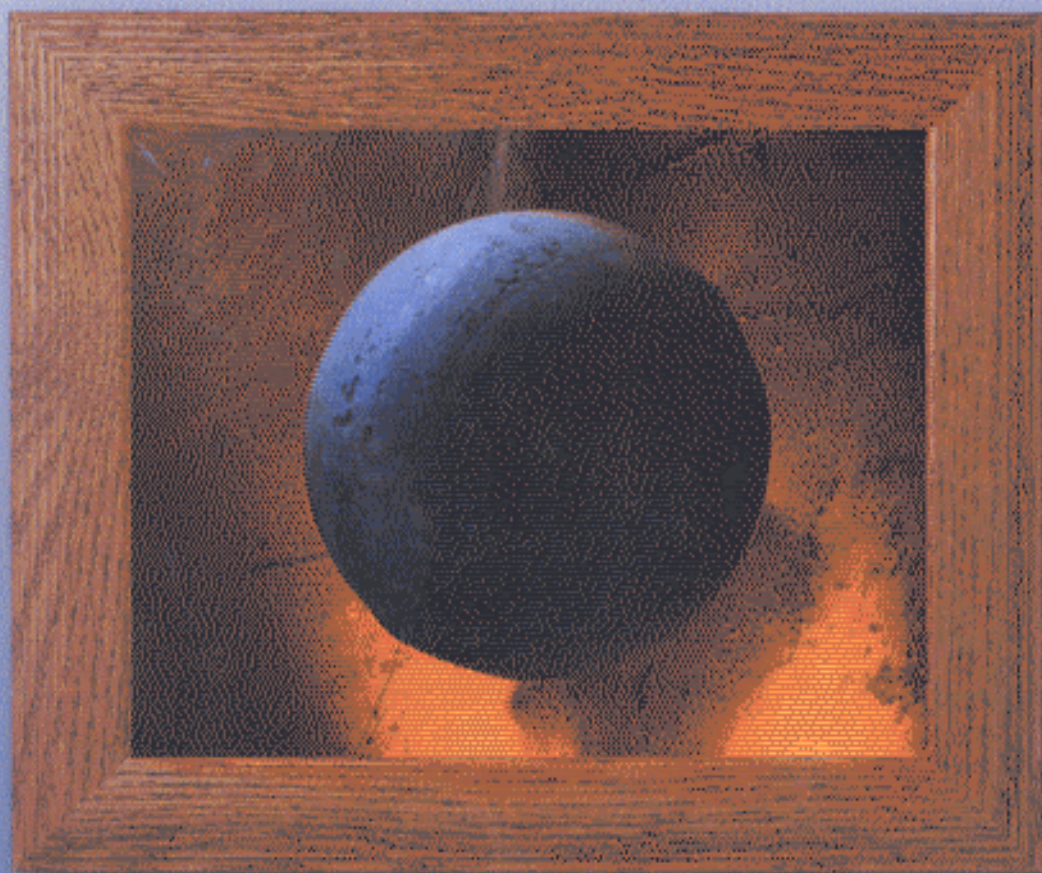


# JAVA<sup>TM</sup>

A Framework for Program Design  
and Data Structures



**Kenneth A. Lambert**

**Martin Osborne**

- 1.1 Collections 1
  - 1.1.1 *Categories of Collections* 1
  - 1.1.2 *Operations on Collections* 3
- 1.2 Abstract Data Types 4
- 1.3 Algorithm Analysis 5
  - 1.3.1 *Simplicity and Clarity* 5
  - 1.3.2 *Space Efficiency* 6
  - 1.3.3 *Time Efficiency* 6
- 1.4 Algorithm Types 7
  - 1.4.1 *Greedy Algorithms* 7
  - 1.4.2 *Divide-and-Conquer Algorithms* 8
  - 1.4.3 *Backtracking Algorithms* 8
- 1.5 The Software Development Process 9
  - 1.5.1 *Complexity* 9
  - 1.5.2 *Fragility* 9
  - 1.5.3 *Malleability* 10
  - 1.5.4 *Interconnectivity* 10
- 1.6 Introduction to Object-Oriented Programming 11
  - 1.6.1 *Procedural Programming* 11
  - 1.6.2 *Functional Programming* 11
  - 1.6.3 *Object-Oriented Programming (OOP)* 11
  - 1.6.4 *An Intuitive Overview of Object-Oriented Programming* 12
- 1.7 The Software Development Life Cycle 14
- 1.8 Our Approach to Software Development 15
  - 1.8.1 *Request* 15
  - 1.8.2 *Analysis* 15
  - 1.8.3 *Design* 16
  - 1.8.4 *Implementation* 16

