

Index

* 1180
+ 189, 1180
:= 448
< 189
<= 1177, 1178
?= 593
@ 327, 328, 882
- 189
-> 588, 1186
'Class (Ada 95) 1094

A

- a posteriori scheme for handling
 abnormal cases 800–801
- a priori scheme for handling abnormal
 cases 798–800
 obstacles 799–800
- Abbott, Russell J. 744
- ABCL/1 999, 1034
- abnormal case 411–438, 797–801,
 1089–1091
 a posteriori scheme 800–801
 a priori scheme 798–800
- abnormal case style for exceptions
 1091
- Abrial, Jean-Raymond xi, 100, 160,
 330
- absolute negative rule 667
- absolute positive rule 666–667
- abstract
 data type, see abstract data type
 machine 792
 object, see under object
 precondition, see under
 precondition
 side effect 757
 state 756–758
 syntax tree 115, 1038
- abstract** (Ada 95) 1093
- abstract data type 18, 121–162, 166,
 171, 172, 173, 174, 216, 229, 231,
 318, 338, 352, 373–377, 399, 500,
 722, 733, 734, 792, 862, 907, 1100,
 1101, 1176, 1193
 advanced topics 148–159
 and abstract machines 792
 and analysis 907
 and assertions 373–377
 and classes 142–147, 373–377
 and information hiding 144–145
 applications beyond software 147–
 148
 complete example specification
 139
 consistency 155
 formal description 129–141
 genericity 131–132, 318
 goals 122–129
 specifying axioms 135–137
 specifying the functions 132–135
 specifying types 130–131
- abstracted module, see under module
- abstracting 860, 930
- abstraction
 elevating the level 861
 for GUI (Graphical User Interface)
 1068–1071
 how to find
 see also finding under class
 how to find abstractions 699–704,
 754
 in methodology rules 669
 varieties of class abstraction 860
 versus precision 905–906
 versus specialization 858–859
- abstraction function 230, 375, 756
- access control 1047
- access, see uniform access
- accessor function 135
- accommodate* 625, 629
- accommodation* 625, 629
- ACCOUNT** 329, 1046
 Ada 95 1093
- account, see bank account
- Accounts* (Ada 95) 1093
- ACCOUNT1** 491, 513
- ACCOUNT2** 491, 492, 513
- ACE Consortium 79
- Ace file 198, 200, 201, 393, 1146
- Ace, see Ace file, Lace
- Acrobat, see under Adobe
- action* 848
- ACTIONABLE** 1040
- activate** (Simula) 1124
- active data structure 774–796
 internal view 786–792
- active object 957–960
 incompatible with inheritance
 959–960
- active process 1123
- ActiveX 8, 67, 955
- actors model 1033
- actual generic parameter, see actual
 under parameter
- actual reclamation 302
- actual, see under argument, parameter
- Ada 46, 53, 56, 61, 84, 89, 90, 99, 100,
 176, 209, 211, 225, 265, 269, 270,
 285, 315, 392, 410, 415–416, 443,
 447, 507, 510, 564, 587, 588, 616,
 716, 876, 892, 897, 980, 1104, 1106,
 1108, 1130, 1137, 1161, 1167–1188
 adding classes 1098
 by default, means 1983 version of
 the language; see also Ada 95
 concurrency mechanism 980
 context and history 1079–1080
 exception handling 410, 415–416,
 438, 1088–1091
 exception rule 1090
 genericity 587–588
 towards an object-oriented version
 1096
- Ada 95 443, 564, 566, 1080, 1092–
 1096, 1097, 1131, 1137, 1161
 assessment 1094–1095
 example 1092–1094
 object-oriented mechanisms 1092–
 1094
- add* 762
- add_vertex* 627, 633

- address* 1039
- ADJ group 160
- Adobe xiv
- Acrobat 108
- Acrobat Reader xiv
- adopt* 311
- adopting an object 311
- ADT, abbreviation for “abstract data type”; see under that term.
- Advanced Curriculum principle 936
- advertiser* 912
- advisory rule 667–668, 823
- after* 755, 782
- Agammemnon 269
- aggregation 258, 907
- Agha, Gul 1033, 1034
- Aho, Alfred V. 20, 100, 745
- airline reservation system 676
- airplane 521–522
- AIRPLANE** 521, 831
- Algol 49, 64, 211, 396, 447, 876, 897, 1080, 1102, 1107, 1114, 1126, 1138
- without further qualification, means Algol 60; see also the next two entries
- Algol W 64, 353, 455
- Algol 68 61, 211
- alias* 440
- aliasing 265–270, 277
- in software and elsewhere 268–269
- semantics 266–267
- all-or-nothing garbage collection 306–307
- Alpha 1152
- Alphard 99, 1079
- AltaVista 78, 1060
- Amako, Katsuya 922
- America, Pierre 642, 1034
- AMPHIBIOUS_VEHICLE** 522
- analysis 217, 271, 506, 725, 732–733, 903–922, 936, 941, 1162, 1198
- and the software process 906
- as negotiation 906
- class 732–733
- contribution of object technology 907
- describing business rules 913
- domain analysis 947
- goals 903–906
- methods 917–919
- requirements on an analysis method and process 904–905
- role in education 936, 941
- tasks 903–904
- traditional analysis methods 906
- TV station programming example 907–913
- analytical modeling 1122
- ancestor 464, 500, 1193
- anchor 601–602
- artificial 1179–1187
- anchor* 1179, 1181
- anchor** (construct not retained) 631
- anchor-equivalent 603, 631
- anchored declaration 598–604, 618, 629, 630–633, 1177, 1178
- a static mechanism 604
- and covariance 630–633
- and once functions 652–653
- rules 603
- when not to use 603–604
- Anchoring approach to the covariance issue 630–633, 639, 642
- and** 454
- between assertion clauses 337
- and then** 454, 570, 578
- Andromaque 671
- animals, distinguishing from plants 841
- anomaly, see under inheritance
- anorexia 314
- ANSI (American National Standards Institute) 1079, 1097, 1102, 1107, 1131
- ANSWER** 688, 694
- antonomasia 269
- ANY** 580–582, 590, 592, 976, 1044, 1187
- APL 754, 876
- applet 956
- application 1064
- as a class 690–693, 1076
- APPLICATION** 688, 689, 690, 690–692, 694, 734
- application builder 1065, 1066, 1072, 1073, 1075, 1076
- application developer 1064, 1066, 1067, 1068, 1070, 1071, 1072, 1073, 1076
- applicative 145, 159, 351–353
- ArchiText 715
- argument 444–446, 651, 764–770
- actual 89, 184, 261, 444–446
- definition 444
- formal 261, 444–446, 637
- definition 444
- polymorphic 637
- ideal number of arguments for a feature 764–770
- in out** (in non-O-O approaches) 765
- passing 444–446
- see also attachment
- polymorphic 637
- see also operand, option
- term used for routines (see also parameter) 322
- to a command, see under command to a once routine 651
- type redefinition 621–641
- Ariane 4 410
- Ariane 5 389, 410
- Aristotle 843, 864, 866
- ARITHMETIC** 180
- Arnold, Ken 222, 939, 1137, 1139
- array 87, 325–328, 372–373, 470–472
- as object 325–326
- properties 326
- slice 383
- static (Fortran, Pascal) 45
- two-dimensional 330
- ARRAY** 325, 330, 348, 372, 441, 470, 530, 540, 583, 584, 844, 882, 1165, 1188
- array_down* 123
- ARRAY_TABLE** 504
- array_up* 123
- ARRAYED** 846
- ARRAYED_LIST** 1165
- ARRAYED_QUEUE** 1188
- ARRAYED_STACK** 530, 539, 540, 837, 844, 845, 846
- ARRAYED_TABLE** 831
- ARRAY2** 330
- arsonist 201–202
- artificial anchor, see artificial under anchor
- ASCII** 847, 850, 851
- assembly 197, 198
- automatic process 200, 1146
- assembly language 1099
- assertion 23, 334, 337–410, 569–580, 617, 779–780, 899, 907, 917, 930, 1193
- and abstract data types 373–377
- and analysis 907, 917
- and encapsulation 779–780
- and generalization 930
- and inheritance 569–580

- and redeclaration 481–482, 570–580
 - expressive power 399–403
 - for documentation 389–392
 - for writing correct software 389
 - in C, C++ etc. 334
 - instruction 378–380
 - monitoring at run time 392–399
 - not a control structure 346–347
 - not for input checking 345–346
 - style rules 899
 - using a comment 399
 - using in practice 389–398
 - Assertion Argument rule 997
 - Assertion Evaluation rule 402
 - Assertion Redeclaration rule 573, 578, 580
 - Assertion Violation rule 346
 - asset 521–522
 - ASSET* 521
 - assignment 448
 - see also attachment
 - assignment attempt 29, 478, 591–595, 617, 1068, 1134, 1193
 - rationale 591–592
 - semantic specification 593–594
 - using properly 594–595
 - associated_state* 692
 - association 907
 - Association Dijonnaise des Tapeventres 161
 - Association of Simula Users 732
 - asynchronous call 966, 1193
 - AT&T Bell Laboratories 12, 328, 802, 1106, 1132, 1135
 - Atkinson, Colin 1034
 - Atkinson, M. 1061
 - attachment 242–244, 261–265, 280, 444
 - and equality 264–265
 - copy attachment 262–263
 - definition 262
 - hybrid 263–264, 412, 445
 - polymorphic 467–470
 - reference attachment 262–263
 - source 262
 - target 262
 - attempt, see assignment attempt
 - attribute 173–176, 1193
 - adding or removing attributes to classes of persistent objects 1045–1046
 - and postconditions 579
 - client privileges 206–208
 - constant 203, 644–645, 884–885
 - default value, see under initialization
 - exporting 205–208
 - in relational databases 1048
 - no precondition 579
 - once? 660
 - redefined from function 491–492, 579
 - versus function 204
 - Austin Mini 811
 - AUTHORS* 1048, 1049
 - automatic update 31
 - available 184, 191, 447
 - available* 299, 316
 - AVL trees 72
 - Avotins, Jon 1160
 - axioms of an abstract data type 135–137
- ## B
- B-tree 82
 - B.O.N., see Business Object Notation
 - Bachelard, Gaston 672–673
 - back* 782, 790
 - Backslash* 645, 653
 - Backus, John 1102
 - BAG* 857
 - Baillon, Henri 843
 - balance* 369, 1046
 - balance_enquiry* 980
 - Balter, R. 1034
 - Bancilhon, François 1061
 - bandwidth 48
 - bank account 56, 364, 368, 369, 472, 491, 492, 513, 1046, 1047
 - BANK_ACCOUNT* 364, 368, 370
 - baritone 1143
 - base class, see under class
 - Base libraries 351, 357, 456, 543, 555, 710, 796, 802, 1146, 1149, 1150, 1152, 1157, 1165–1166
 - Basic 1099, 1106
 - Basic Construct of object-oriented computation 183, 611–612
 - basic triangle of computation 101–103, 964
 - basic type, see basic under type
 - basic_store* 1038
 - batch version of a system 108–109
 - Baudoin, Claude 35, 934, 937
 - Bauhin, Caspar 864
 - BBC (British Broadcasting Corporation) 315
 - BCPL 1106, 1111
 - be versus have 812–814
 - see also is-a relation
 - Beck, Kent 740
 - Beethoven, Ludwig van 1141, 1143
 - before* 782
 - behavior class 72, 503–504, 688, 772, 850, 961, 1002, 1004, 1010, 1021, 1029, 1030, 1031, 1193
 - Bell Labs, see AT&T Bell Laboratories
 - bells and whistles 12
 - Ben-Ari, Mordechai 1033, 1034
 - Bench 1148–1149
 - Berkeley, see California Museum of Paleontology
 - Bert, Didier xi, 100, 330
 - Bertino, Elisa 1061
 - Beta 1137, 1139
 - Bezault, Éric 674, 1160
 - BI_LINKABLE* 597, 624, 625, 797, 1165, 1166
 - bibliography 1203–1224
 - Bielak, Richard 277, 807, 1034, 1061
 - Big Bang 194–196
 - Big Green Button 695
 - Biggerstaff, Ted J. 99
 - binary distribution 79
 - binary search 380–381
 - binary search tree 85, 92, 1188
 - binary tree 1188
 - BINARY_FILE* 1174
 - BINARY_SEARCH_TREE* 1188
 - BINARY_TREE* 97, 604, 1007, 1188
 - BINARY_TREEI* 1008
 - binding
 - dynamic 29, 63, 85, 480–482, 570–577, 1071, 1174, 1175, 1195
 - and assertions 570–580
 - and efficiency 507–515
 - and interactive applications 1071
 - implementation 482
 - overhead 509
 - in C++ 1133
 - static 509–515, 1202
 - as optimization 509–515, 1147
 - static and dynamic binding in C++ 513–515

