

---

# Contents

---

Foreword.....	xix
Introduction .....	xxv
On the Cover.....	xxix
<b>Chapter 1: Clean Code.....</b>	<b>1</b>
<b>There Will Be Code .....</b>	<b>2</b>
<b>Bad Code.....</b>	<b>3</b>
<b>The Total Cost of Owning a Mess .....</b>	<b>4</b>
The Grand Redesign in the Sky.....	5
Attitude.....	5
The Primal Conundrum.....	6
The Art of Clean Code?.....	6
What Is Clean Code? .....	7
<b>Schools of Thought .....</b>	<b>12</b>
<b>We Are Authors .....</b>	<b>13</b>
<b>The Boy Scout Rule .....</b>	<b>14</b>
<b>Prequel and Principles .....</b>	<b>15</b>
<b>Conclusion.....</b>	<b>15</b>
<b>Bibliography.....</b>	<b>15</b>
<b>Chapter 2: Meaningful Names .....</b>	<b>17</b>
<b>Introduction .....</b>	<b>17</b>
<b>Use Intention-Revealing Names .....</b>	<b>18</b>
<b>Avoid Disinformation .....</b>	<b>19</b>
<b>Make Meaningful Distinctions .....</b>	<b>20</b>
<b>Use Pronounceable Names.....</b>	<b>21</b>
<b>Use Searchable Names .....</b>	<b>22</b>

<b>Avoid Encodings</b> .....	23
Hungarian Notation .....	23
Member Prefixes.....	24
Interfaces and Implementations .....	24
<b>Avoid Mental Mapping</b> .....	25
<b>Class Names</b> .....	25
<b>Method Names</b> .....	25
<b>Don't Be Cute</b> .....	26
<b>Pick One Word per Concept</b> .....	26
<b>Don't Pun</b> .....	26
<b>Use Solution Domain Names</b> .....	27
<b>Use Problem Domain Names</b> .....	27
<b>Add Meaningful Context</b> .....	27
<b>Don't Add Gratuitous Context</b> .....	29
<b>Final Words</b> .....	30
<b>Chapter 3: Functions</b> .....	31
<b>Small!</b> .....	34
Blocks and Indenting.....	35
<b>Do One Thing</b> .....	35
Sections within Functions .....	36
<b>One Level of Abstraction per Function</b> .....	36
Reading Code from Top to Bottom: <i>The Stepdown Rule</i> .....	37
<b>Switch Statements</b> .....	37
<b>Use Descriptive Names</b> .....	39
<b>Function Arguments</b> .....	40
Common Monadic Forms.....	41
Flag Arguments .....	41
Dyadic Functions.....	42
Triads.....	42
Argument Objects.....	43
Argument Lists.....	43
Verbs and Keywords.....	43
<b>Have No Side Effects</b> .....	44
Output Arguments .....	45
<b>Command Query Separation</b> .....	45

<b>Prefer Exceptions to Returning Error Codes</b> .....	46
Extract Try/Catch Blocks .....	46
Error Handling Is One Thing .....	47
The Error.java Dependency Magnet .....	47
<b>Don't Repeat Yourself</b> .....	48
<b>Structured Programming</b> .....	48
<b>How Do You Write Functions Like This?</b> .....	49
<b>Conclusion</b> .....	49
SetupTeardownIncluder .....	50
<b>Bibliography</b> .....	52
<b>Chapter 4: Comments</b> .....	53
<b>Comments Do Not Make Up for Bad Code</b> .....	55
<b>Explain Yourself in Code</b> .....	55
<b>Good Comments</b> .....	55
Legal Comments .....	55
Informative Comments .....	56
Explanation of Intent .....	56
Clarification .....	57
Warning of Consequences .....	58
TODO Comments .....	58
Amplification .....	59
Javadocs in Public APIs .....	59
<b>Bad Comments</b> .....	59
Mumbling .....	59
Redundant Comments .....	60
Misleading Comments .....	63
Mandated Comments .....	63
Journal Comments .....	63
Noise Comments .....	64
Scary Noise .....	66
Don't Use a Comment When You Can Use a Function or a Variable .....	67
Position Markers .....	67
Closing Brace Comments .....	67
Attributions and Bylines .....	68

