
Index

It's a d-mn poor mind that can only think of one way to spell a word!
—Andrew Jackson

Numbers

- 0 (zero)**
 - floating-point conversion use (table); 628
 - integer conversion use (table); 627
- 0X/0x (zero x) prefix**
 - hexadecimal literal indicator; 167
 - hexadecimal floating-point literal indicator; 168
- 2D geometry**
 - `java.awt.geom` package; 720

Symbols

- (space)**
 - floating-point conversion use (table); 628
 - integer conversion use (table); 627
- ! (bang)**
 - logical negation operator; 207
- != (bang equals)**
 - inequality operator; 99, 206
 - term definition; 5
- " (double quote)**
 - string literals use of; 3, 168
- # (pound sign)**
 - floating-point conversion use (table); 628
 - integer conversion use (table); 627
 - specifying member names with, doc comments; 483
 - string conversion use; 628
- \$ (dollar sign)**
 - format specifier use; 626
 - multiline mode, dot-all mode pattern matching; 324
 - nested type internal name use; 149

- % (percent)**
 - format specifier prefix, examples of; 23
 - format specifier use; 624
 - remainder operator; 201
 - preserving symmetry around zero; 658
 - term definition; 11
- %% (percent percent)**
 - outputting percent sign; 625
- & (ampersand)**
 - AND operator; 20
 - bitwise AND operator; 208
 - doc comment use; 482
 - generic type declaration use; 253
 - logical AND operator; 207
 - Unicode sequence for; 482
- @ HTML tag,**
 - doc comment use; 482
- && (ampersand ampersand)**
 - conditional AND operator; 20, 207
 - evaluation order differences; 215
- & HTML tag,**
 - doc comment use; 482
- > HTML tag,**
 - doc comment use; 482
- &l t; HTML tag,**
 - doc comment use; 482
- ' (single quote)**
 - character literals use of; 167
- ((left parenthesis)**
 - floating-point conversion use (table); 628
 - integer conversion use (table); 627
- () (parentheses)**
 - method invocation operator; 224
 - parameter list use; 2
 - precedence control in expressions; 222
 - type casting operator; 27, 91, 219

- * **(asterisk)**
 - binary multiplication operator; 201
 - term definition; 6
 - import on demand use; 469
 - multiplication operator, term definition; 6
 - static import on demand use; 72
 - wildcard, match multiple characters; 322
- */ **(asterisk slash)**
 - comment delimiter; 163
 - term definition; 6
 - doc comment delimiter; 482
- *= **(asterisk equals)**
 - assignment use; 11
- + **(plus)**
 - addition operator, term definition; 6
 - binary addition operator; 201
 - concatenation operator
 - String object concatenation with; 59
 - String use; 21
 - term definition; 12
 - floating-point conversion use (table); 628
 - implicit string conversion with; 220
 - integer conversion use (table); 627
 - string concatenation and representation use; 100
 - string concatenation operator; 214
 - unary positive operator; 202
- ++ **(plus plus)**
 - increment operator; 11, 205
 - prefix and postfix expressions, statements; 230
- += **(plus equals)**
 - assignment use; 11
 - concatenation assignment operator, String use; 21
 - concatenation operator, String object concatenation with; 59
 - implicit string conversion with; 220
- , **(comma)**
 - array initializer separator; 175
 - expression separator, in for loops; 237
 - floating-point conversion use (table); 628
 - generic type parameters separated by; 250
 - integer conversion use (table); 627
 - parameter list separator; 2
 - variable list separator; 5
- **(hyphen) format specifier**
 - floating-point conversion use (table); 628
 - integer conversion use (table); 627
 - string conversion use (table); 628
- **(minus)**
 - binary subtraction operator; 201
 - subtraction operator, term definition; 6
 - unary negation operator; 201
- **(minus minus)**
 - decrement operator; 205
 - term definition; 11
 - prefix and postfix expressions, statements; 230
- . **(dot)**
 - dot-all mode pattern matching; 324
 - member access use; 16, 223
 - nested type reference use; 133
 - method invocation use; 16, 58
 - operator, member access use; 3, 411
 - package component separation; 36
 - qualified names; 123
 - precision format specification use; 24
 - static member reference use; 8
 - this reference use; 17
 - wildcard, match single characters; 322
- .* **(dot asterisk)**
 - wildcard, match zero or more characters; 322
- ... **(dot dot dot)**
 - variable number of arguments indicated by; 58, 60
- .class **class literals**; 169
 - wrapper class TYPE constant relationship to; 186
- .**properties files**
 - PropertyResourceBundle class use; 693
- / **(slash)**
 - binary division operator; 201
 - division operator, term definition; 6
 - time zone string identifier use; 700
- /* **(slash asterisk)**
 - comment delimiter; 163
 - term definition; 6
- /** **(slash asterisk asterisk)**
 - comment delimiter; 163
 - term definition; 6
 - doc comment delimiter; 482
- // **(slash slash)**
 - comment delimiter; 163
 - term definition; 6
 - single-line comments; 163
- /= **(slash equals)**
 - assignment use; 11
- ;**(semicolon),**
 - empty statement indication; 34
- <> **(angle brackets)**
 - generic type definition use; 29
- < **(left angle bracket)**
 - doc comment use; 482
 - format specifier use; 626
 - less than, Unicode sequence for; 482
 - less than comparison operator; 5, 206
 - type parameter documentation use; 485
- << **(left angle left angle)**
 - shift bits left; 209

- <= (left angle bracket equals)**
 - less than or equal comparison operator; 5, 206
- <p> HTML tag**
 - doc comment use; 483
- <pre> HTML tag**
 - doc comment use; 483
- = (equals)**
 - assignment operator; 212
 - binary operator combination with; 11
 - term definition; 4
 - equality operator, term definition; 5
- == (equals equals)**
 - equality operator; 206
 - term definition; 5
 - object equality testing; 99
 - IdentityHashMap notion of; 592
 - reference equality testing; 101
 - string comparison issues; 311
 - string object comparison results; 22
- > (right angle bracket)**
 - doc comment use; 482
 - greater than comparison operator; 206
 - term definition; 5
 - type parameter documentation use; 485
 - Unicode sequence for; 482
- >= (right angle bracket equals)**
 - greater than or equal comparison operator; 206
 - term definition; 5
- >> (right angle right angle)**
 - shift bits right, sign fill; 209
- >>> (right angle right angle right angle)**
 - shift bits right, zero fill; 209
- ? (question mark)**
 - abstract type parameter wildcard; 31
 - generic type wildcard; 257
- ?: (question colon)**
 - condition operator; 210
 - evaluation order differences; 215
- @ (at-sign)**
 - annotation use; 35
 - doc comment tag use; 483
 - string representation use; 100
 - Unicode sequence for; 482
- @author tag; 486**
- @code inline tag; 487**
- @Deprecated annotation**
 - @deprecated tag use with; 486
- @deprecated tag; 486**
- @docRoot inline tag; 488**
- @exception tag; 485**
- @inheritDoc inline tag; 488**
 - copying doc comments with; 490
- @link inline tag; 484**
- @linkPlain inline tag; 484**
- @literal inline tag; 487**
- @param doc comment tag; 485**
 - separate inheritance of; 489
- @return tag; 485**
- @see tag; 483**
- @serial tag; 562**
- @serialData tag; 563**
- @serialField tag; 563**
- @since tag; 487**
- @throws tag; 485**
 - inheriting; 490–491
- @value inline tag; 487**
- @version tag; 487**
- [] (square brackets)**
 - array declaration use; 173
 - array syntax use; 2
 - array use; 19
 - wildcard, sets; 322
- [X (open brace X)**
 - notation for primitive array type name; 412
- [Ljava;**
 - notation for object array type name; 412
- ^ (caret)**
 - bitwise exclusive or (**XOR**) operator; 208
 - exclusive **OR** operator; 20
 - logical exclusive **OR** operator; 207
 - multiline mode, dot-all mode pattern matching; 324
 - start-of-line boundary marker, multi-line input use; 646
 - wildcard
 - boundary match beginning of a line; 322
 - negations; 322
- { } (curly braces)**
 - array initializers; 175
 - block operators; 230
 - class member use; 2
 - interface declaration use; 120
 - @link inline tag use; 484
 - @linkPlain inline tag use; 484
- | (vertical bar)**
 - bitwise inclusive **OR** operator; 208
 - inclusive **OR** operator; 20
 - logical inclusive **OR** operator; 207
- || (vertical bar vertical bar)**
 - conditional **OR** operator; 20, 207
 - evaluation order differences; 215
- ~ (tilde)**
 - unary bitwise complement operator; 209
- \ (backslash)**
 - including as a literal in a pattern; 322
 - removing special character meanings with; 322
 - special characters using (table); 750

- \ddd (backslash ddd)**
 - char octal value; 167
 - char octal value character literal; 167
 - \" (backslash double quote)**
 - character literal; 167
 - double quote character literal; 167
 - \' (backslash quote)**
 - single quote character literal; 167
 - \\ (backslash backslash)**
 - backslash character; 322
 - character literal; 167
 - \a (backslash a)**
 - alert (bell) character; 322
 - \b (backslash b)**
 - backspace character literal; 167
 - wildcard, boundary match beginning of a word; 322
 - word boundary pattern matching; 328
 - \d (backslash d)**
 - wildcard, match any digit; 322
 - \e (backslash e)**
 - escape character; 322
 - \f (backslash f)**
 - form feed character literal; 167
 - \n (backslash n)**
 - embedding in strings; 168
 - newline character; 322
 - newline character literal; 167
 - newline embedding in strings; 167
 - newline line separator; 519
 - LineNumberReader class; 527
 - Unix line terminator pattern matching; 324
 - \r (backslash r)**
 - carriage return character literal; 167
 - carriage return line separator, LineNumberReader class; 527
 - \r\n (backslash r backslash n)**
 - carriage return newline line separator; 519
 - LineNumberReader class; 527
 - \s (backslash s)**
 - wildcard, match any whitespace character; 322
 - \t (backslash t)**
 - tab character; 322
 - tab character literal; 167
 - \u0008 (backslash u 0008)**
 - backspace Unicode encoding; 167
 - \u (backslash u 0009)**
 - tab Unicode encoding; 167
 - \u000A (backslash u 000A)**
 - newline Unicode encoding; 167
 - \u000C (backslash u 000C)**
 - form feed Unicode encoding; 167
 - \u000D (backslash u 000D)**
 - carriage return Unicode encoding; 167
 - \u0022 (backslash u 0022)**
 - double quote Unicode encoding; 167
 - \u0027 (backslash u 0027)**
 - single quote Unicode encoding; 167
 - \u005C (backslash u 0005C)**
 - backslash Unicode encoding; 167
 - \u (backslash u)**
 - Unicode encoding
 - inside character literals; 167
 - multiple u's in; 162
 - \W (backslash W)**
 - matching characters that are not part of a word; 328
- A**
- a, A format specifier**
 - floating-point conversion use; 627
 - ABLIterator example class; 614**
 - abort virtual machine**
 - conditions that can cause; 674
 - abrupt completion**
 - term definition; 283
 - abs method**
 - Math class and StrictMath class; 658
 - absolute**
 - path, manipulating; 545
 - placement
 - graphical user interfaces, design problems; 718
 - term definition; 718
 - value; 658
 - abstract**
 - See also: interfaces
 - classes
 - benchmark harness design use; 109
 - collections, use in writing collection implementations; 611
 - interfaces vs.; 131
 - methods and; 97–99
 - enums not permitted to use; 154
 - final keyword vs.; 43
 - interface declaration; 122
 - method modifier; 57
 - restrictions on use of other method modifiers; 58
 - methods
 - enum restrictions; 158
 - overriding concrete methods with; 99
 - overriding considerations; 85
 - reserved keyword (table); 165
 - term definition; 43
 - Abstract Window Toolkit (AWT)**
 - package details; 717–720
 - thread use; 370
 - AbstractBase example class; 472**

- AbstractCollection class**; 612
 - implementation starting point; 611
- abstractions**
 - collection representation of; 567
 - exception chaining use to raise the level of; 292
 - serialized forms as; 559
- AbstractList class**
 - clear method; 615
 - get method; 615
 - implementation
 - guidelines; 612
 - starting point; 611
 - removeRange method; 615
- AbstractMap class**
 - entries method; 615
 - implementation starting point; 611
 - put method; 615
 - remove method; 615
- AbstractQueue class**; 611
 - offer method; 612
 - peek method; 612
 - poll method; 612
- AbstractSequentialList class**
 - add method; 615
 - implementation starting point; 611
 - remove method; 615
 - set method; 615
 - size method; 615
- AbstractSet class**
 - implementation guidelines; 612
 - implementation starting point; 611
- accept method**
 - FileFilter interface; 549
 - FilenameFilter interface; 548
- access/accessibility**
 - See also:* extracting; retrieving
 - accessor methods, term definition; 66
 - annotation, applicability vs.; 394
 - arrays
 - elements; 173
 - linked lists vs.; 615
 - atomic
 - strategies for; 370
 - volatile variable characteristics; 372
 - characters in a String object, with charAt; 22
 - checking, AccessibleObject class; 417
 - class variables; 45
 - control; 47–48
 - extensible framework design; 76, 113
 - java.security.acl package use; 732
 - methods use for; 65–68
 - per-class, not per-object; 48, 68
 - security manager actions; 678
 - term definition; 38
 - thread run methods, anonymous inner class protection; 345
 - current thread, Thread.currentThread; 341
 - decisions, an extensible framework design; 113
 - enclosing objects, inner classes; 138
 - fields, object reference type as determinator of; 86
 - files
 - random access; 541–543
 - security checks, File streams; 541
 - security checks, RandomAccessFile class; 542
 - hidden interface constants; 123
 - immutability vs.; 67
 - inherited members; 86
 - javax.accessibility package; 737
 - members; 223
 - method, overriding algorithms; 472–475
 - methods, object reference class as determinator of; 86
 - modifiers; 48
 - enum; 154
 - method declaration use of; 57
 - not permitted on interface methods; 122
 - private; 48
 - protected; 48
 - public; 48
 - term definition; 19
 - overriding and; 88
 - package; 48, 471–475
 - protected members; 93–95
 - type conflict issues; 94
 - random, array advantages; 615
 - read-only, enforcing; 65
 - reflection, Class class inspection; 402–408
 - requirements for overriding methods; 85
 - sequential, AbstractSequentialList use; 615
 - static nested types; 134
 - static protected members, accessing; 95
 - term definition; 13
- AccessControlContext class**; 683
 - security manager use; 679
- AccessControlException class**
 - privileged code exempted from throwing; 682
 - security manager use; 679
- AccessController class**; 681–683
 - checkPermission method; 681
 - doPrivileged method; 679, 682
 - getContext method; 681

- AccessibleObject class**; 417–418
 - isAccessible method; 418
 - overview and place in introspection hierarchy; 399
 - setAccessible method; 417
- accessor(s)**
 - advantages of; 347
 - methods, term definition; 66
 - synchronized; 347
- accessOrder method**
 - LinkedHashMap class; 592
- acos method**
 - Math class and StrictMath class; 657
- Action example class**; 136
- action(s)**
 - events, term definition; 719
 - list, term definition; 680
 - synchronization, term definition; 372
- ActionEvent class**; 719
 - enableEvents method; 719
- ActionListener interface**; 719
- activatable**
 - servers, java.rmi.activation package; 731
- Activatable class**; 730
- active thread**
 - term definition; 378
- activeCount method**
 - ThreadGroup class; 378, 379
- activeGroupCount method**
 - ThreadGroup class; 379
- add method**
 - AbstractSequentialList class; 615
 - Calendar class; 698
 - Collection interface; 576
 - BlockingQueue interface use; 605
 - List interface; 581
 - ListIterator interface; 473
- addAll method**
 - Collection interface; 576
 - BlockingQueue interface use; 605
 - Collections class; 598
- addElement method**
 - Vector class; 617
- addFirst method**
 - LinkedList class; 583
- addition (+) operator**
 - See also*: arithmetic; numbers/numeric; operator(s)
 - binary; 201
 - term definition; 6
- additive operators**
 - precedence; 221
- addLast method**
 - LinkedList class; 583
- addObserver method**
 - Observable class; 636
- addShutdownHook method**
 - Runtime class; 673
- Adler-32 algorithm**
 - Adler32 class; 737
- after method**
 - Calendar class; 699
 - Date class; 695
- aging**
 - priority, term definition; 358
- alert character**
 - special symbol for; 322
- algorithms**
 - See also*: design issues and strategies; performance
 - binary search; 310
 - execution time, big *O* notation use; 579
 - import, compiler actions; 469–470
 - “most specific”
 - method resolution; 224–228
 - method resolution, generic type modifications; 272–276
 - multiprocessor issues; 359
 - random access list manipulation; 585
 - StrictMath class, in **FDLIBM** “C” library; 659
 - thread priority issues; 359
- alive**
 - term definition; 365
- allocation of memory**
 - during object creation; 49
 - garbage collection handling of problems with; 448
- allOf method**
 - EnumSet class; 595
- ambiguities**
 - See also*: design issues and strategies
 - generic type overloading, handling; 275
 - inheritance problem, multiple interface constant issue; 124
 - method resolution; 227
 - variable argument handling; 61
- ampersand (&)**
 - AND** operator; 20
 - bitwise **AND** operator; 208
 - doc comment use; 482
 - generic type declaration use; 253
 - logical **AND** operator; 207
 - Unicode sequence for; 482
- ampersand ampersand (&&)**
 - conditional **AND** operator; 20, 207
 - evaluation order differences; 215
- and method**
 - BitSet class; 634

- AND (&) operator;** 20
 - bitwise operations; 208
 - logical operations; 207
 - precedence; 222
 - Unicode sequence for; 482
- andNot method**
 - BitSet class; 634
- angle brackets (<>)**
 - generic type definition use; 29
- AnnotatedElement interface;** 414–416
 - Constructor class implementation of; 424
 - Field class implementation of; 419
 - getAnnotation; 414
 - getAnnotations; 414
 - getDeclaredAnnotations; 414
 - isAnnotationPresent; 415
 - overview and place in introspection
 - hierarchy; 399
 - Package class implementation; 477
- Annotation class**
 - See also:* java.text package; text
 - Method class use; 421
- Annotation interface;** 391
 - See also:* java.util.annotation package
- annotation(s);** 387–396
 - See also:* coding style; design issues and
 - strategies; documentation
 - as class modifiers; 43
 - constructor modification; 51
 - @Deprecated, @deprecated tag use with; 486
 - elements
 - declaration rules; 389
 - term definition; 35, 388
 - enum constant declaration use; 154
 - example; 388
 - initialization; 390
 - interface modifier; 122
 - interface named constants; 121
 - Java-defined; 396
 - marker; 391
 - meta, term definition; 394
 - method modification; 57
 - overview; 35–36
 - package; 476
 - program elements; 392–393
 - reflection queries about; 414–416
 - as reflection type; 397
 - restricting the application of; 393–395
 - retention policies; 395, 414
 - SuppressWarnings class; 745
 - types; 389–391
 - term definition; 387
 - term definitions; 35
 - as variable attribute; 170
- ANNOTATION_TYPE constant**
 - ElementType class; 394
- AnnotationTypeMismatchException class**
 - annotation reflection use; 415
- anonymous**
 - array, term definition; 176
 - inner classes; 144–146
 - enum constants as; 158
 - initialization block use; 55
 - thread run method protection with; 345
- API (Application Programming Interface)**
 - compatibility issues; 747
- append method**
 - Appendable interface; 332
 - StringBuilder class; 332
 - Writer class; 511
- Appendable interface**
 - append methods; 332
 - Formatter class use; 332, 631
 - Writer class implementation of; 511
- appendCodePoint method**
 - StringBuffer class; 336
 - StringBuilder class; 336
- appending**
 - characters, to matched characters; 327
 - to string buffers; 332
- appendReplacement method**
 - Matcher class; 327
- appendTail method**
 - Matcher class; 327
- Applet class;** 720–721
 - destroy method; 720
 - init method; 720
 - start method; 720
 - stop method; 720
- applet(s)**
 - See also:* beans; security
 - access control issues, interaction with; 682
 - java.applet package
 - details; 720–721
 - overview; 716
 - lifecycle, methods that constitute; 720
 - native method prohibited in; 74
 - term definition; 720
 - thread security issues; 376
- APPLET tag;** 720
- Application Programming Interface (API)**
 - compatibility issues; 747
- application(s)**
 - access control issues, interaction with; 682
 - evolution of, technical issues; 741–748
 - execution, termination of; 369
 - invocation of, main method use; 73
 - state, printing, as debugging aid; 385
- APT (annotation processing tools)**
 - annotation analysis; 396

Arabic

- encoded bytes, converting to Unicode characters; 512

- arccosine**; 657

archive

- files, **JAR**, `java.util.jar` package; 735–736

- arcsine**; 657

- arctangent**; 657

arguments

- See also*: parameter(s)
- method invocation; 16
- program invocation; 73
- type, term definition; 250
- variable number of, syntax for; 58, 60

arithmetic

- See also*: decimal; floating point; integers; Math class; mathematics; **StrictMath** class
- character; 202
- finite; 202
- floating-point; 202
 - strict vs. non-strict; 203
 - `strictfp` class modifier use; 43, 57
- with infinities; 202
- integer; 202
 - arbitrary precision, **BigInteger** class; 722
- operations; 201
- operators, binary; 201
- operators and value assignment; 5
- two's-complement, integer arithmetic; 201
- ArithmeticException** class; 202, 215
 - BigDecimal** class use; 724
 - RuntimeException** extension; 284

arity

- term definition, (footnote); 60

Array class; 429–432

- `get` method; 430
- `getLength` method; 430
- `newInstance` method; 429
 - compatibility issues; 747
- `set` method; 430

array(s); 173–178

- See also*: collections; data structures
- access, linked list access vs.; 615
- anonymous, term definition; 176
- ArrayList** class; 570, 582
- bounds of, term definition; 19
- bytes
 - converting to/from strings; 319
 - source or destinations of **ByteArray** streams; 521
- casting of; 177

char

- adding to a string buffer; 332
- creating from strings; 319
- extracting from a **StringBuilder** object; 333
- mapping strings to/from; 317
- characters
 - constructing with `toCharArray`; 22
 - source or destinations of **CharArray** streams; 522
- cloning; 106
- collection elements, retrieving, `Collection.toArray`; 575
- comparing; 177
- component type
 - name notation; 412
 - retrieving, `Class.getComponentType` method; 407
- converting sequences to, design concerns; 85
- copying; 176
 - `System.arraycopy`; 665
- CopyOnWriteArrayList** class; 607
- CopyOnWriteArraySet** class; 607
- creating
 - evaluation order; 215
 - explicit; 173
 - generic type implications; 251
 - implicit; 175
 - wildcards vs. parameterized types use; 268–270
- declaration; 173
- dynamic, creating; 430–432
- elements, accessing; 173
- empty; 19
 - term definition; 174
- EnumMap** use; 596
- for-each loop use with; 240
- generic; 428
 - introspection hierarchy figure representation of; 399
- hash code calculation on contents of; 177
- indices; 173
 - numbering conventions; 176
- initialization; 19, 175–176
 - enums compared with; 153
- length; 19, 174
- length field, `String.length` method vs.; 307
- list element indexing compared with; 473
- name notation; 412
- nested; 174
- null array reference vs. empty array; 174
- overview; 18–20
- polymorphism of; 177
- printing, **StringWriter** class use; 524
- range check, optimization of, (footnote); 173

- reference types; 166
- replacing elements in, synchronized use for
 - access control; 349
- representation by `Class` objects; 399
- sequence parameters as, overriding impact on; 85
- shared, client-side synchronization use; 349
- syntax; 2
- term definition; 18, 173, 429
- testing if `Class` object represents,
 - `Class.isArray` method; 405
- types and; 177
- variable argument sequences as; 61
- variables; 173
 - modifiers; 174
- ArrayBlockingQueue class**; 605
- ArrayBunchList example class**; 612
- arraycopy method**
 - `System` class; 176, 665
- ArrayIndexOutOfBoundsException class**; 173
 - array checking us; 19
 - in type hierarchy, (figure); 280
 - unchecked exception; 34, 280
- ArrayList class**; 582
 - `CopyOnWriteArrayList` class; 607
 - `ensureCapacity` method; 582
 - extension of `AbstractList` class; 615
 - `List` interface implementation; 581
 - overview; 570
 - performance, `LinkedList` compared with; 583
 - `RandomAccess` interface implemented by; 584
 - snapshot creation use of; 573
 - `trimToSize` method; 582
 - `Vector` class analogous to; 617
- Arrays class**; 177, 607–609
 - `asList` method; 608
 - `binarySearch` method; 608
 - `deepEquals` method; 608
 - `deepHashCode` method; 608
 - `deepToString` method; 608
 - `equals` method; 608
 - `fill` method; 608
 - `hashCode` method; 608
 - `sort` method; 608
 - `toString` method; 608
- ArrayStoreException class**; 177
 - `System.arraycopy` use; 665
- ASCII**
 - See also*: character(s)
 - character representation; 161
 - data representation; 9
 - standard character encoding; 320
- asin method**
 - `Math` class and `StrictMath` class; 657
- asList method**
 - `Arrays` class; 608
- assert**
 - reserved keyword (table); 165
 - statement; 297
- assertion(s)**; 296–303
 - control flow; 299–300
 - controlling at runtime; 444–445
 - evaluation, turning on and off; 300–303
 - impact on existing code; 743
 - locks; 348
 - removing; 302
 - requiring; 302
 - state; 297–299
 - term definition; 296
- AssertionError class**; 296, 297
- assignment**
 - See also*: equals; equals (=)
 - binary operator combination with equals operator; 11
 - compatibility, term definition; 90
 - compound assignment operators; 213
 - conversions that apply to; 218
 - direct, parameter passing not the same as, (footnote); 90
 - expressions, statements; 230
 - initialization use, during variable declarations; 170
 - literals
 - floating-point; 168
 - integer; 167
 - `null`; 167
 - operators; 212
 - equals (=) as; 5
 - precedence; 222
 - term definition; 5
 - `String` object concatenation with; 59
- assistive technologies**
 - `javax.accessibility` package; 737
- associativity**
 - operator; 221–223
 - term definition; 221
- asSubClass method**
 - `Class` class; 401
- asterisk (*)**
 - import on demand use; 469
 - multiplication operator; 201
 - term definition; 6
 - static import on demand use; 72
 - wildcard, match multiple characters; 322
- asterisk equals (*=)**
 - assignment use; 11
- asterisk slash (aaa*/)**
 - comment delimiter; 163

- asynchronous exceptions; 283**
 - synchronous compared with; 283
 - at-sign (@)**
 - annotation type creation; 388
 - annotation use; 35
 - doc comment tag use; 483
 - string representation use; 100
 - Unicode sequence for; 482
 - atan method**
 - Math class and StrictMath class; 657
 - atan2 method**
 - Math class and StrictMath class; 657
 - atomic**
 - access
 - strategies for; 370
 - volatile variable characteristics; 372
 - term definition; 355
 - variables
 - java.util.concurrent.atomic package; 735
 - java.util.concurrent package; 733–735
 - AtomicInteger class; 735**
 - AtomicIntegerFieldUpdated class; 735**
 - Attr example class; 76**
 - documentation comments; 491–495
 - Attributed example interface; 126**
 - AttributedBody example class; 129**
 - AttributedImpl example class; 128**
 - attributes**
 - See also:* properties
 - term definition; 76
 - Unicode, Unicode Attribute Table definition of; 193
 - Attributes class; 736**
 - Attributes.Name class; 736**
 - audio**
 - javax.sound package; 739
 - sources, resource bundle objects; 691
 - authentication**
 - org.ietf.jgss package; 732
 - author**
 - doc comment tag for; 486
 - available method**
 - InputStream; 504
 - limitations on use of; 505
 - availableCharsets method**
 - Charset class; 321
 - availableProcessors method**
 - Runtime class
 - multiprocessor thread management use; 360
 - Runtime class; 676
 - await method**
 - Condition interface; 735
 - AWT (Abstract Window Toolkit)**
 - JavaBeans use; 722
 - Swing components compared with; 740
 - thread use; 370
- ## B
- b, B format specifier**
 - boolean conversion use; 629
 - Babble example class; 360**
 - Bach, Johann Sebastian**
 - quotation; 161
 - backed by**
 - a collection, term definition; 578
 - term definition; 49
 - backslash (\)**
 - char octal value; 167
 - including as a literal in a pattern; 322
 - removing special character meanings with; 322
 - special characters using (table); 750
 - backslash a (\a)**
 - alert (bell) character; 322
 - backslash b (\b)**
 - backspace character literal; 167
 - wildcard, boundary match beginning of a word; 322
 - word boundary pattern matching; 328
 - backslash backslash (\\)**
 - backslash character; 322
 - character literal; 167
 - backslash d (\d)**
 - wildcard, match any digit; 322
 - backslash double quote (\")**
 - double quote character literal; 167
 - backslash e (\e)**
 - escape character; 322
 - backslash f (\f)**
 - form feed character literal; 167
 - backslash n (\n)**
 - newline character; 322
 - newline character literal; 167
 - newline embedding in strings; 168
 - newline line separator; 519
 - LineNumberReader class use; 527
 - Unix line terminator pattern matching; 324
 - backslash quote (\')**
 - single quote character literal; 167
 - backslash r (\r)**
 - carriage return character literal; 167
 - carriage return line separator; 519
 - LineNumberReader class; 527
 - backslash r backslash n (\r\n)**
 - carriage return newline line separator; 519
 - LineNumberReader class; 527

- backslash s (\s)**
 - wildcard, match any whitespace character; 322
- backslash t (\t)**
 - tab character; 322
 - tab character literal; 167
- backslash u (\u)**
 - Unicode encoding
 - inside character literals; 167
 - multiple u's in; 162
- backslash u 0005C (\u005C)**
 - backslash Unicode encoding; 167
- backslash u 0008 (\u0008)**
 - backspace Unicode encoding; 167
- backslash u 0009 (\u0009)**
 - tab Unicode encoding; 167
- backslash u 000A (\u000A)**
 - newline Unicode encoding; 167
- backslash u 000C (\u000C)**
 - form feed Unicode encoding; 167
- backslash u 000D (\u000D)**
 - carriage return Unicode encoding; 167
- backslash u 0022 (\u0022)**
 - double quote Unicode encoding; 167
- backslash u 0027 (\u0027)**
 - single quote Unicode encoding; 167
- backslash w (\w)**
 - matching characters that are not part of a word; 328
- backspace (\u0008)**
 - character literal; 167
- BadDataSetException example class; 33**
 - chaining exceptions; 292
 - constructors for; 293
- bang (!)**
 - logical negation operator; 207
- bang equals (!=)**
 - inequality operator; 99, 206
 - term definition; 5
- BankAccount example class; 62, 134, 136, 347**
- Bankhead, Tallulah**
 - quotation; 685
- Base example class; 89**
 - generic method overloading; 271
- base10 logarithm; 658**
- Basic Multilingual Plane (BMP)**
 - basic Unicode character set, (footnote); 9
- BasicPermission class; 680**
- BCD (binary-coded decimal)**
 - extracting decimal digits from, shift operator use for; 210
- BCD example class; 210**
- BeanInfo interface; 721**
- beans**
 - java.beans package; 721–722
 - term definition; 721
- before method**
 - Calendar class; 699
 - Date class; 695
- behavior/behavioral**
 - See also:* flow of control
 - component of classes; 343
 - constant-specific, term definition; 156
 - enum; 156–159
 - extending, subclass; 25
 - filters, buffered streams as; 518
 - methods as mechanism for representing; 41
 - per-thread, locks; 346
 - streams that define; 514
 - superclass, overriding; 24
 - synchronization as component of; 352
 - thread
 - group; 376
 - thread groups modification of; 375
 - tokenizer, methods that control,
 - StreamTokenizer class; 535
- bell character**
 - special symbol for; 322
- Benchmark example class; 98**
- benchmarking**
 - harness, extensible framework design issues; 109–114
- Bernstein, Jeremey**
 - quotation; 713
- BetterName example class; 554**
- BetterStringsDemo example class; 21**
- bibliography; 755–760**
- big O notation**
 - See also:* performance
 - AbstractMap element handling
 - performance; 615
 - ArrayList element handling performance; 582
 - Arrays.sort element handling
 - performance; 608
 - balanced tree time requirements, TreeSet; 580
 - HashMap element handling performance; 590, 615
 - LinkedList element handling performance; 583
 - Map element handling performance; 589
 - notation, (footnote); 579
 - priority heap operations; 586
 - TreeMap key/value pair handling
 - performance; 594
- big-endian**
 - UTF character encoding standard; 320
- BigDecimal class; 722**
 - Formatter support for; 625
 - Scanner use, (footnote); 642

- BigInteger class;** 722
 - clearBit method; 722
 - Formatter support for; 625
 - Scanner use, (footnote); 642
 - setBit method; 722
- binary**
 - See also:* character(s); mathematics
 - data, stream transfer; 537–539
 - I/O, term definition; 500
 - name, term definition; 411, 412
 - numbers, string representation,
 - toBinaryString methods; 189
 - operators
 - arithmetic; 201
 - bitwise; 20, 208
 - combination with equals operator in
 - variable assignment; 11
 - logical; 20, 207
 - precedence; 221–222
 - shift operators; 208
 - search
 - algorithm; 310
 - arrays, Arrays.binarySearch; 608
 - lists, Collections.search; 599
 - trees
 - TreeMap class; 570
 - TreeSet class; 569
- binarySearch method**
 - Arrays class; 608
 - Collections class; 574, 599
- binding**
 - else clause in an if statement; 231
- Binding class;** 738
- bit patterns**
 - querying
 - bitCount methods; 190
 - highestOneBit methods; 190
 - lowestOneBit methods; 190
 - numberOfLeadingZeros methods; 190
 - numberOfTrailingZeros methods; 190
 - reverse methods; 190
 - rotateLeft methods; 190
 - rotateRight methods; 190
 - representation
 - floating-point; 192
 - string, toBinaryString method; 189
- bit set**
 - term definition; 633
- bit vectors**
 - EnumSet use; 596
 - resizable, BitSet class; 632
- bitCount method**
 - Integer class; 190
 - Long class; 190
- BitSet class;** 632
 - and method; 634
 - andNot method; 634
 - cardinality method; 634
 - clear method; 633
 - equals method; 634
 - flip method; 633
 - get method; 633
 - hashCode method; 634
 - length method; 634
 - nextClearBit method; 633
 - nextSetBit method; 633
 - or method; 634
 - set method; 633
 - size method; 634
 - xor method; 634
- bitwise operators;** 20, 208
 - AND (&); 208
 - complement operator (~); 209
 - expression types; 215
 - inclusive OR (|); 208
 - logical operators compared with; 208
 - shift operators; 208
 - XOR (^); 208
- blank finals**
 - cloning pitfalls relative to; 106
 - not permitted in named constants; 121
 - term definition; 46
 - variables, term definition; 172
- BLOCKED constant**
 - Thread class, State nested enum; 384
- BlockingQueue interface;** 604–606
 - offer method; 604
 - poll method; 604
 - put method; 604
 - take method; 604
- block(s);** 229–230
 - See also:* flow of control; I/O
 - creating; 232
 - exiting, break use for; 241
 - initialization; 54–55
 - enum use; 156
 - static, term definition; 55
 - term definition; 54
 - loop flow of control mechanism; 9
 - switch, term definition; 232
 - syntax; 230
 - tags,
 - doc comment, format and use; 483
 - documenting separately inherited entities
 - with; 489
 - term definition; 5, 230
 - try; 286–291
 - of Unicode characters, defining,
 - UnicodeBlock class; 195

blocking

See also: synchronized/synchronization event, `CountDownLatch` class; 734
I/O, cancellation request handling; 516
operations,
 `BlockingQueue` interface; 604-606
 interrupt use for canceling; 367
 queues, `BlockingQueue` interface; 604-606
 reading streams without; 505
 thread, thread scheduler handling; 358

BMP (Basic Multilingual Plane)

basic Unicode character set, (footnote); 9

body

comment, term definition; 489
method, term definition; 3, 57

Body example class; 42, 50

accessor methods; 67
constructor syntax illustration; 55

BodyPrint example class; 57**Bombeck, Erma**

quotation; 115

Boolean class

`boolean` primitive type representation by;
 187
converting, strings to/from (table); 316
`getBoolean` method; 665
`parseBoolean` method (table); 316
reading, `DataInput.readBoolean`; 537
wrapper class, `boolean` primitive type; 166
wrapper type hierarchy, (figure); 183
writing, `DataOutput.writeBoolean`; 537

boolean data type

See also: data types
arrays, name notation; 412
`Boolean` class representation of; 187
boxing conversion range; 199
converting to `String`, before adding to a
 string buffer; 332
converting to/from strings (table); 316
definition; 4
field values, default; 45
reading, `DataInput.readBoolean`; 537
reserved keyword; 165
value of; 166
writing, `DataInput.writeBoolean`; 537

boolean(s)

equality operators use; 206
expressions, term definition; 5
format conversion; 629
inversion; 208
literals; 167
operators; 20, 206, 207, 208
 conditional; 20
primitive type; 5
tests, file information; 545

bootstrap

class loader, term definition; 438

Boss Tweed

quotation; 396

boundary(s)

matchers
 special characters for (table); 752
 wildcards, match beginning or end of a
 sequence; 322
text, parsing; 712
word, pattern matching; 328

bounds

See also: generic; wildcard(s)
arrays, term definition; 19
erasure impact on; 267
`getBounds` method, `TypeVariable`
 interface; 426-427
lower
 term definition; 257
 wildcard, generic type use; 257
non-anchoring, line scanning horizon value
 as; 646
type parameters; 252-253
 term definition; 253
upper
 term definition; 253
 wildcard, generic type use; 257
wildcards, term definition; 32, 257

boxing conversions; 198-199

term definition; 91, 184, 198
wrapping vs., (footnote); 184

braces (curly {})

array initializers; 175
interface declaration use; 120

brackets

angle (<>), generic type definition use; 29
square ([])
 array syntax use; 2, 19

break statement; 241-244

cleanup from, using `finally`; 290
exiting `switch` with; 234
labeled vs. unlabeled; 241
reserved keyword (table); 165
`switch` terminator; 235
text, `BreakIterator` class use; 712

BreakIterator class; 712

`getCharacterInstance` method; 712
`getLineInstance` method; 712
`getSentenceInstance` method; 712
`getWordInstance` method; 712

Brecht, Bertolt

quotation; 567

bridge method

term definition; 746

- browsers**
 - applet lifecycle determination by; 720
 - type, reflection classes use for writing; 397
- Buffered streams**; 518–520
 - family of; 500
 - stream that defines behavior; 514
- BufferedInputStream class**; 518
- BufferedOutputStream class**; 518
- BufferedReader class**; 518
 - LineNumberReader as subclass of; 528
 - readLine method; 519
- BufferedWriter class**; 518
 - newLine method; 519
- buffers**
 - See also*: pipes
 - high performance I/O use; 565
 - mapped, java.nio handling; 566
 - string
 - appending to; 332
 - capacity management; 335
 - deleting characters in; 333
 - extending; 332
 - inserting into; 332
 - modifying; 331
 - reversing the ordering of characters in; 333
 - term definition; 499, 565
- building strings, StringBuilder use**; 330–335
- bundles (resource)**; 688–693
 - See also*: internationalization; localization
 - term definition; 689
- busy-waiting**
 - See also*: concurrency
 - term definition; 356
- Butler, Samuel**
 - quotation; 397
- Byte class**; 188–191
 - converting, strings to/from (table); 316
 - parseByte method (table); 316
 - wrapper class, byte primitive type; 166
 - wrapper type hierarchy, (figure); 183
- byte data type**
 - See also*: data types
 - arrays
 - converting to/from strings; 319
 - name notation; 412
 - boxing conversion range; 199
 - byteValue method, Number class; 188
 - converting to/from strings (table); 316
 - definition; 4
 - field values, default; 45
 - reading, DataInput.readByte; 537
 - reserved keyword; 165
 - value of; 166
 - writing, DataInput.writeByte; 537
- byte(s)**
 - arrays of, source or destinations of
 - ByteArray streams; 521
 - files
 - random access; 541
 - reading, FileInputStream class; 541
 - writing, FileOutputStream class; 541
 - flushing; 505
 - reading; 503
 - DataInput.readByte; 537
 - written by DataOutput.writeBytes, strategy for; 538
 - retrieving, String.getBytes; 319
 - reversing, reverseBytes methods; 190
 - skipping; 503
 - a file, RandomAccessFile.skipBytes; 542
 - streams; 501–506
 - ByteArray; 521
 - converting to characters, (footnote); 667
 - converting to character streams from; 512
 - Data; 537
 - Data stream classes; 539
 - DataInput interface; 537
 - DataOutput interface; 537
 - Filter, Filter character streams
 - compared with; 516
 - object, Object streams; 549
 - object, serialization use; 549
 - OutputStream; 505–506
 - PrintStream class; 525
 - PushbackInputStream class; 529–531
 - sequencing, SequenceInputStream class; 528
 - synchronization strategy; 515
 - term definition; 500
 - type tree for, (figure); 502
 - term definition; 500
 - writing; 505
 - DataOutput.writeByte; 537
 - writing to a file, with buffered output streams; 519
- ByteArray streams**; 521–522
 - in-memory stream; 514
- ByteArrayInputStream class**; 521
- ByteArrayOutputStream class**; 522
 - reset method; 522
 - size method; 522
 - toByteArray method; 522
 - toString method; 522
 - writeTo method; 522
- ByteBuffer class**
 - getFloat method; 565
 - putLong method; 565

bytecodes

See also: virtual machine
loading; 435–444
term definition; 2, 38

byteValue method

Number class; 188

C**c, C format specifier**

character conversion use; 629

C language

bibliographic reference; 758

C++ language

further reading; 758

cache/caching

LRU, LinkedHashMap use; 592
soft references use for; 457

CalcThread example class; 367**Calendar class; 696**

add method; 698
after method; 699
before method; 699
calendar fields (table); 698
clear method; 698, 699
Date class superseded by; 695
get method; 698
getAvailableLocales method; 697
getFirstDayOfWeek method; 699
getGreatestMinimum method; 698
getInstance method; 697
 locale use; 687
getLeastMaximum method; 698
getMaximum method; 698
getMinimalDaysInFirstWeek method; 699
getMinimum method; 698
getTime method; 699
getTimeInMillis method; 696
isLenient method; 699
isSet method; 698
roll method; 698
set method; 698, 699
setFirstDayOfWeek method; 699
setLenient method; 699
setMinimalDaysInFirstWeek method; 699
setTime method; 699

calendar field

DateFormat class; 705

calendar(s); 696–698

fields; 698
Gregorian; 696
lenient, term definition; 699
representation of; 695–703

Callable interface; 734**Calvin and Hobbes**

quotation; 39

cancel method

TimerTask class; 654

cancellation

See also: exception(s); finalization;
 initialization
blocking I/O, request handling; 516
blocking operation,
 InterruptedException class; 605
threads; 365–367
 Thread.interrupt vs. Thread.stop;
 381

CANON_EQ flag

Pattern class; 324

canonical

equivalence, pattern matching; 324
name
 package, import on demand use; 469
 term definition; 411
ordering, strings; 309
path, term definition; 544
string ordering, creating; 309

canRead method

File class; 545

canWrite method

File class; 545

capacity

See also: buffer(s); queue(s)
blocking queue, accessing; 605
initial, hashtable design, HashMap class; 591
management, ArrayList; 582
string buffer, management of; 335

capacity method

StringBuilder class; 335
Vector class; 618

capacityIncrement field

Vector class; 619

capture

conversion, term definition; 266
of string parts, regular expression use for; 322
wildcard; 264–266

CardDealer example interface; 125**cardinality method**

BitSet class; 634

caret (^)

bitwise exclusive or (**XOR**) operator; 208
exclusive **OR** operator; 20
logical exclusive **OR** operator; 207
multiline mode, dot-all mode pattern
 matching; 324
start-of-line boundary marker, multi-line
 input use; 646
wildcard
 boundary match beginning of a line; 322
 negations; 322

Carlyle, Thomas

quotation; 303

- carriage return newline** (`\r\n`)
 - line separator; 519
 - `LineNumberReader` class; 527
- carriage return** (`\r`)
 - character literal; 167
 - line separator; 519
 - `LineNumberReader` class; 527
- Carroll, Lewis**
 - quotation; 229
- case**
 - See also:* text
 - conversion, `Character` class methods; 194
 - default locale specifications; 687
 - folding, Unicode-aware, pattern matching; 324
 - ignoring, string comparisons; 575
 - insensitive pattern matching; 324
 - label
 - `switch` statement use; 232
 - term definition; 232
 - locale sensitivity of; 315
 - lower
 - `StreamTokenizer.LOWER_CASE_MODE`; 536
 - term definition; 193
 - testing for, `Character.isLowerCase`; 194
 - reserved keyword (table); 165
 - strings, conversion to/from upper and lower case; 315
 - title
 - term definition; 193
 - testing for, `Character.isTitleCase`; 194
 - titlecase, Unicode case concept; 309
 - Unicode; 165
 - standard for; 193
 - upper
 - term definition; 193
 - testing for, `Character.isUpperCase`; 195
- CASE_INSENSITIVE flag**
 - `Pattern` class; 324
- CASE_INSENSITIVE_ORDER field**
 - `String` class, case-insensitive `Comparator` located in; 575
- cast method**
 - `Class` class; 414
- cast/casting**
 - See also:* boxing conversions; expressions; parameterized types
 - arrays; 177
 - assignment compatibility use; 212
 - conversions that apply to; 218
 - erasure impact on use of; 268–270
 - generic types use instead of; 249
 - hidden constant access use; 123
 - `instanceof` operator use in conjunction with; 93
 - integer literals; 168
 - operator, narrowing conversions requirement of; 91
 - operators, precedence; 221
 - `ParameterizedType.getRawType` use; 427
 - reflection equivalent operation; 401
 - reflection use, `Class.cast`; 414
 - safe, term definition; 92
 - term definition; 91, 219
 - `Throwable` to actual exception types; 293
 - type; 219
 - explicit; 91–92
 - term definition; 27
 - unchecked, generic method pitfalls; 261
 - unsafe, term definition; 92
- catch**
 - clause; 286–291
 - embedded methods compared to; 287
 - erasure impact on use of; 268
 - keyword, reserved keyword (table); 165
 - statement, `try-catch-finally` sequence; 33
 - term definition; 32
- catching exceptions; 286–291**
 - See also:* exception(s)
 - term definition; 32, 279
- cbrt method**
 - `Math` class and `StrictMath` class; 658
- Cedar language**
 - bibliographic reference; 758
- ceil method**
 - `Math` class and `StrictMath` class; 658
- ceiling; 658**
- Cell example class; 93**
 - generic declaration; 247
 - nested generic types; 253
- census taker**
 - quotation; 160
- certificates**
 - See also:* security
 - `java.security.cert` package use; 732
- chain(ing)**
 - assignment expressions; 213
 - exceptions; 291–294
 - `Filter` streams; 518
 - of references, term definition; 447
 - streams, with `Filter` streams; 516–518
- Changeable example interface; 148**
- changeCondition example method**
 - `notify` code use; 355
- changing**
 - See:* modifying/modification

channels

- byte File stream methods; 540
- file, retrieving,
 - RandomAccessFile.getChannel; 543
- term definition; 499, 565

char data type

See also: data types

- arrays
 - adding to a string buffer; 332
 - creating from strings; 319
 - extracting from a StringBuilder object; 333
 - mapping strings to/from (table); 317
 - name notation; 412
- boxing conversion range; 199
- casting to integers; 219
- character arithmetic with; 202
- Character class representation of; 192
- converting to String, before adding to a string buffer; 332
- converting to/from strings (table); 316
- decrement operator use with; 205
- definition; 4
- field values, default; 45
- implicit conversion to int; 216
- Print stream handling; 525
- reading, DataInput.readChar; 537
- reserved keyword; 165
- UTF-16 use; 162
- value of; 166
- writing, DataInput.writeChar; 537

