

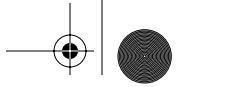
INDEX

- Abstract classes
 - for class hierarchies, 63
 - for filter classes, 288
 - and interfaces, 11–13
 - object adapters for, 27
- ABSTRACT FACTORY pattern, 175
 - challenge solutions for, 394–398
 - and FACTORY METHOD, 180–184
 - GUI kits, 175–180
 - packages in, 184–185
 - summary, 185
- Abstract syntax trees
 - defined, 439
 - for VISITORS, 339
- Abstraction
 - for bridges, 63–68
 - defined, 439
- Access modifiers
 - for FLYWEIGHTS, 150
 - for methods, 210
 - for responsibility, 77–78
- Active Server Pages for .NET (ASP.NET), 125
- ADAPTER pattern, 17
 - challenge solutions for, 348–353
 - class and object adapters, 21–25
 - identifying adapters, 30–31
 - for interfaces, 17–21
 - for JTable data, 25–30
 - summary, 31
- Advertising
 - modeling strategies for, 242–244
 - refactoring for, 244–248
- Aerial shells
 - COMPOSITES for, 56–60
 - defined, 439
 - ITERATORS for, 320, 322
 - VISITORS for, 334–335, 338
- Alexander, Christopher, 1–2
- Algorithms, 209
 - completing, 221–224
 - defined, 439
 - methods for, 213–214
 - vs. STRATEGIES, 241
 - templates for, 217–219
- Alternations, visiting, 335–337
- Anchoring chains of responsibility, 142–144
- AOP (aspect-oriented programming), 136
- Apogee
 - defined, 439
 - sorting data for, 219–221
- app directory, 428
- Application programming interface (API)
 - for database drivers, 69
 - defined, 439
 - JavaBeans, 99
 - JDBC, 69
- Applications
 - multithreaded, 311–313
 - in object-oriented systems, 33
 - thick, 103
- Arcs, equations for, 40
- Arrays, sorting, 218–219

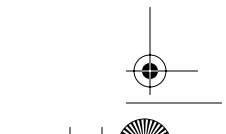
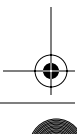
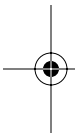
- Arrays class, 218–219, 249
- ASP.NET (Active Server Pages for .NET), 125
- Aspect-oriented programming (AOP), 136
- Assay
 - defined, 439
 - model of, 51
- aster directory, 428
- Attributes in copying, 189
- Automating proxies, 127

- Ballistics, 13
 - GUIs for, 87–92
 - MVC design in, 92–99
- Binary format, serialization for, 205
- Binary relations, inverse, 110
- Bodies, methods, 210
- Boolean conditions, 270–271
- BRIDGE pattern, 63
 - abstraction for, 63–68
 - challenge solutions for, 360–362
 - drivers for, 68–71
 - summary, 71
- Brightness of stars, 301–303
- BufferedWriter class, 287
- BUILDER pattern, 159
 - challenge solutions for, 387–390
 - constraints in, 162–164
 - forgiving, 164–165
 - ordinary, 159–162
 - summary, 165–166
- Building Oozinoz code, 427
- Burn rate in rocket simulation, 88–90, 93
- Business objects
 - defined, 439
 - GUI objects divided from, 94
- Carousel doors
 - modeling state of, 230–233
 - refactoring, 233–237
- Carousels, defined, 439
- Cartesian products, 109
- Categories, pattern, 5–6
- CHAIN OF RESPONSIBILITY pattern, 137
 - anchoring, 142–144
 - challenge solutions for, 378–382
 - without COMPOSITES, 144
 - ordinary, 137–139
 - refactoring in, 139–142
 - summary, 144
- Challenge solutions
 - ABSTRACT FACTORY, 394–398
 - ADAPTER, 348–353
 - BRIDGE, 360–362
 - BUILDER, 387–390
 - CHAIN OF RESPONSIBILITY, 378–382
 - COMMAND, 410–414
 - COMPOSITE, 356–359
 - construction, 386–387
 - DECORATOR, 419–421
 - extensions, 416–417
 - FACADE, 353–356
 - FACTORY METHOD, 390–394
 - FLYWEIGHT, 382–385
 - interfaces, 347–348
 - INTERPRETER, 415–416
 - ITERATOR, 421–422
 - MEDIATOR, 372–376
 - MEMENTO, 400–403
 - OBSERVER, 366–371
 - operations, 403–405
 - PROTOTYPE, 398–400
 - PROXY, 376–378
 - responsibility, 362–364
 - SINGLETON, 364–366
 - STATE, 407–408
 - STRATEGY, 409–410
 - TEMPLATE METHOD, 405–407
 - VISITOR, 423–425
- Challenges
 - description of, 4–5
 - guidelines for, 343

