

Note: page numbers in **boldface** type indicate key terms.

## Special Characters

- > (greater than operator), 177
- < (less than operator), 176
- >= (greater than or equal operator), 177
- <= (less or than equal operator), 176
- { } (braces)
  - program structure, 18
  - defining scope, 224–225
- != (not equal operator), 176
- % (remainder), 256, 292–293, 343
- && (and operator), 232
- /\* (multi-line comment), 82
- // (single line comment), 18
- /\*\* (documentation comment), 83
- \* (asterisk), 400
- + (plus sign), 311
- ++ (increment), 242
- += (additive assignment operator), 242
- / (division), 92, 235, 292
- == (equal operator), 176
- (minus sign), 343
- = (assignment operator), 19, 176, 177
- [ ] (brackets), 349
- || (or operator), 232
- . (dot), 14
- ; (semicolon), 14, 15, 20

## A

- absolute path, **478**
- abstract class, **640**
- abstract methods, **639**, 639–640
- Abstract Windowing Toolkit (AWT), **92**, 92–93
- abstractions, **3**, 3–4
  - pseudocode, 139
  - raising level, 55–56
- AbstractModel method, 702–703

- access modifiers, **148**
- accessor method(s), **286**, 286–289
  - implementing, 359–361
- Accessor Method pattern, 320
- addresses, reference variables, **404**, 404–406
- algorithms, **116**, 766
  - stepwise refinement. *See* stepwise refinement
  - 2D array, 563–565
- aliasing, 406–409, **407**
  - dangers of aliases, 407–409
- allocating arrays, 543
  - 2D arrays, 565–566
- ampersand (&), and operator, 232
- and operator
  - Boolean, **231**
  - Java (&), 232
- animation, 317–318
- anonymous classes, **677**, 677–678
- API (application programming interface), **743**
- architecture, 588, 588–595, 766
  - creating CRC cards, 591–592
  - developing class diagram, 595
  - developing scenarios, 592
  - identifying classes and methods, 589–590
  - walking through scenarios, 592–595
- arguments, **14**
  - passing, 401
- arrays, 519–575
  - accessing specific elements, 522–524
  - allocation, 543
  - creating, 541–547
  - declaring, 542–543
  - dynamic. *See* dynamic arrays; partially filled arrays
  - elements, 521
  - files compared, 540–541
  - finding extreme elements, 533–534
  - index, 521
  - initialization, 544–547

- multi-dimensional. *See* multi-dimensional arrays
- partially filled. *See* partially filled arrays
- passing and returning, 547–550
- patterns, 572–574
- of primitive types, 558–561
- processing all elements in, 527–528
- processing matching elements, 528–529
- searching for specified elements, 529–532
- sorted, inserting into, 552–553
- sorting. *See* sorting arrays
- swapping elements, 525–526
- visualizing, 521–522

ASCII character set, 794–796

assertions, **624**

Assign a Unique ID pattern, 383–384

assignment statements, **192**

- with loops, 191–193

asterisk (\*)

- comments, 83
- multiplicity, 400

attributes, **4**

- assigning, 651
- implementing with instance variables, 275–276
- software objects, 4, 5–6, 13
- types, 13

avenues, **9**

AWT (Abstract Windowing Toolkit), **92**, 92–93

## B

becker library, 243n, 797–814

Big Brothers/Big Sisters, 520

blank final, **305**

blocks, **481**

- temporary variables, **224**, 224–225

body of `when` statements, **174**

Boole, George, 175n

Boolean expressions, **173**, 231–238

- combining, 231–236
- De Morgan's laws, 237
- evaluating, 175–176, 233–234
- legal, form of, 232–233
- negating, 175
- predicates, 187