Character class; 192–198

- charCount method; 196
- codePointAt methods; 196
- codePointBefore methods; 196, 197
- codePointCount methods; 197
- COMBINING_SPACING_MARK constant; 195
- CONNECTOR_PUNCTUATION constant; 195
- CONTROL constant; 195
- CURRENCY_SYMBOL constant; 195
- DASH_PUNCTUATION constant; 195
- DECIMAL_DIGIT_NUMBER constant; 195
- digit method; 193
- ENCLOSING_MARK constant; 195
- END_PUNCTUATION constant; 195
- FINAL_QUOTE_PUNCTUATION constant; 195
- forDigit method; 193
- getNumericValue method; 193
- getType method; 195
- INITIAL_QUOTE_PUNCTUATION constant; 195
- isDefined method; 194
- isDigit method; 194
- isHighSurrogate method; 195
- isIdentifierIgnorable method; 194
- isISOControl method; 194

- isJavaIdentifierPart method; 194
- isJavaIdentifierStart method; 194
- isLetter method; 194
- isLetterOrDigit method; 194
- isLowerCase method; 194
- isLowSurrogate method; 195
- isSpaceChar method; 194
- isSupplementaryCodePoint method; 195
- isSurrogatePair method; 195
- isTitleCase method; 194
- isUnicodeIdentifierPart method; 194
- isUnicodeIdentifierStart method; 194
- isUpperCase method; 195
- isValidCodePoint method; 195
- isWhitespace method; 195
 - Scanner class use; 643
- LETTER_NUMBER constant; 195
- LINE_SEPARATOR constant; 195
- LOWERCASE_LETTER constant; 195
- MATH_SYMBOL constant; 195
- MAX_RADIX constant; 192
- MAX_VALUE constant; 192
- MIN_RADIX constant; 192
- MIN_VALUE constant; 192
- MODIFIER_LETTER constant; 195
- MODIFIER_SYMBOL constant; 195
- NON_SPACING_MARK constant; 195
- offsetByCodePoints methods; 196, 197
- OTHER_LETTER constant; 195
- OTHER_NUMBER constant; 195
- OTHER_PUNCTUATION constant; 195
- OTHER_SYMBOL constant; 195
- PARAGRAPH_SEPARATOR constant; 195
- PRIVATE_USE constant; 195
- SPACE_SEPARATOR constant; 195
- START_PUNCTUATION constant; 195
- SURROGATE constant; 195
- TITLECASE_LETTER constant; 195
- toChars methods; 196, 197
- toCodePoint method; 198
- toLowerCase method; 194
- toTitleCase method; 194
- toUpperCase method; 194
- UNASSIGNED constant; 195
- UPPERCASE_LETTER constant; 195
- wrapper class, char primitive type; 166
- wrapper type hierarchy, (figure); 183

character(s)

- See also:* ASCII; case; EBCDIC; string(s); text; Unicode; US-ASCII character encoding
- alert (bell), special symbol for; 322
- appending, to matched characters; 327
- arithmetic; 202
- array, constructing with toCharArray; 22

character(s) (*cont.*)

- arrays of, source or destinations of
 - CharArray streams; 522
- bell, special symbol for; 322
- bitwise operators use; 208
- case
 - term definition; 193
 - Unicode as standard for; 194
- case-independent comparison; 194
- Character class; 193
- character encoding standards (table); 754
- CharArray streams; 522–523
- CharArrayReader class; 522
- CharArrayWriter class; 523
- CharConversionException class; 563
- class
 - See also:* regular expressions
 - setting, StreamTokenizer methods for; 534
 - special characters for (table); 751
- comment, setting, as tokenizer character class determiner; 534
- converting byte streams to, (footnote); 667
- decrement operator use with; 205
- default settings, StreamTokenizer class; 535
- deleting, from a string buffer; 333
- encoding
 - required support; 320
- escape, special symbol for; 322
- expression types; 216
- files
 - random access; 541
 - reading, FileReader class; 541
 - writing, FileWriter class; 541
- finding occurrences of in a string
 - first, String.indexOf; 308
 - last, String.lastIndexOf; 307
- flushing; 511
- format conversions; 629
- increment operator use with; 205
- length, retrieving, CharSequence.length use; 306
- literals; 167
- matched, replacing; 326–330
- multiple, wildcard for matching; 322
- numeric value, retrieving,
 - Character.digit; 193
- octal representation, strings; 169
- ordinary, term definition; 532
- parsing text into, with BreakIterator class; 712
- primitive type definition; 4
- printing, specifying minimum number of; 23

- reading; 507
 - CharArrayReader class; 523
 - written by DataOutput.writeChars, strategy for; 538
- retrieving
 - from strings, CharSequence.charAt use; 305
 - String.getChars; 318
- reversing the ordering of, a string buffer; 333
- sequence, wildcard for matching beginning or end of; 322
- sequences; 305–306
 - regions, in pattern matching; 329
 - regular expression search in; 321
- sequences of, String manipulation of; 22
- set; 161–163
 - decoders and encoders, rapid I/O use; 565
- single, wildcard for matching; 322
- skipping; 509
- source, for Scanner class use, Readable objects; 642
- special
 - (table); 750
 - term definition; 532
- streams; 507–511
 - buffered, lines of text handling by; 519
 - CharArray; 523
 - Filter, Filter byte streams compared with; 516
 - LineNumberReader class; 525
 - PrintWriter class; 525
 - PushbackReader class; 529–531
 - Reader; 507
 - standard streams relationships with; 511–512
 - String; 523–524
 - synchronization strategy; 515
 - term definition; 500
 - type tree for, (figure); 507
 - Writer; 510–511
- subsequence, retrieving,
 - CharSequence.subSequence use; 306
- supplementary
 - term definition; 162
 - working with; 196–198
- tab, special symbol for; 322
- term definition; 500
- testing, methods for; 194
- Unicode
 - Java character set; 8
 - overview; 8–9
 - range handled by StreamTokenizer; 532
 - value for digit, retrieving,
 - Character.forDigit; 193
- whitespace, wildcard for matching; 322

- writing; 510–511
 - CharArrayWriter class; 523
- Character.Subset class**; 195
- CharArray streams**; 522–523
 - in-memory stream; 514
- CharArrayReader class**; 522
- CharArrayWriter class**; 523
 - reset method; 523
 - size method; 523
 - toCharArray method; 523
 - toString method; 523
 - writeTo method; 523
- charAt method**
 - CharSequence interface; 305
 - String; 22
 - String class; 307
- CharBuffer class**
 - CharSequence interface implemented by; 305
 - mapped buffer use; 566
 - Scanner use; 642
- CharCast example class**; 220
- CharConversionException class**; 563
- charCount method**
 - Character class; 196
- CharSequence interface**; 305–306
 - charAt method; 305
 - copy constructor; 54
 - length method; 306
 - mapped buffer use; 566
 - regular expression compilation use; 323
 - String representation relationship, in
 - Writer class; 511
 - StringBuilder implementation of; 331
- Charset class**; 320
 - availableCharsets method; 321
 - defaultCharset method; 321
 - encoding use; 512
 - forName method; 320
- CharsetDecoder class**; 320
 - encoding use; 512
- CharsetEncoder class**; 320
 - encoding use; 512
- checkAccess method**
 - Thread class; 376
 - ThreadGroup class; 378
- checked**
 - See also*: exception(s)
 - exceptions
 - constructor use of; 54
 - initialization, restrictions; 54, 55
 - term definition; 32, 34, 280
 - unchecked exceptions vs.; 280
 - wrappers; 601–602
- checkedCollection method**
 - Collection class; 602
- CheckedInputStream class**; 737
- checkedList method**
 - Collection class; 602
- checkedMap method**
 - Collection class; 602
- CheckedOutputStream class**; 737
- checkedSet method**
 - Collection class; 602
- checkedSortedMap method**
 - Collection class; 602
- checkedSortedSet method**
 - Collection class; 602
- checkError method**
 - stream error checking use of; 525
- checkPermission method**
 - AccessController class; 680
 - SecurityManager class; 678
- checksums**
 - Adler32 class; 737
 - CheckedInputStream class; 736
 - CheckedOutputStream class; 737
 - CRC32 class; 737
- ChessPiece example enum class**
 - constant-specific behavior; 157
- Chesterton, G. K.**
 - quotations; 465
- child process**
 - See also*: process(es)
 - term definition; 667
- childValue method**
 - InheritableThreadLocal class; 383
- ChoiceFormat class**; 710
- Circle example class**; 9
- clarity**
 - See also*: design issues and strategies
 - programming, correctness tradeoffs; 279
- Class class**; 397–414
 - asSubClass method; 401
 - cast method; 414
 - forName method; 398, 400, 411–413
 - wildcard return values; 401
 - getCanonicalName method; 398, 411
 - getClass method; 400
 - getClasses method; 408
 - getClassLoader method; 437, 438
 - getComponentType method; 407
 - getConstructor method; 409
 - getConstructors method; 408
 - getDeclaredClasses method; 408
 - getDeclaredConstructors method; 408
 - Constructor class use; 425
 - getDeclaredConstructor method; 409
 - getDeclaredField method; 409
 - getDeclaredFields method; 408
 - getDeclaredMethod method; 409
 - getDeclaredMethods method; 408

Class class (*cont.*)

- getDeclaringClass method; 407
- getEnclosingConstructor method; 407
- getEnclosingMethod method; 407
- getEnumConstants method; 407
- getFields method; 408
- getGenericInterfaces method; 406
- getGenericSuperclass method; 405, 406
- getInterfaces method; 407
- getMethod method; 409
- getMethods method; 408
- getModifiers method; 406
- getName method, retrieving fully qualified Class object names with; 412
- getPackage method; 479
- getResourceAsStream method; 442
- getSimpleName method; 411
- getSuperclass method; 398, 407
- isAnnotation method; 405
- isAnonymousClass method; 406
- isArray method; 405
- isAssignableFrom method; 414
- isEnum method; 405
- isInstance method; 414
- isInterface method; 405
- isLocalClass method; 406
- isMemberClass method; 406
- isPrimitive method; 405
- isSynthetic method; 406
- literals; 169
 - retrieving Class objects with; 400
 - retrieving primitive types with; 413
- lock
 - static synchronized data use; 351
 - static synchronized method use; 348
- newInstance method; 423
- reflection use of; 397, 400
- representation of old fields in serialized data use; 560
- retrieving objects of; 400
- serialization issues; 558
- toString method; 405
- wrapper class TYPE field reference to; 186

CLASS constant

- RetentionPolicy class; 395

class(es)

See also: classes (example); classes (Java-defined); field(s); interface(s); member(s); method(s)

- abstract; 97–99
 - interfaces vs.; 131
- annotation of; 392–393
- anonymous inner, initialization block use; 55

character

See also: regular expressions
 setting, StreamTokenizer methods for; 534

- Class.getClasses; 408

- Class.getDeclaredClasses; 408

- class members; 12

- concrete, implementing generic interfaces; 30

- creating, at runtime; 432–435

- declarations,

- term definition; 12

- retrieving, Class.getDeclaringClass method; 407

- defining; 42

- enum; 151–160

- extended, designing a class for extension; 108–114

- extending; 75–116

- constructors in; 80–83

- design strategies; 108–114

- overview; 24–27

- extensible, interfaces as critical component of design; 131

- extension, generic types and; 276–277

- fields; 14

- final

- array behavior like; 178

- design considerations; 96

- generic

- declaration; 247

- overview; 29–32

- initialization, final step in preparing a class for use; 441

- inner; 136–142

- anonymous; 144–146

- anonymous, enum constants as; 158

- Constructor class vs. Class class; 425

- declaration statements; 230

- enclosing object access by; 138

- extending; 139–140

- extending, by unrelated inner classes; 140

- hiding of fields and methods by; 140–142

- local; 142–144

- restrictions on; 137

- scope of; 140

- in static contexts; 144

- term definition; 136

- use of synchronized with enclosing object; 351

- interfaces compared with; 117

- linking, second step in preparing a class for use; 441

- literals; 169

- erasure impact on use of; 268

- obtaining type tokens with; 401

- raw types use with; 270
- retrieving primitive types with; 413
- synchronized statement use; 351
- loader; 435-444
 - bootstrap, term definition; 438
 - context, term definition; 437
 - default; 437
 - first step in preparing a class for use; 441
 - parent, term definition; 438
 - system, term definition; 438
 - term definition; 436
- local, declaration statements; 230
- members; 42
 - accessing; 223
 - controlling access to; 47-48
 - examining; 408
 - hiding of; 88
- methods; 17, 58
 - and objects relationship to; 41
 - term definitions; 17, 58
- modifiers; 43
- naming; 411-413
- nested; 133-150
 - class members; 43, 134
 - interface members; 121, 134
 - in interfaces; 148
 - static; 133, 134
 - term definition; 133
- objects and; 41-74
- overview; 12-15
- path, term definition; 435
- preparing for use; 441
- primitive types represented by; 183
- public, file relationship to; 43
- as reference types; 166
- relationship to contract; 41
- representation by Class objects; 399
- representation of types; 183
- serialization requirements; 551
- super, retrieving,
 - Class.getSuperClasses method; 407
- system, term definition; 438
- term definition; 1, 41
- thread-blind, converting to multithreaded
 - environments; 352
- variables; 44, 45
 - definition of; 8
 - initialization with static initialization
 - block; 55
 - referencing; 8
 - term definition; 14
- wrapper
 - primitive types; 4, 183
 - programming with; 183-200
 - term definition; 183
 - type hierarchy, (figure); 183
- class keyword**
 - arrays, name notation; 412
 - class declaration use; 42
 - reserved keyword (table); 165
- ClassCastException class**; 27, 746
 - casting errors; 92
 - Class.newInstance use; 424
 - conventions of use; 571
 - incompatible ordering as cause of; 600
 - Map interface use; 588
 - RuntimeException extension; 284
 - SequenceInputStream class use; 528
- ClassContents example class**; 410
- classes (example)**
 - See example classes:* ABLIterator;
 - AbstractBase; Action;
 - ArrayBunchList; Attr;
 - AttributedBody; AttributedImpl;
 - Babble; BadDataSetException;
 - BankAccount; Base; BCD; Benchmark;
 - BetterName; BetterStringsDemo;
 - Body; BodyPrint; CalcThread; Cell;
 - CharCast; ChessPiece Circle;
 - ClassContents; Collection;
 - ColorAttr; Concat; Concrete1;
 - Concrete2; Concrete3; Container;
 - CountBytes; CountSpace;
 - DataHandler; Date1; Date2;
 - DebugProxy; Deck; Device;
 - DirFilter; EnhancedSerialPort;
 - ExtendedInner; ExtendedOuter;
 - ExtendShow; Eye; Fibonacci;
 - Fibonacci2; FieldModifiers;
 - FindChar; Friendly; Game;
 - GlobalHello; HerClass;
 - HighSpeedPrinter; Host;
 - ImprovedFibonacci; Inner;
 - IntegerLookup; IntegerStack; Iter;
 - Iterator; Lookup; MemoryWatchTask;
 - MethodBenchmark; More; MyClass;
 - MyUtilities; Name;
 - NoSuchAttributeException;
 - Operations; Outer; PassByValue;
 - PassRef; Permissions; PingPong;
 - Pipe; Pixel; Player; PlayerLoader;
 - Point; Polygon; Port; Primes;
 - Printer; PrintQueue; PrintServer;
 - PrintServer2; PriorityQueue;
 - ProcessFile;
 - ReadPastEndException;
 - ReaperThread; Rectangle;
 - RegionMatch; ResourceImpl;
 - ResourceManager; Reuse;
 - RunPingPong; SeparateGroups;
 - SequenceCount; SerialPort;
 - ShortStrings; ShowJoin;

classes (example) (cont.)*See example classes (cont.):*

SimpleClassDesc; SimpleLookup;
 SimpleSortDouble;
 SingleLinkQueue; SortDouble;
 SortedCharSeqCollection; Sorter;
 SortMetrics; StreamEndException;
 Suit; SuperShow; TestSort;
 TextGenerator; TranslateByte;
 TypeDesc; Unrelated;
 UppercaseConverter; Users; Value;
 WhichChars; X; Y

classes (Java-defined)*See classes:* AbstractCollection;

AbstractList; AbstractMap;
 AbstractQueue;
 AbstractSequentialList;
 AbstractSet;
 AccessControlContext;
 AccessControlException;
 AccessController;
 AccessibleObject; ActionEvent;
 Activatable; Adler32; Annotation;
 AnnotationTypeMismatchException;
 Applet; ArithmeticException;
 Array; ArrayBlockingQueue;
 ArrayIndexOutOfBoundsException;
 ArrayList; Arrays;
 ArrayStoreException;
 AssertionError; AtomicInteger;
 AtomicIntegerFieldUpdated;
 Attributes; Attributes.Name;
 BasicPermission; BigDecimal;
 BigInteger; Binding; BitSet;
 Boolean; BreakIterator;
 BufferedInputStream;
 BufferedOutputStream;
 BufferedReader; BufferedWriter;
 Byte; ByteArrayInputStream;
 ByteArrayOutputStream;
 ByteBuffer; Calendar; Character;
 Character.Subset;
 CharArrayReader; CharArrayWriter;
 CharBuffer;
 CharConversionException; Charset;
 CharsetDecoder; CharsetEncoder;
 CheckedInputStream;
 CheckedOutputStream;
 ChoiceFormat; Class;
 ClassCastException; ClassLoader;
 ClassNotFoundException;
 CloneNotSupportedException;
 CodeSource; CollationKey;
 Collator; Collections; Component;
 CompositeName; ConcurrentHashMap;
 ConcurrentModificationException;

Constructor; Container;
 CopyOnWriteArrayList;
 CountdownLatch; CRC32; Currency;
 CyclicBarrier; DateFormat;
 DatagramSocket;
 DataInputStream;
 DataOutputStream; Date;
 DateFormat; DecimalFormat;
 Deflater; DeflaterOutputStream;
 DelayQueue; Deprecated;
 Dictionary; Documented; Double;
 DuplicateFormatFlagsException;
 ElementType; EmptyStackException;
 Enum;
 EnumConstantNotPresentException;
 EnumMap; EnumSet; EOFException;
 Error; Exception;
 ExceptionInInitializerError;
 Exchanger; Field; FieldPosition;
 File; FileChannel; FileDescriptor;
 FileInputStream; FileLock;
 FileNotFoundExcpetion;
 FileOutputStream; FilePermission;
 FileReader; FileWriter; Filter;
 FilterInputStream;
 FilterOutputStream; FilterReader;
 FilterWriter; Float; FloatBuffer;
 FlowLayoutManager; Format;
 FormatFlagsConversionMismatchEx
 ception; FormattableFlags;
 Formatter;
 FormatterClosedException;
 FutureTask; GregorianCalendar;
 GridBagLayoutManager;
 GZipInputStream;
 GZipOutputStream; HashMap;
 HashSet; Hashtable;
 IdentityHashMap;
 IllegalAccessException;
 IllegalArgumentException;
 IllegalFormatCodePointException;
 IllegalFormatConversionException;
 IllegalFormatException;
 IllegalFormatFlagsException;
 IllegalFormatPrecisionException;
 IllegalFormatWidthException;
 IllegalMonitorStateException;
 IllegalStateException;
 IllegalThreadStateException;
 IncompleteAnnotationException;
 IdentityHashMap;
 IndexOutOfBoundsException;
 InetAddress; Inflater;
 InflaterInputStream;
 InheritableThreadLocal;
 Inherited; InitialContext;

InputMismatchException;
 InputStream; InputStreamReader;
 InstantiationException; Integer;
 InterruptedException;
 InterruptedIOException;
 InvalidClassException;
 InvalidObjectException;
 InvocationTargetException;
 IOException; JarEntry; JarFile;
 JarInputStream; JarOutputStream;
 LineNumberReader; LinkageError;
 LinkedBlockingQueue;
 LinkedHashMap; LinkedHashSet;
 LinkedList; ListIterator;
 ListResourceBundle; Locale; Long;
 MalformedParameterizedTypeException;
 Manifest; MappedByteBuffer;
 Matcher; MatchResult; Math;
 MathContext; MessageFormat;
 Method; MissingBundleException;
 MissingFormatArgumentException;
 MissingFormatWidthException;
 Modifier; MulticastSocket; Name;
 NameClassPair; NetPermission;
 NoSuchElementException;
 NoSuchFieldException;
 NoSuchMethodException;
 NotActiveException;
 NotSerializableException;
 NullPointerException; Number;
 NumberFormat;
 NumberFormatException; Object;
 ObjectInputStream;
 ObjectInputStream.GetField;
 ObjectOutputStream.PutField;
 ObjectOutputStreamClass;
 ObjectOutputStreamException;
 ObjectOutputStreamField; Observable;
 OptionalDataException;
 OutOfMemoryError;
 OutOfMemoryError; OutputStream;
 OutputStreamWriter; Override;
 Package; ParseException;
 Pattern; PatternSyntaxException;
 Permission; PhantomReference;
 PipedInputStream;
 PipedOutputStream; PipedReader;
 PipedWriter; Policy; PrintStream;
 PrintWriter;
 PriorityBlockingQueue;
 PriorityQueue;
 PrivilegedExceptionAction;
 Process; ProcessBuilder;
 Properties; PropertyPermission;
 PropertyResourceBundle;
 ProtectionDomain; Proxy;
 PushbackInputStream;
 PushbackReader; Random;
 RandomAccessFile; Reader;
 ReentrantLock; Reference;
 ReferenceQueue;
 ReflectPermission;
 RemoteException; ResourceBundle;
 Retention; RetentionPolicy;
 RMISecurityManager; RoundingMode;
 Runtime; RuntimeException;
 RuntimePermission; Scanner;
 ScheduledThreadPoolExecutor;
 Security; SecurityException;
 SecurityManager;
 SecurityPermission; Semaphore;
 SequenceInputStream;
 SerializablePermission;
 ServerSocket; Short;
 SimpleDateFormat; SimpleTimeZone;
 Socket; SocketPermission;
 SoftReference; Stack;
 StackTraceElement;
 StreamCorruptedException;
 StreamException; StreamTokenizer;
 StrictMath; String; StringBuffer;
 StringBuilder;
 StringIndexOutOfBoundsException;
 StringReader; StringTokenizer;
 StringWriter; SuppressWarnings;
 SyncFailedException;
 SynchronousQueue; System; Target;
 Thread; ThreadGroup; ThreadLocal;
 ThreadPoolExecutor; Throwable;
 Timer; TimerTask; TimeUnit;
 TimeZone; TreeMap;
 TypeNotPresentException;
 UncaughtExceptionHandler;
 UndeclaredThrowableException;
 UnicastRemoteObject;
 UnicodeBlock;
 UnknownFormatConversionException;
 UnknownFormatFlagsExceptions;
 UnsatisfiedLinkError;
 UnsupportedCharsetException;
 UnsupportedOperationException;
 URI; URL; URLConnection;
 URLEncoder;
 UTFDataFormatException; UUID;
 Vector; VirtualMachineError; Void;
 WeakHashMap; WeakReference;
 WriteAbortedException; Writer;
 ZipEntry; ZipFile; ZipInputStream;
 ZipOutputStream
ClassLoader class; 435–441
 clearAssertionStatus method; 444
 defineClass method; 439–440

ClassLoader class (*cont.*)

definePackage method; 479
 findClass method; 436, 439
 findLoadedClass method; 439
 findResource method; 443
 findResources method; 443
 getClassLoader method; 438
 getContents method; 442
 getParent method; 438
 getResource method; 442
 getResourceAsStream method; 442
 getResources method; 442
 getSystemClassLoader method; 438
 getSystemResource method; 442
 getSystemResourceAsStream method; 442
 loadClass method; 438–439
 resolveClass method; 441
 setClassAssertionStatus method; 444
 setDefaultAssertionStatus method; 444
 setPackageAssertionStatus method; 444

ClassNotFoundException class

annotation reflection use; 415
 Class.forName use; 413
 ObjectInputStream.readObject use; 553

cleanup

See also: cancellation; finalization
 application exit, garbage collection issues; 452
 finally clauses; 34
 guaranteed, shutdown hooks required for; 464
 internal state, finally use for; 289
 interrupted thread handling; 366
 resources, finalization issues; 450
 shutdown hooks use for; 673
 thread cancellation, Thread.interrupt vs. Thread.stop; 381

clear method

AbstractList class; 615
 BitSet class; 633
 Calendar class; 698, 699
 Collection interface; 577
 Map interface; 588
 Reference class; 455
 WeakHashMap class; 593

clearAssertionStatus method

ClassLoader class; 444

clearBit method

BigInteger class; 722

clearChanged method

Observable class; 636

clearProperty method

System class; 664

client

See also: server(s)
 client-side synchronization
 multiple method invocation use; 353
 server-side synchronization vs.; 352
 synchronized statement use of arbitrary objects; 352
 term definition; 349
 term definition; 727

clockSequence method

UUID class; 657

clone method

copying objects with; 54
 Enum class use; 159
 Object class; 100
 cloning method design use; 101–102
 default implementation use; 104
 parameterized type issues; 747

clone/cloning

See also: copy/copying
 copy constructors compared with; 54
 correct; 103
 deep
 shallow vs.; 106
 term definition; 106
 enums not permitted to; 159
 final field consideration; 47
 forbidding, strategies and pitfalls; 103
 objects; 101–107
 serialization vs.; 554
 shallow
 deep vs.; 106
 term definition; 106
 strategies; 101
 term definition; 99

Cloneable interface

arrays implementation of; 177
 cloning method use; 101, 101–102
 marker interface; 130
 property-definition interface; 118

CloneNotSupportedException class

cloning method use; 101–102
 Enum class use; 159
 use recommendations; 106

close method

InputStream; 504
 OutputStream; 505
 Reader; 509
 Scanner class; 644
 stream use; 502
 Writer; 511

Closeable interface; 502

Formatter implementation of; 632
 Scanner class interaction; 644

closing

See also: cancellation; cleanup; finalization
 conversion streams, design issues; 513
 output streams, character; 511
 streams
 importance of; 502, 507
 input byte; 504
 input character; 509
 output byte; 505

code

base
 code source relationship to; 681
 term definition; 681
 commenting out, valid and invalid means of;
 163
 font
 doc comment tags for; 487
 plain font vs., in doc comments; 485
 point, term definition; 162, 192
 privileged, term definition; 681
 source
 code base relationship to; 681
 term definition; 681
 unit, term definition; 162

codePointAt method

Character class; 196
 String class; 336
 StringBuffer class; 336
 StringBuilder class; 336

codePointBefore method

Character class; 196, 197
 String class; 336
 StringBuffer class; 336
 StringBuilder class; 336

codePointCount method

Character class; 197
 String class; 336
 StringBuffer class; 336
 StringBuilder class; 336

CodeSource class; 681**coding style**

See also: design issues and strategies
 correct finally use; 289
 labeled break; 243
 loop
 termination; 239
 test perversions; 239
 programming without a goto, alternative
 constructs; 246
 switch statements; 234
 throws clause use; 284, 285

collation; 708–709

term definition; 708

CollationKey class; 708**Collator class; 708–709**

compare method; 708
 getCollationKey method; 708
 getInstance method; 708

Collection example class; 601**Collection interface; 568, 575–577**

add method; 576
 BlockingQueue interface use; 605
 addAll method; 576
 BlockingQueue interface use; 605
 clear method; 577
 contains method; 575
 containsAll method; 576
 BlockingQueue interface use; 605
 copy constructor; 54
 isEmpty method; 575
 iterator method; 571, 575
 Map interface compared with; 587
 Queue interface extension of; 585–587
 remove method; 576
 removeAll method; 577
 retainAll method; 577
 BlockingQueue interface use; 605
 size method; 575
 toArray method; 575

collection(s); 567–622

See also: container(s); data structures
 AbstractCollection class,
 implementation starting point; 611
 backed by, term definition; 578
 characteristics of; 567
 cloning issues; 102
 concrete, type tree, (figure); 568
 concurrent, synchronized wrappers and;
 602–607
 enum-specific collection classes; 160
 for-each loop use with; 240
 generic types use in defining; 249
 interfaces; 568
 Iterable interface use in defining; 127
 legacy types; 616
 removing elements from; 572
 term definition; 567
 unmodifiable, implementation guidelines;
 612
 viewing maps with; 589
 wildcard use; 260
 wrapped; 597
 writing collection implementations; 611–616

Collections class; 597–602

addAll method; 598
 binarySearch method; 574, 599
 checkedCollection method; 602
 copy method; 599
 disjoint method; 598
 EMPTY_LIST field; 601

Collections class (*cont.*)

EMPTY_MAP field; 601
 EMPTY_SET field; 601
 emptyList method; 600
 emptyMap method; 600
 emptySet method; 600
 fill method; 598
 indexOfSubList method; 599
 lastIndexOfSubList method; 599
 max method; 597
 retrieving largest element with; 575
 min method; 597
 retrieving smallest element with; 575
 nCopies method; 599
 replaceAll method; 598
 reverse method; 598
 reverseOrder method; 597, 598
 rotate method; 598
 shuffle method; 597, 598
 singleton method; 600
 singletonList method; 600
 singletonMap method; 600
 sort method; 574, 599
 swap method; 598
 synchronizedCollection method; 602
 synchronizedList method; 602
 synchronizedMap method; 602
 synchronizedSet method; 602
 synchronizedSortedMap method; 602
 synchronizedSortedSet method; 602
 unmodifiableCollection method; 601
 unmodifiableList method; 601
 unmodifiableMap method; 601
 unmodifiableSet method; 601
 unmodifiableSortedMap method; 601
 unmodifiableSortedSet method; 601
 utility methods; 597

collisions (name)

See also: namespaces; packages
 packages as organizational tool to avoid; 36, 468

color manipulation

java.awt.color package; 719

ColorAttr example class; 78

constructors for; 80, 81

COMBINING_SPACING_MARK constant

Character class; 195

comma (,)

array initializer separator; 175
 expression separator, in for loops; 237
 floating-point conversion use (table); 628
 generic type parameters separated by; 250
 integer conversion use (table); 627
 parameter list separator; 2
 variable list separator; 5

command

line
 assertion evaluation control; 300–302
 assertion evaluation control, runtime control compared with; 444–302
 method, ProcessBuilder class; 671
 process, ProcessBuilder encapsulation of; 670

comment(s); 163

See also: coding style; design issues and strategies; documentation annotations relationship to; 387
 body, term definition; 489
 character, setting, as tokenizer character class determiner; 534
 delimiters, term definition; 6
 doc
 syntax and use; 482
 term definition; 481
 documentation; 481–498
 tags (table); 754
 term definition; 6
 nesting prohibited in; 163
 overview; 6
 skew, term definition; 497
 skipping, StreamTokenizer vs. Scanner use; 648
 StreamTokenizer handling; 536
 term definition; 6
 types of; 163

commentChar method

StreamTokenizer class; 534

communication (thread)

See also: concurrency; thread(s)
 busy-waiting avoidance; 356
 mechanisms for; 354
 preemption time control use; 359

Comparable interface; 118

compareTo method; 118, 574
 Enum class use; 159
 generics issues; 746
 wrapper classes; 186
 differences between floating-point wrappers and primitives; 186
 enums implicitly; 153
 ordering with; 574
 property-definition interface; 118
 snapshotIterator method; 574
 String class implementation of; 309
 wrapper class implementation; 186

Comparator interface

Comparable alternative; 574
 compare method; 574
 ordering with; 574
 SortedSet use; 577

comparator method

PriorityQueue class; 587
 SortedMap interface; 590
 SortedSet interface; 577

compare method

Collator class; 708
 Comparator interface; 574

compareAndSet methods

lock-free and wait-free concurrent algorithm use; 735

compareTo method

Comparable interface; 118, 574
 Enum class use; 159
 generics issues; 746
 wrapper classes; 186
 File class; 544
 natural ordering defined by; 574
 String class; 309
 wrapper classes; 186

compareToIgnoreCase method

String class; 309

comparison

See also: equality
 arrays; 177
 characters, case-independent; 194
 compareTo method, wrapper classes; 186
 mutually comparable, term definition; 118
 operators; 206
 precedence; 221–222
 term definition; 5
 reference, content comparison vs.; 311
 reverse ordering,
 Collections.reverseOrder; 597
 String objects, contents compare vs. object
 compare; 22
 string(s); 308–311, 309
 content vs. reference; 311
 contents, intern.String use for; 312
 ignoring case in; 575
 objects; 22
 strings, locale-dependent; 708

compatibility

See also: design issues and strategies
 API issues; 747
 assignment
 requirements for; 212
 term definition; 90
 bridge method role; 746
 requirements
 design issues and impact; 743
 generics design and impact; 744–748
 type, conversion and; 90–93

compilation

dynamic, shared values; 370
 volatile impact on; 370

compile method

Pattern class; 323, 641

compiler(s)

assert statement interactions; 299
 checked exceptions validated by; 280
 compile time method resolution; 227
 @deprecated tag impact on; 486
 generic type information erased by; 250
 Java, technical issues for applications; 741–748
 regular expressions handling; 323–326
 role relative to source code; 161

complement

bitwise complement operator (~); 209

complementOf method

EnumSet class; 595

completion

abrupt, term definition; 283
 application execution; 369
 run method, thread termination as result of;
 364
 threads, waiting for; 367–369

complexity

as nested generic type issue; 255

Component class; 718

paint method; 718
 repaint method; 718

components

See arrays; packages; path
 beans, term definition; 721
 Swing, javax.swing package; 740

composite/composition

See also: inheritance
 interface use in conjunction with; 117
 streams, creating with Filter streams; 516
 term definition; 129
 type-independent implementation reuse; 76

CompositeName class; 738**compound assignment operators; 213****compression**

ZLIB; 736

Concat example class; 528**concat method**

String class; 316

concatenation

byte streams, SequenceInputStream class;
 528
 operator (+), term definition; 12
 string; 12
 string concatenation operator; 214
 strings, String.concat; 316

concatenation (+) operator

String object concatenation with; 59

concatenation-assign (+=) operator; 21

String object concatenation with; 59

concrete

See also: abstract; implementation; interfaces
classes, implementing generic interfaces; 30
collections, type tree, (figure); 568

Concrete1 example class; 473**Concrete2 example class;** 474**Concrete3 example class;** 474**concurrency/concurrent**

See also: deadlock; deadly embrace; lock(s);
race; synchronized/synchronization;
threads

collections; 604–607

java.util.concurrent package; 602
contents description; 733–735

multiple file stream/same file issues; 541

Observable class mechanisms; 637

queues, BlockingQueue interface; 604–606

synchronization and; 515–516

ConcurrentHashMap class; 606**ConcurrentLinkedQueue interface;** 607**ConcurrentMap interface**

putIfAbsent method; 606

remove method; 607

replace method; 607

ConcurrentModificationException class;

574, 604

unmodifiable wrapper use; 601

weakly consistent iterator non-use of; 596

Condition interface; 734–735

await method; 735

signal method; 735

signalAll method; 735

conditional AND (&&) operator; 20, 207

evaluation order differences; 215

logical AND (&) compared with; 208

logical operations; 207

precedence; 222

conditional (?) operator; 210–212

evaluation order differences; 215

expression, type of, rules for determining;
211

if statement compared with; 211

precedence; 222

conditional OR (||) operator; 20, 207

logical inclusive OR (|) compared with; 208

precedence; 222

configuring threads; 339**conflicts**

name, packages as tool for preventing; 36

package name, compiler resolution; 470

type, protected member accessing; 94

connect method

Piped streams; 521

CONNECTOR_PUNCTUATION constant

Character class; 195

const keyword

reserved keyword (table); 165

constant(s)

See also: enum; enums; literals

advantages in localization design; 690

case labels required to be enums or; 235

constant-specific behavior, term definition;
156

declaration of; 7

defining, with final fields; 46

enum

collections for working with; 594

declarations; 154–159

term definition; 8

term definition, (footnote); 152

interface

inheriting and hiding of; 123

member; 120

named

interface declarations; 29, 121

overview; 7–8

term definition; 7

parameter values, final keyword use in
declaring; 65

term definition; 7

transcendental; 658

unnamed, literals as; 166

value, documentation of; 487

variables, term definition; 46

constant-time operations

term definition; 579

Constructor class; 423–426

getExceptionTypes method; 424

getGenericExceptionTypes method; 424

getGenericParameterTypes method; 424

getParameterAnnotations method; 424

getParameterTypes method; 424

isVarArgs method; 424

newInstance method; 425

overview and place in introspection

hierarchy; 399

CONSTRUCTOR constant

ElementType class; 394

constructors/construction; 50–54

See also: class(es); initialization

annotation of; 392–393

checked exception use; 54

checked exceptions use by; 285

Class.getConstructor; 409

Class.getConstructors; 408

Class.getDeclaredConstructors; 408

Class.getDeclaredConstructor; 409

cloning pitfalls relative to; 106

copy

cloning compared with; 106

term definition; 53

- default, term definition; 53
- enum; 155–156
 - restrictions on; 156
- exiting, return use for; 245
- explicit invocation of; 52
- extended classes; 80–83
- generic; 260–264
- initialization and; 50–56
- no-arg, term definition; 53
- order dependencies; 81–82
- overloading of; 70
- reference objects, including reference queues; 459
- reflection access; 409
- specialized, reasons for providing; 53
- starting a thread in, dangers of; 347
- superclass, invocation of; 80
- synchronized
 - modifier not used with; 347
 - statement use within; 351
- term definition; 14, 50
- thread, specifying thread group in; 376
- Thread class; 342, 347, 376
- ThreadGroup class; 377
- wrapper classes; 185
- wrappers, common to all; 185
- Container class**; 718
 - paint method; 718
- Container example class**; 242
- container(s)**
 - See also:* collections
 - data
 - buffers as; 565
 - public use in defining; 15
 - term definition; 567
- contains method**
 - Collection interface; 575
 - Hashtable class; 620
 - String class (table); 308
- containsAll method**
 - Collection interface; 576
 - BlockingQueue interface use; 605
- containsKey method**
 - Map interface; 588
- containsValue method**
 - Map interface; 588
- content comparison**
 - reference comparison vs.; 311
- Context interface**; 738
- context(s)**
 - See also:* namespaces; package(s)
 - bean, java.beans.beancontext package
 - handling; 722
 - calling, retrieving,
 - AccessController.getContext; 681
 - class loader, term definition; 437
 - expression conversions; 218
 - naming, term definition; 36
 - role in name semantics determination; 178
 - static
 - inner classes in; 144
 - term definition; 144
- continue keyword**
 - reserved keyword (table); 165
- continue statement**; 244
 - finally use for cleanup; 290
 - labeled vs. unlabeled; 244
 - loop use; 239
- contract(s)**
 - See also:* design issues and strategies; signature
 - documentation of; 497
 - inheritance,
 - term definition; 75
 - restrictions implied by; 75
 - marker interfaces as degenerate case of; 130
 - method, documenting changes in; 489
 - public and protected,
 - extensible class design issues; 108
 - as part of; 48
 - synchronization part of, enforcement
 - documentation; 353
 - term definition; 27, 41, 75
 - throws clause, enforcement of; 284
 - type, package access handling; 472
- control**
 - See also:* flow of control; security access; 47–48
 - extensible framework design; 76, 113
 - java.security.acl package use; 732
 - methods use for; 65–68
 - per-class, not per-object; 48, 68
 - term definition; 38
 - thread run methods; 344
 - assertion, runtime; 444–445
 - flow, *See*, flow of control
 - transfer of, with exception handling; 283
 - version
 - doc comments; 497
 - doc comments, tags for; 486
 - objects, serialization issues; 557
- CONTROL constant**
 - Character class; 195
- conventions**
 - exception; 570
 - external documentation; 496–497
 - naming
 - conflict avoidance use of; 36
 - Internet domain name use; 37
 - packages, Internet domain name use; 468

conversion(s)

See also: types/typing

- automatic, between primitive types and wrappers; 183–184
- boxing; 198–199
 - term definition; 91, 184, 198
- byte arrays, to/from strings; 319
- capture, term definition; 266
- case, Character class methods; 194
- floating-point, format specifiers for; 627–629
- format, format specifier use; 624–625
- integer, format specifiers for; 627
- line separator, format specification; 625
- narrowing
 - casts use for; 219
 - term definition; 91, 217
- parameterized types and raw types; 745
- primitive types to String, before adding to a string buffer; 332
- primitive-to-wrapper; 91
- sequence to array, design concerns; 85
- shift operator requirement for floating-point number use; 210
- streams
 - InputStreamReader; 512–514
 - OutputStreamWriter; 512–514
- string; 220, 316–317
 - overview; 23–24
 - to uppercase; 517
 - to/from upper and lower case; 315
- thread-blind classes for a multithreaded environment; 352
- type
 - compatibility and; 90–93
 - explicit type casts; 219
 - implicit; 215
 - term definition; 27
- types that apply to expressions; 217–218
- unboxing, term definition; 184, 198
- unchecked, term definition; 745
- widening, term definition; 91
- widening primitive, term definition; 216

Cook, Rich,

quotation; 740

copy method

Collections class; 599

copy/copying

See also: clone/cloning

- arrays; 176
 - System.arraycopy; 665
- constructors
 - cloning compared with; 106
 - copying a view with; 578
 - term definition; 53
- defensive, term definition; 127

copyInto method

Vector class; 618

copyOf method

EnumSet class; 595

CopyOnWriteArrayList class; 607**copyValueOf method**

String class; 318

CORBA org.omg.CORBA package

contents description; 717, 740

correctness

See also: design issues and strategies programming, clarity tradeoffs; 279

corruption of object state

See also: design issues and strategies Thread.stop danger; 381

COS (Common Object Service)

org.omg.CosNaming package; 717

cos method

Math class and StrictMath class; 657

cosh method

Math class and StrictMath class; 658

cosine; 657

hyperbolic; 658

CountBytes example class; 504, 505**CountDownLatch class; 734****counting**

reference, garbage collection model; 449

countObservers method

Observable class; 637

country default

establishing in Locale constructor; 687

CountSpace example class; 509**countTokens method**

StringTokenizer class; 652

covariant return type

inheritance and; 125
specification of; 102
term definition; 85

CRC-32 algorithm

CRC32 class; 737

"createClassLoader"

RuntimePermission name; 680

createNewFile method

File class; 545

createTempFile method

File class; 547

creating

See also: constructors; defining arrays

- evaluation order; 215
- explicit; 173
- generic type implications; 251
- implicit; 175

blocks; 232

buffered output stream to write bytes to a file; 519

- char arrays, from strings; 319
 - classes, at runtime; 432–435
 - directories
 - File.mkdir; 546
 - File.mkdirs; 546
 - encodings, strings; 317
 - exception types; 280
 - File streams, with FileDescriptor
 - objects; 541
 - files
 - File.createNewFile; 545
 - FileOutputStream class; 541
 - RandomAccessFile class; 541
 - generic arrays, (footnote); 428
 - initial state; 49
 - instance creation expression, term definition; 214
 - objects; 13, 49–50
 - with Class.newInstance; 423–426
 - expressions, statements; 230
 - generic type implications; 251
 - processes; 666–672
 - related strings; 313
 - String objects; 306
 - strings
 - from byte arrays; 319
 - from char arrays; 318
 - temporary files, File.createTempFile; 547
 - thread groups; 377
 - threads; 339
 - critical**
 - regions, term definition; 345
 - sections, term definition; 345
 - Croatian**
 - titlecase subtleties; 193
 - cross references**
 - doc comments
 - @link inline tag; 484
 - @linkplain inline tag; 484
 - @see tag; 483
 - cryptography**
 - java.crypto package; 732
 - java.security package use; 732
 - CSV (comma-separated-values)**
 - Scanner use; 644
 - cubic root**; 658
 - curly braces ({})**
 - array initializers; 175
 - block operators; 230
 - class member use; 2
 - interface declaration use; 120
 - @link inline tag use; 484
 - @linkplain inline tag use; 484
 - currency**
 - localization; 694
 - Currency class**; 694
 - getCurrencyCode method; 694
 - getDefaultFractionDigits method; 694
 - getSymbol method; 694
 - CURRENCY_SYMBOL constant**
 - Character class; 195
 - current**
 - object, term definition; 16
 - thread, accessing, Thread.currentThread; 341
 - currentTimeMillis method**
 - System class; 665, 695
 - cycles**
 - static initializers; 56
 - term definition; 449
 - CyclicBarrier class**; 734
- ## D
- d format specifier**
 - integer conversion use; 627
 - D/d suffix**
 - double-precision floating-point literal indicator; 168
 - daemon**
 - groups
 - daemon threads vs.; 376
 - term definition; 376
 - threads
 - shutdown hook interaction with; 673
 - term definition; 369
 - Timer class use; 654–655
 - user threads compared with; 369
 - dangling references**
 - term definition; 448
 - Darrow, Clarence,**
 - quotation; 387
 - DASH_PUNCTUATION constant**
 - Character class; 195
 - data**
 - See also:* data structures; data types
 - atomic access, strategies for; 370
 - based I/O, term definition; 500
 - binary, stream transfer; 537–539
 - communication between threads, pipes use for; 520
 - containers
 - buffers as; 565
 - public use in defining; 15
 - encapsulation, term definition; 15
 - extending, subclass; 25
 - filtering, with Print streams; 525
 - flushing to disk, strategies for; 541
 - hiding, access control modifiers impact on; 48

- data** (*cont.*)
 - modifiable
 - interface design that includes; 121
 - interfaces, inner class use; 149
 - representation, Unicode; 9
 - static, locking mechanisms; 352
 - transfer, java.awt.datatransfer package; 720
- data streams**; 537–539
 - classes for; 539
 - filter uses of; 539
- data structures**
 - See*: array(s); collection(s); hash, maps; hashtable(s); list(s); map(s)/mapping; queue(s); set(s); stack(s); streams; string(s); tree(s)
- data types**
 - See data types*: boolean; char; double; float; int; long; primitive; short
- databases**
 - relational, java.sql package; 732–733
- DateFormat class**
 - date localization with; 686
- DatagramSocket class**; 726
- DataHandler example class**; 457
- DataInput interface**; 537
 - EOFException class use; 538
 - RandomAccessFile implementation of; 541
 - readFully method; 537
 - readUnsignedByte method; 538
 - readUnsignedShort method; 538
 - skipBytes method; 538
- DataInputStream class**; 539
- DataOutput interface**; 537
 - RandomAccessFile implementation of; 541
 - writeBytes method; 538
 - writeChars method; 538
- DataOutputStream class**; 539
 - encoding used by, default serialization use of; 551
- Date class**; 695
 - after method; 695
 - before method; 695
 - fully-qualified name example use; 36
 - getTime method; 695
 - setTime method; 695
- date(s)**
 - See also*: internationalization; localization; time
 - Formatter class use; 706–708
 - localization of; 686
 - parsing; 703
 - DateFormat class use; 705
 - representation of; 695–703
- Date1 example class**; 37
- Date2 example class**; 37
- DateFormat class**; 696, 703
 - calendar field; 705
 - Date class superseded by; 695
 - format method; 704
 - Format subclass; 710
 - getAvailableLocales method; 704
 - getDateInstance method; 703
 - getDateTimeInstance method; 703
 - getInstance method; 704
 - getNumberFormat method; 704
 - getTimeInstance method; 703
 - isLenient method; 705
 - numberFormat field; 705
 - parse method; 705
 - parseObject method; 705
 - setLenient method; 705
 - setNumberFormat method; 704
- Davis, Miles**
 - quotation; 385
- daylight savings time**; 700
- dead objects**
 - See also*: garbage collection
 - phantom reachability; 456
 - term definition; 448
- deadlock**; 362–365
 - See also*: concurrency; synchronized/ synchronization
 - avoidance; 364
 - precautions against; 353
 - shutdown hook issues; 673
 - term definition; 362
- deadly embrace**; 362–365
 - See also*: concurrency; synchronized/ synchronization
 - term definition; 362
- death**
 - thread, thread local variables impact; 383
- debugging**; 676–677
 - arrays; 19
 - name problems, field names vs. parameter names; 17
 - Proxy class use; 433
 - stack traces; 382
 - threads; 384–385
- DebugProxy example class**; 433–434
- decimal**
 - See also*: numbers/numeric
 - format specifier; 23
 - floating-point conversion use; 627
 - integer conversion use; 627
 - integer literals; 167
 - numbers, arbitrary precision, BigDecimal class; 724
- DECIMAL_DIGIT_NUMBER constant**
 - Character class; 195
- DecimalFormat class**; 710

