

# Index

---

## A

- abstract class, 25
- abstraction, 25, 34
- accessors
  - attributes, encapsulating with, 420–426
  - naming, 421
  - visibility of, 423, 424
- ACID properties, transaction
  - control, 464–465
- action response style, use case, 163–164
- activity diagrams, 29, 263, 270–277
- actors
  - defined, 139, 223
  - identifying, 146–148
- aggregation
  - and composition, 50–52
  - defined, 25
  - hierarchy defined, 25
- agile, starting, 136–139
- Agile Alliance 2001, 9–10
- Agile Model Driven Development (AMDD), 118–121, 402
- agile modeling (AM)
  - easing into, 109–118
  - overview, 107
  - principles, 109–112
  - techniques, 109–117
  - value of, 108
- agile software development, 9–10
- alpha testing, 97
- alternative action courses, 160–163
- analysis class diagrams
  - and modeling, 237–251
- analysis patterns, 254–256
- analytical reporting, 476
- architecture
  - change case model, 279, 289–291
  - composite diagram, 279
  - deployment diagrams, 279, 307–313
  - free-form diagram, 279, 306–307
  - layering, 314–317
  - network diagram, 279, 313
  - object-oriented software,
    - designing, 315
  - package diagram, 279, 291–296
  - purpose of, 278
  - reuse and, 283
  - techniques, 279–289

- associations
  - classes, 47
  - coupling and, 377
  - defined, 25, 45–48, 139
  - and dependency techniques, 368–372
  - implementing, 49
  - modeling, 245–247, 249
  - notation in UML, 245
- attributes
  - and accessors, 420–426
  - defined, 25
  - documenting, 363–364
  - format, UML, 360
  - Java instance attributes, declaring, 409–410
  - Java static methods, 412–416
  - modeling, 360–366
  - naming, 363
  - and operations/methods, 32–34
  - structured defined, 27
  - techniques for, 365
  - visibility, 361
- authentication, 471
- authorization, 471–473
- B**
- bank case study
  - banking system, objects in, 30
  - expansion of, 391
  - overview, 20–22
- behavior diagrams, 28
- beta testing, 97
- black box testing, 76, 89–90, 91
- book, organization of, 16–19
- boundary elements, 223
- boundary-value testing, 76, 90, 91
- brainstorming, 123
- brute force strategy, persistence, 453, 460, 473
- Business Entity analysis pattern, 254
- business intelligence strategy, 476
- business rules
  - Business Rules Markup Language (BRML), 211
  - defined, 210–214
  - Java, implementing, 439
  - Object Constraint Language (OCL), 211
  - requirements gathering, 212
  - Wiki pages, 212
- Business Rules Markup Language (BRML), 211
- C**
- cardinality, 25, 45
- change case model, 279, 289–291
- changes, cost of, 69–74
- Chaos report, 9
- checking account, 21
- class diagrams
  - shadow information and, 447
  - and structured design modeling, 351–380
  - UML, 29, 222
- classes
  - abstract vs. concrete, 43
  - and associations, 47
  - defined, 25, 30, 231
  - interface, implementing, 429–430
  - modeling, 31, 376–380
  - and objects, 28–31
  - reuse between, 247–249
  - structured defined, 27
  - subclasses, 26

