

Contents

List of Figures	x
List of Tables	xvi
List of Bug Alerts	xviii
Preface	xx

1 WHAT IS PROGRAMMING?

1.1 Mechanical Mouse in a Maze	4
1.2 Object-Oriented Programming	11
1.3 Computers and Data Representations	13
1.4 Compilers	19
1.5 Debugging	23
1.6 Applications and Applets	25

2 CLASSES AND METHODS I: BASICS

2.1 Some Simple Programs	31
2.2 Building Simple Classes	40
2.3 Text Output	43
2.4 Drawing in Java	46

3 FUNDAMENTAL DATA TYPES OF JAVA

3.1	Integers	57
3.2	Declarations, Variables, and Assignment Statements	62
3.3	Real Numbers	66
3.4	Strings	71
3.5	Debugging	77
3.6	Pitfalls of Numbers, Strings, and Characters	88

4 DECISION MAKING

4.1	The <code>if</code> Statement	97
4.2	Constructing and Analyzing Boolean Expressions	109
4.3	Comparing Objects	117
4.4	<code>switch</code> Statements	122
4.5	Debugging Decision Making	128
4.6	More About Boolean Operators	136

5 CLASSES AND METHODS II: CLASSES WITH MULTIPLE METHODS

5.1	Building Classes with Multiple Methods	149
5.2	Initialization of Instance Variables	152
5.3	Scope of Variables	153
5.4	Class Constructors with Arguments	156
5.5	A <code>Clock</code> Class	157

6 ITERATION

6.1	<code>while</code> Loops	173
6.2	<code>for</code> Loops	179
6.3	<code>do-while</code> Loops	181
6.4	Loop Invariants	183
6.5	Reading Input in a Loop	184
6.6	Debugging Loops	192
6.7	More Drawing in Java	201
6.8	Iteration in Graphical Programs	203

7 CLASSES AND METHODS III: WORKING WITH OBJECTS

7.1	Object-Oriented Programming	215
7.2	Clocks Revisited	219
7.3	Constructors	221
7.4	Overloading Methods	223
7.5	Methods Invoking Methods	226
7.6	<code>this</code>	230
7.7	Visibility Qualifiers	235
7.8	Mutability	238
7.9	Design Decisions, Representation Independence, and Debugging	242
7.10	What Is <code>main</code> ?	254

8 ONE-DIMENSIONAL ARRAYS

8.1	Array Basics	263
8.2	Simple Array-Processing Loops	269
8.3	Simple Computations on Numerical Data	271
8.4	Arrays of Objects	277
8.5	Debugging Arrays	283
8.6	Sorting and Searching	296
8.7	One-Dimensional Arrays and Graphics	303

9 NESTED LOOPS AND TWO-DIMENSIONAL ARRAYS

9.1	Nested Loops	313
9.2	Two-Dimensional Arrays	321
9.3	Example: Crossword Puzzles	329
9.4	Mouse in a Maze Revisited	336
9.5	Drawing Pictures (<i>advanced</i>)	345

10 CLASSES AND METHODS IV: STATIC METHODS AND VARIABLES

10.1 Class Variables and Class Methods	365
10.2 Classes with No Instance Variables or Methods	369
10.3 Modular Development and Debugging	375
10.4 Interfaces	388

11 THE JAVA AWT PART I: MOUSE EVENTS (OPTIONAL)

11.1 Mouse Events	399
11.2 Objects in GUI Programs	405
11.3 Debugging Classes	410

12 INHERITANCE AND EXCEPTIONS

12.1 Packages	423
12.2 Inheritance	426
12.3 Exceptions	449

13 JAVA AWT PART II (OPTIONAL)

13.1 The Java AWT	473
13.2 The Java AWT Event Model	482
13.3 A Temperature Conversion GUI	485
13.4 Using Conditionals with Reactive Components	491
13.5 Drawing in a Frame	499
13.6 The AWT Component Hierarchy	503
13.7 The Canvas Class	505
13.8 Designing the Screen Layout	509
13.9 A Calendar Program	516
13.10 Java and the Web	522