Deck example class; 18**declaration(s)**

See also: defining
 annotations; 388
 arrays; 173
 classes
 class keyword use; 42
 fields; 14
 methods, static keyword use; 17
 term definition; 12
 constants; 7
 enum; 154–159
 enum; 152–154
 fields; 44, 170
 generic
 classes; 247
 methods; 261
 types; 250–255
 implementation vs., interface's distinction
 from classes; 27
 interfaces; 120
 named constants; 29
 static constants; 29
 local variable; 170
 methods; 57
 native; 74
 term definition; 3
 void keyword use; 15
 statements, term definition; 230
 static initialization blocks; 56
 variables; 170
 local; 4

decode method
 integer wrapper classes; 189
 decoding strings into numbers with; 316

decoding
 strings, toString method use; 317

decrement (--) operator; 205
 term definition; 11

deep cloning
See also: cloning
 shallow cloning vs.; 106
 term definition; 106

deepEquals method
 Arrays class; 608

deepHashCode method
 Arrays class; 608

deepToString
 Arrays class; 608

default
 character encoding; 320
 constructors, term definition; 53
 deserialization; 552
 field values; 45
 initial values, arrays; 175

interface implementation nested class use for;
 149

keyword, reserved keyword (table); 165

label

 switch statement use; 232

 term definition; 232

object references; 13

security manager, (footnote); 677

serialization; 551

setting, StreamTokenizer class; 535

value, writing, synchronization actions; 373

defaultCharset method

Charset class; 321

defaultReadObject method

ObjectInputStream class; 558

defaultWriteObject method

ObjectOutputStream class; 558

defensive copies

term definition; 127

defineClass method

ClassLoader class; 439–440

definePackage method

ClassLoader class; 479

defining

immutable properties; 46

security domains, thread groups use for; 375–
 379

types, interface use; 27

definitely assigned variable

term definition, (footnote); 171

Deflater class; 737**DeflaterOutputStream class; 737****Delayed interface**

getDelay method, DelayQueue use; 606

DelayQueue class; 606**delete method**

File class; 545

StringBuilder class; 333

deleteCharAt method

StringBuilder class; 333

deleteObserver method

Observable class; 636

deleteObservers method

Observable class; 637

deleteOnExit method

File class; 547

deleting

assertions; 302

characters, a string buffer; 333

files, File.delete; 545

objects, garbage collector role in; 15, 49

temporary files, File.deleteOnExit; 547

delimiter method

Scanner class; 644

delimiters

See also: tokens
scanner identification of tokens by; 643
strings, extracting; 313

demand

import on, term definition; 469
static import on, syntax; 72

dependencies

constructor order; 81–82
thread, thread completion wait; 367

deprecated

identifiers, doc comment tag for; 486
methods, `Thread.stop`; 381

Deprecated class; 391, 396**descriptors (file)**

`FileDescriptor` class; 541
retrieving, `RandomAccessFile.getFD`; 543
term definition; 541

deserialization

See also: serialization
default; 552
order; 552–553
term definition; 549
versioning issues; 554

design issues and strategies

See also: coding style; contract;
documentation; performance
abstractions, raising the level with exception
chaining; 292

access control

accessor method advantages; 66
extensible framework design; 113
per-class, not per-object; 48, 68

annotations, working with; 395

anonymous inner classes; 146

applet; 721

assertions

evaluation side effect dangers; 296
turning on and off, precautions; 300

blocking operations, interrupt use; 366

catch clauses; 287–288

classes

access restriction; 16
designing a class to be extended; 108–114
internal package needs; 15
nesting; 133
simple data containers; 15
that implement interfaces; 127
throws clause considerations; 285

cleanup and release of resources, `finally`
use for; 289

cloning; 101

methods; 101, 103

concurrency, multiple file stream/same file
issues; 541

constructors

goals for, in multiple constructor design;
80

method precautions; 83

specialized constructor considerations; 53

contract, impact on class extension design; 75

deadlock avoidance; 364

documentation

doc comment conventions; 482

package and overview; 496

quality and accuracy issues; 497

enum use; 160

exceptions

cleanup handling; 366

when to create; 281–282

when to use; 294–296

expansion vs. specialization, in class

extension; 94

extensible classes, interfaces as critical

component; 131

field access, hiding vs. `private`; 88

final

classes and methods; 96

fields, determining appropriateness of; 47

finalization issues; 450

reference queues use; 464

`super.finalize` method use; 451

flow of control, nested `if-else` vs.

conditional operators; 20

framework, extensibility issues; 109–114

generic types

class extension and; 276–277

methods and constructors; 261–262

one class, multiple parameterized

invocations; 250

overloading; 276

graphical user interfaces; 718

layout issues; 718

hashtables, `HashMap` class; 591

hiding, reasons why it is bad style; 180

I/O

closing conversion stream issues; 513

flushing data to disk strategies; 541

stream closing importance; 502, 507

implementation, fairness, design issues; 362

inheritance hierarchy and packages,

(footnote); 474

initialization, when to use initialization

blocks; 55

interfaces; 126–130

abstract classes vs.; 131

multiple same-name method conflicts;

125

nesting; 133

throws clause considerations; 285

type, references; 120

- internationalization; 685–714
- interrupt handling; 367
- IsA vs. HasA relationships; 107–108
- localization; 685–714
- multithreading
 - interrupt handling; 367
 - Runnable flexibility for; 345
- native method impact on portability and safety; 74
- nested
 - classes in interfaces; 148
 - generic types; 255
 - inner classes; 139
 - types, complex constructor issues; 148
- notification, notify vs. notifyAll; 355
- object-oriented
 - object resurrection issues; 452
 - synchronization issues as illustration of the power of; 352
- overloading; 70–71
- overriding sequence parameters; 85
- package contents; 475–476
- patterns, JavaBeans use; 721
- programming without a goto, alternative constructs; 246
- raw types, purpose and limitations of; 747
- reflection
 - appropriate Method class use; 422
 - drawbacks vs. direct method invocations; 399
- regular expressions, performance vs. clarity tradeoffs; 330
- resource
 - bundles; 690
 - management, finally use for; 289
- security issues for
 - extensible class design; 109
 - extensible framework design; 114
- serialization
 - requirements and issues; 552–553
 - security, Externalizable interface issues; 562
- shutdown; 674
 - hook issues; 673
- static
 - data protection, Class object lock use; 351
 - imports; 73
- synchronization; 352–354
 - memory model; 370–375
 - per-thread behavior of locks impact on synchronized methods; 346
 - synchronized statement advantages vs. synchronized methods; 349
 - volatile; 370–375
- this reference use; 68–69
- threads
 - cancellation advantages; 365
 - constructor start risks; 344
 - scheduling, ordering of locks and notifications; 359
 - scheduling, preemption and priority issues; 358, 359
 - termination, reasons why Thread.stop is deprecated; 381
 - thread local variable dangers; 384
 - while vs. if in wait loops; 355
- time-outs completion strategies
 - join; 368
 - wait; 357
- types
 - fundamental unit of object-oriented design; 117
 - named constants use; 121
 - Unicode subset definitions; 196
 - unmodifiable wrapper use; 601
 - variable argument ambiguities; 61
- version control
 - doc comment tag use; 486
 - exception signaling vs. version numbers; 558
- wait completion strategies, spurious wakeups; 358
- design patterns**
 - See: pattern(s)
- destination**
 - term definition; 499
- destroy method**
 - Applet class; 720
 - Process class; 668
 - System class; 370
 - exit method compared with; 370
 - ThreadGroup class; 378
- Device example class**; 146
- diamond inheritance**; 115
- Dictionary class**; 619
 - Hashtable class relationship; 619
- digit method**
 - Character class; 193
- digit(s)**
 - See also: numbers/numeric
 - character
 - retrieving value for,
 - Character.forDigit; 193
 - testing for, Character.isLetterDigit; 194
 - Unicode
 - meaning; 164
 - testing for, Character.isDigit; 194
 - wildcard for matching; 322

digital

See also: security
certificates, code source inclusion of; 681
signatures, `java.security` package use;
732

Dijkstra, Edsger Wybe

further reading; 756

dimensions

array, when specified; 173

DirContext interface; 739

`getAttributes` method; 739
`modifyAttributes` method; 739

directory method

`ProcessBuilder` class; 671

directory(s)

See also: files(s); I/O

contents

listing, `File.list`; 546
listing, `File.listFiles`; 546

creating

`File.mkdir`; 546
`File.mkdirs`; 546

doc-files, documentation use; 497

name, component of path name; 543

root, retrieving, `File.list`; 546

services, JNDI, `javax.naming` package; 738

testing for, `File.isDirectory`; 545

working

`ProcessBuilder` encapsulation of; 670
specifying; 670

DirFilter example class; 548**disjoint method**

`Collections` class; 598

disk

See also: I/O

flushing data to, strategies for; 541

division

See also: arithmetic; numbers/numeric;
operator(s)

binary division operator; 201

floating-point; 203

integer; 202

operator, term definition; 6

do keyword

reserved keyword (table); 165

do-while statement; 235

See also: flow of control

loop flow of control mechanism; 9

syntax; 236

doc comments

delimiter; 482

syntax and use; 482

tags; 483–488

term definition; 6, 481

doc-files directory

documentation use; 497

documentation

See also: annotations; coding style; design

issues and strategies

annotations use; 387–396

assertion use; 297

checked exceptions, importance of; 280

comment tags

serialization; 562–563

(table); 754

comments; 481–498

methods, inheritance of; 489–491

term definition; 6

contract semantics repository; 41

errors, reason for creating new exception

types; 280

external, conventions for; 496–497

packages, `Package` class use for; 477

quality and accuracy issues; 497

reference, doc comments use for; 481

synchronization, importance of; 353

throws clause use; 284, 285

Documented class; 393, 396**dollar sign (\$)**

format specifier use; 626

multiline mode, dot-all mode pattern

matching; 324

nested type internal name use; 149

domain

protection

class loader use; 440

`ProtectionDomain` class representation

of; 681

term definition; 677, 681

security, defining, thread groups use for;

375–379

doPrivileged method

`AccessController` class; 679, 682

dot (.)

See also: operator(s)

dot-all mode pattern matching; 324

operator

member access use; 3, 16, 58, 223, 411

member access use, nested type reference
use; 133

method invocation use; 16

package component separation; 36

package component separation, qualified
names; 123

static member reference use; 8

`this` reference use; 17

precision format specification use; 24

wildcard, match single characters; 322

dot (.) asterisk (*)

wildcard, match zero or more characters; 322

dot .class class literals; 169

DOTALL flag

Pattern class; 324

dotw method

GregorianCalendar class; 699

Double class

converting, strings to/from (table); 316

extension of Number class; 188

floatToIntBits method; 192

floatToRawIntBits method; 192

infinity constants in; 203

intBitsToFloat method; 192

isNaN method; 206

wrapper class, double primitive type; 166

wrapper type hierarchy, (figure); 183

double data type

See also: data types; floating-point numbers

64-bit IEEE 754–1985 floating-point

number; 38

arrays, name notation; 412

converting to String, before adding to a string buffer; 332

converting to/from strings (table); 316

definition; 4

doubleValue method, Number class; 188

field values, default; 45

floating-point literal; 168

Print stream handling; 525

reading, DataInput.readDouble; 537

reserved keyword; 165

value of; 166

writing, DataInput.writeDouble; 537

double quote (")

string literals; 3, 168

doubleValue method

Number class; 188

dowhenCondition example method

standard pattern for use of wait and

notification; 354

downcasting

See also: casting

term definition; 92

Dr. Who

quotations; 41, 305

drag-and-drop

java.awt.dnd package; 720

drainTo method

BlockingQueue interface; 605

Duff, Tom

quotation; 749

dumpStack method

Thread class, debugging use; 385

DuplicateFormatFlagsException class; 630**dynamic**

array creation; 430–432

compilation, shared values; 370

E**E (element type)**

type variable naming convention; 248

E constant; 658**e, E format specifier**

floating-point conversion use; 627

E/e in floating point literals

exponent indicator in floating-point literals; 168

EBCDIC

See also: character(s)

character representation; 162

efficiency

pattern matching; 329–330

Eiffel language

bibliographic reference; 758

Einstein, Albert

quotation; 74

element method

Queue interface; 585

element(s)

annotating; 392–393

annotation

declaration rules; 389

term definitions; 35, 388

array, accessing; 173

elementAt method

Vector class; 618

elementCount field

Vector class; 618

elementData field

Vector class; 618

elements method

Hashtable class; 619

Vector class; 618

ElementType class; 394**ellipse (. . .)**

variable number of arguments indicated by; 58, 60

else

See also: flow of control

clause, binding of, an if statement; 231

keyword (table); 165

statement(s), introduction; 11

Emerson, Ralph Waldo

quotation; 445

empty

array, term definition; 19, 174

statement, semicolon use as; 34, 229

EMPTY_LIST field

Collections class; 601

EMPTY_MAP field

Collections class; 601

empty method

Stack class; 619

- EMPTY_SET field**
 - Collections class; 601
- emptyList method**
 - Collections class; 600
- emptyMap method**
 - Collections class; 600
- emptySet method**
 - Collections class; 600
- EmptyStackException class; 619**
- enableEvents method**
 - ActionEvent class; 719
- encapsulation**
 - access control modifiers impact on; 48
 - data, term definition; 15
- enclosing**
 - instances, term definition; 136
 - objects
 - inner class relationship to; 136
 - inner classes access; 137
 - term definition; 136
 - scopes, algorithm for the semantics of a name; 180
 - type, static nested type relationship to; 133
- ENCLOSING_MARK constant**
 - Character class; 195
- encodings**
 - See also:* ASCII; Unicode
 - character
 - converting to Unicode characters; 512
 - converting to/from strings; 319
 - encoding standards (table); 754
 - required support; 320
 - string, toString method use; 317
- end**
 - string, testing, String.endsWith; 311
- end method**
 - MatchResult interface; 326
- END_PUNCTUATION constant**
 - Character class; 195
- end-of-file (EOF)**
 - DataInput interface handling; 538
- end-of-line (EOL)**
 - StreamTokenizer handling; 536
- endsWith method**
 - String class; 311
- EnhancedSerialPort example class; 147**
- enqueue method**
 - Reference class; 455
 - forcing a reference into its queue with; 460
- ensureCapacity method**
 - ArrayList class; 582
 - StringBuilder class; 335
- entries method**
 - AbstractMap class; 615
- Entry interface**
 - getKey method; 589
 - getValue method; 589
 - Map interface extended by; 589
 - setValue method; 589
- entrySet method**
 - Map interface; 589
- Enum class; 153, 159–160**
 - getDeclaringClass method; 160
 - ordinal method; 159
 - valueOf method; 160
- enum(s); 151**
 - See also:* constants
 - annotation of; 392–393
 - behavior; 156–159
 - case labels required to be constants or; 235
 - classes, nested; 135
 - collection
 - classes; 160
 - for working with; 594
 - constant
 - declarations; 154–159
 - term definition, (footnote); 152
 - constructors; 155–156
 - restrictions on; 156
 - declarations; 152–154
 - ElementType class; 394
 - enum class declaration use; 152
 - initialization, arrays compared with; 153
 - modifiers; 154
 - representation by Class objects; 399
 - reserved keyword (table); 165
 - RetentionPolicy class; 395
 - retrieving, Class.getEnumConstants method; 407
 - RoundingMode class; 722
 - State, Thread class; 384
 - switch statement use; 235
 - term definition; 8, 151
 - testing if a field is, Field.isEnumConstant; 418
 - type, term definition, (footnote); 152
- EnumConstantNoPresentException class**
 - annotation reflection use; 415
- enumerate method**
 - ThreadGroup class; 378, 379
- enumeration**
 - ordered, token sequence viewed as; 651
 - types; 151–160
 - nesting within an interface; 121
 - term definition; 8, 151
- Enumeration interface; 617**
 - BufferedReader class use; 528
 - ClassLoader.getResources return; 442
 - hasMoreElements method; 617

- nextElement method; 617
- StringTokenizer implementation of; 651
- enumeration method**
 - Collections class, iteration use; 529
- EnumMap class**; 160, 596
- EnumSet class**; **160, 594–596**
 - allOf method; 595
 - complementOf method; 595
 - copyOf method; 595
 - noneOf method; 595
 - of method; 595
 - range method; 596
- environment method**
 - ProcessBuilder class; 671
- environment(s)**
 - process; 669–670
 - ProcessBuilder encapsulation of; 670
 - system; 663
 - variables, term definition; 669
- EOF (end-of-file)**
 - DataInput interface handling; 538
- EOFException class**; 558, 563
 - DataInput interface use; 538
- EOL (end-of-line)**
 - StreamTokenizer handling; 536
- eofIsSignificant method**
 - StreamTokenizer class; 536
- epoch**
 - See also:* date(s); internationalization; localization; time
 - term definition; 36, 665, 695
- equality**
 - See also:* hash code
 - equals method, wrapper classes; 187
 - object
 - IdentityHashMap notion of; 592
 - serialization design issue; 557
 - testing; 99–100
 - operators; 206
 - precedence; 222
 - reference
 - hashtable impact; 101
 - testing; 101
 - reference vs. value; 101
 - strings, testing; 311–313
 - testing for; 5
 - values, testing; 99–100
- equals (=)**
 - See also:* assignment; operator(s)
 - assignment operator; 212
 - binary operator combination with; 11
 - equality operator, term definition; 5
 - operator, term definition; 4
- equals equals (==)**
 - equality operator, term definition; 5
 - operator; 206
 - object equality testing; 99
 - reference equality testing; 101
 - string
 - comparison issues; 311
 - object comparison results; 22
- equals method**
 - Arrays class; 608
 - BitSet class; 634
 - checking for equivalence with
 - ParameterizedType objects; 427
 - TypeVariable objects; 427
 - Collection interface use; 576
 - Comparable.compareTo relationship to; 574
 - Enum class use; 159
 - File class; 544
 - Map interface; 587
 - natural equivalence defined by; 574
 - Object class; 99
 - equivalence tested by; 207
 - String class; 22, 309
 - UUID class; 657
 - wrapper classes; 187
- equalsIgnoreCase method**
 - String class; 309
- equivalence**
 - Arrays class, deepEquals method; 608
 - canonical, pattern matching; 324
 - natural, term definition; 574
 - reference identity compared with; 206
 - term definition; 100, 101
- era**
 - See also:* date(s); internationalization; localization; time
 - TimeZone.getOffset specification; 701
- erasure**
 - compatibility design issues; 744–748
 - raw types and; 267–272
 - term definition; 267
- err field**
 - System class; 662
- Error class**
 - Exception class vs., assertion use; 297
 - most serious runtime exceptions extensions of; 284
 - in type hierarchy; 280
 - unchecked exceptions; 34, 281
- error(s)**
 - See also:* errors (Java-defined) exceptions; failures
 - exceptions handling of; 279–304
 - OutOfMemoryError, during object creation;

- error(s)** (*cont.*)
 - overview; 32–35
 - signaling, exceptions use for; 279
 - standard I/O stream; 662
 - ProcessBuilder redirection control; 670
- errors (Java-defined)**
 - See error classes:* AssertionError; Error; ExceptionInInitializerError; LinkageError; OutOfMemoryError; UnsatisfiedLinkError; VirtualMachineError
- escape**
 - See also:* backslash; regular expressions; special symbols; wildcards
 - characters, special symbol for; 322
 - sequences; 167
 - encoding Unicode characters with; 162
- evaluation**
 - assertion, turning on and off; 300–303
 - exception impact on; 283
 - expressions, order of; 214
 - floating-point, strict vs. non-strict; 203
- event(s)**
 - See also:* exception(s)
 - action, term definition; 719
 - AWT as event-based system; 718
 - based graphical user interface, thread use; 370
 - frameworks, nesting use in; 133
 - handling
 - Observer interface and Observable class use; 635–639
 - thread interruption use; 366
 - java.awt.event package; 720
 - model
 - AWT; 718
 - JavaBeans; 721
 - notification, java.naming.event package; 739
 - queue, term definition; 718
 - scheduling, with Timer and TimerTask; 653
 - thread, term definition; 718
- EventListener interface**
 - marker interface; 130
 - processEvent method; 719
- examining**
 - class members; 408
 - thread group contents; 378
- examples**
 - See:* classes (examples); errors (examples); exceptions (examples); interfaces (examples)
- Exception class; 32**
 - Error class vs., assertion use; 297
 - exceptions extension of; 280
 - in type hierarchy; 280
- exception(s); 279–304**
 - See also:* error(s); exceptions (Java-defined)
 - @throws tag; 485
 - inheritance; 490–491
 - asynchronous; 283
 - casting Throwable to actual exception types; 293
 - catching; 286–291
 - term definition; 32, 279
 - chaining of; 291–294
 - checked
 - constructor use of; 54
 - restrictions for initialization blocks; 54
 - static initialization blocks prohibited from using; 55
 - term definition; 32, 34, 280
 - conflict potential in multiple interface inheritance; 125
 - constructor handling of; 82
 - conventions related to; 570
 - doc comment tags for; 485
 - error(s)
 - event(s)
 - failure(s)
 - format; 630–631
 - I/O, IOException class; 563
 - not included in signature; 70
 - overview; 32–35
 - runtime, unchecked exceptions; 284
 - stack traces; 294
 - synchronous, term definition; 380
 - term definition; 32
 - ThreadDeath class, Thread.stop method use; 381
 - threads and; 379–381
 - throwing
 - method declaration, throws clause use; 284–286
 - term definition; 32, 279
 - thrown by catch or finally clauses, handling of; 287
 - transfer of control; 283
 - type hierarchy, (figure); 280
 - types, creating; 280
 - uncaught
 - method execution terminator; 62
 - term definition; 380
 - unchecked, term definition; 34, 280
 - when to use; 294–296
- ExceptionInInitializerError class**
 - Class.forName use; 413
 - Class.newInstance use; 424
 - field reflection use; 419
 - Method class use; 421

exceptions (Java-defined)*See exception classes:*

ArithmeticException;
 ArrayStoreException;
 ClassCastException;
 EmptyStackException;
 EOFException; Exception;
 InterruptedException;
 IOException;
 NoSuchElementException;
 NotActiveException;
 NullPointerException;
 ParseFormatException;
 PrivilegedExceptionAction;
 RemoteException;
 RuntimeException;
 SecurityException;
 StreamException;
 SyncFailedException;
 UncaughtExceptionHandler

*See: BadDataSetException example***Exchanger class; 734****exclamation (!)**

logical negation operator; 207

exclamation equals (!=)

inequality operator; 206

term definition; 5

not equal to inequality operator; 206

object inequality testing; 99

exclusion (mutual)

See also: synchronized/synchronization
 guaranteed to synchronized threads; 347
 term definition; 346

exclusive OR (^) operator; 20, 207

bitwise operations; 208

precedence; 222

exclusive OR (XOR)

logical, equality operations use; 206

exec method, Runtime class; 666

environment variable passing version; 669

portability issues; 672

working directory specification version; 670

execution

application, termination of; 369

methods; 62

order, thread requirements for, memory

model impact; 375

thread, terminating; 365–369

Executor interface; 734**ExecutorService interface; 734****exists method**

File class; 545

exit method

Runtime class; 672

application execution termination with;
370

conditions that require use of; 674

shutdown initiated by; 674

System class; 666

application execution termination with;
370**exit/exiting**

application, garbage collection issues; 452

blocks, break use for; 241

constructors, return use for; 245

loops, break use for; 242

method execution, return use for; 245

static initializers, return use for; 245

exitValue method

Process class; 668

exp method

Math class and StrictMath class; 658

expansion

specialization vs., class extension; 94

explicit

constructor invocation; 52

type casts; 91–92, 219

exponent; 658**expression(s); 214–216***See also:* mathematics

assignment; 212

statements; 230

boolean

primitive type; 5

term definition; 5

conditional operator, type of, rules for
determining; 211

conversions that apply to; 217–218

evaluation, order of; 214

initialization, initial value setting with; 4

instance creation expression, term definition;
214

operators, and tokens; 161–182

postfix, statements; 230

prefix, statements; 230

statements, term definition; 229

switch, type of; 235

term definition; 214

type compatibility,

context impact on; 91

requirements for variable assignment; 90–
93

type of; 215

variable assignment in; 11

ExtendedInner example class; 139**ExtendedOuter example class; 139**

extending/extensibility

See also: inheritance
 arrays not permitted; 178
 behavior, subclass use; 25
 classes; 75–116
 constructors in; 80–83
 design strategies; 108–114
 generic types and; 276–277
 interfaces, critical design component; 131
 overview; 24–27
 data, subclass use; 25
 framework design issues; 109–114
 inner classes; 139–140
 unrelated inner classes; 140
 interfaces; 122
 static nested classes; 136
 string buffers; 332
 term definition; 24
 types, enums not permitted to; 153

extends keyword

generic type use; 253
 interface extension with; 29, 122
 reserved keyword (table); 165

ExtendShow example class; 86**extensions**

standard extensions mechanism, term
 definition; 681

external documentation

conventions for; 496–497

Externalizable interface; 561–562

marker interface; 130
 readExternal method; 562
 writeExternal method; 561

extracting

See also: accessing
 char arrays, from a `StringBuilder` object;
 333
 delimited strings; 313
 strings, from `StringBuilder` object; 333
 substrings, from `StringBuilder` object; 333

Eye example class; 638**F****f, F format specifier**

floating-point conversion use; 627

F/f suffix

single-precision floating-point literal
 indicator; 168

factoring

synchronized statement code into
 synchronized methods; 352

factory methods

`Collections` class synchronization methods
 use; 603
 creating `EnumSet` objects with; 595

creating UUIDs; 656
 `UUID.nameUUIDFromBytes`; 656
 `UUID.randomUUID`; 656
 inner class use; 147

fail-fast iterators

See also: iteration/iterator
 term definition; 574

failure(s)

See also: error(s); event(s); exception(s)
 network, remote interface handling; 728
 partial, term definition; 728

fairness

`Thread.yield` impact on; 362

false boolean literal; 5, 167

identifiers prohibited from using; 165

FALSE constant

`Boolean` class; 188

FDLIBM “C” library

`StrictMath` class algorithms defined in; 659

Fibonacci example class; 3**Fibonacci2 example class; 7****Field class; 418–420**

get method; 418
 getField method; 409
 getGenericType method; 418
 getType method; 418
 isEnumConstant method; 418
 overview and place in introspection
 hierarchy; 399
 set method; 418

FIELD constant

`ElementType` class; 394

field(s)

See also: class(es); constants (Java-defined);
 fields (Java-defined) interface(s);
 method(s)
 accessing, object reference type as
 determinant of; 86
 annotation; 392–393
 compared to; 390
 class; 14
 members; 12, 42
`Class.getDeclaredFields`; 408
`Class.getFields`; 408
 cloning, a protected clone implementation;
 102
 constant, values, documentation of; 487
 declarations; 44, 170
 as constant; 7
 final; 46
 reflection manipulation of; 420
 handling during clone override; 106
 hidden; 86
 accessing with `this`; 69
 design reasons; 88
 enforcing read-only access by; 66

- initialization; 44
 - constructor order relative to; 82
 - with initialization blocks; 54–55
 - extended classes; 80
- MAX_VALUE, Integer class, BlockingQueue
 - interface use; 605
- modifiers; 44
- namespace; 179
- nested type inheritance comparison with; 146
- non-static; 46
- scope; 180
- serialized; 559–561
- static; 14, 45
 - initialization with static initialization
 - block; 55
 - serialization impact; 551
 - term definition; 14
- System class; 662
- term definition; 1, 44
- transient
 - serialization impact on; 551
 - serialized fields and; 561
- wildcard use; 260
- wrappers, common to all; 184–187
- FieldModifiers example class; 594**
- FieldPosition class**
 - DateFormat class use; 705
 - getBeginIndex method; 705
 - getEndIndex method; 705
 - setBeginIndex method; 705
 - setEndIndex method; 705
- fields (Java-defined)**
 - See FieldPosition class
 - See constants ANNOTATION_TYPE; BLOCKED;
 - CASE_INSENSITIVE_ORDER; CLASS;
 - COMBINING_SPACING_MARK;
 - CONNECTOR_PUNCTUATION;
 - CONSTRUCTOR; CONTROL;
 - CURRENCY_SYMBOL;
 - DASH_PUNCTUATION;
 - DECIMAL_DIGIT_NUMBER; E;
 - EMPTY_LIST; EMPTY_MAP; EMPTY_SET;
 - ENCLOSING_MARK; END_PUNCTUATION;
 - FALSE; FIELD;
 - FINAL_QUOTE_PUNCTUATION;
 - INITIAL_QUOTE_PUNCTUATION;
 - LETTER_NUMBER; LINE_SEPARATOR;
 - LOCAL_VARIABLE; LOWERCASE_LETTER;
 - MATH_SYMBOL; MAX_PRIORITY;
 - MAX_RADIX; MAX_VALUE; METHOD;
 - MIN_PRIORITY; MIN_RADIX;
 - MIN_VALUE; MODIFIER_LETTER;
 - MODIFIER_SYMBOL;
 - NEGATIVE_INFINITY; NEW;
 - NON_SPACING_MARK; NORM_PRIORITY;
 - OTHER_LETTER; OTHER_NUMBER;
 - OTHER_PUNCTUATION; OTHER_SYMBOL;
 - PACKAGE; PARAGRAPH_SEPARATOR;
 - PARAMETER; PI; POSITIVE_INFINITY;
 - PRIVATE_USE; RUNNABLE; RUNTIME;
 - SIZE; SOURCE; SPACE_SEPARATOR;
 - START_PUNCTUATION; SURROGATE;
 - TERMINATED; TIMED_WAITING;
 - TITLECASE_LETTER; TRUE; TYPE; TYPE;
 - UNASSIGNED; UPPERCASE;
 - UPPERCASE_LETTER; WAITING
 - See fields: calendar; capacityIncrement;
 - elementCount; elementData; err;
 - length; line.separator; lock;
 - numberFormat; nval; out;
 - pathSeparator; pathSeparatorChar;
 - separator; separatorChar; sval
- File class; 543–548**
 - canRead method; 545
 - canWrite method; 545
 - compareTo method, path compared; 544
 - createNewFile method; 545
 - createTempFile method; 547
 - delete method; 545
 - deleteOnExit method; 547
 - equals method, path compared; 544
 - exists method; 545
 - Formatter use; 631
 - getAbsoluteFile method; 544
 - getAbsolutePath method; 544
 - getCanonicalFile method; 544
 - getCanonicalPath method; 544
 - getName method; 544
 - getParent method; 544
 - getParentFile method; 544
 - getPath method; 544
 - isAbsolute method; 545
 - isDirectory method; 545
 - isFile method; 545
 - isHidden method; 545
 - lastModified method; 545
 - length method; 545
 - list method; 546
 - listFiles method; 546
 - listRoots method; 546
 - mkdir method; 546
 - mkdirs method; 546
 - pathSeparator field, search string use; 547
 - pathSeparatorChar field, search string use; 547
 - renameTo method; 545
 - separatorChar field, path component
 - separator stored in; 543
 - setLastModified method; 546
 - setReadOnly method; 546
 - toURI method; 544
 - toURL method; 544, 726

file(s)

See also: directory(s); I/O

byte

reading, `FileInputStream` class; 541
writing, `FileOutputStream` class; 541

channel, retrieving,

`RandomAccessFile.getChannel`; 543

character

reading, `FileReader` class; 541
writing, `FileWriter` class; 541

creating

`File.createNewFile`; 545
`FileOutputStream` class; 541
`RandomAccessFile` class; 541

deleting, `File.delete`; 545

descriptor, retrieving,

`RandomAccessFile.getFD`; 543

descriptors

`FileDescriptor` class; 541
term definition; 541

directory, testing for, `File.isDirectory`;
545existence, testing for, `File.exists`; 545

filtering

`FileFilter` interface; 548
`FilenameFilter` interface; 548

format

GZIP, `java.util.zip` package; 736
JAR, `java.util.jar` package; 735–736
ZIP, JAR file format based on; 735
ZIP, `java.util.zip` package; 736

last modified time, retrieving,

`File.lastModified`; 545

length

retrieving, `File.length`; 545
retrieving, `RandomAccessFile.length`;
542
setting,

`RandomAccessFile.setLength`; 542

locking, `FileLock` use; 566mapping, into memory, `MappedByteBuffer`;
565names, manipulation of, `File` class; 543not a directory, testing, `File.isFile`; 545path vs., `File` object manipulations; 543

pathname retrieving components

`File.getAbsolutePath`; 543
`File.getCanonicalPath`; 544
`File.getName`; 543
`File.getParent`; 544
`File.getPath`; 543

pointer

retrieving,
`RandomAccessFile.getFilePointer`;
542
term definition; 541

public classes relationship to; 43

random access; 541–543

reading; 541
term definition; 541
writing; 541

readable, testing for, `File.canRead`; 545renaming, `File.renameTo`; 545

state

changing, `File.setLastModified`; 546
changing, `File.setReadOnly`; 546

streams, `File` streams; 541

temporary

creating, `File.createTempFile`; 547
deleting, `File.deleteOnExit`; 547

working with; 540–549

writeable, testing for, `File.canWrite`; 545writing bytes to, with buffered output
streams; 519**File streams; 540–549, 541**

Buffered streams use with; 519

byte, `getFD` method; 541

FileChannel class

`getChannel` relationship to; 540

locking, `FileLock` use; 566

FileDescriptor class; 541–541

`RandomAccessFile.getFD` method access
of; 543

valid method; 541

FileFilter interface; 548

accept method; 549

FileInputStream class; 541

`getChannel` method; 540, 565

FileLock class

locking files; 566

FilenameFilter interface; 548

accept method; 548

`File.list` and `File.listFiles` use; 546

FileNotFoundException class; 541, 564**FileOutputStream class; 541**

BufferedOutputStream use with; 519

flush method; 541

`getChannel` method; 540

FilePermission class; 679**FileReader class; 541**

InputStreamReader relationship to; 513

FileWriter class; 541

flush method; 541

OutputStreamWriter relationship to; 513

fill method

Arrays class; 608

Collections class; 598

fillInStackTrace method

Throwable class; 294

Filter class

Data class extensions of; 539

Filter streams; 516–518

- Print streams use as; 525
- stream that defines behavior; 514
- synchronization strategy; 515

filtering

- files
 - FileFilter interface; 548
 - FilenameFilter interface; 548
- ListIterator use; 611
- view use of; 578

FilterInputStream class; 516**FilterOutputStream class;** 516**FilterReader class;** 516**FilterWriter class;** 516**final keyword**

- See also:* blank finals
 - abstract vs.; 43
 - array declaration use; 174
 - classes
 - array behavior like; 178
 - design considerations; 96
 - constants
 - declaration use; 7
 - interface declaration of; 29
 - enum not permitted to be; 154, 158
 - fields; 46
 - cloning issues and pitfalls; 106
 - custom serialization vs. default serialization; 556
 - reflection manipulation of; 420
 - local variable use; 171
 - methods
 - design considerations; 96
 - modifier; 57
 - parameters; 65
 - security and optimization advantages; 111
 - overriding considerations; 85
 - read-only access use; 66
 - reserved keyword (table); 165
 - term definition; 43
 - variables; 171–173
 - volatile keyword vs.; 44
- FINAL_QUOTE_PUNCTUATION constant**
Character class; 195
- finalization;** 449–452
See also: cancellation; cleanup; initialization guaranteed, shutdown hooks required for; 464
reachability and; 456, 464
resurrecting objects during; 452
serialization vs.; 554
- finalize method**
not a substitute for close; 502
Object class; 100, 450
enums not permitted to override; 154

finally clause; 286–291, 288–291

- cleanup code in; 34
- impact of Runtime.exit on; 672
- reserved keyword (table); 165
- try-catch-finally sequence; 33
- when executing; 286

FindChar example class; 527**findClass method**

- ClassLoader class; 436, 439

findInLine method

- Scanner class; 644–645

findLoadedClass method

- ClassLoader class; 439

findResource method

- ClassLoader class; 443

findResources method

- ClassLoader class; 443

findWithinHorizon method

- Scanner class; 646

finite arithmetic; 202

- See also:* arithmetic; Math class; numbers/numeric; StrictMath class;

Firesign Theatre

- quotation; 150

first method

- SortedSet interface; 577

first-in-first-out

- Queue use; 585

firstElement method

- Vector class; 618

firstKey method

- SortedMap interface; 590

fixed-arity methods

- term definition, (footnote); 60

fixed-delay task scheduling

- term definition; 655

fixed-rate task scheduling

- term definition; 655

flags

- CANON_EQ, Pattern class; 324
- CASE_INSENSITIVE, Pattern class; 324
- COMMENTS, Pattern class; 325
- DOTALL, Pattern class; 324
- grouping in an EnumSet; 595
- LITERAL, Pattern class; 325
- MUTLLINE, Pattern class; 324
- pattern matching; 324
- UNICODE_CASE, Pattern class; 324
- UNIX_LINES, Pattern class; 324

flags method

- Pattern class; 324

flexibility

- See also:* design issues and strategies
- extensible class design issue; 109

- final disadvantages; 96
- interface advantage, multithreaded design; 353
- flip method**
 - BitSet class; 633
- Float class**
 - converting, strings to/from (table); 316
 - extension of Number class; 188
 - floatToIntBits method; 192
 - floatToRawIntBits method; 192
 - infinity constants in; 203
 - intBitsToFloat method; 192
 - isNaN method; 206
 - parseFloat method; 316
 - wrapper class, float primitive type; 166
 - wrapper type hierarchy, (figure); 183
- float data type**
 - See also:* data types; floating-point numbers
 - arrays, name notation; 412
 - converting to String, before adding to a string buffer; 332
 - converting to/from strings (table); 316
 - definition; 4
 - field values, default; 45
 - floatValue method, Number class; 188
 - Print stream handling; 525
 - reading, DataInput.readFloat; 537
 - reserved keyword; 165
 - value of; 166
 - writing, DataInput.writeFloat; 537
- FloatBuffer class; 565**
- floating-point numbers**
 - arithmetic; 202
 - bit representation, intBitsToFloat methods in floating-point wrapper classes; 192
 - casting to integers; 219
 - comparison and equality operations; 206
 - conversion
 - required for shift operator use with; 210
 - format specifiers for; 627–629
 - strings to/from (table); 316
 - double primitive type, value of; 166
 - expression types; 215
 - float primitive type, value of; 166
 - format specifier; 23
 - IEEE 754–1985 standard, bibliographic reference; 755
 - literals; 168
 - primitive type definitions; 4
 - reading
 - DataInput.readDouble; 537
 - DataInput.readFloat; 537
- strict
 - enums; 154
 - evaluation, strictfp class modifier use; 43, 57
 - interfaces vs. classes; 122
 - strict vs. non-strict arithmetic; 203
 - wrapper classes; 191
 - writing
 - DataOutput.writeDouble; 537
 - DataOutput.writeFloat; 537
- floatToIntBits methods**
 - floating-point wrapper classes; 192
- floatToRawIntBits methods**
 - floating-point wrapper classes; 192
- floatValue method**
 - Number class; 188
- floor method**
 - Math class and StrictMath class; 658
- flow of control; 229–246**
 - See also:* behavior/behavioral; event(s); exception(s); iteration; process(es); reflection; thread(s)
 - assertions; 299–300
 - method invocation and execution; 62
 - nested if–else vs. conditional booleans; 20
 - overview; 9–12
 - term definition; 9
- FlowLayoutManager class; 718**
- flush method**
 - FileOutputStream class; 541
 - FileWriter class; 541
 - OutputStream; 505
 - Writer; 511
- Flushable interface**
 - Formatter implementation of; 632
- flushing**
 - data to disk, strategies for; 541
 - output streams
 - byte; 505
 - character; 511
- font(s)**
 - See also:* characters; text
 - code
 - doc comment tags for; 487
 - in plain font vs., in doc comments; 485
 - java.awt.font package; 720
- for statement; 236–241**
 - enhanced; 239–241
 - while vs.; 10
- for-each loop; 239–241**
- forDigit method**
 - Character class; 193
- form feed (\f)**
 - character literal; 167
- format**
 - internal; 412

Format class; 710

- format method; 710
- parse method; 710

format method

- DateFormat class; 704
- Format class; 710
- Formatter class; 624, 632

format/formatting; 710

- custom; 630
- data, reading, Scanner class use; 641
- dates; 703
- exceptions; 630–631
- file
 - GZIP**, java.util.zip package; 736
 - JAR**, java.util.jar package; 735–736
 - ZIP, JAR** file format based on; 735–736
 - ZIP**, java.util.zip package; 736
- specifier, term definition; 624
- specifiers; 626
 - floating-point conversions; 627–629
 - integer conversions; 626
 - term definition; 23
- string
 - overview; 23–24
 - term definition; 23, 624
 - text, Formatter class; 624–632
 - times; 703

FormatFlagsConversionMismatchException class; 630**Formattable interface**

- formatTo method; 630

FormattableFlags class

- UPPERCASE; 630

Formatter class; 624–632, 631–632

- Appendable interface use by; 332
- dates and times use; 706–708
- format method; 624
- formatted conversion use; 23
- java.util package, formatted output; 501

FormatterClosedException class

- Formatter use; 632

formatTo method

- Formattable interface; 630
- Formatter interface; 629

forName method

- Charset class; 320
- Class class; 398, 400, 411–413
- wildcard return values; 401

forwarding

- interface wrapper multithreaded designs; 353
- term definition; 129
- type-independent implementation reuse; 76

FP-strict

- term definition; 203

frameworks

- event, nesting use in; 133
- extensible, design issues; 109–114

freeMemory method

- Runtime class; 453

Friendly example class; 362**fromString method**

- UUID class; 657

Fuller, R. Buckminster

- quotation; 199

fully-qualified names

- term definition; 36, 412

further reading; 755–760**Future interface; 734****FutureTask class; 734****G****g, G format specifier**

- floating-point conversion use; 627

Game example class; 437**garbage collection; 447–466**

- See also:* cleanup; finalization; memory

- further reading; 756

- interacting with; 452–454

- invoking; 452, 453

- mark-and-sweep; 449

- memory management use of; 15

- model; 448

- reachability; 455

- reference

- counting; 449

- strengths; 455

- RMI**; 731

- term definition; 49, 447

- thread local variables; 383

- threads, ThreadGroup relationship to; 344

- weak references, WeakHashMap class

- implications; 597

gc method

- Runtime class; 452, 453, 676

- System class; 452, 453, 666

generic

- See also:* wildcards

- arrays; 428

- creation; 430–432

- introspection hierarchy figure

- representation of; 399

- classes, declaration; 247

- constructors; 260–264

- further reading; 757

- interfaces, Iterator as example of; 127

- invocations, type inference and; 262–264

- methods; 260–264

- declaration; 261

- documenting type parameters in; 485

- generic** (*cont.*)
 - methods (*cont.*)
 - overloading; 271–271
 - overriding; 271–271
 - type parameters used to declare; 57
 - reification, compatibility design issues; 744–748
 - types; 247–277
 - class extension and; 276–277
 - class extension using; 76
 - compatibility design issues; 744–748
 - declarations; 250–255
 - doc comment tags use; 487
 - impact on method resolution; 225
 - importing; 469
 - inspection; 426–429
 - introspection hierarchy figure
 - representation of; 399
 - invocation, term definition; 250
 - "most specific" algorithm modification
 - for; 272–276
 - nested; 253–255
 - overview; 29–32
 - raw, erasure and; 267–272
 - subtyping; 256–260
 - term definition; 29, 247
 - wildcards and; 256–260
 - working with; 256–260
- GenericArrayType interface**; 428
 - getGenericComponentType method; 428
- GenericDeclaration interface**; 426
 - Constructor class implementation of; 424
 - getTypeParameters method; 405, 426
 - Method class implementation of; 421
 - overview and place in introspection hierarchy; 399
- Georgian language**
 - case handling; 193
- get method**
 - AbstractList class; 612, 615
 - Array class; 430
 - BitSet class; 633
 - Calendar class; 698
 - defining object properties with; 67
 - Field class; 418
 - HashMap class; 615
 - Hashtable class; 619
 - LinkedHashMap class; 592
 - List interface; 581
 - Map interface; 588
 - Reference class; 455
 - WeakHashMap class; 593
- getAbsoluteFile method**
 - File class; 544
- getAbsolutePath method**
 - File class; 544
- getActualTypeArguments method**
 - ParameterizedType interface; 427
- getAllStackTraces method**
 - Thread class; 382
- getAnnotation**
 - AnnotatedElement interface; 414
- getAnnotations**
 - AnnotatedElement interface; 414
- getAttributes method**
 - DirContext interface; 739
- getAvailableIDs method**
 - TimeZone class; 700
- getAvailableLocales method**
 - Calendar class; 697
 - DateFormat class; 704
- getBeginIndex method**
 - FieldPosition class; 705
- getBoolean method**
 - Boolean class; 665
- getBounds method**
 - TypeVariable interface; 426
- getBuffer method**
 - StringBuffer class; 524
- getBundle method**
 - ResourceBundle class; 690
- getBytes method**
 - String class; 319, 319
- getCanonicalFile method**
 - File class; 544
- getCanonicalName method**
 - Class class; 398, 411
- getCanonicalPath method**
 - File class; 544
- getCause method**; 297
 - Throwable interface, reflection use; 413
- getChannel method**
 - FileInputStream class; 540, 565
 - FileOutputStream class; 540
 - RandomAccessFile class; 543, 565
 - Socket class; 565
- getCharacterInstance method**
 - BreakIterator class; 712
- getChars method**
 - String class; 318
 - StringBuilder class; 333
- getClass method**
 - Class class; 400
 - Object class; 100
 - wildcard return values; 401
- getClasses method**
 - Class class; 408
- getClassLoader method**
 - Class class; 437
 - ClassLoader class; 438
- getClassName method**
 - StackTraceElement class; 294

