INDEX

== operator, primitive types, 440–441

A
Abbott (linguistic) analysis, conceptual models, 108–109
accented characters, text search performance, 405–406
Account.java listing, distributed
Observer chains, 129
ACLs (access control lists), application servers, 323, 324–326
advertisement and discovery, Event Notifier, 143–144
agent hosts, mobile agent system architecture, 170
agent life cycles, mobile agent system architecture, 171
agent standards, mobile agent system architecture, 176
agent systems, mobile agent system architecture, 170
AgentHost.java listing, mobile agent system architecture, 173
Agent.java listing, mobile agent system architecture, 177
aggregation, conceptual models, 119
Aglets Workbench, mobile agent system architecture, 176
Algorithm object, JavaSoft security package, 309–311, 315–316
algorithm-specific initialization, JavaSoft security package, 300, 301, 308–309, 310–311, 315, 316
analysis patterns, 155–166
See also modeling and patterns
assigning ranges automatically, 162–166

CategoryObservation class, 159, 160
Measurement pattern, 158–162
multiple classification, 160–161
Observation pattern, 159–161
overview, 155
Phenomenon class, 159, 160
Phenomenon Types, 158–159
Quantity pattern, 157
Range pattern, 162–166
representing quantities, 155–156
single classification, 160, 162
applets, enterprise. See enterprise applets and archive functionality
application servers, 323–335
See also security
ACLs (access control lists), 323, 324–326
bridges, 331–332
capability approach, 323–324, 327–328
confinement problem, 333–334
Confused Deputy problem, 326
delusions, 325–326
granting tickets, 330–333
language-based security, 324
odometer capabilities and interfaces, 327–328
overview, 323–324
post-processor generated code, 329
principals, 324–325
protection proxy design pattern, 324, 325
retro interaction, 332–333
summary, 334
tgen program, 328–330
ticket class generation, 328–330
ticket construction, 330
applications
complex performance. See complex applications performance
enterprise-critical test strategies, 339–350
archive functionality. See enterprise applets and archive functionality
arguments, passing by reference versus by value, 442–443
arithmetic. See Math package
arrays, primitive types and, 451
Assertion class, DBC (Design by Contract), 74, 76, 79
assertions
DBC (Design by Contract), 75, 77–78, 79
JavaBeans testing, 352–355
assigning ranges automatically, analysis patterns, 162–166
assignments, multithreaded. See multithreaded assignment surprises
associations, conceptual models, 109–110, 111
asynchronous notification, Event Notifier, 147
attributes
conceptual models, 111, 116
defined, 111
auto-delegation, creeping featurism (Java paradigm), 8–9

B
base direction, international text in JDK 1.2, 90
BGSound applet (JavaScript), 33–35 “interface” methods, 34
methods, 34–35
BGSound.htm listing, JavaScript, 41–42
BGSound.java listing, JavaScript, 42–45
BigDecimals class, Math package, 461–463
binding IORs to Names, CORBA objects (locating from Java), 196–198
black box testing, enterprise-critical test strategies, 343
block ciphers, JavaSoft security package, 312
Blotter.java listing, distributed Observer chains, 130
bots. See mobile agent system architecture
boundary checks, enumerating types with class, 19
bounding box, TextLayout class (international text), 99
Boyer-Moore algorithm, text search performance, 414–418
Boyer-Moore versus Unicode, text search performance, 416–418
bridges, application servers and, 331–332
broad-based client/server applications and network management, migrating to Java, 29–30
broadcasting, channel deployment of Java client applications, 185–187
browser deployment of Java client applications, 180–183
See also deployment strategies for Java client applications
advantages of, 181–182, 188
class files, 181
combining with channel deployment, 187–189
disadvantages of, 182–183, 188
overview, 180–181
runtime performance, 182
summary, 183
BrowZaInfo applet, calling JavaScript from Java, 38–40
BrowZaInfo.htm listing, JavaScript, 47
BrowZaInfo.java listing, JavaScript, 46–47
BufferedReader object, multithreaded exception handling, 234–235
bytecode interpreters, complex applications performance, 382
bytecodes, complex applications performance, 381–382

C
C++, Event Notifier and run-time information, 142
call site optimization, complex applications performance, 386, 387
Callbacks, Completion, 236–237
calling Java from JavaScript, 32–33
LiveConnect (Netscape Navigator), 32–33
public methods, 33
calling JavaScript from Java, 36–40
  BrowsaInfo applet, 38–40
eval method, 38–39, 40
  JSObject class, 36–37
  JSObject methods, 37
JSObject variable, 36
  MASCIIPT attribute, 38
calling objects, defined, 206
capability approach, application servers, 323–324, 327–328
caret display, TextLayout class (international text), 92–94
caret movement, TextLayout class (international text), 97–98
catastrophic cancellation, Math package, 457
CategoryObservation class, analysis patterns, 159, 160
CBC (Cipher Block Chaining), JavaSoft security package, 310
CHA (Class Hierarchy Analysis), complex applications performance, 384–385
chained locking, complex synchronization, 242, 244, 245
changed methods, modular user interfaces, 258
channel deployment of Java client applications, 185–187
  See also deployment strategies for Java client applications
  advantages of, 186, 188
  combining with browser deployment, 187–189
  disadvantages of, 186–187, 188
  overview, 185–186
summary, 187
  transmitters, 187
  tunnels, 185–187
CheckedThreadException class, multi-threaded exception handling, 233–236
Cipher Block Chaining (CBC), JavaSoft security package, 310
class files, browser deployment of Java client applications, 181
Class Hierarchy Analysis (CHA), complex applications performance, 384–385
class invariant conditions, DBC (Design by Contract), 72–73
class/static methods, conceptual confusion (Java paradigm), 6
classes, enumerating types with, 15–20
client applications, deploying. See deployment strategies for Java client applications
  client/server applications and network management, broad-based, 29–30
  client/server customer service applications, migrating to Java, 30
  clients, returning values to, 211
code reuse, complex applications performance, 380–381
collaborations, Event Notifier, 136–137
collation elements, text search performance, 407–409
CollationElementIterator class, text search performance, 407–408, 413–414
Collator class, text search performance, 406–407, 409
combinations. See counting combinations; generating combinations; permutations and combinations
  Combinatoric.java listing, counting permutations, 52, 53–54
  combining characters, text search performance, 411–412
  comparator, enterprise-critical test strategies, 345
Completion Callbacks, multithreaded exception handling, 236–237
complex applications performance, 379–389
  See also performance bytecode interpreters, 382
  bytecodes, 381–382
call site optimization, 386, 387
CHA (Class Hierarchy Analysis), 384–385
code reuse, 380–381
DFA (Data Flow Analysis), 385
  inheritance, 380
  inlining, 386–389
JIT (Just-In-Time) VMs, 382–383
JOVE compiler, 384–385
Index

