Index

Note: Page numbers followed by "b", "f" and "t" indicate boxes, figures and tables respectively.

A

ABS. *See* Anti-braking system
Abstract Factory pattern, 36, 37f
AbstractButton class, 188, 189f
Abstraction, 21
  enabling techniques for, 22f, 24t
  human smiley face, 21f
  principle, 22
  smells resulting due to violation, 23f
  unutilized, 49–53
AbstractOwnableSynchronizer class, 142
  refactoring, 147, 148f
  unfactored methods in subtypes, 143f
AbstractQueuedLongSynchronizer class, 13–14, 142, 143b
AbstractQueuedSynchronizer class, 13–14, 142
Acyclic Dependencies Principle (ADP), 108
Advanced Encryption Standard (AES), 80
Anti-braking system (ABS), 61
Antipattern, 13
Application Programming Interface (API), 171, 12, 26
Architectural critique tools, 204
Auto-generated code, 101, 107

B

"Big-bang" refactoring, 211
Bloated implementation, 103
Bloated interface and implementation, 103–105
Booch's object model, 16
Bounded Context, 60
Broken Hierarchy, 173
  aliases, 180
  causes, 173
  examples, 174–178
  practical considerations, 180–181
  quality attributes, 179–180
  rationale, 173
  refactoring, 178–179
Broken Modularization, 96, 199–200
  aliases, 101
  causes, 96
  classes in device management application, 97f
  examples, 97–99
  impacted quality attributes, 100–101
  practical considerations, 101–102
  rationale, 96
  refactoring, 99–100
Bug patterns. *See* Code analysis tools
ButtonModel interface, 38f
Buy-in for refactoring, 207–208

C

C-like procedural languages, 127
Calendar class, 13
Call-back functionality, 109–110
CallSite holder, 164b
  refactoring for, 166f–167f
Chained if-else statements, 127
Changeability, 28, 32, 39, 43, 59
ChangeableCallSite, 168
"Class Adapter" pattern, 180–181, 181f
Class with unparameterized methods, 71
Cloud ecosystem, 110
CoC. *See* Cost of Change
Code analysis tools, 204
Code clones, 54b
  detectors, 204–205
Code debt, 2
Collapse hierarchy, 155
Collapsed type hierarchy, 133
ColorModel class, 89, 88
Command pattern, 34
ComponentColorModel class, 87
Composite pattern, 172f, 172b
Comprehension tools, 204
"Compromised" Properties, 175b
ConcurrentLinkedQueue class, 183
ConstantCallSite class, 164
Cooccurring smells, 198–199. *See also* Precursors smells
Copy-paste programming, 55
Cost of Change (CoC), 4
Critique tools, 204–205
CustomListBox class, 75
Cyclic Hierarchy, 187, 197
  AbstractButton class, 189f
  aliases, 191–192
  causes, 188
  examples, 188
  practical considerations, 192
  quality attributes, 191
  rationale, 188
  refactoring, 189–190
  state design pattern, 190f
  strategy design pattern, 190f

Cyclically-dependent Modularization, 200, 108, 201f, 201b
aliases, 117–118
causes, 109–110
breaking, 113f–114f
between classes A and B, 113f
cycles in java.util package, 109f
examples, 110–112
between image and encryption classes, 111f
impacted quality attributes, 116–117
between matcher and pattern, 118f
between order and OrderImpl classes, 112f
between order and TaxCalculator classes, 111f
practical considerations, 118
rationale, 108–109
refactoring, 112–116
understanding cycles, 108b–109b