- short-circuit evaluation, 238
- simplifying, 236–237

boolean method, 350, 436, 440, 444, 470

- `File` class, 479, 480

Boolean operators, 231–232

boolean type, 224, 344–345

`BorderLayout` strategy, 683

bottom factoring, 248–249, **249**

bottom-up design, 133

bottom-up implementation, **133**

bounding boxes, **98**

braces { }, temporary variables, 224–225

brackets ([ ]), nesting objects, 349

breakpoints, **311**

buffering, **465**

bugs, **34**. *See also* debugging

built-in queries, 174–175

byte code, **29**

byte integer type, 338

byte streams, **501**, 503

## C

Capek, Karel, 9n

capitalization of identifiers, 81

caret (^), compile-time errors, 31

Cascading-`if` pattern, 261

cascading-`if` statements, nesting statements, 227–230, **229**

case of identifiers, 81

cast, **664**

Catch an Exception pattern, 452

`char` method, 350

`char` type, 345–347

`Character` class, class methods, 372–373

character input streams, 501–502

character output streams, 502–503

character streams, **501**

checked exceptions, **426**, 427–428

`City` method, 797–799

class(es), **6**, 6–7

- abstract, 640
- anonymous, 677–678
- assigning methods, 652–653
- client, 640

- closed for modification, 67
- collaborating. *See* collaborating classes
- concrete, 640
- developing to specified interface, 378–379
- extending. *See* extending classes
- identifiers, 81
- identifying, 412–413, 589–590, 648–651
- immutable, 612–614
- implementing accessor methods, 359–361
- implementing command/query pairs, 361–366
- inner, 735–737
- modifying versus extending, 305–306
- multiple, using, 394–398
- mutable, 612
- names, 80
- null values, 398
- objects versus, 7
- open for extension, 67
- passing arguments, 401
- reimplementing, 396–397
- relationships, 413
- returning object references, 401–402
- setting up relationships between, 493–494
- single, using, 392–494
- temporary variables, 401
- wrapper, 446–447
- writing, 358–366
- class diagrams, **7**
  - developing, 595
  - lists, 438–439
  - Robot class, 12–15
- class methods, 368–374
  - Character class, 372–373
  - main method, 374
  - Math class, 369–372
  - Test class, 373–374
- class relationships, 649–651
- class variables, **366**, 366–368
  - assigning unique ID numbers, 368
  - guidelines, 368
  - initialization, 791
- client(s), **4**, **400**
- client class, **640**
- clone method, 665–669
  - implementing, 666–667
  - shallow copies versus deep copies, 667–669
  - using, 666
- closed for modification, **67**
- closing files, **461**, 464
- code
  - byte, 29
  - commenting out, 83
  - duplication, putting in helper methods, 608–609
  - packages, 26
  - pseudocode, 138–139
  - quality, writing. *See* writing quality code
  - sample, 744
  - self-documenting, 188
- cohesion, complexity of programs, **615**, 618
- collaborating classes
  - diagraming, 399–400
  - GUIs, 447–449
- collaborators, **591**
- collections, 431–447. *See also* lists; maps; sets
  - foreach* loops, 437
- color chooser, **71**
- column(s), formatting numbers, 342–343
- column-major order, **563**
- combining Boolean expressions, 231–236
  - error common in, 236
  - operator precedence, 234–235
  - operators, 231
- command(s), **4**
  - correctness, 86
  - Draw a Picture, 105–106
  - meaning, 86
  - preconditions, 86
  - software objects, 4, 6
  - specification, 86
  - testing, 330–332
- command interpreter(s), **486**, 486–495
  - implementing, 487–492
  - separating user interface from model, 492–495
- Command Interpreter pattern, 510–511
- Command Invocation pattern, 42
- command/query pairs, implementing, 361–366
- comment(s), **81**, 81–84
  - documentation, 83–84

- multi-line, 82–83
- single-line, 81–82
- commenting out code, **83**
- comparison operators, **176**, 176–177
- compilers, **29**
- compile-time errors, **30**, 30–32
- compiling programs, **29**, 29–32
  - compile-time errors, 30–32
  - without an IDE, 495–496
- complexity of programs, 615–621, 766
  - cohesion, 615, 618
  - coupling, 615, 620–621
  - encapsulation, 615, 616–617
  - information hiding, 615, 619
- components, GUIs, **37**. *See also* graphical user interfaces (GUIs)
- composition, **400**
- computational science, 768
- concatenation, **347**
- concept maps, **45**, 45–46
- concrete class, **640**
- console, reading from, 480–481
- console window, **310**
- constants, **285**
- constraints, **683**
- constructor(s), 13–14
  - extending classes, 59–62
  - implementing, 70
  - powerful, 610–611
- Constructor pattern, 105
- content pane, **37**
- contract, **623**
- control characters, 466
- controllers, **449**
- controllers, GUIs, 698–700
  - building, 709–726
  - event objects, 737–738
  - implementing, 717–719
  - inner classes, 735–737
  - integrating with views, 738–739
  - registering, 719–720
  - variations, 735–740
  - writing and registering controllers, 716–720
- correct programs, **584**
- count-down loop(s), **192**, 192–193

- Count-Down Loop pattern, 204–205
- Counted Loop pattern, 262
- Counting pattern, 259
- coupling, complexity of programs, **615**, 620–621
- CRC, **591**
  - designing cards, 591–592

## D

- dangling **else**, **250**, 250–251
- dash (–), data availability methods, 471
- data, keeping together with processing, 611–612
- data acquisition methods, **467**, 467–471
- data availability methods, **471**, 471–472
- DateTime class, 394–395
- De Morgan, Augustus, 237
- De Morgan’s laws, 237
- debugger, **311**, 311–312
- debugging, **34**
  - debugger, 311–312
  - printing expressions, 310–311
  - stepwise refinement, 135
- declaration statements, **20**
- declaring
  - arrays, 542–543
  - instance variables, 302
- deep copies, **667**, 667–669
- defensive programming, 621–624
  - assertions, 624
  - design by contract, 622–624
  - exceptions, 621–622
- definitions, methods, overriding, 87–89
- deleting elements in arrays, 553
- delimiters, **467**
- design by contract, **623**
- detail, **615**
- development cycle, **595**, 595–599
- development process, **586**, 586–599
  - defining requirements, 587–588
  - designing architecture, 588–595
  - iterative development, 595–599
- diagramming collaborating classes, 399–400
- dialog boxes, 503
- Direction method, 799
- direction variable, 283–286

- discrete structures, 765
- Display a Frame pattern, 44
- displaying images from files, 506–507
- documentation, 606–607
  - API, 743
  - becker library, 243n, 797–814
  - external, 84–85
  - Robot class, 26–29
- documentation comments, **83**, 84–85
- dot (.), messages, 14
- double method, 470
- double type, 338
- do-while loops, 242–243
- Draw a Picture command, 105–106
- drawing using loops, 251–257
  - loop counter, 252–253
  - nesting selection and repetition, 253–257
- dynamic arrays, 551–558
  - combining approaches, 557–558
  - partially filled. *See* partially filled arrays
  - resizing, 554–557