JVM (Java Virtual Machine)
iniciencies, 381–383
native compilation, 383–384
object-oriented language inefficiencies, 380–381
object-oriented optimizations, 384–389
overview, 379
polymorphism, 380–381
PSR (Polymorphic Strength Reduction), 385–386
PTE (Polymorphic Type Elimination), 385–386
reasons for poor performance, 379
static compilation, 383
summary, 389

complex synchronization, 241–252
See also threads
chained locking, 242, 244, 245
condition variables, 246–252
condSignal() method, 246–248
condWait() method, 246–248
CV class, 246–247, 251
monitors, 241
mutexes, 241, 243–244, 245
non-synchronized updates, 241, 242
POSIX threads and Java threads, 243
producers/consumers, 246–248
protection, 241–244
Pthreads, 241–242
RWLocks class, 251–252
synchronized updates, 241, 242
Try/Catch technique, 250
wake up order, 249
when to run, 244–245

components of collusion elements, text search performance, 407, 408–409
ComponentUI, Swing components, 270, 271

compression libraries. See package ZIP
conceptual confusion (Java paradigm), 4–7
class/static methods, 6
no-static interface methods, 7
pass-by-reference, 5
virtual methods, 4–5
conceptual models, 105–120
See also modeling and patterns
Abbott (linguistic) analysis, 108–109

aggregation, 119
associations, 109–110, 111
attributes, 111, 116
common association categories, 111
common concept categories, 108, 109
creating, 107–108
defined, 106–107
dimensions of decomposition, 105–106
elements, 106, 113
generalization/specialization, 118
identifying associations, 109–110
identifying concepts, 108–109
linguistic (Abbott) analysis, 108–109
model quantities as quantities, 117
multiplicity, 111, 112
overview, 105–107, 115–116
partitioning, 119–120
published models, 109
quantities, 117
refining skills for expressive, 115–120
relating to artifacts and Java, 112
specialization/generalization, 118
text versus concepts, 116–117
UML (Unified Modeling Language), 115
congeate methods, enumerating types with class, 19
condition variables, complex synchronization, 246–252
condSignal() method, complex synchronization, 246–248
condWait() method, complex synchronization, 246–248
confinement problem, application servers, 333–334
Confused Deputy problem, application servers, 326
constant-parameters support, creeping featurism (Java paradigm), 9–10
constraint language, DBC (Design by Contract), 74–75
containers, primitive types and, 452–453
contracts, DBC (Design by Contract), 69–86
CORBA (Common Object Request Method Invocation), 391
See also distributed objects
performance
Event Notifier comparison, 153
IDL (Interface Definition Language), 399
method invocation time, 399–401
mobile agent system architecture and, 170
performance of, 397–401
RMI comparison, 391–394, 401–402
CORBA objects (locating from Java), 191–202
binding IORs to Names, 196–198
bootstrapping CORBA applications, 191
connecting to Naming Service, 192–193
CosNaming.idl, 193–194, 201–202
Factory Object, 201
“federated” Naming Service, 200–201
implementing CosNaming module, 201–202
IORs (Interoperable Object References), 191, 195
Names, 195–198
Naming Service, 192–202
navigation, 199–200
options, 199
overview, 191
resolving IORs from Names, 198
summary, 202
CosNaming.idl, CORBA objects (locating from Java), 193–194, 201–202
counting combinations, 54–55
See also generating combinations; permutations and combinations factorials, 54–55
counting permutations, 51–54
See also generating permutations; permutations and combinations Combinatoric.java listing, 52, 53–54
factorials, 52
creeping featurism (Java paradigm), 7–14
auto-delegation, 8–9
constant-parameters support, 9–10
dynamic “static,” 13–14
first-class methods, 8
in/out parameters, 10–11
inlined getters and setters, 11
multi-valued return parameters, 11–12
overview, 7–8
CriticalFaultFilter class, Event Notifier, 150–151
Cryptography Architecture. See JCA (Java Cryptography Architecture)
Cryptography Extensions. See JCE (Java Cryptography Extensions)
cursive forms, international text in JDK 1.2, 88–89
customer service applications, client/server, 30
CV class, complex synchronization, 246–247, 251