Commented-Out Code.....	68
HTML Comments .....	69
Nonlocal Information.....	69
Too Much Information.....	70
Inobvious Connection.....	70
Function Headers.....	70
Javadocs in Nonpublic Code .....	71
Example.....	71
<b>Bibliography.....</b>	<b>74</b>
<b>Chapter 5: Formatting .....</b>	<b>75</b>
<b>The Purpose of Formatting .....</b>	<b>76</b>
<b>Vertical Formatting .....</b>	<b>76</b>
The Newspaper Metaphor.....	77
Vertical Openness Between Concepts .....	78
Vertical Density.....	79
Vertical Distance .....	80
Vertical Ordering.....	84
<b>Horizontal Formatting .....</b>	<b>85</b>
Horizontal Openness and Density.....	86
Horizontal Alignment.....	87
Indentation.....	88
Dummy Scopes.....	90
<b>Team Rules.....</b>	<b>90</b>
<b>Uncle Bob's Formatting Rules.....</b>	<b>90</b>
<b>Chapter 6: Objects and Data Structures .....</b>	<b>93</b>
<b>Data Abstraction.....</b>	<b>93</b>
<b>Data/Object Anti-Symmetry .....</b>	<b>95</b>
<b>The Law of Demeter.....</b>	<b>97</b>
Train Wrecks .....	98
Hybrids.....	99
Hiding Structure .....	99
<b>Data Transfer Objects.....</b>	<b>100</b>
Active Record.....	101
<b>Conclusion.....</b>	<b>101</b>
<b>Bibliography.....</b>	<b>101</b>

<b>Chapter 7: Error Handling</b> .....	103
Use Exceptions Rather Than Return Codes .....	104
Write Your Try-Catch-Finally Statement First .....	105
Use Unchecked Exceptions .....	106
Provide Context with Exceptions .....	107
Define Exception Classes in Terms of a Caller's Needs .....	107
Define the Normal Flow .....	109
Don't Return Null .....	110
Don't Pass Null .....	111
Conclusion .....	112
Bibliography .....	112
<b>Chapter 8: Boundaries</b> .....	113
Using Third-Party Code .....	114
Exploring and Learning Boundaries .....	116
Learning log4j .....	116
Learning Tests Are Better Than Free .....	118
Using Code That Does Not Yet Exist .....	118
Clean Boundaries .....	120
Bibliography .....	120
<b>Chapter 9: Unit Tests</b> .....	121
The Three Laws of TDD .....	122
Keeping Tests Clean .....	123
Tests Enable the -ilities .....	124
Clean Tests .....	124
Domain-Specific Testing Language .....	127
A Dual Standard .....	127
One Assert per Test .....	130
Single Concept per Test .....	131
F.I.R.S.T. .....	132
Conclusion .....	133
Bibliography .....	133
<b>Chapter 10: Classes</b> .....	135
Class Organization .....	136
Encapsulation .....	136

<b>Classes Should Be Small!</b> .....	136
The Single Responsibility Principle.....	138
Cohesion.....	140
Maintaining Cohesion Results in Many Small Classes.....	141
<b>Organizing for Change</b> .....	147
Isolating from Change .....	149
<b>Bibliography</b> .....	151
<b>Chapter 11: Systems</b> .....	153
<b>How Would You Build a City?</b> .....	154
<b>Separate Constructing a System from Using It</b> .....	154
Separation of Main .....	155
Factories .....	155
Dependency Injection.....	157
<b>Scaling Up</b> .....	157
Cross-Cutting Concerns .....	160
<b>Java Proxies</b> .....	161
<b>Pure Java AOP Frameworks</b> .....	163
<b>AspectJ Aspects</b> .....	166
<b>Test Drive the System Architecture</b> .....	166
<b>Optimize Decision Making</b> .....	167
<b>Use Standards Wisely, When They Add <i>Demonstrable Value</i></b> .....	168
<b>Systems Need Domain-Specific Languages</b> .....	168
<b>Conclusion</b> .....	169
<b>Bibliography</b> .....	169
<b>Chapter 12: Emergence</b> .....	171
<b>Getting Clean via Emergent Design</b> .....	171
<b>Simple Design Rule 1: Runs All the Tests</b> .....	172
<b>Simple Design Rules 2–4: Refactoring</b> .....	172
<b>No Duplication</b> .....	173
<b>Expressive</b> .....	175
<b>Minimal Classes and Methods</b> .....	176
<b>Conclusion</b> .....	176
<b>Bibliography</b> .....	176
<b>Chapter 13: Concurrency</b> .....	177
<b>Why Concurrency?</b> .....	178
Myths and Misconceptions.....	179