## INDEX

535

- superclasses, 26
    - superclasses vs. subclasses, 38
  - classifier, 25
  - class-integration testing, 76, 93
  - class normalization, 242
  - class package diagrams, 292–294
  - class responsibility collaborator (CRC) cards, 231–237
  - code
    - blocks, 397
    - inspections, 94–95
    - inspection testing, 76
    - and MDA, 15
    - testing, 88–95
  - cohesion
    - defined, 25
    - and objects, 58–59
  - collaboration
    - defined, 25, 233
    - forms of, 232–234
    - and objects, 54–57
  - collaboration diagrams. *See*
    - composite structure diagrams
  - collaborator, 231
  - collisions, 462–464
  - Command Query Separation
    - Principle, 379
  - communication diagrams, 320
    - message, invoking, 336
    - UML, 29, 55, 334–337
  - components
    - component diagrams, 29, 296–306
    - defined, 25, 63–65
    - designing, 300–302, 305
    - testing, 76
  - composite structure diagrams, 320
    - ORM diagrams and, 350
    - UML, 29, 349
  - composition
    - and aggregation, 50–52
    - associations, modeling, 249
    - defined, 25
    - techniques, 372–373
  - conceptual domain modeling
    - and agility, 258–259
    - defined, 220
    - models, 222
  - concrete class defined, 25
  - concurrency control, 462–463
  - connascence, 379
  - constraints, 210, 215–216
  - contract models
    - creating, 484–485
    - formalizing, 482
  - control elements, 223
  - coupling
    - associations and, 377
    - composition and, 377
    - defined, 25
    - dependency and, 377
    - inheritance, direct via, 378
    - interfaces, 62
    - minimizing, 376–377
    - and objects, 57–58
    - realization and, 377
  - coverage testing, 76, 90, 91
  - CRC model, 222
  - current account defined, 21
- D**
- data
    - development, effective, 444
    - migration log, 480
    - modeling, logical, 222, 251–253
    - modeling, physical, 383–390

- data (*cont.*)
  - normalization rules, 385
  - package diagrams, 294
  - sharing defined, 284
- data access objects (DAOs), 453, 460
- database change log, 480
- database refactoring, 443–444, 471, 477–481, 486
- databases, relational
  - and agile development, 442–443
  - and data storage, 2
  - Java, 458–460
  - mapping objects to, 445–453
  - refactoring, 477–481
  - technology of, 6–7
- data flow diagrams (DFD), 263–268
- Demeter, Law of, 376
- dependencies
  - associations and, 368–372
  - coupling and, 377
  - Java, implementing, 438, 439
  - and object relationships, 52
- deployment diagrams
  - and architecture, 279
  - UML, 29, 307–313
- design patterns, 380–382
- design review testing, 76
  - techniques, modern, 8–15
  - technologies, modern, 2–8
  - test driven development (TDD)
    - AM and, 129
    - and AMDD, 402
    - defined, 97–99, 400–403
    - eXtreme programming (XP), 400
- diagrams. *See individual diagram by name*
- directionality defined, 46
- direction indicator defined, 246
- domain modeling. *See conceptual domain modeling*
- documentation
  - agile, 124–125
  - attributes, 363–364
  - defined, 101
  - methods, 417–419
  - stakeholder, agile, 205–207
- dynamic object modeling, 320–344
- dynamic system development
  - method (DSDM) and AM, 130
- E**
  - encapsulation defined, 25, 34
  - Enterprise Unified Process (EUP), 101
  - entity elements, 223
  - entity types, 227
  - essential use case
    - diagrams, 139–146
    - and usage modeling, 135, 136
    - writing, 149–151
  - EUP. *See Enterprise Unified Process (EUP)*
  - extend association, 168–175
  - extensibility, 63
  - eXtreme programming (XP)
    - and AM, 130
    - class responsibility collaborator (CRC) cards and, 231
    - and modeling, 101
    - spike solutions, 281, 282
    - and TDD, 400
- F**
  - Façade design pattern, 381–382
  - fact types, 227