D

"Data buffer", 136b–137b
Data clumps, 26, 28
Data Encryption Standard (DES), 80
Data Transfer Object (DTO), 102
DataBuffer class, 87
DatagramChannel hierarchy, 159f, 160–161
Deep Hierarchy, 157–159, 198
aliases, 163
causes, 158
examples, 158–159
practical considerations, 163
quality attributes, 161–163
rationale, 157–158
refactoring, 159–161
Deficient Encapsulation, 63, 199–200
aliases, 71
causes, 64
examples, 64–69
global variable "current" in Linux kernel, 68f
impacted quality attributes, 70
practical considerations, 71–72
public data members in java.awt.Point class, 64f
rationale, 63
refactoring, 69–70
smell effect
Definition Editor, 194–196, 195f
"Degenerate inheritance", 187
Dependency diagram, 108b–109b
DES. *See* Data Encryption Standard
DESEncryption class, 110
Design debt, 2
Design fragment, 10–11
Design patterns, 13
Design principle violation, 13
Design smells, 1, 9
address, 15
care about, 10–12
catalog, 9
causes, 12, 12f
design principles violation, 13
inappropriate use of patterns, 13
language limitations, 13–14
nonadherence to practices and processes, 14
procedural thinking in OO, 14
viscosity, 14
classification, 15–16, 17f
design principles based classification, 16
high-level principles, 16t
naming scheme for, 16–18
template for documenting smells, 18–19, 18t–19t
quality attributes, 10t
DeviceData class, 97
Diamond hierarchy, multipath hierarchy *vs*., 182b
Distorted hierarchy, 163
Documentation debt, 2
Don't Repeat Yourself (DRY), 55, 127
principle, 141
DTO. *See* Data Transfer Object
"Dump-all" utility classes, 119
Duplicate Abstraction, 54, 178. *See also* Abstraction
aliases, 59
causes, 55
examples, 56–57
impacted quality attributes, 59
practical considerations, 59–60
rationale, 55
refactoring, 57–58
Duplicate abstraction smell, 194

E

E-commerce application, 47
Encapsulation, 61
application of principle, 61
deficient, 63–72
design smells, 63t
enabling techniques for, 62f
leaky, 72–78
missing, 78–86
smells, 62f
unexploited, 86–91
violated encapsulation enabling techniques, 63t
Environment viscosity, 14
EventObject hierarchy, 153f
Extensibility, 28, 32, 39, 43, 59
"Extract hierarchy" refactoring, 130

F

Factory Disposal Method pattern, 38
Fan-out, 95
Feature-rich e-mail application, 73–75
First-In-First-Out (FIFO), 50
Font class hierarchy, 135f
“Functional decomposition” approach, 33b
Functionally similar clones, 54b
Future-proofing, 155

G

Generalization, 125

H

“Half-hearted operations”, 39
Hard-to-visualize indirect dependencies, 110, 188
HashSet methods, 174–175
Hideable public attributes/methods, 71
Hierarchy
  broken, 173–181
  cyclic, 187–192
  deep, 157–163
  enabling techniques for principle, 124f
  human species classification, 123f
  missing, 127–134
  modern software, 124
  multipath, 182–187
  principle, 124
  rebellious, 163–173
  speculative, 154–157
  type, 124
  unfactored, 140–149
  unnecessary, 134–140
  wide, 150–154
HTMLEditorKit class, 170
HTMLReport hierarchy, 156f
Hub-like Modularization, 118
  aliases, 122
  causes, 119
  examples, 119
  impacted quality attributes, 122
  practical considerations, 122
  rationale, 118–119
  refactoring, 119–121
Human species, 123f