14 RECURSION

14.1	Introduction to Recursion	537
14.2	A First Example	539
14.3	Divide and Conquer	541
14.4	Under the Hood	543
14.5	Processing Arrays Recursively	546
14.6	Recursive Functions on Lists	564
14.7	Dynamic Programming	580
14.8	Recursive Drawings	583

15 TEXT PROCESSING AND FILE INPUT/OUTPUT

15.1	The Classes <code>String</code> and <code>StringBuffer</code>	607
15.2	Sequential Files	609
15.3	Debugging File I/O	615
15.4	A Mail-Merge Application	619
15.5	A Database Application	623
15.6	Reading Input from the Web (<i>Optional</i>)	631

16 CASE STUDY: THE GAME OF REVERSI

16.1	The Game of Reversi	641
16.2	Organization of the Solution	644
16.3	The Classes	646
16.4	The Reversi Classes	648

A OTHER JAVA FEATURES

A.1	Comments	670
A.2	No Preprocessor	671
A.3	Data Types	671
A.4	Control Structures	674
A.5	The <code>final</code> Modifier	675
A.6	Inner Classes	675

A.7	Concurrency-Related Features	676
A.8	The transient and native Modifiers	677

B PRECEDENCE RULES

C CLASSES IN CSLIB AND THE JAVA API

C.1	Classes in the CSLib package	681
C.2	Classes in the java.lang Package	684
C.3	Classes in the java.util Package	687
C.4	Classes in the java.awt Package	688
C.5	Classes in the java.awt.event Package	695
C.6	Classes in the java.applet Package	698
C.7	Classes in the java.io Package	698

D UML CLASS DIAGRAMS

D.1	Specifying a Class	700
D.2	Associations	703
D.3	Inheritance and Generalization	704
D.4	Aggregations	704
D.5	Interfaces and Realization	705

E JAVADOC AND CSLIB

E.1	Javadoc	706
E.2	The CSLib Package	708

Index

735

List of Figures

1.1	The start of the mouse's trip through the maze.	9
1.2	An instance of mouse receiving messages.	11
1.3	The mouse in the maze in object-oriented form. The algorithm is the same as before, but now the mouse and the maze are both objects that communicate by sending messages.	12
1.4	Relationship between a class and its objects.	12
1.5	Positional notation.	14
1.6	Block diagram of a computer. Computation is done by the CPU working with program and data in main memory; data move between these components at extremely high speed. New programs and data can be brought in from the hard disk; it is much larger and much slower than the main memory.	15
1.7	The fetch/execute cycle. The CPU repeats this cycle, fetching its instruction from the next memory location, until the program is finished.	16
1.8	Representation of black-and-white pictures using numbers. Here, an arrowhead is drawn on a screen with $7 \times 8 = 56$ pixels. (Real screens have sizes closer to $1000 \times 1000 = 1,000,000$ pixels, so they need many more numbers and produce much better images.)	18
1.9	Representations of animations using numbers.	19
1.10	(a) Sound is carried through the air. Microphone turns sound into electrical impulses of varying strength. (b) Analog-to-digital (A/D) converter samples signal at regular intervals ($\frac{1}{44,000}$ second intervals in this illustration). The sampled values (in binary, of course) are sent to the device that will process the digitized signal.	20