**INDEX****537**

- fact type tables, 228
  - fat client approach, 2, 3
  - FDD. *See* Feature Driven Development (FDD)
  - Feature Driven Development (FDD)
    - AM and, 131, 180
    - AMDD and, 121
    - and modeling, 101
  - features
    - AM and, 180–181
    - and usage modeling, 135
  - feedback principle, 201
  - file management, 284
  - FLOOT. *See* Full Life Cycle Object Oriented Testing (FLOOT)
  - flow charts, 263, 268–270
  - free-form diagram, 279, 306–307
  - Full Life Cycle Object Oriented Testing (FLOOT), 75–78
    - black box testing, 76, 89–90, 91
    - boundary-value testing, 76, 90, 91
    - class testing, 76, 93
    - class-integration testing, 76, 93
    - code inspection, 76
    - components testing, 76
    - coverage testing, 76, 90, 91
    - design review, 76
    - inheritance-regression, 76, 93, 94
    - installation testing, 76
    - integration testing, 76, 90, 91
    - model review, 76, 85–87
    - path testing, 76, 90, 91
    - prototype review, 77, 84
    - prove it with code, 77
    - quality assurance, 79
    - regression testing, 77, 78
    - technical review, 77
    - unit testing, 90, 91
    - usage scenario testing, 77, 81–83
    - user interfaces testing, 77, 85
    - white-box testing, 77, 90, 91
  - function testing, 76, 95
- G**
- Gane and Sarson notation, 263
  - generalizing specialist, 488–490
  - getter method
    - defined, 420, 423
    - and lazy initialization, 425
  - glossary, 210, 217–218
- I**
- ICONIX, 131
  - ISO 12207, 9, 101
  - ILOG rules language, 211
  - implementation strategies,
    - data-oriented, 460–477
  - include associations, 169, 175
  - information gathering skills, 121–124
  - information hiding, 25, 35–37
  - infrastructure services, 284
  - inheritance
    - coupling, direct via, 378
    - defined, 25, 37–43
    - hierarchy defined, 25
    - and Java, 426
    - multiple defined, 26
    - and reuse, 247–249
    - single, 26
    - state diagrams, 343
    - structures, mapping, 447–449
    - techniques, 366–368
    - and use cases, 171

- inheritance-regression testing, 76, 93, 94
- initialization, lazy, 423, 432
- installation testing, 76, 96
- instance, 25
- instantiate, 26
- integration testing, 76, 90, 91
- interaction diagrams, 28
- interaction overview diagrams, 29, 320, 346–348
- interfaces
  - classes, implementing, 429–430
  - and coupling, 62
  - defined, 26, 62–63
  - defining, 428–429
  - introducing, 373–376
  - Java, implementing, 426–430
  - modeling, UML class diagram, 373
  - naming, 374
  - overview, 298–300
  - sources of, 374
  - uses for, 375
- inter-process communication (IPC), 284
- interviewing skills, 121
- J**
- Java
  - business rules, 439
  - classes, 408
  - class diagrams to, 406
  - collaborations, 439
  - composition, 436–438
  - constructors, implementing, 419–420
  - and databases, 458–460
  - dependencies, 438, 439
  - inheritance, 426
  - instance attributes, 409–410
  - instance methods, 410–412
  - interfaces, 426–430
  - object design to, 404–440
  - relationships
    - classes, between, 436
    - implementing, 429–438
    - many-to-many, 434–435
    - one-to-many, 432–434
    - one-to-one, 432
    - recursive, 436
    - unidirectional & bidirectional, 431
  - sequence diagrams to, 404–406
  - static methods and attributes, 412–416
- K**
- keys
  - natural, 387
  - surrogate, 387, 388
  - tables, assigning, 387
- knowledgebases, 229
- L**
- Law of Demeter, 376
- learning process, 490–492
- legacy analysis, 481–485
- Liskov Substitution Principle, 366, 370
- locking strategies, 462–463
- logic
  - and sequence diagrams, 321
  - shared and referential integrity (RI), 466–471
- logical data model (LDM), 222, 251–253
- Lowest Common Denominator Principle (LCD), 466