I

IDE. *See* Integrated Development Environment
IMPACT refactoring process model, 208, 209f. *See also* Buy-in for refactoring
  action on planned refactoring tasks, 210
  behavior preservation ensuring test, 210
  refactoring activities planning, 209–210
  refactoring candidates identification, 209
Imperative Abstraction, 29
  aliases, 34
  classes
    image processing application, 31f
    report generation functionality, 30f
  examples, 30–31
  impacted quality attributes, 32–34
  practical considerations, 34
  procedural thinking, 29
  rationale, 29
  refactoring, 31–32, 31f–32f
  smell, 29
Inadequate design analysis, 25
Incomplete Abstraction, 34
  aliases, 39
  ButtonModel interface, 38f
  causes, 35
  examples, 35–37
  impacted quality attributes, 39
  practical considerations, 39–40
  rationale, 34–35
  refactoring, 37–38
  symmetric method pairs, 37t
Inheritance hierarchy, 136
“Inline class” refactoring, 47
Insufficient Modularization, 102
  aliases, 107
  causes, 103
  examples, 103–105
  impacted quality attributes, 106–107
  Multifaceted Abstraction *vs*., 102b
  practical considerations, 107
  rationale, 102–103
  refactoring, 105–106
Integrated Development Environment (IDE), 205
International Standard Book Numbers (ISBNs), 25
Interplay of smells, 196
  in isolation, 196
    amplifying other smells, 198
    cooccurring smells, 198–199
    problems, 200–201
    smells as precursors, 199–200
  in java.util library, 197f
Inverted hierarchy, 178f–179f, 179
IS-A relationship, 173, 176b
ISBNs. *See* International Standard Book Numbers

J

java.awt.Component class, 103, 105f
java.awt.Font class, 119, 120f
java.awt.Point class, 64–65

java.awt.Rectangle class, 144, 145f
java.awt.Toolkit superclass, 164–165
java.io.Console class, 69–70
java.io.ObjectStreamClass, 133, 133b
java.io.ObjectStreamConstants class, 46–47
java.lang.invoke.CallSite class, 164, 165f
java.lang.Math classes, 49
java.lang.StrictMath classes, 49
java.lang.System class, 66–69
java.net.SocketPermission, 103
java.nio.Buffer, 137f
java.sql package, 176–177
java.swing.JLabel class, 171
java.swing.JTable class, 105f
java.swing.plaf.windows.XPStyle.java example, 128, 128b
  refactoring suggestion, 130–131
java.text.NumberFormat, 141–142
java.util library, 197f
java.util.Calendar class, 13, 41
java.util.Component class, 119
java.util.Date class, 176, 177f
java.util.EventObject class, 151, 152f
java.util.Stack class, 13
java.util.Timer class, 56
javax.management.timer.Timer class, 56
javax.sound.sampled.AudioFormat class, 128, 129b
  refactoring suggestion, 131–132
javax.swing.ColorChooserDialog hierarchy, 198, 199f
javax.swing.text.rtf.RTFEditorKit, 170
javax.swing.Timer class, 56
javax.swing.WindowConstants interface, 46
JDialog class, 46–47
JFrame class, 46–47
JInternalFrame class, 46–47
JobStateReasons, 174b
  partial inheritance hierarchy, 174f
Joda class, 41

K
Key classes, 107, 122

L
L & F. *See* Look & Feel
Leaky Encapsulation, 72
  aliases, 78
  case study, 77b
  causes, 72–73
  class leaking implementation details, 75f
  examples, 73–75
  impacted quality attributes, 77–78
  practical considerations, 78
  rationale, 72
  refactoring, 75–77
LineStyle hierarchy, 136f
  refactoring for, 138–139, 139f
Linux, 68b
Liskov's Substitution Principle (LSP), 13, 125
List-like inheritance hierarchies, 157
Look & Feel (L & F), 159
LSP. *See* Liskov's Substitution Principle