- versus typing 619–621
 - BIRD* 627, 841, 843
 - Birtwistle, Graham M. 35, 1138
 - block (Simula) 1123
 - block structure 49–50, 281, 282
 - blocking** (in short forms) 996
 - blocking object 996
 - Bobrow, Daniel G. 1139
 - body
 - of class in Simula 1118
 - Boehm, Barry W. 19, 20, 878
 - boldface 900
 - boldface italics 900
 - BON, see Business Object Notation
 - Booch, Grady 744, 918, 922, 1097, 1135
 - book 221–222, 277
 - Bookman font 901
 - BOOKS* 1048, 1049
 - BOOK1* 221
 - BOOK2* 223
 - BOOK3* 226
 - BOOLEAN* 171, 220, 644, 1172
 - BOOLEAN_MATRICES* (Ada) 1180
 - BOOLEAN_MATRIX* 1179
 - BOOLEAN_RING* 1179
 - BOOLEAN_RING_ELEMENT* 1180
 - booting procedure 197
 - Borges, Jorge-Luis 18, 672
 - Borland 211, 515, 1130, 1143
 - Borland Pascal 1101, 1131, 1137
 - boundary cases 353
 - bounded queue, see bounded under queue
 - BOUNDED_ARITY_TREE* 604
 - BOUNDED_BUFFER* 967, 968, 986, 992, 993, 994, 996, 1022, 1031
 - BOUNDED_LIST* 710
 - BOUNDED_QUEUE* 992, 994, 1031
 - BOUNDED_STACK* 576
 - Boussard, Jean-Claude 948
 - Bouy, Reynald 1160
 - BOX* 857
 - Box symbol (Ada) 1083
 - BOY* 623, 634
 - Brachman, Ronald J. 517
 - Brandon, D.H. 878
 - Breu, Ruth and Michael 863
 - Bright, Walter 515, 670
 - Britannicus 1135
 - browser 1153
 - browsing 32, 1156–1159
 - Bruce, Kim B. 629, 642
 - Bruno, John 1034
 - bubble 1150
 - Budde, Reinhardt 642
 - buffer 990–992, 1021–1022
 - BUFFER* 980, 981
 - BUFFER_ACCESS* 1021
 - BUFFER_MANAGER* (Ada) 1091
 - Buffon, Georges-Louis Leclerc, Comte de 843, 865
 - bug 18, 409
 - Build graphical application builder 1076, 1149, 1150, 1160
 - bulimia 314
 - Bull 1079
 - Burstall, Rod M. 160
 - Business Card principle 990, 1020, 1031, 1035, 1036
 - business card scheme 974, 975, 984, 989, 990, 993, 996, 1002, 1011
 - Business Object Notation 271, 464, 517, 914, 919–922, 930, 1150
 - business rule 913
 - for a TV station 913
 - BUTLER* 1005
 - button 533, 1071, 1074
 - BUTTON* 511, 1017
 - buttonhole 533–534
 - Buxton, John M. 99
 - bytecode 956, 1136, 1145, 1147, 1149
- C**
- C 49, 56, 61, 89, 176, 211, 225, 230, 265, 266, 269, 270, 278, 282, 285, 306, 315, 327, 328, 333, 334, 386, 439, 441, 442, 443, 447, 507, 509, 510, 670, 714, 716, 737, 742, 753, 758, 876, 877, 891, 956, 1056, 1065, 1067, 1106–1111, 1130, 1131–1137, 1144, 1146, 1161
 - and exceptions 414–415
 - and memory management 295
 - calling object-oriented mechanisms from C 1144
 - compilation 1146
 - efficiency 510
 - emulating object technology 1106–1111, 1112
 - history 1106–1107
 - need for comments 891
 - Obfuscated 876
 - object-oriented extensions 1131–1137, 1138
 - structure type 1109
 - C++ 35, 46, 56, 100, 208, 209, 211, 239, 278, 294, 295, 305, 306, 310, 334, 443, 444, 513–515, 548, 566, 585, 616, 620, 628, 668, 670, 742, 876, 1050, 1056, 1057, 1099, 1106, 1107, 1132–1135, 1136, 1144, 1161, 1167
 - and memory management 294, 295, 305, 306, 310
 - and novariance 628
 - assessment 1135
 - binding policy 513–515
 - complexity 1134–1135
 - concurrency 1033
 - Obfuscated 876
 - wrapping legacy C++ code 1144
 - CABIN_BUTTON* 1017
 - cache or recently accessed objects 1056
 - CAD-CAM (computer-aided design and manufacturing of engineering products) 1051, 1054
 - calculator 522
 - California Museum of Paleontology (Berkeley) 864, 865, 868
 - call 24, 183–184, 447–448
 - asynchronous 966, 1193
 - calling object-oriented mechanisms from C and other languages 1144
 - chain 418
 - dual semantics under concurrency 966
 - external 311, 439–444
 - and garbage collection 311
 - function 453
 - optimizing 208–209
 - qualified 186–187, 447
 - separate 967
 - synchronous 966, 1202
 - target 184
 - unqualified 186–187, 447
 - callback 440, 505
 - Campbell, Roy H. 1033
 - can-act-as-a relation 497
 - Cancel button 1074
 - Cannon, H. I. 1139
 - Canonical Reduction rule 158
 - Capability Maturity Model 55
 - capacity* 710, 882
 - CAR* 810

- car metaphor for software 672
- CAR_OWNER* 810, 845, 863
- Cardelli, Luca 629, 641, 642
- Carnegie-Mellon University 1079
- Caromel, Denis 987, 1033
- Carriero, Nicholas 1033
- Carroll, Martin 328
- cartesian product 133, 134, 149, 150, 160, 1052
- cartesian_ready* 760
- Case analysis and design workbench 711, 805, 1150
- Case instruction 449
- CASE tools 271
- case, see letter case
- Cassandra 671
- cast 306, 618, 620, 628, 668, 670, 1133
- Castagna, Giuseppe 629, 642
- casual approach, see under memory management
- CAT (Changing Availability or Type) 638
- catalog of graphical abstractions 1066
- catcall 636–638
 - definition 638
- Catcall approach to the covariance issue 639, 642
- Catcall type rule 637, 639
- categories of object orientation criteria 22
- Cattell, R.G. 1061
- CCS (Communicating Concurrent Systems) 1033
- CD-ROM accompanying this book viii, xiv, 1043, 1076, 1165
- Cecil library 1144
- CELL* 526, 604, 607
- Cépage 715
- Ceres 269
- Ceyx 1131, 1139
- CGI script 1152
- chain
 - call chain, see chain under call
- CHAIN* 567
- challenger 999, 1000, 1027, 1031
- Chambers, Craig 215, 1139
- change in software development 81–82
- change or redo dilemma 59
- change, rule of 814–816
- Changing Availability or Type 638
- CHARACTER* 171, 220, 565, 644
- characteristic function 139
- cheating clients 572
- check** 378–380
- check instruction 378–380, 452
- Check_instruction* 432
- checking input values 345–346
- Chen, Peter P.S. 120
- Cheshire Cat 733
- choice* 686
- choose_initial* 691
- CHORUS 1034
- CII-Honeywell Bull 1079
- CIRCLE* 329, 467, 483, 826, 838, 858, 886
- CIRCULAR_LIST* 710
- CITY* 497, 729
- cladistics 865, 866–868
- cladogram 867
- Clark, David 939
- clash, see name clash
- class 23, 165–216, 1194
 - abstract, same as deferred abstracting 860
 - abstraction 860
 - analysis 732–733
 - as module and type 170, 185
 - avoiding useless classes 720–721
 - base class, see under type
 - basic conventions 177–181
 - behavior, see behavior class
 - categories 731–732
 - consistency 771–773
 - correctness, see correctness of a class
 - deferred 30, 142, 143–144, 165, 482–494, 500–506, 518, 686–688, 1174–1188, 1195
 - definition 486
 - role 143–144, 500–506
 - definition 142
 - descriptor, see class descriptor
 - design 734–735
 - do not confuse with object 165–169, 216
 - does not “perform” something 726–727
 - effective 142, 143, 165, 1195
 - definition 486
 - expanded 256
 - factoring 860
 - finding 117
 - finding the classes 719–746, 754
 - general heuristics 731–740
 - the general method 741–743
 - through reuse 740–741
 - flat form, see flat form
 - generating, see generator
 - generic 320–325, 1197
 - how big? 770–774
 - implementation 733–734
 - deferred 734
 - indexing clause 78
 - interface 747–808, 1197
 - documenting 804
 - recommended style 752–754
 - invariant, see class invariant
 - is a new class necessary? 721–723
 - missing important classes 723–724
 - modeling external objects 732–733
 - name
 - as class descriptor 1043
 - naming 879
 - nesting, see nesting
 - no-command 729–730
 - obsolete 802–803
 - parameterized, same as generic
 - passive 776–779
 - reason for rejecting classes 726–731
 - rejecting inadequate classes 725
 - role 169–170
 - root, see root class
 - set 196
 - simple example 172–177
 - single routine 728
 - size 770–774
 - definition 771
 - flat 771
 - immediate 771
 - incremental 771
 - small 714–715
 - specification 1201
 - the ideal class 730–731
 - universal 580–582
 - validity, see class validity
 - versioning 1054
 - versus record 150–151
 - versus type 324–325
 - wrapper, see wrapper
- class descriptor
 - for C implementation of object-oriented concepts 1110–1111
 - for schema evolution in persistent object structures 1043
- Class Elicitation principle 725
- class invariant 118, 146, 363–410, 413, 465, 570, 579, 647, 784, 785, 952, 982–983, 999, 1022–1024, 1194

- and creation procedures 371
- and Design by Contract 368–369
- and generalization 930
- and inheritance 465, 570
- and manifest constants 647
- and reference semantics 403–406
- implementation invariant 376–377, 409, 532, 756
- role in software engineering 367
- violations 409
- when it must be preserved, when not 366–367
- Class Tool 1153, 1154, 1156, 1159
- class validity 627
- Class-ADT Consistency property 375
- class-valid 627, 636
- Class_invariant* 432
- class_name* 433
- classes préparatoires 941
- classification
 - premature 728–729
- classification, see taxonomy
- classwide operation (Ada 95) 1094
- CLEAR 330
- Clemenceau, Georges 932
- clickable, clickability 1158–1159
- client 51, 118, 119, 175, 182–183, 785–786, 907, 1194
 - and analysis 907
 - being honest with clients 573
 - cheating clients 572
 - definition 182
 - dynamic 572
 - independence 861
 - privileges on an attribute 206–208
 - versus inheritance 812–817
- client-server 953, 968, 1039, 1056, 1149, 1151, 1152
- clone* 245, 247, 274, 275, 276, 303, 582, 584, 880, 976
 - defined in terms of *copy* 247
- cloning, see under object
- CLOS 1131, 1139
- close* 1174, 1175
- CLOSED_FIGURE* 483
- closure
 - for modules, why needed 57
 - see also persistence closure, system closure
- clouds and precipice 905–906
- CLU 46, 90, 99, 100, 806, 1081
- cluster 24, 210, 920, 923–924, 925, 926–928, 1194
 - CLU 1081
 - in the Business Object Notation 920
 - subcluster 923
- cluster** (Lace) 198, 199
- Cluster Model of the software lifecycle 926–928
- Coad, Peter 167, 917, 922
- Coad-Yourdon method 917
- Cobol 165, 442, 737, 742, 876, 1099, 1107
- COBOL 1079
- Codd, E.F. 1048, 1061
- Cohen, Bernard 945, 948
- Cohen, Jacques 316
- cohesion 64
- collect_now* 308, 314
- COLLECTION** 857
- collection, see garbage collection
- collection_off* 308, 312, 314
- collection_on* 308
- Collins, Allen G. 868
- color, use in software texts 901, 1152
- column
 - in relational databases 1048
- comb-like structure 892, 894
- command 135, 699–718, 748, 987, 1073
 - arguments 707–708
 - as a class 699–700
 - button 751
 - composite 529, 716
 - creating a command object 703–704
 - executing 707
 - remembering 702–703
- COMMAND** 71, 699, 714, 721, 724, 731, 734
- command-query separation 748–764, 1029
- Command-Query Separation principle 751, 759
- COMMAND_INSTANCE** 709
- comment 890–891
 - as assertion 399
 - header, see header comment
 - non-header 890–891
- COMMERCIAL** 911, 913
- commercial for a TV station 911–912
- COMMON** (Fortran) 48, 656, 736, 742, 1102–1104
 - garbage 48, 736
- common block, see **COMMON** (Fortran)
- Common Lisp 1131
- communication 977, 979–980
- comp.object* Usenet newsgroup 35, 674
- comp.risks, see Risks forum
- compaction 313
- COMPANY** 913
- COMPANY_PLANE** 521
- COMPARABLE** 523, 590, 727, 831, 832, 1176, 1177, 1178, 1183, 1185, 1186
- comparing objects, see equality under object
- comparing references 244
- compatibility 8, 16, 115, 443–444
- compatible* 913
- compilation technology 1144–1148
 - challenges 1144–1145
 - see also assembly
 - speed 31, 618
- Compilist 897
- complementary formalisms 920
- completeness
 - of a garbage collector 305
 - of a specification 153–159
 - persistence, see closure under persistence
 - sufficient 156–159
- COMPLEX** 647, 760, 858, 1186
- complex number 408, 518
- component manufacturer 297
- component, see reusable software component
- component-level approach to memory management 297–301
- composability 42–43, 48, 50, 54
- composite
 - command, see under command
 - figure, see under figure
 - object, see under object
- COMPOSITE_COMMAND** 716
- COMPOSITE_FIGURE** 528, 1071
- Compostela 152
- compromise in the software process 906
- computation
 - ingredients 101–103
- computational reflection 1130
- ComputerWorld* 14, 1136
- computing time 1123
- concrete state 756–758