- Chemical tubs
 - moving, 104–108
 - relational integrity for, 108–115
- Chemicals
 - extracting, 146–148
 - sharing, 148–152
- Children nodes in iteration, 314–317
- Circles, plotting, 39–41
- Class adapters
 - creating, 21–22
 - defined, 439
- Class/instance paradigm, 3
- Class loaders for dynamic proxies, 132–134
- Classes
 - abstract, 11–13
 - as abstraction, 63–64
 - constructors for. *See* Construction
 - for dynamic proxies, 132
 - extending, 14
 - implementation of, 11
 - instantiating, 169–171
 - for layers, 94
 - in Liskov Substitution Principle, 280
 - SINGLETONS, 84–86
 - stub, 14
 - UML, 431–435
 - visibility of, 77–78
- Clients
 - defined, 439
 - needs of, 17–21
 - objects as, 17
- clone() method, 189–190, 313
- Clones
 - for collections, 312–313
 - prototyping with, 189–191
- Code
 - inserting patterns into, 344–345
 - sharing by handshaking classes, 67
- Code smells, removing, 283
- Collections
 - cloning, 312–313
 - iterators for, 167–168
 - loops for, 305
 - sorting, 218–219
 - for thread safety, 307–312
- com directory, 428
- com.oozinoz.testing package, 428
- COMMAND pattern, 251
 - challenge solutions for, 410–414
 - hooks for, 255–257
 - menu commands, 251–254
 - relation to other patterns, 257–258
 - for services, 254–255
 - summary, 259
- Common Object Request Broker Architecture (CORBA)
 - defined, 440
 - for remote proxies, 125
- Comparator interface, 218–219, 221
- Comparisons
 - for Boolean conditions, 270
 - in sorting, 217–219
- Completing algorithms, 221–224
- COMPOSITE pattern, 47
 - challenge solutions for, 356–359
 - cycles in, 56–60
 - ordinary composites, 47–48
 - recursive behavior in, 48–50
 - summary, 60–61
 - trees in, 50–56
- Composites
 - CHAINS OF RESPONSIBILITY without, 144
 - defined, 439
 - enumerators, 321–322
 - iterating over, 313–320
- Concurrent Programming in Java™*, 346
- Consistency, relational, 109–110
- Consistent classes in LSP, 280
- Consolidation languages
 - defined, 440
 - Java as, 3



- Constant states, 238–240
- Constants in interfaces, 14
- Constraints, building under, 162–164
- Construction, 155
 - challenge solutions for, 386–387
 - challenges in, 155–157
 - summary, 157
- Construction patterns. *See* ABSTRACT FACTORY pattern; BUILDER pattern; FACTORY METHOD pattern; MEMENTO pattern; PROTOTYPE pattern
- Constructors
 - for class instantiation, 169
 - defined, 440
- Context-free languages, 440
- Controllers in MVC design, 92–99
- Copying
 - collections, 312–313
 - prototypes, 189–191, 400
- CORBA (Common Object Request Broker Architecture)
 - defined, 440
 - for remote proxies, 125
- Coupling, loose, 108
- Covariant return types
 - defined, 440
 - for subclasses, 404
- Credit
 - ABSTRACT FACTORY for, 180–184
 - class instantiation for, 169–171
- Curves, modeling, 39–41
- Cycles
 - in COMPOSITES, 51, 56–60
 - consequences of, 60
 - defined, 440
 - VISITOR, 333–338
- Database drivers, 69–71
- DECORATOR pattern, 287
 - challenge solutions for, 419–421
 - function wrappers, 295–303
 - relation to other patterns, 303
 - streams and writers, 287–295
 - summary, 303–304
- Decoupling, 137
- Deep copies
 - defined, 440
 - vs. shallow, 399
- Default constructors, 155–156
- Demeter Project, 281–283
- Demos, 33–35, 353–354
- Dependencies, observers for, 87, 95, 100
- Depth of composite enumerators, 321–322
- Descendant nodes in iteration, 314
- Design patterns
 - defined, 440
 - purpose of, 2–3
- Design Patterns* (Gamma, Helm, Johnson, and Vlissides), 1–3, 204
- Directed graphs
 - defined, 440
 - object models as, 51
- Directories, Oozinoz, 428
- Domain objects, 94
- Doors, carousel
 - modeling state of, 230–233
 - refactoring, 233–237
- Double dispatch technique
 - defined, 440
 - for visitors, 332
- Driver pattern. *See* BRIDGE pattern
- Drivers
 - as BRIDGES, 68–69
 - database, 69–71
 - defined, 440
- Duds
 - defined, 440
 - flight path of, 35–38, 42–46
- Durability of mementos, 201
- Dynamic proxies, 131–136



