INTRODUCTION TO

PROGRAMMING AND OBJECT ORIENTED DESIGN USING JAVA

THIRD EDITION

JAIME NINO
FREDERICK A. HOSCH

CONTENTS

chapter 0 Introduction to object-oriented software design 1

0.1 What is a software system? 3
- 0.1.1 Dealing with complexity: composition and abstraction 3
- 0.1.2 Two aspects of a system: data and functionality 6

0.2 Object-oriented systems 8

0.3 Summary 10

chapter 1 Data abstraction: introductory concepts 15

1.1 Objects 16
- 1.1.1 Data and state 16
- 1.1.2 Functionality: queries and commands 19
- 1.1.3 Some cautions 21

1.2 Values and types 22
- 1.2.1 Values and types in Java 23
- 1.2.2 Types and variables 24

1.3 Classes 27

1.4 An example 27

1.5 Reference values: objects as properties of objects 33

1.6 Overview of a complete system 38
- 1.6.1 An example: testing a *Counter* 40

1.7 Java in detail: identifiers and literals 44
- 1.7.1 Identifiers 44
- 1.7.2 Literals 46

1.8 Java in detail: lexical structure 49

1.9 Objects that "wrap" primitive values 51

1.10 Java in detail: primitive types 52

1.11 Java in detail: literals, the rest of the story 54

1.12 Summary 55

chapter 2 Defining a simple class 61

2.1 Object interaction: clients and servers 62

3.7.1 *Static* methods and the class *Math* 165
3.7.2 *Final* features 167
3.7.3 Importing static features. 169
3.8 Summary 169

chapter 4 Conditions 175

4.1 Conditional statements. 176
4.1.1 The *if-then* statement 179
4.1.2 The *if-then-else* statement 180
4.1.3 Compound statements 186
4.2 Boolean expressions 189
4.3 Handling multiple cases. 190
4.3.1 Dangling *else* 197
4.4 Example: the class *Date* 198
4.5 Example: a combination lock 202
4.6 Java in detail: Boolean expressions. 213
4.6.1 Reference equality and object equality 214
4.6.2 Boolean operators 215
4.7 Java in detail: logical operators and conditional expressions . . . 218
4.8 Java in detail: the *switch* statement. 219
4.9 Summary 223

chapter 5 Programming by contract 233

5.1 Programming by contract 234
5.1.1 Verifying preconditions: the *assert* statement 236
5.2 Further examples. 240
5.3 Preconditions and postconditions: a summary of use 245
5.4 Enumeration classes 248
5.5 Summary 251

chapter 6 Testing 259

6.1 Functional testing and unit testing 260
6.2 Developing a test plan. 262
6.3 The JUnit testing framework 264
6.4 An example 267
6.5 Do we need to test everything?. 276
6.6 Summary 278

chapter 7 **Building a text-based user interface** **283**

7.1 Relating the user interface and the model 284

7.2 Stream-based i/o 286

7.2.1 Writing standard output and reading standard input 286

7.3 An example: building an interface for *Rectangle* 292

7.3.1 Repeating actions: the *while* statement 294

7.3.2 Completing the *createRectangle* method 296

7.3.3 Finishing the user interface 298

7.3.4 Loop structure 304

7.4 A second example: using composition 305

7.4.1 Constructing a class by composition 306

7.4.2 Implementing the user interface 309

7.5 Java in detail: the *do* statement 312

7.6 Summary 314

chapter 8 **The software life cycle: building a complete system** **323**

8.1 Software life cycle 324

8.2 Design of a system 325

8.2.1 Functional specification 326

8.2.2 Preliminary design 327

8.2.3 Relations between objects 332

8.2.4 Class specification 334

8.3 Implementing the system 339

8.3.1 The top-level 340

8.3.2 The class *Pile* 340

8.3.3 The class *Player* 340

8.3.4 The class *Game* 345

8.3.5 The class *NimTUI* 346

8.4 Summary 352

chapter 9 **Specifying clients: interfaces** **359**

9.1 Modeling alternative implementations 360

9.2 Interfaces 362

9.2.1 Interface definition 362

9.2.2 Interface implementation 364

9.2.3 Subtyping 366

9.2.4 Putting it together 372

9.3 Clients and interfaces 374

9.4 Multiple inheritance and interface extension 377

9.5 Modifying the system: user vs. computer 379

9.5.1 Player classes 379
9.5.2 The user interface 381
9.5.3 User interface—model interaction 382
9.6 Reclaiming lost ground: the strategy pattern 388
9.7 Java in detail: casting and *instanceof* 390
9.8 Summary 396

