

# INDEX

## Symbols

- ! 145
- 141
- 147
- % 143
- %PATH% 32
- & 145, 148
- && 144
- &= 148
- \* 141
- \*= 148
- + 141, 142
- ++ 147
- += 148
- . 147
- / 141
- /= 148
- < 143, 146
- <<= 148
- <= 143
- <applet> tag 627
- <object> tag 627
- = 148
- == != 144
- > 143
- >= 143
- >> 146
- >>= 148
- >>> 146
- >>>= 148
- ^ 145, 148
- ^= 148
- | 145, 148
- |= 148
- || 144
- ~ 148

## A

- abstract 127
  - classes 276
  - methods 276
- abstract class 267, 297
  - expressing in UML 277
  - purpose of 276
  - term defined 267
- abstract class vs. interface 279, 280
- abstract data types 211

- abstract keyword
  - using to declare classes and methods 278
- abstract method 297
- abstract methods
  - implementing in derived classes 278
- abstract thinking 8
- abstraction
  - problem 8
  - the art of programming 210
- abstractions 381
  - assigning properties to 381
  - selecting the right kinds of 741
- access
  - horizontal 283
  - vertical 283
- Access Control Graph (ACG) 687
- access modifier
  - public
    - use of 124
- access modifiers 215
  - behavior of 283
  - default/package 216
  - private 215
  - protected 216
  - public 215
- AccessControlException 631
- ActionEvent 349, 351, 356
- actionPerformed() 612
- ActiveQueue 499
- adapter class
  - to flat-file data file 531
- adapters 374
- address bus 97
- addressing local machine 561
- aggregation 244, 245, 247, 260
  - aggregate constructors 246
  - composite 245, 246, 260
  - composite example code 249
  - definition 245
  - determining type by who controls object lifetime 246
  - effects of JVM garbage collector 246
- example
  - aircraft engine simulation 251
  - aircraft engine simulation class diagram 252
  - simple 245, 246, 247, 260
  - simple example code 248
  - two types of 688
- algorithm
  - running time 101
  - understanding the concept of 92
  - working definition of 99
- algorithms 92, 99
  - good vs. bad 100
- analysis 66, 680
- anonymous class 349, 372
  - avoiding 17
- anonymous listener 374
- Ansel Adams 680
- ANT
  - website address 244
- Ant 244, 245
- applet 123
  - <applet> tag 630
  - applet tag 628
  - basic example 627
  - connecting to server 631
  - defined 626
  - destroy() 629, 630
  - extended example code 636–638
  - HTML page requirement 628
  - inheritance hierarchy 627
  - init() 629, 630
  - jar file loading explained 630
  - life cycle stages 629
  - milestones 629
  - packaging and distribution 628
  - parameters 634, 635
  - running with browser 628
  - security policy and signed applets 634
  - security restrictions 631, 634
  - start() 629, 630
  - stop() 629, 630
- applet methods
  - four primary 628
- applet objects 123
- applets 626, 626–638
  - benefits of using 626
- application
  - compiling and executing 125
  - graceful recovery 64
  - java class definition
  - structure of 124

- layers 558
  - physical deployment 558
  - physical tier distribution 563
  - tier responsibilities 563
  - tiers 558
  - application class definition structure 124
  - application class structure
    - example of 130
  - application distribution 558
    - across multiple computers 561
    - in single jar file 388
  - application object 123
  - application tiers
    - logical 562
    - separation of concerns 563
  - applications
    - building bigger 126
    - multi-tiered 562
  - architecture
    - flexibility 681
    - modularity 681
    - reliability 681
    - stability 681
  - array
    - converting String elements to ints 134
    - creating with literal values 186
    - declaration syntax 181
    - definition of 180
    - difference between primitive type and reference type arrays 187
    - elements 180
    - functionality provided by array types 182
    - homogeneous elements 180
    - main() method String parameter 200
    - multidimensional 194, 198
      - creating with array literals 196
      - declaration syntax 194
    - of primitive types 183
    - primitive type
      - memory arrangement 184
    - processing 134
    - properties of 183
    - ragged 197
    - references
      - calling Object and Class methods on 185
    - single dimensional 183
    - single dimensional in action 187
    - specifying length 181
      - specifying types 181
      - two dimensional
        - example program 198
        - memory representation of 196
        - visualization of 196
      - type inheritance hierarchy 182
  - array processing 64
  - array types 130
  - arrays 180
    - sorting with Arrays class 201
    - using to solve problems 180
  - Arrays class 201
  - arrays of arrays 194
  - Art of Illusion 398
  - ASCII 29
  - Ashmore's hash code algorithm 706
  - assert 127
  - assert mechanism 231
  - association 245, 260
    - definition 245
  - attribute candidates 65
  - attributes 63
  - autoboxing
    - example 501
  - AWT 305, 350
  - AWT and Swing
    - component naming conventions 307
    - historical background 306
    - package names 307
  - AWTEvent 356, 371
- B**
- bad software architecture
    - characteristics of 740
  - base class 266, 297
    - methods
      - overriding 275
      - source code example 269
  - BaseCommand class 754
  - behavior
    - generalized 266
  - Bertrand Meyer 683, 736
  - Bertrand Meyer's Design by Contract (DbC) 723
  - bit 96, 97
  - Bloch's equals() Method Criterion 704
  - Bloch's hash code algorithm 706
  - blocked
    - thread 456
  - blocking
    - accept() method 584
  - BlockingQueue 500
  - Boolean 149
  - boolean 127
  - BorderLayout 321
    - EAST 321
    - NORTH 321
    - SOUTH 321
    - WEST 321
  - bounds
    - of GUI components 308
  - break 127, 159
  - bridge 555
  - Browser JVM 627
  - BufferedOutputStream
    - buffer behavior 517
    - flushing the buffer 517
    - setting buffer size 517
  - Byte 149
  - byte 96, 97
- C**
- cache memory 96
  - Canvas class
    - extending 595
  - Cartesian coordinate 307
  - case 127, 160
  - casting 274
    - advice on use of 275
  - catch 127
  - ChangeEvent 349, 351, 371
  - char 127
  - Character 149
  - character stream file I/O 524
  - CheckboxListCell 399
  - Christopher Alexander 746
  - class 122, 127, 267, 297
    - abstract 276
      - expressing in UML 277
      - purpose of 276
    - abstract class 267
    - final 285
    - four categories of members 214
    - non-static fields 215
    - non-static methods 215
    - static fields 214
    - static methods 215
    - term definition 122, 267
  - class constant
    - definition of 137
  - class declarations
    - viewed as behavior specifications 724
  - class definition