M
Map domain entities, 29
Mental blocks, 207
Metric tools, 204–205
Missing Abstraction, 24
  aliases, 28
  causes
    inadequate design analysis, 25
    lack of refactoring, 25
    misguided focus on minor performance gains, 25
  examples, 25–26
  impacted quality attributes, 28
  practical considerations, 28–29
  rationale, 24
  refactoring, 26–28
Missing Encapsulation, 78
  aliases, 85
  causes, 79–80
  examples, 80–82
  impacted quality attributes, 85
  practical considerations, 86
  rationale, 78–79
  refactoring, 82–85
Missing Hierarchy, 127
  aliases, 133
  causes, 127–128
  design smells and enabling techniques, 127t
  examples, 128–130
  interacting with external world, 133–134
  quality attributes, 132
  rationale, 127
  refactoring, 130–132
"Missing inheritance", 133
"Mistaken aggregates", 180
Modern software, 124
Modularization, 93
  broken, 96–102
  cyclically-dependent, 108–118
  design smells and violated enabling technique, 95t
  enabling techniques for, 94f, 95
  hub-like, 118–122

insufficient, 102–107
principle, 94
smells resulting from violation, 95f
Module, 94
“Move method” refactoring, 166
Multifaceted Abstraction, 40, 102b, 198–199
aliases, 44
causes, 40–41
examples, 41–42
impacted quality attributes, 43–44
practical considerations, 44
rationale, 40
refactoring, 42–43, 43f
Multipath Hierarchy, 182
aliases, 187
causes, 183
diamond hierarchy *vs*., 182b
examples, 183–186
in Java, 183f
practical considerations, 187
quality attributes, 186–187
rationale, 182–183
refactoring, 186

N

Naming scheme for smells, 16–18
“Naughty children”, 170
NO Operation (NOP), 163
Novel naming scheme, 17–18
NumberFormat class, 142f, 147f

O

Object Adapter pattern, 181f
“Object classes”, 140
Object hierarchy, 124
Object model, 16
Object-oriented context (OO context), 14
decomposition, 33b
languages, 124
principles, 86
Observation Editor, 194–196, 195f
OCP. *See* Open Closed Principle
OffsetCalculator classes, 30–31
OO context. *See* Object-oriented context
Open Closed Principle (OCP), 78–79
Operating System (OS), 135–136
Operation class, 34
Order class, 97–99, 111–112
OrderImpl class, 111–112
OrderList class, 74b
org.eclipse.jdt.internal. compiler.parser package, 56–57
Orphan abstractions, 49–51
OS. *See* Operating System
Over-engineering, 155

P

People role, 212
leading by example, 212
training, 212
PHAME. *See* Principle of Hierarchy, Abstraction, Modularization, and Encapsulation
PMD Copy Paste Detector, 146b
Post-initialization, 183b–184b
Pre-initialization, 183b–184b
Precursors smells, 199–200
Premature interface abstraction, 170
Primitive obsession, 28
Principle of Hierarchy, Abstraction, Modularization, and Encapsulation (PHAME), 16
PrintEvent class, 151–152
Procedural languages, 96
Procedural thinking, 86
Public classes, 65–66
“Pull-up class” refactoring, 152f

R

Rebellious hierarchy, 163
aliases, 169–170
causes, 164
composite pattern, 172b
examples, 164–165
practical considerations, 170–173
quality attributes, 169
rationale, 163–164
refactoring, 166–169
Refactoring, 1, 9, 15, 203
for Broken Hierarchy, 178–179
buy-in for, 207–208
challenges in, 206–207
for Cyclic Hierarchy, 189–190
for Deep Hierarchy, 159–161
for Deficient Encapsulation, 69–70
IMPACT refactoring process model, 208–210, 209f
incompletely refactored design, 185f
intended, 185f
for Leaky Encapsulation, 75–77
for Missing Encapsulation, 82–85
for Missing Hierarchy, 130–132
for Multipath Hierarchy, 186
people, 212
leading by example, 212
training, 212
process, 206
for Rebellious Hierarchy, 166–169

