

Contents

Foreword	xiii
Preface	xv
Acknowledgments	xix
1 Introduction	1
2 Creating and Destroying Objects	5
Item 1: Consider static factory methods instead of constructors . . .	5
Item 2: Consider a builder when faced with many constructor parameters	11
Item 3: Enforce the singleton property with a private constructor or an enum type	17
Item 4: Enforce noninstantiability with a private constructor . . .	19
Item 5: Avoid creating unnecessary objects	20
Item 6: Eliminate obsolete object references	24
Item 7: Avoid finalizers	27
3 Methods Common to All Objects	33
Item 8: Obey the general contract when overriding equals	33
Item 9: Always override hashCode when you override equals	45
Item 10: Always override toString	51
Item 11: Override clone judiciously	54
Item 12: Consider implementing Comparable	62

4 Classes and Interfaces.....67

Item 13: Minimize the accessibility of classes and members.....	67
Item 14: In public classes, use accessor methods, not public fields	71
Item 15: Minimize mutability.....	73
Item 16: Favor composition over inheritance.....	81
Item 17: Design and document for inheritance or else prohibit it ..	87
Item 18: Prefer interfaces to abstract classes	93
Item 19: Use interfaces only to define types.....	98
Item 20: Prefer class hierarchies to tagged classes.....	100
Item 21: Use function objects to represent strategies	103
Item 22: Favor static member classes over nonstatic	106

5 Generics109

Item 23: Don't use raw types in new code	109
Item 24: Eliminate unchecked warnings.....	116
Item 25: Prefer lists to arrays	119
Item 26: Favor generic types.....	124
Item 27: Favor generic methods	129
Item 28: Use bounded wildcards to increase API flexibility	134
Item 29: Consider typesafe heterogeneous containers	142

6 Enums and Annotations147

Item 30: Use enums instead of <code>int</code> constants.....	147
Item 31: Use instance fields instead of ordinals	158
Item 32: Use <code>EnumSet</code> instead of bit fields.....	159
Item 33: Use <code>EnumMap</code> instead of ordinal indexing.....	161
Item 34: Emulate extensible enums with interfaces	165
Item 35: Prefer annotations to naming patterns	169
Item 36: Consistently use the <code>Override</code> annotation.....	176
Item 37: Use marker interfaces to define types	179

7 Methods181

Item 38: Check parameters for validity	181
Item 39: Make defensive copies when needed	184
Item 40: Design method signatures carefully	189
Item 41: Use overloading judiciously.....	191

Item 42: Use varargs judiciously	197
Item 43: Return empty arrays or collections, not nulls	201
Item 44: Write doc comments for all exposed API elements	203

8 General Programming209

Item 45: Minimize the scope of local variables	209
Item 46: Prefer for-each loops to traditional for loops	212
Item 47: Know and use the libraries	215
Item 48: Avoid float and double if exact answers are required	218
Item 49: Prefer primitive types to boxed primitives	221
Item 50: Avoid strings where other types are more appropriate . .	224
Item 51: Beware the performance of string concatenation	227
Item 52: Refer to objects by their interfaces	228
Item 53: Prefer interfaces to reflection	230
Item 54: Use native methods judiciously	233
Item 55: Optimize judiciously	234
Item 56: Adhere to generally accepted naming conventions	237

9 Exceptions241

Item 57: Use exceptions only for exceptional conditions	241
Item 58: Use checked exceptions for recoverable conditions and runtime exceptions for programming errors	244
Item 59: Avoid unnecessary use of checked exceptions	246
Item 60: Favor the use of standard exceptions	248
Item 61: Throw exceptions appropriate to the abstraction	250
Item 62: Document all exceptions thrown by each method	252
Item 63: Include failure-capture information in detail messages	254
Item 64: Strive for failure atomicity	256
Item 65: Don't ignore exceptions	258

10 Concurrency259

Item 66: Synchronize access to shared mutable data	259
Item 67: Avoid excessive synchronization	265
Item 68: Prefer executors and tasks to threads	271
Item 69: Prefer concurrency utilities to wait and notify	273

Item 70: Document thread safety	278
Item 71: Use lazy initialization judiciously	282
Item 72: Don't depend on the thread scheduler	286
Item 73: Avoid thread groups	288
11 Serialization	289
Item 74: Implement Serializable judiciously	289
Item 75: Consider using a custom serialized form	295
Item 76: Write readObject methods defensively	302
Item 77: For instance control, prefer enum types to readResolve	308
Item 78: Consider serialization proxies instead of serialized instances	312
Appendix: Items Corresponding to First Edition	317
References	321
Index	327