## E

- E method, 436
- easy to learn GUIs, **626**
- Eclipse project, 311
- Edison, Thomas, 34
- effectiveness of GUIs, **625**
- efficiency of GUIs, **625**
- Eiffel programming language, 623n
- Either This or That pattern, 202–203
- elements
  - of arrays, **521**
  - of collections, **431**
- else-clause, **183**
- encapsulation, **25**, 615
  - complexity of programs, 615, 616–617
- encoding, 793
  - Unicode, 794–796
- engaging interfaces, GUIs, **625**
- enumeration(s) (enumerated types), 355–358, **356**
- Enumeration pattern, 382–383

- equal sign (=)
  - equal operator, 176
  - statements, 19
- equals method, overriding, 663–665
- Equals pattern, 688
- equivalence, **410**
  - testing for, 410–411
- Equivalence Test pattern, 450–451
- error(s)
  - avoiding with stepwise refinement, 134–135
  - checking user input, 481–484
  - compile-time, 30–32
  - debugging. *See* debugging
  - intent (logic), 30, 33–34
  - run-time, 30, 32–33
  - testing for, with stepwise refinement, 135
  - user, GUI forgiveness of, **628**
- error messages, 11
- error tolerance of GUIs, 626
- Error-Checked Input pattern, 509–510
- escape sequences, **346**, 346–347, 796
- evaluating expressions, **175**, 175–176
- evaluation diagrams, **233**, 233–234
- event(s), **716**, 716–717
- event objects, **716**, 737–738
- example programs, 15–25
  - multiple objects, 24–25
  - program listings, 17–18
  - sending messages, 20
  - setting up initial situation, 19–20
  - situations, 15–16
  - tracing programs, 20–22
- exception(s), **424**, 424–431
  - checked, 426, 427–428
  - defensive programming, 621–622
  - handling, 426–428
  - propagating, 428–429
  - reading a stack trace, 425–426
  - throwing, 424–425
  - unchecked, 426
- Exception class, 424–425
- exponents, **338**
- expressions, **175**
  - evaluating, 175–176

Extended Class pattern, 102–103  
 extending classes, 56–78, **59**, 100–102  
   adding services, 62–64  
   form of extended classes, 59  
   implementing constructors, 59–62  
   implementing methods, 64–66  
   implementing objects, 69–73  
   modification versus extension, 67  
   vocabulary, 58–59  
 extensions, **477**  
 external documentation, 84–85  
 extreme elements, finding in arrays, 533–534

## F

factory method(s), **342**, 660–661  
 Factory Method pattern, 689  
 fence-post problem, 212–213  
 fields, **460**  
 file(s)  
   arrays compared, 540–541  
   closing, 461, 464  
   displaying images from, 506–507  
   manipulating, 479–480  
   opening, 461, 462–463  
   processing, 463  
   reading, 461–462  
   specifying locations, 478–479  
   structure, 466–472  
   writing, 464–466  
 File class, 477–480  
   filenames, 477  
   manipulating files, 479–480  
   specifying file locations, 478–479  
 file formats, **475**, 475–477  
 File method, 479  
 filenames, 477  
 final keyword  
   instance variables, 284–285  
   parameter variables, 300  
   temporary variables, 294  
 final situation, **16**  
 five E's, **625**, 625–626  
 Flasher method, 799–800

Flasher subclass, 76  
 flexibility  
   choosing implementations, 679–680  
   increasing with interfaces, 669–680  
 float type, 338  
 floating-point numbers, **338**, 338–340  
 flow of control, **62**, 62–63  
 FlowLayout strategy, 680–681  
 focus, GUI views, 721  
 fonts, GUI views, 721–725  
 for statements, 239–242  
   examples, 240–242  
   form, 239–240  
 foreach loops, **437**  
   arrays, 528  
   collections, 437  
 forgiveness of GUIs for user errors, 628  
 format specifiers, **343**  
 format string, **343**  
 formatting numbers, 341–343  
   columnar output, 342–343  
   NumberFormat object, 341–342  
 frames, **35**, 35–37  
   content pane, 37

## G

garbage, **409**  
 garbage collection, **409**  
 goToNextRow method, 181–183  
 graphical user interfaces (GUIs), **35**, 34–39,  
   697–758  
   adding components, 37–39  
   animation, 317–318, 569–572  
   AWT and Swing, 92–93  
   building and testing models, 705–709  
   building views and controllers, 709–726  
   collaborating classes, 447–449  
   controllers. *See* controllers, GUIs  
   design principles, 626–628  
   designing, 709–710  
   developing classes to specified interface,  
     378–379

- displaying images from files, 503, 506–507
- drawing using loops, 251–257
- extending, 92–102
- file choosers, 503, 504–506
- frames, 35–37
- helper methods, 151–155
- identifying listeners for components, 741–743
- implementing, 377–378
- informing user interface of changes, 379–380
- invoking methods, 100
- iterative design, 625–626
- Java, 374–380
- laying out components, 711–713
- layout managers, 680–686
- learning to use components, 740–747
- libraries of components, 448
- making graphical components interactive, 750–756
- models, views, and controllers, 698–700
- overriding methods, 97–99
- painting components, 747–749
- patterns, 700
- quality, 624–628
- repainting, 312–318
- scaling images, 196–200
- sequence diagrams, 733–735
- setting up model and view, 700–705
- specifying methods, 375–377
- steps for building, 700
- views. *See* views, GUI
- graphics, 767
- GregorianCalendar class, 394
- GridBagLayout strategy, 683
- GridLayout strategy, 681–682
- GUIs. *See* graphical user interfaces (GUIs)