D
Data Flow Analysis (DFA), complex applications performance, 385
data-dictionary, enterprise–critical test strategies, 341
DBC (Design by Contract), 69–86
Assertion class, 74, 76, 79
assertions, 75, 77–78
assertions and Java, 79
class invariant conditions, 72–73
constraint language, 74–75
defined, 69
developer benefits, 75–77
Exception classes, 82
exception mechanism, 79–81
extendibility, 76
iContract tool, 83
implementation methods, 84, 85
inheritance, 73–74, 78, 81–82
interfaces, 78–79
Java and, 77–78
“Law of (software) Acquisition,” 76–77
obligations and benefits, 71–73
open-closed principle, 75
overview, 69–71
postconditions, 70–71
preconditions, 70
proposed language extension, 83–84
reusability, 76
robustness, 76
DBC (continued)
semantics, 69
summary, 84–86
traceability, 76
working out contracts, 73
decomposition, conceptual models, 105–106
Decoupler. See Event Notifier
decryption, JavaSoft security package, 311–312
Deflater class, enterprise applets and
archive functionality, 422–423
DELAY parameter, JSONObject applet para-

ters, 32
delegation event model, modular user
interfaces, 257, 259–260
delegations, application server, 325–326
deployment strategies for Java client ap-

clications, 179–189
browser deployment, 180–183
channel deployment, 185–187
combining browser and channel de-

ployment, 187–189
options, 180
overview, 179
stand-alone deployment, 183–185
summary, 189
summary of options, 188
traditional deployment, 179–180
deprecation, primitive types, 449–450
DES encryption, JavaSoft security pack-

age, 306–311
Design By Contract. See DBC
DESKeySpec class, JavaSoft security
package, 308–309
developer benefits, DBC (Design by
Contract), 75–77
developer-tester interactions, enterprise-
critical test strategies, 347–348
DFA (Data Flow Analysis), complex ap-
lications performance, 385
digital envelopes (JavaSoft security
package), 312–322
algorithm object, 315–316
code listing, 318–322
opening, 317
RSA algorithm, 314–315
digital signatures, enterprise applets and
archive functionality, 422
digital signatures (JavaSoft security
package), 298–302
computing, 301–302
verifying, 302
dimensions of decomposition, concep-
tual models, 105–106
discovery and advertisement, Event No-
tifier, 143–144
dispatch threads, defined, 206
Dispatcher. See Event Notifier
distributed event notification, Event
Notifier, 144–146
distributed Java, 167–202
deployment strategies for Java client
applications, 179–189
locating CORBA objects from Java,
191–202
mobile agent system architecture,
169–178
distributed objects performance, 391–403
CORBA performance, 397–401
CORBA versus RMI, 391–394
native Java performance, 393–394
overview, 391
RMI performance, 394–397
summary, 402–403
testing method, 392–393
distributed Observer chains, 121–130
See also modeling and patterns
Account.java listing, 129
Blotter.java listing, 130
IDL (Interface Definition Language),
122, 126–127
implementation, 123, 126
IOR (Interoperable Object
Reference), 123
ModelView.idl listing, 123
Object Reference, 123
Observable.java listing, 124
Observer Chaining, 125–129
ORB (Object Request Broker), 122
overview, 121–122
references, 122–123
Stringified Object Reference, 123
summary, 129–130
division, truncated integer, 464
doFinal() method, JavaSoft security
package, 311
drawing, TextLayout class
E
early life cycle testing, enterprise-critical test strategies, 339–350
encapsulating locking mechanisms in separate classes, multithreaded assignment surprises, 221–222
encryption. See JavaSoft security package enterprise applets and archive functionality, 421–431
See also performance
Deflater class, 422–423
digital signatures, 422
future of, 430
getBytes() method, 425, 426–427
getImage() method, 429
getSize() method, 427
getZipInputStream() method, 425, 426
Inflater class, 422–423
JAR (Java Archive) files, 421–422
JZipLoader class, 425–430
overview, 421
package ZIP, 422–425
package ZIP and enterprise applets, 423–425
read() method, 427
readZipEntry() method, 426, 427, 428
resizeBuffer() method, 427–429
summary, 431
ZipEntry class, 423
ZipFile class, 423
enterprise-critical test strategies, 339–350
See also testing
application overview, 339–341
black box testing, 343
comparator, 345
data-dictionary, 341
eyearly life cycle testing, 341–342
early life-cycle issues and developer-tester interactions, 347–348
error handling and range testing, 346–347
GUI integration testing, 345
GUI separation from backend, 340, 348
Java standards, 349
JavaStar issues, 349–350
lessons learned, 347–350
life cycle testing, 341–342
overview, 339
range and error handling testing, 346–347
Reference Spreadsheet, 341, 342–343
Reference Spreadsheet issues, 348
setName() method, 349
sub-system level testing of backend, 342–345
summary, 350
system testing, 346
test approach, 341–347
test data generation, 343–344
test drivers, 344
test strategy, 342
toString() method, 349
use case testing, 346
enumerating types with class, 15–20
boundary checks, 19
concrete methods, 19
enum keyword, 15
implementation, 17–18
instantiation, 18, 19–20
months, 16–20
overview, 15–16
private array, 17
private attributes, 16
private constructor, 16
public methods, 18, 19
type-safe enumeration object, 20
envelopes, digital. See digital envelopes error handling, multithreaded. See multithreaded exception handling error handling and range testing, enterprise-critical test strategies, 346–347
eval method, calling JavaScript from Java, 38–39, 40
event hierarchies, Event Notifier, 140–141
event listeners, modular user interfaces, 262