chapter 10 Class extension and inheritance 409
10.1 Abstraction and classes 410
10.1.1 Class inheritance is single inheritance 411
10.2 Extension and inheritance 412
10.3 Constructors and subclasses 417
10.4 Overloading, overriding, and polymorphism 421
10.4.1 Method overloading 421
10.4.2 Method overriding 426
10.4.3 Overriding and contracts 432
10.5 Java in detail: feature accessibility 434
10.5.1 Protected features 437
10.5.2 Package private features 440
10.5.3 Inner classes 441
10.6 Java in detail: scoping rules 444
10.7 Summary 448

chapter 11 Modeling with abstraction 461
11.1 Abstract classes 462
11.1.1 Interfaces, abstract classes, and concrete classes 463
11.2 Specifying a class for extension 466
11.2.1 Planning for extension 469
11.3 Composition revisited 475
11.3.1 Extension and composition 476
11.3.2 Extension, composition, and reuse 479
11.3.3 Extension, composition, and modifying functionality 481
11.4 Extension and state 483
11.4.1 State as an object 484
11.5 Summary 486

chapter 12 Lists 493
12.1 Containers 494
12.2 Lists 495

12.2.1 Structuring lists 496
12.2.2 Using generics to capture "list-ness" 501
12.3 List specification 504
12.4 Iteration 512
12.4.1 *while* loops and lists 513
12.5 Examples 515
12.5.1 Summing items on a list 515
12.5.2 Summing selected elements of a list 518
12.5.3 Finding the minimum 518
12.5.4 Determining if an object is on a *List*: searching the *List* 521
12.5.5 Removing duplicates 523
12.6 Loop structure: a summary 527
12.7 Java in detail: the *for* statement 528
12.8 What does "equal" mean? 531
12.8.1 Overriding equals 532
12.8.2 Problems when overriding equals 533
12.9 Implementing the interface *List* 540
12.9.1 The class *AbstractList* 540
12.9.2 The class *DefaultList* 541
12.10 Generic type checking is done at compile time 544
12.11 Summary 545

chapter 13 Arrays 555

13.1 Arrays 556
13.1.1 Creating arrays 558
13.1.2 Accessing array components 560
13.2 Examples 560
13.3 Constant arrays 568
13.4 Java in detail: static initializers 570
13.5 Arrays with array components 572
13.5.1 Creating multidimensional arrays 573
13.5.2 Accessing multidimensional array components 575
13.5.3 An example: tic-tac-toe 577
13.6 Types and arrays 580
13.6.1 Component types 580
13.6.2 Subtyping 581
13.7 An example: modeling lists with arrays 583
13.8 Java in detail: command line arguments 587
13.9 Java in detail: variable arity parameters 588

13.10 Summary 589

chapter 14 Sorting and searching 597

14.1 Ordering lists 598

14.2 Two simple sorts 600

14.2.1 Selection sort 600

14.2.2 Analysis of selection sort 605

14.2.3 Bubble sort 606

14.2.4 Generalizing the sort methods 611

14.2.5 The interfaces *java.util.Comparator* and *java.lang.Comparable* . 617

14.3 Ordered lists 621

14.4 Binary search 623

14.4.1 Completing the search 628

14.4.2 Sequential search and binary search 630

14.5 Verifying correctness: using a loop invariant 631

14.6 Summary 635

chapter 15 Failures and exceptions 643

15.1 Failures 644

15.2 The Java exception mechanism 644

15.2.1 Exceptions as objects 645

15.2.2 Catching exceptions 646

15.2.3 Propagated exceptions 649

15.2.4 Checked and unchecked exceptions 649

15.3 Dealing with failure: using exceptions 651

15.3.1 Dealing with exceptions 653

15.3.2 Application defined exceptions 656

15.3.3 Dealing with logical errors 657

15.4 Exceptions and contracts 658

15.5 The *CloneNotSupportedException* and the interface *Cloneable* . . 661

15.6 Summary 665

chapter 16 Stream i/o 671

16.1 Data Streams 672

16.2 The java.io library 672

16.3 Input streams 673

16.3.1 Input byte streams 673

16.3.2 Input character streams 679

16.3.3 Input examples 682

16.4 Output streams 690
16.4.1 Output byte streams 691
16.4.2 Output character streams 693
16.5 The class *File* 697
16.6 Summary 698