|        |                                                        |    |
|--------|--------------------------------------------------------|----|
| 1.9    | Testing During Analysis and Design                     | 16 |
| 1.10   | Testing Code                                           | 17 |
| 1.10.1 | Unit, Integration, and Acceptance Testing              | 18 |
| 1.10.2 | What to Test                                           | 18 |
| 1.10.3 | How to Design Test Data                                | 19 |
| 1.11   | Proofs of Correctness                                  | 20 |
| 1.12   | Some Other Aspects of the Software Development Process | 21 |
| 1.12.1 | Coding Conventions                                     | 21 |
| 1.12.2 | Preconditions and Postconditions                       | 22 |
| 1.13   | Development of Hierarchical Systems                    | 24 |
|        | Exercises                                              | 25 |

## **Chapter 2 Review of Object-Oriented Programming in Java**

26

|        |                                                     |    |
|--------|-----------------------------------------------------|----|
| 2.1    | Classes and Objects                                 | 26 |
| 2.1.1  | Example: String Objects                             | 27 |
| 2.1.2  | Objects, Classes, and Computer Memory               | 28 |
| 2.1.3  | Three Characteristics of an Object                  | 28 |
| 2.1.4  | Clients and Servers                                 | 29 |
|        | Exercise                                            | 29 |
| 2.2    | An Employee Class                                   | 29 |
| 2.2.1  | User Requirements                                   | 29 |
| 2.2.2  | Structure of a Class Template                       | 32 |
| 2.2.3  | Design and Implementation of the Employee Class     | 34 |
| 2.2.4  | Constructor Methods and Exceptions                  | 35 |
| 2.2.5  | What Is this?                                       | 37 |
| 2.2.6  | Accessors, Mutators, and toString                   | 38 |
| 2.2.7  | Testing for Equality                                | 39 |
| 2.2.8  | Comparisons and the Comparable Interface            | 40 |
| 2.2.9  | Copying Objects and the Cloneable Interface         | 41 |
| 2.2.10 | Object Serialization                                | 42 |
| 2.2.11 | The Methods finalize and dispose                    | 44 |
| 2.2.12 | Some Helpful Hints for Working with Objects         | 44 |
|        | Exercises                                           | 45 |
| 2.3    | Inheritance and Polymorphism                        | 45 |
| 2.4    | Implementing a Simple Shape Hierarchy               | 46 |
| 2.4.1  | Implementing the Shape Class                        | 46 |
| 2.4.2  | Implementing the Circle Class                       | 48 |
| 2.4.3  | Constructors and super                              | 49 |
| 2.4.4  | Other Methods and super                             | 50 |
| 2.4.5  | Implementing the Rectangle Class                    | 50 |
| 2.4.6  | Protected Variables and Methods                     | 51 |
| 2.4.7  | Implementation, Extension, Overriding, and Finality | 52 |
|        | Exercises                                           | 52 |
| 2.5    | Using the Shape Classes                             | 53 |
| 2.5.1  | Finding the Right Method                            | 54 |
| 2.5.2  | Arrays of Shapes                                    | 54 |
| 2.5.3  | Declaring an Array of Shapes                        | 54 |
| 2.5.4  | Sending Messages to Objects in an Array of Shapes   | 55 |