- accessor methods 222
  - adding fields 221
  - adding instance methods 221
  - constructor method 222
  - getter methods 222
  - setter methods 226
  - starting 221
  - class fields
    - accessing from main() method 136
  - class file 122
  - class files 28
  - class invariant 724, 726
    - defined 724
  - class invariants 724
  - class methods 15
  - class responsibility assignment 590
  - class variable
    - definition of 137
  - Class.forName() 581, 750
    - use of 618
  - classes
    - number in an application 244
  - CLASSPATH 27
  - client 554, 557
    - application 554, 557
    - hardware 554, 557
  - client-server applications 581, 582
  - client-server protocol 607
  - client-side RMI runtime 570
  - clone() method
    - example code 709
  - CloneNotSupportedException 708
  - Coad's Inheritance Criteria 684
  - code blocks
    - executing in if statements 157
  - code reuse 680
  - coding convention
    - adopting 20
  - cohesion 14
  - collateral roles
    - modeling 686
  - Collection 487
  - collections
    - algorithms 492
    - ArrayList
      - usage example 484
    - Arrays class 493
    - casting retrieved objects 496
    - Collections class 492
    - core interfaces 486
    - creating new data structures
      - from existing 497
    - enhanced for loop
      - when to use 501
  - framework
    - organization 486
    - purpose of 486
  - general characteristics 482
  - general purpose implementation
    - classes 487
  - generic methods 501
  - generics 499
  - infrastructure 493
  - interfaces 482
  - Java 1.4.2 style 493–499
  - Java 5 core interfaces 499
  - Java 5 style 499–502
  - linked list node elements 489
  - performance characteristics
    - arrays 488
    - hashtable 491
    - linked list 489
    - red-black tree 491
  - red-black tree node elements 492
  - Set class
    - example usage of 494
- command line arguments
  - using in program 133
- Command pattern 746, 753
- CommandFactory class 755
- comments
  - javadoc 19
  - multi-line 18
  - single-line 18
  - three Java types 18
- Comparable interface 711
- Comparator interface
  - implementing 713
- compiler errors
  - fixing 12
- compiling multiple source files 244
- complex application behavior 244
- complexity
  - conceptual 12, 244, 245, 260
  - managing physical 14
  - physical 13, 244, 245, 260
  - relationship between physical and conceptual 14
- Component
  - paint() 384
- component 308
  - absolute positioning 316
  - when to paint() 384
- component bounds 308
- ComponentEvent 359
- components 329–337
  - drawing onto 384
- methods for setting renderers 393
- organizing into containers 316
- composite aggregation
  - defined 245
- composition 680, 688
  - as force multiplier 688
- compositional design 244, 688
- compositionists 680
- computer
  - architecture
    - feature set 95
    - feature set accessibility 95
    - feature set implementation 95
    - three aspects of 95
  - definition of 92
  - memory
    - organization 95
  - processing cycle 98
  - system 92
    - components of 93
    - hard drive 93
    - keyboard 93
    - main logic board 93
    - memory 93, 96
    - monitor 93
    - mouse 93
    - processor 93, 94
    - speakers 93
    - system unit 93
  - vs. computer system 92
- computer network
  - definition 554
  - purpose 554
- computer program
  - modeling real-world problem 210
- computer screen coordinate system 305
- computers 92
- conceptual complexity 12, 244
  - managing 12
  - taming 13
- concurrently
  - performing activities 444
- ConcurrentMap 500
- configuration-management tool 14
- connection
  - incoming client 582
- const 128
- constant 65
  - declaring in main() method 132
  - definition of 132
  - usage in main() method 132
- constructor
  - purpose of 135

constructor call 135  
 constructor method 15  
 consumer thread 468  
 container 308  
 containers  
   top-level 308  
 containing aggregate 247  
 containment  
   by reference 246  
   by value 246  
   polymorphic 688  
 contains 247  
 continue 128  
 control bus 97  
 controller 751  
 coordinates 383  
 coupling 14  
 creativity  
   and problem abstraction 210  
 current position 66  
 custom editor  
   example use of 412  
 custom exceptions  
   creating 438  
 custom painting 385  
 custom renderer 379  
   example use of 412  
   writing 392

