

Table of Contents

Foreword.....	xi
Preface.....	xiii
1. Clean Code.....	1
1.1 What Is a Code Smell?	1
1.2 What Is Refactoring?	2
1.3 What Is a Recipe?	2
1.4 Why Clean Code?	2
1.5 Readability, Performance, or Both	3
1.6 Software Types	3
1.7 Machine-Generated Code	3
1.8 Naming Considerations Throughout the Book	4
1.9 Design Patterns	4
1.10 Programming Language Paradigms	4
1.11 Objects Versus Classes	5
1.12 Changeability	5
2. Setting Up the Axioms.....	7
2.0 Introduction	7
2.1 Why Is It a Model?	8
2.2 Why Is It Abstract?	8
2.3 Why Is It Programmable?	8
2.4 Why Is It Partial?	9
2.5 Why Is It Explanatory?	9
2.6 Why Is It About Reality?	10
2.7 Inferring the Rules	10
2.8 The One and Only Software Design Principle	10

3. Anemic Models.....	17
3.0 Introduction	17
3.1 Converting Anemic Objects to Rich Objects	18
3.2 Identifying the Essence of Your Objects	19
3.3 Removing Setters from Objects	21
3.4 Removing Anemic Code Generators	23
3.5 Removing Automatic Properties	25
3.6 Removing DTOs	26
3.7 Completing Empty Constructors	29
3.8 Removing Getters	30
3.9 Preventing Object Orgy	33
3.10 Removing Dynamic Properties	35
4. Primitive Obsession.....	37
4.0 Introduction	37
4.1 Creating Small Objects	38
4.2 Reifying Primitive Data	39
4.3 Reifying Associative Arrays	41
4.4 Removing String Abuses	43
4.5 Reifying Timestamps	44
4.6 Reifying Subsets as Objects	45
4.7 Reifying String Validations	46
4.8 Removing Unnecessary Properties	49
4.9 Creating Date Intervals	50
5. Mutability.....	53
5.0 Introduction	53
5.1 Changing var to const	55
5.2 Declaring Variables to Be Variable	56
5.3 Forbidding Changes in the Essence	58
5.4 Avoiding Mutable const Arrays	59
5.5 Removing Lazy Initialization	61
5.6 Freezing Mutable Constants	63
5.7 Removing Side Effects	65
5.8 Preventing Hoisting	66
6. Declarative Code.....	69
6.0 Introduction	69
6.1 Narrowing Reused Variables	70
6.2 Removing Empty Lines	71
6.3 Removing Versioned Methods	72
6.4 Removing Double Negatives	73

6.5 Changing Misplaced Responsibilities	74
6.6 Replacing Explicit Iterations	76
6.7 Documenting Design Decisions	77
6.8 Replacing Magic Numbers with Constants	77
6.9 Separating “What” and “How”	78
6.10 Documenting Regular Expressions	80
6.11 Rewriting Yoda Conditions	81
6.12 Removing Comedian Methods	81
6.13 Avoiding Callback Hell	82
6.14 Generating Good Error Messages	84
6.15 Avoiding Magic Corrections	86
7. Naming.....	89
7.0 Introduction	89
7.1 Expanding Abbreviations	89
7.2 Renaming and Breaking Helpers and Utils	91
7.3 Renaming MyObjects	94
7.4 Renaming Result Variables	95
7.5 Renaming Variables Named After Types	97
7.6 Renaming Long Names	98
7.7 Renaming Abstract Names	99
7.8 Correcting Spelling Mistakes	100
7.9 Removing Class Names from Attributes	100
7.10 Removing the First Letter from Classes and Interfaces	101
7.11 Renaming Basic / Do Functions	102
7.12 Converting Plural Classes to Singular	104
7.13 Removing “Collection” from Names	104
7.14 Removing “Impl” Prefix/Suffix from Class Names	105
7.15 Renaming Arguments According to Role	106
7.16 Removing Redundant Parameter Names	107
7.17 Removing Gratuitous Context from Names	108
7.18 Avoiding Data Naming	110
8. Comments.....	111
8.0 Introduction	111
8.1 Removing Commented Code	111
8.2 Removing Obsolete Comments	113
8.3 Removing Logical Comments	114
8.4 Removing Getter Comments	116
8.5 Converting Comments to Function Names	117
8.6 Removing Comments Inside Methods	118
8.7 Replacing Comments with Tests	120