1.11	Files occupy space on the hard disk. Files consist of many binary digits. A file also has a name and a type. In most systems, the type is given as the latter part of the name; for example, “xls” indicates a spreadsheet created by Excel.	21
1.12	(a) Programmer types program using a <i>text editor</i> . File is stored on disk with type “JAVA.” Text is represented in binary. (b) Programmer invokes Java compiler. Compiler reads source file and produces object file. (c) Programmer invokes Java interpreter to run object file	22
1.13	The cycle of editing, compiling, and debugging a program. . . .	24
1.14	Applications versus applets.	25
2.1	The output of the HitWall application.	32
2.2	Structure of the HitWall program.	33
2.3	The variable <code>morte</code> is made to refer to a <code>TrickMouse</code> object.	35
2.4	The output of the HelloMorte application.	38
2.5	The output of the WarningMouseClient application.	41
2.6	Output for the Forecast program.	45
2.7	A picture made up of tiny pixels.	46
2.8	The DrawingBox window as a grid.	47
2.9	Drawing concentric circles.	49
3.1	Variables are like containers.	63
3.2	The effect of <code>cent++</code>	64
3.3	The InputBox of the Temperature application.	68
3.4	The OutputBox of the Temperature application.	68
4.1	A flowchart for straight-line code	98
4.2	A flowchart for voting.	98
4.3	Flowchart for the Social Security computation.	99
4.4	A flowchart for cascading if-else statements.	103
4.5	A flowchart for nested if-else statements.	105
4.6	Flowchart for the <i>intended</i> meaning of code (1).	106
4.7	Flowchart for the <i>actual</i> meaning of code (1).	107
4.8	Reorganized code to avoid the dangling <code>else</code>	108
4.9	Flowchart for determining whether exams and homeworks improve.	113
4.10	Improved flowchart for determining whether exams and homeworks improve.	114
4.11	Two variables, each referring to the same object.	117
4.12	Two variables, referring to distinct (!=) but “equal” objects.	118
4.13	Flowchart for the vehicle choice code fragment.	124
4.14	Flowchart for deciding which movies are allowable.	124
4.15	Flowchart for the panty hose computation.	129
5.1	UML diagram for <code>Hose</code> class.	152
5.2	Sample graphical output for a clock.	161

5.3	Geometry of the <code>Clock</code> drawing.	162
5.4	Output from the <code>TwoClocks Client</code>	165
5.5	UML diagram for the <code>TwoClocks</code> program.	166
6.1	Flowchart for the <code>Table of Temperatures</code> program.	175
6.2	Output from a sample program.	176
6.3	Flowchart for the <code>Ten in a Bed</code> program.	178
6.4	Construction of reading/processing loop.	186
6.5	Flowchart for the loop-and-a-half problem.	187
6.6	Drawing in a program.	202
6.7	Drawing in a program.	204
7.1	Rights of clients and classes.	216
7.2	The heap during execution of the <code>Temperature</code> program.	218
7.3	A <code>Clock</code> object displayed in the middle of the <code>DrawingBox</code>	225
7.4	The heap while executing mutating code.	239
7.5	After assigning <code>c1</code> to <code>c2</code>	241
7.6	UML class diagram for the <code>Clock-DrawingBox</code> association.	242
7.7	An object containing a reference to another object.	244
7.8	UML class diagram for the <code>Clock-DrawingBox</code> association.	245
7.9	UML class diagram for the <code>Clock-DrawingBox</code> composition.	248
7.10	Unexpected output from the <code>Clock</code> containing <code>DrawingBox</code>	249
8.1	Simple object created by <code>SimpleClient</code>	266
8.2	An array of strings.	267
8.3	Two array variables pointing to the same array.	268
8.4	<code>grades1</code> and <code>grades2</code> are parallel arrays.	275
8.5	Parallel arrays versus array of <code>Student</code> objects.	280
8.6	UML diagram for <code>GradeBook</code> that uses an array of <code>Students</code>	280
8.7	UML diagram for <code>GradeBook</code> whose client uses an array of <code>Students</code>	282
8.8	Structure of the <code>Histogram</code> program.	284
8.9	UML diagram for the <code>Histogram</code> program.	285
8.10	Output from version 1 of the <code>Histogram</code> program.	287
8.11	Output from version 2 of the <code>Histogram</code> program.	288
8.12	Output from version 3 of the <code>Histogram</code> program.	289
8.13	Output from version 4 of the <code>Histogram</code> program.	289
8.14	Output from version 6 of the <code>Histogram</code> program.	291
8.15	Output from version 7 of the <code>Histogram</code> program.	292
8.16	Output from final version of the histogram program, with intervals 0–0, 1–25, 26–50, 51–75, and 76–100.	295
8.17	Animation images.	306
9.1	Output from simple nested loop.	314
9.2	Output from second simple nested loop.	314
9.3	Wind chill table.	318