**D**

data bus 97  
 data encapsulation 235  
   violating 235  
 data type 65  
   primitive 182  
   reference 182  
 data types  
   array 182  
   primitive  
     boolean 129  
     byte 129  
     char 129  
     double 130  
     float 130  
     int 130  
     long 130  
     short 129  
     working with 130  
   reference 130  
 DataInputStream 610  
 DataInputStream class 581  
 DataInputStream.readUTF() 584  
 DataOutputStream 610, 612  
 DataOutputStream class 581

DbC 723  
 deadlock 474  
 deep copy  
   defined 708  
 default 128  
 Department of Defense 556  
 dependencies  
   managed 681  
 dependency 245  
   definition 245  
   effects of dependency relationships between classes 245  
 dependency inversion principle 740  
 dependency relationship 260  
 dependency vs. association 245  
 derived class 266, 297  
   source code example 270  
 derived requirements 590  
 design 680  
   design by composition 244  
 Design by Contract 723  
 design pragmatists 680  
 development cycle 61  
   applying 61  
   code 61, 782  
   deploying 61  
   factor 61  
   integrate 61  
   plan 61, 782  
   test 61, 782  
   using 61  
 dialog 309  
   modality 309  
 Dialogue with a Skeptic 313  
 difference between abstract class and interface 279  
 direction 65  
 distributed applications 554  
 do 128  
 dominant roles  
   modeling 686  
 Double 149  
 double 128  
 double buffered 385  
 double buffering 385  
 Dr. Barbara Liskov 723  
 Dr. Bertrand Meyer 723  
 DragList 416  
 DressingBoard class 382  
 dynamic class loading 750  
   example code 755  
 Dynamic Factory Pattern  
   advantages of 751

dynamic polymorphic behavior 266  
 DynamicArray  
   case study 482

**E**

Eclipse 30  
 editor  
   writing custom 408  
 effects of change  
   predicting 681  
 Eiffel 723  
 else 128  
 Emeril Lagasse 288  
 empty statement  
   example 138  
 enabling assert in javac 231  
 enabling assertions in Java VM 231  
 enclosing class 366  
 engineering trade-off 680  
 enhanced for loop 499, 501  
 entity diagram 650  
 enum 128  
 enumeration  
   typesafe 254  
 environment variables  
   CLASSPATH 31  
   PATH 31  
   purpose of CLASS & CLASSPATH 32  
   setting 27  
   setting in Windows 31  
 equals() method  
   equivalence relation 703  
   example code 704  
   testing 705  
 Erich Gamma 747  
 error checking 64  
 Error class 429  
 errors  
   compiler 12  
   correcting in source file 29  
 Ethernet 565  
 Event 350  
 event  
   custom code example 290  
   event listener interfaces 349  
   Event listeners 351  
   event listeners  
     registering with components 351  
     registration methods 355  
   event registering methods 352  
   event-handling framework 350  
 EventListeners

- defining 403
  - EventObject 350, 351, 371
  - Events
    - defining 403
  - events
    - choosing the right type 352
    - handling GUI 350
    - low level 349
    - naming conventions 351
    - semantic 349
  - exceptions 428
    - checked 430
    - checked vs. unchecked 430
    - custom
      - creating 438
    - Error class 429
    - Exception class 429
      - extending 438
    - handling 430
    - low vs. high level 438
    - low-level to high-level translation 534
    - multiple try/catch blocks 431
    - ordering
      - importance of 432
      - purpose of 428
    - RuntimeException class 429
    - throwing 436
    - throws clause 436
    - try/catch block 430
    - try/catch/finally block 435
    - try/finally block 435
    - unchecked 430
  - expression 138
  - extends 128
  - extends keyword
    - using to create derived classes 278
  - extension inheritance
    - complications from using 688
    - vs. functional variation 688
- F**
- Façade 746
  - Factory 746
  - factory class 607
    - interfaces involved to employ 686
  - factory class pattern
    - in action 602
  - factory pattern
    - example code 692
  - false 129
  - faux-composite component 393
  - File class 508
    - using 513
  - file I/O 508
    - ASCII 509
    - buffer
      - flushing 511
    - BufferedInputStream 521
    - BufferedOutputStream 516
    - BufferedReader 528
    - BufferedWriter 525
    - buffering
      - purpose of 511
      - write operations 511
    - byte streams 515
    - categorizing I/O classes 510
    - character streams 524
    - choosing the right class 510
    - DataInputStream 522
    - DataOutputStream 517
    - File class
      - using 513
    - file terminal classes 511
    - FileInputStream 521
    - FileOutputStream 515
    - FileReader 527
    - FileWriter 524
    - fixed-length-record
      - locking with lock\_token 534
    - InputStreamReader 527
    - InputStreams 520
    - intermediate classes 511
    - Object
      - calling notifyAll() method 534
      - calling wait() method 534
    - ObjectInputStream 523
    - ObjectOutputStream 518
    - OutputStreamWriter 526
    - PrintStream 520
    - PrintWriter 526
    - Properties class
      - example of use 529
    - RandomAccessFile class
      - use of 531–544
    - Reader classes 527
    - sink
      - term defined 508
    - source
      - term defined 508
    - synchronized keyword usage 534
    - Unicode characters 509
    - user fronting classes 511
    - UTF-8 encoding 509
  - files
    - reading bytes into byte array 521
    - reading contents into character array 527
  - final 128
  - final class 285
    - example of 714
  - final method 285
  - final variable 132
  - finalize method 581
  - finalize()
    - use of 618
  - finalize() method
    - alternative approach 710
    - time-sensitive issues 710
  - finally 128
  - Float 149
  - float 128
  - floor 66
  - flow 10
    - achieving 10
    - concept of 10
    - stages 10
  - FlowLayout 318
    - alignment options 318
  - for 128
  - for each loop 501
  - for-each loop 501
  - frame 309
  - fundamental language features 64
- G**
- garbage collection 136
  - garbage collector 618
  - Garment class 381
  - gate 746
  - gateway 555
  - generalization
    - expressing in UML 268
  - generalized behavior
    - specifying 266
  - generic collection classes 501
  - generic method 501
  - generic methods 501
  - generics 499
  - good design
    - goals of 681
  - good software architecture
    - characteristics of 741
  - goto 128
  - Graphics 383, 421
    - offscreen 384
    - onscreen 384
    - performing drawing operations on 384