- Eclipse, 33, 427
- Edges in graphs, 50
- Encapsulation
 - defined, 440
 - in object-oriented programming, 79
- Enterprise JavaBeans (EJB)
 - defined, 440
 - for remote proxies, 125
- Enumerators, 305
 - depth of, 321–322
 - in iteration, 301, 315
 - for leaves, 322–323
- Equations, parametric, 39–41
- Exceptions
 - for reservations, 162–163
 - throwing, 212
- Expertise for visitors, 339
- Extending classes, 14
- Extensible markup language (XML), 159, 441
- Extensions, 279
 - challenge solutions for, 416–418
 - code smells, 283
 - Law of Demeter (LOD), 281–283
 - object-oriented design principles, 279–281
- Extensions patterns. *See* DECORATOR pattern; ITERATOR pattern; VISITOR pattern
- Extensible Markup Language (XML), 159
- Extracting flyweight immutable part, 146–148
- FACADE pattern, 33
 - challenge solutions for, 353–356
 - parametric equations for, 39–41
 - refactoring in, 35–38, 42–46
 - utility and demos in, 33–35
- Factories, prototypes as, 187–188
- FACTORY METHOD pattern
 - and ABSTRACT FACTORIES, 180–184
 - challenge solutions for, 390–394
 - class instantiation in, 169–171
 - ITERATORS in, 167–168
 - in parallel hierarchies, 171–173
 - recognizing, 168–169
 - summary, 173–174
- Families of objects, 182, 184
- FileWriter class, 288
- Filters for streams, 288–293
- FilterWriter class, 296
- Finding Oozinoz files, 428
- Fireworks production
 - recursive behavior in, 48–50
 - trees in, 50–56
- Flight paths
 - plotting, 39–41
 - refactoring in, 35–38, 42–46
- Flow control for INTERPRETERS, 271–272
- FLYWEIGHT pattern, 145
 - challenge solutions for, 382–385
 - immutability in, 145–148
 - sharing in, 148–152
 - summary, 152
- for loops, 305
- foreach loops, 305
- Forgiving builders, 164–165
- Function wrappers, 295–303
- Gamma, Erich, 1
- Gang of Four, 3
- Grammars
 - defined, 441
 - for generators, 275
- Graph theory
 - defined, 441
 - trees in, 51
- Graphs
 - for composites, 50–56
 - defined, 441