2.5.5 *The instance of Operator* 55

2.5.6 *Casting to the Rescue* 56

Exercises 56

2.6 Shapes as Parameters and Return Values 57

2.6.1 *Rectangle In, Circle Out* 57

2.6.2 *Any Shape In, Circle Out* 58

2.6.3 *Any Shape In, Any Shape Out* 58

Exercises 60

2.7 Decomposition of Object-Oriented Systems 60

2.7.1 *Finding Classes* 60

2.7.2 *Assigning Responsibilities* 62

2.7.3 *Finding Data Attributes* 63

2.7.4 *Finding Methods* 64

2.7.5 *Relationships between Classes and the Object Model* 64

## **Chapter 3 Introduction to GUI-Based Applications with Java 66**

3.1 A Brief Overview of BreezyGUI and Its Features 66

3.1.1 *The Philosophy of BreezyGUI* 66

3.1.2 *How to Use BreezyGUI* 67

3.1.3 *Laying Out Window Objects* 67

3.1.4 *Handling Events* 69

3.1.5 *Numeric Data Fields* 69

3.1.6 *Message Boxes* 70

3.1.7 *Tester Programs* 71

Exercises 73

3.2 Some Simple Example Programs 73

3.2.1 *Displaying an Employee's Data* 73

3.2.2 *Computing an Employee's Total Hours and Weekly Pay* 75

3.2.3 *Third Example: Using Fields to Modify an Employee's Data* 77

Exercise 78

3.3 Building More Complex GUI Programs 79

3.3.1 *Dialogs* 79

3.3.2 *The Model/View Pattern* 82

Exercise 83

3.4 Case Study: A Simple Employee Database 83

3.4.1 *User Request* 83

3.4.2 *Analysis* 83

3.4.3 *Classes* 84

3.4.4 *Design and Implementation of Class EmployeeModel* 85

3.4.5 *Design and Implementation of the Class*

*EmployeeManagerView* 86

Exercises 89

## **Chapter 4 Complexity 90**

4.1 Measuring the Efficiency of Algorithms 90

4.1.1 *Algorithms and the Resources They Use* 90

4.1.2 *Measuring Execution Time of an Algorithm* 91

|           |                                                                      |     |
|-----------|----------------------------------------------------------------------|-----|
| 4.1.3     | <i>Counting Instructions</i>                                         | 93  |
| 4.1.4     | <i>Measuring the Memory Used by an Algorithm</i>                     | 96  |
| Exercises |                                                                      | 96  |
| 4.2       | <i>Big-O Analysis</i>                                                | 97  |
| 4.2.1     | <i>Orders of Complexity</i>                                          | 97  |
| 4.2.2     | <i>Big-O Notation</i>                                                | 98  |
| 4.2.3     | <i>The Role of the Constant of Proportionality</i>                   | 99  |
| 4.2.4     | <i>Best, Worst, and Average Behavior</i>                             | 99  |
| Exercises |                                                                      | 99  |
| 4.3       | <i>Search Algorithms</i>                                             | 100 |
| 4.3.1     | <i>Linear Search of an Array</i>                                     | 100 |
| 4.3.2     | <i>Binary Search of an Array</i>                                     | 101 |
| Exercises |                                                                      | 103 |
| 4.4       | <i>Sort Algorithms</i>                                               | 104 |
| 4.4.1     | <i>Selection Sort</i>                                                | 104 |
| 4.4.2     | <i>Bubble Sort</i>                                                   | 105 |
| 4.4.3     | <i>Insertion Sort</i>                                                | 107 |
| Exercises |                                                                      | 108 |
| 4.5       | <i>Case Study: Recording Running Times and Counting Instructions</i> | 109 |
| 4.5.1     | <i>Request</i>                                                       | 109 |
| 4.5.2     | <i>Analysis</i>                                                      | 109 |
| 4.5.3     | <i>Design of the Algorithms Class</i>                                | 110 |
| 4.5.4     | <i>Implementation of the Algorithms Class</i>                        | 111 |
| 4.5.5     | <i>Design and Implementation of the AlgorithmProfiler Class</i>      | 112 |
| Exercise  |                                                                      | 113 |

## **Chapter 5    Arrays and Linked Data Structures    114**

|           |                                                               |     |
|-----------|---------------------------------------------------------------|-----|
| 5.1       | <i>Characteristics of Arrays</i>                              | 114 |
| 5.1.1     | <i>Random Access and Contiguous Memory</i>                    | 114 |
| 5.1.2     | <i>Static Memory and Dynamic Memory</i>                       | 115 |
| 5.1.3     | <i>Physical Size and Logical Size</i>                         | 116 |
| 5.2       | <i>Operations on Arrays</i>                                   | 117 |
| 5.2.1     | <i>Increasing the Size of an Array</i>                        | 118 |
| 5.2.2     | <i>Decreasing the Size of an Array</i>                        | 119 |
| 5.2.3     | <i>Inserting an Item into an Array That Grows</i>             | 119 |
| 5.2.4     | <i>Removing an Item from an Array</i>                         | 120 |
| 5.2.5     | <i>A Tester Program for Array Methods</i>                     | 121 |
| 5.2.6     | <i>Complexity Trade-off: Searching and Modifying an Array</i> | 122 |
| Exercises |                                                               | 123 |
| 5.3       | <i>Linked Structures</i>                                      | 124 |
| 5.3.1     | <i>Singly Linked Structures and Doubly Linked Structures</i>  | 124 |
| 5.3.2     | <i>Noncontiguous Memory and Nodes</i>                         | 125 |
| 5.3.3     | <i>Defining a Singly Linked Node Class</i>                    | 127 |
| 5.3.4     | <i>Using the Singly Linked Node Class</i>                     | 127 |
| Exercises |                                                               | 130 |
| 5.4       | <i>Operations on Singly Linked Structures</i>                 | 130 |
| 5.4.1     | <i>Traversal</i>                                              | 130 |