- concurrency 102, 951–1036, 1056–1057, 1059–1060, 1091–1092, 1118–1121
 - and inheritance 959–960, 1121–1122
 - examples 1003–1022
 - in Ada 980, 1098
 - in object-oriented databases 1056–1057
 - intra-application 954
 - library mechanisms 972–973, 1027, 1030
 - multi-layer architecture 970
 - proof rule 1022–1024
 - semantic specification 1026–1027
 - summary of mechanism 951–952, 1025–1027
 - summary of the mechanism 1025–1027
 - syntax specification 1025
 - validity constraints 997, 1025–1026
 - validity rules 973–976
- CONCURRENCY** 998, 1000, 1001, 1019, 1027, 1030
- Concurrency Control File 971–972
- concurrent 1194
 - accesses to an object 982–983, 1031
 - precondition paradox 995, 1036
 - see also concurrency, concurrent engineering
- concurrent engineering 924–925
- conditional correctness, conditional proof, see conditional under correctness
- conditional critical region, see conditional under critical region
- conditional instruction 448–449
- configuration management 66, 1042
 - see also versioning under class, object
- CONFIRMATION** 692
- Conflicting Assignments To Actual 446
- conformance 469, 474, 591, 598, 616, 1194
- conforms_to* 582, 640
- conjugate* 600
- CONS** (Lisp) 282
- consistency
 - in naming features and classes 883–884
 - of a library 69
 - of an abstract data type 155
- of analysis models using different views 920
- static-dynamic types 475
- Const_value** 650
- constant 452–453, 643–660, 884–886, 1081
 - attribute, see constant under attribute
 - how to use 645–646, 884–886
 - initialization 656–657
 - manifest 452–453, 646–647, 885
 - and class types 646–647
 - of a basic type 643–645
 - of a class type 646–648
 - of string type 653–654, 657
 - symbolic, see constant under attribute
 - where to put declarations 886
- constant inheritance, see under facility inheritance
- Constantine, Larry 120
- constrained genericity, see under genericity
- constructor (C++) 1133
- constructor function (abstract data type) 135
- consumer, see under reuse; see also client
- CONTAINER** 857
- container data structure 471, 472, 496, 587, 1194
- content-based search tool 1060
- context (in graphical systems) 1072–1076
- Context-Event-Command-State model of interactive graphical applications 1073–1076, 1150
- continuity 44–45, 47, 48, 50, 51, 54, 56, 65, 103
- contour model of programming languages 315
- contract 341–342, 1194
 - see also Design by Contract
- CONTRACT_EMPLOYEE** 853
- contracting 919
 - in BON 919
 - see also Design by Contract
- contravariance 624, 625, 626, 628, 1194
- control (for graphical applications) 1066, 1067
- control structure 346–347
- control structure style for exceptions 1091
- CONTROLLER** (Simula) 1120
- convenience inheritance 824
- convenience, see marriage of convenience
- Cook, William R. 642
- coordinates in GUI (Graphical User Interface) systems 1070
- copy
 - attachment, see under attachment
- copy* 247, 274, 275, 276, 582, 583, 584
- copying, see under object
- CORBA 8, 955, 970
 - CORBA 2 955
- coroutine 1012–1014, 1030, 1036, 1118–1126, 1139, 1140
 - example 1119–1121
 - Simula 1118–1126, 1139, 1140
- COROUTINE** 1013
- COROUTINE_CONTROLLER** 1014
- correct* 681, 682, 686
- correct ADT expression 154
- correct_mismatch* 1044, 1045, 1046
- correction, see under persistence
- correctness 4–5, 16, 52, 331, 332, 369–373, 389, 427–430
 - a relative notion 333–334
 - and exceptions 427–430
 - conditional 4–5, 52, 401
 - of a class 369–373
 - partial 337
 - total 337
 - versus efficiency 394–398
- correctness formula 334–335, 369
- Correctness property 333
- cosmetics 875–879
- Così fan tutte 1000
- count* 777, 882
- COUNTABLE_SEQUENCE** 755
- coupling 64
 - weak 48
- covariance 621–642, 1194
 - Anchoring approach 630–633, 639, 642
 - Catcall approach 636–638, 639, 642
 - Global approach 633–636, 639
- Cox, Brad J. 34, 80, 100, 119, 672, 715, 1112, 1131, 1138
- CPU 965, 1014

CRC cards 740
 creation 231–236, 236–239, 279–316, 518, 752
 and inheritance 465–467, 479–480, 518
 and overloading 239
 by a function 752
 call 236–239
 instruction 232–239, 448
 effect 233, 237
 multiple creation procedures 236–239
 patterns 316
 polymorphic 479–480
 three modes 280–291
 why explicit 235–236
 Creation Inheritance rule 466
 creation procedure 196, 236–239, 371, 430, 647
 and class invariants 371
 and exceptions 430
 using a parent's creation procedure 539–540
 creativity 878–879
 creator 135
 Cristian, Flaviu 438
 criteria of object orientation 21–36
 critical region 978, 979, 980, 984, 990
 conditional 978, 990
 cryptography 953
 CSP (Communicating Sequential Processes) 979, 980, 1033, 1091–1092
 cuckoo 859
 Cunningham, Ward 740
Current 181, 446, 452, 453, 602
 role 185–186
Current as anchor 602
 current demo 927
 current instance, see under instance
 Curry, Gael A. 99
 currying 215, 1076
 cursor 461, 462, 488, 489, 490, 504, 752, 754, 755, 756, 759, 774–796
custom_independent_store 1040
 customer, is always wrong 336, 343, 347, 353, 393, 428, 572
 Cuvier, Georges 864, 865
 Cybele 269
 cycles, first and second (France) 941
 cyclic structures under reference
 counting 303–304

D

Dahl, Ole-Johan 35, 745, 1114, 1138
 Daigakuin 941
 Dami, Laurent 642
 Darwin, Charles 843, 860, 865, 866
 Data Division (Cobol) 737, 742
 data sharing 50
 data structure, see active data structure, container data structure, polymorphic data structure
 data transmission 684
 database 32, 1047–1062, 1198
 engine 1053
 locking 1047, 1054, 1055, 1057, 1061
 avoiding locks in Matisse 1056
 in Versant 1057
 optimistic, see optimistic locking
 long transaction, see long under transaction
 object-oriented 1037, 1050–1062, 1152
 advanced concepts 1058–1060
 an oxymoron? 1058–1060
 engine 1053
 examples of object-oriented database management systems 1055–1057
 fundamental concepts 1050–1055
 threshold model 1053
 query 1049, 1055, 1057
 in object-oriented database management systems 1055
 in Versant 1057
 relational 1037, 1048–1053
 definition 1048
 limitations 1051–1053
 operations 1048–1049
 used with object-oriented software 1050–1053, 1152
 when not applicable 1051–1053
 transaction, see transaction
 using unstructured information in lieu of databases 1060
DATABASE 968
DATE 910
 Date, Chris J. 1048, 1061
 De Cindio, Fiorella 948
 De Moel, Joost 1160
 dead code removal 1147

dead object, see unreachable under object
 deadlock 989–990, 1004, 1031, 1035
debug 452
 debug instruction 452
 debugging 392–398, 1153–1159
 decentralization 7, 498, 643
 declaration
 anchored, see anchored declaration
 syntax 203
 decomposability 40–41, 47, 48, 50, 54
 decomposition
 functional 103–114, 197
 object-based 114–115
 deduction versus induction 859–860
 deep operations, see cloning under object
deep_clone 248, 276
deep_equal 248, 276
deep_import 976, 977, 1035
 default values, see under initialization
default_rescue 430, 1044
 defect 347
 defensive programming 343–345, 1195
deferred 484, 486
 Deferred Class No-Instantation rule 487
 deferred class, see under class
 deferred feature, see under feature
define (C) 266
 definitions (full list) 1189
 Dekleva, Sasha M. 20
 Delphi 211, 515, 1130, 1143
delta (in the Business Object Notation) 920
demand 1000
 demanding style for preconditions 354–357
 DeMarco, Tom 120
 Demeter 269
 Law, see Law of Demeter
 Department of Defense, see US Department of Defense
 dependency analysis for compilation 1146
 dependent 250, 1146
 direct 250
deposits_list 364, 368, 1046
 Deramat, Frédéric 1160
 DeRemer, Franklin D. 20
 derivation, see generic derivation

- Dernbach, Frédéric 1160
- Descartes, René 37, 43, 673
- descendant 464, 1195
- descendant hiding 626–627, 835–843
and subtype inheritance 835–843
avoiding 838–839
using 839–840, 843
why needed 837
- describing objects and their relations 118
- design 150, 506, 725, 734–735, 936, 941, 1162, 1198
class 734–735
reusability 70–71
role in education 936, 941
- Design by Contract 127, 146, 331–410, 411, 419, 435, 569–580, 617, 756, 805, 907, 919, 941, 952, 981, 994, 1022, 1028, 1029, 1090, 1195
and analysis 907, 919
and concurrency 952, 981, 994, 1028, 1029
and inheritance 569–580
and invariants 368–369
in the Business Object Notation 919
middleman 575
role in education 941
subcontracting 576
- design patterns 71–72, 100, 529, 675–718, 735, 745, 759–764, 817, 855, 871–874, 991, 1068
- destructor (C++) 310
- detach** (Simula) 988, 1119
- detachment 283–284
- detection, see under memory management, persistence
- developer 933
- developer exception, see under exception
- developer_exception_code* 434
- developer_exception_context* 435
- developer_exception_name* 434
- development object 1153–1159
- DEVICE** 602, 1173, 1174, 1184
- diagonal* 499, 591
- diagram, see transition diagram
- Diderot, Denis 121, 148
- Dijkstra, Edsger Wybe xi, 72, 122, 160, 316, 347, 407, 664, 665, 667, 678, 750, 835, 1003, 1033, 1135, 1138
- dining philosophers 1003–1006, 1033
- Dioscorides of Anazarbus 864
- direct instance, see under instance
- direct manipulation 1063
- direct mapping 47, 54, 931
- directory 199
- DIRECTORY** 1174
- dirty read 1056
- disciplinary approach 1100, 1101, 1108
- discipline and creativity 878–879
- Disciplined Exception Handling principle 417, 419, 427
- Discrimination principle 655
- DISK** 1173, 1175
- DISPATCHER** 1018
- dispenser 127
- display* 594, 681, 682, 686, 688, 1071
- display* (for a button) 555
- display* (for composite figures) 529
- DISPLAY_ITERATOR** 529
- dispose* 310, 314, 316
- disposing of objects when garbage-collected 310
- distribution formats for reusable software components 79–80
- distribution of knowledge 63
- Dittrich, Klaus 1061
- divide* 762
- DLL (Dynamic Link Library) 440
- do_if* 849
- documentation 14–15, 18, 32, 54–55, 803–805
and generalization 930
external 14
interface 14
internal 14
see also self-documentation system-level 805
- Documentation principle 804
- DoD, see US Department of Defense
- dogmatism in assessing object orientation 21–22
- domain 138, 377, 580
- domain analysis 947
- Don't mode me in 1075
- DOOR** 720
- DOTTED_SEGMENT** 829
- DOUBLE** 171, 220, 522
- downcasting 1134
- drag-and-drop 1156
see also pick-and-throw
- typed 1157
- DRAGOON 1034
- DRIVER** 544, 545
- Drix, Philippe 869
- duality between actions and objects 102, 146
- Dubois, Paul F. xi, 642, 715, 718, 765, 1034, 1112, 1160
- duel 999–1000, 1031
- Dufour, Sylvain 1160
- Duke, Roger 160
- Duncan, Thomas 868
- DURATION** 910
- dynamic 1195
aliasing, see aliasing
allocation, see memory management
binding, see dynamic under binding client 572
IP address 1043
link library, see DLL
typing, see dynamic under typing
- Dynamic Binding principle 511
- DYNAMIC_LIBRARY** 440
- DYNAMIC_ROUTINE** 440