chapter 17 Building a graphical user interface 705
17.1 Event-driven interfaces 706
17.2 An introduction to Swing 707
17.2.1 Components 707
17.2.2 Containers 711
17.3 Creating a display 715
17.3.1 The top-level frame 715
17.3.2 The event dispatching thread 716
17.3.3 Adding components: layout 718
17.4 Events: programming the user interface 722
17.4.1 An example 724
17.5 A more complex example 735
17.6 Menus, dialogs, fonts, and graphics 745
17.6.1 Menus and menu bars 745
17.6.2 Basic dialogs 746
17.6.3 Fonts 752
17.6.4 Graphics 753
17.7 Some class features 755
17.7.1 *Component* 755
17.7.2 *Container* 756
17.7.3 *Window* 756
17.7.4 *Frame* 757
17.7.5 *JComponent* 757
17.7.6 *JFrame* 758
17.8 Summary 758

chapter 18 Integrating user interface and model: the Model-View-Controller pattern 765
18.1 Model-View-Controller 766
18.2 Implementing MVC in Java 767
18.2.1 The model 767
18.2.2 The class *Observable* and interface *Observer* 769
18.2.3 An *Observer* 771
18.2.4 Observer, Observable, and subtyping 773
18.2.5 A simple view and controller 773

18.2.6 A graphic view 782
18.2.7 A logger as a view 784

18.3 Adding a graphical interface to the nim game 788

18.4 The MVC pattern and Swing components 793

18.5 Summary 796

chapter 19 Recursion 801

19.1 Recursion and iteration 802
19.1.1 Iteration 802
19.1.2 Recursion 802
19.1.3 Some simple examples 803

19.2 Example: the tower puzzle 820

19.3 Quicksort 824

19.4 An inefficient algorithm 831

19.5 Indirect recursion 832

19.6 Backtracking 833

19.7 Object recursion 837

19.8 Summary 848

chapter 20 Generic structures 855

20.1 Multiple and bounded parameters 856

20.2 Wildcard types 860
20.2.1 *Generics and subtyping, revisited* 860
20.2.2 Wildcards 862
20.2.3 Using wildcards 868
20.2.4 Wildcards and generic methods 871

20.3 Erasure 876
20.3.1 Restrictions due to erasure 879

20.4 Summary 882

chapter 21 Implementing lists: linked implementations 889

21.1 A linked *List* implementation 889
21.1.1 Implementing *LinkedList* methods 892

21.2 Linked list variations 897

21.3 Doubly linked lists 900

21.4 Limitations of linked structures 902

21.5 Dynamic storage allocation 904

21.6 Summary 905

chapter 22 **Iterators** **913**

22.1 Iterators 913

22.1.1 Iterator classes 916

22.1.2 Creating an iterator. 921

22.2 *List<Element>* methods with iterators as arguments 924

22.2.1 Improving *LinkedList<Element>.LocalIterator* 926

22.2.2 *Iterator* extensions 927

22.2.3 Iterators and list modification 928

22.2.4 Internal iterators: *forEachDo* 930

22.3 Comparing implementations 932

22.4 The java.util *Collection<Element>* hierarchy 932

22.4.1 Implementing *for each* 935

22.5 Summary 936

supplement a **Systems and software** **941**

a.1 A model of a computer system 941

a.2 Software tools 944

supplement b **Programming errors** **949**

b.1 Errors in the programming process 949

supplement c **Applets** **951**

c.1 Applets 951

c.1.1 The class *JApplet* 952

c.1.2 Applets as applications. 955

c.1.3 An example: a simple clock 956

c.1.4 An example: an animated box 958

supplement d **Enumeration types: the rest of the story** **963**

d.1 Enumeration types 963

appendix i **Compiling, executing, and documenting** **969**

i.1 Compiling and running an application 969

i.1.1 Compilation units. 969

i.1.2 Package name and directory hierarchy 970

i.1.3 The environment variable *CLASSPATH*. 970

i.1.4 Compiling and running 971

i.2 Generating documentation 972

appendix ii DrJava 975

appendix iii Controls and basic Latin: the first 128 Unicode characters 977

glossary 979

references 997

index 999