Contents

Foreword xix
Introduction xxv
Introduction (from Long Ago) xxix
About the Author xxxi

Chapter 1 Clean Code 1
- There Will Be Code 2
- Bad Code 3
  - Attitude 4
  - The Primal Conundrum 5
- The Art of Clean Code 5
  - What Is Clean Code? 6
- Putting All This Together 9
  - Why Should We Be Clean? 9
  - Productivity 10
  - Livability 13
- We Read More Than We Write 13
- The Boy Scout Rule 14

PART I Code 17

Chapter 2 Clean That Code! 19
- The Cleaning Process 30
- Conclusion 37
- Postscript: Future Bob Playing with Grok3 37
- Postscript Conclusion 40

Chapter 3 First Principles **41**

Everything Small, Well Named, Organized, and Ordered 42

Functions 42

A More Significant Example 44

Independent Deployability 68

A Final Thought 68

Chapter 4 Meaningful Names **71**

Use Intention-Revealing Names 72

Build a System of Names 73

Avoid Disinformation 74

Make Meaningful Distinctions 75

Use Pronounceable Names 76

Use Searchable Names 77

Use Names of Appropriate Length 78

Avoid Encodings 80

Use Appropriate Parts of Speech 81

Consider Keyword Parameters 82

Don't Be Cute 83

Pick One Word per Concept 83

Use Solution Domain Names 83

Use Problem Domain Names 84

Add Meaningful Context 84

Don't Add Gratuitous Context 86

Final Words 87

Chapter 5 Comments **89**

Compensating for Failure 90

Hidden or Obscured Comments 90

Lying Comments 91

Comments That Are Too Intimate 92

Comments Do Not Make Up for Bad Code 92

Explain Your Intent in Code 92

Good Comments 92

Legal Comments 93

Informative Comments 93

Explanation of Intent 94

Clarification 95

Warning of Consequences 96

Amplification 96
Javadocs (and Their Ilk) in Public APIs 97
Bad Comments 97
Mumbling 97
Redundant Comments 98
Misleading Comments 99
Redundancy and Imprecision 99
Mandated Comments 102
Journal Comments 102
Noise Comments 103
Scary Noise 105
TODO Comments 106
Use a Function or a Variable Instead of a Comment 106
Position Markers 107
Attributions and Bylines 107
Commented-Out Code 107
HTML Comments 108
Nonlocal Information 109
Too Much Information 109
Unobvious Connection 110
Function Headers 110
Javadocs in Nonpublic Code 111
Example 111
Conclusion 115

Chapter 6 Formatting 117
The Purpose of Formatting 118
Vertical Formatting 118
Vertical Openness between Concepts 120
Vertical Density 121
Vertical Distance 122
Variable Declarations 122
Dependent Functions 125
Conceptual Affinity 126
Horizontal Formatting 127
Horizontal Openness and Density 128
Horizontal Alignment 129
Indentation 130
Breaking Indentation 132

Team Rules 132
Uncle Bob's Formatting Rules 133

Chapter 7 Clean Functions 137
Small! 138
 Well-Written Prose 139
 One Level of Abstraction per Function 139
Reading Code from Top to Bottom: The Stepdown Rule 140
 Entanglement 142
Switch Statements 142
Clean Functions: A Deeper Look 144
 Contextual 144
 Nameable 145
 Descriptive 145
 Convenient 147
 Insulated 148
 Homogenous 151
 Pure 153
 Partial Purity 155
Conclusion 157

Chapter 8 Function Heuristics 159
Function Arguments 159
 Variadic Arguments 161
 More Than Three? 161
 Keyword Arguments 161
 Flag Arguments 161
 Output Arguments 162
 Error Codes 163
Command Query Separation 164
Prefer Exceptions to Returning Error Codes 165
 Caveat Emptor 165
 Extract `Try/Catch` Blocks 167
 Error Handling Is One Thing 168
 The Dependency Magnet of Error Codes 168
DRY: Don't Repeat Yourself 169
 Simple Repeated Code 169
 Similar Code 170
 Loop Duplication 174

Accidental versus Essential Duplication 177
Side Effects 177
We Are Bad at This 178
Functional Languages 178
Object-Oriented Languages 180
Structured Programming 181
Sequence 182
Selection 182
Iteration 182
This Is Too Much to Constantly Keep in Mind 183
Conclusion 184

Chapter 9 The Clean Method 185
Make It Right 186
Example 188
Considering the Design and Architecture 204
Conclusion 210