## H

- handling things, 12
- hanging, **213**
- harvestIntersection method, 179–181
- Has-a (Composition) pattern, 449–450
- has-a relationships, **400**

- hashing, **441**
- helper classes, delegating work to, 614–615
- helper method(s), **120**
  - declaring parameters, 153
  - GUIs, 151–155
  - making private, 608
  - nesting statements, 227
  - putting duplicated code in, 608–609
  - using parameters, 153–155
- Helper Method pattern, 155–156
- high-fidelity prototypes, **625**
- host name, **460**
- human-computer interaction, 767

## I

- icons
  - changing size, 75
  - transparency, 75–76
- identifiers, **79**, 79–81
  - capitalization, 81
- if statements, 169–171
  - flowcharts, 169
  - general from, 173
  - nesting statements, 225–226
  - semantics, 174
  - syntax, 174
  - then-clause, 173
  - while statements compared, 168–174
- if-else statements, 183–186
  - else-clause, 183
  - example using, 184–186
  - then-clause, 183
- images from files, displaying, 506–507
- immutable classes, **612**, 612–614
- implementing
  - constructors, 70
  - objects, extending classes, 69–73
  - services, 70–71
- implicit parameters, 63–64, **64**
- indenting programs, 79
- IndexOf method, 355



- indices
  - arrays, **521**, 560–561
  - strings, **350**
- infinite loops, **213**, 213–214
- infinite recursion, **88**
- information hiding, complexity of programs, **615**, 619
- information management, 768
- inheritance, **59**, 635–640
  - abstract classes, 639–640
  - adding new methods, 637–639
  - choosing between interfaces and, 646
  - polymorphism via, 635–640
- inheritance hierarchy, **68**
- inherited methods, 87–92
  - method resolution, 90–92
  - overriding method definitions, 87–89
  - side effects, 92
- initial situation, **16**
  - setting up, 19–20
- initial value, **219**
- initializing
  - array elements, 544–547
  - class variables, 791
  - instance variables. *See* initializing instance variables
  - objects, 74
  - parameter variables, 792
  - temporary variables. *See* initializing temporary variables
  - 2D arrays, 565–566
- initializing instance variables, 302–303, 791
  - using parameters, 298–300
- initializing temporary variables, 792
  - delaying, 294–295
- inner classes, **735**, 735–737
- input, **461**
- input streams, **500**
- insertion point, **466, 721**
- instance(s), **6**
- instance variable(s), **276**
  - accessing, 278–280
  - accessor methods, 286–289
  - declaring, 276–277, 302
  - direction, 284–286
  - extending classes, 300–305
  - final, blank, 305
  - final keyword, 284–285
  - implementing attributes with, 275–276
  - initializing. *See* initializing instance variables
  - lifetime, 276
  - maintaining, 303
  - making private, 609–610
  - modifying, 280–282
  - parameter variables compared, 289, 306–307
  - static keyword, 285
  - temporary variables compared, 289, 306–307, 308–310
  - using, 303–304
- Instance Variable pattern, 319–320
- instantiation, **6**, 6–7
- int integer type, 338
- int method, 350, 436, 440, 444, 470
- integer(s)
  - ranges, 338
  - types, 337–338
- integer division, **292**
- intelligent systems, 768
- intent errors, **30**, 33–34
- interaction, **615**
- interfaces, **376**, 669–680
  - choosing between inheritance and, 646
  - flexibility in choosing implementations, 679–680
  - increasing flexibility using, 669–680
  - mixin, 673–674
  - polymorphism via, 643–645
  - sorting in Java library, 673
  - Strategy pattern, 674–679
  - using, 671–674
- intersection(s), 9–10
  - factoring out differences, 146–147
- Intersection method, 800–802
- invoking methods, 100
- IP addresses, **460**
- IPredicate method, 802–804
- is-a relationships, **400**
- iterative approach, 595–599



## J

.jar files, 499–500  
 Java Archive, **499**, 499–500  
 Java library, sorting, 539–540, 673  
 Java Program pattern, 40–41  
*Java Tutorial*, 744  
 javadoc tool, 84–85  
 JFileChooser, 503, 504–506  
 JFrame object, 36  
 JPanel object, components, 37–39  
 justification, columnar number format, 343

## K

key(s), **431, 530**  
   maps, 431  
   multiple, sorting using, 676–677  
   searching using, 530  
 keyboard focus, **721**  
 keywords, **79**, 80

## L

layout(s), **680**  
 layout managers, **680**, 680–686  
   BorderLayout strategy, 683  
   FlowLayout strategy, 680–681  
   GridBagLayout strategy, 683  
   GridLayout strategy, 681–682  
   nesting layout strategies, 684–686  
   SpringLayout strategy, 683  
 learning, ease of, of GUIs, 626  
 left justification, **343**  
 less than operator (<), 176  
 less than or equal operator (<=), 176  
 lexicographic order, **351**, 351–352  
 libraries, 495–500  
   compiling without an IDE, 495–496  
   creating and using a package, 497–499  
   .jar files, 499–500  
   Java, sorting, 539–540, 673  
 lifetime of instance variables, **276**