graphics  
 property related methods 383

graphics abilities  
 of Java API 382

Graphics class  
 function of 383

graphics drawing operations  
 examples of 383

graphics method  
 three categories of 383

Graphics2D 386, 421

GridBagConstraints 323, 324  
 properties of 324

GridLayout 323  
 GridBagConstraints 323

GridLayout 320

GUI  
 mathematics 307  
 screen coordinate systems 307

GUI events 354–375

handling 350  
 external class with fields 362  
 with anonymous class 372  
 with external class 358, 369  
 with inner class (field level)  
 366  
 with inner class (local-variable  
 level) 368

GUI programming 306–347

## H

hardest thing about learning to program 4

has a 247

hash code  
 algorithm 706

hashCode() method  
 example code 707  
 general contract 706  
 purpose of 705  
 testing 707

HashMap  
 used to store properties 530

homogeneous data types 180

horizontal access 216, 283

host 557, 567

## I

I/O

file 508  
 java.io package overview 508

IDE 27, 30

identifier

class name examples 21  
 constant name examples 21  
 correct formulation of 126  
 method name examples 22  
 naming 19

example of 126

naming rules 126  
 testing for validity 127

unicode 127  
 variable name examples 21  
 well-named 127

if 128

if/else statement 158

images 383

drawing an array of 382

loading 387

loading with Toolkit class exam-  
 ple code 593

immutable properties 381

implements 128

import 128

incoming client connections 582

inheritance 680, 682–686

first purpose of 266

good reasons for using 682

Meyer's Taxonomy 683

object-oriented programming  
 with 266

second purpose of 267

simple example 269

third purpose of 267

three purposes of 266

valid usage checkpoints 684

inheritance form

constant 684

extension 683

facility 684

functional variation 684

implementation 684

machine 684

model 683

reification 684

restriction 683

software 684

structure 684

subtype 683

type variation 684

uneffecting inheritance 684

variation 684

view 684

inheritance hierarchy

assessing with Coad's criteria  
 685

inheritists 680

inner class 349, 366

avoiding 17

inner classes

generated class files 368

input

simple

getting into program 133

InputStream 610

InputStream class 508

InputStreams 508

instance constant

definition of 137

instance methods 15

instance variable

definition of 137

instanceof 128

int 128

Integer 149

integrated development environment  
 (IDE) 27

integrated development environ-  
 ments 30

interface 128, 266, 267, 297

authorized members 279

purpose of 279

reducing dependencies with 686

role of 686

term definition 267

interfaces 680

expressing in UML 280

implementing 266

internationalization

with Readers and Writers 524

Internet Protocol (IP) 565

Internet protocol layers 564

Internet Protocols 556

invoking remote methods 569

IOStreams

use in network programming 554

IP 565

is a relationship

implementing 267

Iterator 487, 499, 501

example use 494

next() 494

purpose of 501

use of 496

## J

J2SDK 31

downloading 31

installing 31

selecting installation directory 31

- jar 27
  - jar files
    - creating from complex package structure 668
    - for applets 630
  - Java
    - API
      - class inheritance hierarchy navigating 112
      - deprecated methods 115
      - documentation 111
      - obtaining information about 111
    - API package diagram 110
    - application
      - two meanings 123
    - applications
      - talking about 123
      - writing 123
    - class construct 214
    - HotSpot Virtual Machine 110
    - JRE 110
    - platform 110
    - SDK 110
    - SDK vs. JRE 110
    - type categories 129
  - java 27, 28
    - using 125
  - Java 5 collections 482
  - Java API
    - support for unanticipated uses 399
  - Java Beans 223
  - Java class structure
    - example 14
  - Java Collections Framework 482–503
    - purpose of 486
  - Java compiler tool
    - using 125
  - Java Database Connectivity 626
  - Java HotSpot Virtual Machine 102
    - architecture 103
    - client 102
    - obtaining 102
    - server 102
  - Java platform
    - continuous evolution effects of 482
  - Java Plug-In 627
  - Java Project 27
    - steps to creating 27
  - java project
    - steps to creating 28
  - Java RMI runtime environment 569
  - Java source file structure 14
  - Java source files
    - general rules for creating 16
  - java source files 28
  - Java virtual machine
    - running programs with 125
  - java.awt.Graphics2D 382
  - java.rmi.Remote interface 599
  - java.util.EventListener 351
  - javac 16, 27, 28
    - assertions
      - enabling 726
    - compiling entire source directory 245
    - compiling multiple source files 244
    - using 125
  - javadoc 27
    - embedding HTML within 19
    - generating HTML API pages 19
  - JBuilder 30
  - JButton 331
  - JCheckBox 331
  - JColorChooser 330
  - JComboBox 330, 369, 408
  - JDBC 626, 639–672
    - 3-tiered application architecture 641
    - driver classes 640
    - packages described 639
    - PreparedStatement
      - recommended usage of 672
    - project description 641–643