9.4	The two-dimensional array <code>energyTable</code>	322
9.5	The array <code>energyTable</code> , after the input loop.	323
9.6	Output from the <code>energyTable</code> program.	324
9.7	The preliminary version of the <code>Crossword</code> class.	331
9.8	Blank crossword.	332
9.9	Final version of the <code>Crossword</code> class (first part).	333
9.9	Final version of the <code>Crossword</code> class (second part).	334
9.10	Partially completed crossword.	336
9.11	The <code>Maze</code> class.	338
9.12	The UML diagram of the mouse-in-a-maze application.	340
9.13	The <code>Mouse</code> class.	341
9.13	The <code>Mouse</code> class (continued).	342
9.14	The <code>MouseDrawer</code> class.	342
9.14	The <code>MouseDrawer</code> class (continued).	343
9.15	The <code>MazeDrawer</code> class.	343
9.15	The <code>MazeDrawer</code> class (continued).	344
9.16	The <code>MouseController</code> class.	344
9.17	The mouse in a maze after the eighth move.	345
9.18	Each picture is made up of pixels.	346
9.19	Drawing grid lines.	349
9.20	Drawing long lines.	350
9.21	The first general line-drawing method.	351
9.22	Drawing lines correctly.	352
9.23	Bresenham's algorithm for drawing lines (incomplete).	353
9.24	Compute the circle relative to (0, 0) and then translate to (x_0, y_0)	355
9.25	The first version of <code>drawCircle</code>	355
9.26	Drawing circles—first try.	355
9.27	Computing one point on the circle gives seven others.	356
9.28	A second version of <code>drawCircle</code>	357
9.29	Drawing circles.	357
10.1	Class with class variable <code>a</code> and instance variables <code>x</code> and <code>y</code>	366
10.2	A UML class diagram for the <code>MouseMazeGUI</code> program.	389
11.1	UML diagram for the <code>ClickableClock</code> program.	401
11.2	Snapshot of the <code>TwoEyes</code> program.	406
11.3	The <code>TwoEyes</code> class.	407
11.4	The <code>Eyes</code> class.	408
11.5	UML diagram for the <code>TwoEyes</code> program.	409
11.6	Geometry of the <code>Eyes</code> program.	410
11.7	Running the <code>Draw</code> program.	411
11.8	A badly drawn <code>Clear</code> button.	414
11.9	The next-to-last implementation doesn't work.	416

12.1	The effect of inheritance. The unfilled closed-head arrow is UML notation for inheritance. A typical C object is shown on the right.	427
12.2	PreciseClock inheriting from Clock. A typical PreciseClock object is shown on the right.	428
12.3	Inheritance hierarchy	430
12.4	PreciseClock inheriting from Clock, overriding toString.	432
12.5	The structure of the Mouse classes.	444
12.6	The final structure of the Mouse classes.	446
12.7	Final version of the Mouse class.	447
12.8	Final version of the RightMouse class.	448
12.9	Final version of the StraightMouse class.	449
12.10	Final version of the MouseController class.	450
12.11	UML for the two main classes of the calculator.	453
12.12	Main loop of the calculator, without error recovery, and client.	454
12.13	Operation class, with parse and calculate.	455
12.14	Main loop of the calculator, with error recovery.	456
13.1	A UML class diagram for the TwoButtons program.	477
13.2	A UML class diagram for the alternative TwoButtons program.	478
13.3	Outline of a program with one reactive component, a button	483
13.4	A UML class diagram for a program with a Button.	483
13.5	A GUI temperature conversion program.	486
13.6	Output from the temperature conversion program.	487
13.7	A UML class diagram for ClosableFrame.	489
13.8	The ClosableFrame class.	490
13.9	Typical output from the BodyMass program.	498
13.10	Typical drawing code.	500
13.11	A screen dump of the RandomCircles program.	501
13.12	A screen dump of the new RandomCircles program with update redefined.	503
13.13	UML for the AWT Component class inheritance hierarchy.	504
13.14	Snapshot of the Clock program.	508
13.15	Poor layout for the BodyMass program.	510
13.16	The BorderLayout manager.	511
13.17	The final body mass applet, resized.	515
13.18	Output of the Calendar program.	516
13.19	A screen dump showing the HTML document on page 523 in Internet Explorer.	524
13.20	A screen dump showing the HTML document on page 524 in Internet Explorer.	525
13.21	A screen dump showing the HTML document on page 525 in Internet Explorer.	526