Light class, 77–78  
 Light method, 804–805  
 Linear Search pattern, 573–574  
 Liskov, Barbara, 645  
 lists, **431**, 432–439  
   adding elements, 433–434  
   class diagrams, 438–439  
   construction, 432–433  
   *foreach* loops, 437  
   getting, setting, and removing elements, 434–435  
   processing elements, 436–438  
 local variables. *See* temporary variable(s)  
 logic errors, **30**, 33–34  
 logical negation operator, **175**  
 logical operators, 231–232  
   precedence, 234–235  
 long integer type, 338  
 long method, File class, 480  
 loop(s), **174**. *See also* while loops  
   assignment statements, 191–193  
   count-down, 192–193  
   do-while, 242–243  
   drawing. *See* drawing using loops  
   *foreach*, 437, 528  
   guidelines on use, 246  
   infinite, 213–214  
   nested, 253  
   repetition, 253–257  
   when statements, 174  
   while-true, 243–246  
 loop counter, 252–253  
 Loop-and-a-Half pattern, 257–258  
 loop-and-a-half problem, **213, 246**  
 low-fidelity prototypes, **625**

## M

main method, 704–705  
   class methods, 374  
   multiple, 335–337  
 maintainable programs, 586  
 maintaining instance variables, 303  
 mantissa, **338**

- maps, **431**, 441–445
    - construction, 442–443
    - methods, 443–444
    - processing elements, 444
  - Mark II computer, 34
  - matching elements, processing in arrays, 528–529
  - Math class, class methods, 369–372
  - memory, **404**, 404–406
  - messages, **11**, **14**
    - sending, 20
  - method(s), **62**
    - abstract, 639–640
    - ArrayList class, 435–436
    - assigning to classes, 652–653
    - defining, 85–86
    - helper. *See* helper method(s)
    - implementing, 64–66, 414–423, 653–655
    - inherited. *See* inherited methods
    - invoking, 100
    - keeping short, 607–608
    - names, 81
    - overloading, 298
    - overriding, 97–99, 298, 411, 662–669
    - private, 147–150
    - protected, 147–150
    - signatures, 87
    - specifying with interfaces, 375–377
    - stepwise refinement. *See* stepwise refinement
  - method resolution, **90**, 90–92
  - Meyer, Bertrand, 623n
  - Miller, George A., 607
  - minus sign (-), format specifier, 343
  - mixin interfaces, **673**, 673–674
  - models, **2**, 2–9, **448**
    - abstractions, 3–4
    - creating using software objects, 4–7
    - definition, 2
    - GUIs. *See* models, GUIs
    - overview, 2–4
    - robots, 7–9
    - separating user interface from, 492–495
  - models, GUIs, 698–700
    - building and testing, 705–709
    - infrastructure, 701–703
    - setting up, 700–705
  - Model-View-Controller pattern, 448–449, 700, 756–757
  - monospaced fonts, 722
  - move messages, 11
  - move method, direction, 286
  - moving, 11
  - multi-dimensional arrays, 562–569
    - allocating and initializing 2D arrays, 565–567
    - arrays of arrays, 566–569
    - 2D array algorithms, 563–565
  - multi-line comments, **82**, 82–83
  - multiple keys, sorting using, 676–677
  - multiple objects, 24–25
  - multiple robots, 140–141
    - with threads, 142–146
  - Multiple Threads pattern, 156–157
  - multiplicity, **400**
  - mutable classes, **612**
- ## N
- name(s). *See also* identifiers
    - commands, 86
    - parameters, 300
  - Named Constant pattern, 318–319
  - naming conventions, 80–81
  - natural language, **138**
  - negating predicates, **175**, 175–176
  - negations, simplifying, 236–237
  - nested loops, **253**
    - avoiding, 607
  - nesting layout strategies, 684–686
  - nesting statements, 225–230, **226**
    - cascading-if statements, 227–230
    - examples using if and while, 225–226
    - helper methods, 227
  - net-centric computing, 766
  - newline character, **466**
  - not equal operator (!=), 176
  - null, **398**
  - null values, 398
  - NumberFormat object, 341–342

- numeric types, 337–344
  - converting between, 340–341
  - floating-point, 338–340
  - formatting numbers, 341–343
  - integer, 337–338
  - shortcuts, 344
- numerical methods, 768

## O

- object(s)
  - classes versus, 7
  - event, 737–738
  - identifying, 412–413, 589–590, 648–651
  - initializing, 74
  - instantiation, 6–7
  - multiple, 24–25
  - representing records as, 472–477
  - software. *See* software objects
- object diagrams, **5**
- object equality, **410**
  - testing for, 410–411
- object identity, **410**
- Object Instantiation pattern, 41–42
- object references, returning, 401–402
- object-oriented design methodology, 412–424, 588–595, 647
  - identifying objects and classes, 412–413, 589–590, 648–651
  - identifying services, 414–423, 652–655
  - solving the problem, 423–424, 655–661
- object-oriented programming languages, **4**
- Once or Not at All pattern, 201
- Open File for Input pattern, 508
- Open File for Output pattern, 508
- open for extension, **67**
- opening files, **461**, 462–463
- operands, **232**
- operating systems, 766
- operators, **231**
  - Boolean, 231–232
  - comparison, 176–177
  - logical. *See* logical operators