9. Standards.....	123
9.0 Introduction	123
9.1 Following Code Standards	123
9.2 Standardizing Indentations	126
9.3 Unifying Case Conventions	127
9.4 Writing Code in English	128
9.5 Unifying Parameter Order	130
9.6 Fixing Broken Windows	131
10. Complexity.....	133
10.0 Introduction	133
10.1 Removing Repeated Code	133
10.2 Removing Settings/Configs and Feature Toggles	135
10.3 Changing State as Properties	137
10.4 Removing Cleverness from Code	139
10.5 Breaking Multiple Promises	141
10.6 Breaking Long Chains of Collaborations	142
10.7 Extracting a Method to an Object	143
10.8 Looking After Array Constructors	145
10.9 Removing Poltergeist Objects	147
11. Bloaters.....	149
11.0 Introduction	149
11.1 Breaking Too Long Methods	149
11.2 Reducing Excess Arguments	151
11.3 Reducing Excess Variables	152
11.4 Removing Excessive Parentheses	154
11.5 Removing Excess Methods	155
11.6 Breaking Too Many Attributes	157
11.7 Reducing Import Lists	158
11.8 Breaking “And” Functions	159
11.9 Breaking Fat Interfaces	161
12. YAGNI.....	163
12.0 Introduction	163
12.1 Removing Dead Code	163
12.2 Using Code Instead of Diagrams	165
12.3 Refactoring Classes with One Subclass	167
12.4 Removing One-Use Interfaces	168
12.5 Removing Design Pattern Abuses	170
12.6 Replacing Business Collections	170

13. Fail Fast.....	173
13.0 Introduction	173
13.1 Refactoring Reassignment of Variables	173
13.2 Enforcing Preconditions	175
13.3 Using Stricter Parameters	177
13.4 Removing Default from Switches	178
13.5 Avoiding Modifying Collections While Traversing	180
13.6 Redefining Hash and Equality	181
13.7 Refactoring Without Functional Changes	183
14. Ifs.....	185
14.0 Introduction	185
14.1 Replacing Accidental Ifs with Polymorphism	186
14.2 Renaming Flag Variables for Events	192
14.3 Reifying Boolean Variables	193
14.4 Replacing Switch/Case/Elseif Statements	195
14.5 Replacing Hardcoded If Conditions with Collections	197
14.6 Changing Boolean to Short-Circuit Conditions	198
14.7 Adding Implicit Else	199
14.8 Rewriting Conditional Arrow Code	200
14.9 Avoiding Short-Circuit Hacks	202
14.10 Rewriting Nested Arrow Code	203
14.11 Preventing Return Boolean Values for Condition Checks	205
14.12 Changing Comparison Against Booleans	207
14.13 Extracting from Long Ternaries	208
14.14 Converting Nonpolymorphic Functions to Polymorphic	209
14.15 Changing Equal Comparison	211
14.16 Reifying Hardcoded Business Conditions	212
14.17 Removing Gratuitous Booleans	213
14.18 Rewriting Nested Ternaries	214
15. Null.....	217
15.0 Introduction	217
15.1 Creating Null Objects	217
15.2 Removing Optional Chaining	220
15.3 Converting Optional Attributes to a Collection	223
15.4 Using Real Objects for Null	224
15.5 Representing Unknown Locations Without Using Null	227
16. Premature Optimization.....	231
16.0 Introduction	231
16.1 Avoiding IDs on Objects	232