<b>Challenges</b> .....	180
<b>Concurrency Defense Principles</b> .....	180
Single Responsibility Principle .....	181
Corollary: Limit the Scope of Data .....	181
Corollary: Use Copies of Data .....	181
Corollary: Threads Should Be as Independent as Possible .....	182
<b>Know Your Library</b> .....	182
Thread-Safe Collections .....	182
<b>Know Your Execution Models</b> .....	183
Producer-Consumer .....	184
Readers-Writers .....	184
Dining Philosophers .....	184
<b>Beware Dependencies Between Synchronized Methods</b> .....	185
<b>Keep Synchronized Sections Small</b> .....	185
<b>Writing Correct Shut-Down Code Is Hard</b> .....	186
<b>Testing Threaded Code</b> .....	186
Treat Spurious Failures as Candidate Threading Issues .....	187
Get Your Nonthreaded Code Working First .....	187
Make Your Threaded Code Pluggable .....	187
Make Your Threaded Code Tunable .....	187
Run with More Threads Than Processors .....	188
Run on Different Platforms .....	188
Instrument Your Code to Try and Force Failures .....	188
Hand-Coded .....	189
Automated .....	189
<b>Conclusion</b> .....	190
<b>Bibliography</b> .....	191
<b>Chapter 14: Successive Refinement</b> .....	193
<b>Args Implementation</b> .....	194
How Did I Do This? .....	200
<b>Args: The Rough Draft</b> .....	201
So I Stopped .....	212
On Incrementalism .....	212
<b>String Arguments</b> .....	214
<b>Conclusion</b> .....	250

<b>Chapter 15: JUnit Internals</b> .....	251
<b>The JUnit Framework</b> .....	252
<b>Conclusion</b> .....	265
<b>Chapter 16: Refactoring SerialDate</b> .....	267
<b>First, Make It Work</b> .....	268
<b>Then Make It Right</b> .....	270
<b>Conclusion</b> .....	284
<b>Bibliography</b> .....	284
<b>Chapter 17: Smells and Heuristics</b> .....	285
<b>Comments</b> .....	286
C1: <i>Inappropriate Information</i> .....	286
C2: <i>Obsolete Comment</i> .....	286
C3: <i>Redundant Comment</i> .....	286
C4: <i>Poorly Written Comment</i> .....	287
C5: <i>Commented-Out Code</i> .....	287
<b>Environment</b> .....	287
E1: <i>Build Requires More Than One Step</i> .....	287
E2: <i>Tests Require More Than One Step</i> .....	287
<b>Functions</b> .....	288
F1: <i>Too Many Arguments</i> .....	288
F2: <i>Output Arguments</i> .....	288
F3: <i>Flag Arguments</i> .....	288
F4: <i>Dead Function</i> .....	288
<b>General</b> .....	288
G1: <i>Multiple Languages in One Source File</i> .....	288
G2: <i>Obvious Behavior Is Unimplemented</i> .....	288
G3: <i>Incorrect Behavior at the Boundaries</i> .....	289
G4: <i>Overridden Safeties</i> .....	289
G5: <i>Duplication</i> .....	289
G6: <i>Code at Wrong Level of Abstraction</i> .....	290
G7: <i>Base Classes Depending on Their Derivatives</i> .....	291
G8: <i>Too Much Information</i> .....	291
G9: <i>Dead Code</i> .....	292
G10: <i>Vertical Separation</i> .....	292
G11: <i>Inconsistency</i> .....	292
G12: <i>Clutter</i> .....	293