- Groups of objects. *See* COMPOSITE pattern
- GUIs (graphical user interfaces)
 - defined, 441
 - kits, 175–180
 - MEDIATORS, 103–108
 - for menu commands, 252
 - MVC design in, 92–99
 - OBSERVERS in, 87–92
 - table components in, 30
 - for visualization, 198
- Handshaking classes, 67
- Head nodes in iteration, 314
- Headers in methods, 210
- Helm, Richard, 1
- Hierarchies
 - class, 63
 - FACTORY METHOD in, 171–173
 - for interpreters, 266–268
- Hooks, 224–225
 - for commands, 255–257
 - defined, 441
- Hoppers
 - defined, 441
 - purpose of, 221
- IDE (integrated development environment)
 - defined, 441
 - description of, 33
- Idea tool, 427
- Identifying adapters, 30–31
- Image proxies, 117–124
- images directory, 428
- Immutable objects, 417
 - defined, 441
 - in FLYWEIGHTS, 145–148
- Implementation
 - class, 11
 - defined, 441
- Infinite loops, 316
- Inner classes for FLYWEIGHTS, 150
- Inputs
 - in algorithms, 213
 - streams for, 287–295
- Instances
 - lazy-initialization of, 82–84
 - in Liskov Substitution Principle, 280
- Instantiating classes, 169–171
- Integrated development environment (IDE)
 - defined, 441
 - description of, 33
- Integrity, relational, 108–115
- Interface patterns. *See* ADAPTER pattern; BRIDGE pattern; COMPOSITE pattern; FACADE pattern
- Interfaces, 11
 - and abstract classes, 11–13
 - challenge solutions for, 347–348
 - defined, 441
 - dynamic proxies with, 131–132
 - marker, 348, 399
 - obligations for, 13–15
 - UML, 435–436
 - for visitors, 325
- INTERPRETER pattern, 261
 - challenge solutions for, 415–416
 - example, 261–274
 - languages and parsers for, 274–275
 - summary, 275
- Interpreters, 441
- Inverse binary relations, 110
- InvocationHandler interface, 132
- ITERATOR pattern, 305
 - challenge solutions for, 421–422
 - COMPOSITE enumerator depth in, 321–322
 - for COMPOSITES, 313–320
 - enumerating leaves in, 322–323

- ordinary, 305–307
 - summary, 324
 - thread-safe iteration in, 307–313
- Iterators
 - for class instantiation, 169
 - for collections, 167–168
- Java Development Kit (JDK)
 - for abstract classes, 25
 - defined, 442
- Java™ Programming Language*, 346
- JavaBeans API, 99
- JDBC
 - for database drivers, 69
 - defined, 442
- Johnson, Ralph, 1
- JTable class, adapting data for, 25–30
- JUnit framework
 - for code testing, 428
 - defined, 442
- Kits
 - defined, 442
 - GUI, 175–180
- Languages, interpreters for, 275
- Large images, loading, 117–124
- Law of Demeter (LOD)
 - defined, 442
 - in object-oriented design, 281–283
- Layers
 - defined, 442
 - in MVC design, 94–95, 98
- Lazy-initialization
 - of attributes, 248
 - defined, 442
 - of planners, 226–227
 - of SINGLETONS, 82–84
- Learning resources, 345–346
- Leaves
 - in COMPOSITES, 47
 - defined, 442
 - enumerating, 322–323
- Libraries, function wrappers for, 295–303
- Liskov Substitution Principle (LSP)
 - defined, 442
 - in object-oriented design, 279–281
 - for menu commands, 252
- Listeners
 - in Swing, 87
 - for visualization, 198
- Loaders for dynamic proxies, 132
- Loading large images, 117–124
- Locks
 - defined, 442
 - for multithreaded applications, 311–313
 - for objects, 83–84
- LOD (Law of Demeter)
 - defined, 442
 - in object-oriented design, 281–283
- Logical classes, 280
- Look-and-feel visualization, 175–180
- Loops
 - for, 305
 - infinite, 316
- Loose coupling, 137
 - defined, 442
 - mediators for, 108
- LSP (Liskov Substitution Principle)
 - defined, 442
 - in object-oriented design, 279–281
- Machines
 - anchoring chains for, 142–144
 - parallel hierarchies for, 171–173
 - refactoring chains for, 139–142
 - refactoring TEMPLATE METHODS for, 225–227
- Maintaining observable objects, 99–101