E

- E_CLASS** 169
- ease of use 11–12, 15, 16
- eating car 522
- EATING_CAR** 522
- Eco, Umberto viii, 163
- economic analogy for object-oriented concepts 127
- economics of software procurement 76
- economy 14, 15, 16
- ECOOP concurrency workshops 1034
- editor 724, 1066, 1074–1075
for graphical abstractions 1066
- education, see teaching object technology
- effecting 485, 1195
- effective
class, see effective under class
effective feature, see under feature
efficiency 9–11, 15, 16, 19, 68, 208–209, 307, 327, 394–398, 482, 507–515, 548, 616, 773, 1043, 1146, 1147–1148
and dynamic binding 507–515
and genericity 327

- and inheritance 507–515
- and static typing 616
- of garbage collection 307
- of repeated inheritance 548
- of the compilation process 1144–1146
- of the environment's generated code 1144
- versus correctness 394–398
- egoful design 878
- egoless expression 878
- egoless programming 878
- Eiffel 1162
- Einstein, Albert 672
- electronic collocation 925
- elevator 720
- ELEVATOR** 720, 1016
- elevator system (concurrency example) 1014–1019, 1036
- Eliëns, Anton 34
- Élinck, Philippe 642
- ELKS 1150
- ELLIPSE** 483, 527, 826, 838
- Ellis Island 538
- Ellis, Margaret 328, 668, 1138
- Elmasri, Ramez 1061
- else** 179, 448, 450
- elseif** 449
- embedded SQL 1049
- EMPLOYEE** 853
- empty** 777, 883
- empty structures 353
- emu 859
- emulation 1099–1112
- Encapsulate Repetition 984
- encapsulation 53, 779–780, 1053, 1195
 - and assertions 779–780
 - and databases 1053
 - see also: information hiding; the next three entries.
- encapsulation language 53, 1079–1098, 1099, 1106
- encapsulation level of object-oriented support 1099
- encapsulation of non-object-oriented software 441–443
- end user 109, 1064, 1065, 1071, 1074, 1075
- ending_time** 909, 910
- engine
 - execution 1147
 - for object-oriented database 1053
- ENGINEER** 815, 816, 853
- engineering
 - see forward engineering, rearchitecting, reverse engineering
- engineering schools (France) 941
- ENQUIRY_ON_FLIGHTS** 688
- ensure** 112, 338
- ensure then** 578
- enter** 802
- entity 1196
 - default value, see under initialization
 - operations on generic entities 323–324
 - polymorphic 469, 488
 - precise definition 213
- entity-relationship 120, 737, 742
- enumerated type
 - Ada 1175
 - Pascal 660
- enumerated type, see unique value
- ENVIR** 650
- environment for developing object-oriented software 1143–1160
- ÉPÉE 1034
- epilogue 1161–1162
- equal** 246, 247, 248, 265, 274, 275, 276, 582, 584
- equality, see under object; see also comparison under reference
- EQUIVALENCE** (Fortran) 266
- error 347
 - precise terminology 347–348
- esprit de l'escalier 932
- Euclid's algorithm 877
- European Space Agency 410, 1080
- evaluating candidate decompositions 736
- event
 - in interactive system 1071–1076
 - handling 1072–1076
- EVENT** 1071
- event list 1123, 1124
- event notice 1123, 1124
- event-driven computation 1071–1076, 1196
- EVENT_NOTICE** 1140
- Everham, Edwin McKay III 842, 859, 863
- evolution
 - in biology 859, 865, 866, 867, 868
 - of the language 1144
- schema evolution, see under persistence
- see also change in software development, extendibility
- evolutionary taxonomist 866–868
- exception 25, 241, 411–438, 801, 999, 1088–1091, 1134, 1196
 - abnormal case style 1091
 - advanced exception handling 431–435
 - as object 436–437
 - basic concepts 411–414
 - causes 413–414
 - comparison of mechanisms 1089–1091
 - control structure style 1091
 - definition 412
 - developer 434–435, 1088–1089
 - examples 422–427
 - handling 414–422, 1088–1089
 - history table 420–422
 - in Ada 415–416, 438, 1088–1091
 - in C-Unix 414–415
 - queries 431–433
 - raising 434–435, 1088–1089
 - recovering 423–424
 - role 801
 - sources 412–413
 - taxonomy exception, see exception under taxonomy
- exception** (Ada) 415
- exception handler 1089
- exception to methodology rule 668–669
- EXCEPTION_CONSTANTS** 431
- EXCEPTIONS** 431, 998, 1000, 1001
- execute** (for **APPLICATION**) 691
- execute** (for **COMMAND**) 699
- execute** (for **STATE**) 687
- execute_interruptibly** 1001, 1002
- execute_session** 680, 688
- execute_state** 681, 682
- execution engine 1147
- execution of a system 195, 234
 - remote 1147
- exists** (in the Business Object Notation) 920
- expanded** 254
- expanded client 259
- Expanded Client rule 259
- expanded, see under class, type
- Explicit Interfaces, see under interface
- explicitness 876–877
- export

- see information hiding
 - selective, see selective export
 - to the class itself 193–194
 - unrestricted 191
 - export** 582, 605
 - exported feature, see exported under feature
 - express message 999
 - expression 452–456
 - with operators 453–454
 - extend** 882
 - extendibility 6–7, 15, 16, 115, 441, 644, 735, 768, 905, 1066, 1196
 - and symbolic constants 644
 - in analysis 905
 - in graphical applications 1066
 - extendible** 883
 - extension and specialization 499–500
 - extension inheritance 826–827
 - definition 826
 - external** 373, 440, 1144
 - external call, external software, see external under call
 - external class 458
 - external object, see under object
 - external routine, see external under call
 - extract** (potential feature) 610
- F**
- facility inheritance 532–533, 832–833, 847–851
 - constant 850
 - definition 832
 - forms 850
 - machine 850
 - factor, see under quality
 - factoring 85–88, 860, 930
 - factoring out common behaviors 85–88
 - failure 411–412, 1090, 1196
 - definition 412
 - Failure principle 420
 - False** 220, 452
 - false alarm 417, 1196
 - family, see under module
 - fault 347
 - fault tolerance 424–425
 - Faust 604
 - feasibility study 904, 926
 - feature 24, 90, 143, 172–216
 - call 24, 183–184
 - effect 184
 - classification 173–175
 - clause 191–192, 889
 - header comment 889
 - multiple clauses 191–192, 889
 - deferred 30, 482–494, 686–688, 1174, 1195
 - effective 1195
 - exported 1196
 - see also public
 - feature history, displayed in the environment 1158
 - final name 549
 - frozen 583–585
 - when to use 585
 - how many arguments? 764–770
 - how to recognize various kinds 177–178
 - identifier 189
 - immediate 464, 771
 - infix 189, 327–328, 586
 - inherited 464
 - joining 552–553
 - naming 127–128, 879–884
 - obsolete 802–803
 - of a package 90
 - operator 187–191
 - precursor 493
 - prefix 189
 - renaming 535–540, 834, 1196
 - and redeclaration 538
 - discussion 563–564
 - effects 537
 - replication 544–548
 - secret 192–193
 - selection under repeated
 - inheritance 553–555
 - specification 1201
 - undefining 551–553
 - universal 276, 582
 - using the original version of a
 - redefined feature 493–494, 555–561
 - feature** 177, 210
 - Feature Call principle 186
 - Feature Call rule 473, 591, 592, 594, 613, 634
 - Feature Tool 1153, 1155, 1156, 1158
 - Feldman, Jerome A. 1034
 - Feldman, Michael B. 1097
 - Feldman, Stuart I. 65
 - Few Interfaces, see under interface
 - Feynman, Richard 672
 - field 219, 220–221, 228, 1156, 1196
 - displaying in an object-oriented environment 1156
 - in relational databases 1048
 - simple 220–221
 - figure 1068–1070
 - composite 527–529
 - FIGURE** 329, 472, 480, 482, 483, 505, 528, 591, 658, 858, 1071, 1093
 - file 88, 737–738, 742, 1036, 1139, 1150
 - FILE** 310, 645, 1174
 - FILE_TABLE** 831
 - filter module 345
 - final** 689
 - final name 549
 - finalization
 - of objects (garbage collection), see disposing of objects, under garbage collection
 - optimized form of compilation 1148
 - Finalize 1154
 - finding the classes and objects, see finding under class
 - finding the top 107–108
 - finish** 782
 - Finkelstein, Sheldon 1061
 - Firesmith, Donald 34, 919
 - fitted** 640
 - FIXED_STACK** 503
 - Fixing What is Broken 671
 - fixpoint 635
 - flat form 541–543, 579
 - uses 542–543
 - flat size 771
 - flat-short form 543, 803, 1147
 - flattening, see flat form
 - Flavors 1131, 1139
 - FLIGHT_DATABASE** 1059
 - flight_details** 1059
 - flights** 1059
 - FLOOR** 745
 - FLOOR_BUTTON** 1018
 - flow of information, see data transmission
 - Floyd, Robert W. 407
 - fly** 627, 841, 843
 - focus_line** 838, 839
 - font, use in software texts 900–901
 - forall** (in the Business Object Notation) 920
 - force** 882
 - foresight 629
 - FORK** 1005

- form (Alphard) 99
- formal generic parameter, see formal under parameter
- formal methods 5, 52, 129–162, 1022–1024, 1034
- formal text for requirements 914
- formal texts as a basis for natural-language requirements 916–917
- formal, see under argument, parameter
- formatting of software texts, see layout
- formats for storing objects 1038–1039
- forth* 755, 782, 789
- forth1* (not retained) 788
- Fortran 12, 45, 211, 266, 327, 439, 441, 446, 510, 714, 736, 742, 876, 947, 1081, 1102–1106, 1111, 1161
- efficiency 510
 - emulating object technology 1102–1106
 - Fortran II 1102
 - Fortran IV 1102
 - Fortran 77 657, 1104, 1106
 - Fortran 90 1102
 - Fortran 95 1102
 - history 1102
 - Waterloo 947
- forward engineering 1150
- fragile input 422–423
- FrameMaker 108, 391, 696, 1159
- framework 72
- France
- university curriculum 941
- Franceschi, Fabrice 316, 1160
- Franklin, Benjamin 673
- free* 978
- free memory management 280–287
- Free On-Line Dictionary of Computing 750
- Freeze 1154
- freezing 1145, 1146, 1148
- freezing a feature, see frozen under feature
- FRENCH_DRIVER* 544, 546
- FRENCH_US_DRIVER* 545, 546
- fresh* 298, 299
- friend (C++) 1133
- from* 451
- frozen* 583
- frozen elements of a system 1145
- frozen feature, see under feature
- full* 883
- function 89, 105–106, 174, 203, 447, 1196
- and evolution 108–109
 - call 453
 - categories 134–135
 - creating an object 752
 - domain 580
 - finite 1076
 - in an assertion 400–403
 - more than one 105–106
 - of a system 105–106
 - of an abstract data type 132–135
 - once, see once function, once routine
 - partial, see partial function
 - redefined into attribute 491–492, 579
 - result 179–180, 210–213
 - achieving the effect of multiple results 446, 758
 - rationale for the convention 210–213
 - see also partial function, total function
 - side effect, see side effect
 - total, see total function
 - transition, see transition function
 - versus attribute 204
 - versus object 146
- functional 89, 103–114, 197, 678–684, 714, 1100
- functional level of object-oriented support 1099
- functional variation inheritance, see functional under variation inheritance
- functionality 12–13, 15, 16
- Fusion method 918
- Futatsugi, Kokichi 160