Event Notifier, 131–153
  See also modeling and patterns
accessing event services, 139
advertisement and discovery, 143–144
applicability, 134–135
asynchronous notification, 147
C++ and, 142
classes, 136
collaborations, 136–137
consequences, 137–139
CORBA comparison, 153
CriticalFaultFilter class, 150–151
described, 132
determining event types at run time, 141–142
discovery and advertisement, 143–144
distributed event notification, 144–146
dynamic typing, 152
Event class, 150
event hierarchies, 140–141
event proxies, 144–146
event types, 138–139, 141–143, 152
EventService class, 148–149
fault tolerance and performance, 146–147
filtering events, 138, 150
implementation, 139–147
JavaBeans comparison, 153
managed objects, 132
mediators, 133, 134
modeling events, 139–141
need for, 132–134
NIH class library and, 142
Observer design pattern, 133–134
overview, 131
Paging System subscriber, 151
proxies, 144–146
publishers, 138, 141, 151
push and pull semantics, 147
reclaiming type safety, 142–143
related patterns, 153
RemoteEventService class, 144
results, 137–139
Router class (publishers), 151
sample code, 148–152

structure of, 135–136
subscriber anchoring, 147
subscribers, 151
subscriptions, 137–138
transiency, 138

Exception classes, DBC (Design by Contract), 82
exception handling, multithreaded. See multithreaded exception handling
exception handling and range testing, enterprise-critical test strategies, 346–347
exception mechanism, DBC (Design by Contract), 79–81
exceptionOccurred() method, multithreaded exception handling, 235, 239
exceptions, listening for, 229–230

ExitButtonController class, modular user interfaces, 259

Explorer, calling Java from JavaScript, 33
extendibility, DBC (Design by Contract), 76

F
factorials
  counting combinations, 54–55
  counting permutations, 52

Factory object, CORBA objects (locating from Java), 201
fault tolerance and performance, Event Notifier, 146–147
featurism. See creeping featurism
  “federated” Naming Service, CORBA objects (locating from Java), 200–201
feedback mode, JavaSoft security package, 310

FileNotFoundException class, multithreaded exception handling, 225–226, 227, 235

FileOpener class, multithreaded exception handling, 225–226, 233–234, 237
filtering events, Event Notifier, 138, 150
first-class methods, creeping featurism (Java paradigm), 8

FlatUI, Swing components, 273, 277–278, 279–280
G

generalization/specialization, conceptual models, 118

generate-and-test technique, permutations and combinations, 60–61
generatePublic() method, JavaSoft security package, 315
generateSecret() method, JavaSoft security package, 308–309
generating combinations, 59–60

  See also counting combinations; permutations and combinations

generating permutations, 56–58

  See also counting permutations; permutations and combinations

GenericAgentHost.java, mobile agent system architecture, 174–175
getBytes() method, enterprise applets and archive functionality, 425, 426–427
getImage() method, enterprise applets and archive functionality, 429
goFFset() method, text search performance, 413–414

getSeed() method, JavaSoft security package, 297
ggetSize() method, enterprise applets and archive functionality, 427

getters, inlined, 11
getZipInputStream() method, enterprise applets and archive functionality, 425, 426
glyphs, international text in JDK 1.2, 88

GUI integration testing, enterprise-critical test strategies, 345

GUI separation from backend, enterprise-critical test strategies, 340, 348

GUIs. See user interfaces

H

handler threads, defined, 206

HandlerThread class, multithreading proxy methods, 208–209

highlighting selections, TextLayout class (international text), 95–97

hit-testing, TextLayout class (international text), 94–95

I

iContract tool, DBC (Design by Contract), 83

IDL (Interface Definition Language)

  CORBA, 399

  distributed Observer chains, 122, 126–127

IEEE floating point, Math package, 458

ignorable characters, text search performance, 411–412

immutability, primitive types, 452, 453

implementation

  DBC (Design by Contract) methods, 84, 85

  distributed Observer chains, 123, 126

  enumerating types with class, 17–18

  Event Notifier, 139–147

  Observer Chaining, 127–129

in/out parameters, creeping featureism (Java paradigm), 10–11

Inflater class, enterprise applets and archive functionality, 422–423

inheritance

  complex applications performance, 380

  DBC (Design by Contract), 73–74, 78, 81–82

inline graphics, international text in JDK 1.2, 90

inlined getters and setters, creeping featureism (Java paradigm), 11

inlining, complex applications performance, 386–389

instances versus references, multithreaded assignment surprises, 219

instantiation, enumerating types with class, 18, 19–20

integer division, truncated, 464

interface methods

  See also user interfaces

  BGSound applet (JavaScript), 34

  no-static, 7

InterfacePanel and PanelModel pairs, Panel States, 289–292
Index

international text in JDK 1.2, 87–101
distributed objects performance, 391–403
distributed Observer chains, 121–130
distributed. See distributed Java
eclipse applets and archive functionality, 421–431
enterprise-critical test strategies, 339–350
event Notifier, 131–153
Event Foundation Classes. See JFC
international text in JDK 1.2, 87–101
techniques, 49–101
enterprise Beans testing, 351–356
enterprise Beans and, 31–47
JavaSoft security package, 295–322
JUnit testing, 357–376
JavaScript and, 33–46
Math package, 455–464
migrating to. See migrating to Java
mobile agent system architecture, 169–178
modeling and patterns, 103–166
modular user interfaces, 255–264
multithreaded assignment surprises, 215–222
multithreaded exception handling, 223–240
multithreading, 205–214
overview. See Java paradigm
Panel States, 281–292
performance, 377–431
primitive types, 435–454
sartorial feature request list, 12–13
security, 293–335
Swing components, 265–280
techniques, 49–101
testing, 337–376
text search performance, 405–420
threads, 203–252
user interfaces, 253–292
Virtual Machine. See JVM
Java Cryptography Architecture. See JCA
Java Cryptography Extensions. See JCE
Java Foundation Classes. See JFC
Java paradigm, 3–14
conceptual confusion, 4–7
creeper, 7–14
overview, 3
sartorial Java feature request list, 12–13