13.22	A screen dump showing the HTML document on page 526 in Internet Explorer.	527
13.23	An HTML document with an embedded applet.	529
13.24	A screen dump showing the modified HTML document with an embedded applet.	531
14.1	Representing the list 17, 10945, 616, −14.	565
14.2	Building the list 17, 10945, 616, −14.	568
14.3	Tracing the <code>addToEnd_mut</code> method.	574
14.4	Schematic representation of merge sort.	580
14.5	The “call tree” for <code>choose(30, 15)</code> . Notice that <code>choose(29, 13)</code> is computed twice.	581
14.6	Sierpiński curves of orders 1, 2, 3, and 4.	584
14.7	Recursive structure of the Sierpiński curve	584
14.8	Geometry of the Sierpiński curve of order 1.	585
14.9	Geometry of Sierpiński curve of order i . Here, ℓ is the length of the side of a curve of order $i - 1$	585
14.10	Output of the Sierpiński applet.	587
14.11	Hilbert curves of orders 1, 2, 3, and 4.	590
14.12	Recursive structure of the Hilbert curve.	591
14.13	Output of the Hilbert program.	598
15.1	A UML class diagram for some Java input classes.	610
15.2	A UML class diagram for some Java output classes.	613
15.3	The <code>IOException</code> inheritance hierarchy.	618
15.4	A UML class diagram for the <code>CDDatabase</code> program.	625
15.5	Appearance of the CD applet’s <code>QueryManager</code>	634
16.1	The initial board configuration.	642
16.2	White has two possible moves.	643
16.3	White at (6,6) can flip on two spokes.	643
16.4	UML class diagram for the Reversi game.	646
E.1	A screen shot of Javadoc web documentation.	710

List of Tables

2.1	Keywords in Java.	37
2.2	Methods of the <code>TrickMouse</code> class.	40
2.3	Some methods of the <code>OutputBox</code> class.	43
2.4	Some methods of the <code>DrawingBox</code> class.	48
3.1	Integer arithmetic operations.	61
3.2	Precedence rules for evaluating arithmetic expressions.	62
3.3	Real arithmetic operations.	67
3.4	Predefined numeric methods.	70
3.5	Review of data types	71
3.6	Predefined <code>String</code> methods.	72
3.7	Predefined <code>InputBox</code> methods.	93
4.1	Relational operators.	110
4.2	Expanded precedence rules.	115
4.3	ASCII codes for some characters.	121
4.4	Comparison methods of Java's <code>String</code> class.	122
4.5	Boolean operators.	137
4.6	Truth table evaluation of a Boolean expression.	138
4.7	Truth table verification of DeMorgan's law.	139
5.1	Methods for the <code>Horse</code> class.	150
5.2	Methods for the <code>Clock</code> class.	158
6.1	Method of CSLib's <code>Timer</code> class.	205
9.1	Methods of the <code>SFormat</code> class.	321
9.2	U.S. energy consumption, by source	321
9.3	Methods of the <code>Crossword</code> class.	330
9.4	Methods of the <code>Maze</code> class.	337
9.5	Methods of the <code>Mouse</code> class.	340
11.1	Methods defined in the <code>MouseListener</code> interface.	400
11.2	Methods of the <code>MouseEvent</code> class.	418