    - purpose of 639
    - ResultSetMetaData 655
    - specification 639
    - SQL 640
    - Statement 672
    - steps to employ 641
    - using in project with applets and RMI 641
    - What you need to know to use... 640
  - JDialog 309, 330
  - Jeremy Gibbons 477
  - JFrame 309, 330
  - JLabel 331
  - JList 330, 373, 389, 400
  - JMenu 330
  - JMenuBar 330
  - JMenuItem 331
  - John Vlissides 747
  - JOptionPane 330
  - JPanel 330
  - JPasswordField 331
  - JProgressBar 371
  - JRadioButton 331
  - JRadioButtons
    - five button types 369
  - JRMP
    - version 1.1 vs. 1.2 570
  - JScrollPane 330
  - JSlider 330, 371
  - JSpinner 371
  - JTabbedPane 371
  - JTable 373, 408
  - JTextArea 331
  - JTextField 331
  - JToggleButton 331, 369
  - JToolTip 331
  - JTree 408
  - JViewport 371
  - JVM
    - running multiple on one machine 558
    - starting multiple in Linux 559
    - starting multiple in UNIX 559
    - starting multiple in Windows 558
  - JVMs
    - running multiple 562
  - JWindow 309, 330
- ## K
- keyboard focus 364
  - KeyEvent 349, 351, 364
  - keywords
    - reserved 127
  - killing UNIX processes 560
- ## L
- language features 60, 781
  - layout manager 317
    - BorderLayout 321
    - constraints 321
    - FlowLayout 318
    - GridBagLayout 323
    - GridLayout 320
  - layout managers 305
    - combining with JPanels 327
  - LayoutManager2 321
  - LinkedList 494, 499, 501
  - Liskov Substitution Principle
    - relationship to Meyer Design by Contract Programming 723
    - three rules of 735
  - Liskov Substitution Principle (LSP)

- 723
  - List 487
  - listeners
    - linking to events 354
  - ListeningMainFrame 381
  - listing UNIX processes 560
  - ListIterator 487
  - ListModel 389
    - interface methods 390
    - using 389
  - ListSelectionEvent 349, 351
  - Local Area Network 554
  - localhost 561
  - localhost address 586
  - lock
    - Object 460
  - Long 149
  - long 128
  - low-level events 349
  - LSP 723
  - LSP & DbC
    - common goals 723
    - designing with 724
    - Java support for 723
- M**
- machine code 95
  - Macintosh OS X
    - developer tools 27
  - magic values
    - eliminating the need for 613
  - main application
    - creating separate file 14
  - main application class file
    - creating 17
  - main() method 124
    - body 124
    - different forms of 124
    - parameter 124
  - main() method String array
    - using 133
  - Map 487
  - memory
    - address bus 97
    - alignment 97
    - bit 96, 97
    - byte 96, 97
    - cache 96
    - control bus 97
    - data bus 97
    - hierarchy 96
    - non-volatile 96
    - organization 95
  - RAM 96
  - ROM 96
  - volatile 96
  - word 96, 97
  - menu 65
  - metadata 655
  - method
    - cohesion 217
    - definition structure 217
    - final 285
  - method stubbing 12
  - methods 64, 217
    - abstract 276
    - body 219
    - example definitions 219
    - local variable scoping 233
    - modifiers 218
    - name 219
    - naming 217
    - overloading 220
    - parameter behavior 232
    - parameter list 219
    - passing arguments to 232
    - return types 219
    - signatures 220
    - synchronizing 466
    - using return values as arguments 233
  - methods rule 735
  - modal 309
  - model 63, 751
  - modeling 63
    - collateral roles 687
    - dominant roles 687
    - dynamic roles 687
  - Model-View-Controller 746
  - Model-View-Controller (MVC) 751
  - modulus operator
    - example use 494
  - monitor 460
  - mouse events
    - handling 361
  - MouseEvent 349, 351, 359, 361
  - MouseListener 361, 400
  - MouseMotionListener 361
  - MouseWheelListener 361
  - multiple JVMs 562
  - multi-threaded 588
    - term defined 567
  - multi-threaded server
    - example 610
  - multi-threaded server applications 582
  - multi-threading 444
  - multi-tiered applications 554, 562
  - mutable properties 382
  - MVC 751, 753
    - and ActionListener interface 753
    - Controller
      - using Factory pattern 754
    - simple example of 752
  - MySQL 626, 641, 643–656
    - access control tables 647
    - column permissions 649
    - configuration 643
    - creating databases 647
    - creating DB permissions 648
    - creating tables 650
    - creating tables with SQL script 650
    - db table 648
    - establishing database security 647–650
    - executing SQL script 650, 651
    - helpful commands 646
    - host table 649
    - how to enter SQL commands at monitor prompt 645
    - inserting data into tables 651
    - joining tables 652
    - mysql monitor program 645
    - mysqladmin program 644
    - obtaining and installing 644
    - overview 643
    - populating table data 651
    - security strategy 649
    - select statement 652
    - showing tables and table structure 645
    - steps required to run examples 643
    - table permissions 648
    - updating table data 652
    - user table 647
  - MySQL driver class 655
- N**
- Naming.bind()
    - use of 572, 598
  - native 128
  - NetBeans 30
  - network
    - definition 554
    - homogeneous vs. heterogeneous 555
    - purpose 554
  - network application