Chapter 10 One Thing 211
Extract Method Refactoring 212
This Shouldn't Be Controversial 214
1. Drowning 214
2. Small Functions Don't Obscure Intent 216
3. Performance 216
4. Bouncing Around 216
5. Entanglement 217
What Are Large Functions Anyway? 217
Extraction and Classes 235
Conclusion 239

Chapter 11 Be Polite 241
The Newspaper Metaphor 243
Be Polite 244
The Stepdown Rule: Once Again 245
The Abstraction Roller Coaster 246
This Is How We Write, but Not How We Want to Read 246

Chapter 12 Objects and Data Structures 249
What Is an Object? 250
Data Abstraction 251

Data/Object Antisymmetry 252
The Law of Demeter 255
Trainwrecks 256
Hybrids 256
Hiding Structure 257
Data Transfer Objects 258
The Object/Relational "Impedance Mismatch" 259
Using Objects and Data Structures 259
`switch` Statements 260
The OO Solution 263
Not So Fast, Johnson 264
The OO/Procedural Trade-off 264
But What About Performance? 265
Conclusion 265

Chapter 13 Clean Classes 267
Classes and Modules versus Files 267
What Should a Class Contain? 268
Characterizing Class Design 269
Heuristics and Characteristics 270
When Is a Class Too Large? 271
Policies in Code 273
Where Reasons to Change Hide 274
Fixing the Problem 275
An Overly Open Implementation 277
Should We Do Anything Now? 278
What About Now? 278
Closed, Cohesive, Single-Responsibility Classes 282
When Policies Change 284
Is This Overengineering? 284
Simpler Testing 285
Enter AI 286
It Will Be Wrong 287

Chapter 14 Testing Disciplines 289
Discipline 1: Test-Driven Development (TDD) 291
The Three Laws of TDD 291
Discipline 2: Test && Commit || Revert (TCR) 292

	Discipline 3: Small Bundles	293
	Design	293
	Discipline	294
	Tedious, Boring, and Slow	294
	Debugging	294
	Documentation	295
	Reliability	295
	Design	296
	Reprise	296
	The Angel and the Devil	297
	Muzzling the Devil	297
	Complications and Loopholes	298
	Costs and Repercussions	299
	Keeping Tests Clean	299
	Tests Enable the -ilities	300
Chapter 15	**Clean Tests**	**303**
	Domain-Specific Testing Language	307
	Composed Assertions	307
	Composed Test Results	307
	Dual Standard	309
	The Single Assert Rule	309
	The Single Act Rule	310
	F.I.R.S.T.	310
	Fast	310
	Isolated	310
	Repeatable	310
	Self-Validating	311
	Timely	311
	Test Design	311
	Conclusion	311
Chapter 16	**Acceptance Testing**	**313**
	The Acceptance Testing Discipline	314
	The Discipline	315
	The Continuous Build	316
	Conclusion	316

Chapter 17 **AIs, LLMs, and God Knows What** 317
Programming by Prompt 319
Infancy 324
A SWAG 324
Conclusion 328

PART II **Design** 329

Chapter 18 **Simple Design** 331
YAGNI 333
Covered by Tests 333
An Asymptotic Goal 333
Design? 334
Maximize Expression 334
The Underlying Abstraction 336
Tests: The Other Half of the Problem 337
Minimize Duplication 337
Accidental Duplication 338
Minimize Size 339
Simple Design 339

Chapter 19 **The SOLID Principles** 341
SRP: The Single Responsibility Principle 343
Accidental Duplication 344
Solutions 345
Higher Levels 346
OCP: The Open–Closed Principle 347
A Thought Experiment 347
Directional Control 350
Information Hiding 350
Conclusion 351
LSP: The Liskov Substitution Principle 351
LSP and Design 352
Taxi Aggregator 353
ISP: The Interface Segregation Principle 354
ISP and Language 355
ISP and Design 355
DIP: The Dependency Inversion Principle 356
Stable Abstractions 358

Factories 359
Concrete Components 360

Chapter 20 Component Principles 363
Components 364
A Brief History of Components 364
Relocatability 367
Linkers 367
Component Cohesion 368
The Reuse/Release Equivalence Principle (REP) 369
The Common Closure Principle (CCP) 370
The Common Reuse Principle (CRP) 371
The Tension Diagram for Component Cohesion 372
In Summary... 373
Component Coupling 373
The Acyclic Dependencies Principle (ADP) 374
The Stable Dependencies Principle (SDP) 379
The Stable Abstractions Principle (SAP) 383
Conclusion 387

Chapter 21 Continuous Design 389
Continuous Change 390
Continuous Design 391
Sailing on the Four Cs of Continuous Design 392
Clarity 393
Conciseness 398
Confirmability 406
Cohesion 416
When Else Do We Design? 420
Up-front Design 420
Readying for Work 421
Starting Work 421
Doing Work 422