- getCollationKey method**
Collator class; 708
- getComponentType method**
Class class; 407
- getConstructor method**
Class class; 409
- getConstructors method**
Class class; 408
- getContents method**
ClassLoader class; 442
ListResourceBundle class; 691
- getContext method**
AccessController class; 681
- getContextClassLoader method**
Thread class; 438
- getCountry method**
Locale class; 687
- getCurrencyCode method**
Currency class; 694
- getCurrencyInstance method**
NumberFormat class; 710
- getDateInstance method**
DateFormat class; 703
- getTimeInstance method**
DateFormat class; 703
- getDeclaredAnnotations**
AnnotatedElement interface; 414
- getDeclaredClasses method**
Class class; 408
- getDeclaredConstructors method**
Class class; 408
- getDeclaredConstructor method**
Class class; 409
- getDeclaredConstructors method**
Class class, Constructor class use; 425
- getDeclaredField method**
Class class; 409
- getDeclaredFields method**
Class class; 408
- getDeclaredMethod method**
Class class; 409
- getDeclaredMethods method**
Class class; 408
- getDeclaringClass method**
Class class; 407
Enum generic class; 160
Member interface; 416
- getDefault method**
Locale class; 687
TimeZone class; 700
- getDefaultFractionDigits method**
Currency class; 694
- getDefaultUncaughtExceptionHandler method**
Thread class; 380
- getDefaultValue method**
Method class; 421
- getDelay method**
Delayed interface, DelayQueue use; 606
- getDisplayCountry method**
Locale class; 687
- getDisplayLanguage method**
Locale class; 687
- getDisplayName method**
Locale class; 688
TimeZone class; 700
- getDisplayVariant method**
Locale class; 687
- getDSTSavings method**
TimeZone class; 700
- getEnclosingConstructor method**
Class class; 407
- getEnclosingMethod method**
Class class; 407
- getEncoding method**
InputStreamReader class; 513
OutputStreamWriter class; 513
- getEndIndex method**
FieldPosition class; 705
- getEnumConstants method**
Class class; 407
- getenv method**
System class; 670
- getErrorMessage method**
Process class; 667
- getExceptionTypes method**
Constructor class; 424
Method class; 421
- getFD method**
File byte streams; 541
RandomAccessFile class; 543
- getField method**
Field class; 409
- getFields method**
Class class; 408
- getFileName method**
StackTraceElement class; 294
- getFilePointer method**
RandomAccessFile class; 542
- getFirst method**
LinkedList class; 583
- getFirstDayOfWeek method**
Calendar class; 699
- getFloat method**
ByteBuffer class; 565
- getGenericComponentType method**
GenericArrayType interface; 428
- getGenericExceptionTypes method**
Constructor class; 424
Method class; 420

- getGenericInterfaces method**
Class class; 406
- getGenericParameterTypes method**
Constructor class; 424
Method class; 420
- getGenericReturnType method**
Method class; 420
- getGenericSuperClass method**
Class class; 405, 406
- getGenericType method**
Field class; 418
- getGreatestMinimum method**
Calendar class; 698
- getGregorianCalendarChange method**
GregorianCalendar class; 702
- getID method**
Thread class, debugging use; 384
TimeZone class; 700
- getImplementationTitle method**
Package class; 477
- getImplementationVendor method**
Package class; 477
- getImplementationVersion method**
Package class; 477
- getInputStream method**
Process class; 667
- getInstance method**
Calendar class; 697
default locale use; 687
Collator class; 708
DateFormat class; 704
- getInteger method**
Integer class; 665
- getIntegerInstance method**
NumberFormat class; 710
- getInterfaces method**
Class class; 407
- getInvocationHandler method**
Proxy class; 434
- getISO3Country method**
Locale class; 688
- getISO3Language method**
Locale class; 688
- getKey method**
Entry interface; 589
- getKeys method**
ResourceBundle class; 689, 693
- getLanguage method**
Locale class; 687
- getLast method**
LinkedList class; 583
- getLeastMaximum method**
Calendar class; 698
- getLeastSignificantBits method**
UUID class; 656
- getLength method**
Array class; 430
- getLineInstance method**
BreakIterator class; 712
- getLineNumber method**
LineNumberReader class; 527
StackTraceElement class; 294
- getLong method**
Long class; 665
- getLowerBounds method**
WildcardType interface; 428
- getMaximum method**
Calendar class; 698
- getMaxPriority method**
ThreadGroup class; 378
- getMessage method**
Throwable class; 280
- getMethod method**
Class class; 409
- getMethodName method**
StackTraceElement class; 294
- getMethods method**
Class class; 408
- getMinimalDaysInFirstWeek method**
Calendar class; 699
- getMinimum method**
Calendar class; 698
- getModifiers method**
Class class; 406
Member interface; 416
- getMostSignificantBits method**
UUID class; 656
- getName method**
Class class, retrieving fully qualified Class
object names with; 412
File class; 544
Member interface; 416
Package class; 432, 477
Thread class; 341
ThreadGroup class; 377
TypeVariable interface; 426
- getNumberFormat method**
DateFormat class; 704
- getNumberInstance method**
NumberFormat class; 710
- getNumericValue method**
Character class; 193
- getObject method**
ResourceBundle class; 689
- getOffset method**
TimeZone class; 701
- getOutputStream method**
Process class; 667
- getOwnerType method**
ParameterizedType interface; 427

- getPackage method**
 - Class class; 479
 - Package class; 432, 479
- getPackages method**
 - Package class; 432, 479
- getParameterAnnotations method**
 - Constructor class; 424
 - Method class; 421
- getParameterTypes method**
 - Constructor class; 424
 - Method class; 420
- getParent method**
 - ClassLoader class; 438
 - File class; 544
 - ThreadGroup class; 378
- getParentFile method**
 - File class; 544
- getPath method**
 - File class; 544
- getPercentInstance method**
 - NumberFormat class; 710
- getPriority method**
 - Thread class; 359
- getProperties method**
 - System class; 664
- getProperty method**
 - Properties class; 620
 - System class; 664
- getProxyClass method**
 - Proxy class; 433, 435
- getRawOffset method**
 - TimeZone class; 700
- getRawType method**
 - ParameterizedType interface; 427
- getResource method**
 - ClassLoader class; 442
- getResourceAsStream method**
 - Class class; 442
 - ClassLoader class; 442
- getResources method**
 - ClassLoader class; 442
- getReturnType method**
 - Method class; 420
- getRuntime method**
 - Runtime class; 453, 666
- getSecurityContext method**
 - SecurityManager class; 679
- getSecurityManager method**
 - System class; 678
- getSentenceInstance method**
 - BreakIterator class; 712
- getSimpleName method**
 - Class class; 411
- getSpecificationTitle method**
 - Package class; 477
- getSpecificationVendor method**
 - Package class; 478
- getSpecificationVersion method**
 - Package class; 477
- getStackTrace method**
 - StackTraceElement class; 294
 - Throwable class; 382
- getState method**
 - Thread class, debugging use; 384
- getString method**
 - ResourceBundle class; 688
- getStringArray method**
 - ResourceBundle class; 688
- getSuperClass method**
 - Class class; 398
- getSuperClass method**
 - Class class; 407
- getSymbol method**
 - Currency class; 694
- getSystemClassLoader method**
 - ClassLoader class; 438
- getSystemResource method**
 - ClassLoader class; 442
- getSystemResourceAsStream method**
 - ClassLoader class; 442
- getThreadGroup method**
 - Thread class; 376
 - ThreadGroup class; 376
- getTime method**
 - Calendar class; 699
 - Date class; 695
 - GregorianCalendar class; 696
- getTimeInMillis method**
 - Calendar class; 696
- getTimeInstance method**
 - DateFormat class; 703
- getTimeZone method**
 - TimeZone class; 700
- getting**
 - See accessing; retrieving
- getType method**
 - Character class; 195
 - Field class; 418
- getTypeParameters method**
 - GenericDeclaration interface; 405, 426
 - Method class; 421
- getUncaughtExceptionHandler method**
 - Thread class; 380
- getUpperBounds method**
 - WildcardType interface; 428
- getValue method**
 - Entry interface; 589
- getVariant method**
 - Locale class; 687
- getWordInstance method**
 - BreakIterator class; 712

- Gilbert and Sullivan**
 - quotation; 75
 - GlobalHello example class**; 689
 - GMT (Greenwich Mean Time)**
 - epoch measured in terms of; 695
 - raw offset; 700
 - Goethe, Johann Wolfgang von**
 - quotation; 336
 - Gotlieb, Sandra**
 - quotation; 467
 - goto**
 - See also:* flow of control
 - cleanup use, finally as replacement; 290
 - labeled break vs.; 243
 - programming without, alternatives; 246
 - reserved keyword (table); 165
 - granularity**
 - locks, synchronized statement advantages vs. synchronized methods; 350
 - graph**
 - object, term definition; 549
 - graphical user interfaces (GUIs)**
 - java.awt package, details; 717–720
 - thread use; 370
 - GraphicalCardDealer example interface**; 125
 - GraphicalComponent example interface**; 125
 - graphics components**
 - resource bundle objects; 691
 - greater than (>)**
 - comparison operator, term definition; 5
 - greater than comparison; 206
 - Unicode sequence for; 482
 - greater than greater than (>>)**
 - shift bits right, sign fill; 209
 - greater than greater than greater than (>>>)**
 - shift bits right, zero fill; 209
 - greater than or equal (>=)**
 - comparison operator; 206
 - term definition; 5
 - greedy**
 - non-greedy qualifier use, Pattern class capture groups; 650
 - quantifiers, special characters for (table); 752
 - tokenizer, term definition; 164
 - Greenwich Mean Time (GMT)**
 - epoch measured in terms of; 695
 - Gregorian calendar**; 696
 - GregorianCalendar class**; 696, 701–703
 - getGregorianCalendarChange method; 702
 - getTime method; 696
 - isLeapYear method; 702
 - setGregorianCalendarChange method; 702
 - GridBagLayoutManager class**; 718
 - group method**
 - MatchResult interface; 326
 - groupCount method**
 - MatchResult interface; 326
 - groups**
 - daemon
 - daemon threads vs.; 376
 - term definition; 376
 - thread
 - creating; 377
 - printing state, as debugging aid; 385
 - security; 375–379
 - security modification; 376
 - term definition; 375
 - Guaspari, David**
 - quotation; 754
 - GUI (graphical user interface)**
 - design, further reading; 759–760
 - java.awt package, details; 717–720
 - thread use; 370
 - Guthrie Woody**
 - quotation; 183
 - GZIP file format**
 - java.util.zip package; 736
 - GzipInputStream class**; 737
 - GzipOutputStream class**; 737
- ## H
- h, H format specifier**
 - hash code conversion use; 629
 - halt method**
 - Runtime class; 674
 - handleGetObject method**
 - ResourceBundle class; 693
 - handling**
 - events, Observer interface and Observable class use; 635–639
 - HasA relationship**
 - term definition; 107
 - hasChanged method**
 - Observable class; 636
 - hash**
 - See also:* data structures; equality; hash tables; map(s)/mapping
 - codes
 - arrays; 177
 - format conversion; 629
 - resource recovery key; 461
 - string concatenation and representation use; 100
 - wrapper classes; 187
 - maps, serializing; 550
 - hashCode method**
 - Arrays class; 608
 - BitSet class; 634
 - Enum class use; 159
 - Object class; 100

- String class; 312
- wrapper classes; 187
- HashMap class**; 590–591
 - class inspection output example; 404
 - get method; 615
 - hash code use; 313
 - HashSet relationship; 579
 - Hashtable class compared with; 619
 - implementing attribute-value pairs in; 128
 - Map interface implemented by; 590
 - overview; 570
 - readObject method; 554
 - serialization issues; 554
 - WeakHashMap class compared with; 593
 - writeObject method; 554
- HashSet class**; 579
 - implementation of Set interface; 579
 - overview; 569
- Hashtable class**; 619
 - contains method; 620
 - elements method; 619
 - get method; 619
 - isEmpty method; 619
 - keys method; 619
 - Properties class extension of; 620
 - put method; 619
 - remove method; 619
 - size method; 619
- hashtables**
 - See also*: data structures
 - Arrays class
 - deepHashCode method; 608
 - hashCode method; 608
 - characteristics and design issues, HashMap class; 591
 - ConcurrentHashMap class; 606
 - hash code use; 312
 - HashMap class; 570, 590
 - Map interface implemented by; 590
 - HashSet class; 569, 579
 - Hashtable class; 619
 - object identity vs. equivalence; 101
 - WeakHashMap class; 570
- hasMoreElements method**
 - Enumeration interface; 617
- hasMoreTokens method**
 - StringTokenizer class; 652
- hasNext method**
 - Iterator interface; 143, 571
 - ListIterator interface; 572
 - Scanner class; 642
- hasNextDouble method**
 - Scanner class; 642
 - localization support; 651
- hasNextLine method**
 - Scanner class; 645
- hasPrevious method**
 - ListIterator interface; 572
- hasTransparentBounds method**
 - Matcher class; 329
- head element**
 - Queue interface; 569, 585
- header**
 - term definition; 3, 57
- headMap method**
 - SortedMap interface; 590
- headSet method**
 - SortedSet interface; 578
- heap**
 - See also*: memory
 - priority, PriorityQueue use; 586
 - term definition; 13
- HerClass example class**; 102
- hexadecimal**
 - See also*: mathematics; number(s)
 - (a, A) format specifier, floating-point conversion use; 628
 - numbers
 - floating-point literals; 168
 - format specifier; 23
 - integer literals; 167
 - string, toHexString methods; 190
 - (x, X) format specifier, integer conversion use; 627
- hiding/hidden**
 - data, access control modifiers impact on; 48
 - enclosing methods, by inner class method; 141–142
 - field(s); 86
 - accessing with this; 69
 - design reasons; 88
 - enforcing read-only access by; 66
 - parameter naming implications; 17
 - inner class considerations; 140
 - interface constants; 123
 - local inner classes; 144
 - overriding vs., nested type definitions; 148
 - path, testing for, File.isHidden; 545
 - reasons why it is bad style; 180
 - static members; 89
 - type variables, in generic inner classes; 255
- hierarchy**
 - See also*: composite/composition; data structures; inheritance; trees
 - byte streams, type tree, (figure); 502
 - character streams, (figure); 507
 - introspection, (figure); 400
 - root, Object class as; 99–101
 - type
 - Class methods that walk; 400
 - exception, (figure); 280
 - wrapper classes, (figure); 183

high-surrogates range
Unicode supplementary characters; 162

highestOneBit method
Integer class; 190
Long class; 190

HighSpeedPrinter example class; 147

Hoare, C. A. R.
further reading; 756
quotation; 132

holdsLock method
Thread class; 347

hooks (shutdown); 672–673
term definition; 672

horizon
term definition; 646

Host example class; 141

HTML
paragraph breaks, doc comment use; 483
tags, doc comment use; 482

hyperbolic
cosine; 658
sine; 658
tangent; 658

hyphen (-)
floating-point conversion use (table); 628
integer conversion use (table); 627

hypot method
Math class and StrictMath class; 658

hypotenuse; 658

I

I/O

See also: directory(s); disk; file(s); streams

blocking
cancellation request handling; 516
InterruptedIOException use; 367
data-based, term definition; 500
exceptions, IOException class; 563–565
high performance, java.nio use with; 565
network, java.net package use; 499
non-blocking
java.nio use; 566
term definition; 499
package; 499–566
pipes as I/O-based communication
mechanism between threads; 520
standard I/O streams; 662
security setting; 662
text-based, term definition; 500

idempotent
term definition; 728

identifiers; 164
See also: variables

Java

testing for,
Character.isJavaIdentifierPart;
194
testing for,
Character.isJavaIdentifierStart;
194
Unicode vs.; 195
keywords prohibited from being; 165
package; 468
part of variable declarations; 170
semantics of; 178–181
term definition; 164
time zone; 700
Unicode
testing for,
Character.isUnicodeIdentifierPart;
194
testing for,
Character.isUnicodeIdentifierStart;
194

identity

hashcode, retrieving,
System.identityHashCode; 666
reference
equality operator testing for; 206
equivalence compared with; 206
term definition; 100

identityHashCode method

System class; 101, 666

IdentityHashMap class

; 592

idioms

coding, correct finally use; 289

IEEE 754–1985 standard

bibliographic reference; 755
floating-point definition by; 38

IEEEremainder method

Math class and StrictMath class; 658

if

See also: flow of control
reserved keyword (table); 165
while vs., synchronization implications; 355

if-else statement(s)

See also: flow of control
avoiding nesting of, conditional operators use
for; 20
condition operator compared with; 211
definition; 230
flow of control mechanism; 9
introduction; 11

IllegalAccessError class

Class.newInstance use; 424
Field class use; 418
field reflection use; 419
Method class use; 421

- IllegalArgumentException class**; 284
 - array checking use; 19
 - Array class use; 430
 - conventions of use; 571
 - enum use; 160
 - EnumSet.range; 596
 - field reflection use; 419
 - Format.format use; 710
 - Method class use; 421
 - String class use; 336
- IllegalFormatCodePointException class**; 629, 630
- IllegalFormatConversionException class**; 631
- IllegalFormatException class**; 626, 630
- IllegalFormatFlagsException class**; 631
- IllegalFormatPrecisionException class**; 631
- IllegalFormatWidthException class**; 631
- IllegalMonitorStateException class**; 357
- IllegalStateException class**; 294, 473
 - Collection.add use; 605
 - Iterator interface use; 143
 - reaper thread use; 463
 - shutdown use; 673
- IllegalThreadStateException class**; 339, 369, 378
- images**
 - including in doc-files documentation directory; 497
 - java.awt.image package; 720
 - resource bundle objects; 691
- immutable**
 - accessibility vs.; 67
 - BigInteger objects; 722
 - list, creating, Collections.nCopies; 599
 - objects, UUID class; 656
 - properties, term definition; 46
 - singleton
 - list, creating, Collections.singletonList; 600
 - map, creating, Collections.singletonMap; 600
 - set, creating, Collections.singleton; 600
 - String objects, vs. StringBuilder mutability; 305
 - term definition; 7, 22, 171
 - wrappers; 601
 - class values; 185
- implementation**
 - See also:* abstract; inheritance; interface(s); object(s)
 - collections; 611–616
 - declaration vs., interface's distinction from classes; 27
 - independent types, defining with interfaces; 76
 - inheritance
 - interface inheritance vs.; 115
 - term definition; 75
 - interfaces; 127
 - enums permitted to; 153
 - use of; 129
 - iterators; 609–611
 - modifiers, not permitted in interface methods; 122
 - nested types; 149
 - replacing, defining characteristic of overriding; 84
 - Runnable interface; 341
 - synchronization part of; 348, 353
 - version numbers, Package methods accessing; 478
- implements reserved keyword**
 - interface implementation with; 28, 119 (table); 165
- implicit**
 - String object creation; 306
 - type conversions; 216
- import**
 - reserved keyword (table); 165
 - statement
 - importing packages with; 37
 - mechanism description; 469–470
- import(ing)**
 - mechanism, package use with; 469
 - on demand, term definition; 469
 - single type, term definition; 469
 - static
 - import on demand; 72
 - term definition; 46
 - type imports a generalization of; 469
 - static member names; 71–73
 - static nested types; 470
 - types; 37, 469–471
- import static keyword phrase**
 - importing static members with; 72
- ImprovedFibonacci example class**; 9
- inclusive OR (|) operator**; 20
 - precedence; 222
- IncompleteAnnotationException class**
 - annotation reflection use; 415
- increment (++) operator**; 205
 - term definition; 11
- inDaylightTime method**
 - TimeZone class; 701
- IdentityHashMap class**; 101
- index(ing)**
 - array; 173
 - numbering conventions; 176

- index(ing)** (*cont.*)
 - characters in a String object, with `charAt`; 22
 - list elements; 473
 - position for strings, out of bounds exceptions; 307
- indexOf** method
 - List interface; 581
 - String class (table); 308
 - StringBuilder class; 331
 - Vector class; 618
- indexOfSubList** method
 - Collections class; 599
- IndexOutOfBoundsException** class
 - BitSet class use; 633
 - example of; 892
 - List interface use; 581
 - standard use of; 501
 - String class use; 307
 - System.arraycopy use; 665
 - UTF-16 methods use; 198
- inequality**
 - See also:* equality
 - object, testing; 99–100
- InetAddress** class; 724
- infinity**
 - See also:* mathematics
 - arithmetic with; 202
 - floating-point overflow to; 201
 - NEGATIVE_INFINITY constant; 191, 203
 - POSITIVE_INFINITY constant; 191, 203
 - testing for, isInfinite methods in floating-point wrapper classes; 191
- Inflater** class; 737
- InflaterInputStream** class; 737
- InheritableThreadLocal** class
 - childValue method; 383
- inheritance**
 - See also:* class(es); composite/composition; interface(s)
 - @throws tag; 485
 - accessing inherited members; 86
 - constants, interfaces; 123
 - constructors not part of; 81
 - contract, term definition; 75
 - diamond; 115
 - doc comments; 482
 - forms of; 75
 - generic types; 256
 - implementation
 - interface inheritance vs.; 115
 - term definition; 75
 - inner classes; 140–142
 - interface, term definition; 115
 - members; 84–90
 - methods
 - documentation comments; 489–491
 - genericity prohibited from being added to; 272
 - interfaces; 125
 - multiple
 - implementation vs. interface; 123
 - interface use for; 28
 - interfaces and composition as alternative to; 117
 - methods, throws clause requirements; 286
 - single inheritance vs.; 114
 - nested types; 146–148
 - Object class, class hierarchy root; 77
 - single, multiple inheritance vs.; 114
 - term definition; 24
 - type, term definition; 75
- Inherited** class; 396
- init** method
 - Applet class; 720
- initCause** method
 - Throwable class, chaining support; 292
- INITIAL_QUOTE_PUNCTUATION** constant
 - Character class; 195
- initial state**
 - creating; 49
- InitialContext** class; 739
- initialization**
 - See also:* cancellation; cleanup; finalization
 - annotations; 390
 - arrays; 19, 175–176
 - blocks; 54–55
 - enum use; 156
 - term definition; 54
 - classes, final step in preparing a class for use; 441
 - constructors and; 50–56
 - deserialization interaction with; 556
 - enums, arrays compared with; 153
 - expression, initial value setting with; 4
 - fields; 44
 - constructor order relative to; 82
 - extended classes; 80
 - final variables; 171
 - initializers, checked exceptions not permitted; 284
 - lazy, term definition; 47
 - local variable requirements; 170
 - loops; 237
 - objects; 13
 - variables; 49
 - order of; 56
 - parameter variables; 171
 - part of variable declarations; 170

- static; 55
 - checked exceptions not permitted; 284
 - exiting, return use for; 245
 - static initialization block, term definition; 55
- inline/inlining**
 - tags, doc comment, format and use; 483
 - term definition; 96
- inner classes**; 136–142
 - See also*: class(es)
 - anonymous; 144–146
 - enum constants as; 158
 - inner classes, initialization block use; 55
 - thread run method protection with; 345
 - Constructor class vs. Class class; 425
 - declaration statements; 230
 - enclosing object access by; 138
 - extending; 139–140
 - unrelated inner classes; 140
 - hiding of fields and methods by; 140
 - local; 142–144
 - ReaperThread example class; 463
 - restrictions on; 137
 - scope of; 140
 - in static contexts; 144
 - term definition; 136
 - use of synchronized with enclosing object; 351
- Inner example class**; 139, 141
- input**; 499–566
 - See also*: I/O; output; writing
 - sources, Scanner; 643
 - standard I/O streams; 662
 - streams
 - InputStreamReader; 512
 - Reader; 507
 - term definition; 500
- InputMismatchException class**
 - Scanner use; 643
- InputStream class**; 503–505
 - available; 504
 - LineNumberReader relationships with; 511
 - read; 503, 503
 - Reader contrasted with; 509
 - skip; 503
- InputStreamReader class**; 512–514, 513
 - converting bytes to characters, input to a
 - Filter sequence; 518
 - getEncoding method; 513
- insert method**
 - StringBuilder class; 332
- insertElementAt method**
 - Vector class; 617
- insertion**
 - into string buffers; 332
 - order, term definition; 580, 591
- point, ordered lists, Collections.search
 - determination of; 599
- into a queue, Queue.offer; 586
- inspection**
 - class, Class class use; 402–408
- instance(s)**
 - creation expression, term definition; 214
 - enclosing, inner class relationship to; 136
 - members, accessing; 223
 - term definition; 1, 41
 - variables; 44
 - term definition; 14
- instanceof**
 - operator; 208
 - accessing types at runtime with, reflection
 - contrasted with; 414
 - erasure impact on use of; 268
 - testing reference types with; 92
 - reserved keyword (table); 165
- instantiation**
 - objects; 13
 - term definition; 13
- InstantiationException class**
 - Class.newInstance use; 424
- int data type**
 - See also*: data types
 - arrays, name notation; 412
 - boxing conversion range; 199
 - converting to String, before adding to a
 - string buffer; 332
 - converting to/from strings (table); 316
 - definition; 4
 - field values, default; 45
 - integer literals; 168
 - intValue method, Number class; 188
 - Print stream handling; 525
 - reading, DataInput.readInt; 537
 - reserved keyword; 165
 - value of; 166
 - writing, DataInput.writeInt; 537
- intBitsToFloat methods**
 - floating-point wrapper classes; 192
- Integer class**; 188–191
 - bitCount method; 190
 - converting, strings to/from (table); 316
 - getInteger method; 665
 - highestOneBit method; 190
 - lowestOneBit method; 190
 - MAX_VALUE field, BlockingQueue interface
 - use; 605
 - numberOfLeadingZeros method; 190
 - numberOfTrailingZeros method; 190
 - parseInt method; 316
 - reverse method; 190
 - reverseBytes method; 190
 - rotateLeft method; 190

Integer class (*cont.*)

- rotateRight method; 190
- signum method; 191
- toBinaryString method; 189
- toHexString method; 190
- toOctalString method; 189
- toString method; 189
- wrapper class, int primitive type; 166
- wrapper type hierarchy, (figure); 183

IntegerLookup example class; 30**integers**

- See also:* data types; Math class; mathematics; numbers/numeric; StrictMath class
- arithmetic; 202
 - arbitrary precision, BigInteger class; 722
- bitwise operators use; 208
- byte primitive type, value of; 166
- casting; 219
- conversions,
 - format specifiers for; 627
 - strings to/from (table); 316
- expression types; 215
- implicit conversion to floating-point; 217
- int primitive type, value of; 166
- literals; 167
- long primitive type, value of; 166
- primitive type definitions; 4
- reading
 - DataInput.readInt; 537
 - DataInput.readLong; 537
 - DataInput.readShort; 537
- shift operator use with; 210
- short primitive type, value of; 166
- writing
 - DataOutput.writeInt; 537
 - DataOutput.writeLong; 537
 - DataOutput.writeShort; 537

IntegerStack example class; 103**interface keyword**

- arrays, name notation; 412
- interface declaration use; 120
- reserved keyword (table); 165

interface(s); 117–132

- See also:* class(es); field(s); interfaces (example); interfaces (Java-defined); method(s)

abstract

- classes vs.; 131
- type definitions; 117
- annotation; 392–393
- types as special kind of; 388
- class members; 12
- collection; 568
 - abstractions represented by; 567

- declarations; 120

- default implementation, nested class use for; 149

- defining new implementation-independent types with; 76

- design issues; 126–130

- extending; 122

- generic

- Iterator as example of; 127

- overview; 29–32

- hiding, constants; 123

- implementation; 127

- enums permitted to; 153

- use of; 129

- inheritance

- implementation inheritance vs.; 115

- term definition; 115

- inheriting, constants; 123

- listener, term definition; 719

- marker; 130

- marker annotations compared with; 391

- RandomAccess; 584

- Serializable; 550

- Type; 399

- methods

- declaration of; 122

- overloading and overriding of; 125

- modifiable variables in; 149

- modifiers; 122

- methods; 122

- multithreaded design advantages; 353

- named constant declarations; 121

- nested; 133–150

- classes in; 148

- class members; 43, 134

- interface members; 121, 135

- static; 133

- term definition; 29, 133

- overview; 27–29

- public and protected, extensible class design issues; 109

- reference types; 166

- remote, term definition; 727

- representation

- by Class objects; 399

- of types; 183

- retrieving, Class.getInterfaces method; 407

- synchronized wrappers, multithreaded design use; 352

- term definition; 28, 117

- testing if Class object represents,

- Class.isInterface method; 405

- types,

- references, as powerful design tool; 120

- represented by; 183

interfaces (example)

See example interfaces: Attributed;
 CardDealer; Changeable;
 GraphicalCardDealer;
 GraphicalComponent; List<T>;
 Lookup<T>; Resource;
 SerializableRunnable;
 SortedCollection; Verbose; X; Y; Z

interfaces (Java-defined)

See interfaces: ActionListener;
 AnnotatedElement; Annotation;
 Appendable; BeanInfo;
 BlockingQueue; Callable;
 CharSequence; Cloneable;
 Closeable; Collection; Comparable;
 Comparator;
 ConcurrentLinkedQueue;
 ConcurrentMap; Condition; Context;
 DataInput; DataOutput; Delayed;
 DirContext; Entry; Enumeration;
 EventListener; Executor;
 ExecutorService; Externalizable;
 FileFilter; FilenameFilter;
 Flushable; Formattable; Future;
 GenericArrayType;
 GenericDeclaration; Iterable;
 Iterator; LayoutManager;
 LayoutManager2; List;
 ListIterator; Lock; Map;
 MatchResult; Member;
 ObjectInputValidation; Observer;
 ParameterizedType;
 PrivilegedAction; Queue;
 RandomAccess; Readable;
 ReadWriteLock; Remote; Runnable;
 Serializable; Set; SortedMap;
 SortedSet; String; Throwable; Type;
 TypeVariable;
 UncaughtExceptionHandler;
 WildcardType

interference

thread, synchronized method prevention; 347

intern method

String class; 312

internal

format, array names, array names; 412
 JVM errors, asynchronous exceptions; 283
 state, cleanup, finally use for; 289

internationalization

See also: localization; Unicode
 impact on string comparison; 308
 localization and; 685–714
 term definition; 685
 text; 708

Internet domain name

naming packages convention; 37, 468

interning

strings; 311–313

interrupt method

Thread class; 339, 366, 379, 381

interrupted method

Thread class; 366

InterruptedException class; 357, 366

ReferenceQueue.remove use; 459
 throwable by wait; 356

InterruptedException class; 564

blocked thread use; 516
 I/O blocking use; 367

interruption

threads
 cancellation strategy; 365
 synchronization actions; 373

intersection types

term definition; 264

initialValue method

ThreadLocal class; 383

introspection

hierarchy, (figure); 400
 term definition; 397

intValue method

Number class; 188

invalidating

methods, overriding concrete methods with
 abstract; 99

InvalidClassException class; 558, 564**InvalidObjectException class; 564****invariant(s)**

term definition; 296

inversion

boolean; 208

invocation

See also: declaration
 applications, main method use; 73
 constructors, from constructors; 52
 garbage collector; 452, 453
 generic
 type inference and; 262–264
 type, term definition; 250
 methods; 16, 58–60
 conversions that apply to; 218
 finding the correct one; 223
 multiple, thread design issues; 353
 statements; 230
 term definition; 15
 with checked exceptions; 285
 stack, exception handling use; 279
 superclass
 constructors; 80
 methods; 25

InvocationTargetException class

Method class use; 421

- invoke method**
 - Method class; 421
- IOException class**; 563–565
 - checked exception; 34
 - exception design use; 285
 - File.getCanonicalPath use; 544
 - I/O exception handling by; 501
 - InterruptedException subclass of; 367
 - not used by Print streams; 525
 - Piped streams use; 521
 - pipes use; 520, 521
 - Pushback streams use; 530
 - read methods, InputStream; 503
 - Scanner use; 643
- IsA relationship**
 - term definition; 107
- isAbsolute method**
 - File class; 545
- isAccessible method**
 - AccessibleObject class; 418
- isAlive method**
 - Thread class; 365
 - Thread.join defined in terms of; 369
- isAnnotation method**
 - Class class; 405
- isAnnotationPresent**
 - AnnotatedElement interface; 415
- isAnonymousClass method**
 - Class class; 406
- isArray method**
 - Class class; 405
- isAssignableFrom method**
 - Class class; 414
- isDaemon method**
 - ThreadGroup class; 378
- isDefined method**
 - Character class; 194
- isDigit method**
 - Character class; 194
- isDirectory method**
 - File class; 545
- isEmpty method**
 - Collection interface; 575
 - Hashtable class; 619
 - Map interface; 588
- isEnqueued method**; 460
 - Reference class; 455
 - determining if a reference is in a queue; 460
- isEnum method**
 - Class class; 405
- isEnumConstant method**
 - Field class; 418
- isFile method**
 - File class; 545
- isHidden method**
 - File class; 545
- isHighSurrogate method**
 - Character class; 195
- isIdentifierIgnorable method**
 - Character class; 194
- isInfinite methods**
 - floating-point wrapper classes; 191
- isInstance method**
 - Class class; 414
- isInterface method**
 - Class class; 405
- isInterrupted method**
 - Thread class; 366
- isISOControl method**
 - Character class; 194
- isJavaIdentifierPart method**
 - Character class; 194
- isJavaIdentifierStart method**
 - Character class; 194
- isLeapYear method**
 - GregorianCalendar class; 702
- isLenient method**
 - Calendar class; 699
 - DateFormat class; 705
- isLetter method**
 - Character class; 194
- isLetterOrDigit method**
 - Character class; 194
- isLocalClass method**
 - Class class; 406
- isLowerCase method**
 - Character class; 194
- isLowSurrogate method**
 - Character class; 195
- isMemberClass method**
 - Class class; 406
- isNaN method**
 - Double class; 206
 - alternative test to, (footnote); 206
 - Float and Double class use; 166
 - Float class; 206
 - alternative test to, (footnote); 206
 - floating-point wrapper classes; 191
- isNativeMethod method**
 - StackTraceElement class; 294
- ISO (International Standards Organization);**
 - 320
 - 8858–1 character encoding; 320
 - 8859–6, Arabic encoded bytes, converting to
 - Unicode characters; 512
 - control character, testing for,
 - Character.isISOControl; 194
 - LATIN-1 character encoding; 320
 - data representation; 9
 - locale codes, retrieving; 688
- ISO 3166 codes**
 - bibliographic reference; 755

ISO 639 codes

bibliographic reference; 755

ISO 646–US character encoding; 320**ISO 8859–1 standard**

meaning of (table); 754

isPrimitive method

Class class; 405

isProxyClass method

Proxy class; 435

isSealed method

Package class; 478

isSet method

Calendar class; 698

isSpaceChar method

Character class; 194

isSupplementaryCodePoint method

Character class; 195

isSurrogatePair method

Character class; 195

isSynthetic method

Class class; 406

Member interface; 417

isTitleCase method

Character class; 194

isUnicodeIdentifierPart method

Character class; 194

isUnicodeIdentifierStart method

Character class; 194

isUpperCase method

Character class; 195

isValidCodePoint method

Character class; 195

isVarArgs method

Method class; 421

isWhitespace method

Character class; 195

Scanner class use; 643

Iter example class; 143**Iterable interface**

Collection extension of; 571

for-each loop use; 240

Iterator interface vs., Scanner

implications, (footnote); 642

iterator method; 127

list representation use; 129

overview; 569

iteration/iterator; 571–574

See also: collection(s); flow of control;

looping

collection class use; 570

Enumeration interface use; 528

fail-fast iterators, term definition; 574

HashSet, order of time for; 579

list; 572

loop use; 236

over text, with BreakIterator class; 712

sorted order, SortedSet and SortedMap use;
569

SortedSet interface; 577

synchronized wrappers, synchronization of;
604

unmodifiable wrapper use; 601

value stream, Scanner handling; 642

weakly consistent

ArrayBlockingQueue use; 605

ConcurrentLinkedQueue interface; 607

EnumMap use; 596

EnumSet use; 596

LinkedBlockingQueue use; 605

writing iterator implementations; 609–611

Iterator example class; 143**Iterator interface**

AbstractList implementation; 613

Collection extension of; 571

Enumeration interface analogous to; 616

generic interface example; 127

hasNext method; 143, 571

Iterable interface vs., Scanner

implications, (footnote); 642

local inner class use; 142–144

next method; 143, 571

overview; 569

remove method; 143, 572

iterator method

Collection interface; 571, 575

Iterable interface; 127

J**J2SE Development Kit (JDK)**

editing and compiling with; 2

JAR (Java ARchive) file format

java.util.jar package; 735–736

manifest, term definition; 736

JarEntry class; 736**JarFile class; 736****JarInputStream class; 736****JarOutputStream class; 736****Java**

2 Platform Standard Edition, java package

as root package for; 715

identifiers

testing for,

Character.isJavaIdentifierPar

t; 194

testing for,

Character.isJavaIdentifierSta

rt; 194

Unicode vs.; 194

language overview; 1–40

platform, overview; 38

source, versions, characteristics; 742

- java.applet package**
 - contents description
 - details; 720
 - overview; 716
- java.awt.color package**
 - contents description; 719
- java.awt.datatransfer package**
 - contents description; 720
- java.awt.dnd package**
 - contents description; 720
- java.awt.event package**
 - contents description; 720
- java.awt.font package**
 - contents description; 720
- java.awt.geom package**
 - contents description; 720
- java.awt.im package**
 - contents description; 720
- java.awt.image package**
 - contents description; 720
- java.awt package**
 - contents description
 - details; 717–720
 - overview; 716
- java.awt.print package**
 - contents description; 720
- java.beans.beancontext package**
 - contents description; 722
- java.beans package**
 - contents description; 716, 721–722
- java command**
 - program invocation use; 73
- java.crypto package; 732**
- java.io package; 499–566**
 - classification of classes and interfaces in; 500
 - contents description; 715
 - File stream classes; 540–549
 - Serializable interface; 123
 - type tree for byte streams, (figure); 502
- java.lang.annotation package; 387**
 - reflection classes contained in; 397
- java.lang.instrument package**
 - contents description; 716
- java.lang.management package**
 - contents description; 716
- java.lang package**
 - Appendable interface, Writer class
 - implementation of; 511
 - CharSequence interface; 305
 - contents description; 715
 - Deprecated class; 391
 - Enum class; 153, 159–160
 - Math class; 623
 - Package class; 477
 - Readable interface, Scanner use; 642
 - reflection classes contained in; 397
 - SecurityManager class; 678
 - StrictMath class; 623
 - system programming classes; 661
- java.lang.ref package**
 - contents description; 715
 - reference objects; 455
- java.lang.reflect package**
 - contents description; 715
 - reflection classes contained in; 397
- Java language**
 - evolution, technical issues for applications; 741–748
 - further reading; 755
- java.math package**
 - BigDecimal class, Formatter support for; 625
 - BigInteger class, Formatter support for; 625
 - contents description; 716
 - details; 722
- java.naming.event package**
 - contents description; 739
- java.naming.spi package**
 - contents description; 739
- java.net package; 499**
 - contents description; 716
 - details; 724–727
 - Socket class, getChannel method; 565
 - URI class, converting a File object to; 544
 - URL class, converting a File object to; 544
- java.nio.channels package**
 - byte File stream channel methods for
 - working with; 540
 - FileLock class; 566
- java.nio.charset package; 320**
 - contents description; 716
- java.nio.concurrent package**
 - contents description; 716
- java.nio package; 499, 565–566**
 - CharBuffer class; 305
 - Scanner use; 642
 - Charset class, encoding use; 512
 - CharsetDecoder class, encoding use; 512
 - CharsetEncoder class, encoding use; 512
 - contents description; 716
- java package**
 - root Java 2 Platform Standard Edition
 - package; 715
- java.rmi.activation package; 731**
- java.rmi package**
 - contents description; 716
 - details; 727
 - Remote interface; 130
- java.rmi.server package**
 - contents description, details; 727
- java.security.ac1 package; 732**

- java.security.auth package;** 732
- java.security.cert package;** 732
- java.security.interfaces package;** 732
- java.security package**
 - contents description; 715
 - details; 732
 - Permission class; 678
 - Policy class; 680–681
- java.security.sasl package;** 732
- java.security.spec package;** 732
- java.sql package**
 - contents description; 716
 - details; 732–733
- java.text package**
 - contents description; 716
 - localization facilities; 708
 - NumberFormat class, numeric value localization; 651
- java.util.concurrent.atomic package**
 - contents description; 735
- java.util.concurrent.locks package**
 - contents description; 734–735
- java.util.concurrent package**
 - concurrent collection use; 602, 604
 - contents description; 733–735
 - purpose of; 339
- java.util.jar package**
 - contents description; 716
 - details; 735–736
- java.util.logging package**
 - contents description; 716
- java.util package;** 623–657
 - Array class; 177
 - Cloneable and Serializable interface implementations; 570
 - collection interfaces and classes; 567–622
 - Comparator interface; 574
 - contents description; 715
 - Date class; 37
 - EventListener interface; 130
 - Formatter class
 - dates and times use; 706–708
 - formatted conversion use; 23
 - HashMap class; 554
 - IdentityHashMap class; 101
 - Iterator interface; 127
 - local inner class use; 142–144
 - Scanner class, text scanning and conversion; 532
 - subpackages descriptions; 733–737
 - WeakHashMap class, weak reference use; 458
- java.util.prefs package**
 - contents description; 716
- java.util.regex package;** 321
 - See also:* regular expressions; wildcards
 - contents description; 715
 - MatchResult class, Scanner use; 644
 - Pattern class, Scanner class use; 641
- java.util.zip package**
 - contents description; 717
 - details; 736–737
- Java Virtual Machine (JVM)**
 - internal errors, asynchronous exceptions; 283
 - model, advantages; 730
- Java Virtual Machine Tool Interface (JVMTI)**
 - stopThread methods, asynchronous exceptions; 283
- JavaBeans**
 - automatic property manipulation system; 67
 - java.beans package; 721–722
 - Swing components compared with; 740
- javadoc**
 - command, documentation generation with; 481
 - documentation generator tool, specifying doc comment use with; 493
 - term definition; 481
- javax.accessibility package**
 - contents description; 717
 - details; 737
- javax.naming.directory package**
 - contents description, details; 738
- javax.naming package**
 - contents description; 717
 - details; 738
- javax packages**
 - purpose of; 737
- javax.sound.midi package**
 - contents description, details; 739
- javax.sound package**
 - contents description; 717
- javax.sound.sampled package**
 - contents description, details; 739
- javax.swing package**
 - contents description; 717
 - details; 740
- JDBC (Java Database Connectivity)**
 - java.sql package; 732–733
- JDK (Java Development Kit)**
 - evolution, technical issues for applications; 742
- JNDI (Java Naming and Directory Interface)**
 - javax.naming package; 738
- join method**
 - Thread class; 368
 - thread completion wait use; 367
- Julian calendar**
 - Gregorian calendar vs.; 702
- JVM (Java Virtual Machine)**
 - internal errors, asynchronous exceptions; 283
 - model, advantages; 730

JVM Tool Interface (JVMTI)
 debugging support; 677

JVMTI (Java Virtual Machine Tool Interface)
 stopThread methods, asynchronous
 exceptions; 283

JVMTI (JVM Tool Interface)
 debugging support; 677

K

K (key type)
 type variable naming convention; 248

Kerberos
 java.security.sasl package; 732

key(s)
See also: hashtables; map(s)/mapping
 localized objects, ListResourceBundle
 mapping; 691
 Map entries, retrieving, Entry.getValue;
 589
 resource recovery use; 460
 term definition; 460, 587

key/value pair
 Map interface distinguishing characteristic;
 587
 Properties class use; 620

keys method
 Hashtable class; 619

keySet method
 Map interface; 589

keyword(s)
 new, design issues and impact; 743
 (table); 165, 749

L

L/I suffix
 long integer literal indicator; 167

labels
 break use; 241
 case, switch use of; 232
 continue use; 244
 namespace; 179
 @see doc tag use; 484
 statement use; 241

language
 default, establishing in Locale constructor;
 687

largest element
 retrieving, with Collections.max; 575, 597

last method
 SortedSet interface; 578

last-in-last-out
 Queue use; 585

lastElement method
 Vector class; 618

lastIndexOf method
 List interface; 581
 String class (table); 307, 308
 StringBuilder class; 331
 Vector class; 618

lastIndexOfSubList method
 Collections class; 599

lastKey method
 SortedMap interface; 590

lastModified method
 File class; 545

Latin-1 character set
 control character, testing for,
 Character.isISOControl; 194
 data representation; 9
 Unicode relationship to; 162

layout
 converted values, format specifier use; 625

layout manager
 term definition; 718

LayoutManager interface; 718

LayoutManager2 interface; 718

lazy initialization
 term definition; 47

Leach-Salz UUID variant; 657

leaks
 memory, garbage collection handling of; 448

leap second
 (footnote); 701

Least Recently Used (LRU) cache
 LinkedHashMap use; 592

left angle bracket (<)
 format specifier use; 626
 less than comparison operator; 5, 206

left angle bracket equals (<=)
 less than or equal comparison operator; 5,
 206

left angle bracket left angle bracket (<<)
 shift bits left; 209

left parenthesis (()
 floating-point conversion use (table); 628
 integer conversion use (table); 627

legacy
 code, raw types use with; 270
 collection types; 616

Lehman, John
 quotation; 683

length
 arrays; 19, 174
 strings vs.; 307

file
 get, RandomAccessFile.length; 542
 set, RandomAccessFile.setLength;
 542

identifiers; 164

- strings
 - arrays vs.; 307
 - retrieving, `CharSequence.length` use; 306
- Length field**
 - arrays; 19
- Length method**
 - `BitSet` class; 634
 - `CharSequence` interface; 306
 - `File` class; 545
 - `RandomAccessFile` class; 542
 - `String` class; 22, 307
- lenient calendar**
 - term definition; 699
- less than (<)**
 - Unicode sequence for; 482
 - less than comparison operator; 5, 206
- less than or equal (<=)**
 - less than or equal comparison operator; 5, 206
- LETTER_NUMBER constant**
 - `Character` class; 195
- letters**
 - character, testing for,
 - `Character.isLetter`; 194
 - Unicode meaning; 164
- lexical**
 - See also:* parsing
 - elements, Java; 161
 - scanners, Pushback stream use; 529
- libraries**
 - evolution, technical issues for applications; 742
 - native code, loading; 676
- lifecycle**
 - applets, methods that constitute; 720
- limiting**
 - abstract type parameter wildcard use; 31
- Lincoln, Abraham**
 - quotation; 277
- line(s)**
 - breaks, dot-all mode pattern matching; 324
 - multiline mode, dot-all mode pattern matching; 324
 - number, obtaining,
 - `StreamTokenizer.lineno`; 536
 - parsing text into, with `BreakIterator` class; 712
 - scanning, `Scanner` class; 644–647
 - separator
 - format specification; 23, 625
 - `println` method use; 525
 - text, buffered character stream handling; 519
 - tracking, while reading text,
 - `LineNumberReader` class; 525
 - Unix line terminator pattern matching; 324
 - wildcard for matching beginning of; 322
- LINE_SEPARATOR constant**
 - `Character` class; 195
- line.separator field**
 - line separator string defined by; 525
 - `String` class; 519
- lineno method**
 - `StreamTokenizer` class; 536
- LineNumberReader class; 527–528**
 - `getLineNumber` method; 527
 - `setLineNumber` method; 528
 - stream with no output counterpart; 514
 - `System.in` relationships with; 511
- LinkageError class; 281**
 - `Class.forName` use; 413
 - in type hierarchy, (figure); 280
- linked/linking**
 - classes, second step in preparing a class for use; 441
 - lists
 - `LinkedList` class; 583
 - performance of; 580
- LinkedBlockingQueue class; 605**
- LinkedHashMap class; 591–592**
 - `accessOrder` method; 592
 - `get` method; 592
 - `put` method; 592
 - `putAll` method; 592
- LinkedHashSet class; 580**
- LinkedList class; 583–584**
 - `addFirst` method; 583
 - `addLast` method; 583
 - extension of `AbstractSequentialList` class; 615
 - `List` interface implementation; 581
 - overview; 570
 - performance, `ArrayList` compared with; 583
 - `Queue` implementation of; 586
 - reasons why `RandomAccess` interface not implemented by; 585
 - `removeFirst` method; 583
 - `removeLast` method; 583
- List interface; 580**
 - `add` method; 581
 - `get` method; 581
 - `indexOf` method; 581
 - `lastIndexOf` method; 581
 - `listIterator` method; 581
 - overview; 569
 - `remove` method; 581
 - `set` method; 581
 - `subList` method; 581
 - `Vector` class implementation of; 617