12.1	Java packages.	424
12.2	Access provided by visibility modifiers (R is receiver access; C is client access).	435
12.3	Java exceptions.	452
13.1	Some methods of the <code>Button</code> class.	479
13.2	Some constants and methods of the <code>Label</code> class.	479
13.3	Some methods of the <code>TextField</code> class.	480
13.4	Some methods of the <code>TextArea</code> class.	480
13.5	Some methods of the <code>Component</code> class.	481
13.6	The <code>ActionListener</code> interface. Action events are generated by buttons and text fields.	482
13.7	Some methods of the <code>Integer</code> class.	488
13.8	Some constants of the <code>WindowEvent</code> class.	488
13.9	The <code>WindowListener</code> interface. Window events are generated by mouse actions on a <code>Window</code> object.	489
13.10	Methods of the <code>EventObject</code> class.	492
13.11	Some methods of the <code>Double</code> class.	493
13.12	Some methods of the <code>Checkbox</code> class.	495
13.13	Some constants and methods of the <code>ItemEvent</code> class.	495
13.14	The <code>ItemListener</code> interface. Item events are generated by check boxes.	495
13.15	Some methods of the <code>CheckboxGroup</code> class.	496
13.16	Some constants and methods of the <code>Font</code> class.	515
13.17	The <code>AdjustmentListener</code> interface. Adjustment events are generated by scrollbars.	518
13.18	Some constants and methods of the <code>Scrollbar</code> class.	518
13.19	A few constants and methods of Java's <code>Calendar</code> class.	522
13.20	Some methods of the <code>Applet</code> class.	530
14.1	Some methods and public instance variables of the <code>Polygon</code> class.	591
15.1	Some instance methods of <code>StringBuffer</code>	608
15.2	The constructor of the <code>FileReader</code> class.	611
15.3	Methods of the <code>BufferedReader</code> class.	611
15.4	The constructor of the <code>FileWriter</code> class.	613
15.5	Methods of the <code>BufferedWriter</code> class	614
15.6	Methods of the <code>PrintWriter</code> class	614
15.7	Methods of the <code>URL</code> class.	632
16.1	Instance methods for a <code>Vector</code>	647
16.2	Instance methods for an <code>Enumeration</code>	647
16.3	Instance variables and methods for the <code>Board</code> class.	653
A.1	Vital statistics for the primitive types.	672
A.2	Bitwise operations of Java.	673

List of Bug Alerts

2.1	Each Class Requires a Separate File	32
2.2	Erroneous Punctuation	51
3.1	Misleading Compiler Error Messages	81
3.2	Capitalization Errors	82
3.3	Logic Errors	86
3.4	Limitation of Type <code>int</code>	88
3.5	Finite Precision of Type <code>double</code>	89
4.1	Dangling <code>else</code>	109
4.2	Combining Relational Operations	111
4.3	Equal/Not Equal Comparisons with Type <code>double</code>	111
4.4	Equals Sign versus Double Equals Sign	112
4.5	Missing <code>break</code>	128
4.6	Double Ampersands Mean <i>and</i> , Double Bars Mean <i>or</i>	136
6.1	Body of <code>while</code> Loop Consists of Single Statements	174
6.2	<code>while</code> Conditions and Type <code>double</code>	176
6.3	Multistatement <code>for</code> Loops	181
6.4	Check Loop Conditions to Avoid <i>Off-by-1</i> Errors	183
8.1	Arrays Are Subscripted from Zero	267
8.2	Array References Can Be Assigned, Arrays Can Be Copied	268
8.3	Remember to Increment Index Variable in <code>while</code> Loop	272
8.4	An Array of Objects Is Initially an Array of Nulls	283
8.5	Swapping Arguments	298
9.1	Don't Use Commas for Indexing Two-Dimensional Arrays	323

12.1	Overriding vs. Overloading	433
12.2	Do Not Assign Superclass Object to Subclass Variable	436
13.1	Be Sure to Set the Frame's Size.	475
13.2	Be Sure to Make the Frame Visible.	476
13.3	Painting, Repainting, and Updating	502
13.4	Don't Paint, Repaint!	509
15.1	FileNotFoundException must be caught for FileReader	611
15.2	IOException must be caught for BufferedReader	611
15.3	FileNotFoundException must be caught for FileWriter	614
15.4	IOException must be caught for BufferedWriter and PrintWriter.	614
16.1	Remember to use nextElement ()	647
16.2	You must cast Vector elements	648
E.1	Always Provide a Zero-Argument Constructor.	714