Java

analysis patterns, 155–166
application servers, 323–335
calling from JavaScript, 32–33
calling from JavaScript, 32–33
complex applications performance, 379–389
complex synchronization, 241–252
conceptual models, 105–120
CORBA objects (locating from Java), 191–202
Cryptography Architecture. See JCA
Cryptography Extensions. See JCE
DBC (Design by Contract), 69–86
deployment strategies for client applications, 179–189

JAR (Java Archive) files, enterprise applets and archive functionality, 421–422

J

References, 191
See also CORBA objects (locating from Java)
binding to Names, 196–198
contents of, 195
distributed Observer chains, 123
persistent, 196
resolving from Names, 198
transient, 196
summary, 14
Java Runtime Environment (JRE), stand-alone deployment of Java client applications, 184
JavaBeans, Event Notifier comparison, 153
JavaBeans testing, 351–356
See also testing method signatures, 353 overview, 351–352 preconditions, assertions, and post-conditions, 352–355 self-testing, 355–356 summary, 356
testing strategies, 352
JavaScript, 31–47
BGSound applet, 33–35
BGSound.htm listing, 41–42
BGSound.java listing, 42–45
BrowzIno applet, 38–40
BrowzIno.htm listing, 47
BrowzIno.java listing, 46–47
calling from Java, 36–40
calling Java from, 32–33
JSONObject applet parameters, 32 overview, 31–32 possible uses of, 35–36, 40 security, 35 summary, 41
JavaSoft security package, 295–322
JCA (Java Cryptography Architecture), 295, 296–302, 305
JavaStar issues, enterprise-critical test strategies, 349–350
JCA (Java Cryptography Architecture), JavaSoft security package, 295, 296–302, 305
JComponent, Swing components, 269–270 JDK 1.1 API, text search performance, 409–411
JFC (Java Foundation Classes), 265–267
history of, 265–266
overview, 265, 266–267
JIT (Just-In-Time) VMs, complex applications performance, 382–383
join() and ThreadWrapper class, multi-threaded exception handling, 230
JOVE compiler, complex applications performance, 384–385


K

KaratePuzzle listing, permutations and combinations, 67–68
key objects, JavaSoft security package, 307
KeyFactory class, JavaSoft security package, 314–315
KeyPairGenerator class, JavaSoft security package, 298–300, 301

L

language-based security, application servers, 324
“Law of (software) Acquisition,” DBC (Design by Contract), 76–77
layout metrics, TextLayout class (international text), 98–99
leading, TextLayout class (international text), 98
libraries, compression. See package ZIP
life cycle testing, enterprise-critical test strategies, 341–342
ligatures, international text in JDK 1.2, 88–89
line breaks, TextLayout class (international text), 99–100
linguistic (Abbott) analysis, conceptual models, 108–109
listeners, separating from user interfaces, 273–274
listening for exceptions, multithreaded exception handling, 229–230
LiveConnect (Netscape Navigator), calling Java from JavaScript, 32–33
locked objects, multithreaded assignments and, 215–219
locking mechanisms, encapsulating in separate classes, 221–222
LOOP parameter, jsonObject applet parameters, 32