## INDEX

539

**M**

- manual processes and AM, 157
- mapping
  - databases, objects to, 445–453
  - inheritance structures, 447–449
  - relationships, 451–453
- messages, 26
  - and collaboration, 56
- methods
  - defined, 26, 32
  - documenting, 417–419
  - logic of, and sequence diagrams, 321
  - names and cohesiveness, 58
  - structured, 27
  - techniques for, 358–359
  - testing, 76, 92, 93
  - visibility, 355–357
- Model Driven Architecture (MDA)
  - code and, 15
  - overview, 14–15
- modeling
  - agile. *See* Agile Modeling
  - analysis class diagrams, 237–251
  - associations, 245–247
  - attributes, 360–366
  - class diagrams, 351–380
  - classes, 31, 241–245, 376–380
  - class responsibility collaborator (CRC) cards, 231–237
  - communication diagrams, 320
  - composite structure diagrams, 320
  - composition associations, 249
  - conceptual domain, 220, 222, 258–259
  - data, logical, 251–253
  - data, physical, 383–390
  - data flow diagrams (DFD), 263–268
  - data flow diagram (DFD) rules, 266
  - defined, 101
  - design, methods, 352–359
  - dynamic object modeling, 319
    - and Enterprise Unified Process (EUP), 101
    - and eXtreme programming (XP), 101
  - essential use case, 135, 136
    - and Feature Driven Development (FDD), 101
  - features and usage modeling, 135
  - flow charts, 263, 268–270
  - interaction overview diagrams, 320
  - philosophies, 102–105
  - positions vs. actors, 147
  - process modeling, 263–270
    - and Rational Unified Process (RUP), 101, 262
  - responsibilities, 241–245
  - sequence diagrams, 320
  - state machine diagram, 320, 337–344
  - structured design modeling, 351–380
  - system use cases, 135, 136
  - timing diagrams, 320–321, 334
  - unknowns, 48
  - usage modeling, 134–176
  - use case diagrams and, 135
  - use cases, 134–176, 177
  - visual, 105
    - vocabularies, 250–251
- models, testing, 80–88
- modularity, 63

- multiple inheritance, 26
- multiplicity
  - defined, 46
  - indicators, UML, 246, 369
- N**
- naming
  - accessors, 421
  - attributes, 363
  - interfaces, 374
  - methods, 357
- natural keys, 387
- network diagram, 279, 313
- n-tiered approach, 2, 3
- O**
- Object Constraint Language (OCL)
  - and business rules, 211
  - overview, 216
- object diagrams, 29, 222, 256–258
- object-oriented concepts
  - overview, 24
  - programming, purpose of, 393
  - structured, 27–28
- object-oriented testing techniques, 92–93
- Object Role Model (ORM) diagram
  - composite structure diagrams, 350
  - defined, 222, 227–230
  - entity types, 227
  - fact types, 227
  - fact type tables, 228
- objects
  - and classes, 28–31
  - cohesion and, 58–59
  - collaboration, 54–57
  - and coupling, 57–58
  - defined, 26, 28
- dependencies and object relationships, 52
- inventory control system, 30
- and Java, 404–440
- mapping to databases, 445–453
- persistent, 26, 43
- searching, 473–477
- structured defined, 27
- and UML diagrams, 29, 222, 256–258
- object technology
  - overview, 2–5
  - purpose of, 2
- observation skills, 123
- operation signature, 353
- operations testing, 76, 96
- optionality, 26, 45
- override defined, 26
- P**
- package diagrams
  - architecture model, 279
  - class, 292–294
  - data, 294
  - model, 279
  - UML defined, 30, 291–296
  - use cases, 294–296
- package method
  - attribute visibility and, 362
  - and method visibility, 356
- packages
  - defined, 140
  - and use cases, 172–173
- path testing, 76, 90, 91
- pattern, 26, 65–66
- persistence
  - code
    - and brute force strategy, 453, 460



## INDEX

541

data access objects (DAOs)  
     and, 453, 460  
     implementing, 453–455  
 defined, 26, 43–44, 284  
 frameworks  
     meta-data driven, SQL, 474  
     and persistence code, 454  
 persistent object, 26  
 pilot testing, 97  
 platform independent model  
     (PIM), 14  
 platform specific model (PSM), 15  
 poker game and polymorphism,  
     59  
 polymorphism  
     defined, 26, 59–62  
     poker game and, 59  
 ports, overview, 298–300  
 printing, 284  
 private method  
     attribute visibility and, 362  
     and method visibility, 356  
 process models, 262  
 programming, effective, 394–399  
 project stakeholders, 106–107  
 property, 26, 50  
 protected method  
     attribute visibility and, 362  
     and method visibility, 356  
 prototype review testing, 77, 84  
 prototyping, 193  
 prove it with code testing, 77  
 public method  
     attribute visibility and, 362  
     and method visibility, 356