List method

File class; 546
 Properties class; 621
 ThreadGroup class, debugging use; 385

list(s)

See also: collection(s); data structures
 AbstractList class, implementation starting point; 611
 AbstractSequentialList class, implementation starting point; 611
 ArrayList class; 570, 582
 binarySearch method, Collections class; 599
 converting an Enumeration to, Collections.list use; 617
 copy method, Collections class; 599
 CopyOnWriteArrayList class; 607
 directory contents
 File.list; 546
 File.listFiles; 546
 elements, indexing; 473
 emptyList, Collections class; 600
 fill method, Collections class; 598
 indexOfSubList method, Collections class; 599
 iteration through; 572
 lastIndexOfSubList method, Collections class; 599
 linked, performance of; 580
 LinkedList class; 570, 583
 List interface; 569
 ListIterator interface; 569, 572
 modifying; 473
 nCopies method, Collections class; 599
 random access, RandomAccess interface; 584
 randomizing, Collections.shuffle; 597
 replaceAll method, Collections class; 598
 representation of, Iterable interface use; 129
 resizable
 creating; 612
 improving performance of; 615
 reverse method, Collections class; 598
 reverse ordering of, Collections.reverse; 597
 rotate method, Collections class; 598
 shuffle method, Collections class; 598
 shuffling, Collections.shuffle; 597
 singleton, creating,
 Collections.singletonList; 600, 660
 sort method, Collections class; 599
 swap method, Collections class; 598
 term definition; 580

unmodifiable, creating; 612
 viewing arrays as, Arrays.asList; 608

List<T> example interface; 29**listener**

interface, term definition; 719
 term definition; 719

ListFiles method

File class; 546

ListIterator class

filtering operations; 611

ListIterator interface; 572

AbstractList implementation; 613
 add method; 473
 hasNext method; 572
 hasPrevious method; 572
 next method; 572
 nextIndex method; 473
 overview; 569
 previous method; 572
 previousIndex method; 473
 remove method; 473
 set method; 473

ListIterator method

AbstractSequentialList class; 615
 List interface; 581

ListResourceBundle class; 691–692

getContents method; 691
 ResourceBundle implementation by; 691

ListRoots method

File class; 546

literals

See also: constant(s)
 boolean; 167
 character; 167
 class; 169
 obtaining type tokens with; 401
 raw types use with; 270
 retrieving Class objects with; 400
 retrieving primitive types with; 413
 synchronized statement use; 351
 wrapper class TYPE field relationship to; 186
 false; 165
 floating-point number; 168
 integer; 167
 null; 165
 reference; 167
 string; 21, 168
 equality testing; 311–313
 term definition; 3
 String object creation with; 305
 term definition; 7, 166
 true; 165
 types and; 166

little-endian

UTF character encoding standard; 320

live objects

- term definition; 448

load factor

- hashtable design, `HashMap` class; 591

Load method

- `Properties` class; 621
- `Runtime` class; 676
- `System` class; 666

LoadClass method

- `ClassLoader` class; 438–439

loaders/loading

- class
 - bootstrap, term definition; 438
 - context, term definition; 437
 - default; 437
 - first step in preparing a class for use; 441
 - parent, term definition; 438
 - system, term definition; 438
- class(es), term definition; 436
- classes; 435–444
- native code; 676
- resources; 442–444

LoadLibrary method

- `Runtime` class; 676
- `System` class; 666

local

- class, declaration statements; 230
- inner classes; 142–144
- variables
 - declarations; 170, 230
 - declaring; 4
 - loop use; 237
 - namespace; 179
 - threads, `ThreadLocal` class use; 382
 - wildcard use; 260

LOCAL_VARIABLE constant

- `ElementType` class; 394

Locale class; 686

- `Formatter` use; 631
- `getCountry` method; 687
- `getDefault` method; 687
- `getDisplayCountry` method; 687
- `getDisplayLanguage` method; 687
- `getDisplayName` method; 688
- `getDisplayVariant` method; 687
- `getISO3Country` method; 688
- `getISO3Language` method; 688
- `getLanguage` method; 687
- `getVariant` method; 687
- locale-sensitive operations implemented in; 685
- `setDefault` method; 687
- `toString` method; 688

Locale method

- `Scanner` class; 651

localization/locale(s); 686–688

- `Formatter` support of; 631, 631–632
- formatting dates and times; 703
- impact on string comparison; 308
- internationalization and; 685–714
- localize, term definition; 686
- `Scanner` class support for; 641, 651
- sensitive
 - operations, term definition; 685
 - term definition; 686
- sensitivity, case issues; 315
- term definition; 685
- text; 708

Lock field

- character stream use for synchronization; 515

Lock interface

- `Condition` interface use; 735
- `Lock` interface implementation; 734–735

locks

- See also:* concurrency; deadlock; synchronized/synchronization; thread(s)
- acquisition order, design precautions; 359
- `Class` class
 - `static synchronized` data use; 351
 - `static synchronized` method use; 348
- deadlock risk avoidance; 362–365
- deadly embrace risk avoidance; 362–365
- files, `FileLock` use; 566
- granularity of, synchronized statement
 - advantages vs. synchronized methods; 350
- in-use status indicator; 338
- `java.util.concurrent.locks` package; 734–735
- `java.util.concurrent` package; 733–735
- monitor
 - `ReentrantLock` class; 735
 - thread synchronization use; 39
 - `volatile` variables compared with; 372
- owner
 - asserting; 348
 - determining; 347
- ownership of; 346
- per-thread behavior; 346
- `ReadWriteLock` interface; 735
- release of; 346
 - `wait` method; 355
- selection of objects for; 349
- synchronization based on, collection classes
 - use; 602
- term definition; 345

Log method

- `Math` class and `StrictMath` class; 658

Log10 method

- `Math` class and `StrictMath` class; 658

- logarithms**; 658
 - logical AND (&)**; 207
 - conditional AND (&&) compared with; 208
 - logical inclusive OR (|)**; 207
 - conditional OR (|) compared with; 208
 - logical negation (!)**; 207
 - logical operators**; 20, 207
 - bitwise operators compared with; 208
 - equality operations use; 206
 - special characters for (table); 753
 - Long class**; 188–191
 - bitCount method; 190
 - converting, strings to/from (table); 316
 - getLong method; 665
 - highestOneBit method; 190
 - lowestOneBit method; 190
 - MAX_VALUE constant; 453
 - numberOfLeadingZeros method; 190
 - numberOfTrailingZeros method; 190
 - parseLong method; 316
 - reverse method; 190
 - reverseBytes method; 190
 - rotateLeft method; 190
 - rotateRight method; 190
 - signum method; 191
 - toBinaryString method; 189
 - toHexString method; 190
 - toOctalString method; 189
 - toString method; 189
 - wrapper class, long primitive type; 166
 - wrapper type hierarchy, (figure); 183
 - long data type**
 - See also*: data types
 - arrays, name notation; 412
 - converting to String, before adding to a string buffer; 332
 - converting to/from strings (table); 316
 - definition; 4
 - field values, default; 45
 - integer literals; 167
 - longValue method, Number class; 188
 - Print stream handling; 525
 - reading, DataInput.readLong; 537
 - reserved keyword; 165
 - value of; 166
 - writing, DataInput.writeLong; 537
 - longValue method**
 - Number class; 188
 - Lookup example class**; 28
 - Lookup<T> example interface**; 30
 - loops(ing)**
 - See also*: flow of control; iteration
 - continue use with; 244
 - exiting, break use for; 242
 - flow of control mechanism; 9
 - for vs. while; 10
 - for-each; 239–241
 - initialization of; 237
 - statements
 - do-while; 9, 235
 - for; 9, 236
 - while; 5, 9, 235
 - variables, in for statement; 11
 - low-surrogates range**
 - Unicode supplementary characters; 162
 - lower bound**
 - wildcard, term definition; 257
 - lowercase**
 - See also*: case; text
 - StreamTokenizer handling; 536
 - term definition; 193
 - testing for, Character.isLowerCase; 194
 - LOWERCASE_LETTER constant**
 - Character class; 195
 - toLowerCaseMode method**
 - StreamTokenizer class; 536
 - lowestOneBit method**
 - Integer class; 190
 - Long class; 190
 - LRU (Least Recently Used) cache**
 - LinkedHashMap use; 592
- ## M
- magic numbers**
 - term definition; 7
 - magnitude**
 - precision vs.; 216
 - main method**; 73
 - term definition; 2
 - MalformedParameterizedTypeException class**; 427
 - GenericArrayType interface; 428
 - WildcardType interface; 428
 - management/manager**
 - layout, term definition; 718
 - names; 178
 - resource, reference queue and key use; 460–464
 - security
 - RMISecurityManager class; 730
 - setting of; 678
 - term definition; 38, 677
 - string buffer capacity; 335
 - thread, security and thread groups; 375–379
 - manifest**
 - term definition; 736
 - Manifest class**; 736
 - Map interface**; 587–590
 - clear method; 588
 - containsKey method; 588
 - containsValue method; 588

- Dictionary class analogous to; 619
- entrySet method; 589
- equals method; 587
- get method; 588
- Hashtable class relationship; 619
- isEmpty method; 588
- keySet method; 589
- overview; 569
- put method; 588
- putAll method; 588
- remove method; 588
- size method; 588
- values method; 589
- map(s)/mapping**
 - See also:* data structures
 - AbstractMap class, implementation starting point; 611
 - buffers, java.nio handling; 566
 - char arrays, to and from strings; 317
 - ConcurrentHashMap class; 606
 - emptyMap, Collections class; 600
 - EnumMap class; 596
 - EnumMap enum specific class; 160
 - files, into memory, MappedByteBuffer; 565
 - hash, serializing; 550
 - HashMap class; 570, 590
 - Map interface; 569, 587
 - singleton, creating,
 - Collections.singletonMap; 600
 - SortedMap interface; 569, 587
 - strings to resources
 - localization tool; 686
 - ResourceBundle use; 688
 - TreeMap class; 569
 - WeakHashMap class; 570
- mapLibraryName method**
 - System class; 676
- MappedByteBuffer class;** 565
- mark-and-sweep garbage collection**
 - term definition; 449
- marker**
 - annotations; 391
 - interfaces; 130
 - marker annotations compared with; 391
 - RandomAccess; 584
 - Serializable; 550
 - Type; 399
- match method**
 - Scanner class; 644
- Matcher class;** 325
 - appendReplacement method; 327
 - appendTail method; 327
 - convenience methods that avoid using; 315
 - hasTransparentBounds method; 329
 - non-anchoring bounds, line scanning horizon value as; 646
 - region method; 329
 - regionEnd method; 329
 - regionStart method; 329
 - regular expression compilation use; 323
 - replaceAll method; 327
 - replaceFirst method; 327
 - reset method; 325
 - StringBuffer class use with, (footnote); 326
 - useAnchoringBounds method; 329
 - usePattern method; 326
 - useTransparentBounds method; 329
- matcher method**
 - Pattern class; 324
- matches method**
 - Pattern class; 324
- matching**
 - See also:* patterns; regular expressions; wildcard(s)
 - multiple characters, wildcard for; 322
 - pattern
 - flags that affect; 324
 - modifying the state of; 325
 - replacing matched characters; 326–330
 - replacing patterns; 326
 - resetting the pattern matcher; 325
 - patterns, efficiency; 329–330
 - regular expression; 321–329
 - single character, wildcard for; 322
- MatchResult interface**
 - end method; 326
 - group method; 326
 - groupCount method; 326
 - Scanner use; 644
 - start method; 326
 - toMatchResult method; 326
- Math class;** 657–659
 - See also:* mathematics; StrictMath class;
 - abs method; 658
 - acos method; 657
 - asin method; 657
 - atan method; 657
 - atan2 method; 657
 - cbrt method; 658
 - ceil method; 658
 - cos method; 657
 - cosh method; 658
 - exp method; 658
 - floor method; 658
 - hypot method; 658
 - IEEEremainder method; 658
 - importing members of; 72
 - log method; 658
 - log10 method; 658
 - max method; 658
 - min method; 658

Math class (*cont.*)

pow method; 658
 random method; 659
 rint method; 658
 round method; 658
 signum method; 658
 sin method; 657
 sinh method; 658
 sqrt method; 16, 658
 static methods, reasons for; 18
 tan method; 657
 tanh method; 658
 toDegrees method; 658
 toRadians method; 657

MATH_SYMBOL constant

Character class; 195

MathContext class; 722**mathematics**

See also: absolute value; arccosine; arcsine; arctangent; arithmetic; `BigDecimal` class; `BigInteger` class; ceiling; cosine; decimal; exponent; floating point; integers; logarithms; `Math` class; `MATH_SYMBOL` constant; maximum; minimum; NaN (Not a Number); numbers/numeric; PI constant; powers; random method; random numbers; round method; sine; square root; `strictfp` keyword; `StrictMath` class; tangent; transcendental
`java.math` package; 716, 722
 professor, U.C. Berkeley, quotation; 133

max method

`Collections` class; 597
 retrieving largest element with; 575
`Math` class and `StrictMath` class; 658

MAX_PRIORITY constant

`Thread` class; 359

MAX_RADIX constant

`Character` class; 192

MAX_VALUE constant

`Character` class; 192
`Long` class; 453
 wrapper classes use; 166

MAX_VALUE field

`Integer` class, `BlockingQueue` interface use; 605
 wrapper classes; 186

maximum; 658**maxMemory method**

`Runtime` class; 453

Member interface; 416–417, 416

`Field` class interaction; 418
`getDeclaringClass` method; 416
`getModifiers` method; 416
`getName` method; 416

`isSynthetic` method; 417
 overview and place in introspection hierarchy; 399
`toGenericString` method; 416
`toString` method; 416

member(s)

See also: class(es); field(s); interface(s); method(s)

access; 223

class; 42

controlling access to; 47–48

examining; 408

fields as; 42

methods as; 43

nested classes and interfaces as; 43

overview; 12

constructors compared with; 51

enclosing types, static nested type as; 134

fields, overview; 12

inheritance of; 84–90

inherited, accessing; 86

interface

constants; 120

methods; 120

nested classes and interfaces; 120

overview; 12

methods, overview; 12

non-static, accessing; 223

overview; 12

protected, access issues; 93–95

redefinition of; 84–90

reflection access; 416–417

static

accessing; 223

hiding of; 89

importing, class name use; 71–73

raw types use with; 270

referencing; 8

static protected, accessing; 95

term definition; 1

types, doc comment specification of; 483

memory

allocation

during object creation; 49

garbage collection handling of problems

with; 448

garbage collection and; 447–466

in-memory streams; 514

leaks, garbage collection handling of; 448

management

garbage collection role; 49

garbage collector; 15

mapping files into, `MappedByteBuffer`; 565

model

synchronization and `volatile`; 370–375

term definition; 371

- MemoryWatchTask example class**; 653
- Mesa language**
 - bibliographic reference; 758
- MessageFormat class**; 710
- meta-annotations**
 - term definition; 394
- Method class**; 420–422
 - Constructor class compared with; 424
 - getDefaultVaLue method; 421
 - getExceptionTypes method; 421
 - getGenericExceptionTypes method; 420
 - getGenericParameterTypes method; 420
 - getGenericReturnType method; 420
 - getParameterAnnotations method; 421
 - getParameterTypes method; 420
 - getReturnType method; 420
 - getTypeParameters method; 421
 - invoke method; 421
 - isVarArgs method; 421
 - overview and place in introspection hierarchy; 399
- METHOD constant**
 - ElementType class; 394
- method(s)**; 56–68
 - See also:* class(es); field(s); interface(s); methods (Java-defined)
 - abstract; 97–99
 - access control use of; 65–68
 - accessibility, overriding algorithms; 472–475
 - accessing, class of, as determinator of method implementation use; 86
 - accessor, term definition; 66
 - annotation of; 392–393
 - body, term definition; 3, 57
 - bridge, term definition; 746
 - calls, statements; 230
 - class; 17
 - class members; 12, 43
 - constructor compared with; 50
 - constructors
 - impact of constructor ordering on; 83
 - precautions for design of; 83
 - vs.; 81
 - declaration, term definition; 3
 - documentation comments, inheritance of; 489–491
 - equals, Object class, equivalence tested by; 207
 - execution and return of; 62
 - final
 - advantages over final classes; 96
 - design considerations; 96
 - finding the correct one; 223
 - generic; 260–264
 - declaration; 261
 - documenting type parameters in; 485
 - overloading; 271–271
 - overriding; 271–271
 - type parameters used to declare; 57
 - headers, term definition; 3, 57
 - interface
 - declaration of; 122
 - member; 120
 - overloading and overriding of; 125
 - invalidating, overriding concrete methods with abstract; 99
 - invocation(s); 15, 58–60
 - conversions that apply to; 218
 - erasure rules; 746
 - multiple, thread design issues; 353
 - operator; 224
 - modifiers; 57
 - overriding issues and restrictions; 85
 - multiple inheritance throws clause requirements; 286
 - namespace; 179
 - native; 74
 - non-static, term definition; 17
 - overloading; 69–71
 - overriding, throws clause handling; 285
 - parameters
 - overview; 15–18
 - passing object references in; 68–69
 - syntax; 58
 - per-thread behavior of locks impact on design of; 346
 - primitive types, wrappers as container for; 184
 - protecting in a multithreaded environment, synchronized accessor use; 347
 - resolution
 - "most specific" algorithm; 224–228
 - "most specific" algorithm, generic type modifications; 272–276
 - retrieving
 - Class.getDeclaredMethod; 409
 - Class.getDeclaredMethods; 408
 - Class.getMethod; 409
 - Class.getMethods; 408
 - return
 - type; 59
 - value of; 62
 - scope; 180
 - semantics of, contract component; 41
 - signature of; 69
 - static; 17, 58
 - synchronized; 348
 - synchronized, using the lock of; 351
 - term definition; 58
 - this reference not applicable to; 68
 - super.finalize, good programming practice; 451

method(s) (*cont.*)

- synchronized; 346–348
 - factoring synchronized statement code into; 352
 - synchronized statement advantages over; 349
 - synchronized statement relationship to; 349
- term definition; 1, 15, 41, 56
- terminating, return use for; 245
- throws clause of; 284
- unsynchronized, synchronized objects use within; 350
- varargs; 60–61
 - overriding; 85
- variable number of arguments; 60–61
 - syntax for; 58, 60
- with checked exceptions, invocation of; 285
- wrappers, common to all; 184–187

MethodBenchmark example class; 98**methods (Java-defined)**

See methods: abs; accept; accessOrder;

acos; activeCount;
 activeGroupCount; add; addAll;
 addElement; addFirst; addLast;
 addObserver; addShutdownHook;
 after; allOf; and; andNot; append;
 appendCodePoint;
 appendReplacement; appendTail;
 arraycopy; asin; asList;
 asSubClass; atan; atan2; available;
 availableCharsets;
 availableProcessors; await;
 before; binarySearch; byteValue;
 cancel; canRead; canWrite;
 capacity; cardinality; cast; cbrt;
 ceil; charAt; charCount;
 checkAccess; checkedCollection;
 checkedList; checkedMap;
 checkedSet; checkedSortedMap;
 checkedSortedSet; checkError;
 checkPermission; childValue;
 clear; clearAssertionStatus;
 clearBit; clearChanged;
 clearProperty; clockSequence;
 clone; close; codePointAt;
 codePointBefore; codePointCount;
 command; commentChar; comparator;
 compare; compareAndSet; compareTo;
 compareToIgnoreCase; compile;
 complementOf; concat; connect;
 contains; containsAll;
 containsKey; containsValue; copy;
 copyInto; copyOf; copyValueOf; cos;
 cosh; countObservers; countTokens;
 createNewFile; createTempFile;

currentThread; currentTimeMillis;
 decode; deepEquals; deepHashCode;
 deepToString; defaultCharset;
 defaultReadObject;
 defaultWriteObject; defineClass;
 delete; deleteCharAt;
 deleteObserver; deleteObservers;
 deleteOnExit; delimiter;
 deprecated; destroy; digit;
 directory; disjoint; doPrivileged;
 dotw; doubleValue; dumpStack;
 element; elementAt; elements;
 empty; emptyList; emptyMap;
 emptySet; enableEvents; end;
 endsWith; enqueue; ensureCapacity;
 entries; entrySet; enumerate;
 enumeration; environment;
 equals; equalsIgnoreCase; equals;
 equalsIgnoreCase; exec; exists;
 exit; exitValue; exp; fill;
 finalize; findClass; findInLine;
 findLoadedClass; findResource;
 findResources; findWithinHorizon;
 first; firstElement; firstKey;
 flags; flip; floatToIntBits;
 floatToRawIntBits; floatValue;
 floor; flush; forDigit; format;
 formatTo; forName; freeMemory;
 fromString; gc; get;
 getAbsolutePath; getAbsolutePath;
 getAllStackTraces; getAnnotation;
 getAnnotations; getAttributes;
 getAvailableIDs;
 getAvailableLocales;
 getBeginIndex; getBoolean;
 getBounds; getBuffer; getBundle;
 getBytes; getCanonicalFile;
 getCanonicalName;
 getCanonicalPath; getCause;
 getChannel; getCharacterInstance;
 getChars; getClass; getClasses;
 getClassLoader; getClassName;
 getCollationKey;
 getComponentType; getConstructor;
 getConstructors; getContents;
 getContext;
 getContextClassLoader;
 getCountry; getCurrencyCode;
 getCurrencyInstance;
 getDateInstance;
 getDateTimeInstance;
 getDeclaredAnnotations;
 getDeclaredClasses;
 getDeclaredConstructor;
 getDeclaredConstructors;
 getDeclaredField;

getDeclaredFields;
 getDeclaredMethod;
 getDeclaredMethods;
 getDeclaringClass; getDefault;
 getDefaultFractionDigits;
 getDefaultUncaughtExceptionHandler; getDefaultVa-
 lue; getDelay;
 getDisplayCountry;
 getDisplayLanguage;
 getDisplayName;
 getDisplayVariant; getDSTSavings;
 getEnclosingConstructor;
 getEnclosingMethod; getEncoding;
 getEndIndex; getEnv;
 getErrorStream;
 getExceptionTypes; getFD;
 getField; getFields; getFileName;
 getFilePointer; getFirst;
 getFirstDayOfWeek; getFloat;
 getGenericComponentType;
 getGenericDeclaration;
 getGenericExceptionTypes;
 getGenericInterfaces;
 getGenericParameterTypes;
 getGenericReturnType;
 getGenericSuperclass;
 getGenericType;
 getGreatestMinimum;
 getGregorianChange; getID;
 getImplementationTitle;
 getImplementationVendor;
 getImplementationVersion;
 getInputStream; getInstance;
 getInteger; getIntegerInstance;
 getInterfaces;
 getInvocationHandler;
 getISO3Country; getISO3Language;
 getKey; getKeys; getLanguage;
 getLast; getLeastMaximum;
 getLeastSignificantBits;
 getLength; getLineInstance;
 getLineNumber; getLong;
 getLowerBounds; getMaximum;
 getMaxPriority; getMessage;
 getMethod; getMethodName;
 getMethods;
 getMinimalDaysInFirstWeek;
 getMinimum; getModifiers;
 getMostSignificantBits; getName;
 getName; getNumberFormat;
 getNumberInstance;
 getNumericValue; getObject;
 getOffset; getOutputStream;
 getOwnerType; getPackage;
 getPackages;
 getParameterAnnotations;
 getParameterTypes; getParent;
 getParent; getParentFile; getPath;
 getPercentInstance; getPriority;
 getProperties; getProperty;
 getProxyClass; getRawOffset;
 getRawType; getResource;
 getResourceAsStream;
 getResources; getReturnType;
 getRuntime; getSecurityContext;
 getSecurityManager;
 getSentenceInstance;
 getSimpleName;
 getSpecificationTitle;
 getSpecificationVendor;
 getSpecificationVersion;
 getStackTrace; getState;
 getString; getStringArray;
 getSuperClass; getSuperClass;
 getSymbol; getSystemClassLoader;
 getSystemResource;
 getSystemResourceAsStream;
 getThreadGroup; getTime;
 getTimeInMillis; getTimeInstance;
 getTimeZone; getType;
 getTypeParameters;
 getUncaughtExceptionHandler;
 getUpperBounds; getValue;
 getVariant; getWordInstance;
 group; groupCount; halt;
 handleGetObject; hasChanged;
 hashCode; hasMoreElements;
 hasMoreTokens; hasNext;
 hasNextDouble; hasNextLine;
 hasPrevious;
 hasTransparentBounds; headMap;
 headSet; holdsLock; hypot;
 identityHashCode; IEEEremainder;
 inDaylightTime; indexOf;
 indexOfSubList; init; initCause;
 insert; insertElementAt;
 intBitsToFloat; intern; interrupt;
 interrupted; intValue;
 intValue; invoke; isAbsolute;
 isAccessible; isAlive;
 isAnnotation;
 isAnnotationPresent;
 isAnonymousClass; isArray;
 isAssignableFrom; isDaemon;
 isDefined; isDigit; isDirectory;
 isEmpty; isEnqueued; isEnum;
 isEnumConstant; isFile; isHidden;
 isHighSurrogate;
 isIdentifierIgnorable;
 isInfinite; isInstance;
 isInterface; isInterrupted;
 isISOControl;

methods (Java-defined) (cont.)*See methods (cont.):*

isJavaIdentifierPart;
 isJavaIdentifierStart;
 isLeapYear; isLenient; isLetter;
 isLetterOrDigit; isLocalClass;
 isLowerCase; isLowSurrogate;
 isMemberClass; isNaN;
 isNativeMethod; isPrimitive;
 isProxyClass; isSealed; isSet;
 isSpaceChar;
 isSupplementaryCodePoint;
 isSurrogatePair; isSynthetic;
 isTitleCase;
 isUnicodeIdentifierPart;
 isUnicodeIdentifierStart;
 isUpperCase; isValidCodePoint;
 isVarArgs; isWhitespace; iterator;
 join; keys; keySet; last;
 lastElement; lastIndexOf;
 lastIndexOfSubList; lastKey;
 lastModified; length; lineno; list;
 listFiles; listIterator;
 listRoots; load; loadClass;
 loadLibrary; locale; log; log10;
 longValue; lowerCaseMode; main;
 mapLibraryName; match; matcher;
 matches; max; maxMemory; min; mkdir;
 mkdirs; modifyAttributes; name;
 nameUUIDFromBytes; nanoTime;
 nCopies; newInstance; newline;
 newProxyInstance; next;
 nextBoolean; nextBytes;
 nextClearBit; nextDouble;
 nextElement; nextFloat;
 nextGaussian; nextIndex; nextInt;
 nextLine; nextLong; nextSetBit;
 nextToken; node; noneOf; notify;
 notifyAll; notifyObservers; of;
 offer; offsetByCodePoints;
 openConnection; or; ordinal;
 ordinaryChar; ordinaryChars;
 paint; parentOf; parse;
 parseBoolean; parseByte;
 parseDouble; parseFloat; parseInt;
 parseLong; parseNumbers;
 parseObject; parseShort;
 parseType; peak; peek; poll; pop;
 pow; previous; previousIndex;
 printf; println; printStackTrace;
 processEvent; propertyName; push;
 pushBack; put; putAll; putIfAbsent;
 putLong; quote; quoteChar; random;
 randomUUID; range; read; read;
 readBoolean; readChar; readDouble;
 readExternal; readFloat;
 readFully; readInt; readLine;
 readLong; readObject; readResolve;
 readShort; readUnshared;
 readUnsignedByte;
 readUnsignedShort; readUTF; ready;
 redirectErrorStream; region;
 regionEnd; regionMatches;
 regionStart; registerValidation;
 remove; removeAll;
 removeAllElements; removeElement;
 removeElementAt; removeFirst;
 removeLast; removeRange;
 removeShutdownHook; renameTo;
 repaint; replace; replaceAll;
 replaceFirst; reset; resetSyntax;
 resolveClass; retainAll; reverse;
 reverseOrder; rint; roll; rotate;
 round; run; runFinalization;
 schedule; scheduleAtFixedRate;
 scheduledExecutionTime; search;
 seek; set; setAccessible;
 setBeginIndex; setBit; setChanged;
 setCharAt;
 setClassAssertionStatus;
 setContextClassLoader; setDaemon;
 setDefault;
 setDefaultAssertionStatus;
 setElementAt; setEndIndex; setErr;
 setFirstDayOfWeek;
 setGregorianChange; setID; setIn;
 setLastModified; setLength;
 setLenient; setLineNumber;
 setMaxPriority;
 setMinimalDaysInFirstWeek;
 setName; setNumberFormat; setOut;
 setPackageAssertionStatus;
 setPriority; setProperties;
 setProperty; setRawOffset;
 setReadOnly; setSecurityManager;
 setSeed; setSize; setStackTrace;
 setTime;
 setUncaughtExceptionHandler;
 setValue; shortValue; shuffle;
 signal; signalAll; signal; sin;
 singleton; singletonList;
 singletonMap; sinh; size; skip;
 skipBytes; slashSlashComments;
 slashStarComments; sleep;
 snapshotIterator; sort; split;
 sqrt; start; startsWith; stop;
 stopThread; store; subList; subMap;
 subSequence; subSet; substring;
 swap; synchronizedCollection;
 synchronizedList;
 synchronizedMap; synchronizedSet;
 synchronizedSortedMap;

- synchronizedSortedSet; tailMap;
- tailSet; take; tan; tanh; timestamp;
- toArray; toBinaryString;
- toByteArray; toCharArray; toChars;
- toCodePoint; toDegrees;
- toGenericString; toHexString;
- toLowerCase; toMatchResult;
- toOctalString; toRadians;
- toString; totalMemory;
- toTitleCase; toUpperCase; toURI;
- toURL; traceInstructions;
- traceMethodCalls; trim;
- trimToSize; typeValue;
- uncaughtException;
- unmodifiableCollection;
- unmodifiableList;
- unmodifiableMap; unmodifiableSet;
- unmodifiableSortedMap;
- unmodifiableSortedSet; unread;
- update; useAnchoringBounds;
- useDaylightTime; useLocale;
- usePattern; userDelimiter;
- userRadix; useTransparentBounds;
- valid; validateObject; valueOf;
- values; variant; version; wait;
- waitFor; whitespaceChars;
- wordChars; write; writeBoolean;
- writeByte; writeBytes; writeChar;
- writeChars; writeDouble;
- writeExternal; writeFloat;
- writeInt; writeLong; writeObject;
- writeReplace; writeShort; writeTo;
- writeUnshared; writeUTF; xor;
- yield
- MIDI (Musical Instrument Digital Interface)**
- javax.sound package; 739
- min method**
- Collections class; 597
- retrieving smallest element with; 575
- Math class and StrictMath class; 658
- MIN_PRIORITY constant**
- Thread class; 359
- MIN_RADIX constant**
- Character class; 192
- MIN_VALUE constant**
- wrapper classes use; 166, 186
- minimum**; 658
- minus (-)**
- binary subtraction operator; 201
- subtraction operator, term definition; 6
- unary negation operator; 201
- minus equals (-=)**
- assignment use; 11
- minus minus (--)**
- decrement operator; 205
- term definition; 11
- prefix and postfix expressions, statements; 230
- MissingBundleException class**; 690
- MissingFormatArgumentException class**;
- 626, 631
- MissingFormatWidthException class**; 631
- mkdir method**
- File class; 546
- mkdirs method**
- File class; 546
- models**
- event
- AWT; 718
- JavaBeans; 721
- garbage collection; 448
- JVM, advantages; 730
- memory
- synchronization and volatile; 370–375
- term definition; 371
- security, JVM, advantages; 730
- single-threaded programming, term
- definition; 337
- modification/modifying**
- See also:* immutable; mutability
- data
- interface design that includes; 121
- interfaces, inner class use; 149
- lists; 473
- implementation guidelines; 612
- objects, parameter passing implications; 63–64
- of state, pattern matching; 325
- string buffer; 331
- strings, StringBuilder use; 330–335
- threads, thread group potential; 376
- unmodifiable wrappers; 601
- Modifier class**; 416
- MODIFIER_LETTER constant**
- Character class; 195
- MODIFIER_SYMBOL constant**
- Character class; 195
- modifiers**
- access; 48
- method declaration use of; 57
- not permitted on interface methods; 122
- private; 48
- protected; 48
- public; 48
- term definition; 19
- annotation relationships to; 392–393
- array variables; 174
- class; 43
- enum; 154

modifiers (*cont.*)
 field; 44
 interface; 122
 methods; 122
 method; 57
 overriding issues and restrictions; 85
 part of variable declarations; 170
 retrieving, `Class.getModifiers` method;
 406
 term definition; 43
 as variable attribute; 170
modifyAttributes method
`DirContext` interface; 739
"modifyThread"
`RuntimePermission` name; 680
Modula-3 language
 bibliographic reference; 759
modular
 two's-complement arithmetic, integer
 arithmetic; 201
modulus (%)
 operator, term definition; 11
monitor locks
`ReentrantLock` class; 735
 thread synchronization use; 39
 volatile variables compared with; 372
monitoring
 operations, stack traces use; 382
monitors
 further reading; 757
More example class; 89
MulticastSocket class; 726
multiple inheritance
See also: inheritance
 doc comments, resolution of; 489–491
 implementation, interface vs.; 115
 interfaces and composition as alternative to;
 117
 methods, throws clause requirements; 286
 semantics, implementation vs. interface; 123
 single inheritance vs.; 114
multiplication (*) operator; 201
See also: arithmetic; numbers/numeric;
 operator(s)
 term definition; 6
multiplicative operators
 precedence; 221
multiprocessing
See also: process(es)
 algorithm issues; 359
 hardware, shared memory values, impact by;
 370
 thread management,
 `Runtime.availableProcessors` use;
 360

multithreading
See also: concurrency; synchronized/
 synchronization; thread(s)
 design issues and strategies, `Runnable`
 flexibility for; 345 xx
 memory model interaction with; 375
 shutdown strategies; 674
 term definition; 338
music
`javax.sound` package; 739
Musical Instrument Digital Interface (MIDI)
`javax.sound` package; 739
mutability
See also: modification/modifying
`StringBuilder` objects, vs. `String`
 immutability; 306
MUTLILINE flag
`Pattern` class; 324
mutual exclusion
 guaranteed to synchronized threads; 347
 term definition; 346
mutually comparable
 term definition; 118
MyClass example class; 102
Mythical Man-Month
 bibliographic reference; 759
MyUtilities example class; 33

N

n line separator
 format specification; 625
Name class; 738
Name example class; 551
name method
`String` class, enum use; 155
name(s)/naming
See also: namespaces; package(s)
 algorithm for determining the semantics of;
 180
 array component type, notation for; 412
 attributes, term definition; 76
 binary, term definition; 411, 412
 canonical
 package, import on demand use; 469
 term definition; 411
 classes; 407, 411–413
 conflicts, packages as tool for preventing; 36
 constructors; 51
 conventions, Internet domain name use; 37
 file, manipulation of, `File` class; 543
 fully qualified
 package naming use; 469
 retrieving, `Class.getName` method; 412
 static import use; 72
 term definition; 412

- fully-qualified, term definition; 36
- JNDI**, `javax.naming` package; 738
- meaning of; 178–181
- packages; 468
- parameters, impact on field name use; 17
- RMI** registry; 731
- simple, term definition; 411
- static
 - members, importing; 71–73
 - nested class; 135
- threads; 341
- type
 - creation by both classes and interfaces; 118
 - term definition; 42
 - with dollar signs in them; 149
- NameClassPair class**; 738
- named constants**
 - interface declarations; 121
 - overview; 7–8
 - term definition; 7
- namespaces**
 - See also*: name(s)/naming; package(s)
 - class loader definition of; 437
 - kinds of; 178
 - packages creation of; 467
 - term definition; 178
- nameUUIDFromBytes method**
 - UUID class; 656
- NaN (Not a Number)**
 - See also*: mathematics; number(s)
 - casting to zero; 219
 - comparison and equality operations with; 206
 - differences between floating-point wrappers and primitives; 191
 - constant, `Float` and `Double` class use; 166
 - floating-point
 - arithmetic use; 202
 - wrapper classes; 191
 - formatting; 628
 - invalid arithmetic expression use; 202
 - term definition; 166
 - testing for; 206
 - alternative test, (footnote); 206
 - `isNaN` methods in floating-point wrapper classes; 191
- nanoTime method**
 - `System` class; 665
- narrowing**
 - conversion, term definition; 91
 - conversions, casts use for; 219
 - primitive conversion, term definition; 217
- native**
 - code, loading; 676
 - method modifier; 57
 - methods; 74
 - throws clause issues; 286
 - native method declaration with; 74
 - overriding considerations; 85
 - reserved keyword (table); 165
- natural**
 - equivalence, term definition; 574
 - ordering
 - `SortedSet` default; 577
 - term definition; 574
- nCopies method**
 - `Collections` class; 599
- negation**
 - unary negation (-) operator; 201
- negative**
 - zero, testing for; 202
- NEGATIVE_INFINITY constant**; 203
 - `Float` and `Double` class use; 166
 - floating-point wrapper classes; 191
- nested/nesting**
 - See also*: inner classes
 - arrays; 174
 - classes; 133–150
 - class members; 43, 134
 - interface members; 121, 134
 - in interfaces; 148
 - static; 133, 134
 - static, `Character.Subset`; 195
 - static, `Character.UnicodeBlock`; 195
 - term definition; 133
 - comments prohibited from; 163
 - enum classes; 135
 - generic types; 253–255
 - interfaces; 133–150
 - class members; 43, 134
 - interface members; 121, 135
 - term definition; 29, 133
 - loops, `continue` use with; 244
 - package; 476
 - scopes; 180
 - hiding not permitted in a code block; 181
 - static nested types, importing; 470
 - types
 - accessing information about, `Class` class use; 406
 - implementation of; 149
 - inheritance of; 146–148
 - name notation; 412
 - static; 133–136
- NetPermission class**; 679
- networks**
 - failures, remote interface handling; 728
 - I/O, `java.net` package use; 499
 - `java.net` package; 724–727
- NEW constant**
 - `Thread` class, `State` nested enum; 384

- new operator**; 214
 - constructor relationship to new execution; 50
 - creating objects with; 13, 49
 - evaluation order; 215
 - expressions, statements; 230
 - precedence; 221
 - reflection relationship to; 397
 - reserved keyword (table); 165
- newInstance method**
 - Array class; 429
 - compatibility issues; 747
 - Class class; 423
 - Constructor class; 425
- newline**
 - character literal; 167
 - embedding in strings; 168
 - format specifier; 23
 - line separator; 519
 - LineNumberReader class; 527
- newLine method**
 - BufferedWriter class; 519
- newInstance method**
 - Proxy class; 433
- next method**
 - Iterator interface; 143, 571
 - ListIterator interface; 572
 - Scanner class; 642
- nextBoolean method**
 - Random class; 640
- nextBytes method**
 - Random class; 640
- nextClearBit method**
 - BitSet class; 633
- nextDouble method**
 - Random class; 640
 - Scanner class; 642
- nextElement method**
 - Enumeration interface; 617
- nextFloat method**
 - Random class; 640
- nextGaussian method**
 - Random class; 640
- nextInt method**
 - ListIterator interface; 473
- nextInt method**
 - Random class; 640
 - Scanner class, localization support; 651
- nextLine method**
 - Scanner class; 645
- nextLong method**
 - Random class; 640
- nextSetBit method**
 - BitSet class; 633
- nextToken method**
 - StreamTokenizer class; 532
 - StringTokenizer class; 652
- no-arg constructors**
 - term definition; 53
- node method**
 - UUID class; 657
- NON_SPACING_MARK constant**
 - Character class; 195
- non-blocking I/O**
 - java.nio use; 566
 - term definition; 499
- non-static**
 - field, term definition; 14, 46
 - initialization blocks, checked exceptions use
 - by; 285
 - members, accessing; 223
 - method, term definition; 17
- noneOf method**
 - EnumSet class; 595
- NORM_PRIORITY constant**
 - Thread class; 359
- NoSuchAttributeException example class**; 281
- NoSuchElementException class**
 - conventions of use; 571
 - iterator implementation use; 610
 - Iterator interface use; 143
 - Scanner use; 643
- NoSuchFieldException class**
 - Class class use; 409
- NoSuchMethodException class**
 - Class class use; 409
- not FP-strict**
 - term definition; 203
- NotActiveException class**; 564
- notation**
 - array component type names; 412
 - big *O*
 - AbstractMap element handling
 - performance; 615
 - ArrayList element handling
 - performance; 582
 - Arrays.sort element handling
 - performance; 608
 - balanced tree time requirements,
 - TreeSet; 580
 - (footnote); 579
 - HashMap element handling performance;
 - 590, 615
 - LinkedList element handling
 - performance; 583
 - Map element handling performance; 589
 - priority heap operations; 586
 - TreeMap key/value pair handling
 - performance; 594

- scientific
 - (e, E) format specifier, floating-point conversion use; 627
 - format specifier; 23
 - (g, G) format specifier, floating-point conversion use; 628
- notification(s)**
 - design strategies, `notify` vs. `notifyAll`; 355
 - events, `java.naming.event` package; 739
 - ordering, design precautions; 359
 - thread methods for; 354
- notify method**
 - Object class; 354, 357
 - Thread class, `notifyAll` use vs.; 355
- notifyAll method**
 - Object class; 354, 357
 - Thread class, `notify` vs.; 355
- notifyObservers method**
 - Observable class; 636
- NotSerializableException class**; 552, 556, 564
- null**
 - array reference vs. empty array; 174
 - cautions against using in queues; 586
 - collection elements, optional handling of; 577
 - element, set restrictions; 577
 - empty queue value; 586
 - invalid argument, `NullPointerException` thrown; 311
 - object reference
 - assignment compatibility of; 91
 - default; 13
 - passing as an argument; 61
 - reference literal; 167
 - String conversion; 221
 - term definition; 13
 - type testing use; 92
 - NullPointerException class**; 298
 - conventions of use, Collection methods; 571
 - documentation of; 485
 - dot operator with `null` reference; 224
 - field reflection use; 419
 - Map interface use; 588
 - Method class use; 421
 - PriorityQueue use; 587
 - standard use of, I/O stream methods; 501
 - String methods use of; 311
 - Number class**; 188–192
 - `byteValue` method; 188
 - `doubleValue` method; 188
 - `floatValue` method; 188
 - `intValue` method; 188
 - `longValue` method; 188
 - `shortValue` method; 188
 - wrapper type hierarchy, (figure); 183
 - NumberFormat class**; 710
 - DateFormat class use; 704
 - `getCurrencyInstance` method; 710
 - `getIntegerInstance` method; 710
 - `getNumberInstance` method; 710
 - `getPercentInstance` method; 710
 - numeric value localization; 651
 - numberFormat field**
 - DateFormat class; 705
 - NumberFormatException class**; 184, 188, 317
 - numberOfLeadingZeros method**
 - Integer class; 190
 - numberOfTrailingZeros method**
 - Integer class; 190
 - Long class; 190
 - numbers/numeric**
 - See also*: arithmetic; decimal; floating-point numbers; integers; mathematics
 - decimal, arbitrary precision, `BigDecimal` class; 724
 - decoding strings into; 317
 - floating-point, primitive type definitions; 4
 - integers, primitive type definitions; 4
 - magic, term definition; 7
 - promotion
 - See also*: widening, primitive conversions conversions that apply to; 218
 - pseudo-random, `Random` class; 639–641
 - random, `Random` class; 639–641
 - recognized by `StreamTokenizer`; 532
 - replacing, pattern matching use; 328
 - values
 - localization of, `NumberFormat` class use; 651
 - retrieving, `Character.digit`; 193
 - retrieving, `Character.getNumericValue`; 193
 - nval field**
 - String class, `TT_NUMBER` value contained in; 532

O

 - o format specifier**
 - integer conversion use; 627
 - O notation**
 - See*: big O
 - Oberon language**
 - bibliographic reference; 759

Object class; 26, 99–101

- class hierarchy root; 77
- clone method; 100
 - cloning method design attitudes towards; 102
 - cloning method design use; 101
- dangers of using as a generic reference; 247
- equals method; 100
 - equivalence tested by; 207
- finalize method; 100, 450
 - enums not permitted to override; 154
- getClass method; 100
 - wildcard return values; 401
- hashCode method; 100
- notify method; 354, 357
- notifyAll method; 354, 357
- readResolve method; 557
- toString method; 59, 100
- utility methods; 100
- wait method; 354, 357
 - ReferenceQueue.remove compared with; 460
- wrapper type hierarchy, (figure); 183
- writeReplace method; 557

object(s)

See also: class(es); inheritance; reference

- byte streams, serialization use; 549
- classes and; 41–74
- cloning; 101–107
- construction of; 50–54
- Constructor, inner class use; 425
- creating; 13, 49–50, 423–426
 - creation expressions as statements; 230
 - generic type implications; 251
- current, term definition; 16
- dead
 - phantom reachability; 456
 - term definition; 448
- deleting
 - garbage collector role; 15, 49
- enclosing
 - inner class relationship to; 136, 138
- equality
 - IdentityHashMap notion of; 592
 - serialization design issue; 557
 - testing; 99–100
- graph, term definition; 549
- immutable, UUID class; 656
- initialization of; 50–56
- live, term definition; 448
- modification, parameter passing
 - implications; 64–64
- object-oriented design, synchronization
 - issues as illustration of the power of; 352
- overview; 12–15
- receiver, term definition; 16

- references; 454–465
 - method invocation use; 3, 58
 - method parameter use; 68–69
 - object type vs.; 26
 - parameter passing implications; 63–64
 - term definition; 13, 455
 - vs.; 42
- references, binding referent object to; 455
- resurrecting, during finalization; 452
- serialization; 549–563
 - cloning compared with; 106
 - term definition; 501
 - state, string representation of; 59
 - string, overview; 21–24
 - target, term definition; 16
 - term definition; 1, 41
 - text representation, Formatter class; 624–632
- types
 - object type vs.; 26
 - string representation of; 428–429
 - variables, initialization of; 49
 - versioning, serialization issues; 557–559

Object streams; 549**object-oriented design**

- further reading; 756

ObjectInputStream class; 549

- defaultReadObject method; 558
- readObject method; 553
- readUnshared method; 551, 560

ObjectInputStream.GetField class

- serialization use; 561

ObjectInputValidation interface

- registerValidation method; 557
- validateObject method; 557

Objective C language

- bibliographic reference; 759

ObjectOutputStream class; 549

- defaultWriteObject method; 558
- writeObject method; 551
- writeUnshared method; 551, 560