- Marker interfaces
 - defined, 442
 - purpose of, 348, 399
- Mass in rocket simulation, 12–13
- Mathematical functions, 299–301
- MEDIATOR pattern, 103
 - challenge solutions for, 372–376
 - GUI, 103–108
 - for relational integrity, 108–115
 - summary, 106
- Mediators for visualization, 198–200
- MEMENTO pattern, 193
 - challenge solutions for, 400–403
 - durability in, 201
 - persisting in, 201–205
 - summary, 205
 - for undo, 193–201
- Memory, loading large images into, 117–124
- Menu commands, 251–254
- Messages in Law of Demeter, 282
- Methods
 - for algorithms, 213–214
 - defined, 442
 - in Law of Demeter, 282
 - and operations, 209–211
 - polymorphism for, 214
 - in stub classes, 14
 - template. *See* TEMPLATE METHOD pattern
 - timing, 254–255
 - visibility of, 77–78
- Model/View/Controller (MVC)
 - design
 - defined, 443
 - for OBSERVERS, 92–99
- Modeling
 - states, 229–233
 - strategies for, 241–244
- Modifiers
 - for FLYWEIGHTS, 150
 - for methods, 210
 - for responsibility, 77–78
- Moles
 - defined, 443
 - in substances, 146
- Mortars
 - defined, 443
 - launching from, 56
- Moving chemical tubs, 104–108
- Multithreaded environments
 - safe iteration in, 311–313
 - in SINGLETONS, 83–84
- Mutexes
 - defined, 443
 - for thread control, 311–312
- MVC (Model/View/Controller)
 - design
 - defined, 443
 - for OBSERVERS, 92–99
- N*-tier systems
 - defined, 443
 - layers in, 98
- Nested type for FLYWEIGHTS, 150
- Nodes
 - in graphs, 50–55
 - in iteration, 314–319
- Notifications for visualization, 197–198
- NULL OBJECT pattern, 257
- Object adapters
 - defined, 443
 - working with, 21–25
- Object model relations, 109–111
- Object-oriented programs
 - concurrent, 83
 - Demeter Project for, 281–283
 - design principles for, 279

- mapping in, 108
 - responsibility in, 77
- Objects
- as clients, 17
 - families of, 182, 184
 - locking, 83–84
 - UML, 436–437
- Obligations for interfaces, 13–15
- Observable class, 95–96, 99–101
- OBSERVER pattern, 87
- challenge solutions for, 366–371
 - in GUIs, 87–92
 - maintenance in, 99–101
 - MVC design in, 92–99
 - summary, 101
- One-to-many dependencies, 87, 100
- Oozinoz Fireworks example company, 6–7
- defined, 443
 - source code for, 427–429
- Operations, 209
- algorithms and polymorphism, 213–214
 - challenge solutions for, 403–405
 - defined, 443
 - exceptions, 212
 - and methods, 209–210
 - signatures, 211
 - summary, 214–215
- Operations patterns. *See* COMMAND pattern; STATE pattern; STRATEGY pattern; TEMPLATE METHOD pattern
- Ordered pairs, 109
- Outputs
- in algorithms, 213
 - streams for, 287–295
- Packages
- in ABSTRACT FACTORIES, 184–185
 - as toolkits, 33
 - UML diagrams for, 4
- Parabolic flight paths
- plotting, 39–41
 - refactoring in, 35–38, 42–46
- Parallel hierarchies
- defined, 443
 - FACTORY METHOD in, 171–173
- Parametric equations
- defined, 443
 - for plotting curves, 39–41
- Parent machines, 138
- Parent nodes, 51
- Parsers
- for BUILDERS, 159, 161
 - defined, 443
 - INTERPRETERS for, 275
 - VISITORS for, 339
- Paths
- in COMPOSITES, 51
 - defined, 443
- Pattern Hatching: Design Patterns*
- Applied*, 346
- Patterns, 1–2
- defined, 443
 - inserting into codebase, 344–345
- Persistent storage
- defined, 444
 - for MEMENTOS, 201–205
- Polar coordinates, 40
- Polymorphism, 209
- algorithms and, 211
 - defined, 444
 - for methods, 214
 - in STATE refactoring, 237
 - in STRATEGIES, 249
- Portland Pattern Repository, 279
- Postorder traversal
- defined, 444
 - in iteration, 314
- Preorder traversal
- defined, 444
 - in iteration, 314