**Q**

qualifier, 46  
 quality assurance, 79

**R**

Rational Unified Process (RUP). *See*  
     *also* Unified Process (UP)  
     AM and, 131  
     and modeling, 101, 262  
     robustness analysis and, 221  
     specifications, supplementary  
         and, 218  
 RDB. *See* databases, relational  
 refactoring, 394  
 refactoring, databases, 443–444,  
     471, 477–481, 486  
 referential integrity (RI), 437,  
     466–471  
 regression testing, 77, 78  
 relationships  
     defined, 44–54  
     many-to-many, 434–435, 452  
     mapping, 451–453  
     one-to-many, 432–434, 452  
     one-to-one, 432, 451  
     recursive, 436, 453  
     unidirectional & bidirectional,  
         431  
 reports, generating, 476–477  
 requirements  
     gathering  
         and business rules, 212  
         process, goals of, 163  
     supplementary, 210  
 responsibility, 231, 232  
 reuse  
     architecture, 283  
     between classes, 247–249  
     classes, between, 247–249  
     and inheritance, 247–249  
     principle of user interface design  
         defined, 201  
     use cases, 167–172

robustness analysis, 221  
 robustness diagram, 221–227  
 role, 46  
 RUP. *See* Rational Unified Process (RUP)

**S**

savings account, 20  
 scaffolding code, 434–436  
 security access control (SAC), 471–473  
 security, 284  
 sequence diagrams  
   and dynamic object modeling, 320  
   Enroll in University use case, 322, 324, 326  
   to Java, 404–406  
   logic methods and, 321  
   and logic services, 321  
   service level, 324  
   UML defined, 30, 53, 320–321, 334, 405  
   usage scenario, 321  
   visual coding and, 332  
 sequence numbers, format of, 335  
 services, 321  
 setter method, 420, 423  
 shadow information defined, 446–447  
 simplicity principle, 201  
 single inheritance, 26  
 Singleton design pattern, 66, 380  
 specifications, supplementary, 218  
 SQL  
   brute force strategy, 473  
   design to code, 455–460  
   persistence framework, 474

  query objects, 473  
 state chart diagrams. *See* state machine diagram  
 state machine diagram  
   and inheritance, 343  
   and dynamic object modeling, 320, 337–344  
 states, representing, 339  
 stereotype, 26  
 stress testing, 77, 96  
 structural design modeling, 351  
 structure diagrams, 28  
 structured paradigm, overview, 4  
 structured principle, 201  
 subclasses  
   defined, 26  
   and superclasses, 38  
 superclasses  
   defined, 26  
   and subclasses, 38  
 support testing, 96  
 surrogate keys, 387, 388  
 system boundary boxes, 140  
 system management, 284  
 system testing, 95–96  
 system use cases, 153–160  
   diagrams, 153  
   and flow diagramming, 198  
   usage modeling and, 135, 136

**T**

tables, assigning keys to, 387  
 Taylorism, 488  
 TDD. *See* test driven development (TDD)  
 technical requirements, 210, 214–215  
 technical review testing, 77  
 test driven development (TDD)