G

- G_swap* 1168
- Gakubu 941
- Gamma, Erich 71, 100, 735, 745
- Gannon, John D. 897, 901
- garbage collection 30, 304–314, 332, 1133, 1196
- a practical environment 312–314
 - advanced approaches 308–309
 - algorithm 313
 - all-or-nothing 306–307
 - and external calls 311
 - basis 306–307
 - complete 305
 - disposing of objects 310
- efficiency 307
 - parallel 308–309
 - practical issues 309–311
 - requirements 305–306
 - role 304
 - sound 305
 - timely 305
- garbage common block 48, 736
- Gates, William Henry 1136
- Gelernter, David 1033
- Gemstone 1055
- GENERAL* 201, 580–582, 583, 584, 976, 1044
- general relativity 194
- general-purpose facilities 180–181
- GENERAL_ACCOUNT* 606
- GENERAL_BOUNDARY_VALUE_PROBLEM* 766
- GENERAL_PHILOSOPHER* 1004, 1005
- GENERAL_WINDOW* 817, 818, 819
- generalist 933
- generalization 926, 928–930, 1196
- generated code efficiency 1144
- generating class, same as generator
- generation scavenging 308, 313, 316
- generator 219, 582, 1196
- generator* 582
- generic* (Ada) 1085, 1169
- generic derivation 96, 321, 322, 324, 325, 586–587, 1197
- constrained 586–587
- generic parameter, see actual and formal under parameter
- genericity 26, 84, 96–98, 317–330, 585–590, 617, 628–629, 877, 1084–1085, 1128, 1167–1188, 1197
- and efficiency 327–328
 - and inheritance 470–472, 585–590, 1167–1188
 - and once functions 652, 660
 - and repeated inheritance 561–562
 - as a solution to the covariance issue 628–629
 - combining with inheritance 470–472, 585–590, 1184–1187
 - constrained 27–28, 329, 330, 585–590, 617, 1170–1188, 1194
 - emulating 1176–1181
 - how to achieve in the presence of inheritance 1185–1187
 - non-object-oriented equivalents 587–588

- used recursively 590
 - cost 328
 - emulating with inheritance 1176–1183
 - for abstract data types 131–132, 318
 - in Ada 1084–1085
 - in C 1112
 - unconstrained 590, 1168–1188, 1202
 - as special case of constrained 590, 1187
 - emulating 1181–1183
 - how to achieve in the presence of inheritance 1184–1187
 - Geoffroy Saint-Hilaire, Étienne 865
 - Germany
 - university curriculum 941
 - Geschke, C.M. 64
 - Gessner, Conrad 864
 - get_and_remove* 987
 - get_integer* 423
 - getint* (C) 753
 - Ghezzi, Carlo 20, 1034
 - Gil, Yossi 863
 - Gindre, Cyrille 934
 - Giraudoux, Jean 671
 - GIRL* 623, 631, 634
 - GIRL_ROOM* 625
 - Girod, Xavier 823, 863
 - GKS graphical standard 1064, 1112
 - Global approach to the covariance issue 634, 639
 - global object, global value, global variable, see shared under object
 - global type analysis 633–636
 - go* 782, 790, 1070
 - Go To Statement Considered Harmful (Dijkstra) 664–665
 - go_before* 789
 - Goethe, Johan Wolfgang von 843
 - Goguen, Joseph A. 99, 160, 408
 - Goldberg, Adele 35, 119, 824, 937, 1075, 1126, 1138
 - Gore, Jacob 35, 497, 745, 948
 - Gosling, James 1139
 - goto* instruction (in traditional approaches) 277, 519, 664, 677–678, 835
 - Gouraud, Henri 72
 - graduate courses 941
 - Graham, Ian 271, 919
 - grammatical categories 742
 - grand mistake (in identifying classes) 726
 - graphical abstractions 1068–1071
 - graphical classes and operations 1071
 - graphical conventions 271, 464, 487, 537, 921–922, 1150
 - in the Business Object Notation 921–922, 1150
 - graphical form for requirements 914
 - graphical user interface, see GUI
 - graphics 33, 1063–1076
 - Great Encyclopedia 148
 - Gries, David xi, 407, 986
 - Grundstudium 941
 - GUE electronic mailing list 869
 - GUI (Graphical User Interface) 818, 1063–1076, 1150
 - interaction mechanisms 1071–1076
 - mathematical model 1076
 - needed tools 1064–1066
 - portability and platform adaptation 1066–1068
 - principal abstractions 1068–1071
 - GUIDE 1034
 - Guidéc, Frédéric 1034
 - Gulliver’s Travels 672
 - Guralnick, Robert 868
 - Gurvets, Joseph 674
 - Gutttag, John V. 160, 408
 - Gymnasium 941
- H**
- hacking, see organized hacking
 - Hadamard, Jacques 672, 725
 - Halbert, Daniel C. 746
 - handle 817–820, 855
 - handling
 - an event (e.g. in interactive graphics), see handling under event
 - an exception, see handling under exception
 - an object (in concurrent computation) 965–966
 - handshake 958
 - Hansen, Wilfred J. 12
 - Harbison, Samuel P. 1139
 - hardware parallelism 1007–1009
 - hardware support for object technology 309
 - hardware-software machine 11, 95, 352
 - has* 82, 86, 94, 883
 - HASHABLE* 590
 - hashing function 587
 - Hauptstudium 941
 - have versus be 812–814
 - Hawksworth, D.L. 868
 - header comment 178, 886–891
 - header file (C) 1108
 - heir 118, 462, 1197
 - Helena of Troy 269
 - Heliotis, James 948
 - HELP* 692
 - help facilities 15
 - Henderson-Sellers, Brian 34, 65, 408, 919, 922, 934, 948
 - Hennig, Willi 866, 868
 - Hercules 129
 - Hewitt, Carl 1033
 - Hewlett-Packard 1152
 - hidden* (Simula) 1115
 - hidden clause in contracts 994
 - hide* 1071
 - hiding, see information hiding
 - high school education 941
 - HIN syndrome 75
 - history* 704
 - history list 704–715
 - representation 710–711
 - history of a feature 1158
 - history of taxonomy 864–867
 - HISTORY_LIST* 734
 - HISTORY_LOG* 724
 - Hoare, C.A.R. xi, 19, 64, 72, 160, 335, 375, 395, 396, 407, 408, 449, 979, 1023, 1033, 1138
 - Hocking, Geoff 811
 - hold* (construct not retained) 978, 984
 - holder 999, 1000, 1009, 1027
 - hole 533, 1156, 1159
 - hole, see also programs with holes
 - Hollywood 953
 - honesty 573
 - Hopkins, Trevor 1138
 - Horning, Jim J. 160, 897
 - Horowitz, Ellis 99
 - HOUSE* 522
 - HOUSTON* 729
 - HP 9000 1152

HTML 108, 1159
 HTML (HyperText Markup Language) 391, 818
 Hugo, Victor 802
 Hullot, Jean-Marie 1139
 humility in software construction 673–674
 Huxley, Julian 866
 Hybrid 987, 1034
 hybrid language 294, 443–444, 616
 Hydra 129
HYDROPLANE 522

I

IBM 505, 1152
 IBM 704 12, 1102
 Ichbiah, Jean D. xi, 1079–1080, 1095, 1096
 identifier 457
 identifier feature, see identifier under feature
 identity, see under object
 IDL (Interface Definition Language) 955
 idle process 1123
 IEEE Computer Society 408
if 448
 IFL (Intermediate Functional Language) 401, 402
 Illustra 1055
 image processing 1051
 immediate
 feature, see immediate under feature
 size 771
 immigrant 1045, 1046
 impedance mismatch 931, 933, 1050, 1060, 1061
 imperative 145–146, 351–353
 name 727–728
 implementation 607–609, 713, 844–846, 931–933, 934, 936, 1162, 1198
 class 733–734
 helps analysis 713
 partial 503–504
 rehabilitated 608, 713, 931–933, 934
 role in education 936
 role in the software process 713
implementation 1179
 implementation inheritance 832–833, 844–846

 definition 832
 implementation invariant, see under class invariant
 implementation variation 84, 122–125
 implicitness 137, 149–150
implies 454, 456
 import 210
 see also the next entry and *deep_import*
 importing an object structure 976–977
 IMS 505
in out argument (in non-O-O approaches) 765, 1169
 in-line expansion, see inlining
in_schedule 910
include (C) 1108
 include file, should not be needed in object technology 1146
Incorrect_inspect_value 432
 incremental size 771
 incrementality
 of a garbage collector 307
 of the recompilation process 1145
 indentation 894–895
independent_store 1038, 1040, 1151
 index 1225
index 909
 index word 78
indexing 78, 177, 178
 indexing clause 78, 178–179, 890
 Indirect Invariant Effect 405–406
 induction versus deduction 859–860
 industrial training, see under teaching
 object technology
infix 189
 infix feature, see infix under feature
INFOMERCIAL 911
 information
 unstructured 1060
 information hiding 25, 51–53, 55, 65, 144–145, 191–194, 205–208, 605–609, 796, 804, 805, 1081–1087, 1197
 and abstract data types 144–145
 and inheritance 605–609
 and preconditions 357–359
 descendant hiding, see descendant hiding
 in Ada 1081–1087
 not a security measure 52
 Informix 1055
 Ingalls, Daniel H. H. 1126, 1138
 ingredients of computation 101–103

Ingres 820, 1152
inherit 462
 inheritance 26–30, 63, 119, 184, 459–874, 877, 1097, 1167–1188, 1197
 adaptation 735
 and analysis 907
 and assertions 569–580
 and class invariants 465, 570
 and concurrency 959–960, 1028–1029, 1121–1122
 and configuration management 66
 and creation 465–467, 479–480, 518
 and decentralization 498
 and efficiency 507–515
 and genericity 470–472, 585–590, 1167–1188
 and information hiding 605–609
 and selective exports 609
 anomaly 980, 981, 1028, 1034, 1035
 basic conventions and terminology 464–465
 clause 462
 combining with genericity 470–472, 585–590, 1184–1187
 constant, see constant under facility inheritance
 convenience 824
 dealing with multiple inheritance
 criteria 851–858
 efficiency consequences 482
 emulating with genericity 1175
 extension, see extension inheritance
 facility, see facility inheritance
 functional variation, see variation inheritance
 global inheritance structure 580–582
 graphical conventions 464, 487
 how not to use 809–812, 823–824
 how to design inheritance
 structures 858–861
 implementation, see implementation inheritance
 machine, see machine under facility inheritance
 mathematical model 828
 meaning and usage 494–500, 809–870
 summary 862
 methodology of applying 809–870
 summary 862
 model, see model inheritance
 module and type views 494–497, 1095–1096