- Presses. *See* Star presses
- Printing for processes, 334–338
- Private access modifiers, 78
- Procedures. *See* Algorithms
- Protected access modifiers, 78
- PROTOTYPE pattern, 187
- challenge solutions for, 398–400
 - for clones, 189–191
 - factories in, 187–188
 - summary, 192
- PROXY pattern, 117
- challenge solutions for, 376–378
 - dynamic proxies, 131–136
 - image proxies, 117–124
 - remote proxies, 125–131
 - summary, 136
- Public access modifiers, 78
- Pull approach for observable objects, 99
- Push approach for observable objects, 99
- Random case
- defined, 444
 - example, 419
- Recognizing SINGLETONS, 84–86
- Recommendations
- modeling strategies for, 242–244
 - refactoring for, 244–248
- Recursive behavior in composites, 48–50
- Refactoring
- to CHAINS OF RESPONSIBILITY, 139–142
 - defined, 444
 - FACADES, 35–38, 42–46
 - for FLYWEIGHTS, 146
 - in Law of Demeter, 283
 - for MEDIATORS, 103–104, 107
 - in MVC, 92–93
 - for OBSERVERS, 91
 - to STATES, 233–237
 - to STRATEGY, 244–248
 - to TEMPLATE METHOD, 225–227
- Refactoring, 346
- Refactoring to Patterns, 346
- Refactoring Workbook, 346
- References, copying, 400
- Reflection
- defined, 444
 - for instantiation, 156
- Registry, RMI, 127–130
- Relational integrity, mediators of, 108–115
- Relations
- defined, 444
 - object model, 109–111
- Relationships, UML, 433–435
- Remote Method Invocation (RMI)
- defined, 444
 - for remote proxies, 125–131
- Remote proxies, 125–131
- repeat loops, 305
- Reservations
- changes to, 164–165
 - constraints for, 162–164
- Responsibility, 75
- challenge solutions for, 362–364
 - ordinary, 75–77
 - summary, 79
 - with visibility, 77–78
- Responsibility patterns. *See* CHAIN OF RESPONSIBILITY pattern; FLYWEIGHT pattern; MEDIATOR pattern; OBSERVER pattern; PROXY pattern; SINGLETON pattern
- Return types
- covariant, 404
 - for methods, 210–211
- Reusability of toolkits, 33
- Risks, VISITOR, 338–340
- RMI (Remote Method Invocation)
- defined, 444
 - for remote proxies, 125–131
- Robots, interpreters for, 261–274

- Rocket simulation
 - adapting data for JTable for, 25–30
 - class and object adapters for, 21–25
 - client needs for, 17–21
 - GUIs for, 87–92
 - interface for, 12–13
 - MVC design for, 92–99
 - obligations for, 13–15
 - remote proxies for, 125–131
 - sorting data for, 219–221
- Roman candles
 - defined, 444
 - described, 224
- Root nodes and objects
 - in anchoring chains, 142–144
 - defined, 444
 - in graphs, 51–52
- Selection of strategies, 248
- Sequences
 - of instructions, 213
 - visiting, 337
- Serialization, 205
- Services, command for, 254–255
- Sessions
 - defined, 444
 - persisting mementos across, 201–205
- Shallow copies
 - of collections, 313
 - vs. deep, 399
 - defined, 444
- Sharing FLYWEIGHTS, 148–152
- Shells, aerial
 - composites for, 56–60
 - defined, 439
 - ITERATORS for, 320, 322
 - VISITORS for, 334–335, 338
- ShowBallistics class, 89–94
- ShowBallistics3 class, 97
- ShowBetaVisualization class, 179
- ShowBrightness class, 302
- ShowCircle class, 41
- ShowComparator class, 219
- ShowConcurrentFor class, 309–311
- ShowConcurrentIterator class, 308–309
- ShowConcurrentMutex class, 311
- ShowDecorator class, 287
- ShowDown class, 269
- ShowDynamicProxy program, 134–136
- ShowFilters class, 293–294
- ShowFlight class, 36–38, 42, 103
- ShowFlight2 class, 42–46
- ShowForeach class, 306
- ShowFun class, 300
- ShowInterpreter class, 263
- ShowIterator class, 167
- ShowLowerCase class, 291
- ShowMachineTreeModel class, 328
- ShowOptionPane class, 34–35
- ShowPrettyVisitor class, 337
- ShowProcessIteration class, 319
- ShowProcessIteration2 class, 321
- ShowProxy class, 118
- ShowRakeVisitor class, 333
- ShowReflection class, 156–157
- ShowRocketClient class, 129–131
- ShowRocketTable class, 29–30
- ShowUnforgiving class, 163
- ShowVisualization class, 177
- ShowWhile class, 273
- Signatures
 - defined, 444
 - for methods, 210
 - operations, 211
- SINGLETON pattern, 81
 - challenge solutions for, 364–366
 - mechanics of, 81–83
 - recognizing, 84–86
 - summary, 86
 - threads in, 83–84
- Smells, code, 283
- Sorting, 217–221