Chapter 22 Concurrency 423
Why Concurrency? 424
Myths and Misconceptions 425
Challenges 426

Concurrency Defense Principles 427
Single Responsibility Principle 427
Corollary: Limit the Scope of Data 427
Corollary: Use Copies of Data 428
Corollary: Threads Should Be as Independent as Possible 428
Know Your Language and Library 428
Thread-Safe Collections 429
Know Your Execution Models 429
Producer-Consumer 430
Readers-Writers 430
Dining Philosophers 431
Beware Dependencies between Synchronized Methods 431
Keep Synchronized Sections Small 431
Writing Correct Startup and Shutdown Code Is Hard 432
Testing Threaded Code 432
Treat Spurious Failures as Candidate Threading Issues 433
Get Your Nonthreaded Code Working First 433
Make Your Threaded Code Pluggable 434
Make Your Threaded Code Tunable 434
Run with More Threads Than Processors 434
Run on Different Platforms 434
Instrument Your Code to Try and Force Failures 435
Hand-Coded 435
Automated 436
2025 Update and Report from the Field 437
Data Integrity 437
Conclusion 442

PART III Architecture 443

Chapter 23 The Two Values of Software 445
Keeping Options Open 446

Chapter 24 Independence 449
Use Cases 450
Operation 450
Development 451
Deployment 451
Leaving Options Open 451

Chapter 25 Architectural Boundaries **453**
What Lines Do You Draw, and When? 454
Plug-in Architecture 456
Case Study: FitNesse 457
Conclusion 459

Chapter 26 Clean Boundaries **461**
Third-Party IoT Framework: Lots o' Boundaries 462
UI/Application Boundary 466
- SOLID and Hexagonal Architecture 469
- Exploring and Learning Boundaries 470
- Using Code That Does Not Yet Exist 473

Clean Boundaries 474

Chapter 27 The Clean Architecture **475**
The Dependency Rule 476
- Entities 477
- Use Cases 477
- Interface Adapters 478
- Frameworks and Drivers 478
- Only Four Circles? 478
- Crossing Boundaries 479
- What Data Crosses the Boundaries 479
- A Typical Scenario 479

Conclusion 481

PART IV Craftmanship **483**
"A Great Number" 484
Eight Decades 484
- Nerds and Saviors 487
- Notoriety 488
- Role Models and Villains 489
- We Rule the World 490
- Catastrophes 491

The Oath 492

Chapter 28 Harm **495**
No Harm to Society 496
Harm to Function 497

No Harm to Structure 499
Soft 500
Tests 501

Chapter 29 **No Defect in Behavior or Structure** **503**
Making It Right 504
What Is Good Structure? 504
Eisenhower's Matrix 506
Programmers Are Stakeholders 507
Do Your Best 508

Chapter 30 **Repeatable Proof** **511**
Dijkstra 511
Proving Correctness 512
Structured Programming 514
Functional Decomposition 516
Test-Driven Development et al. 517

Chapter 31 **Small Cycles** **519**
The History of Source Code Control 519
Punch Cards 519
Continuous Integration 523
Short Cycles 524
Continuous Integration 524
Branches versus Toggles 525
Continuous Deployment 527
Continuous Build 528

Chapter 32 **Relentless Improvement** **529**
Test Coverage 529
Mutation Testing 530
Semantic Stability 530
Cleaning 531
Creations 531

Chapter 33 **Maintain High Productivity** **533**
Viscosity 533
Building 534
Testing 534

Debugging 535
Deploying 535
Managing Distractions 535
Meetings 536
Music 536
Mood 537
The Flow 537
Time Management 538

Chapter 34 Work as a Team 539
Collaborative Programming 539
Open/Virtual Office 540

Chapter 35 Estimate Honestly and Fairly 543
Lies 544
Honesty, Accuracy, Precision 544
Lessons from Me 545
Story #1: Vectors 545
Story #2: pCCU 546
The Lesson 547
Accuracy and Precision 547
Aggregation 548
Honesty 549
Pressure 550

Chapter 36 Respect for Fellow Programmers 553

Chapter 37 Never Stop Learning 555

Afterword 557

Appendix: The Clean Code Debate 561
Introductions 562
Method Length 563
Method Length Summary 575
Comments 576
Comments Summary 590
John's Rewrite of `PrimeGenerator` 591
A Tale of Two Programmers 596

Bob's Rewrite of `PrimeGenerator2` 599
Test-Driven Development 603
TDD Summary 612
Closing Remarks 613

Bibliography **615**

Index **619**