Bertrand Meyer

# Object-oriented Software Construction

PRENTICE HALL INTERNATIONAL SERIES IN COMPUTER SCIENCE

C.A.R. HOARE SERIES EDITOR

2-005-514-1

2-005-514-1

# OBJECT-ORIENTED SOFTWARE CONSTRUCTION

Bertrand Meyer

Interactive Software Engineering,
Santa Barbara, California
and
Société des Outils du Logiciel,
Paris

PRENTICE HALL

NEW YORK LONDON TORONTO SYDNEY TOKYO SINGAPORE

# Contents

**Preface** xiii

**Syntax notation** xvii

**PART 1 ISSUES AND PRINCIPLES** 1

**Chapter 1 Aspects of software quality** 3

1.1 External and internal factors 3
1.2 External quality factors 4
1.3 About software maintenance 7
1.4 The key qualities 9
1.5 Key concepts 10
1.6 Bibliographical notes 10

**Chapter 2 Modularity** 11

2.1 Five criteria 12
2.2 Five principles 18
2.3 The open-closed principle 23
2.4 Key concepts 25
2.5 Bibliographical notes 26
Exercises 26

**Chapter 3 Approaches to reusability** 27
3.1 Repetition in programming 27
3.2 Simple approaches 30
3.3 Five requirements on module structures 31
3.4 Routines 35
3.5 Packages 36
3.6 Overloading and genericity 37
3.7 Key concepts 39
3.8 Bibliographical notes 40

**Chapter 4 The road to object-orientedness** 41
4.1 Process and data 41
4.2 Functions, data and continuity 42
4.3 The top-down functional approach 43
4.4 Why use the data? 49
4.5 Object-oriented design 50
4.6 Finding the objects 51
4.7 Describing objects: abstract data types 52
4.8 A precise definition 59
4.9 Seven steps towards object-based happiness 60
4.10 Key concepts 63
4.11 Bibliographical notes 63
Exercises 64

**PART 2 TECHNIQUES OF OBJECT-ORIENTED DESIGN AND PROGRAMMING** 65

**Chapter 5 Basic Elements of Eiffel programming** 67
5.1 Objects 67
5.2 A first view of classes 71
5.3 Using classes 73
5.4 Routines 79
5.5 Reference and value semantics 86
5.6 From classes to systems 90
5.7 Classes vs. objects 94
5.8 Discussion 94
5.9 Key concepts 101
5.10 Syntactical summary 102

**Chapter 6 Genericity** 105
6.1 Parameterizing classes 105
6.2 Arrays 108
6.3 Discussion 109
6.4 Key concepts 110
6.5 Syntactical summary 110
6.6 Bibliographical notes 110

**Chapter 7 Systematic approaches to program construction** 111
7.1 The notion of assertion 112
7.2 Preconditions and postconditions 113
7.3 Contracting for software reliability 115
7.4 Class invariants and class correctness 123
7.5 Some theory 129
7.6 Representation invariants 131
7.7 Side-effects in functions 132
7.8 Other constructs involving assertions 140
7.9 Using assertions 143
7.10 Coping with failure: disciplined exceptions 144
7.11 Discussion 155
7.12 Key concepts 161
7.13 Syntactical summary 162
7.14 Bibliographical notes 163
Exercises 163

**Chapter 8 More aspects of Eiffel** 165
8.1 Style standards 165
8.2 Lexical conventions 168
8.3 External routines 169
8.4 Argument passing 170
8.5 Instructions 172
8.6 Expressions 176
8.7 Strings 179
8.8 Input and output 180
8.9 Key concepts 181
8.10 Syntactical summary 181

**Chapter 9 Designing class interfaces** 183
9.1 Lists and list elements 184
9.2 Objects as machines 191
9.3 Dealing with abnormal cases 199
9.4 Selective exports 203
9.5 Documenting a class 204
9.6 Discussion 210
9.7 Key concepts 214
9.8 Syntactical summary 215
9.9 Bibliographical notes 215
Exercises 215

**Chapter 10 Introduction to inheritance** 217
10.1 Polygons and rectangles 218
10.2 The meaning of inheritance 228
10.3 Deferred classes 234
10.4 Multiple inheritance 241
10.5 Discussion 250
10.6 Key concepts 251
10.7 Syntactical summary 252
10.8 Bibliographical notes 253
Exercises 253