|        |                                                                        |     |
|--------|------------------------------------------------------------------------|-----|
| 5.4.2  | <i>Searching (Object or ith)</i>                                       | 131 |
| 5.4.3  | <i>Replacement (Object or ith)</i>                                     | 132 |
| 5.4.4  | <i>Inserting at the Beginning</i>                                      | 133 |
| 5.4.5  | <i>Inserting at the End</i>                                            | 133 |
| 5.4.6  | <i>Removing at the Beginning</i>                                       | 134 |
| 5.4.7  | <i>Removing at the End</i>                                             | 135 |
| 5.4.8  | <i>Insertions (Object or ith)</i>                                      | 136 |
| 5.4.9  | <i>Removals (Object or ith)</i>                                        | 138 |
| 5.4.10 | <i>Complexity Trade-off: Time, Space, and Singly Linked Structures</i> | 139 |
| 5.5    | <i>Variations on a Link</i>                                            | 140 |
| 5.5.1  | <i>A Circular Linked Structure with a Dummy Header Node</i>            | 140 |
| 5.5.2  | <i>Doubly Linked Structures</i>                                        | 141 |
|        | <b>Exercises</b>                                                       | 144 |

## **Chapter 6 Overview of Collections**

**145**

|       |                                                                  |     |
|-------|------------------------------------------------------------------|-----|
| 6.1   | <i>Introduction</i>                                              | 145 |
| 6.1.1 | <i>Summary of the Book's Collections</i>                         | 146 |
| 6.2   | <i>Multiple Implementations</i>                                  | 149 |
| 6.3   | <i>Collections and Casting</i>                                   | 150 |
|       | <b>Exercises</b>                                                 | 152 |
| 6.4   | <i>Collections and Serialization</i>                             | 153 |
| 6.5   | <i>Iterators</i>                                                 | 154 |
| 6.5.1 | <i>Using an Iterator</i>                                         | 155 |
| 6.5.2 | <i>Implementing Tiny and the iterator Method</i>                 | 156 |
|       | <b>Exercises</b>                                                 | 159 |
| 6.6   | <i>Collection and collectionView</i>                             | 159 |
| 6.6.1 | <i>Two Examples</i>                                              | 160 |
| 6.6.2 | <i>Summary of Methods in the Collection Interface</i>            | 161 |
| 6.6.3 | <i>Implementation of the collectionView Method (optional pv)</i> | 163 |
| 6.6.4 | <i>Implementation of AbstractCollection (optional pv)</i>        | 164 |
|       | <b>Exercises</b>                                                 | 166 |
| 6.7   | <i>Prototypes and Professional Versions</i>                      | 167 |
| 6.7.1 | <i>Overview of the Tiny Professional Version (optional pv)</i>   | 167 |
| 6.7.2 | <i>The Tiny Interface (optional pv)</i>                          | 169 |
| 6.7.3 | <i>Assigning the Tiny Methods to Classes (optional pv)</i>       | 170 |
| 6.7.4 | <i>The Coding Details (optional pv)</i>                          | 171 |
|       | <b>Exercises (optional pv)</b>                                   | 175 |
| 6.8   | <i>Location Guide</i>                                            | 177 |

## **Chapter 7 Stacks**

**178**

|       |                             |     |
|-------|-----------------------------|-----|
| 7.1   | <i>Overview of Stacks</i>   | 178 |
| 7.2   | <i>A Stack Prototype</i>    | 179 |
| 7.3   | <i>Using a Stack</i>        | 181 |
| 7.3.1 | <i>Matching Parentheses</i> | 182 |