M

managed objects, Event Notifier, 132
Math package, 455–464
BigDecimal class, 461–463
catastrophic cancellation, 457
Dribble class, 463
IEEE floating point, 458
input error, 459
interval arithmetic, 458–460
overview, 455
polynomial function, 455–458
setScale() method, 462
summary, 464
truncated integer division, 464
MAYSCRIPT attribute, calling JavaScript from Java, 38
Measurement pattern, analysis patterns, 158–162
mediators, Event Notifier, 133, 134
metatypes, primitive types as, 435
method invocation time
CORBA, 399–401
RMI, 394–397
methods
  class/static, 6
  first-class, 8
  no-static interface, 7
  public. See public methods
virtual, 4–5
Microsoft Internet Explorer, calling Java from JavaScript, 33
migrating to Java, 21–47
  audience, 23–24
  background skills, 26–27
  broad-based client/server applications and network management, 29–30
  client/server customer service applications, 30
company context, 25
experience, 24
goals, 25–26
JavaScript and Java, 31–47
migration patterns, 28–30
objectives, 24
overview, 23–30
training, 25–28
training alternatives, 28
training mediums, 27–28
Web-based application development, 29
mobile agent system architecture, 169–178
  advantages of, 170
  agent hosts, 170
  agent life cycles, 171
  agent standards, 176
  agent systems, 170
AgentHost.java listing, 173
Agent.java listing, 177
Aglets Workbench, 176
CORBA and, 170
element, 171–175
GenericAgentHost.java listing, 174–175
object model, 171–172
Object Serialization, 172
OMG (Object Management Group), 176
overview, 169–170
reasons for, 170
RMI (Remote Method Invocation), 172
security, 175–176
summary, 176–178
mobile agents, defined, 169
model quantities as quantities, conceptual models, 117
Model-View-Controller architecture. See MVC architecture
modeling events, Event Notifier, 139–141
modeling and patterns, 103–166
  analysis patterns, 155–166
  conceptual models, 105–120
  distributed Observer chains, 121–130
  Event Notifier, 131–153
Models, Panel States and, 287
ModelView.idl listing, distributed Observer chains, 123
modular user interfaces, 255–264
  application code, 263
  changed methods, 258
  delegation event model, 257, 259–260
  event listeners, 262
  ExitButtonController class, 259
  Java support of MVC, 256–257
  MVC in Java, 260–262
  MVC (Model-View-Controller) architecture, 256–257
  objects of, 262–263
  observer/observable model, 256, 257–258, 262
  overview, 255
  summary, 263–264
  update methods, 258
monitors, defined, 241
months, enumerating types with class, 16–20
multi-valued return parameters, creeping featurism (Java paradigm), 11–12
Multicaster class, multithreaded exception handling, 231
multiple classification, analysis patterns, 160–161
multiplicity, conceptual models, 111, 112
multithreaded assignment surprises, 215–222
See also threads
instances versus references, 219
Maxim 2: never assign to locked ob-
jects, 215–219
Maxim 3: lock with one class and as-
sign with another, 220–221
Maxim 4: encapsulate locking mecha-
nisms in separate classes, 221–222
multithreaded exception handling, 223–240
See also threads
alternate solution, 236–237
BufferedReader object, 234–235
CheckedIOException class, 233, 234, 235, 236
Completion Callbacks, 236–237
exceptionOccurred() method, 235, 239
FileNotFoundException class, 225–226, 227, 235
FileOpener class, 225–226, 233–234, 237
initial code, 225–227
join() and ThreadWrapper class, 230
listening for exceptions, 229–230
Multicaster class, 231
overview, 223–225
PrimaryFrame class, 235, 238
Runnable class, 227–228, 235, 236
SmartThread class, 230–231, 232
start() and Try/Catch technique, 228–229
state management, 224
summary, 237–239
ThreadExceptionCleanup interface, 232–233, 234
ThreadExceptionListener interface, 231, 234
ThreadWrapper class, 228–229, 230
Try/Catch technique, 227–229
Vector class, 231
multithreading, 205–214
See also threads
difficulty of, 212
general design approach, 206–207
HandlerThread method, 208–209
overview, 205
proxy methods, 206–210
returning values to clients, 211
solution domains, 207
summary, 212
Thread Pool, 209
thread separation, 205–206
thread-separated servers, 210–211
ThreadSeparatedCall class, 207–208
ThreadSeparatedFifo class, 208
ThreadSeparator class, 209–210
ThreadSepRetValAdapter class, 211
mutexes, complex synchronization, 241, 243–244, 245
MVC (Model-View-Controller) archi-
tecture, 260–262, 268–269
See also modular user interfaces
Java support of, 256–257
Swing components, 267–269
N
Names
binding IORs to, 196–198
CORBA objects (locating from Java), 195–198
resolving IORs from, 198
Naming Context, CORBA objects
(locating from Java), 193–194, 197–198, 199–200
Naming Service, CORBA objects (lo-
cating from Java), 192–202
native compilation, complex
applications performance, 383–384
NavigationMap class, Panel States,
287–288, 290–291
Netscape Navigator (LiveConnect), call-
ing Java from JavaScript, 32–33
NIH class library, Event Notifier and, 142
no-static interface methods, conceptual
confusion (Java paradigm), 7
non-synchronized updates, complex
synchronization, 241, 242
null comparisons, primitive types,
453
O
Object Management Group (OMG),
mobile agent system architecture, 176
object model, mobile agent system architecture, 171–172
Object Reference, distributed Observer chains, 123
Object Request Broker (ORB), distributed Observer chains, 122
Object Serialization, mobile agent system architecture, 172
object-oriented language inefficiencies, complex applications performance, 380–381
object-oriented optimizations, complex applications performance, 384–389
objects
calling, 206
distributed. See distributed objects performance
instances versus references, 219
primitive types and, 436
primitive types incompatibility, 443–444
task, 206
Observable.java listing, distributed Observer chains, 124
Observation pattern, analysis patterns, 159–161
Observer Chaining, 125–129
See also distributed Observer chains example, 125–129
IDL, 126–127
implementation, 127–129
ObserverChain.idl listing, 126
Order.java listing, 128
overview, 125
Observer design pattern, Event Notifier, 133–134
observer/observable model, modular user interfaces, 256, 257–258, 262
OMG (Object Management Group), mobile agent system architecture, 176
opcodes, primitive types and, 438–439, 450–451
open-close principle, DBC (Design by Contract), 75
optimized searching, text search performance, 414–415
ORB (Object Request Broker), distributed Observer chains, 122
ordering, international text in JDK 1.2, 89–90
Order.java listing, Observer Chaining, 128
P
package ZIP, enterprise applets and archive functionality, 422–425
Paging System subscriber, Event Notifier, 151
Panel States, 281–292
See also user interfaces
design of, 287
interactions, 290–291
InterfacePanel and PanelModel pairs, 289–292
Models, 287
overview, 281–282
Panel State Applet, 288–289
Pluggable View Controllers, 285–287
React-Update pattern, 286–287
State Machines, 284–285
State objects, 284
State Transition Diagrams, 282–285
summary, 291–292
paradigm. See Java paradigm
partitioning, conceptual models, 119–120
pass-by-reference, conceptual confusion (Java paradigm), 5
passing arguments by reference versus by value, primitive types, 442–443
patterns. See modeling and patterns
performance, 377–431
complex applications, 379–389
distributed objects, 391–403
enterprise applets and archive functionality, 421–431
primitive types and verification, 437–439, 450–451
text search, 405–420
permutations and combinations, 51–68
Combinations listing, 65–66
counting combinations, 54–55
counting permutations, 51–54
generate-and-test technique, 60–61
permutations and combinations
(continued)
generating combinations, 59–60
generating permutations, 56–58
KaratePuzzle listing, 67–68
Permutations listing, 62–64
persistent IORs (Interoperable Object References), 196

