

# Java

## Program Design

James P. Cohoon

*University of Virginia*

Jack W. Davidson

*University of Virginia*

LIBRARY OF THE ASIA FOUNDATION  
NOT FOR RE-SALE

QUÀ TẶNG CỦA QUỸ CHÂU Á  
KHÔNG ĐƯỢC BÁN LẠI

**Mc  
Graw  
Hill**

**Higher Education**

Berlin • Burr Ridge, IL • Dubuque, IA • Madison, WI • New York • San Francisco • St. Louis  
Bangkok • Bogotá • Caracas • Kuala Lumpur • London • Madrid • Mexico City  
Milan • Montreal • New Delhi • Santiago • Seoul • Singapore • Sydney • Taipei • Tokyo

# Detailed table of contents

## CHAPTER 1: BACKGROUND, I

- 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, 79
  - 2.6.1 Swapping, 81
  - 2.6.2 Assignment precedence and associativity, 83
  - 2.6.3 Increment and decrement, 85
- 2.7 Case study — averaging five numbers, 87
- 2.8 Review, 90
- 2.9 Self-test, 93
- 2.10 Exercises, 94
- 2.11 Programming project — you, 100
- 2.12 Programming project — training zone, 102
- 2.13 Self-test answers, 105

### **CHAPTER 3: USING OBJECTS, 107**

- 3.1 Classes, 108
- 3.2 Objects and variables, 108
  - 3.2.1 Initialization, 110
  - 3.2.2 Null and uninitialized references, 112
- 3.3 Assignment, 113
- 3.4 Final variables, 114
- 3.5 String operations and methods, 116
- 3.6 Case study — date translation, 124
- 3.7 Review, 130
- 3.8 Self-test, 131
- 3.9 Exercises, 131
- 3.10 Programming project — harvester, 135
- 3.11 Self-test answers, 137

### **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, 145
  - 4.2.3 An instance method, 147
  - 4.2.4 Usage, 151
- 4.3 Methods with parameters and return values, 154
- 4.4 Summary, 165
- 4.5 Review, 166
- 4.6 Self-test, 168
- 4.7 Exercises, 168
- 4.8 Programming project — rationality, 171



## 4.9 Self-test answers, 174

**CHAPTER 5: DECISIONS, 177**

- 5.1 Boolean algebra and truth tables, 178
  - 5.1.1 Logical expressions, 179
- 5.2 Boolean type, 180
  - 5.2.1 Boolean equality and ordering operators, 181
  - 5.2.2 Operator precedence revisited, 184
- 5.3 If statement, 185
  - 5.3.1 Avoiding gotchas, 189
- 5.4 If-else statement, 190
- 5.5 Nested constructs, 193
- 5.6 If-else-if construct, 195
- 5.7 Testing objects for equality, 200
- 5.8 Switch statement, 204
- 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, 266
- 6.6 Case study — data set analysis, 273
- 6.7 Review, 281
- 6.8 Self-test, 282
- 6.9 Exercises, 283
- 6.10 Programming project — four hobo problem, 287
- 6.11 Self-test answers, 288

**GRAPHICS INTERLUDE I: GUI-BASED PROGRAMMING, 291**

- 6.12 GUI and event-driven programming, 292
- 6.13 Windchill calculator, 294
  - 6.13.1 Class constants and instance variables, 299
  - 6.13.2 GUI construction, 303
  - 6.13.3 Event handling and actionPerformed(), 306
  - 6.13.4 Method main(), 308
- 6.14 Review, 310
- 6.15 Self-test, 311

- 6.16 Exercises, 312
- 6.17 Programming project — training zones, 313
- 6.18 Self-test answers, 315

## **CHAPTER 7: PROGRAMMING WITH METHODS AND CLASSES, 317**

- 7.1 Modifier static, 318
- 7.2 Parameter passing, 325
- 7.3 This, 334
- 7.4 Inherited methods and overriding, 338
- 7.5 Scope and reuse, 343
  - 7.5.1 Local scope rules, 343
  - 7.5.2 Name reuse, 344
- 7.6 Overloading, 347
- 7.7 Illustrations, 354
- 7.8 Review, 362
- 7.9 Self-test, 364
- 7.10 Exercises, 366
- 7.11 Programming project — automobile financing, 374
- 7.12 Self-test answers, 378

## **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 Simple array processing, 393
  - 8.3.1 Extraction, 393
  - 8.3.2 Searching for a key value, 396
  - 8.3.3 Searching for the minimum value, 398
- 8.4 Arrays and methods, 399
  - 8.4.1 Sequential and binary search, 400
  - 8.4.2 Zeroing, 403
  - 8.4.3 Display, 406
  - 8.4.4 Extraction and reversal, 406
- 8.5 Increasing representation capacity, 409
- 8.6 Sorting, 410
  - 8.6.1 Method selectionSort(), 411
  - 8.6.2 Quality of selectionSort(), 413
- 8.7 Command-line parameters, 414
- 8.8 Multidimensional arrays, 415
  - 8.8.1 Matrices, 419
- 8.9 Collections framework, 420
- 8.10 ArrayList, 421
- 8.11 Collections algorithms, 427

- 8.12 Case study — pie charts, 431
- 8.13 Review, 439
- 8.14 Self-test, 441
- 8.15 Exercises, 442
- 8.16 Programming project — matrices, 448
- 8.17 Self-test answers, 449

## **CHAPTER 9: INHERITANCE AND POLYMORPHISM, 455**

- 9.1 Object-oriented design, 456
  - 9.1.1 ThreeDimensionalPoint, 458
  - 9.1.2 ColoredPoint, 465
- 9.2 Polymorphism, 469
- 9.3 Inheritance nuances, 471
  - 9.3.1 Controlling access, 472
  - 9.3.2 Data fields, 475
  - 9.3.3 Typing, 478
  - 9.3.4 Late binding, 480
  - 9.3.5 Finality, 481
- 9.4 Case study — extending BufferedReader, 483
- 9.5 Abstract base classes, 487
- 9.6 Interfaces, 491
- 9.7 Preparing the aquarium, 495
- 9.8 Review, 507
- 9.9 Self-test, 509
- 9.10 Exercises, 511
- 9.11 Programming project — change maker, 515
- 9.12 Self-test answers, 518

## **GRAPHICS INTERLUDE II: GUI-BASED PROGRAMMING, 519**

- 9.13 Case study — personality typing, 520
  - 9.13.1 Background, 520
- 9.14 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 Case study — value-by-value extraction, 569
- 10.5 Review, 580
- 10.6 Self-test, 581
- 10.7 Exercises, 581
- 10.8 Programming project — a second look, 582
- 10.9 Self-test answers, 582

## **CHAPTER 11: RECURSIVE PROBLEM SOLVING, 587**

- 11.1 Recursive methods, 588
- 11.2 Fibonacci numbers and squares, 591
- 11.3 Case study — recursive binary search, 595