Refactoring (*Continued*)
repay technical debt, 203, 204f
to repay technical debt, 211–212
for Speculative Hierarchy, 155–156
suggestion, 26–28
tools, 203
categories, 205
code clone detectors, 204–205
comprehension tools, 204
critique tools, 204–205
metric tools, 204–205
in practice, 205–206
technical debt quantification, 205
visualization tools, 205
for Unexploited Encapsulation, 88–89
for Unfactored Hierarchy, 146–148
for Unnecessary Hierarchy, 138–139
for Wide Hierarchy, 151–153
"Refused bequest", 169
"Refused parent bequest", 170
Reification, 34
"Remove method" refactoring, 166
Repaying technical debt, 203, 204f
"Repeated inheritance", 187
Reusability, 28, 32–33, 39, 43–44, 48, 59
Rich Text Format (RTF), 170
RTFEditorKit inheritance hierarchy, 170f
RTF. *See* Rich Text Format

S
SaveHandler classes, 30–31
Scrum process, 53b
SecureDocument class, 110
Sibling types, 140
Single Responsibility Principle (SRP), 11–12, 35, 102b
Smell ecosystem, 193
interplay of smells, 196
in isolation, 196
amplifying other smells, 198
cooccurring smells, 198–199
problems, 200–201
smells as precursors, 199–200
in java.util library, 197f
role of context, 193
Definition Editor, 195f
design decision, 193–194
developers and designers, 196
Observation Editor, 195f
smells, 194
Software design, 193
Software viscosity, 14
Speculative generalization, 158
Speculative Hierarchy, 154
aliases, 157
causes, 155
examples, 155
practical considerations, 157
quality attributes, 157
rationale, 155
refactoring, 155–156
SQL. *See* Structured Query Language
SRP. *See* Single Responsibility Principle
State pattern, 34
Strategy pattern, 34
StrictMath methods, 49
Structured Query Language (SQL), 176–177
sun.misc.Service class, 50
Supertype process, 124, 140
"Swiss-knife" types, 164
Switch-based-on-type-code, 127, 134
"Switch-case" statements, 133
SynCheckBoxUI class hierarchy, 160f

T
"Tag class", 133
"Tagged types", 127
TaxCalculator class, 112
Taxonomy mania (Taxomania), 134, 140
Technical bankruptcy, 2
Technical debt, 1–2, 207
impact of, 4
causes, 6–7
dimensions of, 3f
management, 7
multiple sources of, 2
quantification, 205
Test debt, 2
Testability, 28, 33, 44
TextGadget class, 184f
Textually similar clones, 54b
TodoList class, 73
Type hierarchy, 124

U
Unencapsulated class, 71
Unexploited Encapsulation, 86
aliases, 91
causes, 86–87
examples, 87–88
impacted quality attributes, 89–91
rationale, 86
refactoring, 88–89
Unfactored Hierarchy, 140, 194–196
aliases, 149
causes, 141

code clone detector tool, 146b
examples, 141–146
practical considerations, 149
quality attributes, 148–149
rationale, 140–141
refactoring, 146–148
Unfactored implementation, 140
Unfactored interface, 140
Unnecessary Abstraction, 44, 1965
aliases, 48
causes, 45
examples, 45–47
impacted quality attributes, 48
practical considerations, 48–49
rationale, 44–45
refactoring, 47–48
Unnecessary Hierarchy, 134
aliases, 140
causes, 134
"data buffer", 136b–137b
examples, 135–136
practical considerations, 140
quality attributes, 139
rationale, 134
refactoring, 138–139
Unreferenced abstractions, 49–50
Unutilized Abstraction, 49, 51f
aliases, 53
causes, 49–50
examples, 50–52
impacted quality attributes, 53
practical considerations, 53
rationale, 49
refactoring, 52–53

V
Variation Encapsulation Principle (VEP), 79
Viscosity, 14
Visualization tools, 205

W
Weighted Methods per Class (WMC), 205
Wide Hierarchy, 150
aliases, 154
causes, 151
examples, 151
practical considerations, 154
quality attributes, 153
rationale, 150–151
refactoring, 151–153
WMC. *See* Weighted Methods per Class

Y
Yet-to-be implemented functionality, 170–171
You Aren't Gonna Need It (YAGNI), 49, 155