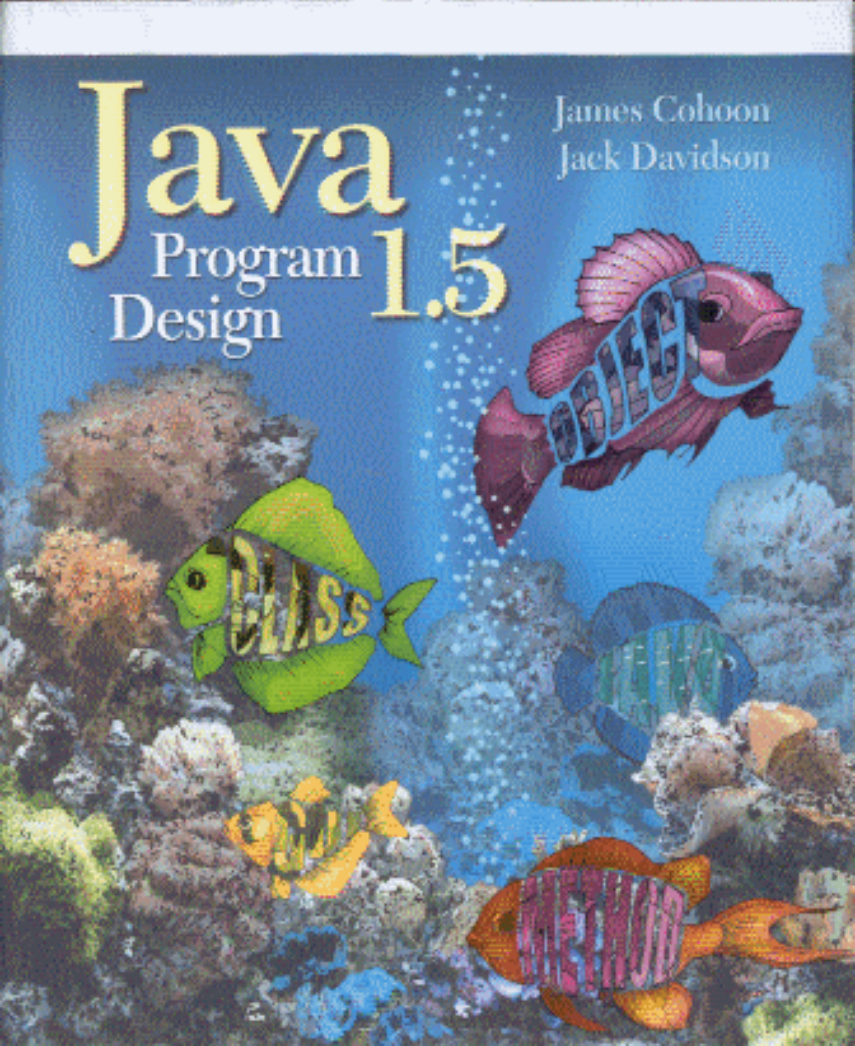


Java Program Design 1.5

James Cohoon
Jack Davidson



Detailed table of contents

CHAPTER 1: BACKGROUND, 1

- 1.1 Introduction, 2
- 1.2 Computer organization, 3
 - 1.2.1 Computing units of measure, 5
 - 1.2.2 Buying a personal computer, 6
- 1.3 Internet computing, 10
 - 1.3.1 Networks, 11
- 1.4 Software and Java, 14
 - 1.4.1 Programming languages, 14
 - 1.4.2 Running a Java program, 15
- 1.5 Engineering software, 18
 - 1.5.1 Software engineering principles, 20
- 1.6 Object-oriented design, 23
- 1.7 Problem solving, 26
 - 1.7.1 Ask questions as needed, 29
 - 1.7.2 Find out as much as you can, 30
 - 1.7.3 Break complex problems into subproblems, 31
 - 1.7.4 Reuse and expect future reuse, 32
 - 1.7.5 Further reading, 33
- 1.8 Review, 34
- 1.9 Self-test, 35
- 1.10 Exercises, 35
- 1.11 Self-test answers, 38

CHAPTER 2: JAVA BASICS, 39

- 2.1 A first program, 40
 - 2.1.1 Commenting and whitespace, 41
 - 2.1.2 Classes, keywords, and identifiers, 43
 - 2.1.3 Methods, 44
 - 2.1.4 Selecting methods print() and println(), 46
 - 2.1.5 Escape sequences, 49
- 2.2 Simple computations, 52
 - 2.2.1 Constants, 54
 - 2.2.2 Variables, 56
 - 2.2.3 Operations, 57
- 2.3 Primitive types, 61
 - 2.3.1 int type, 61
 - 2.3.2 char type, 63