|                                                                  |     |
|------------------------------------------------------------------|-----|
| Exercises                                                        | 183 |
| 7.4 Implementations of the Stack Prototype                       | 184 |
| 7.4.1 <i>Array Implementation</i>                                | 184 |
| 7.4.2 <i>Linked Implementation</i>                               | 187 |
| 7.4.3 <i>Time and Space Analysis for the Two Implementations</i> | 189 |
| Exercises                                                        | 190 |
| 7.5 Three Applications of Stacks                                 | 191 |
| 7.5.1 <i>Evaluating Arithmetic Expressions</i>                   | 191 |
| 7.5.2 <i>Backtracking</i>                                        | 195 |
| 7.5.3 <i>Memory Management</i>                                   | 198 |
| Exercises                                                        | 200 |
| 7.6 Interface for a Professional Version                         | 203 |
| 7.7 Implementation of the Professional Version (optional pv)     | 204 |
| Exercises                                                        | 208 |
| 7.8 Case Study: Evaluating Postfix Expressions                   | 209 |
| 7.8.1 <i>Request</i>                                             | 209 |
| 7.8.2 <i>Analysis</i>                                            | 209 |
| 7.8.3 <i>Design</i>                                              | 213 |
| 7.8.4 <i>Implementation</i>                                      | 216 |
| Exercises                                                        | 220 |

## Chapter 8 Queues

222

|                                                                  |     |
|------------------------------------------------------------------|-----|
| 8.1 Overview of Queues                                           | 222 |
| 8.2 A Queue Prototype                                            | 224 |
| Exercises                                                        | 226 |
| 8.3 Implementations of the Queue Prototype                       | 226 |
| 8.3.1 <i>Linked Implementation</i>                               | 226 |
| 8.3.2 <i>Array Implementation</i>                                | 228 |
| 8.3.3 <i>Time and Space Analysis for the Two Implementations</i> | 230 |
| Exercises                                                        | 231 |
| 8.4 Two Applications of Queues                                   | 232 |
| 8.4.1 <i>Simulations</i>                                         | 232 |
| 8.4.2 <i>Round-Robin CPU Scheduling</i>                          | 234 |
| Exercise                                                         | 235 |
| 8.5 Interface for a Professional Version                         | 235 |
| 8.6 Implementation of the Professional Version (optional pv)     | 237 |
| Exercises                                                        | 239 |
| 8.7 Case Study 1: Simulating a Supermarket Checkout Line         | 240 |
| 8.7.1 <i>Request</i>                                             | 240 |
| 8.7.2 <i>Analysis</i>                                            | 240 |
| 8.7.3 <i>Overall Design</i>                                      | 243 |
| 8.7.4 <i>Design of MarketModel</i>                               | 243 |
| 8.7.5 <i>Implementation of MarketModel</i>                       | 244 |
| 8.7.6 <i>Design of Cashier</i>                                   | 245 |



- 8.7.7 *Implementation of Cashier* 246
- 8.7.8 *Design of Customer* 248
- 8.7.9 *Implementation of Customer* 248