- multiple 26–27, 519–568, 617, 1198
 - examples 519–534
 - what not to use as introductory example 520–521
 - one mechanism or more? 833–835, 1095–1096
 - parallel hierarchies 625
 - redundant 549
 - reification, see reification
 - inheritance
 - relation between classes, not objects 814
 - repeated 27, 543–563, 834, 1128
 - and efficiency 548
 - and genericity 561–562
 - unobtrusive 548–549
 - restriction, see restriction
 - inheritance
 - single 26, 851, 1201
 - software, see software inheritance
 - structure, see structure inheritance
 - subtype, see subtype inheritance
 - taxonomy of inheritance forms 822–835
 - deferred and effective classes 833
 - type variation, see variation
 - inheritance
 - uneffecting, see uneffecting
 - inheritance
 - versus client 812–817
 - view, see view inheritance
- Inheritance rule 822
- Inheritance Simplicity rule 823
- inherited feature, see inherited under feature
- inheriting general-purpose facilities 180–181
- initial* 689
- initialization 233, 236–237, 656–657
 - default values 233
 - overriding 236–237
 - of global constants and shared objects 656–657
- inlining 510, 1147
- Inmos 1033
- inner** (Simula) 959, 1121–1122
- input 457
 - checking 345–346, 422–423
 - fragile 422–423
- input* 686
- INRIA 1131
- insist* 1000, 1035
- inspect** 450
- Simula 1116
- inspector 746
- instance 166, 167, 168, 475, 1197
 - and mold 167–169
 - current 181–182, 453, 1194
 - direct 219, 475, 1195
 - variable (Smalltalk) 1197
- instruction 447–452
- int_minimum* (Ada) 1171
- int_swap* 1169
- INTBINTREE** 90, 91
- INTEGER** 171, 220, 319, 497, 522, 644, 745, 831, 1172, 1178
- INTEGER_COMPARABLE** 1178, 1185, 1186
- INTEGER_QUEUEABLE** 1183, 1184
- INTEGER_STACK** 318
- INTEGER_swap** 1168
- INTEGER_TABLE_HANDLING** 90
- INTEGRABLE_FUNCTION** 715, 718
- integration 927
- integrity 14, 15
- integrity constrained
 - see also class invariant
- integrity constraint 1047
- interaction mechanism, see under user interface
- interactive systems 675–718
- interactive version of a system 108–109
- intercontinental driver 544
- interface 51, 109–110
 - builder, see:Build;application builder
 - Explicit Interfaces 50, 65
 - Few Interfaces 47–48, 65
 - Small Interfaces 48–50, 65
- interface, see also under: class; user
- interfacing with external software, see external under call
- internal free list 302
- Internet 78, 925, 951, 953, 954, 955, 956, 969, 1015, 1032, 1136, 1147
- interpretation 618
- Intranet 78, 1147
- introductory courses, see under teaching object technology
- invariant** 364, 451
- Invariant rule 366
- invariant, see class invariant, loop invariant
- inverse* 799
- invert* 574, 579, 800
- IP address 1043
- is-a relation 497, 811, 812, 813, 816, 817, 824, 825, 844, 845
- Is-a rule of inheritance 811
- is-a-kind-of relation 497
- is_assertion_violation* 432
- is_concurrency_interrupt* 1000
- is_developer_exception* 435
- is_developer_exception_of_name* 435
- is_equal* 274, 275, 584
- is_signal* 432
- is_system_exception* 432
- ISE (Interactive Software Engineering Inc.) xi
- ISO certification 55
- italics 900
- Itasca 1055
- item* 755, 777, 791, 882, 1178, 1179, 1180
- iterative development 713
- iterator 529, 567, 848–850, 1197
- ITERATOR** 849

J

- Jackson, Michael 114, 120, 858
- Jacobson, Ivar 738, 740
- Jaeschke, Rex 1134
- Jakobson, Roman 867, 933
- Jalloul, Ghinwa 1033
- Japan
 - university curriculum 941
- Java 46, 56, 100, 208, 209, 211, 239, 392, 443, 548, 566, 595, 616, 670, 956, 957, 970, 1099, 1106, 1136–1137, 1139, 1147, 1153, 1159, 1161, 1162
 - bytecode 956, 1136, 1147
 - chip 1137
 - virtual machine 956, 1136, 1147
 - Workshop (Sun) 1153, 1159
- Jazayeri, Mehdi 20
- JEIDA (Japan Electronic Industry Development Association), <http://www.jeida.or.jp> 80
- Jenkins, Trevor 315
- Jézéquel, Jean-Marc 35, 948, 1034
- Johnson, Paul 316, 771, 772
- Johnston, J.B. 315

- join (in relational databases) 1048, 1051
- joining features 552–553
- Jones, Cliff B. xi, 160, 408
- Jones, John 316
- Jones, T. Capers 99
- Jonker, Dewi 1160
- Journal of Object-Oriented Programming 35
- JOVIAL (Jules's Own Version of the International Algorithmic Language) 1080
- Junia 1135
- Jussieu, Antoine-Laurent de 843, 865, 868
- JUSTIFIER** 850
- K**
- K&R 1107
- Kansas City 976
- Kay, Alan 1126
- Kempe, Magnus 1097
- Kemper, Alfons 1061
- Kernel library 522–523, 580–582, 592, 1150, 1165
- Kernighan, Brian 1107
- Kerstholt, John 948
- Khawam, Patrice 1160
- Khoshafian, Setrag 1061
- Kim, Won 1061
- Kindsoul, Jill 670
- kiwi 859
- Know the User (Hansen) 12
- knowledge, distribution of 63
- Knox, Sandra L. xiv
- Knuth, Donald Ervin xiv, 100, 409, 745, 807, 808, 1112, 1135
- Kraemer, Vincent 1160
- Krakowiak, Sacha 1034
- Krief, Philippe 745
- L**
- L'OBJET** 35
- Lace 198, 200, 393, 396
- Lahire, Philippe 1160
- laissez-faire policy for the society of modules 127
- Lalanne, Frédéric 1160
- Lalonde, Wilf R. 1138
- Lamarck, Jean-Baptiste 865
- lambda calculus 642
- Lampson, Butler W. 408
- language for object-oriented software construction 1143, 1144
- languages
 - non-object-oriented 1099–1112
 - object-oriented 1113–1140
 - usable versus useful 613–614
- Lano, Kevin 408
- Larch 160, 400, 408
- launch** 988
- launch_one** 988
- Laurin, Michel 863
- Law of Demeter 668, 671, 674
- law of inversion 684
- layout and presentation of software texts 891–901
 - fonts 900–901
- lazy evaluation 988
- lazy wait, see wait by necessity
- Le Vourch, Xavier 316, 1033, 1160
- Lea, Rodger 1034
- League for Programming Freedom 79, 80, 100
- Leeuw, Guus Jr. 1160
- legacy software 441–443
- Legacy++ 1144
- letter case 457, 881
- levels paradox 506
- Lex (Unix tool) 75
- Lex library 1149, 1151
- lexical conventions 457–458
- Ley, Michael 1061
- Lezard, Tony 316
- library 33, 42, 72, 197, 666, 747, 856, 941, 1065–1071, 1081, 1143, 1150–1152, 1165–1166
 - and analysis 907
 - consistency of design 69
 - dynamically linked, see DLL
 - evolution 33
 - for concurrency 1027
 - for GUI (Graphical User Interface) 1066–1071
 - indexing 34
 - role in education 941–946
 - role in methodology 666
 - see also Base libraries, Kernel library, Lex library, Math library, Net library, Parse library, Vision library, WEL library
- Lieberherr, Karl J. 668
- Lieberman, Henry 965
- Lientz, Bennet P. 17–18, 20, 125
- lifecycle 85, 103, 923–934
 - steps and tasks 926
- like** 601
- limited private** (Ada) 1085
- Linda 970, 1033
- LINE_DELETION** 699, 700, 727
- LINE_INSERTION** 709
- linear algebra 573
- linear equation solver 799
- LINEAR_ITERATOR** 849, 850, 851
- Linguistic Modular Units principle 53–54, 90, 1101
- LINKABLE** 298, 596, 597, 598, 600, 602, 604, 624, 776, 796, 1165
- linkable element 596–597
- LINKABLE1** 776
- linked** 123
- linked list 297–301, 596–597, 774–796
- LINKED_LIST** 297, 460, 466, 567, 596, 598, 600, 625, 727, 775, 795, 1165
- LINKED_LIST1** 777
- LINKED_QUEUE** 1188
- LINKED_STACK** 567
- LINKED_TABLE** 831
- LINKED_TREE** 542
- Linnaeus, Carolus, see Linné, Carl
- Linné, Carl 734, 843, 852, 864, 865, 867
- lint 635
- Linux 1152
- Lisa 1101
- Liskov, Barbara H. xi, 160, 408, 438, 806, 1079, 1097
- Lisp 65, 265, 269, 282, 315, 564, 988, 1126, 1129, 1130–1131
 - object-oriented extensions 564, 1130–1131, 1138
- list 1150
 - circular 567
 - see also linked list
 - with cursor 488, 752, 754, 755–759
- LIST** 472, 474, 489, 512, 526, 567, 602, 607, 848, 908, 1071, 1165
- live** 961, 962
- local** 213
- lock 1009–1012, 1036
 - minimizing 1056
 - see also locking under database

- LOCKABLE** 1009
LOCKER 1009
locking, see under database
LOCKING_PROCESS 1036
Löhr, Peter 948, 1034
LONDON 535, 538
London Ambulance Service 292–293, 302, 315–316
London, Ralph 1079
Long, Darrell D.E. 125
longjmp 414
Loomis, Mary E. S. 1061
loop 451
 how to get right 380–388
 proving 383–385
 syntax 386–388
loop 451
loop invariant 380–388, 413, 1197
loop variant 380–388, 1197
Loop_invariant 432
Loop_variant 432
Loops 1131, 1139
LOUISIANA_INSTITUTIONS 834
lowest common denominator approach
 for graphical libraries 1067
LPG 100, 330, 1167
Lukács, Georg 831
- M**
- M 330
machine 751–752
 abstract 792
machine inheritance, see under facility
 inheritance
Macintosh 696, 1101
macro 440
macro 440
Maddison, David 863
Madsen, Ole-Lehrmann 1139
Mail-A-Pet 474, 478
main program
 no such notion in object-oriented
 development 197–198
maintenance 17–19, 43, 68
 and reusability 68
Make 802
Make file, should not be needed in
 object technology 1146
make_cartesian 239, 762
make_philosophers 1006
make_polar 239
Mallet, Olivier 1160
Management Information Systems 6
manager 57, 59
MANAGER 852
Mandrioli, Dino 20, 1034
Manfredi, Raphaël 316, 1160
manifest constant, see under constant
map
 as model for graphical abstractions
 1068–1070
Marcus, Robert 1138
mark-and-sweep 313
marriage of convenience 530–532,
 844, 850
Martin, James 271, 918
Martin-Odell method 918
Math library 442–443, 1149, 1152,
 1160
Matisse 1055, 1056–1057, 1061
MATRICES (Ada) 1172, 1173
MATRIX 171, 522, 573, 1179, 1185,
 1186, 1188
Matsuoka, Satoshi 956, 1034
Maughan, Glenn 1160
max 756
maxdiag 593
Maximum_interval 910
Mayr, Ernst 866, 868
McCall, James 19, 20
McGregor, John D. 863
McIlroy, M. Douglas 67, 99
McKim, James C. xi, 215, 277, 408,
 642, 807, 948
McMenamin, Stephen M. 120
meaning 432
MEL library 442, 818, 1067, 1149,
 1150
Mellor, Steve 918, 922
Melt 1154
melting 1145
Melting Ice Technology 618, 1145–
 1146
melting software elements 1145
member_of (in the Business Object
 Notation) 920
MEMORY 309, 314, 998
memory management 30, 279–316
 and ease of development 295–296
 and reliability 294–295
 automatic 301–314
 rationale 301
 casual approach 290, 291–293
 compaction 313
 component-level approach 297–
 301
 detection 293
 free mode 280–287
 manual reclamation 290, 294–296
 reclaiming memory 293–294
 see also garbage collection,
 reference counting
 space reclamation 282–283
 precise nature of task 302
 stack-based mode 280–287
 static mode 280–287
 the problem in object-oriented
 development 290
 three modes of object creation
 280–291
Mencken, H.L. 608
Menger, Mario 1160
Mephistopheles 604
Mesa 99, 1081
message 592, 1127–1128, 1197
 binary 1127
 keyword 1127
 unary 1127
message 681, 682, 687
Message not understood (Smalltalk)
 320
meta-object protocol 1130
metaclass 168–169, 1198
metaphor 671–673, 674, 751
 on the Net 674
method
 of object-oriented software
 construction 1143, 1162
 see also methods under analysis
method (Smalltalk term) 1126, 1198
 see feature, routine
methodologies, see methods under
 analysis
methodology 663–948
 abstraction and precision in rules
 669
 exception to rules 668–669
 limitations 670–671
 of using inheritance 809–870
 summary 862
 role and principles 663–674
 role of practice 665–666
 role of reusability 666
 role of theory 665
 typology of rules 666
 use of this word in the singular 664
 useful O-O techniques summary
 871–874