ObjectOutputStream.PutField class

- serialization use; 561

ObjectStreamClass class

- serialization issues; 558

ObjectStreamException class; 551, 564**ObjectStreamField class**

- representation of old fields in serialized data use; 560

Observable class; 635–639

- addObserver method; 636
- clearChanged method; 636
- countObservers method; 637
- deleteObserver method; 636
- deleteObservers method; 637
- hasChanged method; 636

- notifyObservers method; 636
- setChanged method; 636
- Observer interface**; 635–639
- update method; 636
- octal**
 - See also*: numbers/numeric integer literals; 168
 - numbers, string representation, toOctalString method; 189
 - (o) format specifier, integer conversion use; 627
 - value of a char, character literal; 167
- ODBC (Open Database Connectivity)**
 - java.sql package; 732–733
- of method**
 - EnumSet class; 595
 - UnicodeBlock class; 195
- offer method**
 - AbstractQueue class; 612
 - BlockingQueue interface; 604
 - Queue interface; 586
- offsetByCodePoints method**
 - Character class; 196, 197
 - String class; 336
 - StringBuffer class; 336
 - StringBuilder class; 336
- once-only task scheduling**
 - term definition; 655
- openConnection method**
 - URL class; 725
- operation(s)**
 - arithmetic; 201
 - class, methods used to implement; 15
 - order, thread synchronization issues; 347
 - String class; 306–308
- Operations example class**; 382
- operator(s)**
 - See also*: methods
 - AND (&)**, bitwise operations; 208
 - arithmetic
 - binary; 201
 - term definition; 5
 - assignment; 212
 - term definition; 11
 - associativity; 221
 - term definition; 221
 - binary, combination with equals operator in variable assignment; 11
 - bitwise; 20
 - boolean; 20, 207, 208
 - cast, narrowing conversions requirement of; 91
 - comparison, term definition; 5
 - compound assignment; 213
 - condition; 210–212
 - conditional; 20
 - curly braces ({}); 230
 - decrement; 205
 - expressions, and tokens; 161–182
 - increment; 205
 - instanceof; 208
 - testing reference types with; 92
 - logical; 20
 - bitwise operators compared with; 208
 - new; 214
 - postfix, term definition; 11
 - precedence; 221, (table); 221–222, 750
 - prefix, term definition; 11
 - shift; 209
 - ternary (?:); 210–212
 - unary
 - bitwise complement (~); 208
 - new; 214
 - XOR (^)**, bitwise operations; 208
- optimization**
 - See also*: performance
 - array range check, (footnote); 173
 - clarity tradeoffs, in pattern matching; 330
 - final
 - advantages for; 96
 - method advantages; 111
 - method parameter impact on; 65
 - Iterator implementation
 - strict floating-point evaluation impact on; 203
- OptionalDataException class**; 564
- or method**
 - BitSet class; 634
- OR (|) operator**
 - bitwise operations; 208
 - logical operations; 207
- OR operators**
 - exclusive **OR**; 20
 - inclusive **OR**; 20
- order(ing)**
 - See also*: sorting
 - byte, UTF-16; 320
 - catch clauses; 288
 - characters in a string buffer, reversing; 333
 - class
 - loading; 439
 - modifiers; 43
 - collection class use; 570
 - Comparable and Comparator interface use; 574
 - constructor dependencies; 81–82
 - constructor invocation, relative to variable initialization; 50
 - deserialization; 552–553
 - differences between floating-point wrappers and primitives; 191
 - enumeration, token sequence viewed as; 651

order(ing) (*cont.*)

- execution, thread requirements for; 375
- expression evaluation; 214
- first-in-first-out, Queue use; 585
- garbage collection issues; 452
- initialization, impact on static initialization
 - block execution; 56
- insertion, term definition; 580, 591
- invocation, constructors in extended classes; 80
- iteration, SortedSet and SortedMap use; 569
- last-in-last-out, Queue use; 585
- lists; 580
- lock acquisition, design precautions; 359
- map entries, LinkedHashMap class; 591
- method modifiers, method declarations; 58
- name scope search, algorithm for the semantics of a name; 180
- natural
 - Comparable interface characteristic; 118
 - SortedSet default; 577
 - term definition; 574
- notifications, design precautions; 359
- operation, thread synchronization issues; 347
- priority heap, PriorityQueue use; 586
- program, term definition; 371
- resource, term definition; 364
- reverse
 - Collections.reverse; 598
 - Collections.reverseOrder; 597
- serialization; 552–553
- shutdown hook issues; 673
- SortedMap interface; 589
- string, creating a canonical ordering; 309
- synchronization, term definition; 372
- thread priority issues; 359
- total, term definition; 574
- Unicode character, String.compareTo; 309
- variable update, synchronization issues; 370

ordinal method

- Enum generic class; 159

ordinary characters

- term definition; 532

ordinaryChar method

- StreamTokenizer class; 534

ordinaryChars method

- StreamTokenizer class; 534

org.ietf.jgss package; 732**org.omg.CORBA package**

- contents description; 717
- details; 740

org.omg.CosNaming package

- contents description; 717
- details; 740

organization

- See:* control flow; inheritance; package(s)

OTHER_LETTER constant

- Character class; 195

OTHER_NUMBER constant

- Character class; 195

OTHER_PUNCTUATION constant

- Character class; 195

OTHER_SYMBOL constant

- Character class; 195

out field

- System class; 662

Outer example class; 139, 141, 351**OutOfMemoryError class; 215**

- garbage collector use; 456
- impact on evaluation of new expression; 215
- during object creation; 49

output; 499–566

- See also:* I/O; input; writing
- standard I/O streams; 662

streams

- OutputStream; 505–506
- OutputStreamWriter; 512
- term definition; 500
- Writer; 510–511

OutputStream class; 505–506

- close; 505
- flush; 505
- Formatter use; 631
- write; 505

- Writer class compared with; 511

- Writer contrasted with; 511

OutputStreamWriter class; 512–514, 513

- converting bytes to characters, output from a Filter sequence; 518
- getEncoding method; 513

overflow

- floating-point; 202

overloading

- See also:* overriding

- design strategies and issues; 70–71
- enclosing methods, inner class method hiding of; 141–142

- generic methods; 271–271

- methods; 69–71

- finding the correct one; 223

- interfaces; 125

- “most specific” method resolution algorithm; 224–228

- generic type modifications; 272–276

- overriding compared with; 84

- term definition; 6, 69, 84, 126

Override class; 396**override-equivalent signatures**

- term definition; 271

overriding; 84

See also: inheritance; overloading
 accessibility and; 88
 conflict potential in multiple interface inheritance; 125
 fields prohibited from; 86
 generic methods; 271–271
 hiding vs., nested type definitions; 147
 method accessibility and; 472–475
 methods
 interfaces; 125
 throws clause handling; 285
 overloading compared with; 84
 prevention of, private use for, (footnote); 554
 superclass behavior; 24
 term definition; 25, 84
 varargs methods; 85

overview; 1–40

HTML file, documentation use; 496

owner

lock, determining; 347

Oxford Union Society

quotation; 481

P**Package class**; 432, 477–479, 477

getImplementationTitle method; 477
 getImplementationVendor method; 477
 getImplementationVersion method; 477
 getName method; 432, 477
 getPackage method; 432, 479
 getPackages method; 432, 479
 getSpecificationTitle method; 477
 getSpecificationVendor method; 478
 getSpecificationVersion method; 477
 isSealed method; 478
 overview and place in introspection hierarchy; 399

PACKAGE constant

ElementType class; 394

package.html file

documentation use; 496

package keyword

reserved keyword (table); 165

package(s); 467–480

See also: context(s); name(s)/naming;
 namespaces; packages (Java-defined)
 access; 471–475
 protected members; 94
 scope; 48
 annotations; 392–393, 476
 contents; 475–476
 documenting; 496
 I/O; 499–566
 mechanism, naming packages with; 468

namespace; 178

naming; 468

nesting; 476

overview; 36–37

sealed, term definition; 478

standard; 715–740

subpackage trust issues; 472

utility; 733–737

package-info.java file

documentation use; 496

packages (Java-defined)

See packages: java.applet;

java.awt.color;

java.awt.datatransfer;

java.awt.dnd; java.awt.event;

java.awt.font; java.awt.geom;

java.awt.im; java.awt.image;

java.awt; java.awt.print;

java.beans.beancontext;

java.beans; java.crypto; java.io;

java.lang.annotation;

java.lang.instrument;

java.lang.management; java.lang;

java.lang.ref; java.lang.reflect;

java.math; java.naming.event;

java.naming.spi; java.net;

java.nio.channels;

java.nio.charset;

java.nio.concurrent; java.nio;

java; java.rmi.activation;

java.rmi; java.rmi.server;

java.security.acl;

java.security.auth;

java.security.cert;

java.security.interfaces;

java.security;

java.security.sasl;

java.security.spec; java.sql;

java.text;

java.util.concurrent.atomic;

java.util.concurrent.locks;

java.util.concurrent;

java.util.jar; java.util.logging;

java.util; java.util.prefs;

java.util.regex; java.util.zip;

javax.accessibility;

javax.naming.directory;

javax.naming; javax.sound.midi;

javax.sound; javax.sound.sampled;

javax.swing; org.ietf.jgss;

org.omg.CORBA; org.omg.CosNaming

paint method

Component class; 718

Container class; 718

- pair**
 - key/value, Map interface distinguishing characteristic; 587
- PARAGRAPH_SEPARATOR constant**
 - Character class; 195
- PARAMETER constant**
 - ElementType class; 394
- parameter(s)**
 - See also:* arguments; generic abstract type, T as example of; 30 annotation of; 392–393 bounded type; 252–253 doc comment tag for; 485 list, term definition; 2 methods; 58 overview; 15–18 passing object references in; 68–69 object references, implications for object member modification; 63–64
 - pass
 - by reference vs. pass by value; 65
 - by value implications; 63
 - passing, direct assignment not the same as, (footnote); 90
 - scope; 180
 - type
 - bounded, term definition; 253
 - doc comment syntax; 485
 - term definition; 250
 - values; 63–65
 - variable number; 60–61
 - arguments, syntax for; 58, 60
 - variables; 171
- parameterized types**
 - See also:* generic; wildcards array restrictions; 268–270 bounded wildcards and; 257 Class objects as; 400–402 clone method issues; 747 reflection; 427 term definition; 250 variables, raw types relationship to; 745
- ParameterizedType interface**
 - Field class interaction; 418
 - getActualTypeArguments method; 427
 - getOwnerType method; 427
 - getRawType method; 427
 - problems,
 - MalformedParameterizedTypeException use; 427
- parent**
 - class loader, term definition; 438
 - process, term definition; 667
 - ThreadGroup, relationships of; 377
- parentheses (C)**
 - method invocation operator; 224
 - parameter list use; 2
 - precedence control in expressions; 222
 - type casting operator; 27, 91, 219
- parentOf method**
 - ThreadGroup class; 378
- Parker, Dorothy**
 - quotation; 151
- parse method**
 - DateFormat class; 705
 - Format class; 710
- parseBoolean method**
 - Boolean class; 316
- parseByte method**
 - Byte class; 316
- parseDouble method**
 - Double class; 316
- parseFloat method**
 - Float class; 316
- ParseFormatException class; 710**
- parseInt method**
 - Integer class; 316
- parseLong method**
 - Long class; 316
- parseNumbers method**
 - StreamTokenizer class; 535
- parseObject method**
 - DateFormat class; 705
- parseShort method**
 - Short class; 316
- parseType method**
 - floating-point wrapper classes; 192
 - integer wrapper classes; 188
 - wrapper classes; 187
- parsing; 710**
 - See also:* delimiters; lexical; token(s) data, Scanner class use; 641 dates; 703
 - DateFormat class use; 705
 - lexical elements; 161
 - Pushback stream use; 529
 - streams for building parsers; 515
 - strings into tokens; 651
 - text, StreamTokenizer class; 532–536
 - text boundaries; 712
 - times; 703
 - tokens as product of; 164
- partial failures**
 - term definition; 728
- pass by reference**
 - pass by value vs.; 65
 - term definition; 65
- pass by value**
 - pass by reference vs.; 65
 - term definition; 63

PassByValue example class; 63

PassRef example class; 64

path

See also: file(s); names/naming

absolute, testing for, `File.isAbsolute`; 545

canonical, term definition; 544

class, term definition; 435

file

retrieving components,
`File.getAbsolutePath`; 543

retrieving components,
`File.getCanonicalPath`; 544

retrieving components, `File.getName`;
543

retrieving components,
`File.getParent`; 544

retrieving components, `File.getPath`;
543

hidden, testing for, `File.isHidden`; 545

representation of, File objects; 543

searching

`File.pathSeparator` use; 547

`File.pathSeparatorChar` use; 547

term definition; 543

pathSeparator field

File class, search string use; 547

pathSeparatorChar field

File class, search string use; 547

Pattern class

CANON_EQ flag; 324

CASE_INSENSITIVE flag; 324

COMMENTS flag; 325

`compile` method; 323, 641

convenience methods that avoid using; 315

DOTALL flag; 324

flags method; 324

LITERAL flag; 325

`matcher` method; 324

`matches` method; 324

MULTILINE flag; 324

non-greedy qualifier use; 650

`pattern` method; 323

`quote` method; 324

regular expression compilation use; 323

Scanner class use; 641

`Scanner.delimiter` use; 644

`split` method; 324

`toString` method; 324

UNICODE_CASE flag; 324

UNIX_LINES flag; 324

pattern method

Pattern class; 323

pattern(s)

design

bibliographic reference; 756

Template Method, abstract class role; 97

factory method, inner class use; 147

including special characters as literals in; 322

matching

character sequence use; 329

efficiency; 329–330

flags that affect; 324

modifying the state of; 325

regular expression use; 321

replacing matched characters; 326–330

replacing patterns; 326

non-greedy qualifier use; 650

producer/consumer; 639

replacing; 326

resetting pattern matchers; 325

Scanner class use; 641

Template Method

thread wait and notification; 354

usage, hashtable design, HashMap class; 591

PatternSyntaxException class; 315, 323

peek method

AbstractQueue class; 612

peek method

Queue interface; 585

Stack class; 619

per-thread behavior

locks; 346

percent (%)

format specifier prefix, examples of; 23

percent percent (%)

outputting percent sign; 625

percent sign (%)

binary remainder operator; 201

format specifier use; 624

operator, term definition; 11

remainder operator, preservation of

symmetry around zero; 658

performance

See also: big *O* notation; design issues and

strategies; optimization

byte stream issues; 505

clarity tradeoffs, in pattern matching; 330

collation issues; 708

extensible class design issue; 109

hashtables, HashMap class; 591

high performance I/O, java.nio use with;

565

high-performance concurrency,

`java.util.concurrent.atomic`

package; 735

regular expression compilation; 323–326

resizable lists, improvement of; 615

strategies, WeakHashMap class; 593

string comparisons, reference vs. contents;

312

synchronized statement advantages for;

349

- perl programming language**
 - Java regular expression compared with; 321
- Permission class**; 680
 - security manager use; 678
- permission(s)**; 679–680
 - term definition; 677
 - thread, group; 376
- Permissions example class**; 62, 134
- phantom**
 - reachable, term definition; 456
 - references, reference queue use with; 460
- PhantomReference class**; 455
 - garbage collector actions; 456
 - resource management use; 463
- PI constant**; 658
- Picasso, Pablo**
 - quotation; 659
- PingPong example class**; 340
- Pipe example class**; 521
- Piped streams**; 520–521
 - connect method; 521
 - stream that defines behavior; 514
- PipedInputStream class**; 520
- PipedOutputStream class**; 520
- PipedReader class**; 520
- PipedWriter class**; 520
- pipes**
 - See also:* buffers; I/O
 - term definition; 520
- Pirsig, Robert**
 - quotation; 479
- Pixel example class**; 25
- placement**
 - absolute, graphical user interfaces, design problems; 718
 - relative, graphical user interfaces, design advantages; 718
- plain font**
 - plain font vs., in doc comments; 485
- platform**
 - See also:* Java
 - dependence, character encoding; 320
 - independent, character set handling; 513
 - Java, overview; 38
- Player example class**; 436
- PlayerLoader example class**; 436, 439
- plus (+)**
 - addition operator, term definition; 6
 - binary addition operator; 201
 - concatenation operator
 - String use; 21
 - term definition; 12
 - floating-point conversion use (table); 628
 - implicit string conversion with; 220
 - integer conversion use (table); 627
 - string concatenation and representation use; 100
 - string concatenation operator; 214
 - unary positive operator; 202
- plus equals (+=)**
 - assignment use; 11
 - concatenation assignment operator, String use; 22
 - implicit string conversion with; 220
- plus plus (++)**
 - increment operator; 205
 - term definition; 11
- Poincare, Henri**
 - quotation; 621
- Point example class**; 12
 - interface implementation; 119
- pointer (file)**
 - retrieving,
 - RandomAccessFile.getFilePointer; 542
 - term definition; 541
- Policy class**; 680–681
- policy(s)**
 - annotation retention; 395, 414
 - security; 680–681
 - term definition; 38, 677
 - synchronization, documentation importance; 353
- poll method**
 - AbstractQueue class; 612
 - BlockingQueue interface; 604
 - Queue interface; 585
 - ReferenceQueue class; 459
- polling**
 - reference queues; 459, 463
 - term definition; 338
- Polygon example class**; 584
- polymorphism**
 - array; 177
 - term definition; 25, 75
- pooling of threads**
 - multithreading design; 345
 - term definition; 384
- pop method**
 - Stack class; 619
- Port example class**; 146
- portability**
 - applet characteristic; 721
 - native method impact on; 74
 - ProcessBuilder class and exec method impact on; 672
- position**
 - indexing list elements by; 473
- POSITIVE_INFINITY constant**; 203
 - Float and Double class use; 166
 - floating-point wrapper classes; 191

POSIX character classes

special characters for (table); 752

possessive quantifiers

special characters for (table); 753

postcondition

term definition; 298

postfix

See also: prefix

expressions, statements; 230

operators; 205

precedence; 221

term definition; 11

pound sign (#)

floating-point conversion use (table); 628

integer conversion use (table); 627

specifying member names with, doc
comments; 483

string conversion use; 628

pow method

Math class and StrictMath class; 658

powers; 658**precedence**

assignment vs. inequality test; 212

operator; 221–223

term definition; 221

operators (table); 750

precision

arbitrary

decimal numbers, BigDecimal class; 724

integer arithmetic, BigInteger class; 722

format specification; 24

combining with width specification; 24

loss of; 217

magnitude vs.; 216

precision indicator

term definition; 625

precondition

term definition; 298

preemption

thread, term definition; 358

time control, thread communication

mechanism use; 359

prefix

See also: postfix

expressions, statements; 230

operators; 205

term definition; 11

package, naming conflict avoidance use; 36

previous method

ListIterator interface; 572

previousIndex method

ListIterator interface; 473

Primes example class; 56**primitive types**

See also: data types

behavior representation, wrapper classes; 183

conversion to wrappers; 183–184

converting to reference types; 91

creating arrays of, Class class use; 429

Field class methods for getting and setting;
419

hierarchy, (figure); 183

Method class use; 422

reflection access of name; 413

representation by Class objects; 399

representation by wrapper classes; 183

(table); 166

term definition; 4

testing if Class object represents,

Class.isPrimitive method; 405

wrapper classes for; 183, 183–200

print(ing)

characters, specifying minimum number; 23

java.awt.print package; 720

streams, viewed as filters; 518

Print streams; 525–527

println method; 525

stream with no input counterpart; 514

Printer example class; 146**printf method**

PrintStream class; 624

PrintWriter class; 624

varargs method; 61

printStackTrace method

Throwable class; 294

println method

overloading of; 6

Print streams; 525

PrintQueue example class; 356**PrintServer example class; 343****PrintServer2 example class; 344****printStackTrace method**

uncaught exception handling use; 380

PrintStream class; 525

Formatter use; 631

printf method; 624

System.out relationships with; 512

PrintWriter class; 525

printf method; 624

priority

aging, term definition; 358

heap, PriorityQueue use; 586

queue, term definition; 94

term definition; 358

thread

management strategies; 359

upper limit, thread group control of; 377

PriorityBlockingQueue class; 606**PriorityQueue class; 586–587**

comparator method; 587

not Cloneable; 570

PriorityQueue example class; 94

private

access modifier; 48
 design decisions in an extensible framework;
 114
 enum constructors, implications; 156
 fields, hiding fields vs.; 88
 impact on accessibility and overrideability;
 88
 package access; 471
 read-only access use; 66
 reserved keyword (table); 165
 term definition; 19

PRIVATE_USE constant

Character class; 195

privileged code

security interaction with; 682
 term definition; 681

PrivilegedAction interface; 683**PrivilegedExceptionAction class; 683****Process class; 667–669**

destroy method; 668
 exitValue method; 668
 facilities of; 661
 getErrorStream method; 667
 getInputStream method; 667
 getOutputStream method; 667
 waitFor method; 668

process(es)

See also: flow of control; multiprocessing
 child, term definition; 667
 creating; 666–672
 environments; 669–670
 key attributes of, `ProcessBuilder`
 encapsulation of; 670
 parent, term definition; 667
 term definition; 666

ProcessBuilder class; 670–671

command method; 671
 directory method; 671
 environment method; 671
 Process class use; 661
 redirectErrorStream method; 671
 start method; 670

processEvent method

EventListener interface; 719

ProcessFile example class; 451**program(s)**

invocation of, main method use; 73
 order, term definition; 371

programming

system; 661–684
 languages
 perl, regular expressions compared; 321

promotion

See also: widening, primitive conversions
 numeric, conversions that apply to; 218

properties

See also: attributes
 classes, modifiers for specifying; 43
 files, resource bundles; 693
 immutable, term definition; 46
 interfaces use for defining; 118
 system; 663
 security; 664
 term definition; 67
Properties class; 620
 getProperty method; 620
 list method; 621
 load method; 621
 propertyNames method; 621
 setProperty method; 621
 store method; 621
 system properties stored in; 663
 uses for; 616

propertyNames method

Properties class; 621

PropertyPermission class; 679**PropertyResourceBundle class; 692–693**

ResourceBundle implementation by; 691

protected

access modifier; 48
 design decisions in an extensible framework;
 113
 method modifier, subclass field access with;
 79
 package access; 471
 reserved keyword (table); 165
 semantics of; 93–95

protection domain

class loader use; 440
 packages creation of; 467
 ProtectionDomain class representation of;
 681
 term definition; 677, 681

ProtectionDomain class; 681

ClassLoader.defineClass use; 440

protocol

serialization stream; 561

providers

term definition; 732

Proxy class; 432–435

getInvocationHandler method; 434
 getProxyClass method; 433, 435
 isProxyClass method; 435
 newProxyInstance method; 433

pseudo-random numbers

Random class; 639–641

public; 15

access modifier; 48
 design decisions in an extensible framework;
 113
 fields, when to use; 111

- interface declaration; 122
- package access; 471
- reserved keyword (table); 165
- term definition; 2, 13, 19, 43
- push method**
 - Stack class; 619
- pushBack method**
 - StreamTokenizer class; 536
- Pushback streams;** 529–531
 - parser building stream; 515
- PushbackInputStream class;** 530
- PushbackReader class;** 530
 - unread methods; 531
- put method**
 - AbstractMap class; 615
 - BlockingQueue interface; 604
 - Hashtable class; 619
 - LinkedHashMap class; 592
 - Map interface; 588
 - WeakHashMap class; 593
- putAll method**
 - LinkedHashMap class; 592
 - Map interface; 588
- putIfAbsent method**
 - ConcurrentMap interface; 606
- putLong method**
 - ByteBuffer class; 565

Q

- qualified this reference**
 - term definition; 138
- querying/query**
 - See:* retrieving; search
 - bit patterns
 - bitCount methods; 190
 - highestOneBit methods; 190
 - lowestOneBit methods; 190
 - numberOfLeadingZeros methods; 190
 - numberOfTrailingZeros methods; 190
 - reverse methods; 190
 - rotateLeft methods; 190
 - rotateRight methods; 190
 - type, at runtime; 414
- question mark (?)**
 - abstract type parameter wildcard; 31
 - generic type wildcard; 257
- question/colon (?:)**
 - operator; 210
- Queue interface;** 585–587
 - element method; 585
 - offer method; 586
 - overview; 569
 - peek method; 585
 - poll method; 585
 - remove method; 585
- queue(s)**
 - See also:* collections; data structures
 - AbstractQueue class; 611
 - ArrayBlockingQueue class; 605
 - BlockingQueue interface; 604–606
 - DelayQueue class; 606
 - empty, null as value that indicates; 586
 - event, term definition; 718
 - LinkedBlockingQueue class; 605
 - LinkedList class; 570
 - LinkedList use; 583
 - priority, term definition; 94
 - PriorityBlockingQueue class; 606
 - Queue interface; 585–587
 - reference; 459–464
 - term definition; 459
 - resource, phantom reference use; 460
 - SynchronousQueue class; 606
 - term definition; 94
- quotations**
 - Bach, Johann Sebastian; 161
 - Bankhead, Tallulah; 685
 - Bernstein, Jeremy; 713
 - Bombeck, Erma; 115
 - Boss Tweed; 396
 - Brecht, Bertolt; 567
 - Butler, Samuel; 397
 - Calvin and Hobbes; 39
 - Carlyle, Thomas; 303
 - Carroll, Lewis; 229
 - census taker; 160
 - Chesterton, G. K.; 465
 - Cook, Rich; 740
 - Darrow, Clarence; 387
 - Davis, Miles; 385
 - Dr. Who; 41, 305
 - Duff, Tom; 749
 - Einstein, Albert; 74
 - Emerson, Ralph Waldo; 445
 - Firesign Theatre; 150
 - Fuller, R. Buckminster; 199
 - Gilbert and Sullivan; 75
 - Goethe, Johann Wolfgang von; 336
 - Gotlieb, Sandra; 467
 - Guaspari, David; 754
 - Guthrie, Woody; 183
 - Hoare, C. A. R.; 132
 - Lehman, John; 683
 - Lincoln, Abraham; 277
 - math professor, U.C. Berkeley; 133
 - Oxford Union Society; 481
 - Parker, Dorothy; 151
 - Picasso, Pablo; 659
 - Pirsig, Robert; 479
 - Poincare, Henri; 621
 - Rickard, Jack; 337

quotations (*cont.*)

Rogers, Will; 623
 Rukeyser, Murial; 498
 Russell, Bertrand; 201
 Sagan, Carl; 181
 Shakespeare, William; 661
 Transportation Commission on Unmet
 Needs, California; 715
 travel agent; 1
 Trillin, Calvin; 228
 Twain, Mark; 447
 U.C. Berkeley Math Professor; 133
 U.S. Army's PS magazine; 279
 Williams, H. H.; 246
 Williams, Peter; 499
 Woolf, Virginia; 566
 Zappa, Frank; 117

quote method

Pattern class; 324

quote(s)

double
 character literal for; 167
 string literal definition with; 3
 string literals use of; 168
 single
 character literal for; 167
 character literals use of; 167
 special characters for (table); 753

quoteChar method

StreamTokenizer class; 534

R**race**

See also: concurrency
 conditions, shutdown hook issues; 673
 hazard
 motivation for atomic lock release on
 wait; 355
 term definition; 338

radians function; 657**radix**

Scanner class, userRadix method; 643
 term definition; 185

random access

access, array advantages; 615
 access files
 reading; 541
 term definition; 542
 writing; 541
 access lists, RandomAccess interface; 584

random numbers

lists, Collections.shuffle; 598
 numbers, Random class; 639–641

Random class; 639–641

Math.random vs.; 659
 nextBoolean method; 640
 nextBytes method; 640
 nextDouble method; 640
 nextFloat method; 640
 nextGaussian method; 640
 nextInt method; 640
 nextLong method; 640

random method

Math class; 659
 StrictMath class; 659

RandomAccess interface; 584–585**RandomAccessFile class; 541–543**

getChannel method; 543, 565
 getFD method; 543
 getFilePointer method; 542
 length method; 542
 seek method; 542
 setLength method; 542
 skipBytes method; 542

randomUUID method

UUID class; 656

range

check, arrays, optimization of, (footnote);
 173
 precision vs.; 217

range method

EnumSet class; 596

raw types

arrays of, reflection method return,
 (footnote); 408
 compatibility design issues; 744–748
 erasure and; 267–272
 names, import use for generic types; 469
 ParameterizedType.getRawType; 427
 purpose and limitations of; 747
 term definition; 30, 250, 267, 745

reachable/reachability

See also: garbage collection
 finalization and; 464
 phantom reachable, term definition; 456
 reference strengths and; 455
 softly reachable, term definition; 456
 states, term definition; 454–465, 454
 strongly reachable, term definition; 456
 term definition; 448
 unreachable, term definition; 456
 weakly reachable, term definition; 456

read method

Buffered input stream; 519
 InputStream; 503, 503
 Readable interface; 642
 Reader; 507, 508

read-only

- access, enforcing; 65
- setting a file to, `File.setReadOnly`; 546
- term definition; 22
- view, collections, wrapped collections; 597

Readable interface

- read method; 642
- Scanner use; 642

readBoolean method

- `DataInput` interface (table); 537

readByte method

- `DataInput` interface (table); 537

readChar method

- `DataInput` interface (table); 537

readDouble method

- `DataInput` interface (table); 537

Reader class; 507

- `close` method; 509
- filter use; 517
- `InputStream` contrasted with; 509
- read methods; 507, 508
- ready method; 509
- `skip` method; 509
- synchronization strategy; 515

readers

- term definition; 500

readExternal method

- `Externalizable` interface; 562

readFloat method

- `DataInput` interface (table); 537

readFully method

- `DataInput` interface; 537

reading

- See also:* I/O; input; output; writing
- booleans, `DataInput.readBoolean`; 537
- bytes
 - `DataInput.readByte`; 537
 - written by `DataOutput.writeBytes`, strategy for; 538
- bytes of data; 503
- characters; 507, 508
 - `CharArrayReader` class; 523
 - written by `DataOutput.writeChars`, strategy for; 538
- files
 - byte, `FileInputStream` class; 541
 - character, `FileReader` class; 541
- floating-point numbers
 - `DataInput.readDouble`; 537
 - `DataInput.readFloat`; 537
- formatted data, Scanner class use; 641
- integers
 - `DataInput.readInt`; 537
 - `DataInput.readLong`; 537
 - `DataInput.readShort`; 537
- random access files; 541

- strings, `StringReader` class; 523
- text, tracking lines while,
 - `LineNumberReader` class; 527

readInt method

- `DataInput` interface (table); 537

readLine method

- `BufferedReader` class; 519

readLong method

- `DataInput` interface (table); 537

readObject method

- `HashMap` class; 554
- `ObjectInputStream` class; 553

ReadPastEndException example class; 295**readResolve method**

- `Object` class; 557

readShort method

- `DataInput` interface (table); 537

readUnshared method

- `ObjectInputStream` class; 551, 560

readUnsignedByte method

- `DataInput` interface; 538

readUnsignedShort method

- `DataInput` interface; 538

readUTF method

- `DataInput` interface (table); 537

ReadWriteLock interface; 735**ready method**

- `Reader`; 509

reaper threads

- resource manager use; 463

ReaperThread example class; 463**receiver object**

- term definition; 16

recovery

- resource, keys use; 460

Rectangle example class; 559**recursion**

- building strings with; 59
- infinite, annotation type restrictions to prevent, (footnote); 391
- reflection use of; 404

redefining

- members, extended classes; 84–90

redirectErrorStream method

- `ProcessBuilder` class; 671

ReentrantLock class; 734

- Lock interface implementation; 735

Reference class; 455, 738

- `clear` method; 455
- `enqueue` method; 455
- `get` method; 455
- `isEnqueued` method; 455

reference(s)

- See also:* garbage collection; object(s)
- ambiguous, multiply-inherited interface constant issue; 124

- caching, soft references use for; 457
 - chain, term definition; 447
 - comparison, content comparison vs.; 311
 - counting, garbage collection model; 449
 - cross
 - doc comments, @link inline tag; 484
 - doc comments, @linkplain inline tag; 484
 - doc comments, @see tag; 484
 - dangling, term definition; 448
 - documentation, doc comments use for; 481
 - equality
 - hashtable impact; 101
 - testing; 101
 - value equality vs.; 101
 - field values, default; 45
 - garbage collection relationship to; 447
 - interface type
 - powerful design tool; 120
 - restrictions on; 120
 - literals; 167
 - object(s); 454–465
 - binding referent object to; 455
 - method invocation use; 3, 58
 - method parameter use; 68–69
 - object types vs.; 26
 - parameter passing implications; 63–64
 - term definition; 455
 - vs.; 42
 - obtaining, to current Runtime object; 453
 - phantom, reference queue use with; 460
 - qualified super, term definition; 138
 - qualified this, term definition; 138
 - queues; 459–464
 - term definition; 459
 - relative, doc comments, @docRoot inline tag; 488
 - root, term definition; 447
 - self, this use for; 17
 - static members; 8
 - strengths, reachability and; 455
 - string, comparing, String.intern use; 312
 - this
 - self referencing use; 17
 - super reference compared with; 25
 - thread, ThreadGroup storage of; 344
 - types; 166
 - compatibility issues; 90
 - converting primitive types to; 91
 - equality operator use; 206
 - static member access through; 89
 - term definition; 13
 - variables, object creation use; 49
- weak
 - uses for, DataHandler example class; 458
 - WeakHashMap class; 570
 - WeakHashMap use; 593
- ReferenceQueue class**
 - poll method; 459
 - remove method; 459
 - SoftReference and WeakReference constructor use; 457
- referent**
 - binding to reference object; 455
 - term definition; 455
- reflection; 397–446**
 - access checking, AccessibleObject class; 417
 - annotation queries; 414–416
 - arrays; 429–432
 - dynamic, creating; 430
 - generic; 428
 - assertions, controlling at runtime; 444–445
 - class
 - inspection; 402–408
 - loading; 435–444
 - Class class; 399
 - custom serialization use; 556
 - Field class; 418
 - final field manipulation by; 420
 - generic
 - arrays; 428
 - type inspection; 426–429
 - JavaBeans use of; 722
 - loading, classes; 435–444
 - Method class; 420
 - Modifier class; 416
 - modifier queries; 416
 - nested class instance use, handling names with dollar signs in them; 149
 - packages; 432
 - parameterized types; 427
 - private method access during serialization, (footnote); 554
 - Proxy class; 432–435
 - runtime type queries; 414
 - security
 - access in; 417
 - use by, with Class class; 409
 - term definition; 397
 - types
 - generic, inspection; 426–429
 - Package objects different from; 477
 - parameterized; 427
 - runtime, queries; 414
 - string representation of; 428–429
 - wildcards; 428
- ReflectPermission class; 679**

- region method**
 - Matcher class; 329
- regionEnd method**
 - Matcher class; 329
- RegionMatch example class**; 310
- regionMatches method**
 - String class; 310
- regions**
 - character sequence, pattern matching use; 329
 - strings, comparing; 310
 - synchronization of, synchronized statement advantages for; 349
 - term definition; 329
- regionStart method**
 - Matcher class; 329
- registering**
 - reference objects with reference queues; 459
- registerValidation method**
 - ObjectInputValidation interface; 557
- registry**
 - RMI; 731
- regular expression(s)**
 - See also:* CharSequence interface; java.util.regex package; Matcher class; Pattern class; pattern(s)
 - capturing parts of a string with; 322
 - compiling; 323–326
 - further reading; 758
 - matching; 321–329
 - performance vs. clarity tradeoffs; 330
 - Scanner class use; 641
 - scanning lines with, Scanner class; 644
 - special characters for (table); 751–753
 - String methods that compare strings using; 314
 - strings and; 305–336
 - term definition; 321
- reification**
 - generics, compatibility design issues; 744–748
- relational**
 - databases, java.sql package; 732–733
- relative**
 - placement
 - graphical user interfaces, design advantages; 718
 - term definition; 718
 - references, doc documents, @docRoot inline tag; 488
- release**
 - resource, guaranteed, shutdown hooks required; 464
- releasing**
 - locks; 346
 - resources, finally use for; 289
- reluctant quantifiers**
 - special characters for (table); 753
- remainder**
 - floating-point; 203
 - operator (%); 201
 - term definition; 11
- remainingCapacity method**
 - BlockingQueue interface; 605
- Remote interface**; 727
 - marker interface; 130
- remote interface**
 - term definition; 727
- Remote Method Invocation (RMI)**
 - CORBA compared with; 740
 - java.rmi.activation package; 731
 - java.rmi package; 716, 727
 - RMI Security Manager class; 730
 - thread use; 370
- RemoteException class**; 727
 - Remote class use; 727
- remove method**
 - AbstractMap class; 615
 - AbstractSequentialList class; 615
 - Collection interface; 576
 - ConcurrentMap interface; 607
 - Hashtable class; 619
 - Iterator interface; 143, 571
 - List interface; 581
 - ListIterator interface; 473
 - Map interface; 588
 - Queue interface; 585
 - ReferenceQueue class; 459
 - Scanner class; 642
 - WeakHashMap class; 593
- removeAll method**
 - Collection interface; 577
- removeAllElements method**
 - Vector class; 618
- removeElement method**
 - Vector class; 618
- removeElementAt method**
 - Vector class; 617
- removeFirst method**
 - LinkedList class; 583
- removeLast method**
 - LinkedList class; 583
- removeRange method**
 - AbstractList class; 615
- removeShutdownHook method**
 - Runtime class; 673
- removing**
 - See:* deleting
- renameTo method**
 - File class; 545
- renaming**
 - files, File.renameTo; 545

- repaint method**
 - Component class; 718
- replace method**
 - ConcurrentMap interface; 607
 - String class; 314
 - StringBuilder class; 333
- replaceAll method**
 - Collections class; 598
 - Matcher class; 327
 - String class; 314
- replaceFirst method**
 - Matcher class; 327
 - String class; 314
- replacing**
 - See also:* modifying/modification
 - array elements, synchronized use for access control; 349
 - matched characters; 326
 - numbers, pattern matching use; 328
 - patterns; 326
- representation**
 - See also:* semantics
 - bit patterns, toBinaryString method; 189
 - boolean primitive type; 189
 - calendars; 695
 - data, Unicode, ASCII, and ISO Latin-1 use; 9
 - dates; 695
 - fields, in serialized data; 560
 - hexadecimal numbers, toHexString method; 190
 - lists, Iterable interface use; 129
 - no return type, Void class use; 187
 - object state, toString use; 59
 - octal numbers, toOctalString method; 189
 - string
 - an object, Object.toString; 100
 - binary numbers; 189
 - octal numbers; 189
 - toBinaryString method, Integer class; 189
 - toBinaryString method, Long class; 189
 - toHexString method, Integer class; 190
 - toHexString method, Long class; 190
 - toOctalString method, Integer class; 189
 - toOctalString method, Long class; 189
 - toString method, Integer class; 189
 - toString method, Long class; 189
 - toString method, wrapper classes; 187
 - of type objects; 428–429
 - time; 695
 - types; 183, 411
- rescheduling**
 - voluntary, threads; 360–362
- reset method**
 - ByteArrayOutputStream class; 522
 - CharArrayWriter class; 523
 - Matcher class; 325
- resetSyntax method**
 - StreamTokenizer class; 535
- resetting**
 - pattern matchers; 325
 - syntax, StreamTokenizer class; 535
- resizable**
 - bit vectors, resizable, BitSet class; 633
 - lists
 - creating; 612
 - improving performance of; 615
- resolution**
 - method
 - "most specific" algorithm; 224–228
 - "most specific" algorithm, generic type modifications; 272–276
 - methods; 223
- resolveClass method**
 - ClassLoader class; 441
- Resource example interface; 460**
- resource(s)**
 - bundles; 688–693
 - localization tool; 686
 - term definition; 689
 - garbage collection ordering issues; 452
 - guaranteed release, shutdown hooks required for; 464
 - keys use for recovery of; 460
 - loading; 442–444
 - management, reference queue and key use; 460–464
 - ordering, term definition; 364
 - release of, finally use for; 289
 - searching for, internationalization code; 690
 - system, loading; 442
 - system-wide, System class as repository for; 662
- ResourceBundle class**
 - getBundle method; 690
 - getKeys method; 689, 693
 - getObject method; 689
 - getString method; 688
 - getStringArray method; 688
 - handleGetObject method; 693
 - subclassing; 693
- ResourceImpl example class; 461**
- ResourceManager example class; 462**
- restricting**
 - behavior, subclasses; 25
- resurrecting**
 - objects during finalization, design issues; 452

retainAll method

Collection interface; 577
 BlockingQueue interface use; 605

retention

annotation, policies for; 395, 414

Retention class

as meta-annotation; 396

RetentionPolicy class; 395, 414**retrieving**

See also: accessing

arrays of collection elements,

Collection.toArray; 575

bytes, String.getBytes; 319

character value for digit,

Character.forDigit; 193

characters

from strings, CharSequence.charAt
 use; 305

String.getChars; 318

delimited strings; 313

file

channel,

RandomAccessFile.getChannel;
 543

descriptor, RandomAccessFile.getFD;
 543

last modified time, File.lastModified;
 545

length, File.length; 545

length, RandomAccessFile.length; 542

pointer,

RandomAccessFile.getFilePointer;
 542

file pathname components

File.getAbsolutePath; 543

File.getCanonicalPath; 544

File.getName; 543

File.getParent; 544

File.getPath; 543

first occurrence of a character or a substring

in a string, String.indexOf; 308

last occurrence of a character or substring in

a string, String.lastIndexOf; 307

length, strings, CharSequence.length use;
 306

number of bytes available; 504

numeric value of character

Character.digit; 193

Character.getNumericValue; 193

pattern match information

capturing groups; 326

index of last character; 326

input subsequence; 326

start index; 326

return

covariant, term definition; 102

reserved keyword (table); 165

run method, thread termination as result of;
 364

statement; 245

assignment compatibility requirement; 91

finally use for cleanup; 290

method execution termination by; 62

switch terminator; 235

term definition; 16

type

constructors do not use; 50

covariant, term definition; 85

interface inheritance; 125

lack of, Void class representation of; 187

method declaration use; 2

methods; 59

not included in signature; 70

overriding requirements; 85

required for methods; 57

value

doc comment tag for; 485

methods; 62

nested class use for returning multiple
 values; 148

requirements; 63

term definition; 16

void use to indicate none; 15

values, upper bounded wildcard use; 258

reusability

name conflict impact on; 36

Reuse example class; 179**reverse method**

Collections class; 598

Integer class; 190

Long class; 190

StringBuilder class; 333

reverse ordering

characters, a string buffer; 333

Collections.reverse; 598

Collections.reverseOrder; 597

reverseBytes method

Integer class; 190

Long class; 190

Short class; 190

reverseOrder method

Collections class; 597, 598

reversing

bytes, reverseBytes method s; 190

RFC 222

java.security.sasl package; 732

Rickard, Jack

quotation; 337

rint method

Math class and StrictMath class; 658

RMI (Remote Method Invocation)

CORBA compared with; 740
 java.rmi.activation package; 731
 java.rmi package; 716, 727
 RMISecurityManager class; 730
 thread use; 370

RMISecurityManager class; 730**Rogers, Will**

quotation; 623, 748

roles

multiple, impact on class design; 108

roll method

Calendar class; 698

root

class hierarchy, Object class as; 99–101
 directories, retrieving, File.list; 546
 reference, term definition; 447

rotate method

Collections class; 598

rotateLeft method

Integer class; 190
 Long class; 190

rotateRight method

Integer class; 190
 Long class; 190

round method

Math class and StrictMath class; 658

RoundingMode class; 722**RSA keys**

java.security.interfaces package; 732

Rukeyser, Murial

quotation; 498

rules

including in doc-files documentation
 directory; 497

run method

Runnable interface; 341
 Thread class; 339
 completion, thread termination as result
 of; 364
 TimerTask class; 654

runFinalization method

Runtime class; 453, 676
 System class; 453, 666

RUNNABLE constant

Thread class, State nested enum; 384

Runnable interface

flexibility of; 345 xx
 property-definition interface; 118
 run method; 341
 Thread constructors that specify; 342
 thread creation use; 340
 threads use of; 341

RunPingPong example class; 342**runtime**

annotations, AnnotatedElement methods
 access; 414
 assertion control; 444–445
 assertions, turning on and off; 300–303
 checking, pitfalls when forbidding cloning;
 103
 creating classes at; 432–435
 erasure impact at; 267–270
 exceptions
 @throws tag inheritance; 491
 unchecked exceptions; 284
 generic type information erased; 250
 loading classes; 435–444
 method resolution; 228
 security
 permissions affecting; 680
 shutting down; 672
 system
 facilities for interacting with; 661–684
 term definition; 38
 type queries at; 414

Runtime class

addShutdownHook method; 673
 availableProcessors method; 676
 multiprocessor thread management use;
 360
 exec method; 666
 environment variable passing version; 669
 portability issues; 672
 working directory specification version;
 670
 exit method; 672
 application execution termination with;
 369
 conditions that require use of; 674
 shutdown initiated by; 674
 facilities of; 661
 freeMemory method; 453
 functionality of; 675–677
 gc method; 452, 453, 676
 getRuntime method; 453, 666
 halt method; 674
 loadLibrary method; 676
 removeShutdownHook method; 673
 runFinalization method; 453, 676
 totalMemory method; 453
 traceInstructions method; 677
 traceMethodCalls method; 677

RUNTIME constant
 RetentionPolicy class; 395, 414

RuntimeException class
 runtime exceptions extensions of; 284
 in type hierarchy, (figure); 280
 unchecked exceptions; 34
 extension of; 280

RuntimePermission class; 679

Russell, Bertrand

quotations; 201

S

s, S format specifier

string conversion use; 629

safety

native method impact on; 74
safe casting, term definition; 92
thread, term definition; 353
type

enum advantage; 152

enum use; 8

Sagan, Carl

quotation; 181

sampled sound

javax.sound.sampled package; 739

sandbox

term definition; 720

SASL (Simple Authentication and Security Layer)

java.security.sasl package; 732

Scanner class; 641–641

close method; 644

delimiter method; 644

findInLine method; 644–645

findWithinHorizon method; 646

hasNext method; 642

hasNextDouble method; 642

localization support; 651

hasNextLine method; 645

java.util package, formatted input; 501

locale method; 651

localization support; 651

match method; 644

next method; 642

nextDouble method; 642

nextInt method, localization support; 651

nextLine method; 645

remove method; 642

skip method; 646

text scanning and conversion; 532

useLocale method; 651

userDelimiter method; 643

userRadix method; 643

using; 647–650

scanning

See also: parsing

lexical

elements; 161

Pushback stream use; 529

lines, Scanner class; 644–647

schedule method

Timer class; 655

scheduleAtFixedRate method

Timer class; 655

scheduledExecutionTime method

TimerTask class; 654

ScheduledThreadPoolExecutor class

Executor interface implementation; 734

scheduling

See also: concurrency; time

tasks

fixed-delay, term definition; 655

fixed-rate, term definition; 655

once-only, term definition; 655

with Timer and TimerTask; 653

threads; 358–362

Thread.yield impact on; 360

voluntary rescheduling; 360–362

scientific notation

(e, E) format specifier, floating-point
conversion use; 627

format specifier; 23

(g, G) format specifier, floating-point
conversion use; 628

scope/scoping

See also: namespaces; nested/nesting
algorithm use in determining the semantics of
a name; 180

enclosing class, relative to inner classes; 138

inner classes; 140–142

local, local variables compared to; 142

namespace management with; 178

parameters; 180

purpose of; 181

term definition; 140, 179

types

nested classes and interfaces; 133

search order, when determining semantics

of a name; 181

static nested; 134

variables

local; 171, 180

loop; 180

parameter; 171

scratch files

creating, File.createTempFile; 547

sealed packages

term definition; 478

search method

Stack class; 619

search/searching

See also: access/accessibility

arrays, Arrays.binarySearch; 608

binary, algorithm; 310

character sequences, regular expression use;
321

Comparable objects; 574

Comparator objects; 574

search/searching (*cont.*)