- layers 558
- physical deployment 558
- tiers 558
- network applications 554
- network clients
  - running multiple on same machine 561
- network programming
  - client-server scenario 567
  - java support for 567
- networking 554
- networking protocols
  - role of 555
- new 128
- new operator 134
- noun 65
- noun lists
  - suggesting possible application objects 64
- nouns 64, 65
  - mapping to data structures 65
- null 129
- NumberFormat class
  - example usage 191

## O

- Object
  - lock 460
- object 122
  - applet 123
  - application 123
  - creating with new operator 135
  - definition of 135
  - object memory address 135
  - term definition 122
  - their associated type 267
- object attributes 64
- object usage scenario evaluation checklist 700
- ObjectInputStream 582
- object-oriented analysis 680
- object-oriented architecture
  - extending 722
  - preferred characteristics 722
  - reasoning about 722
  - understanding 722
- object-oriented design approach 7
- object-oriented programming 210
- object-oriented programming enablers 680
- ObjectOutputStream 582
- objects
  - natural ordering 711
  - operations upon 267
  - proper behavior of 700
  - seven usage scenarios 700
  - well-behaved 700
- OCP 736
  - defined 736
  - example 736
- offscreen graphics 384, 413
- offscreen image 384
  - creating 379
- Ogden Nash 635
- onscreen graphics 384
- open-closed principle 735
  - achieving 736
- operating system commands
  - UNIX and MS-DOS 56
- operator
  - bitwise
    - OR 148
    - XOR 148
  - bitwise AND 148
  - boolean AND 145
  - boolean OR 145
  - boolean XOR 145
  - combination assignment 148
  - complement 148
  - conditional AND 144
  - conditional OR 144
  - equality 144
  - greater than 143
  - instanceof 147
  - left shift 146
  - less than 143
  - member access 147
  - modulus 143
  - new
    - usage of 134
  - right shift 146
  - String concatenation 142
  - ternary conditional 145
  - unary postfix increment 147
  - unary prefix decrement 147
  - unary prefix increment 147
- operator precedence 139
  - using parentheses to force 141
- operators 139
  - arithmetic 141
- OutputStream 610
- OutputStream class 510
- OutputStreams 508
- overriding
  - clone() method 708
  - equals() method 703
  - finalize() method 710

- hashCode() method 703, 705
- toString() method 703

## P

- package 128
- packages 17
  - compiling code in complex package structures 667
  - complex package structure example 656
  - importing 380
  - naming 17
  - using to organize code 380
- packet 556
- paint() method 385
  - overriding 594
- paintChildren() method 385
- paintComponent() method 385
- painting 382
- part objects 245, 246
- PATH 27
- PATH environment variable
  - fully-qualified path name 33
- pattern
  - singleton 581
- patterns
  - command 746
  - façade 746
  - factory 686, 746
  - factory class 581
  - MVC 746
  - singleton 686, 746
  - typesafe enumeration 254
- pen 65
- Peter Coad 684
- physical complexity 13, 244
- Pi
  - computing with thread 452
- pixel coordinates 307
- polygons 383
- polymorphic behavior 266
  - example of 276
- polymorphic code 485
- polymorphic containment 688
- polymorphic substitution 686
- polymorphism 680
  - applied 687
  - defined 285, 687
  - goal of programming with 687
  - planning for proper use of 687
- port 567
- postcondition 726
  - defined 725

- postconditions 724
  - changing in derived class methods 731
- PowerPC 94
- precondition 726
  - defined 724
- preconditions 724
  - changing preconditions of derived class methods 728
  - weakening 728
- primitive type wrapper classes 149
- primitive types 129
- private 128, 283
- problem abstraction 8, 210
  - and the development cycle 210
  - end result of 211
  - mantra 210
  - performing problem analysis 211
  - process of 210
- problem domain 6, 60, 64, 781
- procedural-based design approach 7
- processing cycle 98
  - decode 98, 99
  - execute 98, 99
  - fetch 98, 99
  - store 98, 99
- processor 94
  - block diagram 94
  - CISC 94
  - machine code 95
  - RISC 94
- producer thread 468
- producer-consumer relationship 468
- production coders vs. design theorists 681
- program
  - computer perspective 98
  - definition of 98
  - human perspective 98
  - java
    - definition of 123
  - simple
    - skills required to create 123
  - two views of 98
- program control flow statements 156
- programming 4
  - basic concepts 122
  - challenges & frustrations 4
  - skills required 4
- programming as art 4
- programming cycle 11
  - code 11
  - integrate 11
  - plan 11