- methods, see under analysis
- metrics for object technology 65–66
- Microsoft 8, 67, 955, 1043, 1130, 1134
 - Foundation Classes 1134
 - Word 108, 696
- middle initial 125
- middleman in contracts 575
- middleware 8
- military-aerospace community 1080
- millennium problem 18
- Mills, Harlan D. xi, 334, 341
- Milner, Robin 641, 956, 957, 1033
- Mingins, Christine 65, 948, 1160
- minimum* 1177
- minimum* (Ada) 1170, 1171
- Minimum_duration* 910
- Minsky, Naftaly 948
- mirroring 1056
- MIS (Management Information Systems) 230
- mismatch, see: impedance mismatch; mismatch under object
- mismatch_information* 1044
- missionary 933
- Mitchell, John G. 1097
- Mitchell, Richard 948
- mixed abstractions 730
- ML 641
- MML (FrameMaker) 391, 1159
- moa 859
- MOBILE_HOME* 522
- mode in interactive systems 1075, 1153–1159
- model inheritance 825
- modeling 228–231
 - role of expanded types 256–258
 - see also simulation
- modesty 441
- modifier function 135
- Modula-2 53, 84, 90, 99, 211, 265, 447, 1079, 1081, 1097, 1106, 1137
- Modula-3 1137, 1139
- modular
 - composability, continuity, decomposability, protection, understandability
 - see under each of these terms
- Modular Protection principle 345
- modular units, see Linguistic Modular Units principle
- modularity 16, 39–66, 83–98, 146, 643
 - criteria 40–46
 - principles 53–63
 - rules 46–53
- module 24, 39–66, 83–98, 209–210, 494–497, 643, 923, 1198
 - abstracted 73, 90
 - and inheritance 495–497
 - and type 170, 185, 1095–1096
 - as a syntactic concept 170
 - family 84, 99
 - generic 96
 - interface, see interface under: class; package
 - Mesa term 1081
 - Modula term 90, 1081
 - see also cluster
 - specification, see interface under: class; package
 - super-module 209–210, 923
 - traditional module structures 89–93
 - why closed 57
 - why open 57
- module-type identification 185
- Moffat, David V. 660
- mold and instance 167–169
- Monash University 1160
- monitoring assertions at run time 392–399
- Moon, David A. 1139
- MooZ 408
- Morgan, David 1160
- Morrison, R. 415
- MORTGAGE* 839
- MOSES method 919
- Motif 442, 818, 1064, 1065, 1067, 1150
- MOTIF* 818, 819
- MOTOR* 1015
- mountain path 905
- mouse 1071
- move* 880
- move_pixel* 1070
- move_proportional* 1070
- MS_WINDOWS* 818
- multi-branch instruction 449–451
- multi-launcher 988, 1006
- multi-panel systems 675–694
- multimedia 1051
- multiple criteria for inheritance 851–858
- multiple inheritance, see multiple under inheritance
- multiple results for a function 446, 758
- multiple views 55, 836, 851–852, 914–917
 - in analysis 914–917, 920
 - maintaining consistency 915
 - of software 55
- multiple-entry subroutine (Fortran) 1104–1106
- multiprocessing 953
- multiprogramming 954
- multithreading, see thread
- Munch, Edvard 81
- MUSE 1034
- MVC model 734, 745
- N**
- N-version programming 426–427
- NAG library 443, 1151
- name
 - adapting to the local context 538–539
 - see also name clash, naming conventions, style
- name clash 535–540
 - and naming conventions 539
 - precise definition and rule 562
- naming conventions 127–128, 539, 879–884
 - and name clashes 539
 - benefits 883–884
 - features 879–884
 - for classes 879
 - general rules 879–880
 - grammatical categories 881
 - letter case 881
 - local entities and routine arguments 880
 - standard feature names 882–883
- narrowing 595
- NASA 1080
- native 933
- NATO Science Affairs Committee 99
- natural-language requirements 914, 916–917
 - deduced from formal text 916–917
- necessity, see wait by necessity
- needs 196, 198
- needs directly 196

- negative rule, see absolute negative rule, advisory rule
- negotiation in analysis 906
- Nero 864, 1135
- Nerson, Jean-Marc 271, 277, 517, 715, 772, 919, 922, 1034, 1160
- nesting 49–50, 209–210, 524–525, 923, 1070
 of windows 524–525, 1070
- Net library 1039, 1149, 1151, 1152
- Neumann, Peter G. 315
 see also Risks forum
- new** (Ada) 1085
- new** (Simula) 1115
- NEW_MORTGAGE** 840, 843
- NEW_YORK** 535, 536
- Newton, Sir Isaac 864
- next** 909, 910
- next_in_list** 910
- NEXTSTEP 1131, 1132
- Nierstrasz, Oscar 1034
- NIH syndrome 75–76
- no hidden clause principle 994
- No-Instantiation rule 487
- No_more_memory** 432
- nominal detection policy (persistence) 1042
- non-deterministic wait 979
- non-object-oriented languages and environments 1099–1112
- Non-Redundancy principle 343, 355
- non-separate 1198
- non-strict boolean operator, see boolean under operator
- NON_UNDOABLE** 717
- NONE** 582
- nonlinear_ode** 765
- NORMAL** 1174
- normalized relational database 1048
- Norwegian Computing Center 1114
- notation design 278
- notification, see under persistence
- noun 720
- novariance 628, 1198
- null record** (Ada 95) 1092
- NUMERIC** 171, 522, 523, 589, 610, 831
- numerical phenetics, see phenetics
- Nygaard, Kristen 35, 1114, 1137, 1138
- O**
- O'Brien, Patrick D. 746
- Oberon 1137, 1139
- Obfuscated
 C 876
 C++ 876
- OBJ-2 160, 330, 400, 408
- object 165, 217–278, 1198
 abstract 1103, 1193
 active, see active object
 adopting 311
 as a modeling tool 228–231
 as machine 751–752
 attaching to a reference, see attachment
 automatic duplication 1056
 basic form 219–220
 blocking, see blocking object
 cache 1056
 cloning 245–246, 247–249, 274–275, 583–584
 deep 247–249
 shallow 247–249
 composite 254–261
 concurrent access 982–983, 1031
 conversion 1042
 copying 247, 274–275, 583–584
 created, external 968–969
 creation and destruction 279–316
 see also memory management, garbage collection
 creation, see creation
 current, see current under instance
 deallocation 294–296
 definition 218–219
 describing 118
 development, see development
 object
 do not confuse with class 165–169, 216
 emulation, see emulation
 equality 245–246, 264–265
 and attachment 264–265
 external 218, 219, 732–733
 finding the objects, see finding under class
 global, see shared
 here for the picking 117, 720, 733
 identity 225, 1052–1053, 1061, 1197, 1198
 importing an object structure 976–977
 integrity 513, 982–985, 999, 1056
 lifecycle 365
 live 285
 manipulating 231–236
 mirroring 1056
 mismatch 1041, 1042, 1043, 1044, 1045, 1046, 1060
 motto 116
 moving 312–313
 optimized placement on disks 1056
 persistent 32, 225, 1037–1062, 1199
 see also persistence
 precomputing 708–709
 reachable
 definition 290
 see also unreachable
 reclamation, see space reclamation under memory management
 request broker, see object request broker
 reserving 983–985, 1027
 root, see root object
 run-time object structure 227–228
 separate 967
 shared 643–660
 how to obtain 648–649
 initialization 656–657
 software 219
 storing and retrieving object structures 250–253, 1037–1062
 formats 1038–1039
 technology, see object technology
 transient 1202
 unreachable 284–290
 definition 290
 in classical approaches 285–287
 in object-oriented development 288–290
 user interface 1072
 versioning 1054, 1056, 1057, 1059, 1061
 in Matisse 1056
 in Versant 1057
 versus function 146
 weaning 311
 wrapper
 see object under wrapper
- Object Currents 35
- Object Database Management Group, see ODMG standard
- Object Magazine 35
- Object Management Group 955
- Object Pascal 443, 616, 1101, 1161
- Object Pursuit 720
- object request broker 955
- object technology

- and Ada 95 1094–1095
- applied to graphical developments 1066
- contribution to analysis 907
- education, teaching, training, see teaching object technology
- emulating in non-object-oriented environments 1099–1112
- for building search tools 1060
- hardware support 309
- levels of language support 1099–1100
- list of criteria 21–36
- rationale 101–120
- Object Tool 1153, 1155, 1156
- object-based decomposition 114–115
- object-oriented 1198
 - analysis, see analysis
 - computation, basic mechanism 611–612
 - contrasted with top-down architecture 684–693
 - database, see database
 - design, see design
 - education, teaching, training, see teaching object technology
 - environment 1143–1160
 - languages 1053, 1100, 1113–1140
 - methodology 663–948
 - methods, see methods under analysis
 - rearchitecturing 441–443
 - rearchitecturing, see rearchitecturing
 - style of computation 181–191
 - teaching plan, see under teaching object technology
- object-oriented computation
 - basic mechanism 183
- object-oriented software construction
 - definition 116, 147
 - issues 117–118
- Object-Z 160, 400, 408
- Objective-C 294, 443, 1099, 1106, 1107, 1131–1132
- Objectivity 1055
- ObjectStore 1055
- ObjEdit library 1149, 1152
- Objekt Spektrum 35
- obsolescence, obsolete
 - see obsolete under class and feature
- obsolete** 802
- Occam 980, 1033
- Occam2 1033
- occurrence* 777
- OCX 67
- Odell, Jim 271, 918, 919
- ODMG standard (Object Database Management Group) 1055, 1057
- Ogor, Robert 948
- OK button 1074
- old** 340
- oldest* 962
- OLE-COM 8, 67, 955, 1043
- OMG, see Object Management Group
- OMT 917, 1162
- on-the-fly
 - garbage collection, see parallel under garbage collection
 - object conversion 1041, 1042
- on_item* 784
- once** 648
- once attribute? 660
- once function 647–650
 - and anchored types 652–653
 - and genericity 652, 660
 - emulating unique values 660
 - returning result of basic type 650
- Once Function rule 653
- once procedure 651
- once routine 647–660, 1036
 - and concurrency 1036
 - applications 648–653
 - see also once function, once procedure
- one* 522
- one_command* 425
- Ong, C.L. 745
- Ontos 1055
- OOIE (Object-Oriented Information Engineering) 918
- OOPSLA concurrency workshops 1034
- OOSE method 918
- OOZE 408
- open** 1174, 1175
- OPEN unified method 919
- Open-Closed principle 57–61, 63, 65, 83, 465, 495, 496, 511, 514, 517, 536, 577, 583, 592, 607, 633, 735, 768, 803, 830, 834, 837, 839, 861, 869, 959, 1092, 1116, 1174
 - and Ada 95 1092
- OPEN_FIGURE** 483, 527
- openness for modules, why needed 57
- openness of an object-oriented language 439–444, 1144
- operand 766–770
 - definition 766
 - distinguishing from option 767
- Operand principle 767
 - benefiting 769
 - checklist 770
 - possible exceptions 769
- operating system 107–108, 197, 413
 - booting procedure 197
 - signal 417
- operator
 - binary 453
 - boolean
 - non-strict 454–456, 458
 - expression 453–454
 - precedence 896
 - unary 453
- operator feature, see operator under feature
- operator overloading, see overloading (see also operator under feature)
- optimistic locking 1055, 1057
 - in Matisse 1056
 - in Versant 1057
- optimization 208–209, 509–511, 1147–1148
- option 766–770
 - definition 766
 - distinguishing from operand 767
- or** 454
- or else** 454, 578
- Oracle 820, 1152, 1055
 - Oracle 8 1055
- orange 147
- order relation 523
- ordering constraints 110–112, 202, 738–740
- organized hacking 60–61, 830, 869
- organized panic 417, 1090, 1198
- origin 285
 - definition 290
 - reference 286
- original_class_name* 433
- original_recipient_name* 433
- original_tag_name* 432
- Orr, Ken T. 120
- OS, abbreviation for operating system
- OS/2 201, 442, 818, 1064, 1150, 1152
- OSA method 918
- Osmond curves 13, 20
- Osmond, Roger F. 13, 20
- ostrich 859
- OSTRICH** 627, 841, 843