- lists, `Collections.binarySearch`; 599
- name scopes, algorithm for the semantics of a name; 180
- order, type scopes, when determining semantics of a name; 181
- path
 - `File.pathSeparator` use; 547
 - `File.pathSeparatorChar` use; 547
- resources, internationalization code; 690
- string, invalid index handling; 311

security; 677–683

- access, reflection; 417
- access control
 - `java.security.acl` package use; 732
 - security manager actions; 678
- applet, environment; 721
- checks
 - file access, `File` streams; 541
 - file access, `RandomAccessFile` class; 542
 - `File` class methods; 545
- class loader; 438
- design issue
 - extensible class design; 109
 - extensible framework design; 114
- domain, defining, thread groups use for; 375–379
- domain protection, class loader use; 440
- `Externalizable` interface issues; 562
- final method advantages; 96, 111
- impact on reflection for custom serialization; 556
- manager
 - `RMI SecurityManager` class; 730
 - setting of; 678
 - term definition; 38, 677
 - thread monitoring by; 376
- model, **JVM**, advantages; 730
- native code loading; 676
- policy; 680–681
 - term definition; 38, 677
 - thread group modification control; 375
- privileged code, interaction with; 682
- process creation and; 667
- `ProcessBuilder` class and `exec` method impact on; 672
- reflection use of, with `Class` class; 409
- runtime
 - permissions affecting; 680
 - shutting down; 672
- serialization, (footnote); 554
- standard I/O streams, setting of; 662
- system properties; 664
- thread group, modification; 376

threads

- behavior that impacts; 376
- creation; 376
- managing, with `ThreadGroup` class; 375–379
- modification; 376

Security class; 732**SecurityException class**; 376

- `Class` class use; 409
- `Class.newInstance` use; 424
- `ClassLoader` class use; 438, 440
- file accessing checks
 - `File` streams; 541
 - `RandomAccessFile` class; 542
- `File` methods that can throw; 545
- native code library loading use; 676
- shutdown use; 672
- `System` class
 - process creation use; 664
 - system property methods use; 664
 - use; 662

SecurityManager class; 678–679

- `checkPermission` method; 678
- `getSecurityContext` method; 679

SecurityPermission class; 679

See also: design issues and strategies

seek method

- `RandomAccessFile` class; 542

Self language

- bibliographic reference; 759

self referencing

- with the `this` reference; 17

semantics

See also: representation

- annotations potential misuse; 396
- application, impact on `IsA` vs. `HasA` relationships; 107–108
- contract component; 41
- final fields; 46
- `HelloWorld.java`; 2
- multiple inheritance, implementation vs. interface; 123
- names; 178–181
- overriding, interface method inheritance; 125
- protected keyword; 93–95
- statement, impact on method execution; 62

Semaphore class; 733**semicolon (;)**

- empty statement indication; 34, 229
- statement terminator; 3, 229

sentences

- parsing text into, with `BreakIterator` class; 712

SeparateGroups example class; 350

separator

- field, `String` class, path component separator stored in; 543
- line
 - `newLine` method, `BufferedWriter` handling; 519
 - `println` method use; 525
- path component, `File.separatorChar` use; 543
- terminator vs., (footnote); 229

separatorChar field

- `File` class, path component separator stored in; 543

SequenceCount example class; 530**SequenceInputStream class; 528–529****SequenceInputStream stream**

- stream with no output counterpart; 514

sequences/sequential

- access, `AbstractSequentialList` use; 615
- byte streams, `SequenceInputStream` class; 528
- character; 305–306
 - regions, in pattern matching; 329
 - wildcard for matching beginning or end of; 322
- characters, `String` manipulation of; 21
- converting to arrays, design concerns; 85
- escape, encoding Unicode characters with; 162
- execution, blocks as control mechanism; 9
- of parameters, syntax for; 58, 60

serial version UID

- serialization use; 557

Serializable interface; 123, 551

- arrays implementation of; 177
- enums implicitly; 153
- `Externalizable` interface extension of; 561
- marker interface; 130
- as marker interface; 550
- property-definition interface; 118

SerializablePermission class; 679**SerializableRunnable example interface; 123****serialization**

- See also:* streams
- customized; 554
- default; 551
- documentation comment tags; 562–563
- `Externalizable` interface; 562
- fields; 559–561
- hash maps; 550
- making classes serializable; 551
- new superclass handling; 559
- object; 549–563
 - cloning compared with; 106
 - term definition; 501

- object versioning issues; 557–559
- order; 552–553
- security, (footnote); 554
- stream protocol; 561
- term definition; 549

SerialPort example class; 146**server(s)**

- activatable, `java.rmi.activation` package; 732
- server-side synchronization, client-side synchronization vs.; 352
- term definition; 727

ServerSocket class; 724**Service Provider Interface (SPI)**

- `java.naming.spi` package; 739

Set interface; 577

- `CopyOnWriteArraySet` class; 607
- `HashSet` implementation; 579
- overview; 569

set method

- `AbstractSequentialList` class; 615
- `Array` class; 430
- `BitSet` class; 633
- `Calendar` class; 698, 699
- defining object properties with; 67
- `Field` class; 418
- `List` interface; 581
- `ListIterator` interface; 473

set(s)

- See also:* collection(s); data structures
- `AbstractSet` class, implementation starting point; 611
- `CopyOnWriteArraySet` class; 607
- `emptySet`, `Collections` class; 600
- `EnumSet` class; 594–596
- `EnumSet` enum specific class; 160
- `HashSet` class; 569, 579
- `Set` interface; 569, 577
- singleton, creating, `Collections.singleton`; 600
- `SortedSet` interface; 569
- term definition; 577
- `TreeSet` class; 569, 579
- wildcard characters for; 322

setAccessible method

- `AccessibleObject` class; 417

setBeginIndex method

- `FieldPosition` class; 705

setBit method

- `BigInteger` class; 722

setChanged method

- `Observable` class; 636

setCharAt method

- `StringBuilder` class; 331

setClassAssertionStatus method

- `ClassLoader` class; 444

- setContextClassLoader** method
 - Thread class; 438
- setDaemon** method
 - Thread class; 369
 - ThreadGroup class; 378
- setDefault** method
 - Locale class; 687
 - TimeZone class; 700
- setDefaultAssertionStatus** method
 - ClassLoader class; 444
- setElementAt** method
 - Vector class; 617
- setEndIndex** method
 - FieldPosition class; 705
- setErr** method
 - System class; 662
- setFirstDayOfWeek** method
 - Calendar class; 699
- setGregorianChange** method
 - GregorianCalendar class; 702
- setID** method
 - TimeZone class; 700
- setIn** method
 - System class; 662
- setLastModified** method
 - File class; 546
- setLength** method
 - RandomAccessFile class; 542
 - StringBuilder class; 332
- setLenient** method
 - Calendar class; 699
 - DateFormat class; 705
- setLineNumber** method
 - LineNumberReader class; 528
- setMaxPriority** method
 - ThreadGroup class; 377, 378
- setMinimalDaysInFirstWeek** method
 - Calendar class; 699
- setName** method
 - Thread class; 341
- setNumberFormat** method
 - DateFormat class; 704
- setOut** method
 - System class; 662
- setPackageAssertionStatus** method
 - ClassLoader class; 444
- setPriority** method
 - Thread class; 359
- setProperty** method
 - System class; 664
- setProperty** method
 - Properties class; 621
 - System class; 664
- setRawOffset** method
 - TimeZone class; 700
- setReadOnly** method
 - File class; 546
- "setSecurityManager"**
 - RuntimePermission name; 680
- setSecurityManager** method
 - System class; 678
- setSeed** method
 - Random class; 640
- setSize** method
 - Vector class; 618
- setStackTrace** method
 - StackTraceElement class; 294
- setTime** method
 - Calendar class; 699
 - Date class; 695
- setting**
 - file, length,
 - RandomAccessFile.setLength; 542
- setUncaughtExceptionHandler** method
 - Thread class; 380, 381
- setValue** method
 - Entry interface; 589
- shadowing**
 - term definition, (footnote); 180
- Shakespeare, William**
 - quotation; 661
- shallow cloning**
 - deep cloning vs.; 106
 - term definition; 106
- shared**
 - arrays, client-side synchronization use; 349
 - data, interface design that includes; 121
 - memory, values in, multiprocessing
 - hardware impact on; 370
 - state, interfaces, inner class use; 149
- shift operators; 209**
 - precedence; 221–222
- Short class; 188–191**
 - converting, strings to/from (table); 316
 - parseShort method; 316
 - reverseBytes method; 190
 - wrapper class, short primitive type; 166
 - wrapper type hierarchy, (figure); 183
- short data type**
 - See also:* data types
 - arrays, name notation; 412
 - boxing conversion range; 199
 - converting to/from strings (table); 316
 - definition; 4
 - field values, default; 45
 - integer literals; 168
 - reading, `DataInput.readShort`; 537
 - reserved keyword; 165
 - `shortValue` method, `Number` class; 188
 - value of; 166
 - writing, `DataInput.writeShort`; 537

- ShortStrings example class**; 609
- shortValue method**
 - Number class; 188
- ShowJoin example class**; 368
- shuffle method**
 - Collections class; 597, 598
- shuffling lists**
 - Collections.shuffle; 597, 598
- shutdown**; 672–675
 - application execution; 369
 - hooks; 672–673
 - term definition; 672
 - sequence; 674
 - strategies; 674
- side effect(s)**
 - assertion evaluation dangers; 296
 - assignment during expression evaluation; 212
 - expression evaluation order impact on; 215
- SIGKILL signal**
 - virtual machine abort cause; 674
- signal method**
 - Condition interface; 735
- signalAll method**
 - Condition interface; 735
- signaling**
 - errors, exceptions use for; 279
- signatures**
 - See also:* contract; overloading; security
 - overloading vs. overriding; 84
 - override-equivalent, term definition; 271
 - overriding requirements; 85
 - term definition; 2, 57, 69
- signed**
 - integer types are all; 166
- signum**; 658
- signum method**
 - Integer class; 191
 - Long class; 191
 - Math class and StrictMath class; 658
- Simple Authentication and Security Layer (SASL)**
 - java.security.sasl package; 732
- simple name**
 - term definition; 411
- SimpleClassDesc example class**; 398
- SimpleDateFormat class**; 696
- SimpleLookup example class**; 28
- SimpleSortDouble example class**; 112
- SimpleTimeZone class**; 696, 701–703
 - TimeZone subclass; 703
- SIMULA language**
 - bibliographic reference; 759
- sin method**
 - Math class and StrictMath class; 657
- sine**; 657
 - hyperbolic; 658
- single inheritance**
 - multiple inheritance vs.; 114
- single quote (')**
 - character literal; 167
- single type import**
 - term definition; 469
- single-threaded**
 - programming model, term definition; 337
 - systems, polling use; 338
- SingleLinkedList example class**; 94
 - dynamic array creation; 430
 - generic declaration; 248
 - nested generic types; 253
 - PrintQueue class use; 356
- singleton method**
 - Collections class; 600
- singletonList method**
 - Collections class; 600
- singletonMap method**
 - Collections class; 600
- sinh method**
 - Math class and StrictMath class; 658
- size**
 - string buffer, managing the; 335
- SIZE field**
 - wrapper classes; 186
- size method**
 - AbstractList class; 612
 - AbstractSequentialList class; 615
 - BitSet class; 634
 - ByteArrayOutputStream class; 522
 - CharArrayWriter class; 523
 - Collection interface; 575
 - Hashtable class; 619
 - Map interface; 588
 - WeakHashMap class; 593
- skip method**
 - InputStream; 503
 - Reader; 509
 - Scanner class; 646
- skipBytes method**
 - DataInput interface; 538
 - RandomAccessFile class; 542
- skipping**
 - bytes, a file,
 - RandomAccessFile.skipBytes; 542
 - bytes of data; 503
 - characters; 509
 - comments, StreamTokenizer vs. Scanner use; 648
- slash (/)**
 - binary division operator; 201
 - division operator, term definition; 6
 - time zone string identifier use; 700

- slash asterisk (/*)**
 - comment delimiter; 163
- slash asterisk asterisk (/**)**
 - comment delimiter; 163
- slash equals (/=)**
 - assignment use; 11
- slash slash (//)**
 - comment delimiter; 163
 - single-line comments; 163
- slash-asterisk (/*)**
 - comment delimiter, term definition; 6
- slashSlashComments method**
 - StreamTokenizer class; 536
- slashStarComments method**
 - StreamTokenizer class; 536
- sleep method**
 - Thread class; 360
 - InterruptedException thrown by; 366
- slicing**
 - time, term definition; 358
- smallest element**
 - retrieving, with Collections.min; 575, 597
- Smalltalk-80 language**
 - bibliographic reference; 759
- snapshot**
 - calling context, retrieving,
 - AccessController.getContext; 681
 - creating; 573
 - copying view; 578
 - not guaranteed for Iterator and
 - ListIterator objects; 473
 - StringTokenizer enumeration as; 651
 - view compared with; 578
- snapshotIterator method**
 - Comparable interface; 574
- Socket class; 724**
 - getChannel method; 565
- SocketPermission class; 679**
- sockets**
 - accessing channels for, getChannel method; 565
 - java.net package; 724
- softly reachable/soft references**
 - caching use; 457
 - term definition; 456
- SoftReference class; 455**
 - constructor use of ReferenceQueue objects; 457
 - garbage collector actions; 456
- sort(ing)**
 - See also:* ordering
 - algorithms, benchmark harness design; 109–114
 - arrays, Arrays.sort; 608
 - collections, SortedSet interface; 577
 - Comparable and Comparator interface use; 574
 - Comparable objects; 574
 - Comparator objects; 574
 - generic type use; 252
 - lists, Collections.sort; 599
 - order, differences between floating-point wrappers and primitives; 191
 - PriorityQueue ordering differences; 586
 - SortedMap interface; 589
 - strings, String.compareTo; 309
- sort method**
 - Arrays class; 608
 - Collections class; 574, 599
- SortDouble example class; 109**
 - reflection use; 423
- SortedCharSeqCollection example class; 253**
- SortedCollection example interface; 253**
- SortedMap interface; 587–590**
 - comparator method; 590
 - firstKey method; 590
 - headMap method; 590
 - instance returned by
 - Charset.availableCharsets method; 321
 - lastKey method; 590
 - Map interface compared with; 589
 - overview; 569
 - subMap method; 590
 - tailMap method; 590
 - TreeMap implementation of; 593
- SortedSet interface; 577**
 - comparator method; 577
 - first method; 577
 - headSet method; 578
 - last method; 578
 - overview; 569
 - SortedMap interface compared with; 590
 - subSet method; 578
 - tailSet method; 578
 - TreeSet implementation; 579
- Sorter example class; 120**
- SortMetrics example class; 111**
- sound**
 - javax.sound package; 739
 - javax.sound.sampled package; 739
- source**
 - Java, versions, characteristics; 742
- SOURCE constant**
 - RetentionPolicy class; 395
- source(s)**
 - characters, for Scanner class use, Readable objects; 642
 - input, Scanner; 643
 - term definition; 499

space

- character, testing for, `Character.isSpaceChar`; 194

SPACE_SEPARATOR constant

- `Character` class; 195

special characters

- See also:* regular expressions; wildcards
- term definition; 532

specialization

- class extension use for; 75
- expansion vs., class extension; 94
- term definition; 75

specification(s)

- package implementation of; 476
- reference documentation vs.; 481
- version numbers, `Package` methods
 - accessing; 478

specifiers (format)

- format; 626
- integer conversions; 626, 627–629
- term definition; 624

SPI (Service Provider Interface)

- `java.naming.spi` package; 739
- `javax.sound.midi.spi` package; 739
- `javax.sound.sampled.spi` package; 739

split method

- `Pattern` class; 324
- `String` class; 314

spurious wakeups

- term definition; 358

sqrt method

- `Math` class; 16, 658
- `StrictMath` class; 658

square brackets ([])

- array
 - declaration use; 173
 - syntax use; 2, 19
- wildcard, sets; 322

square root; 658**Stack class; 619**

- empty method; 619
- peek method; 619
- pop method; 619
- push method; 619
- search method; 619

stacks

- See also:* data structures
- `ArrayList` use; 583
 - `Stack` class replacement; 619
- invocation, exception handling use; 279
- `Queue` use; 585
- `Stack` class; 619
- traces; 294, 382

StackTraceElement class; 294, 382**standard(s)**

- character encoding (table); 754
- I/O streams; 662
- packages; 715–740
- standard extensions mechanism, term
 - definition; 681
- streams, character streams relationships with; 511

start(ing)

- string, testing, `String.startsWith`; 311
- threads, synchronization actions; 372

start method

- `Applet` class; 720
- `MatchResult` interface; 326
- `ProcessBuilder` class; 670
- `Thread` class; 339

START_PUNCTUATION constant

- `Character` class; 195

start-of-line (^) boundary marker

- multi-line input use; 646

startsWith method

- `String` class; 311

starvation

- priority aging use for prevention of; 358

State enum

- `Thread` class; 384

state(s)

- application, printing, as debugging aid; 385
- assertion testing of; 297–299
- component of classes; 343
- corruption of, `Thread.stop` danger; 381
- file
 - changing, `File.setLastModified`; 546
 - changing, `File.setReadOnly`; 546
- initial, creating; 49
- internal, cleanup, `finally` use for; 289
- manipulation of, methods as mechanism for; 57
- object
 - copying, as cloning method objective; 101
 - string representation of; 59
 - thread vs., interruptions as strategy for handling; 367
- pattern matching, modifying; 325
- reachable; 454–465, 454
- serialization of; 552
- setting and getting values of; 67
- shared, interfaces, inner class use; 149
- system, `System` class methods for
 - manipulating; 662
- term definition; 1
- thread group, printing, as debugging aid; 385
- variables, initializing; 80
- virtual machine, impact on reachability; 464

- statement(s)**; 229–230
 - assert; 297
 - block; 230
 - break; 241–244
 - exiting switch with; 234
 - continue; 244
 - loop use; 239
 - declaration, term definition; 230
 - do-while; 9, 235
 - empty
 - semicolon indication of; 34
 - semicolon use as; 229
 - expression, term definition; 229
 - expressions, that can be made into; 230
 - for; 9, 236–241
 - if; 230
 - if-else; 9
 - import, mechanism description; 469–470
 - local variable use; 170
 - loop flow of control mechanism; 9
 - return; 245
 - switch; 9, 232
 - synchronized; 348–352
 - advantages over synchronized methods; 349
 - factoring into synchronized methods; 352
 - inner class use; 351
 - synchronized methods relationship to; 349
 - syntax and use; 11
 - term definition; 1
 - throw; 282
 - try; 286–291
 - while; 9, 235
 - loop mechanism; 5
 - with labels; 241
- static**
 - See also:* class(es)
 - class
 - field declaration use; 7, 14
 - methods declaration use; 17
 - constants, interface declaration of; 29
 - contexts
 - inner classes in; 144
 - term definition; 144
 - data, locking mechanisms; 352
 - fields; 14, 45
 - generic type handling; 251
 - initialization with static initialization block; 55
 - serialization impact; 551
 - term definition; 14
 - import, term definition; 46
 - import on demand, syntax; 72
 - imports, type imports a generalization of; 469
 - initialization blocks, checked exceptions not permitted; 284
 - members
 - accessing; 223
 - hiding of; 89
 - importing, class name use; 71–73
 - raw types use with; 270
 - referencing; 8
 - method modifier; 57
 - methods; 17, 58
 - synchronized; 349
 - synchronized, using the lock of; 351
 - term definition; 58
 - this reference not applicable to; 68
 - nested
 - classes; 134
 - types; 133–136
 - nested classes
 - Character.Subset; 195
 - Character.UnicodeBlock; 195
 - nested enums; 154
 - nested type, importing; 470
 - protected members, accessing; 95
 - reserved keyword (table); 165
 - static initialization block, term definition; 55
 - static initialization block declaration with; 55
 - term definition; 2
 - type, generic type inference based on; 263
 - variables, nested generic types relationship; 253
- stop method**
 - Applet class; 720
 - Thread class, reasons for deprecation; 283, 381
- stopThread method**
 - JVM TI, asynchronous exceptions; 283
- store method**
 - Properties class; 621
- StreamCorruptedException class**; 564
- StreamEndException example class**; 295
- StreamException class**; 289
- streams**
 - See also:* data structures; file(s); I/O; input; output; reading; serialization; writing
 - Buffered; 518–520
 - family of; 500
 - stream that defines behavior; 514
 - byte; 501–506
 - ByteArray; 521–522
 - converting to characters, (footnote); 667
 - converting to Unicode streams from; 512
 - Data; 537
 - Data stream classes; 539
 - DataInput interface; 537
 - DataOutput interface; 537

- Filter, `Filter` character streams
 - compared with; 516
 - object, serialization use; 549
 - `OutputStream`; 505–506
 - `PrintStream` class; 525
 - `PushbackInputStream` class; 529–531
 - sequencing, `SequenceInputStream` class; 528
 - synchronization strategy; 515
 - term definition; 500
 - type tree for, (figure); 502
- character; 507–511
 - buffered, lines of text handling by; 519
 - `CharArray`; 522–523
 - `Filter`, `Filter` byte streams compared with; 516
 - `LineNumberReader` class; 525
 - `PrintWriter` class; 525
 - `PushbackReader` class; 529–531
 - `Reader`; 507
 - standard streams relationships with; 511–512
 - `String`; 523–524
 - synchronization strategy; 515
 - term definition; 500
 - type tree for, (figure); 507
 - `Writer`; 510–511
- closing
 - importance of; 502, 507
 - input byte; 504
 - input character; 509
 - output byte; 505
 - output character; 511
- composite, creating with `Filter` streams; 516
- conversion
 - `InputStreamReader`; 512–514
 - `OutputStreamWriter`; 512–514
- `Data`; 537–539
- files, `File` streams; 541
- `Filter`; 516–518
 - stream that defines behavior; 514
 - synchronization strategy; 516
- in-memory; 514
- input
 - `InputStreamReader`; 512
 - `Reader`; 507
- object byte, `Object` streams; 549
- output
 - output byte; 505
 - `OutputStream`; 505–506
 - `OutputStreamWriter`; 512
 - `Writer`; 510–511
- parser building; 515
- `Piped`; 520
 - stream that defines behavior; 514
- `Print`; 525–527
- `Pushback`; 529
 - serialization protocol; 561
 - standard I/O; 662
 - character streams relationships with; 511
 - synchronization and concurrency in streams; 515
- `System.err`, character streams relationships with; 511
- `System.in`, character streams relationships with; 511
- `System.out`, character streams relationships with; 511
 - term definition; 499
 - that define behavior; 514
 - types, overview of; 514
 - Unicode, converting byte streams to; 512
 - value, `Scanner` handling; 641–644
- `StreamTokenizer` class**; 532–536
 - `commentChar` method; 534
 - `eofIsSignificant` method; 536
 - `lineno` method; 536
 - `lowerCaseMode` method; 536
 - `nextToken` method; 532
 - `ordinaryChar` method; 534
 - `ordinaryChars` method; 534
 - `parseNumbers` method; 535
 - parser building stream; 515
 - `pushBack` method; 536
 - `quoteChar` method method; 534
 - `resetSyntax` method; 535
 - `Scanner` class operational model comparisons; 647–650
 - `Scanner` class vs.; 642
 - `slashSlashComments` method; 536
 - `slashStarComments` method; 536
 - `toString` method; 536
 - `whitespaceChars` method; 534
 - `wordChars` method; 534
- strict floating-point**
 - See also:* floating-point numbers; `Math` class; numbers/numeric; `StrictMath` class
 - arithmetic, non-strict vs.; 203
 - evaluation, term definition; 203
 - interfaces vs. classes; 122
- strictfp keyword**
 - enums; 154
 - floating-point evaluation impact; 204
 - interface declaration; 122
 - method modifier; 57
 - overriding considerations; 85
 - reserved keyword (table); 165
 - `StrictMath` objects defined with; 658
 - term definition; 43

- StrictMath class**; 657–659
 - abs method; 658
 - acos method; 657
 - asin method; 657
 - atan method; 657
 - atan2 method; 657
 - cbrt method; 658
 - ceil method; 658
 - cos method; 657
 - cosh method; 658
 - exp method; 658
 - floor method; 658
 - hypot method; 658
 - IEEERemainder method; 658
 - log method; 658
 - log10 method; 658
 - max method; 658
 - min method; 658
 - pow method; 658
 - random method; 659
 - rint method; 658
 - round method; 658
 - signum method; 658
 - sin method; 657
 - sinh method; 658
 - sqrt method; 658
 - tan method; 657
 - tanh method; 658
 - toDegrees method; 658
 - toRadians method; 657
- String class**; 306–221
 - advantages over string literals; 305
 - CASE_INSENSITIVE_ORDER field, case-insensitive Comparator located in; 575
 - char array mapping; 317
 - charAt method; 22, 307 xx
 - CharSequence interface implementation, regular expression search in; 321
 - CharSequence interface implemented by; 305
 - codePointAt method; 336
 - codePointBefore method; 336
 - codePointCount method; 336
 - compareTo method; 309
 - compareToIgnoreCase method; 309
 - concat method; 316
 - copy constructor; 54
 - copyValueOf method; 318
 - endsWith method; 311
 - equals method; 22, 309
 - equalsIgnoreCase method; 309
 - getBytes method; 319
 - getChars method; 318
 - hashCode method; 312
 - indexOf methods (table); 308
 - intern method; 312
 - lastIndexOf methods (table); 307, 308
 - length method; 22
 - array length field vs.; 307 xx
 - line.separator field; 519
 - name method, enum use; 155
 - offsetByCodePoints method; 336
 - overview; 21–24
 - putIn method; 312
 - regionMatches method; 310
 - regular expression compilation use; 323
 - replace method; 314
 - replaceAll method; 314
 - replaceFirst method; 314
 - separator field, path component separator stored in; 543
 - split method; 314
 - startsWith method; 311
 - StringBuffer class compared with; 22
 - StringWriter use; 524
 - substring method; 313
 - toCharArray method; 22, 318
 - toLowerCase method; 315
 - toString method; 524
 - toUpperCase method; 315, 316
 - trim method; 314
 - valueOf method, converting other types to strings with (table); 316
- String interface**
 - toString method, Enum class use; 159
- string(s)**; 305–336
 - See also*: character(s); data structures
 - beginning of, testing, `String.startsWith`; 311
 - binary numbers representation, `toBinaryString` methods; 189
 - bit patterns representation, `toBinaryString` methods; 189
 - buffer, modifying; 331
 - buffers
 - appending to; 332
 - capacity management; 335
 - deleting characters in; 333
 - extending; 332
 - inserting into; 332
 - modifying; 332
 - reversing the ordering of characters in; 333
 - building, `StringBuilder` use; 330–335
 - canonical ordering, creating; 309
 - capturing parts of, regular expression use for; 322
 - character, reading and writing, with `DataInput` and `DataOutput`; 538
 - character subsequence, retrieving, `CharSequence.subSequence` use; 306

- comparing; 308
 - `intern.String` use for; 312
 - comparison; 308–311
 - ignoring case in; 575
 - locale-dependent; 708
 - regions of; 310
 - concatenation; 59
 - operator; 214
 - `String.concat`; 316
 - concatenation of; 12
 - conversion; 220
 - byte arrays to/from; 319
 - overview; 23–24
 - to uppercase; 517
 - to/from other types; 316–317
 - to/from upper and lower case; 315
 - tokens to, `StreamTokenizer.toString`; 536
 - conversions that apply to; 218
 - copying into a character array; 319
 - creating
 - from byte arrays; 319
 - from char arrays; 318
 - related; 313
 - `String` class use; 306
 - decoding; 316
 - `toString` method use; 317
 - delimited, extracting; 313
 - encoding, `toString` method use; 317
 - end, testing, `String.endsWith`; 311
 - equality testing; 311
 - expression types; 216
 - extracting from `StringBuilder` object; 333
 - finding
 - first occurrence of a character in, `String.indexOf`; 308
 - last occurrence of a character or substring in, `String.lastIndexOf`; 307
 - format, term definition; 624
 - format conversion; 629
 - format specifier; 23
 - formatting, overview; 23–24
 - hexadecimal numbers representation, `toHexString` methods; 190
 - index position, out of bounds exceptions; 307
 - interning; 311–313
 - length, retrieving, `String.length` use; 307
 - xx
 - length of, retrieving, `CharSequence.length` use; 306
 - literals; 168
 - equality testing; 311
 - term definition; 3
 - mapping char arrays to and from; 317
 - modifying, `StringBuilder` use; 330–335
 - object representation as; 624
 - objects, overview; 21–24
 - octal numbers representation, `toOctalString` methods; 189
 - parsing, `parseType` method; 187
 - parsing into tokens, `StringTokenizer` class; 651
 - pattern matching, regular expression use; 321
 - processing
 - in `StreamTokenizer`; 535
 - `StreamTokenizer.quoteChar`; 534
 - reading, `StringReader` class; 523
 - references, comparing, `String.intern` use; 312
 - regular expression search in; 321
 - representation
 - an object, `Object.toString`; 100
 - `toBinaryString` method, `Integer` class; 189
 - `toBinaryString` method, `Long` class; 189
 - `toHexString` method, `Integer` class; 190
 - `toHexString` method, `Long` class; 190
 - `toOctalString` method, `Integer` class; 189
 - `toOctalString` method, `Long` class; 189
 - `toString` method, `Integer` class; 189
 - `toString` method, `Long` class; 189
 - `toString` method, wrapper classes; 187
 - of type objects; 428–429
 - resource
 - bundle mapping, as localization tool; 686
 - property file, as resource bundles; 693
 - retrieving characters from, `CharSequence.charAt` use; 305
 - search, invalid index handling; 311
 - sorting, `String.compareTo`; 309
 - `StringTokenizer` class; 651
 - `toBinaryString` method
 - `Integer` class; 189
 - `Long` class; 189
 - `toHexString` method
 - `Integer` class; 190
 - `Long` class; 190
 - `toOctalString` method
 - `Integer` class; 189
 - `Long` class; 189
 - `toString` method
 - `Integer` class; 189
 - `Long` class; 189
 - wrapper classes; 187
 - value representation as; 624
 - writing, `StringWriter` class; 524
- String streams**
- for characters; 523–524
 - in-memory stream; 514

- StringBuffer class**; 335
 - appendCodePoint method; 336
 - codePointAt method; 336
 - codePointBefore method; 336
 - codePointCount method; 336
 - copy constructor; 54
 - getBuffer method; 524
 - Matcher class use with, (footnote); 326
 - offsetByCodePoints method; 336
 - String class compared with; 22
 - StringWriter use; 524
- StringBuilder class**; 330–335
 - append method; 332
 - appendCodePoint method; 336
 - CharSequence interface implementation,
 - regular expression search in; 321
 - CharSequence interface implemented by; 305
 - codePointAt method; 336
 - codePointBefore method; 336
 - codePointCount method; 336
 - copy constructor; 54
 - delete method; 333
 - deleteCharAt method; 333
 - ensureCapacity method; 335
 - extracting, strings, substrings, and char arrays from; 333
 - Formatter use; 631
 - getChars method; 333
 - indexOf method; 331
 - insert method; 332
 - lastIndexOf method; 331
 - offsetByCodePoints method; 336
 - regular expression compilation use; 323
 - replace method; 333
 - reverse method; 333
 - setCharAt method; 331
 - setLength method; 332
 - String class compared with; 22
 - substring method; 331
- StringIndexOutOfBoundsException class**; 307
- StringReader class**; 523
- StringTokenizer class**; 651–653
 - countTokens method; 652
 - hasMoreTokens method; 652
 - nextToken method; 652
- StringWriter class**; 524
 - synchronization strategy; 515
- strong typing**
 - term definition; 27, 90
- strongly reachable/strong reference**
 - reference queue importance; 459
 - term definition; 456
- structure application**
 - reflection use to examine; 397
- subclass(ing)**
 - inherited method documentation comments; 489–491
 - ResourceBundle class; 693
 - restrictions, alternatives to; 730
 - superclass synchronization impact on; 348, 353
 - term definition; 24, 75
- subinterfaces**
 - term definition; 123
- subList method**
 - List interface; 581
- subMap method**
 - SortedMap interface; 590
- subSequence method**
 - CharSequence interface; 306
- subSet method**
 - SortedSet interface; 578
- subsets**
 - Unicode, defining, Character.Subset class; 194
- substring method**
 - String class; 313
 - StringBuilder class; 331
- substrings**
 - See also*: data structures; strings
 - comparing; 308, 310
 - extracting from StringBuilder object; 333
 - finding
 - first occurrence of in a string, String.indexOf; 308
 - last occurrence of in a string, String.lastIndexOf; 307
- subtraction (-) operator**; 201
 - See also*: arithmetic; numbers/numeric; operator(s)
 - term definition; 6
- subtypes**
 - exceptions, overriding method use; 286
 - generic types; 256
 - term definition; 29, 117
- Suit example class**; 8
- Suit example enum class**; 151
- super keyword**
 - good programming practice
 - invoking super.clone method; 106
 - invoking super.finalize method; 451
 - invoking superclass constructor with; 80
 - reference, method overriding use of; 85
 - referencing superclass objects with; 89
 - reserved keyword (table); 165
 - term definition; 25
- super reference**
 - qualified, term definition; 138

superclass(es)

See also: class(es)
 accessing, super use for; 89
 behavior, overriding; 24
 constructors
 enum constructors not permitted to
 invoke; 156
 invoking; 80
 inherited method documentation comments;
 489–491
 invoking methods from; 25
 new, serialization handling of; 559
 retrieving, `Class.getSuperClasses`
 method; 407
 term definition; 24, 75

superinterfaces

term definition; 119, 123

SuperShow example class; 86**supertypes**

term definition; 29, 117

supplementary characters

See also: code, points; UTF-16
 term definition; 162
 working with; 196–198

SuppressWarnings class; 396, 745**SURROGATE constant**

Character class; 195

surrogate pairs

See also: code, points; Unicode; UTF-16
 working with; 196–198

suspending threads

busy-waiting vs.; 356
 wait method use; 354

sval field

`String` class, `TT_WORD` value contained in;
 532

swap method

`Collections` class; 598

swapping

words, pattern matching use; 328

Swing components

`javax.swing` package; 740

switch

block, term definition; 232
 reserved keyword (table); 165
 statement; 232
 enum use with; 160
 expressions, type of; 235
 flow of control mechanism; 9

symmetry around zero

remainder (%) operator preservation of; 658

SyncFailedException class; 564**synchronized/synchronization**

See also: concurrency/concurrent; deadlock;
 lock(s); threads
 accessors, advantages of; 347

actions, term definition; 372

atomic access strategies; 370

client-side

server-side synchronization vs.; 352
 synchronized statement use of arbitrary
 objects; 352
 term definition; 349

code, term definition; 345

concurrency and; 515–516

constructors

modifier not used with; 347
 statement use within; 351

`CountDownLatch` class; 734

`CyclicBarrier` class; 734

design; 352–354

enforcement, implementation concern; 353

exceptions

asynchronous compared with; 283
 term definition; 380

`Exchanger` class; 734

`Hashtable` class methods; 619

inner class use; 351

`java.util.concurrent` package; 733–735

locks use for; 345

memory model and `volatile`; 370–375

methods; 346–348

 factoring synchronized statement code
 into; 352

 synchronized statement advantages
 over; 349

 synchronized statement relationship to;
 349

modifier not used with constructors; 347

objects, unsynchronized methods use of; 350

`Observable` class mechanisms; 637

order, term definition; 372

overriding considerations; 85

policy, documentation importance; 353

requirements, class implementation
 component; 348

reserved keyword (table); 165

resource recovery key use; 461

scheduling behavior impact; 362

`Semaphore` class; 733

server-side, client-side synchronization vs.;
 352

shutdown hook issues; 673

statements; 348–352

 advantages over synchronized methods;
 349

 factoring into synchronized methods;
 352

 synchronized methods relationship to;
 349

static methods; 348

 using the lock of; 351

synchronized/synchronization (*cont.*)

- synchronized keyword, method modifier; 57
- term definition; 345
- thread execution impact; 346
- threads; 345
- Vector methods; 617
- view, collections, wrapped collections
 - provision of; 597
- volatile variables as alternative to; 370
- wrappers
 - collection classes use; 602–604
 - concurrent collections and; 602–607
 - iteration; 604
 - multithreaded design use of interfaces; 352
 - term definition; 602
- synchronizedCollection method**
 - Collections class; 602
- synchronizedList method**
 - Collections class; 602
- synchronizedMap method**
 - Collections class; 602
- synchronizedSet method**
 - Collections class; 602
- synchronizedSortedMap method**
 - Collections class; 602
- synchronizedSortedSet method**
 - Collections class; 602
- SynchronousQueue class**; 606
- syntax**
 - block; 230
 - break statement; 241
 - continue statement; 244
 - do-while statement; 236
 - if-else statement; 230
 - initialization blocks; 55
 - labels; 241
 - method declarations; 57
 - package separator vs. member access, confusion with; 37
 - resetting, StreamTokenizer class; 535
 - return; 245
 - switch statement; 232
 - throw statement; 282
 - try blocks; 286
 - variable number of arguments; 58, 60
 - while statement; 235
- system**
 - class loader, term definition; 438
 - classes, term definition; 438
 - programming; 661–684
 - properties; 663
 - line.separator; 524
 - resources, loading; 442

System class; 662–666

- arraycopy method; 176, 665
- clearProperty method; 664
- currentTimeMillis method; 665
- err field; 662
- exit method; 666
 - application execution termination with; 369
- facilities of; 661
- gc method; 452, 666
- getenv method; 670
- getProperties method; 664
- getProperty method; 664
- getSecurityManager method; 678
- identityHashCode method; 666
- load method; 666
- loadLibrary method; 666
- mapLibraryName method; 676
- nanoTime method; 665
- out field; 662
- runFinalization method; 666
- setProperties method; 664
- setProperty method; 664
- setSecurityManager method; 678
- utility methods; 665
- System.err field**
 - character streams relationships with; 511
- System.in field**
 - character streams relationships with; 511
- System.out field**
 - character streams relationships with; 511
- system properties**
 - security; 664

T**T abstract type parameter**

- term definition; 30
- type variable naming convention; 248

tab character

- special symbol for; 167, 322

tables

- useful, (appendix); 749–754

tags (doc comment); 483–488

- @author tag; 486
- @code inline tag; 487
- @deprecated doc comment tag; 486
- @docRoot inline tag; 488
- @exception inline tag; 485
- @inheritDoc inline tag; 488
 - copying doc comments with; 490
- @link inline tag; 484
- @linkPLAIN inline tag; 484
- @literal inline tag; 487
- @param doc comment tag; 485
- @return doc comment tag; 485

- @see tag; 483
- @since tag; 487
- @throws doc comment tag; 485
- @throws tag, inheritance; 490–491
- @value inline tag; 487
- @version tag; 487
- block
 - doc comment, format and use; 483
 - documenting separately inherited entities with; 489
 - doc comment use; 483
 - HTML, doc comment use; 482
 - inline, doc comment, format and use; 483
 - serialization; 562–563
 - (table); 754
- tailMap method**
 - SortedMap interface; 590
- tailSet method**
 - SortedSet interface; 578
- take method**
 - BlockingQueue interface; 604
- tan method**
 - Math class and StrictMath class; 657
- tangent; 657**
 - hyperbolic; 658
- tanh method**
 - Math class and StrictMath class; 658
- Target class; 394**
 - as meta-annotation; 396
- target object**
 - term definition; 16
- task scheduling**
 - fixed-delay, term definition; 655
 - fixed-rate, term definition; 655
 - once-only, term definition; 655
 - with Timer and TimerTask; 653
- Template Method pattern**
 - abstract class role; 97
- temporary files**
 - creating, File.createTempFile; 547
 - deleting, File.deleteOnExit; 547
- term definitions**
 - abrupt completion; 283
 - absolute, placement; 718
 - abstract keyword; 43
 - abstract type parameters
 - T; 30
 - wildcards; 31
 - access
 - accessor methods; 66
 - control; 38
 - modifiers; 19
 - accessibility; 13
 - accessor(s), methods; 66
 - actions
 - events; 719
 - list; 680
 - synchronization; 372
 - active thread; 378
 - addition (+) operator; 6
 - aging, priority; 358
 - alive; 364
 - annotations
 - elements; 35, 388
 - meta; 394
 - types; 35, 387
 - anonymous, array; 176
 - applet(s); 720
 - argument, types; 250
 - arity, (footnote); 60
 - array(s); 18, 173, 429
 - anonymous; 176
 - bounds of; 19
 - empty; 19, 173
 - assertion; 296
 - assignment
 - compatibility; 90
 - operators; 5
 - associativity, operator; 221
 - atomic; 355
 - attributes; 76
 - backed by; 49
 - a c collection; 578
 - base, code; 681
 - beans; 721
 - behavior, constant-specific; 156
 - binary
 - I/O; 500
 - name; 411, 412
 - bit set; 633
 - blank, finals; 46, 172
 - block(s); 5, 230
 - initialization; 54, 55
 - switch; 232
 - body
 - comment; 489
 - method; 3, 57
 - boolean(s), expressions; 5
 - bootstrap, class loader; 438
 - bound
 - arrays; 19
 - lower; 257
 - type parameter; 253
 - upper; 253
 - wildcard; 32, 257
 - boxing conversions; 91, 184, 198
 - bridge method; 746
 - buffers; 499, 565
 - bundles, resource; 689
 - busy-waiting; 356

term definitions (*cont.*)

- byte(s); 500
 - streams; 500
- bytecodes; 2, 38
- calendar, lenient; 699
- canonical
 - name; 411
 - path; 544
- capture, conversion; 266
- case
 - label; 232
 - lower; 193
 - title; 193
 - upper; 193
- casting; 91, 219
 - safe; 92
 - unsafe; 92
- catch; 32
- catching, exceptions; 32, 279
- chain, of references; 447
- channels; 499, 565
- character(s); 500
 - case; 193
 - ordinary; 532
 - special; 532
 - streams; 500
 - supplementary; 162
- checked exceptions; 32, 34, 280
- child, process; 667
- class(es); 1, 41
 - declarations; 12
 - inner; 136
 - loader; 436
 - loader, bootstrap; 438
 - loader, context; 437
 - loader, parent; 438
 - loader, system; 438
 - methods; 17, 58
 - nested; 133
 - path; 435
 - system; 438
 - variables; 14
 - wrapper; 183
- client; 727
- client-side, synchronization; 349
- clone; 100
- cloning
 - deep; 106
 - shallow; 106
- code
 - base; 681
 - point; 162, 192
 - privileged; 681
 - source; 681
 - unit; 162
- collation; 708
- collection(s); 567
 - backed by; 578
- comment(s); 6
 - delimiters; 6
 - doc; 481
 - documentation; 6
 - skew; 497
- comparable, mutually; 118
- comparison, operators; 5
- compatibility, assignment; 90
- completion, abrupt; 283
- components, beans; 721
- composition; 129
- concatenation (+) operator; 12
- constant(s); 7
 - enum; 8
 - enum, (footnote); 152
 - named; 7
 - variables; 46
- constant-specific behavior; 156
- constant-time operations; 579
- constructors; 14, 50
 - copy; 53
 - default; 53
 - no-arg; 53
- container(s); 567
- context
 - class loader; 437
 - naming; 36
- contract; 27, 41, 75
 - inheritance; 75
- control, access; 38
- conversions
 - boxing; 91, 184, 198
 - capture; 266
 - narrowing; 91
 - narrowing primitive; 217
 - unboxing; 184, 198
 - widening; 91
 - widening primitive; 216
- copies, defensive; 127
- copy constructors; 53
- covariant return; 102
- creating, instance creation expression; 214
- critical
 - regions; 345
 - sections; 345
- current, object; 16
- cycles; 449
- daemon
 - group; 376
 - threads; 369
- dangling references; 448
- data, encapsulation; 15
- data-based I/O; 500
- dead, objects; 448

- deadlock; 362
- deadly embrace; 362
- declaration
 - classes; 12
 - methods; 3
 - statements; 230
- decrement (--) operator; 11
- deep, cloning; 106
- default
 - constructors; 53
 - label; 232
- defensive copies; 127
- definitely assigned variable, (footnote); 171
- descriptors, file; 541
- deserialization; 549
- destination; 499
- division (/) operator; 6
- doc comments; 481
- documentation, comments; 6
- domain, protection; 677, 681
- downcasting; 92
- elements, annotation; 35, 388
- empty, array; 19, 173
- encapsulation, data; 15
- enclosing
 - instances; 136
 - objects; 136
- enum; 8, 151
 - constants, (footnote); 152
 - types, (footnote); 152
- enumeration types; 8, 151
- environment(s), variables; 669
- epoch; 37, 665, 695
 - control; 36
- equivalence; 100, 101
 - natural; 574
- erasure; 267
- escape, sequence; 167
- event(s)
 - action; 719
 - queue; 718
 - thread; 718
- exception(s); 32
 - catching; 32, 279
 - checked; 32, 34, 280
 - synchronous; 380
 - throwing; 32, 279
 - uncaught; 380
 - unchecked; 34, 280
- exclamation equals (!=), inequality operator; 5
- expression(s); 214
 - Boolean; 5
 - instance creation expression; 214
 - statements; 229
- extending; 24
- fail-fast iterators; 574
- failure(s), partial; 728
- field(s); 1, 44
 - non-static; 14, 46
 - static; 14
- file(s)
 - descriptors; 541
 - pointer; 541
 - random access; 541
- final
 - blank; 46, 172
 - keyword; 43
- fixed-delay task scheduling; 655
- fixed-rate task scheduling; 655
- flow of control; 9
- format
 - specifier; 23, 624
 - string; 23, 624
- forwarding; 129
- FP-strict; 203
- fully-qualified, names; 36, 412
- garbage; 447
 - collection; 49, 447
- generic
 - type invocation; 250
 - types; 29, 247
- graph, object; 549
- greedy tokenizer; 164
- groups
 - daemon; 376
 - thread; 375
- HasA relationship; 107
- header; 3, 57
- heap; 13
- hooks, shutdown; 672
- horizon; 646
- I/O
 - data-based; 500
 - non-blocking; 499
 - text-based; 500
- idempotent; 728
- identifiers; 164
 - universally unique (UUID); 656
- identity; 100
- immutable; 7, 22, 171
 - properties; 46
- implementation, inheritance; 75
- import
 - on demand; 469
 - single type; 469
- in scope; 140
- inheritance; 24
 - contract; 75
 - implementation; 75
 - interface; 115
 - type; 75

term definitions (*cont.*)