- refactor 11
- repeating 12
- summarized 12
- test 11
- programs 92
  - creating simple 123
  - why they crash 99
- project
  - stages
    - automating 30
    - main application execution 29
    - source code compilation 29
    - source file creation 29
- project approach strategy 6
  - application requirements 6
  - design 7
  - in a nutshell 8
  - language features 7
  - problem domain 6
  - strategy areas 6
- project complexity
  - managing 12
- project objectives 63
- project requirements 6
- project specification 65
- projects
  - creating
    - steps to use the J2SDK 31
    - creating in Windows 2000 31
- properties
  - immutable 381
  - mutable 382
- Properties class
  - use of
    - extended example 613–619
- properties rule 735
- property keys
  - use of 618
- protected 128, 283
- protocols
  - custom client-server 588
  - proprietary application 581
- ps command 560
- public 128, 283
  - access modifier
    - use of 124

## Q

- quality without a name 746
- Queue 500
  - creating with linked list 497
- QueueListenerInterface 499
- QWAN 746

## R

- Rabinowitz and Wagon 477
- race condition 460
- race conditions 459
- ragged arrays 197
- Ralph Johnson 747
- RandomAccessFile 508
- RandomAccessFile class 510
- Reader class 510
- Readers 508
- realization 280
  - expanded form 280
  - expressing in UML 280
  - lollipop diagram 280
  - simple form 280
- record locking 534
- reference data types 130
  - arrays 130
- reference to object combinations 271
- reference types 129
  - working with 134
- reference variable
  - definition of 135
- registry
  - starting externally 573
- relational databases 626
  - creating tables 650
  - foreign key 650
  - primary key 650
  - relating tables 650
- reliable object-oriented software
  - creating 723
- Remote interface
  - extending
    - example code 597
  - use of 570
- Remote Method Invocation 581
- Remote Method Invocation (RMI) 569–576
- renderer
  - custom
    - plugin 379
- repaint() 595
- repaint() method 385
- repainting
  - GUI components 382
- requirements 6, 60, 781
  - gaining insight through pictures 65
- requirements gathering 6
- reserved keywords 127
- resources
  - loading
    - absolute vs. relative URLs 388

- loading via relative URLs 388
  - loading via URL 387
  - return 128
  - Richard Helm 747
  - RMI 590, 596
    - client
      - running 574
    - client application 573
    - client code example 600
    - differences between Java 1.4.2 and Java 1.5 575
    - extended example 596–619
    - extending Remote interface 570
    - extending UnicastRemoteObject 571
    - Java Remote Method Protocol (JRMP) 570
    - Naming.lookup()
      - use of 573
    - overview 569
    - registry
      - starting externally 573
      - starting programmatically 572
    - server
      - running 574
    - server application 572
    - steps to creating application 570
  - RMI registry
    - default port 572
    - starting from program 598
  - RMI runtime 599, 607, 618
  - RMI server-side objects 607
  - rmic tool
    - example use 599, 667
    - use of 571
  - Robert's Rules of Order 555
  - robot rat project specification 62
    - analyzing 63
  - RobotRat project specification 588
  - rule-of-thumb
    - one class-one file 17
- S**
- screen coordinates 308
    - origin 307
    - x axis 308
    - y axis 308
  - selection statements 156
  - self-commenting code
    - writing 19
  - semantic events 349
  - separable model architecture 389
  - Serializable interface 716
  - server 554, 557
    - application 554, 557
    - hardware 554, 557
    - treated as capital equipment 557
  - ServerSocket 567, 582
    - example use of 584
  - ServerSocket class 581
  - ServerSocket.accept() 584
  - Set 487
  - setter methods 226
  - shallow copy
    - defined 708
  - shallow vs. deep copy 708
  - Short 149
  - short 128
  - signature rule 735
  - simple aggregation
    - defined 245
  - simple programs
    - writing 122
  - simple vs. composite aggregation 246
  - SimpleServer application 583
  - simplification
    - of real-world problems 210
  - Singleton 746
  - Singleton pattern 581, 614, 615
  - Socket 567, 582, 610, 612
    - passing to separate thread 567
  - Socket class 581
  - Socket.close() 583
  - Socket.getInputStream() 584
  - Socket.getOutputStream() 584
  - socket-based client
    - extended example 611
  - socket-based client example 585
  - socket-based client-server 582
  - socket-based client-server connection scenario 582
  - socket-based server
    - extended example 607–619
  - software design 212
  - software design patterns 746
    - Abstract Factory 749
    - background 746
    - Command 753
    - definition 746
    - Dynamic Factory 750
    - Factory 749
    - Factory Method 749
    - Singleton 748
    - specification template 747
  - software development roles 5
    - analyst 5
    - architect 5
    - programmer 6
  - SortedMap 487
  - SortedSet 487
  - sorting
    - arrays with Arrays class 201
  - source files
    - compiling 28
    - saving with .java extension 29
  - specialization
    - expressing in UML 268
  - spigot algorithm
    - for generating Pi 477
  - SQL 640
    - executing via JDBC 654
    - insert statement 651
    - joining tables 652
    - nested select statement example 654
    - running script in MySQL 651
    - select statement 652
      - clauses 652
    - update statement 652
  - statement 138
    - empty 138
    - for
      - personality of 165
  - statements
    - break 168
    - chained if/else 159
    - continue 168
    - control flow 156
    - do/while 164
      - personality of 164
    - executing consecutive if 157
    - for 165
      - relationship to while 165
    - if 156
    - if/else 156, 158
    - iteration 162
      - nesting 166
    - labeled break 169
    - labeled continue 170
    - mixing selection and iteration 166
    - selection statements 156
    - switch 156, 159
      - falling through cases 160
      - nested 161
      - using break in 159
    - table of 171
    - unlabeled break 168
    - unlabeled continue 170
    - while 162
      - personality of 163