- 2.3.3 double type, 65
- 2.4 Expressions, 67
 - 2.4.1 Unary and binary operators, 68
 - 2.4.2 Precedence, 69
 - 2.4.3 Widening and narrowing operand conversion, 71
 - 2.4.4 Overflow, underflow, and exceptions, 73
- 2.5 Interactive programs, 73
- 2.6 Primitive variable assignment, 78
 - 2.6.1 Swapping, 81
 - 2.6.2 Assignment precedence and associativity, 82
 - 2.6.3 Increment and decrement, 83
- 2.7 Case study — averaging five numbers, 86
- 2.8 Review, 89
- 2.9 Self-test, 91
- 2.10 Exercises, 93
- 2.11 Programming project — you, 98
- 2.12 Programming project — training zone, 101
- 2.13 Self-test answers, 103

CHAPTER 3: USING OBJECTS, 105

- 3.1 Classes, 106
- 3.2 Objects and variables, 107
 - 3.2.1 Initialization, 108
 - 3.2.2 Null and uninitialized references, 110
- 3.3 Assignment, 111
- 3.4 Final variables, 113
- 3.5 String operations and methods, 115
- 3.6 Case study — date translation, 123
- 3.7 Review, 129
- 3.8 Self-test, 130
- 3.9 Exercises, 130
- 3.10 Programming project — harvester, 134
- 3.11 Self-test answers, 136

CHAPTER 4: BEING CLASSY, 139

- 4.1 Preparation, 140
- 4.2 A very simple class, 142
 - 4.2.1 Instance variables and attributes, 144
 - 4.2.2 A default constructor, 146
 - 4.2.3 An instance method, 147
 - 4.2.4 Usage, 151
- 4.3 Methods with parameters and return values, 154
- 4.4 Summary, 166
- 4.5 Review, 167
- 4.6 Self-test, 169
- 4.7 Exercises, 169
- 4.8 Programming project — rationality, 172

4.9 Self-test answers, 175

CHAPTER 5: DECISIONS, 179

- 5.1 Boolean algebra and truth tables, 180
 - 5.1.1 Logical expressions, 181
- 5.2 Boolean type, 182
 - 5.2.1 Boolean equality and ordering operators, 183
 - 5.2.2 Operator precedence revisited, 186
- 5.3 If statement, 187
 - 5.3.1 Avoiding gotchas, 191
- 5.4 If-else statement, 192
- 5.5 Nested constructs, 195
- 5.6 If-else-if construct, 198
- 5.7 Testing objects for equality, 202
- 5.8 Switch statement, 206
- 5.9 Case study — checksum validation, 213
- 5.10 Case study — triangles, 217
- 5.11 Review, 226
- 5.12 Self-test, 228
- 5.13 Exercises, 230
- 5.14 Programming project — medical assistant, 235
- 5.15 Self-test answers, 239

CHAPTER 6: ITERATION, 241

- 6.1 While statement, 242
- 6.2 For statement, 252
 - 6.2.1 Index variable scope, 256
 - 6.2.2 Computing the number of combinations, 256
- 6.3 Do-while statement, 258
- 6.4 Nested loops, 262
- 6.5 Simple file processing, 265
- 6.6 Case study — data set analysis, 272
- 6.7 Review, 280
- 6.8 Self-test, 281
- 6.9 Exercises, 282
- 6.10 Programming project — four hobo problem, 285
- 6.11 Self-test answers, 287

GRAPHICS INTERLUDE 1: GUI-BASED PROGRAMMING, 289

- G1.1 GUI and event-driven programming, 290
- G1.2 Windchill calculator, 293
 - G1.2.1 Class constants and instance variables, 298
 - G1.2.2 GUI construction, 301
 - G1.2.3 Event handling and actionPerformed(), 304
 - G1.2.4 Method main(), 306
- G1.3 Review, 308
- G1.4 Self-test, 309

- G1.5 Exercises, 310
- G1.6 Programming project — training zones, 311
- G1.7 Self-test answers, 313

CHAPTER 7: PROGRAMMING WITH METHODS AND CLASSES, 315

- 7.1 Modifier static, 316
- 7.2 Parameter passing, 322
- 7.3 This, 331
- 7.4 Inherited methods and overriding, 334
- 7.5 Scope and name reuse, 340
 - 7.5.1 Local scope rules, 340
 - 7.5.2 Name reuse, 341
- 7.6 Overloading, 344
- 7.7 Illustrations, 351
- 7.8 Generic classes, 358
- 7.9 Review, 361
- 7.10 Self-test, 363
- 7.11 Exercises, 365
- 7.12 Programming project — automobile financing, 373
- 7.13 Self-test answers, 377

CHAPTER 8: ARRAYS AND COLLECTIONS, 381

- 8.1 Basic list requirements, 382
- 8.2 One-dimensional arrays, 382
 - 8.2.1 Definitions, 383
 - 8.2.2 Element access, 385
 - 8.2.3 Explicit initialization, 388
 - 8.2.4 Constant arrays, 389
 - 8.2.5 Members, 390
- 8.3 Iterator for loop, 393
- 8.4 Simple array processing, 396
 - 8.4.1 Extraction, 396
 - 8.4.2 Searching for a key value, 398
 - 8.4.3 Searching for the minimum value, 400
- 8.5 Arrays and methods, 401
 - 8.5.1 Sequential and binary search, 402
 - 8.5.2 Zeroing, 405
 - 8.5.3 Display, 408
 - 8.5.4 Extraction and reversal, 411
 - 8.5.5 Increasing representation capacity, 411
- 8.6 Sorting, 413
 - 8.6.1 Method selectionSort(), 414
 - 8.6.2 Quality of selectionSort(), 416
- 8.7 Command-line parameters, 417
- 8.8 Multidimensional arrays, 419
 - 8.8.1 Matrices, 422
- 8.9 Collections framework, 424