- precedence, 234–235, 787–792
- primitive and reference types, 351
- or operator, **231**
- organization, 766
- origin, **9**
- output streams, **500**
- overloading methods, **298**
- overriding methods, 97–99, 298, 411
  - Object class, 662–669
  - toString method, 349

## P

- package(s), **26, 495**
- package statement, 497–499
- paintComponent method, 312–318
- painting
  - GUI components, 747–749
  - repainting, 312–318
- @param tag, 83
- parameter(s), **14**, 189–196
  - assignment statements, 191–193
  - declaring, 153
  - stepwise refinement, 193–196
  - using, 153–155
  - while statements with, 190
- parameter variables, 296–300
  - final keyword, 300
  - initializing, 792
  - initializing instance variables using, 298–300
  - instance variables compared, 289, 306–307
  - name conflicts, 300
  - temporary variables compared, 296–298, 306–307
- Parameterized Method pattern, 158–159
- Parameterless Command pattern, 105
- partially filled arrays, **551**, 551–554
  - deleting elements, 553
  - problems, 554
  - sorted, inserting into, 552–553
- Pascal, Blaise, 568
- Pascal's Triangle, 568

- paths
  - absolute, 478
  - relative, 478–479
- patterns, 39–44, 102–106
  - Accessor Method, 320
  - Assign a Unique ID, 383–384
  - Cascading-if, 261
  - Catch an Exception, 452
  - Command Interpreter, 510–511
  - Command Invocation, 42
  - Constructor, 103–104
  - Count-Down Loop, 204–205
  - Counted Loop, 262
  - Counting, 259
  - Display a Frame, 44
  - Either This or That, 202–203
  - Enumeration, 382–383
  - Equals, 688
  - Equivalence Test, 450–451
  - Error-Checked Input, 509–510
  - Extended Class, 102–103
  - Factory Method, 689
  - Has-a (Composition), 449–450
  - Helper Method, 155–156
  - Instance Variable, 319–320
  - Java Program, 40–41
  - Linear Search, 573–574
  - Loop-and-a-Half, 257–258
  - Model-View-Controller, 448, 700, 756–757
  - Multiple Threads, 156–157
  - Named Constant, 318–319
  - Object Instantiation, 41–42
  - Once or Not at All, 201
  - Open File for Output, 508
  - Parameterized Method, 158–159
  - Parameterless Command, 105
  - Polymorphic Call, 686–687
  - Predicate, 260–261
  - Process All Elements, 452–453, 573
  - Process File, 508–509
  - Query, 259–260
  - Scale an Image, 205
  - Sequential Execution, 43
  - Simple Predicate, 203–204
  - Strategy, 687
  - Template Method, 157–158
  - Temporary Variable, 258–259
  - Test Harness, 381
  - Throw an Exception, 451
  - toString, 381–382
  - Zero or More Times, 202
- percent sign (%), format specifier, 343
- pickThing messages, 12
- picture elements, **36**
- pipe (|), or operator, 232
- pixels, **36**
- plus operator (+), 311
- points, **723**
- Polymorphic Call pattern, 686–687
- polymorphism, 633–690
  - abstract classes, 639–640
  - adding new methods, 637–639
  - assigning attributes, 651
  - assigning methods to classes, 652–653
  - choosing between interfaces and inheritance, 646
  - class relationships, 649–651
  - examples, 640–642
  - identifying objects and classes, 648–651
  - identifying services, 652–655
  - implementing methods, 653–655
  - interfaces. *See* interfaces
  - overriding methods in Object class, 662–669
  - substitution principle, 645–646
  - via inheritance, 635–640
  - via interfaces, 643–645
  - without arrays, 661–662
- positionForNextHarvest method, 181
- postconditions, **623**, 623–624
- precedence, operators, 234–235, **235**, 787–792
- preconditions, **86**, **622**, 622–624
- precision, **339**
- predicate(s), **175**, 186–188, **589**
  - Boolean expressions, 187
  - negating, 175–176
  - using non-Boolean queries, 188
- Predicate pattern, 260–261
- primary key, **676**
- primitive(s), **132**
- primitive type(s), **337**

primitive type arrays, 558–561  
   double, 558–559  
   indices, 560–561  
 printing expressions, 310–312  
   System.out object, 310–311  
 private keyword, 148–150  
 problem solving, 116  
 Process All Elements pattern, 452–453, 573  
 Process File pattern, 508–509  
 processing files, 463  
 processing streams, **501**  
 professional issues, 768  
 program(s), **4**  
   compiling, 29–32  
   complexity. *See* complexity of programs  
   correct, 584  
   evaluating with users, 598  
   example. *See* example programs  
   form, 25–26  
   hanging, 213  
   identifiers, 79–81  
   keywords (reserved words), 79, 80  
   maintainable, 586  
   modifying with stepwise refinement, 136–138  
   quality software, 584–586  
   reliability, 584  
   running, 32–34  
   special symbols, 79  
   style, 78–85  
   testable, 586  
   tracing. *See* tracing programs  
   understandability, 585  
   usability, 584  
 program fragments, **169n**  
 programming defensively. *See* defensive programming  
 programming fundamentals, 765  
 programming languages, 767  
   Eiffel, 623n  
   object-oriented, 4  
 prompt, **85, 481**  
 Prompt class, 485  
 prototyping, **625**  
 provider streams, **501**

pseudocode, **138**, 138–139  
 public keyword, 63–64, 148–150  
 putThing method, 178–179