Phenomenon class, analysis patterns, 159, 160
Phenomenon Types, analysis patterns, 158–159
pluggable look-and-feel, Swing components, 268
Pluggable View Controllers, Panel States, 285–287
Polymorphic Strength Reduction (PSR), complex applications performance, 385–386
Polymorphic Type Elimination (PTE), complex applications performance, 385–386
polymorphism
complex applications performance, 380–381
primitive types and, 445–446
polynomial function, Math package, 455–458
POSIX threads, Java threads and, 243
postconditions
DBC (Design by Contract), 70–71
JavaBeans testing, 352–355
preconditions
DBC (Design by Contract), 70
JavaBeans testing, 352–355
PrimaryFrame class, multithreaded exception handling, 235, 238
primitive types, 435–454
== operator, 440–441
accessing values of, 438
arrays and, 451
containers and, 452–453
deprecation, 449–450
dichotomy of basic semantics, 439–441
dichotomy with objects, 436, 443–444
examples of problems with, 439–441
immutability, 452, 453
JVM (Java Virtual Machine) and, 437
metatypes, 435
multiple different instances, 451
null comparisons, 453
objects incompatibility, 436, 443–444
opcodes, 438–439, 450–451
overview, 435–439
passing arguments by reference versus by value, 442–443
performance and verification, 437–439, 450–451
polymorphism and, 445–446
proposed solution to problem of, 449–454
reasons for, 436–439
reflection and, 446–449
semantic disparities, 439–441
speed of execution, 438–439
stack frames and, 437–438
string literals and, 451–452
summary, 454
tag values, 453
wrapper classes, 441
principals, application server, 324–325
private array, enumerating types with class, 17
private attributes, enumerating types with class, 16
private constructor, enumerating types with class, 16
PRNG (pseudo-random number generator) and SecureRandom, JavaSoft security package, 296–297
producers/consumers, complex synchronization, 246–248
protection proxy design pattern, application servers, 324, 325
proxies, Event Notifier, 144–146
proxy methods, multithreading, 206–210
pseudo-random number generator. See PRNG
PSR (Polymorphic Strength Reduction), complex applications performance, 385–386
PTE (Polymorphic Type Elimination), complex applications performance, 385–386
Index 481

Pthreads, complex synchronization, 241–242
public methods
calling Java from JavaScript, 33
enumerating types with class, 18, 19
Publish-Subscribe. See Event Notifier publishers, Event Notifier, 138, 141, 151
push and pull semantics, Event Notifier, 147
push technology, channel deployment of Java client applications, 185–187
Q
quantities
analysis patterns, 155–156
conceptual models, 117
Quantity pattern, analysis patterns, 157
R
range and error handling testing, enterprise-critical test strategies, 346–347
Range pattern, analysis patterns, 162–166
ranges, assigning automatically (analysis patterns), 162–166
RC# algorithms, JavaSoft security package, 312
React-Update pattern, Panel States, 286–287
read() method, enterprise applets and archive functionality, 427
readZipEntry() method, enterprise applets and archive functionality, 426, 427, 428
Reference Spreadsheet, enterprise-critical test strategies, 341, 342–343, 348
distributed Observer chains, 122–123
passing arguments by (versus by value), 442–443
references versus instances, multithreaded assignment surprises, 219
reflection, primitive types and, 446–449
Remote Method Invocation. See RMI
RemoteEventService class, Event Notifier, 144
resizeBuffer() method, enterprise applets and archive functionality, 427–429
retro interaction, application servers, 332–333
return parameters, multi-valued, 11–12
returning values to clients, multithreading, 211
reusability, DBC (Design by Contract), 76
RMI (Remote Method Invocation)
See also distributed objects
performance
CORBA comparison, 391–394, 401–402
method invocation time, 394–397
mobile agent system architecture, 172
performance of, 394–397
Router class (publishers), Event Notifier, 151
RSA algorithm, JavaSoft security package, 314–317
RSA public key object, JavaSoft security package, 314–315
RuleBasedCollator class, text search performance, 406–408
Runnable class, multithreaded exception handling, 227–228, 235, 236
RWLock class, complex synchronization, 251–252
S
satiric Java feature request list, 12–13
script order, international text in JDK 1.2, 89
scripting, JavaScript, 31–47
SecretKeyFactory class, JavaSoft security package, 308–309
SecureRandom and PRNG (pseudorandom number generator), JavaSoft security package, 296–297
security, 293–335
application servers, 323–335
JavaScript, 35
JavaSoft security package, 295–322
language-based, 324
mobile agent system architecture, 175–176
selection highlighting, TextLayout class (international text), 95–97
self-testing, JavaBeans testing, 355–356
semantic disparities, primitive types, 439–441
semantics, DBC (Design by Contract), 69
Serialization. See Object Serialization
servers
application, 323–335
thread-separated, 210–211
setName() method, enterprise-critical
test strategies, 349
setOffset() method, text search performance, 413–414
setScale() method, Math package, 462
setters, inlined, 11
shaping, international text in JDK 1.2, 88–89
shift distances, text search performance, 415–418
signatures, digital. See digital signatures
signed applets, JavaSoft security package, 298
single classification, analysis patterns, 160, 162
SmartThread class, multithreaded exception handling, 230–231, 232
solution domains, multithreading, 207
specialization/generalization, conceptual models, 118
SRC parameter, JSONObject applet parameters, 32
stack frames, primitive types and, 437–438
stand-alone deployment of Java client
applications, 183–185
See also deployment strategies for
Java client applications
advantages of, 184, 188
disadvantages of, 184–185, 188
JRE (Java Runtime Environment), 184
overview, 183–184
summary, 185
start() and Try/Catch technique, multi-
threaded exception handling, 228–229
State Machines, Panel States, 284–285
state management, multithreaded exception handling, 224
State objects, Panel States, 284
State Transition Diagrams, Panel States, 282–285
static
“dynamic,” 13–14
no-static interface methods, 7
static compilation, complex applications
performance, 383
static testing, JUnit testing, 363
static/class methods, conceptual confusion (Java paradigm), 6
stream ciphers, JavaSoft security package, 312
string literals, primitive types and, 451–452
Stringified Object Reference, distributed
Observer chains, 123
sub-system level testing of backend, en-
terprise-critical test strategies, 342–345
subscriber anchoring, Event Notifier, 147
subscribers, Event Notifier, 151
subscriptions, Event Notifier, 137–138
Swing components, 265–280
See also user interfaces
ComponentUI, 270, 271
example, 271–272, 275–276
FlatUI, 273, 277–278, 279–280
implementing custom look-and-feel,
272–273, 276–277
JComponent, 269–270
JFC (Java Foundation Classes), 265–267
MVC (Model–View–Controller) ar-
tecture, 267–269
overview, 265, 267
pluggable look-and-feel, 268
separating listeners from user interfaces, 273–274
summary, 274
UI factory, 270–271, 272
symmetric algorithms, JavaSoft security
package, 312
synchronization, complex. See complex
synchronization
synchronized updates, complex
synchronization, 241, 242
system testing, enterprise-critical test
strategies, 346
T
tag values, primitive types, 453
task objects, defined, 206
Index 483