16.2 Removing Premature Optimization	235
16.3 Removing Bitwise Premature Optimizations	236
16.4 Reducing Overgeneralization	237
16.5 Changing Structural Optimizations	238
16.6 Removing Anchor Boats	240
16.7 Extracting Caches from Domain Objects	241
16.8 Removing Callback Events Based on Implementation	243
16.9 Removing Queries from Constructors	244
16.10 Removing Code from Destructors	245
17. Coupling	249
17.0 Introduction	249
17.1 Making Hidden Assumptions Explicit	249
17.2 Replacing Singletons	251
17.3 Breaking God Objects	253
17.4 Breaking Divergent Change	256
17.5 Converting 9999 Special Flag Values to Normal	257
17.6 Removing Shotgun Surgery	259
17.7 Removing Optional Arguments	261
17.8 Preventing Feature Envy	263
17.9 Removing the Middleman	264
17.10 Moving Default Arguments to the End	266
17.11 Avoiding the Ripple Effect	267
17.12 Removing Accidental Methods on Business Objects	269
17.13 Removing Business Code from the User Interface	270
17.14 Changing Coupling to Classes	273
17.15 Refactoring Data Clumps	275
17.16 Breaking Inappropriate Intimacy	277
17.17 Converting Fungible Objects	278
18. Globals	281
18.0 Introduction	281
18.1 Reifying Global Functions	281
18.2 Reifying Static Functions	282
18.3 Replacing GoTo with Structured Code	284
18.4 Removing Global Classes	285
18.5 Changing Global Date Creation	287
19. Hierarchies	289
19.0 Introduction	289
19.1 Breaking Deep Inheritance	289
19.2 Breaking Yo-Yo Hierarchies	292

19.3 Breaking Subclassification for Code Reuse	293
19.4 Replacing “is-a” Relationship with Behavior	295
19.5 Removing Nested Classes	297
19.6 Renaming Isolated Classes	299
19.7 Making Concrete Classes Final	300
19.8 Defining Class Inheritance Explicitly	301
19.9 Migrating Empty Classes	303
19.10 Delaying Premature Classification	304
19.11 Removing Protected Attributes	306
19.12 Completing Empty Implementations	308
20. Testing.....	311
20.0 Introduction	311
20.1 Testing Private Methods	312
20.2 Adding Descriptions to Assertions	313
20.3 Migrating assertTrue to Specific Assertions	315
20.4 Replacing Mocks with Real Objects	316
20.5 Refining Generic Assertions	318
20.6 Removing Flaky Tests	319
20.7 Changing Float Number Assertions	321
20.8 Changing Test Data to Realistic Data	322
20.9 Protecting Tests Violating Encapsulation	324
20.10 Removing Irrelevant Test Information	326
20.11 Adding Coverage for Every Merge Request	328
20.12 Rewriting Tests Depending on Dates	329
20.13 Learning a New Programming Language	330
21. Technical Debt.....	333
21.0 Introduction	333
21.1 Removing Production-Dependent Code	334
21.2 Removing Defect Trackers	335
21.3 Removing Warning/Strict Off	337
21.4 Preventing and Removing Todos and FixMes	338
22. Exceptions.....	341
22.0 Introduction	341
22.1 Removing Empty Exception Blocks	341
22.2 Removing Unnecessary Exceptions	342
22.3 Rewriting Exceptions for Expected Cases	344
22.4 Rewriting Nested Try/Catches	346
22.5 Replacing Return Codes with Exceptions	347
22.6 Rewriting Exception Arrow Code	349

22.7 Hiding Low-Level Errors from End Users	350
22.8 Narrowing Exception Tries	351
23. Metaprogramming.....	353
23.0 Introduction	353
23.1 Removing Metaprogramming Usage	354
23.2 Reifying Anonymous Functions	357
23.3 Removing Preprocessors	359
23.4 Removing Dynamic Methods	361
24. Types.....	363
24.0 Introduction	363
24.1 Removing Type Checking	363
24.2 Dealing with Truthy Values	365
24.3 Changing Float Numbers to Decimals	368
25. Security.....	371
25.0 Introduction	371
25.1 Sanitizing Inputs	371
25.2 Changing Sequential IDs	373
25.3 Removing Package Dependencies	375
25.4 Replacing Evil Regular Expressions	376
25.5 Protecting Object Deserialization	377
Glossary of Terms.....	381
Index.....	395