## Q

quality code, writing. *See* writing quality code  
 quality software  
   programmer's perspective, 585–586  
   user's perspective, 584  
 queries, **4**, 330–337  
   built-in, 174–175  
   integer, testing, 176–177  
   multiple main methods, 335–337  
   non-Boolean, predicates, 188  
   return statements, 223  
   side effects, 224  
   size, 198–199  
   software objects, 4–5  
   storing results, 222–223  
   String object, 350–352  
   testing commands, 330–332  
   testing queries, 332–335  
   writing, 223–224  
 Query pattern, 259–260

## R

random access, **541**  
 ranges, integers, **338**  
 reading, **461**  
   from console, 480–481  
   files, 461–462  
   records as objects, 472–475  
 records, **460**  
   representing as objects, 472–477  
 recursion, infinite, 88  
 refactoring, **586, 597**  
 reference variables, 403–411, **404**  
   aliases, 406–409  
   garbage collection, 409  
   memory, 404–406  
   testing for equality, 410–411

refinement, stepwise. *See* stepwise refinement  
 registration, controllers, **719**, 719–720  
 relative paths, **478**, 478–479  
 reliability of programs, **584**  
 remainder operator (%), **256**, **292**, 292–293  
 requirements, **587**  
   defining in development process, 587–588  
 reserved words, **79**, 80  
 resizing arrays, 554–557  
 responsibilities, **590**  
 responsiveness of GUIs, 626–627  
 return statements, 223  
 right justification, **343**  
 roads, **9**  
 Roberts, Eric, 243  
 robot(s), services, 10–12  
 Robot class  
   class diagram, 12–15  
   documentation, 26–29  
 Robot class diagram, 12–15  
   services, 13, 14–15  
 Robot constructor, 13–14  
 Robot method, 805–808  
 Robot object, attributes, 13  
 RobotRC method, 808–809  
 RobotSE method, 809–810  
 row-major order, **563**  
 running programs, 32–34  
 run-time errors, **30**, 32–33

## S

sample code, 744  
 sans serif fonts, 722  
 Scale an Image pattern, 205  
 scaling images, 196–200  
   size queries, 198–199  
 scenarios, **592**  
   choosing, 596  
   developing, 592  
   implementing, 596–597  
   walking through, 592–595  
 scientific notation, **338**  
 scope, temporary variables, 224–225, **225**  
 searching, **530**  
   keys, 530  
   for specific elements in arrays, 529–532  
 secondary key, **676**  
 Selection Sort  
   coding, 536–538  
   overview, 535–536  
 self-documenting code, **188**  
 semantics, **174**  
 semicolons (;)  
   messages, 14, 15  
   statements, 20  
 sequence diagrams, **733**, 733–735  
 Sequential Execution pattern, 43  
 serifs, **722**  
 servers, **4**, **400**  
 services, **4**, 13, 14–15, 62–64. *See also*  
   command(s); queries  
   flow of control, 62–63  
   identifying, 414–423, 652–655  
   implementing, 70–71  
   implicit parameters, 63–64  
   public keyword, 64  
   robots, 10–15  
   void keyword, 64  
 sets, **431**, 439–441  
   construction, 439  
   limitations, 441  
   methods, 440  
   processing elements, 440–441  
 shallow copies, **667**, 667–669  
 short integer type, 338  
 short-circuit evaluation, **238**  
 shortcuts, numeric types, 344  
 side effects, **92**, **224**  
 signatures, methods, **87**, **298**  
 Sim method, 810–811  
 Simple Predicate pattern, 203–204  
 SimpleBot class, testing, 282–283  
 simplifying Boolean expressions, 236–237  
   De Morgan’s laws, 237  
   negations, 236–237  
 simulation  
   program execution. *See* tracing programs  
   pseudocode, 139

- single-line comments, **81**, 81–82
- sink, **500**
- situations, 15–16
  - final, 16
  - initial. *See* initial situation
- size of icons, 75
- size queries, 198–199
- social issues, 768
- software engineering, 768
- software objects, **4**
  - attributes, 4, 5–6, 13
  - class diagrams, 7
  - classes, 6–7
  - commands, 4, 6
  - modeling robots using, 12–15
  - queries, 4–5
- Sojourner, 8–9
- solving problems, 116
- sorting arrays, 534–540
  - coding Selection Sort, 536–538
  - without helper methods, 538–539
  - Java library, 539–540
  - Selection Sort overview, 535–536
- sorting using Java library, 539–540, 673
- sound, 429–431
- source, **500**
- source code, **17**, 17–25
- spaghetti code, **243**
- special symbols, **79**
- specification,
  - programs, **587**
  - commands, **86**
  - defining in development process, 587–588
- SpringLayout strategy, 683
- stack traces, 425–426, **426**
- state, **6**
- state change diagrams, **6**
- statement(s), **20**. *See also specific statements*
- static keyword, instance variables, 285
- static variables. *See* class variables
- stepwise refinement, **117**, 117–138
  - error avoidance, 134–135
  - future modifications, 136–138
  - helper methods, 120
  - identifying required services, 118–119
  - parameters, 193–196
  - style, 247
  - testing and debugging, 135
  - understandability of programs, 133–134
- Strategy pattern, **674**, 674–679, 687
  - anonymous classes, 677–678
  - applications of strategy objects, 678–679
  - Comparator interface, 674–676
  - sorting with multiple keys, 676–677
- streams, **500**, 500–503
  - byte, 501, 503
  - character input, 501–502
  - character output, 502–503
  - processing, 501
  - provider, 501
- street(s), **9**
- Streetlight method, 811–812
- StreetLight subclass, 76
- String method
  - Scanner class, 470
  - File class, 479
- String type, 347–355
  - Java support, 347–348
  - overriding toString method, 349
  - querying strings, 350–352
  - transforming strings, 352–353
- stroke, **200**
- structure of files, 466–472
- structured programming, **243**
- stubs, **124**
- style, 246–251
  - positively stated simple expressions, 247–249
  - stepwise refinement, 247
  - visual structure of code, 250–251
- styles of fonts, 722
- subclasses, **58**
- subject, **589**
- substitution principle, **645**, 645–646
- superclasses, **58**, 59
- swapping array elements, 525–526
- Swing, **92**, 92–93
- syntax, **174**