G13: <i>Artificial Coupling</i> .....	293
G14: <i>Feature Envy</i> .....	293
G15: <i>Selector Arguments</i> .....	294
G16: <i>Obscured Intent</i> .....	295
G17: <i>Misplaced Responsibility</i> .....	295
G18: <i>Inappropriate Static</i> .....	296
G19: <i>Use Explanatory Variables</i> .....	296
G20: <i>Function Names Should Say What They Do</i> .....	297
G21: <i>Understand the Algorithm</i> .....	297
G22: <i>Make Logical Dependencies Physical</i> .....	298
G23: <i>Prefer Polymorphism to If/Else or Switch/Case</i> .....	299
G24: <i>Follow Standard Conventions</i> .....	299
G25: <i>Replace Magic Numbers with Named Constants</i> .....	300
G26: <i>Be Precise</i> .....	301
G27: <i>Structure over Convention</i> .....	301
G28: <i>Encapsulate Conditionals</i> .....	301
G29: <i>Avoid Negative Conditionals</i> .....	302
G30: <i>Functions Should Do One Thing</i> .....	302
G31: <i>Hidden Temporal Couplings</i> .....	302
G32: <i>Don't Be Arbitrary</i> .....	303
G33: <i>Encapsulate Boundary Conditions</i> .....	304
G34: <i>Functions Should Descend Only One Level of Abstraction</i> .....	304
G35: <i>Keep Configurable Data at High Levels</i> .....	306
G36: <i>Avoid Transitive Navigation</i> .....	306
<b>Java</b> .....	307
J1: <i>Avoid Long Import Lists by Using Wildcards</i> .....	307
J2: <i>Don't Inherit Constants</i> .....	307
J3: <i>Constants versus Enums</i> .....	308
<b>Names</b> .....	309
N1: <i>Choose Descriptive Names</i> .....	309
N2: <i>Choose Names at the Appropriate Level of Abstraction</i> .....	311
N3: <i>Use Standard Nomenclature Where Possible</i> .....	311
N4: <i>Unambiguous Names</i> .....	312
N5: <i>Use Long Names for Long Scopes</i> .....	312
N6: <i>Avoid Encodings</i> .....	312
N7: <i>Names Should Describe Side-Effects.</i> .....	313

<b>Tests</b> .....	313
T1: <i>Insufficient Tests</i> .....	313
T2: <i>Use a Coverage Tool!</i> .....	313
T3: <i>Don't Skip Trivial Tests</i> .....	313
T4: <i>An Ignored Test Is a Question about an Ambiguity</i> .....	313
T5: <i>Test Boundary Conditions</i> .....	314
T6: <i>Exhaustively Test Near Bugs</i> .....	314
T7: <i>Patterns of Failure Are Revealing</i> .....	314
T8: <i>Test Coverage Patterns Can Be Revealing</i> .....	314
T9: <i>Tests Should Be Fast</i> .....	314
<b>Conclusion</b> .....	314
<b>Bibliography</b> .....	315
<b>Appendix A: Concurrency II</b> .....	317
<b>Client/Server Example</b> .....	317
The Server .....	317
Adding Threading .....	319
Server Observations .....	319
Conclusion .....	321
<b>Possible Paths of Execution</b> .....	321
Number of Paths .....	322
Digging Deeper .....	323
Conclusion .....	326
<b>Knowing Your Library</b> .....	326
Executor Framework .....	326
Nonblocking Solutions .....	327
Nonthread-Safe Classes .....	328
<b>Dependencies Between Methods</b>	
<b>Can Break Concurrent Code</b> .....	329
Tolerate the Failure .....	330
Client-Based Locking .....	330
Server-Based Locking .....	332
<b>Increasing Throughput</b> .....	333
Single-Thread Calculation of Throughput .....	334
Multithread Calculation of Throughput .....	335
<b>Deadlock</b> .....	335
Mutual Exclusion .....	336
Lock & Wait .....	337

No Preemption.....	337
Circular Wait .....	337
Breaking Mutual Exclusion.....	337
Breaking Lock & Wait.....	338
Breaking Preemption.....	338
Breaking Circular Wait.....	338
<b>Testing Multithreaded Code.....</b>	<b>339</b>
<b>Tool Support for Testing Thread-Based Code .....</b>	<b>342</b>
<b>Conclusion.....</b>	<b>342</b>
<b>Tutorial: Full Code Examples .....</b>	<b>343</b>
Client/Server Nonthreaded.....	343
Client/Server Using Threads .....	346
 <b>Appendix B: org.jfree.date.SerialDate .....</b>	 <b>349</b>
 <b>Appendix C: Cross References of Heuristics.....</b>	 <b>409</b>
 <b>Epilogue.....</b>	 <b>411</b>
 <b>Index .....</b>	 <b>413</b>