Exercises 249

- 8.8 *Priority Queues* 250

- 8.8.1 *Implementation of a Linked Priority Queue* 251

Exercises 252

- 8.9 *Case Study 2: An Emergency Room Scheduler* 253

- 8.9.1 *Request* 253

- 8.9.2 *Analysis* 253

- 8.9.3 *Proposed Interface* 254

Exercise 254

## **Chapter 9 Lists**

**255**

- 9.1 *Overview of Lists* 255

- 9.2 *A List Prototype* 257

Exercises 261

- 9.3 *Using Lists* 261

Exercises 265

- 9.4 *Implementations of the List Prototype* 265

- 9.4.1 *Static Array Implementation* 266

- 9.4.2 *Doubly Linked Implementation* 269

- 9.4.3 *Time and Space Analysis for the Two Implementations* 274

Exercises 277

- 9.5 *Three Applications of Lists* 278

- 9.5.1 *Heap Storage Management* 278

- 9.5.2 *Organization of Files on a Disk* 279

- 9.5.3 *Implementation of Other ADTs* 280

Exercises 281

- 9.6 *Interface for a Professional Version* 282

- 9.7 *Implementation of the Professional Version (optional pv)* 285

Exercises 289

- 9.8 *Case Study: Maintain a List of Tasks (a To-Do List)* 292

- 9.8.1 *Request* 293

- 9.8.2 *Analysis* 293

- 9.8.3 *Design* 295

- 9.8.4 *Implementation* 296

Exercises 301

## **Chapter 10 Recursion, Searching, Sorting, and Backtracking**

**304**

- 10.1 *Overview of Recursion* 304

- 10.1.1 *Implementing Recursion* 306

- 10.1.2 *Tracing Recursive Calls* 307



|           |                                                                       |     |
|-----------|-----------------------------------------------------------------------|-----|
| 10.1.3    | <i>Guidelines for Writing Recursive Methods</i>                       | 308 |
| 10.1.4    | <i>Runtime Support for Recursive Methods</i>                          | 309 |
| 10.1.5    | <i>Time Analysis for Two Recursive Methods</i>                        | 311 |
| 10.1.6    | <i>Space Analysis for Two Recursive Methods</i>                       | 313 |
| Exercises |                                                                       | 313 |
| 10.2      | Recursion and Searching                                               | 315 |
| 10.2.1    | <i>Linear Search of an Array</i>                                      | 315 |
| 10.2.2    | <i>Binary Search of an Array</i>                                      | 316 |
| Exercises |                                                                       | 319 |
| 10.3      | Recursion and Sorting                                                 | 320 |
| 10.3.1    | <i>Quicksort</i>                                                      | 320 |
| 10.3.2    | <i>Mergesort</i>                                                      | 326 |
| Exercises |                                                                       | 329 |
| 10.4      | Recursion and Backtracking                                            | 331 |
| Exercise  |                                                                       | 332 |
| 10.5      | Why Recursion                                                         | 334 |
| 10.5.1    | <i>Getting Rid of Recursion</i>                                       | 334 |
| 10.5.2    | <i>Tail-Recursion</i>                                                 | 335 |
| Exercises |                                                                       | 336 |
| 10.6      | Case Study 1: A Maze Solver                                           | 337 |
| 10.6.1    | <i>Request</i>                                                        | 337 |
| 10.6.2    | <i>Analysis</i>                                                       | 337 |
| 10.6.3    | <i>Classes</i>                                                        | 340 |
| 10.6.4    | <i>Implementation of MazeView</i>                                     | 340 |
| 10.6.5    | <i>Implementation of MazeModel</i>                                    | 341 |
| Exercises |                                                                       | 344 |
| 10.7      | Recursive Descent and Programming Languages                           | 344 |
| 10.7.1    | <i>Introduction to Grammars</i>                                       | 344 |
| 10.7.2    | <i>Recognizing, Parsing, and Interpreting Sentences in a Language</i> | 346 |
| 10.7.3    | <i>Lexical Analysis and the Scanner</i>                               | 347 |
| 10.7.4    | <i>Parsing Strategies</i>                                             | 347 |
| 10.8      | Case Study 2: A Recursive Descent Parser                              | 348 |
| 10.8.1    | <i>Request</i>                                                        | 348 |
| 10.8.2    | <i>Analysis</i>                                                       | 348 |
| 10.8.3    | <i>Proposed Interface</i>                                             | 349 |
| 10.8.4    | <i>Implementation of Parser</i>                                       | 350 |
| Exercise  |                                                                       | 352 |

## Chapter 11 Introduction to Trees

354

|        |                                                       |     |
|--------|-------------------------------------------------------|-----|
| 11.1   | Overview of Trees                                     | 354 |
| 11.1.1 | <i>Talking About Trees</i>                            | 356 |
| 11.1.2 | <i>Formal Definitions of General and Binary Trees</i> | 358 |
| 11.2   | Representations of Trees                              | 359 |
| 11.2.1 | <i>Linked Representations</i>                         | 359 |
| 11.2.2 | <i>Array Representation of a Complete Binary Tree</i> | 361 |