## T

tab stops, **347**  
tags, **83**  
Template Method pattern, 157–158  
temporary variable(s), **218**, 218–225,  
290–296, 401  
    boolean type, 224  
    counting Things on an intersection, 219–220  
    delaying initialization, 294–295  
    final keyword, 294  
    initializing. *See* initializing temporary variables  
    instance variables compared, 289, 306–307,  
    308–310  
    parameter variables compared, 296–298,  
    306–307  
    scope, 224–225  
    storing results of queries, 222–223  
    tracing code, 221–222  
    writing queries, 223–224  
Temporary Variable pattern, 258–259  
Test class, class methods, 373–374  
test harness, **330**  
Test Harness pattern, 381  
test reversal, 247–248  
testable programs, 586  
testing  
    commands, 330–332  
    for equality, 410–411  
    queries, 332–335  
    SimpleBot class, 282–283  
then-clause, **173**, **183**  
Thing class  
    extending, 67–78  
    inheritance hierarchy, 68  
Thing method, 812–813  
ThingBag, 13  
this keyword, 279  
threads, **100**  
    multiple robots, 142–146  
Throw an Exception pattern, 451  
throwing an exception, **424**, 424–425  
tokens, **467**  
top factoring, **249**

top-down design, **132**. *See also* stepwise  
refinement  
toString method, overriding, 349, 662  
toString pattern, 381–382  
tracing programs, **20**, 20–22  
    pseudocode, 139  
    temporary variables, 221–222  
transparency, icons, 75–76  
turning, 11  
turnLeft messages, 11  
2D array(s), 563–565  
    allocating and initializing, 565–566  
2D array algorithms, 563–565  
    printing every element, 563–564  
    summing columns, 565  
    summing every element, 564–565  
types, attributes, **13**

## U

unchecked exceptions, **426**  
understandability, 585  
    GUIs, 627–628  
    positively stated simple expressions, 247–249  
    stepwise refinement, 133–134  
Unicode, 794–796  
usability of programs, **584**  
use cases. *See* scenarios  
user(s), 480–485  
    checking input for errors, 481–484  
    error, GUI forgiveness, 628  
    Prompt class, 485  
    reading from console, 480–481  
user interfaces. *See* graphical user interfaces  
(GUIs)

## V

V method, 444  
validation, **56**  
values, maps, **431**  
variable(s), **19**, 273–322, **274**  
    class (static). *See* class variables

- identifiers, 81
- instance. *See* instance variable(s)
- names, 80
- non-numeric types. *See* boolean type; char type; **String** type
- numeric types. *See* numeric types
- reference. *See* reference variables
- selecting, rules of thumb for, 307
- temporary (local). *See* temporary variable(s)
- variable declarations, **19**, **276**, 276–277
- verification, **56**
- views, **449**
- views, GUIs, 698–700
  - building, 709–726
  - infrastructure, 703–704
  - integrating with controllers, 738–739
  - making graphical components interactive, 750–756
  - multiple views, 726–735
  - painting components, 747–749
  - refining, 721–725
  - setting up, 700–705
  - updating, 713–715
  - view patterns, 725–726
- visual computing, 767
- visualizing arrays, 521–522
- void** keyword, 63–64
- void** method, 436, 440, 444

## W

- walk-throughs, **592**
- wall(s), **10**
- Wall** method, 814
- waterfall model, **598**
- when statements, 174

- while loops, 212–218
  - avoiding common errors, 212–214
  - four-step process for constructing, 214–218
- while statements, 171–173
  - flowcharts, 169
  - general form, 173–174
  - if statements compared, 168–174
  - nesting statements, 225–226
  - using with parameters, 190
- while-true loops, 243–246
  - example, 245–246
  - form, 244
  - structured programming, 243
- white space, programs, 78–79
- whitespace, **466**
- working directory, **463**
- working incrementally, 744–747
- wrapper classes, **446**, 446–447
- writing files, **464**, 464–466
- writing quality code, 606–615
  - avoiding nested loops, 607
  - delegating work to helper classes, 614–615
  - document classes and methods, 606–607
  - keeping data and processing together, 611–612
  - keeping methods shore, 607–608
  - making helper methods private, 608
  - making instance variables private, 609–610
  - putting duplicated code in helper methods, 608–609
  - writing immutable classes, 612–614
  - writing powerful constructors, 610–611

## Z

- Zero or More Times pattern, 202

