- static 128
  - static initializer 595
    - example code 594
  - static initializers 234
  - static method
    - main() 137
  - static polymorphism 501
  - strategy
    - project approach 6
  - strengthening preconditions 730
  - strictfp 129
  - String
    - array of 190
  - Strings
    - special treatment given to 190
  - StringTokenizer
    - use of 635
  - stubbing 12
  - super 129
  - super method
    - passing arguments to base class
      - constructor via 273
    - use of 270
  - supertypes & subtypes
    - reasoning about 723
  - Swing 305, 350
    - separable model architecture
      - 379, 389
  - Swing component
    - when to Paint() 384
  - Swing component painting
    - explanation of events 385
  - switch 129
  - switch statement 159
  - synchronized 129
  - synchronized code block 462
  - synchronized methods 466
- T**
- tagging interface
    - concept of 351
  - TCP 565
  - TCP/IP 554, 555, 556, 564–567
    - application layer 564
    - data link layer 565
    - Java support for 565
    - network layer 565
    - physical layer 565
    - transport layer 565
  - Ternary 145
  - test driver program 223
  - testing
    - user-defined type 223
  - TextPad™ 30
  - The 639
  - the art of programming 4, 8
    - inspiration 9
    - money but no time 9
    - mood setting 10
    - time but no money 9
    - where not to start 9
    - your computer 9
  - thinking outside the box 210
  - this 129
  - thread 443
    - analogy 444
    - blocked 456
    - Clock1 class 446
    - computing Pi 452
    - consumer 468
    - deadlock 474
    - definition of 444
    - interrupting 448
    - lock 460
    - main 444
    - priority 455
    - producer 468
    - race condition 460
    - race conditions 459
    - scheduling 455
    - sleeping 448
    - synchronization 460
    - synchronization rules 462–466
    - synchronized keyword 462
    - synchronizing methods 466
  - Thread Class 581
  - Thread class
    - constructor methods 449
    - methods of 447
    - notify() methods 470
    - run() method 449
    - start() method 449
    - wait() methods 470
    - yield() method 454
  - ThreadGroups 444
  - threads 444–477
    - creating your own 449
    - of a simple Java program 445
  - throw 129
  - throw keyword 436, 437
  - Throwable
    - manipulating object 433
    - methods of 433
  - Throwable class
    - catching all errors with 432
  - Throwable class hierarchy 428
  - throwing exceptions 436
  - throws 129
  - throws clause 436
  - tight spiral development 61
  - timeless way 746
  - Toolkit class 595
  - top-level container
    - API 312
    - constructors 312
    - methods 314
  - top-level containers 308
  - toString() method
    - testing 705
  - transient 129
  - transitivity
    - exhibited by inheritance hierar-  
chies 267
  - Transmission Control Protocol  
(TCP) 565
  - transparency
    - using to render GUI components  
413
  - TreePrinterUtils class
    - use of 317
  - true 129
  - try 129
  - try/catch block 431
  - try/finally block
    - good use of 436
  - type 267, 297
  - type coercion 274
  - types
    - array 182
  - typesafe enumeration
    - example code 254
    - when to use 254
  - typesafe enumeration pattern 254
- U**
- UDP 554, 565
  - UML 13, 210, 244, 260
    - class diagram 213
    - composite aggregation 247, 249
    - expressing abstract class 277
    - expressing inheritance 268
    - expressing interfaces 280
    - expressing realization 280
    - expression aggregation 246
    - realization
      - diagram
        - expanded form 281
        - simple form 281
    - sequence diagram
      - aircraft engine object creation

- 254
  - sequence diagrams 250, 251
  - simple aggregation 246, 247
  - using to tame conceptual complexity 244
  - UML class diagram
    - purpose of 213
  - UnicastRemoteObject 599
    - example use of 597
    - usage of 571
  - UNICODE 29
  - Unified Modeling Language (UML) 13
  - UNIX
    - killing processes 560
    - listing processes 560
  - URI 554
  - URL 554
  - URL class
    - use of 568
  - User Datagram Protocol (UDP) 565
  - user-defined types 211
  - UTF Strings 584
    - reading 584
    - writing 584
  - UTFDataFormatException 584
  - utility methods
    - definition of 216
- V**
- variable 65
    - declaring in main() method 131
    - definition of 131
    - final 132
    - usage in main() method example 131
  - Vector
    - serializing to disk 519
  - verb phrases 64
  - verbs 64
  - vertical access 283
  - view 751
  - virtual machine 102
    - Java classic vs. HotSpot 103
    - Java HotSpot 102
  - void 129
  - volatile 129
- W**
- while 129
  - whole object 245
  - whole objects 246
  - whole/part class relationship 245
  - window 309
  - window close options 311
  - window decorations 310
  - word 96, 97
  - word processor
    - using to create source files 29
  - world
    - imperfect understanding of 680
  - wrapper classes 149
    - usage of 134
  - Writer class 510
  - Writers 508
- X**
- x
    - screen axis 308
  - XML
    - property files 529
- Y**
- y
    - screen axis 308