## INDEX

543

- AM and, 129
  - and AMDD, 402
  - defined, 97–99, 400–403
  - eXtreme programming (XP), 400
  - testing
    - alpha, 97
    - beta, 97
    - black box, 76, 89–90, 91
    - boundary-value, 76, 90, 91
    - class, 76, 93
    - class-integration, 76, 93
    - code, 88–95
    - code inspection testing, 76
    - components, 76
    - concepts, traditional, 89–91
    - and the cost of change, 69–70
    - coverage, 76, 90, 91
    - design review, 76
    - FLOOT, 76
    - function, 76, 95
    - inheritance-regression, 76, 93, 94
    - installation, 76, 96
    - integration, 76, 90, 91
    - methods, 76, 93
    - model review, 76, 85–87
    - models, 80–88
    - object-oriented techniques, 92–93
    - operations, 76, 96
    - path, 76, 90, 91
    - philosophies, 74
    - pilot, 97
    - prototype review, 77, 84
    - prove it with code, 77
    - regression, 77, 78
    - stress, 77, 96
    - support, 96
    - system, 95–96
    - technical review, 77
    - terminology, 88
    - tools, 89
    - usage scenario, 77, 81–83
    - user, 96–97
    - user interface, 77, 85
    - white-box testing, 77, 90, 91
  - timing diagrams
    - and dynamic object modeling, 320–321, 334
    - seminar case study, 344, 345
    - UML defined, 30, 344–346
  - tolerance principle, 201
  - transaction control, 464–466
  - transaction management defined, 284
  - transcripts, outputting, 330
  - transitions
    - depicting, 342
    - labels, format for, 339
  - transitory object, 26
- U**
- Unified Modeling Language (UML)
    - activity diagrams, 20, 263, 270–277
    - associations, notation, 245
    - attribute format, 360
    - behavior diagrams, 28
    - class diagrams, 29, 222
    - communication diagrams
      - defined, 29, 55, 334–337
      - and dynamic object modeling, 320
      - message, invoking, 336
    - component diagrams
      - defined, 29, 296–306
    - composite structure diagrams
      - defined, 29, 349

- Unified Modeling Language (*cont.*)
  - and dynamic object modeling, 320
  - ORM diagrams and, 350
  - data modeling, 251–253, 383–390
  - deployment diagrams
    - defined, 29, 307–313
    - model, architecture defined, 279
  - diagrams, overview, 28–30
  - interaction diagrams, 28
  - interaction overview diagrams
    - defined, 29, 346–348
    - and dynamic object modeling, 320
  - interface modeling, UML class diagram, 373
  - multiplicity indicators, 246, 369
  - object diagrams, 29, 222, 256–258
  - operation signature format, 353
  - overview, 11–12
  - package diagrams
    - architecture model, 279
    - class, 292–294
    - data, 294
    - defined, 30, 291–296
    - model, 279
    - use cases, 294–296
  - sequence diagrams
    - defined, 30, 53, 320–321, 334, 405
    - and dynamic object modeling, 320
    - Enroll in University use case, 322, 324, 326
    - to Java, 404–406
    - and logic services, 321
    - service level, 324
    - usage scenario, 321
    - visual coding and, 332
  - state machine diagrams, 320, 337–344
  - structure diagrams, 28
  - timing diagrams
    - defined, 30, 344–346
    - and dynamic object modeling, 320–321, 334
  - use case diagrams, 30, 135
- Unified Process (UP). *See also* Enterprise Unified Process (EUP); Rational Unified Process (RUP)
  - overview, 13–14
  - technical prototypes, 281
- unit testing, 90, 91
- unknowns, modeling, 48
- update log, 480
- usage modeling, 134
- usage scenario
  - defined, 152
  - and sequence diagrams, 321
  - testing, 77, 81–83
- use cases
  - defined, 139, 223
  - diagrams
    - UML defined, 30
    - and usage modeling, 135
  - essential vs. system, 164–167
  - identifying, 151–153
  - modeling, 134–176, 177
  - package diagrams, 294–296
  - reuse in, 167–172
- user acceptance testing (UAT), 97
- user interfaces
  - design strategies, 200–204

## INDEX

545

development  
  artifacts, 185  
  overview, 184  
  prototyping, 185–197  
flow diagramming, 197–199  
prototyping, essential vs.  
  traditional, 185  
testing, 77, 85  
tolerance principle, 201  
usability, 199  
visibility principle, 201  
user stories, 177–180  
utility computing, 7

**V**

visibility principle, 201  
visual coding and sequence  
  diagrams, 332  
visual modeling, 105  
vocabularies, modeling,  
  250–251

**W**

Web services  
  overview, 7–8  
  utility computing, 7  
*Whiteboard Photo*, 127  
white boards, 125–129  
white-box testing, 77, 90,  
  91  
whitespace, 398  
Wiki pages, 212

**X**

XML  
  overview, 5  
  purpose of, 2  
XP. *See* eXtreme programming  
  (XP)

**Z**

Zachman Framework (ZF),  
  285–288