|        |                                                                 |     |
|--------|-----------------------------------------------------------------|-----|
| 11.3   | Binary Tree Operations                                          | 363 |
| 11.3.1 | <i>A Binary Tree Prototype</i>                                  | 363 |
| 11.3.2 | <i>A Tester Program for BinaryTreePT</i>                        | 367 |
| 11.3.3 | <i>Design and Implementation of the Class BinaryTreePT</i>      | 369 |
|        | Exercises                                                       | 372 |
| 11.4   | Case Study 1: Parsing and Expression Trees                      | 373 |
| 11.4.1 | <i>Request</i>                                                  | 374 |
| 11.4.2 | <i>Interface</i>                                                | 374 |
| 11.4.3 | <i>Analysis and Design of the Class ExpressionTree</i>          | 375 |
| 11.4.4 | <i>Analysis, Design, and Implementation of the Class Parser</i> | 377 |
|        | Exercise                                                        | 378 |
| 11.5   | General Tree Operations                                         | 378 |
| 11.5.1 | <i>The Tree Interface</i>                                       | 379 |
| 11.5.2 | <i>The TreeIterator Interface</i>                               | 383 |
| 11.6   | Implementation of the General Tree Class (optional pv)          | 387 |
| 11.6.1 | <i>Responsibilities of the Classes in the Implementation</i>    | 387 |
| 11.6.2 | <i>Design and Implementation of the Class AbstractTree</i>      | 388 |
| 11.6.3 | <i>Design and Implementation of the Class LinkedTree</i>        | 390 |
| 11.6.4 | <i>Design and Implementation of the Tree Iterator</i>           | 393 |
|        | Exercises                                                       | 396 |
| 11.7   | Case Study 2: A Skills Database                                 | 396 |
| 11.7.1 | <i>Request</i>                                                  | 396 |
| 11.7.2 | <i>Analysis</i>                                                 | 396 |
| 11.7.3 | <i>Design and Implementation of the Model Classes</i>           | 399 |
| 11.7.4 | <i>Design and Implementation of the View Classes</i>            | 402 |

## Chapter 12 Special-Purpose Trees

405

|        |                                                    |     |
|--------|----------------------------------------------------|-----|
| 12.1   | Heaps                                              | 405 |
| 12.1.1 | <i>The Shapes of Binary Trees</i>                  | 406 |
| 12.1.2 | <i>Comparable Objects Revisited</i>                | 407 |
| 12.1.3 | <i>The Heap Interface</i>                          | 407 |
| 12.1.4 | <i>The Heap Implementation</i>                     | 409 |
| 12.1.5 | <i>Implementing the Heap Sort</i>                  | 410 |
| 12.1.6 | <i>Implementing add and pop</i>                    | 411 |
| 12.1.7 | <i>Analysis of the Heap Sort</i>                   | 414 |
| 12.1.8 | <i>Implementation of the Iterator</i>              | 414 |
| 12.1.9 | <i>Using a Heap to Implement a Priority Queue</i>  | 415 |
|        | Exercise                                           | 419 |
| 12.2   | Binary Search Trees                                | 419 |
| 12.2.1 | <i>The Sorted Collection ADT and Its Interface</i> | 420 |
| 12.2.2 | <i>Implementation of the Sorted Collection ADT</i> | 421 |
| 12.2.3 | <i>Binary Search Revisited</i>                     | 423 |
| 12.2.4 | <i>The Binary Search Tree Implementation</i>       | 424 |
| 12.2.5 | <i>Analysis of Binary Search Trees</i>             | 427 |
|        | Exercises                                          | 428 |
| 12.3   | "Better" Binary Search Trees (optional)            | 428 |
| 12.3.1 | <i>2-3 Trees</i>                                   | 429 |
| 12.3.2 | <i>AVL Trees</i>                                   | 430 |
|        | Exercises                                          | 433 |

**Chapter 13 Unordered Collections: Sets, Maps, and Bags****435****13.1 Overview of Sets, Maps, and Bags 435***13.1.1 Sets 435**13.1.2 Maps 436**13.1.3 Bags 438**13.1.4 Iterators for Unordered Collections 438**13.1.5 A Brief Example 438***Exercise 440****13.2 Implementation Considerations 440***13.2.1 Hashing 440**13.2.2 Analysis of the Chaining Method of Hashing 441**13.2.3 Other Hashing Strategies 442***13.3 A Prototype Class for Maps 443***13.3.1 The Interface for the Class HashMapPT 443**13.3.2 Design of HashMapPT 444***Exercises 449****13.4 A Prototype Class for Sets 450***13.4.1 The Interface for the Class HashSetPT 450**13.4.2 Design and Implementation 450***Exercises 453****13.5 A Prototype Class for Bags 454***13.5.1 The Interface for Class HashBagPT 454**13.5.2 Design and Implementation 455***Exercises 455****13.6 The Map and Set Classes in the Java Collections Framework 456***13.6.1 Maps 456**13.6.2 Sets 460***Exercises 462****13.7 Adding a Bag ADT to the Collections Framework (optional pv) 462***13.7.1 The Bag Interface 462**13.7.2 The Bag Implementation 464***Exercises 467****13.8 Case Study 1: A File Concordance System 468***13.8.1 Request 468**13.8.2 Analysis 468**13.8.3 Design and Implementation 469***Exercises 475****13.9 Case Study 2: A Credit Card Approval System 475***13.9.1 Request 475**13.9.2 Analysis 475**13.9.3 Classes 476**13.9.4 Design and Implementation 476*