- Source code, 427–429
- Splitting strings, 162
- SQL (Structured Query Language)
 - for database drivers, 69
 - defined, 445
- Stack for visualization state, 194–197
- Star presses
 - algorithms for, 221–224
 - defined, 445
 - hooks for, 255–257
 - unloading, 273–274
- Stars
 - in aerial shells, 56
 - brightness of, 301–303
 - defined, 445
- STATE pattern, 229
 - challenge solutions for, 407–408
 - constant, 238–240
 - modeling, 229–233
 - refactoring, 233–237
 - vs. STRATEGY, 248–249
 - summary, 240
- States and state information
 - defined, 445
 - MEMENTOS for, 193–197
 - UML, 230, 437–438
- Static methods, 33
- Storage, persistent
 - defined, 444
 - for MEMENTOS, 201–205
- Strategies, defined, 445
- STRATEGY pattern, 241
 - challenge solutions for, 409–410
 - for modeling, 241–244
 - refactoring in, 244–248
 - vs. STATE, 248–249
 - summary, 250
 - vs. TEMPLATE METHOD, 249
- Streams
 - defined, 445
 - designing, 287–295
- Strings
 - immutable, 146
 - parsing, 159
 - splitting, 162
 - tokenizing, 161
- Structured Query Language (SQL)
 - for database drivers, 69
 - defined, 445
- Stubs, 14, 30
- Subclasses
 - for class adapters, 21
 - covariant return types for, 404
 - function wrappers for, 297–299
 - for INTERPRETERS, 263
- Subsets, 109
- Superclasses, 280
- Surface area of fuel, 88
- Synchronizing
 - locks, 83
 - threads, 307, 313
- TEMPLATE METHOD pattern, 217
 - challenge solutions for, 405–407
 - for completing algorithms, 221–224
 - hooks in, 224–225
 - refactoring, 225–227
 - sorting, 217–221
 - vs. STRATEGY, 249
 - summary, 228
- Testing code, 428
- Text
 - parsing, 159
 - wrapping, 293
- Thick applications, 103
- Thread-safe iteration, 307–313
- Threads
 - in loading images, 120
 - in SINGLETONS, 83–84
- throw statement, 212
- throws clause, 210
- Thrust in rocket simulation, 13, 88–90

- Tiers
 - benefits of, 98
 - defined, 445
- Timing methods, 254–255
- Title case
 - defined, 445
 - example, 293–294
- Tokenizing strings, 161
- Toolkits, reusable, 33
- Traversing composite structures, 313–320
- Trees
 - abstract syntax tree, 339
 - in COMPOSITES, 50–56
 - defined, 445
- try/catch statement, 212
- Tubs
 - moving, 104–108
 - relational integrity for, 108–115
- UML. *See* Unified Modeling Language (UML)
- UML Distilled*, 431
- Undirected graphs, 51
- Undo, mementos for, 193–201
- UnicastRemoteObject class, 126
- Unified Modeling Language (UML)
 - classes in, 431–435
 - defined, 445
 - interfaces in, 435–436
 - notation in, 4
 - objects in, 436–437
 - relationships in, 433–435
 - state machines, 230
 - states in, 437–438
- Unified Modeling Language User Guide*, 346, 431
- Uniform Resource Locators (URLs), 445
- Unloading star presses, 273–274
- Utilities
 - for FACADES, 33–35
 - static methods in, 33
- Variables
 - for INTERPRETERS, 266
 - for STATES, 229
- Views in MVC design, 92–99
- Visibility and visibility modifiers
 - for FLYWEIGHTS, 150
 - for methods, 210
 - for responsibility, 77–78
- Visited nodes, 54–55
- VISITOR pattern, 325
 - challenge solutions for, 423–425
 - cycles in, 333–338
 - mechanics of, 325–327
 - ordinary, 327–333
 - risks in, 338–340
 - summary, 340
- Visualization
 - look-and-feel, 175–180
 - mementos for, 193–201
 - menu commands for, 252–254
 - persistence in, 201–205
- Vlissides, John, 1
- while loops, 305
- Work in process (WIP)
 - defined, 445
 - in SINGLETONS, 84
- Wrappers
 - for dynamic proxies, 132–136
 - function, 295–303
- Writers, 287–295
- XML (Extensible Markup Language), 159, 441