**Chapter 11 More about inheritance** 255
11.1 Inheritance and assertions 255
11.2 Redefinition vs. renaming 259
11.3 The Eiffel type system 261
11.4 Declaration by association 266
11.5 Inheritance and information hiding 272
11.6 Repeated inheritance 274
11.7 Key concepts 279
11.8 Syntactical summary 280
11.9 Bibliographical note 280
Exercises 280

**Chapter 12 Object-oriented design: case studies** 281
12.1 Outline of a window system 281
12.2 Undoing and redoing 285
12.3 Full-screen entry systems 291
Exercises 304

**Chapter 13 Constants and shared objects** 305
13.1 Constants of simple types 306
13.2 Use of constants 306
13.3 Constants of class types 308
13.4 Constants of string type 314
13.5 Discussion 316
13.6 Key concepts 321
13.7 Syntactical summary 321
Exercises 322
13.4 Bibliographical notes 322

**Chapter 14 Techniques of object-oriented design** 323
14.1 Design philosophy 323
14.2 Finding the classes 326
14.3 Interface techniques 328
14.4 Inheritance techniques 329
14.5 Would you rather buy or inherit? 332
14.6 Bibliographical notes 334
Exercises 334

**Chapter 15 Implementation: the Eiffel programming environment** 335
15.1 The implementation 335
15.2 Compilation and configuration management 336
15.3 Generating C packages 341
15.4 Performance issues 343
15.5 Other aspects of the environment 345

**Chapter 16 Memory management** 353
16.1 What happens to objects 353
16.2 The casual approach 358
16.3 Reclaiming memory: the issues 359
16.4 Programmer-controlled deallocation 359
16.5 The self-management approach 360
16.6 Automatic storage management 365
16.7 The Eiffel approach 367
16.8 Key concepts 369
16.9 Bibliographical notes 370
Exercises 370

**PART 3 APPLYING OBJECT-ORIENTED TECHNIQUES IN OTHER ENVIRONMENTS** 373

**Chapter 17 Object-oriented programming in classical languages** 375
- 17.1 Levels of language support 375
- 17.2 Object-oriented programming in Pascal? 376
- 17.3 Fortran 376
- 17.4 Object-oriented programming and C 379
- 17.5 Bibliographical notes 383
- Exercises 383

**Chapter 18 Object-oriented programming and Ada** 385
- 18.1 Packages 386
- 18.2 A stack implementation 386
- 18.3 Hiding the representation: the private story 390
- 18.4 Exceptions 392
- 18.5 Tasks 396
- 18.6 Key concepts 397
- 18.7 Bibliographical notes 398
- Exercises 398

**Chapter 19 Genericity versus inheritance** 399
- 19.1 Genericity 400
- 19.2 Inheritance 406
- 19.3 Simulating inheritance with genericity 409
- 19.4 Simulating genericity with inheritance 410
- 19.5 Genericity and inheritance in Eiffel 418
- 19.6 Discussion 420
- 19.7 Key concepts 421
- 19.8 Bibliographical notes 421
- Exercises 422

**Chapter 20 Other object-oriented languages** 423
- 20.1 Simula 423
- 20.2 Smalltalk 437
- 20.3 C extensions 440
- 20.4 Lisp extensions 442
- 20.5 Other languages 443
- 20.6 Bibliographical notes 443
- Exercises 444

**Chapter 21 Further issues** 445
21.1 Implementing reusability 445
21.2 Persistency 446
21.3 Concurrency 447
21.4 Bibliographical notes 448

**PART 4 APPENDICES** 449

**Appendix A Extracts from the Eiffel library** 451
A.1 Arrays 451
A.2 General lists 453
A.3 Array lists 455
A.4 Linkable elements 455
A.5 Linked lists 463
A.6 Two-way lists 469
A.7 Trees and nodes 471

**Appendix B Eiffel: a quick overview** 475
B.1 Design principles 475
B.2 Classes 476
B.3 Assertions 479
B.4 Exceptions 480
B.5 Generic classes 482
B.6 Multiple inheritance 482
B.7 Polymorphism 483
B.8 Deferred classes 485
B.9 The implementation 487
B.10 The environment 488

**Appendix C Eiffel grammar** 489
C.1 Lexical conventions 489
C.2 Syntactical specification 490
C.3 Operator precedence 494

**Appendix D Reserved words and special symbols** 495
D.1 Reserved words 495
D.2 Special symbols 495

**Appendix E Input, output and strings** 497

E.1 Standard input and output 497

E.2 Files 498

E.3 Strings 501

**Appendix F Eiffel Syntax diagrams** 505

**Bibliography** 511

**Index** 521