**Chapter 14 Graphs****483**

- 14.1 Introduction 483
- 14.2 Terminology 484
- 14.3 Representations of Graphs 487
  - 14.3.1 *Adjacency Matrix* 488
  - 14.3.2 *Adjacency List* 489
  - 14.3.3 *Analysis of the Two Representations* 489
  - 14.3.4 *Further Run-Time Considerations* 490
- 14.4 Graph Traversals 490
  - 14.4.1 *A Generic Traversal Algorithm* 491
  - 14.4.2 *Breadth-First and Depth-First Traversals* 491
  - 14.4.3 *Graph Components* 493
- 14.5 Trees Within Graphs 494
  - 14.5.1 *Spanning Trees and Forests* 494
  - 14.5.2 *Minimum Spanning Tree* 495
  - 14.5.3 *Algorithms for Minimum Spanning Trees* 495
- 14.6 Topological Sort 497
- 14.7 The Shortest Path Problem 498
  - 14.7.1 *Dijkstra's Algorithm* 499
  - 14.7.2 *The Initialization Step* 499
  - 14.7.3 *The Computation Step* 500
  - 14.7.4 *Analysis* 501
- 14.8 A Graph Prototype 501
  - 14.8.1 *Example Use of the Graph Prototype* 501
  - 14.8.2 *The Class `LinkedGraphPT`* 503
  - 14.8.3 *The Class `LinkedVertexPT`* 509
  - 14.8.4 *The Class `LinkedEdgePT`* 512
- 14.9 A Professional Graph Implementation (optional pv) 515
  - 14.9.1 *The Graph Interface* 515
  - 14.9.2 *Weights* 519
  - 14.9.3 *Implementation* 520
  - 14.9.4 *A Note on Performance* 524
- 14.10 Case Study: Testing Graph Algorithms 525
  - 14.10.1 *Request* 525
  - 14.10.2 *Analysis* 525
  - 14.10.3 *The Class `GraphDemoView`* 526
  - 14.10.4 *The Class `GraphDemoModel`* 529
- Exercises 532

**Chapter 15 Multithreading, Networks, and Client-Server Programming****533**

- 15.1 Threads and Processes 533
  - 15.1.1 *Threads* 534
  - 15.1.2 *Sleeping Threads* 536
  - 15.1.3 *Producer, Consumer, and Synchronization* 538
- Exercises 544

|        |                                                     |     |
|--------|-----------------------------------------------------|-----|
| 15.2   | Networks, Servers, and Clients                      | 545 |
| 15.2.1 | <i>IP Addresses</i>                                 | 545 |
| 15.2.2 | <i>Ports, Servers, Clients</i>                      | 547 |
| 15.2.3 | <i>Sockets and a Day/Time Client Program</i>        | 548 |
| 15.2.4 | <i>Server Sockets and a Day/Time Server Program</i> | 550 |
| 15.2.5 | <i>Making the Server Handle Several Clients</i>     | 554 |
| 15.2.6 | <i>Using a Server Daemon</i>                        | 555 |
| 15.2.7 | <i>Using a Client Handler</i>                       | 557 |
| 15.2.8 | <i>The Model/View Pattern Revisited</i>             | 559 |
| 15.3   | Case Study: A Campus Directory                      | 560 |
| 15.3.1 | <i>Request</i>                                      | 560 |
| 15.3.2 | <i>Analysis</i>                                     | 560 |
| 15.3.3 | <i>Design</i>                                       | 561 |
| 15.3.4 | <i>Implementation</i>                               | 562 |
|        | Exercises                                           | 566 |

|                   |                                                        |            |
|-------------------|--------------------------------------------------------|------------|
| <b>Appendix A</b> | <b>Review of Java Language Features</b>                | <b>567</b> |
| <b>Appendix B</b> | <b>BreezyGUI</b>                                       | <b>589</b> |
| <b>Appendix C</b> | <b>The AWT</b>                                         | <b>597</b> |
| <b>Appendix D</b> | <b>Summary of Hierarchies, Interfaces, and Classes</b> | <b>645</b> |
| <b>Appendix E</b> | <b>Installation Instructions</b>                       | <b>647</b> |
|                   | <b>Glossary</b>                                        | <b>655</b> |
|                   | <b>Index</b>                                           | <b>673</b> |