- 8.10 ArrayList<T>, 425
- 8.11 Collections algorithms, 430
- 8.12 Case study — pie charts, 434
- 8.13 Review, 443
- 8.14 Self-test, 445
- 8.15 Exercises, 446
- 8.16 Programming project — matrices, 452
- 8.17 Self-test answers, 453

CHAPTER 9: INHERITANCE AND POLYMORPHISM, 459

- 9.1 Object-oriented design, 460
 - 9.1.1 ThreeDimensionalPoint, 462
 - 9.1.2 ColoredPoint, 469
- 9.2 Polymorphism, 474
- 9.3 Inheritance nuances, 476
 - 9.3.1 Controlling access, 477
 - 9.3.2 Data fields, 480
 - 9.3.3 Typing, 483
 - 9.3.4 Late binding, 484
 - 9.3.5 Finality, 486
- 9.4 Abstract base classes, 487
- 9.5 Interfaces, 491
- 9.6 Case study — preparing the aquarium, 496
- 9.7 Review, 507
- 9.8 Self-test, 509
- 9.9 Exercises, 511
- 9.10 Programming project — change maker, 515
- 9.11 Self-test answers, 518

GRAPHICS INTERLUDE 2: GUI-BASED PROGRAMMING, 519

- G2.1 Case study — personality typing, 520
 - G2.1.1 Background, 520
- G2.2 Programming project — Smiley guessing game, 535

CHAPTER 10: EXCEPTIONS, 551

- 10.1 Exception handling, 552
- 10.2 Finally and the command type, 561
- 10.3 Creating and throwing exceptions, 563
- 10.4 Review, 568
- 10.5 Self-test, 569
- 10.6 Exercises, 569
- 10.7 Programming project — a second look, 570
- 10.8 Self-test answers, 570

CHAPTER 11: RECURSIVE PROBLEM SOLVING, 573

- 11.1 Recursive methods, 574
 - 11.1.1 Fibonacci numbers and squares, 577
- 11.2 Case study — recursive binary search, 581

- 11.3 Method mergeSort(), 587
- 11.4 How fast can we sort?, 592
- 11.5 Recursion versus iteration, 593
- 11.6 Case study — string permutation, 595
- 11.7 Review, 601
- 11.8 Self-test, 601
- 11.9 Exercises, 602
- 11.10 Programming project — Sierpinski fractal, 605
- 11.11 Self-test answers, 608

CHAPTER 12: THREADS, 609

- 12.1 Scheduling, 610
 - 12.1.1 Running after a delay, 612
 - 12.1.2 Running repeatedly, 615
 - 12.1.3 Running at a chosen time, 619
- 12.2 Sleeping, 624
- 12.3 Case study — animation, 626
- 12.4 Case study — swimming fish, 632
- 12.5 Review, 639
- 12.6 Self-test, 640
- 12.7 Exercises, 640
- 12.8 Programming project — better fish, 642
- 12.9 Self-test answers, 644

CHAPTER 13: TESTING AND DEBUGGING, 645

- 13.1 Testing, 646
 - 13.1.1 Testing — an example, 647
 - 13.1.2 Testing fundamentals, 656
 - 13.1.3 Reviews and inspections, 659
 - 13.1.4 Black-box and white-box testing, 662
 - 13.1.5 Integration and system testing, 666
- 13.2 Debugging, 667
 - 13.2.1 Scientific method, 667
 - 13.2.2 Debugging tips and techniques, 671
- 13.3 Review, 675
- 13.4 References, 675
- 13.5 Self-test, 676
- 13.6 Exercises, 676
- 13.7 Programming project — getList(), 677
- 13.8 Self-test answers, 679

APPENDIX A: TABLES AND OPERATORS, 681

- A.1 Unicode character set, 682
- A.2 Reserved words, 683
- A.3 Operators and precedence, 683

APPENDIX B: NUMBER REPRESENTATION, 687

- B.1 Binary number representation, 688

B.2 Two's-complement representation, 689

APPENDIX C: FORMATTED I/O, 691

C.1 Introduction, 692

C.2 Format String Syntax, 692

APPENDIX D: APPLETS, 697

D.1 A simple applet, 698

D.2 Applet methods, 700

D.2.1 Method `init()`, 700

D.2.2 Method `start()`, 701

D.2.3 Method `stop()`, 701

D.2.4 Method `destroy()`, 701

D.2.5 Method `paint()`, 701

D.3 Applets and threads, 705

D.4 Applet security, 707

D.5 Summary, 708

APPENDIX E: STANDARD JAVA PACKAGES, 709

INDEX, 903