- initialization
 - blocks; 54
 - lazy; 47
 - static initialization block; 55
- inlining; 96
- inner classes; 136
- input, streams; 500
- insertion order; 580, 591
- instance(s); 1, 41
 - creation expression; 214
 - enclosing; 136
 - variables; 14
- instantiation; 13
- interface(s); 28, 117
 - inheritance; 115
 - listener; 719
 - nested; 29, 133
 - remote; 727
- internationalization; 685
- intersection types; 264
- introspection; 397
- invariant; 296
- invocation
 - generic type; 250
 - methods; 16
- IsA relationship; 107
- iteration, fail-fast iterators; 574
- iterator, weakly consistent; 596
- JAR** (Java ARchive) file format, manifest; 736
- javadoc; 481
- key; 460, 587
- layout manager; 718
- lazy, initialization; 47
- lenient calendar; 699
- list(s); 580
- listener; 718
 - interface; 719
- literals; 7, 166
 - string; 3
- live, objects; 448
- loaders
 - class; 436
 - class, bootstrap; 438
 - class, parent; 438
 - class, system; 438
 - context class; 437
- locale(s); 685
 - sensitive; 686
 - sensitive, operations; 685
- localization; 685
- localize; 686
- locks; 345
- lower bound; 257
- lowercase; 193
- magic numbers; 7
- main method; 2
- manager
 - layout; 718
 - security; 38, 677
- manifest; 736
- mark-and-sweep garbage collection; 449
- member(s); 1
- memory, model; 371
- meta-annotations; 394
- method(s); 1, 15, 41, 56
 - accessor; 66
 - body; 3, 57
 - bridge; 746
 - class; 17
 - declaration; 3
 - header; 3
 - headers; 57
 - invoking; 16
 - main; 2
 - non-static; 17
 - static; 17, 58
- models
 - memory; 371
 - single-threaded programming model; 337
- modifiers; 43
 - access; 19
- multithreading; 338
- mutually
 - comparable; 118
 - exclusive; 346
- name
 - attributes; 76
 - binary; 411
 - canonical; 411
 - constants; 7
 - fully qualified; 412
 - fully-qualified; 36
 - simple; 411
 - type; 42
- namespaces; 178
- NaN (Not a Number); 166
- narrowing
 - conversion; 92
 - primitive conversion; 217
- natural
 - equivalence; 574
 - ordering; 574
- nested
 - classes; 133
 - interfaces; 29, 133
- no-arg constructors; 53
- non-blocking I/O; 499
- non-static
 - fields; 14, 46
 - methods; 17

- not FP-strict; 203
- null; 13
- object(s); 1, 41
 - current; 16
 - dead; 448
 - enclosing; 136
 - graph; 549
 - live; 448
 - receiver; 16
 - reference; 13, 455
 - serialization; 501
 - target; 16
- once-only task scheduling; 655
- operator(s), associativity; 221
- order
 - insertion; 591
 - natural; 574
 - resource; 364
 - synchronization; 372
 - total; 574
- ordinary characters; 532
- output streams; 500
- overloading; 6, 69, 84, 126
- overriding; 25, 84
- package(s), sealed; 478
- parameterized types; 250
- parameters
 - bounded type; 253
 - list; 2
 - type; 250
- parent
 - class loader; 438
 - process; 667
- partial, failures; 728
- pass
 - by reference; 65
 - by value; 63
- path; 543
 - canonical; 544
- permission(s); 677
- phantom, reachable; 456
- pipes; 520
- pointer, file; 541
- policy, security; 38, 677
- polling; 338
- polymorphism; 25, 75
- pooling of threads; 383
- postcondition; 298
- precedence, operator; 221
- precision indicator; 625
- precondition; 298
- preemption, thread; 358
- primitive types; 4
- priority; 358
 - aging; 358
 - queue; 94
- private keyword; 19
- privileged, code; 681
- process; 666
 - child; 667
 - parent; 667
- program order; 371
- properties; 67
 - immutable; 46
- protection domain; 677, 681
- providers; 732
- public keyword; 2, 13, 19, 43
- qualified super reference; 138
- qualified this reference; 138
- queue; 94
 - event; 718
 - priority; 94
 - reference; 459
- race
 - condition; 338
 - hazard; 338
- radix; 185
- random, access files; 542
- raw, types; 250
- raw types; 30, 267, 745
- reachability
 - phantom reachable; 456
 - softly reachable; 456
 - strongly reachable; 456
 - unreachable; 456
 - weakly reachable; 456
- reachable; 448, 454
- read-only; 22
- readers; 500
- receiver object; 16
- reference(s)
 - chain of; 447
 - counting; 449
 - dangling; 448
 - objects; 455
 - queues; 459
 - root; 447
 - types; 13
- referent; 454
- reflection; 397
- regions; 329
- regular expressions; 321
- relative, placement; 718
- remote, interface; 727
- resource(s)
 - bundles; 689
 - ordering; 364
- return
 - covariant; 102
 - value; 16
- root, reference; 447
- runtime, system; 38

term definitions (*cont.*)

- safe casting; 92
- safety, thread; 353
- sandbox; 720
- scheduling
 - tasks, fixed-delay; 655
 - tasks, fixed-rate; 655
 - tasks, once-only; 655
- scope; 140, 179
- sealed packages; 478
- security
 - manager; 38, 677
 - policy; 38, 677
- serialization; 549
 - object; 501
- server(s); 727
- set(s); 577
- shadowing, (footnote); 180
- shallow, cloning; 106
- shutdown, hooks; 672
- signature; 2, 57, 69
- simple name; 411
- single-threaded, programming model; 337
- slicing, time; 358
- softly reachable; 456
- source; 499
 - code; 681
- special, characters; 532
- specialization; 75
- specifiers, format; 23, 624
- spurious wakeups; 358
- standard extensions mechanism; 681
- state; 1
- statement(s); 1
 - declaration; 230
 - expression; 229
- static**
 - contexts; 144
 - fields; 14
 - keyword; 2, 14
 - methods; 17, 58
 - static initialization block; 55
- streams; 499
 - byte; 500
 - character; 500
- strictfp** keyword; 43
- string, format; 23
- string(s), literals; 3
- strong typing; 27, 90
- strongly reachable; 456
- subclassing; 24, 75
- subinterfaces; 123
- subtraction (-) operator; 6
- subtypes; 29, 117
- super keyword; 25
- super reference, qualified; 138
- superclass; 24, 75
- superinterfaces; 119, 123
- supertypes; 29, 117
- supplementary characters; 162
- switch, block; 232
- synchronized; 345
 - actions; 372
 - client-side; 349
 - code; 345
 - exceptions; 380
 - order; 372
 - wrappers; 602
- system
 - class loader; 438
 - classes; 438
- target object; 16
- task
 - scheduling, fixed-delay; 655
 - scheduling, fixed-rate; 655
 - scheduling, once-only; 655
- text, -based I/O; 500
- this reference, qualified; 138
- thread(s); 337
 - active; 378
 - daemon; 369
 - event; 718
 - groups; 375
 - pooling; 383
 - preemption; 358
 - safety; 353
 - single-threaded programming model; 337
- throw; 32
- throwing, exceptions; 32, 279
- time, slicing; 358
- titlecase; 193
- token; 164
 - type; 100, 401
- tokenization; 532
- total, ordering; 574
- type(s); 41, 117, 166
 - annotation; 35, 387
 - arguments; 250
 - bounded type parameter; 253
 - casting; 27
 - enum, (footnote); 152
 - enumeration; 8, 151
 - generic; 29, 247
 - generic, type invocation; 250
 - inheritance; 75
 - intersection; 264
 - names; 42
 - narrower; 90
 - parameterized; 250
 - parameters; 250
 - raw; 30, 250, 267, 745
 - token; 100, 252

- tokens; 401
 - of variables; 4
 - variables; 248
 - wider; 90
- typing, strong; 27
- unbounded wildcard; 32, 257
- unboxing conversions; 184, 198
- uncaught exception; 380
- unchecked exceptions; 34, 280
- unnamed package; 37
- unreachable; 456
- unsafe casting; 92
- upcasting; 92
- upper bound; 253
- uppercase; 193
- UUID (Universally Unique Identifier); 656
- values
 - attributes; 76
 - pass by value; 63
 - return; 16
- variables; 169
 - blank final; 172
 - class; 14
 - constant; 46
 - definitely assigned, (footnote); 171
 - environment; 669
 - instance; 14
 - type; 248
 - types of; 4
- verifier; 38
- view(s)/viewing, collection; 578
- virtual machine; 38
- visibility; 13
- void keyword; 2
- weakly
 - consistent iterator; 596
 - reachable; 456
- whitespace; 164
- widening
 - conversion; 91
 - primitive conversions; 216
- width indicator; 625
- wildcards
 - abstract type parameter; 31, 32
 - bounded; 32, 257
 - unbounded; 32, 257
- wrapper(s); 183
 - classes; 4
 - synchronized; 602
- writers; 500
- TERMINATED constant**
 - Thread class, State nested enum; 384
- termination**
 - See also:* cancellation; finalization
 - application execution; 369
 - daemon thread use for; 369
 - blocks, break use for; 241
 - loops, break use for; 242
 - method execution; 62
 - return use for; 245
 - switch statements; 235
 - thread execution; 365–369
 - cancellation strategy advantages; 365
 - threads, synchronization actions; 372
 - uncaught exception cause of; 283
 - virtual machine; 672–675
- terminator**
 - separator vs., (footnote); 229
- ternary (?) operator**; 210–212
- testing**
 - beginning of string, `String.startsWith`; 311
 - character encoding support; 320
 - character type, methods for; 194
 - end of string, `String.endsWith`; 311
 - equality
 - object; 99–100
 - values; 99–100
 - infinity, `isInfinite` methods in floating-point wrapper classes; 191
 - instances; 208
 - NaN; 206
 - stream read readiness; 509
 - string literal equality; 311
 - types; 92
- TestSort example class**; 112
 - reflection use; 423
- text**
 - See also:* character(s); string(s); Unicode
 - based I/O, term definition; 500
 - boundaries; 712
 - break points, `BreakIterator` class use; 712
 - formatted, `Formatter` class; 624–632
 - formatting, customization; 630
 - internationalization for; 708
 - lines of, buffered character stream handling; 519
 - localization for; 708
 - parsing, `StreamTokenizer` class; 532–536
 - thread reading of, with pipe streams; 520
 - tokenizing, `StreamTokenizer` class; 532–536
 - tracking lines while reading, `LineNumberReader` class; 527
- TextGenerator example class**; 520
- this reference**; 17
 - constructor order relative to; 81
 - explicit constructor invocation use of; 52
 - extended class constructor invocation use; 80
 - inner class use; 138
 - non-static methods use of; 68–69
 - qualified, term definition; 138

this reference (*cont.*)

reserved keyword (table); 165
 static methods prohibited from using; 18, 58
 super reference compared with; 25
 synchronized method implicit use of; 349

Thread class; 32

applet use; 720
 checkAccess method; 376
 constructor, specifying thread group in; 376
 currentThread method; 341
 dumpStack method, debugging use; 385
 getAllStackTraces method; 382
 getContextClassLoader method; 438
 getDefaultUncaughtExceptionHandler method; 380
 getID method, debugging use; 384
 getName method; 341
 getPriority method; 359
 getState method, debugging use; 384
 getThreadGroup method; 376
 getUncaughtExceptionHandler method; 380
 holdsLock method; 347
 interrupt method; 339, 366, 379, 381
 interrupted method; 366
 isAlive method; 365
 Thread.join defined in terms of; 369
 isInterrupted method; 366
 join method; 368
 thread completion wait use; 367
 MAX_PRIORITY constant; 359
 MIN_PRIORITY constant; 359
 NORM_PRIORITY constant; 359
 run method; 339
 completion, thread termination as result of; 364
 Runnable interface implementation; 341
 setContextClassLoader method; 438
 setDaemon method; 369
 setName method; 341
 setPriority method; 359
 setUncaughtExceptionHandler method; 380, 381
 sleep method; 360
 InterruptedException thrown by; 366
 start method; 339
 State enum; 384
 stop method
 reasons for deprecation; 283
 reasons why it is deprecated; 381
 toString method, debugging use; 384
 wait method, InterruptedException
 thrown by; 366
 yield method; 360

thread(s); 337–385

See also: concurrency; deadlock; deadly embrace; multithreading; synchronized/synchronization;
 Thread class; ThreadGroup class;
 accessing, Thread.currentThread; 341
 actions, memory model impact on; 371
 active, term definition; 378
 blocking, thread scheduler handling; 358
 blocking I/O cancellation; 516
 BlockingQueue implementations use; 605
 cancellation; 365–367
 Thread.interrupt vs. Thread.stop; 381
 communication
 busy-waiting avoidance; 356
 mechanisms for; 354
 pipes use for; 520
 preemption time control use; 359
 completion, waiting for; 367–369
 configuring; 339
 constructors, Runnable object specification; 342
 creating; 339
 daemon
 shutdown hook interaction with; 673
 term definition; 369
 user threads compared with; 369
 death, thread local variables impact; 383
 debugging; 384–385
 dependencies, thread completion wait; 367
 event, term definition; 718
 exceptions and; 379–381
 further reading; 757
 garbage collection, ThreadGroup
 relationship to; 344
 groups
 creating; 377
 garbage collection; 344
 printing state, as debugging aid; 385
 security; 375–379
 security handling; 375–379
 security modification; 376
 term definition; 375
 interference, synchronized method
 prevention; 347
 InterruptedException impact on; 525
 interruption of, cancellation strategy; 365
 kinds of
 daemon; 369
 user; 369
 local inner class restrictions relationship to; 142
 local variables; 382
 management, security and thread groups; 375–379

- management of, with ThreadGroup class; 375
 - naming; 341
 - Observer/Observable mechanism; 639
 - Observer/Observable mechanism compared with; 638
 - pooling
 - local variable dangers; 384
 - multithreading design; 345
 - term definition; 384
 - preemption, term definition; 358
 - priority
 - management strategies; 359
 - upper limit, thread group control of; 377
 - reaper, resource manager use; 463
 - safety, term definition; 353
 - scheduling; 358–362
 - Thread.yield impact on; 360
 - voluntary rescheduling; 360–362
 - security
 - behavior that impacts; 376
 - check issues; 679
 - creation; 376
 - handling, with ThreadGroup class; 375–379
 - modification; 376
 - single-threaded programming model, term definition; 337
 - specifying thread group in constructor; 376
 - suspending
 - busy-waiting vs.; 356
 - wait method use; 354
 - synchronization; 345
 - actions that facilitate multithreading; 372
 - byte stream and character stream strategies for; 515
 - Collections class, wrappers for; 597
 - Object.notify; 354, 357
 - Object.notifyAll; 354, 357
 - Object.wait; 354, 357
 - Process.waitFor; 668
 - synchronized methods; 345 xx
 - synchronized statements; 345 xx
 - Thread.join; 368
 - Vector class; 617
 - term definition; 337
 - termination; 365–369
 - uncaught exception cause; 279
 - thread group specification; 376
 - thread-blind classes, converting to multithreaded environments; 352
 - Timer class use; 654–655
 - uncaught exception handling; 380
 - user, daemon threads compared with; 369
- ThreadGroup class**
- activeCount method; 378, 379
 - activeGroupCount method; 379
 - applet use; 720
 - checkAccess method; 378
 - enumerate method; 378, 379
 - getMaxPriority method; 378
 - getName method; 377
 - getParent method; 377
 - isDaemon method; 378
 - list method, debugging use; 385
 - parentOf method; 378
 - setDaemon method; 377
 - setMaxPriority method; 377, 378
 - specifying in thread constructor; 376
 - thread management and security with; 375–379
 - Thread.stop method use; 381
 - toString method, debugging use; 385
 - uncaughtException method; 379, 380
- ThreadLocal class**; 382–384
- initialValue method; 383
- ThreadPoolExecutor class**
- Executor interface implementation; 734
- threads**
- See also:* concurrency/concurrent; synchronized/synchronization
- throw**
- reserved keyword (table); 165
 - statement; 282
 - location in a constructor; 54
 - overriding considerations; 85
 - switch terminator; 235
 - term definition; 32
- Throwable class**; 33
- deprecated stop method use; 382
 - erasure impact on use of; 268
 - exceptions extension of; 280
 - getMessage method; 280
 - getStackTrace method; 382
 - initCause method, chaining support; 292
 - subtypes of, throw statement use; 282
- Throwable interface**
- getCause method, reflection use; 413
- throwing exceptions**
- term definition; 32, 279
- throws**
- clause
 - method declaration use; 284–286
 - method overriding and; 285
 - native methods and; 286
 - reserved keyword (table); 165
- Tibetan**
- word for rose, (footnote); 550
- tilde (~)**
- unary bitwise complement operator; 209

time

See also: date(s); internationalization; localization; scheduling
 accessing type 1 (time-based) **UUIDs**; 657
 algorithm execution, big *O* notation use; 579
 current, retrieving,
 System.currentTimeMillis; 665
 Formatter class use; 706–708
 -out
 join completion strategy; 368
 ReferenceQueue.remove use; 459
 wait completion strategy; 357
 parsing; 703
 preemption control, thread communication
 mechanism use; 359
 representation of; 695–703
 slicing, term definition; 358
 zones; 700

TIMED_WAITING constant

Thread class, State nested enum; 384

Timer class; 653–656

schedule method; 655
scheduleAtFixedRate method; 655

TimerTask class; 653–656

cancel method; 654
run method; 654
scheduledExecutionTime method; 654

timestamp method

UUID class; 657

TimeUnit class

blocking queue use; 605

TimeZone class; 696, 700–701

getAvailableIDs method; 700
getDefault method; 700
getDisplayName method; 700
getDSTSavings method; 700
getID method; 700
getOffset method; 701
getRawOffset method; 700
getTimeZone method; 700
inDaylightTime method; 701
setDefault method; 700
setID method; 700
setRawOffset method; 700
useDaylightTime method; 701

titlecase

See also: case; text
 term definition; 193
 testing for, **Character.isTitleCase**; 195
 Unicode case concept; 309

TITLECASE_LETTER constant

Character class; 195

toArray method

Collection interface; 575

toBinaryString method

Integer class; 189
 Long class; 189

toByteArray method

ByteArrayOutputStream class; 522

toCharArray method

CharArrayWriter class; 523
 String; 22, 318

toChars method

Character class; 196, 197

toCodePoint method

Character class; 198

toDegrees method

Math class and **StrictMath** class; 658

toGenericString method

Member interface; 416

toHexString method

Integer class; 190
 Long class; 190

token(s); 164

See also: delimiters
 breaking input into, Pushback stream use;
 529
 operators, and expressions; 161–182
 parsing strings into; 651
 product of compiler scanning process; 161
 pushing back, **StreamTokenizer.pushBack**;
 536
 scanner identification of; 643
 term definition; 164
 types; 532
 reflection; 400–402
 term definition; 100, 252, 401

tokenization

input lines, Scanner class methods; 645
 term definition; 532

tokenizer

behavior, methods that control,
 StreamTokenizer class; 535

toLowerCase method

Character class; 194
 String class; 315

toMatchResult method

Matcher class; 326

toOctalString method

Integer class; 189
 Long class; 189

tools

annotation processing by; 387–396

toRadians method

Math class and **StrictMath** class; 657

toString method

Arrays class; 608
 ByteArrayOutputStream class; 522
 CharArrayWriter class; 523
 CharSequence interface; 306

- Class class; 405
- converting primitive values to strings with; 23
- Enum class use; 159
- Integer class; 189
- Locale class; 688
- Long class; 189
- Member interface; 416
- Object class; 59, 100
- overriding, string encoding and decoding use; 317
- Pattern class; 324
- StreamTokenizer class; 536
- String class; 524
- Thread class, debugging use; 384
- ThreadGroup class, debugging use; 385
- UUID class; 657
- wrapper classes; 187
- total ordering**
 - term definition; 574
- totalMemory method**
 - Runtime class; 453
- toLowerCase method**
 - Character class; 194
- toUpperCase method**
 - Character class; 194
 - String class; 315, 316
- toURI method**
 - File class; 544
- toURL method**
 - File class; 544, 726
- traceInstructions method**
 - Runtime class; 677
- traceMethodCalls method**
 - Runtime class; 677
- traces**
 - object construction (table); 83
 - stack; 294
 - debugging use of; 382
- transcendental constants; 658**
- transfer of control**
 - See also:* break statement; continue statement; flow of control
 - with exception handling; 283
- transforming**
 - data, a sequence of operations, Filter streams use for; 516
- transient**
 - default deserialization handling; 552
 - fields
 - serialization impact on; 551
 - serialized fields and; 561
 - reserved keyword (table); 165
- TranslateByte example class; 506**
- translation**
 - language, resource bundles as tool for; 688
- Transportation Commission on Unmet Needs**
 - California, quotation; 715
- travel agent quote; 1**
- tree(s)**
 - See also:* collection(s); data structures; hierarchy
 - binary
 - TreeMap class; 570
 - TreeSet class; 569
 - TreeSet class; 579
 - type
 - byte streams, (figure); 502
 - character streams, (figure); 507
 - collections, in java.util package, (figure); 568
 - concrete collections, (figure); 568
 - serialization use; 552–553
- TreeMap class; 593–594**
 - collation use; 710
 - Map interface implemented by; 590
 - overview; 570
- TreeSet**
 - TreeMap comparison with; 593
- TreeSet class; 580**
 - collation use; 710
 - overview; 569
- Trillin, Calvin**
 - quotation; 228
- trim method**
 - String class; 314
- trimToSize method**
 - ArrayList class; 582
 - StringBuilder class; 335
- true**
 - boolean; 5, 167
 - identifiers prohibited from using; 165
- TRUE constant**
 - Boolean class; 188
- trust**
 - package access handling; 471
- try**
 - reserved keyword (table); 165
 - statement; 286–291
 - try-catch sequence; 286
 - try-catch-finally sequence; 33, 286
 - try-finally sequence; 289
- TT_EOF token type; 532**
- TT_EOL token type; 532**
- TT_NUMBER token type; 532**
- TT_WORD token type; 532**
- Twain, Mark**
 - quotation; 447
- two's-complement arithmetic**
 - integer arithmetic; 201

TYPE constant

ElementType class; 394
 wrapper classes; 186

Type interface

as marker interface; 399
 overview and place in introspection
 hierarchy; 399

type(s)/typing

See also: class(es); conversions; data types;
 generic; inheritance; interfaces;
 primitive types; return types; wildcards

accessing, at runtime; 414

advantages of; 183

annotation; 389–391

term definition; 387

term definitions; 35

annotation of; 392–393

argument, term definition; 250

arrays and; 177

casting

explicit; 91–92

operator; 27

term definition; 27

checking, final class advantages for; 97

collection, legacy; 616

compatibility; 90

and conversion of; 90–93

requirements with assignment; 212

components of, methods for examining; 408

concrete, type variables relationship to; 248

conditional operator expression, rules for
 determining; 211

conflicts, protected member accessing; 94

contract relationship to; 41

conversion

explicit type casts; 219

implicit; 216

strings to and from other types; 315

term definition; 27

covariant return, term definition; 85

defining; 27

design, named constants use; 121

enclosing, static nested type relationship to;
 133

enum, term definition, (footnote); 152

enumeration; 151–160

nesting within an interface; 121

term definition; 8, 151

examining, reflection classes use for; 397

exception, creating; 280

expressions; 215

generic; 247–277

class extension and; 276–277

class extension using; 76

compatibility design issues; 744–748

declarations; 250–255

doc comment tags use; 487

further reading; 757

impact on method resolution; 225

importing; 469

inspection; 426–429

introspection hierarchy figure

representation of; 399

invocation, term definition; 250

"most specific" algorithm modification

for; 272–276

nested; 253–255

overview; 29–32

subtyping; 256–260

term definition; 29, 247

wildcards and; 256–260

working with; 256–260

hierarchy

Class methods that walk; 400

exception, (figure); 280

wrapper classes, (figure); 183

implementation-independent, defining,

interfaces use for; 76

importing; 37

imports; 469–471

inference, generic invocations and; 262–264

inheritance, term definition; 75

interface, references, as powerful design tool;
 120

intersection, term definition; 264

introspection hierarchy figure representation
 of; 399

literals and; 166

members, doc comment specification of; 483

names

creation by both classes and interfaces;
 118

term definition; 42

namespace; 178

narrower, term definition; 91

nested

accessing information about, Class class
 use; 406

implementation of; 149

inheritance of; 146–148

name notation; 412

object, object references vs.; 26

objects, string representation of; 428–429

parameter, bounded, term definition; 253

parameterized

array restrictions; 268–270

bounded wildcards and; 257

raw types relationship to; 745

reflection; 427

term definition; 250

- parameters
 - bounded; 252–253
 - doc comment syntax; 485
 - term definition; 250
 - part of variable declarations; 170
 - primitive
 - converting to reference types; 91
 - creating arrays of, `Class` class use; 429
 - `Field` class methods for getting and setting; 419
 - `Method` class use; 422
 - representation by wrapper classes; 183 (table); 166
 - type hierarchy, (figure); 183
 - wrapper classes for; 183–200
 - primitives, reflection access of name; 413
 - raw
 - compatibility design issues; 744–748
 - erasure and; 267–272
 - purpose and limitations of; 747
 - term definition; 30, 250, 267
 - reference; 166
 - assignment compatibility requirements; 90
 - converting primitive types to; 91
 - equality operator use; 206
 - term definition; 13
 - representation of; 183, 411
 - return
 - handling of; 245
 - lack of, `Void` class representation of; 187
 - safety
 - enum advantage; 152
 - enum use; 8
 - scoping
 - nested classes and interfaces; 133
 - search order, when determining semantics of a name; 181
 - shift operator results; 210
 - single type import, term definition; 469
 - static nested; 133–136
 - importing; 470
 - strong; 27
 - term definition; 90
 - switch expressions; 235
 - term definition; 41, 117, 166
 - testing for; 92
 - token, term definition; 100, 252
 - tokens; 532
 - reflection; 400–402
 - term definition; 401
 - tree
 - byte streams, (figure); 502
 - character streams, (figure); 507
 - collections, in `java.util` package, (figure); 568
 - concrete collections, (figure); 568
 - serialization use; 552–553
 - type-safe view, collections, wrapped collections provision of; 597
 - Unicode, retrieving, `Character.getType`; 195
 - UUID**, See, versions, **UUID**
 - variables
 - See also*: generic binding issues; 262
 - collection iteration use; 573
 - in `Collections` class method declarations; 597
 - generic inner classes; 255
 - generic method declaration use; 261
 - inspection of; 426–427
 - introspection hierarchy figure
 - representation of; 399
 - storage variables different from, (footnote); 169
 - term definition; 4, 248
 - wider, term definition; 90
 - wildcard, restrictions on parameter passing use; 265
 - TypeDesc example class**; 402–404
 - TypeNotPresentException class**; 427
 - annotation reflection use; 415
 - `GenericArrayType` interface; 428
 - `WildcardType` interface; 428
 - typeValue methods**
 - wrapper classes; 186
 - TypeVariable interface**; 426–427
 - array returned by
 - `GenericDeclaration.getTypeParameters`; 405
 - `Field` class interaction; 418
 - `getBounds` method; 426
 - `getGenericDeclaration` method; 426
 - `getName` method; 426
 - `Method` class use; 421
- ## U
- U.C. Berkeley Math Professor**
 - quotation; 133
 - U.S. Army's PS magazine**
 - quotation; 279
 - UID**
 - serial version, serialization use; 557
 - unary operators**
 - arithmetic negation (-); 201
 - arithmetic positive (+); 202
 - bitwise complement (~); 209
 - logical negation (!); 207
 - new; 214
 - precedence; 221

- UNASSIGNED constant**
 - Character class; 195
- unbounded wildcard**
 - term definition; 32, 257
- unboxing conversion**
 - term definition; 184, 198
- uncaught exception**
 - term definition; 380
- uncaughtException method**
 - ThreadGroup class; 379
 - UncaughtExceptionHandler interface; 380
- UncaughtExceptionHandler class; 32**
- UncaughtExceptionHandler interface**
 - uncaughtException method; 380
- unchecked**
 - cast, generic method pitfalls; 261
 - conversion, term definition; 745
 - exceptions
 - checked exceptions vs.; 280
 - term definition; 34, 280
 - warnings, when emitted; 745
- UndeclaredThrowableException class**
 - Proxy class use; 435
- undefined**
 - local variable implication; 5
- underflow**
 - floating-point; 202
- undoing**
 - read operations, Pushback stream use; 529
- UnicastRemoteObject class; 730**
- Unicode**
 - See also:* character(s); UTF-16
 - 16-bit characters, characters comprised of; 500
 - Attribute Table, Unicode character
 - definitions in; 193
 - aware case folding, pattern matching; 324
 - blocks, defining, UnicodeBlock class; 195
 - case
 - issues; 309
 - standard; 193
 - characters
 - character stream support for; 507
 - literal encodings; 167
 - overview; 8–9
 - parsing issues; 712
 - testing for, Character.isDefined; 194
 - comments; 163
 - data representation; 9
 - digits, testing for, Character.isDigit; 194
 - encodings; 162
 - backslash; 167
 - backspace; 167
 - carriage return; 167
 - double quote; 167
 - form feed; 167
 - inside character literals; 167
 - newline; 167
 - reasons for multiple u's in, (footnote); 162
 - return; 167
 - single quote; 167
 - tab; 167
 - identifiers
 - Java vs.; 194
 - testing for,
 - Character.isUnicodeIdentifierPart; 194
 - testing for,
 - Character.isUnicodeIdentifierStart; 194
 - impact on string comparison; 308
 - internationalization and localization tool; 685
 - Java character representation; 162
 - numeric value of character, retrieving,
 - Character.getNumericValue; 193
 - PropertyResourceBundle class
 - restrictions; 693
 - range handled by StreamTokenizer; 532
 - streams, converting byte streams to; 512
 - subsets
 - defining, Character.Subset class; 195
 - design issues and strategies; 196
 - type, retrieving, Character.getType; 195
 - UTF relationship to; 537
 - UTF-16, working with; 196–198, 336
- UNICODE_CASE flag**
 - Pattern class; 324
- Unicode character classes**
 - special characters for (table); 752
- UnicodeBlock class; 195**
 - of method; 195
- uniqueness**
 - checking, readUnshared and writeUnshared use; 551
- UNIX**
 - line terminator pattern matching; 324
 - SIGKILL signal, virtual machine abort cause; 674
- UNIX_LINES flag**
 - Pattern class; 324
- UnknownFormatConversionException class; 631**
- UnknownFormatFlagsExceptions class; 631**
- unlabeled**
 - break statement; 241
- unmodifiable**
 - collections, implementation guidelines; 612
 - wrappers; 601
- unmodifiableCollection method**
 - Collections class; 601
- unmodifiableList method**
 - Collections class; 601

- unmodifiableMap method**
 - Collections class; 601
- unmodifiableSet method**
 - Collections class; 601
- unmodifiableSortedMap method**
 - Collections class; 601
- unmodifiableSortedSet method**
 - Collections class; 601
- unnamed package**
 - term definition; 37
- unreachable**
 - term definition; 456
- unread methods**
 - PushbackReader class; 531
- Unrelated example class; 140**
- unsafe casting**
 - term definition; 92
- UnsatisfiedLinkError class**
 - native code library loading use; 676
- unsigned values**
 - handling with signed types; 166
- UnsupportedCharsetException class; 321**
- UnsupportedEncodingException class; 319, 513, 564**
 - character encoding test use; 321
- UnsupportedOperationException class; 568, 612**
 - conventions of use; 571
 - iterator implementation use; 610
 - Iterator interface use; 143
- UnsupportedOperationException class**
 - Scanner.remove use; 642
- UnsupportedOperationException class**
 - unmodifiable wrapper use; 601
- upcasting**
 - term definition; 92
- update method**
 - Observer interface; 636
- upper bound**
 - term definition; 253
 - wildcard
 - generic type use; 257
 - return value use; 258
- uppercase**
 - See also:* case; text
 - term definition; 193
 - testing for, Character.isUpperCase; 195
- UPPERCASE field**
 - FormattableFlags class; 630
- UPPERCASE_LETTER constant**
 - Character class; 195
- UppercaseConverter example class; 517**
- URI class**
 - converting a File object to; 544
- URIs (Uniform Resource Identifiers)**
 - java.net package; 724
- URL class; 725**
 - converting a File object to; 544
 - openConnection method; 725
- URL (Uniform Resource Locator)**
 - java.net package; 724
 - resource bundle objects; 691
- URLConnection class**
 - openConnection method; 725
- URLEncoder class; 726**
- US-ASCII character encoding; 320**
- US-ASCII standard**
 - meaning of (table); 754
- usage**
 - patterns, hashtable design, HashMap class; 591
- useAnchoringBounds method**
 - Matcher class; 329
- useDaylightTime method**
 - TimeZone class; 701
- useLocale method**
 - Scanner class; 651
- usePattern method**
 - Matcher clasinterfaces; 326
- user,**
 - java.awt package, details; 717–720
 - interface
 - design, further reading; 759–760
 - graphical, thread use; 370
 - threads, daemon threads compared with; 369
- userDelimiter method**
 - Scanner class; 643
- userRadix method**
 - Scanner class; 643
- Users example class; 637**
- useTransparentBounds method**
 - Matcher class; 329
- UTC (Coordinated Universal Time)**
 - GregorianCalendar class ranges (table); 701
 - UT vs., (footnote); 701
- UTF (Unicode Transformation Format)**
 - character encoding standards; 320
 - Unicode character relationship to; 536
 - writing, DataInput.readBoolean; 537
- UTF-16 standard**
 - See also:* Unicode
 - character encoding; 320
 - meaning of (table); 754
 - Unicode encoding format; 162
 - working with; 196–198, 336
- UTF-16BE standard; 320**
 - meaning of (table); 754
- UTF-16LE standard; 320**
 - meaning of (table); 754
- UTF-8 standard; 320**
 - meaning of (table); 754

UTFDataFormatException class; 565

utilities

miscellaneous; 623–657

utility methods

Collections class; 597

Object class; 99–101

System class; 665–666

UUID class; 656–657

clockSequence method; 657

equal method; 657

fromString method; 657

getLeastSignificantBits method; 656

getMostSignificantBits method; 656

nameUUIDFromBytes method; 656

node method; 657

randomUUID method; 656

timestamp method; 657

toString method; 657

type 1 (time-based), accessing; 657

variant method; 657

version method; 656

UUID (Universally Unique Identifier)

term definition; 656

variants; 657

versions, accessing, **UUID.version**; 656

V

V (value type)

type variable naming convention; 248

valid method

FileDescriptor class; 541

validateObject method

ObjectInputValidation interface; 557

validation

serialized objects,

ObjectInputValidation interface;
557

Value example class; 252

valueOf method

Enum generic class; 160

integer wrapper classes; 188

static method defined for enums; 153

String class; 318

converting other types to strings with
(table); 316

wrapper classes; 185

values

See also: variable(s)

attributes, term definition; 76

constants, documentation of; 487

equality, reference equality vs.; 101

fields

default; 45

initializing; 44

key/value pair, term definition; 587

Map entries

retrieving, **Entry.getValue**; 589

setting, **Entry.setValue**; 589

numeric

characters, retrieving, **Character.digit**;
193

characters, retrieving,

Character.getNumericValue; 193

localization of, **NumberFormat** class use;
651

parameter; 63–65

pass by value, term definition; 63

primitive, wrapper class handling; 186

primitive types, wrappers as container for;
183

return

doc comment tag for; 485

methods; 62

requirements; 63

term definition; 16

upper bounded wildcard use; 258

shared memory, multiprocessing hardware
impact on; 370

streams of, **Scanner** handling; 641–644

testing for equality; 99–100

text representation, **Formatter** class; 624–
632

typeValue methods, wrapper classes; 186

unsigned, handling with signed types; 166

valueOf methods, wrapper classes; 185

wrapper class, immutability of; 185

values method

Map interface; 589

static method defined for enums; 153

varargs methods; 60–61

overriding design issues; 85

variable(s); 169–178

annotation of; 392–393

array; 173

modifiers for; 174

atomic

java.util.concurrent.atomic
package; 735

java.util.concurrent package; 733–
735

blank final, term definition; 172

class; 44, 45

definition of; 8

referencing; 8

term definition; 14

constant, term definition; 46

declaration of; 170

local; 4

definitely assigned, term definition,

(footnote); 171

environment, term definition; 669

- expression assignment, type compatibility requirements; 90
- final; 171–173
- instance; 44
 - term definition; 14
- local
 - declaration statements; 230
 - declarations; 170
 - declaring; 4
 - loop use; 237
 - to threads, ThreadLocal class use; 382
 - wildcard use; 260
- loop, in for statement; 11
- modifiable, interfaces and inner class use; 149
- number of arguments; 60–61
- object, initialization of; 49
- overview; 3–6
- parameter; 171
- primitive types, wrappers as container for; 184
- reference, object creation use; 49
- shared memory, memory model impact on values visible in; 370–375
- state, initializing; 80
- term definition; 169
- type
 - binding issues; 262
 - collection iteration use; 573
 - in Collections class method
 - declarations; 597
 - generic inner classes; 255
 - generic method declaration use; 261
 - inspection of; 426–427
 - introspection hierarchy figure
 - representation of; 399
 - storage variables different from, (footnote); 169
 - term definition; 4, 248
- volatile
 - monitor locks compared with; 372
 - synchronization alternative; 370
- variable-arity methods**
 - term definition, (footnote); 60
- variant method**
 - UUID class; 657
- variants**
 - UUID; 657
- Vector class; 617**
 - addElement method; 617
 - capacity method; 618
 - capacityIncrement field; 619
 - copyInto method; 618
 - elementAt method; 618
 - elementCount field; 618
 - elementData field; 618
 - elements method; 618
 - firstElement method; 618
 - indexOf method; 618
 - insertElementAt method; 617
 - lastElement method; 618
 - lastIndexOf method; 618
 - removeAllElements method; 618
 - removeElement method; 618
 - removeElementAt method; 617
 - setElementAt method; 617
 - setSize method; 618
 - Stack class extension of; 619
- vector(s)**
 - bit
 - EnumSet use; 596
 - resizable, BitSet class; 632
- Verbose example interface; 121**
 - control flow; 233
- verifier**
 - term definition; 38
- version control**
 - doc comments
 - issues for; 497
 - tag for; 487
 - objects, serialization issues; 557–559
- version method**
 - UUID class; 656
- versions**
 - applications, technical issues; 741–748
 - implementation number, Package methods
 - accessing; 478
 - Java source and target, versions, characteristics; 742
 - specification number, Package methods
 - accessing; 478
 - UUID**
 - accessing, UUID.version; 656
 - creating, UUID.nameUUIDFromBytes; 656
 - creating, UUID.randomUUID; 656
- vertical bar (|)**
 - bitwise inclusive OR operator; 208
 - inclusive OR operator; 20
 - logical inclusive OR operator; 207
- vertical bar vertical bar (||)**
 - conditional OR operator; 20, 207
- view(s)/viewing**
 - arrays, lists, Arrays.asList; 608
 - collection, term definition; 578
 - collections, wrapped collections provision of; 597
 - maps, collections use for; 589
 - snapshot compared with; 578

virtual machine

- See also:* bytecodes; JVM (Java Virtual Machine)
- abort, conditions that can cause; 674
- evolution, technical issues for applications; 741–748
- state, impact on reachability; 464
- term definition; 38
- termination; 672–675

VirtualMachineError class; 281**visibility**

- names, scoping control of; 178
- term definition; 13

void

- reflection access of name; 413–s

Void class; 187

- wrapper type hierarchy, (figure); 187

void keyword

- method declaration use; 15
- method return type; 59
- representation by Class object; 397
- reserved keyword (table); 165
- term definition; 2
- type, conditional expression, compile-time error; 212

volatile keyword

- final keyword vs.; 44
- reserved keyword (table); 165
- synchronization alternative; 370
- variables, monitor locks compared with; 372

voluntary

- rescheduling, threads; 360–362

W**wait**

- blocking queue, specification, BlockingQueue interface; 605

wait method

- Object class; 354, 357
- ReferenceQueue.remove compared with; 460
- Thread class, InterruptedException thrown by; 366

wait/notify mechanism

- Observer/Observable mechanism compared with; 639

waitFor method

- Process class; 668

waiting

- thread completion; 367–369

WAITING constant

- Thread class, State nested enum; 384

wakeups

- spurious, term definition; 358

warnings

- @deprecated tag impact on; 486
- unchecked, when emitted; 745

weak references

- WeakHashMap class; 570, 597
- WeakHashMap class**; 592–593
- clear method; 593
- get method; 593
- Map interface implemented by; 590
- not Cloneable nor Serializable; 570
- overview; 570
- put method; 593
- reference queues and; 459
- remove method; 593
- size method; 593
- weak reference use; 458

weakly consistent iterator

- ArrayBlockingQueue use; 605
- ConcurrentLinkedQueue interface; 607
- EnumMap use; 596
- EnumSet use; 596
- LinkedBlockingQueue use; 605
- term definition; 596

weakly reachable/weak references

- term definition; 456
- uses for; 457
- DataHandler example class; 458
- WeakHashMap use; 593

WeakReference class; 455

- constructor use of ReferenceQueue objects; 457
- garbage collector actions; 456
- WeakHashMap use; 593

WhichChars example class; 635**while**; 235

- See also:* flow of control
- for vs.; 10
- if vs., synchronization implications; 355
- loop flow of control mechanism; 9
- loop mechanism; 5
- reserved keyword (table); 165

whitespace

- character, wildcard for matching; 322
- components and use; 5
- term definition; 164
- testing for, Character.isWhitespace; 195

whitespaceChars method

- StreamTokenizer class; 534

widening

- conversion, term definition; 91

widening primitive conversions

- term definition; 216

width

format specification; 23
 combining with precision specification;
 24

indicator, term definition; 625

wildcard(s)

See also: CharSequence interface; escape
 characters; generic; java.util.regex
 package; Matcher class; parameterized
 types; Pattern class; regular
 expressions; special symbols

abstract type parameter

bounded, term definition; 32
 term definition; 31

unbounded, term definition; 32

asterisk (*), match multiple characters; 322

backslash **b** (\b), boundary match beginning
 of a word; 322

backslash **d** (\d), match any digit; 322

backslash **s** (\s), match any whitespace
 character; 322

basic permissions use; 680

boundary matchers, match beginning or end
 of a sequence; 322

bounded

lower bound, generic type use; 257

lower, term definition; 257

term definition; 257

upper bound, generic type use; 257

capture; 264–266

caret (^)

boundary match beginning of a line; 322
 negations; 322

dot (.), match single characters; 322

dot (.) asterisk (*), match zero or more
 characters; 322

further reading; 757

generic types; 256–260

introspection hierarchy figure representation
 of; 399

method restrictions; 260

references, Constructure.newInstance,
 handling; 425

regular expression matching use; 321

square brackets (□), match sets; 322

types, restrictions on parameter passing use;
 265

unbounded, term definition; 257

WildcardType interface; 428

WildcardType interface; 428

getLowerBounds method; 428

getUpperBounds method; 428

Williams, H. H.

quotation; 246

Williams, Peter

quotation; 499

Woolf, Virginia

quotation; 566

word(s)

boundary, pattern matching; 328

matching characters that are not part of; 328

parsing text into, with BreakIterator class;
 712

swapping, pattern matching use; 328

wildcard for matching beginning of; 322

wordChars method

StreamTokenizer class; 534

working directory

ProcessBuilder encapsulation of; 670

specifying; 670

wrapper classes

converting primitive values to strings of; 23

primitive types, term definition; 4

wrapper(s)

See also: method(s); primitive types

boxing conversions; 198–199

checked; 601–602

classes, primitive types; 166

collections; 597

compareTo method; 186

constructors; 185

common to all; 185

conversion, of primitive types to; 91, 183–
 184

equals method; 187

fields, common to all; 184–187

floating-point; 191

hashCode method; 187

integer

decode method; 189

parseType method; 188

valueOf method; 188

MAX_VALUE field; 186

methods, common to all; 184–187

methods common to all; 184–187

MIN_VALUE field; 186

parseType method; 187

primitive type, advantages of; 183

programming with; 183–200

SIZE field; 186

synchronized

collection classes use; 602–604

concurrent collections and; 602–607

iteration; 604

multithreaded design use of interfaces;
 352

term definition; 602

term definition; 183

toBinaryString method

Integer class; 189

Long class; 189

wrapper(s) (*cont.*)

- toHexString method
 - Integer class; 190
 - Long class; 190
- toOctalString method
 - Integer class; 189
 - Long class; 189
- toString method; 187
 - Integer class; 189
 - Long class; 189
- TYPE field; 186
- type hierarchy, (figure); 183
- valueOf methods; 186
- unmodifiable; 601
- value immutability; 185
- valueOf method; 185

write method

- Buffered output stream; 519
- OutputStream; 505
- Writer; 510–511

WriteAbortedException class; 565

writeBoolean method

- DataOutput interface (table); 537

writeByte method

- DataOutput interface (table); 537

writeBytes method

- DataOutput interface; 538

writeChar method

- DataOutput interface (table); 537

writeChars method

- DataOutput interface; 538

writeDouble method

- DataOutput interface (table); 537

writeExternal method

- Externalizable interface; 561

writeFloat method

- DataOutput interface (table); 537

writeInt method

- DataOutput interface (table); 537

writeLong method

- DataOutput interface (table); 537

writeObject method

- HashMap class; 554
- ObjectOutputStream class; 551

Writer class; 510–511

- append method; 511
- close; 511
- flush; 511
- OutputStream contrasted with; 511
- synchronization strategy; 515
- write methods; 510–511

Writer classes

- filter use; 517

writeReplace method

- Object class; 557

writers

- term definition; 500

writeShort method

- DataOutput interface (table); 537

writeTo method

- ByteArrayOutputStream class; 522
- CharArrayWriter class; 523

writeUnshared method

- ObjectOutputStream class; 551, 560

writeUTF method

- DataOutput interface (table); 537

writing

- See also:* I/O; input; output; reading
- booleans, DataOutput.writeBoolean; 537
- bytes of data; 505
- characters; 510–511
 - CharArrayWriter class; 523
- files
 - byte, FileOutputStream class; 541
 - character, FileWriter class; 541
- floating-point numbers
 - DataOutput.writeDouble; 537
 - DataOutput.writeFloat; 537
- implementations
 - collection; 611–616
 - iterator; 609–611
- integers
 - DataOutput.writeInt; 537
 - DataOutput.writeLong; 537
 - DataOutput.writeShort; 537
- random access files; 541
- strings, StringWriter class; 524

WWW (World Wide Web) consortium

- further reading; 756

X

X example class; 82

X example interface; 123

x, X format specifier

- integer conversion use; 627

X500

- java.security.sasl package; 732

XOR (exclusive OR)

- logical, equality operations use; 206

xor method

- BitSet class; 634

XOR (^) operator; 20

- bitwise operations; 208
- logical operations; 207
- precedence; 222

Y

Y example class; 82

Y example interface; 123

yield method

Thread class; 360

Z

Z example interface; 124

Zappa, Frank, quotations; 117

zero (0)

floating-point conversion use (table); 628

integer conversion use (table); 627

negative, testing for; 203

ZIP file format

JAR file format based on; 735

java.util.zip package; 736

ZipEntry class; 736

ZipFile class; 736

ZipInputStream class; 736

ZipOutputStream class; 736

ZLIB compression

Deflater class; 737

DeflaterOutputStream class; 737

Inflater class; 737

InflaterInputStream class; 737

Then the bowsprit got mixed with the rudder sometimes...
—Lewis Carroll, *The Hunting of the Snark (an Agony in Eight Fits)*