leading, 98
line breaks, 99–100
overview, 90–91
selection highlighting, 95–97
tgen program, application servers, 328–330
Thread Pool, multithreading proxy methods, 209
thread separation, multithreading, 205–206
thread-separated servers, 210–211
See also multithreading
ThreadExceptionCleanup interface, multithreaded exception handling, 232–233, 234
ThreadExceptionListener interface, multithreaded exception handling, 231, 234
threads, 203–252
complex synchronization, 241–252
dispatch, 206
handler, 206
multithreaded assignment surprises, 215–222
multithreaded exception handling, 223–240
multithreading, 205–214
POSIX and Java, 243
ThreadSeparatedCall class, multithreading proxy methods, 207–208
ThreadSeparatedFifo class, multithreading proxy methods, 208
ThreadSeparator class, multithreading proxy methods, 209–210
ThreadSepRetValAdapter class, returning values to clients, 211
ThreadWrapper class, multithreaded exception handling, 228–229, 230
toString() method, enterprise-critical test strategies, 349
traceability, DBC (Design by Contract), 76
training, Java migration, 25–28
transiency, Event Notifier, 138
transient IORs (Interoperable Object References), 196
transmitters, channel deployment of Java client applications, 187
text versus concepts, conceptual models, 116–117
TextLayout class (international text), 90–100
bounding box, 99
caret display, 92–94
caret movement, 97–98
described, 87
drawing, 91–92
highlighting selections, 95–97
hit-testing, 94–95
layout metrics, 98–99
techniques, 49–101
DBC (Design by Contract), 69–86
international text in JDK 1.2, 87–101
permutations and combinations, 51–68
testing, 337–376
enterprise-critical test strategies, 339–350
JavaBeans, 351–356
JUnit, 357–376
TestRunner tool, JUnit testing, 373–374
text, international in JDK 1.2, 87–101
text search performance, 405–420
accented characters, 405–406
Boyer-Moore algorithm, 414–418
Boyer-Moore versus Unicode, 416–418
collation elements, 407–409
CollationElementIterator class, 407–408, 413–414
Collator class, 406–407, 409
combining characters, 411–412
components of collation elements, 407, 408–409
getOffset() method, 413–414
ignorable characters, 411–412
Java 2, 413–414, 417, 418
JDK 1.1 API, 409–411
optimized searching, 414–415
overview, 405–406
RuleBasedCollator class, 406–408
setOffset() method, 413–414
shift distances, 415–418
summary, 418–419
Unicode and, 405, 411
Unicode versus Boyer-Moore, 416–418

© Cambridge University Press
www.cambridge.org
truncated integer division, Math package, 464
Try/Catch technique
  complex synchronization, 250
  multithreaded exception handling, 227–229
tuners, channel deployment of Java client applications, 185–187
type-safe enumeration object, enumerating types with class, 20
types, enumerating with class, 15–20

U
UI factory, Swing components, 270–271, 272
UML (Unified Modeling Language), conceptual models, 115
Unicode, text search performance, 405, 411
Unicode versus Boyer-Moore, text search performance, 416–418
update methods
  JavaSoft security package, 311
  modular user interfaces, 258
use case testing, enterprise-critical test strategies, 346
user interfaces, 253–292
  DBC (Design by Contract), 78–79
  modular, 255–264
  Panel States, 281–292
  separating listeners from, 273–274
  Swing components, 265–280

V
  value objects, JUnit testing, 360
  values, passing arguments by (versus by reference), 442–443
  variables, complex synchronization and condition, 246–252
  Vector class, multithreaded exception handling, 231
  verifying signatures, JavaSoft security package, 302
  View Controllers, Pluggable, 285–287
  Virtual Machine. See JVM
  virtual methods, conceptual confusion (Java paradigm), 4–5

W
  wake up order, complex synchronization, 249
  Web bots. See mobile agent system architecture
  Web browsers. See browsers
  Web-based application development, migrating to Java, 29
  wrapper classes, primitive types, 441

Z
  ZIP files. See package ZIP
  ZipEntry class, enterprise applets and archive functionality, 423
  ZipFile class, enterprise applets and archive functionality, 423