- others** (Ada) 415
- output 457
- over** 961
- overlay 282
- overloading 93–98, 239, 564–566, 1134, 1199
- and creation 239
 - semantic 95–96
 - syntactic 93–95
- overspecification 125, 573
- O2 1055
- P**
- package 90–98, 209–210, 1081, 1169–1188, 1199
- as abstracted module 90
 - assessment of contribution to reusability 92–93
 - generic (in Ada) 1084–1085
 - implementation (in Ada) 1084
 - in Ada 392, 1081–1096, 1169–1188
 - in Java 392
 - interface (in Ada) 1082–1083
 - not needed thanks to selective exports 209–210
 - package-class 1098
 - pattern 1170
 - see also cluster
 - specification, see interface under package
 - use in a client (in Ada) 1083–1084
- Paepcke, Andreas 1139
- Page-Jones, Meilir 34, 119, 120, 730, 745, 863
- painting** 198
- pan** 1070
- panel-driven systems 675–694, 709
- panic, see organized panic
- Papathomas, Michael 1033, 1034
- paradox of levels 506
- PARAGRAPH** 850
- parallel garbage collection, see under garbage collection
- parallel inheritance hierarchies 625
- parallelism, see concurrency, hardware parallelism
- parameter
- actual 96, 321
 - formal 96, 318
 - constrained 588–590
 - term used only for genericity (see also argument, system parameter) 322
- parent 462, 500, 1199
- Parents' Invariant rule 570
- Parnas, David Lorge xi, 64, 160, 806, 1113
- Parse library 1149, 1151
- parse tree 115
- parsing, object-oriented approach 1151
- part of relation 907
- partial correctness 337
- partial function 138–139, 377, 580
- alternatives 151–152
- partial implementation 503–504
- Pascal 45, 49, 56, 61, 64, 165, 176, 211, 225, 230, 265, 269, 270, 282, 285, 286, 315, 327, 346, 386, 439, 443, 447, 449, 507, 616, 716, 737, 876, 940, 947, 1100–1101, 1106, 1108, 1130, 1161, 1168
- emulating object technology 1100–1101
 - modular extensions 1101
 - object-oriented extensions 1101, 1136, 1137
 - UCSD 947
- path expression 979, 980, 1033
- Pavarotti, Luciano 674
- payroll system 105–106
- Peano, Giuseppe 171
- pebble 1156
- pedagogy
- see under teaching object technology
- PEL library 442, 1067, 1150
- PENTAGON** 467
- Pérec, Georges 672
- perfect foresight 629
- performance, synonym for efficiency (see this term) 9
- perimeter** 461, 834
- Perl 754, 1152
- Perlis, Alan J. 99
- PERMANENT** 853
- persistence 32, 250–253, 1037–1062, 1149, 1199
- closure 32, 252, 1037–1038, 1039–1040, 1152
 - when not applicable 1039–1040
 - completeness, see closure under persistence
 - correction 1042, 1045–1046
 - detection 1042, 1042–1043, 1044, 1045, 1046, 1060
 - from the language 1037–1039
 - notification 1042, 1044, 1060
 - schema evolution 1041–1046, 1060, 1201
 - naïve approaches 1041–1042
- Persistence Closure principle 252, 1037, 1038, 1039–1040, 1152
- persistent object, see under object
- person 277
- PERSON** 810
- pessimism, pessimistic, see pessimism under typing
- Petri net 979, 981
- phenetics 866–868
- PHIGS graphical standard 1064
- PHILOSOPHER** 1004
- pick-and-throw 1156–1159
- picking (objects here for the –) 117, 720, 733
- Pinson, Lewis J. 1138
- pixel 1070
- PL/I 46, 269, 270, 442, 898, 1080, 1107
- PL/360 1111
- place_pixel** 1070
- place_proportional** 1070
- plants, distinguishing from animals 841
- PLATFORM_WINDOW** 817
- platitude versus principle 667
- Pliny the Elder 864
- plug-in 955, 956, 1147
- Poet 1055
- POINT** 166, 172, 173, 175, 176, 180, 216, 218, 859
- POINT_2D** 858
- pointers 315
- polar_ready** 760
- police chief 336
- polygon 460–461
- POLYGON** 460, 465, 466, 474, 475, 483, 497, 527, 626, 627, 633, 834, 1122
- Simula 1116
- POLYLINE** 483
- polymorphic argument 637
- polymorphic assignment 469
- polymorphic attachment 467–470
- polymorphic call 638

- polymorphic catcall, see catcall
- polymorphic data structure 329, 470–472, 512, 585, 593, 692, 1199
- type-specific access 593
- polymorphic entity 469, 488, 637
- polymorphic perversity 625–626
- polymorphism 28, 63, 467–472, 570–580, 816–817, 1097, 1174, 1175, 1199
- and assertions 570–580
 - limits 474
- Polymorphism rule 817
- polyonymy 269
- POOL 1034
- Pooley, Robert J. 1138
- Popper, Karl R. 812, 867, 868
- portability 11, 15, 16, 19, 1066–1068, 1152
- of GUI (Graphical User Interface) tools 1066–1068
- positive rule, see absolute positive rule, advisory rule
- post_action* 1040
- post_retrieve* 1040
- post_store* 1040
- postal code 18
- postcondition 338–410, 983, 993, 994, 1022–1024, 1177, 1199
- and attributes 579
 - and generalization 930
 - and inheritance 570–580
 - under concurrency 995, 997
- Postcondition* 432
- postcondition paradox 995
- postgraduate 941
- POSTGRES 1055
- Postscript 108, 818
- Potter, John 567, 948, 1033
- practice, role in software methodology 665–666
- precomputing a polymorphic instance set 708–709
- pre_store* 1040
- precedence of operators 896
- precepts (full list) 1189
- precipice and clouds 905–906
- precision in methodology rules 669
- precision versus abstraction 905–906
- precompilation 1146–1147
- precondition 146, 338–410, 838, 1022–1024, 1199
- abstract 576–577
 - and export status 357–359
 - and inheritance 570–580
 - in abstract data types 138–139
 - tolerant or demanding style 354–357, 359–363
 - under concurrent execution 993–997
- Precondition* 432
- Precondition Availability rule 358
- precondition checking paradox 397
- Precondition Paradox 995, 1036
- precursor 507
- Precursor* 493, 494, 507, 517, 555–560, 1128, 1140
- precursor of a feature 493
- predicate, see assertion
- Pree, Wolfgang 71, 100
- prefix** 189
- prefix feature, see prefix under feature
- premature ordering, see ordering constraints
- preorder 523
- prepare_cartesian* 761
- preprocessor 42
- Presentation Manager 442, 818, 1064, 1065, 1067, 1150
- preserve* 1040
- Principle of Least Surprise 454, 896
- Principle of Modesty 441
- principle of selfishness 722
- principle of selfishness 147–148
- principle versus platitude 667
- principles (full list) 1189
- Principles of Truth 663
- print* 582, 963
- PRINT_CONTROLLER** 968
- print_line* 582
- PRINTER** 596, 602, 967, 1035
- Simula 1120
- PRINTER_1** 960
- private** (Ada) 1086, 1087, 1098
- private, see secret
- procedural
- means “imperative” (not the antonym of “object-oriented”); see functional 1100
- procedure 89, 174, 203, 447–448, 1199
- call 447–448
 - creation, see creation procedure
 - once, see once procedure
- process 956–963, 1123–1126, 1139, 1140
- active 1123
 - idle 1123
 - in discrete-event simulation 1123–1126, 1139, 1140
 - programmed as a class 960–963
 - software process, see lifecycle
 - suspended 1123
 - terminated 1123
- process* 681, 682, 687
- Process 970
- PROCESS** 961, 962, 963, 1004, 1030, 1036, 1140
- PROCESS** (Simula) 1123–1126, 1139, 1140
- processor 964–1032, 1199
- definition 964
- procurement of software 76
- PRODUCER** (Simula) 1120
- producer, see under reuse; see also supplier
- PROGRAM** 912
- program, see system
- PROGRAM_FRAGMENT** 910, 911
- programming a TV station 911–912
- programs with holes 72, 505–506
- project lifecycle, see lifecycle
- Project Tool 1153, 1154, 1155, 1156, 1159
- projecting software texts on a screen 901
- projection (in relational databases) 1048
- proof by analogy 672
- proof rule 1022–1024
- proper ancestor 464, 500, 1200
- proper descendant 464, 1200
- protected** (Simula) 1115
- protection 45–46, 47, 48, 54
- prototyping 518
- Proust, Marcel 887
- proxy 968, 969, 989
- prunable* 883
- prune* 779, 882
- pSather 1034, 1137
- pseudo-random number generation 754–755
- public 51
- Pugh, John R. 1138
- put* 778, 884, 1178, 1179
- put_left* 783

put_right 298, 607, 783
put_state 691
put_transition 692
 PVM (Parallel Virtual Machine) 970

Q

qua (Simula) 1116
QUADRANGLE 467, 483, 858
 Qualified Call rule 447, 453
 qualified call, see qualified under call
 quality 3–20, 294–296
 external factors 4–16
 factors 3–16
 internal factors 3
 tradeoffs 15
quasi_inverse 424
 query 135, 748, 987
 button 751
 expression 154
 in databases, see query under database
 property-based 1047
QUEUABLE 1181, 1183, 1184
 queue 127, 162, 410, 710–711, 734,
 990–992, 1150, 1169, 1183
 bounded 710–711, 992, 1183
QUEUE 734, 882, 1181, 1184,
 1187, 1188
QUEUES (Ada) 1170
 Quicksort 72, 877
QUOTATION 232

R

Racine, Jean 1135
raise 1072
raise (Ada) 415, 436, 1090
raise_mismatch_exception 1044
 Ramaekers, Jean 315
 Randell, Brian 438
 random numbers, see pseudo-random
 number generation
RANKED_GIRL 641
 Rannou, Robert 948
 rapid prototyping 518
rating 910
 Ray, John 864, 865
 reachable object, see unreachable
 under object
reactivate (Simula) 1124
read 681, 682, 687

readability 615, 644
 and static typing 615
 and symbolic constants 644
readable 883
REAL 171, 174, 188, 189, 220, 522,
 644
REAL_STACK 318
REAL_STACKS (Ada) 1082, 1084
 realism, realistic, see realism under
 typing
 reality 230–231
 virtual 231
 rearchitecting 441–443, 1151
 Reasonable Precondition principle
 356, 357
 reattachment 231–232
 Réaumur, René Antoine Ferchault de
 673
recipient_name 433
reclaim 297
 reclamation, see space reclamation
 under memory management
 recompilation
 see compilation
 time 1144
 reconciliation 1150
 record
 in relational databases 1048
 record type (Pascal) 737
 with variants 61–63
 recovering from operating system
 signals 423–424
 recovery block 438
 rectangle 462–463, 524–525
RECTANGLE 463, 465, 466, 483,
 511, 513, 525, 527, 591, 626, 627,
 633, 634, 826, 858, 1122
 recursion 170, 212, 235, 247, 248,
 252, 253, 259, 277, 280, 295
 recursive dispose 295
recycle 299
 recycling objects 299–300
 redeclaration 491–494, 1200
 and assertions 570–580
 and renaming 538
 and typing 595–598
 conflicting under repeated
 inheritance 551–561
 definition 485
redeem 839, 843
redeemable 839
redefine 462, 507, 538
 redefinition 28, 204, 834, 1200

and once routines 648
 how to prohibit 583
 of an argument type 621–641
 using the original version 493–494
redo 706
 redoing, see undoing
 reengineering, see rearchitecting
ref (Simula) 1115
 reference 222–225, 226–228, 240–
 253, 265–270, 272–274, 445, 815,
 1053, 1200
 and class invariants 403–406
 attaching to an object, see
 attachment
 attachment, see under attachment
 comparison 244
 counting, see reference counting
 declaring 226
 definition 224
 disciplined approach 277
 encapsulating reference
 manipulations 269–270
 in databases 1053
 no references to subobjects 260–
 261
 not sufficient 254
 operations 242–253
 origin 286
 self-reference 226–227
 semantics 261–265
 separate 967
 states 240
 type, see under type
 versus simple values 272–274
 void 240–241, 244–245
reference (notation not retained)
 272, 286
 reference counting 302–304
 referential transparency 749–751,
 752, 753, 754, 756
 definition 750
 reflection 1130
 region, see critical region
 registration mechanism for detection
 of object mismatches in persistence
 1042
 reification inheritance 831
 definition 831
 relation
 in relational databases 1048
 relational algebra 1048
 relational database, see relational
 under database
 relativity 194

- reliability 16, 68, 294–295, 331, 332, 441, 615, 1043
 - and memory management 294–295
 - and static typing 615
- remembered* 710
- remote execution 955–956, 1147
 - across the Internet 1147
- remove* 298, 783, 880, 882
- remove_all_right* 710
- remove_left* 298
- remove_oldest* 963
- remove_right* 298
- remove_two* 987
- rename* 536, 538
- renaming rule 549–550, 562–563
- renaming, see under feature
- rendez-vous 958
- repairability 14
- Repeated Inheritance rule 546
- repeated inheritance, see repeated under inheritance
- repetition in software development 74
- replace* 783, 882
- replication under repeated inheritance 544–548
- representation independence 84–85, 98, 499
- representation invariant, see “implementation invariant” under class invariant
- require* 112, 338
- require else* 578
- requirements document 116, 720–725, 914–917
 - various forms (natural language, graphical, formal) 914–917
- rescue* 419, 452
- rescue clause 419–422, 452
 - correctness 427–430
 - role 429–430
 - tasks 427–430
 - when absent 430
- RESERVATION* 692
- reserve* 978
- reserving an object 983–985, 1027
- restaurant 522
- restore* 1040
- restriction inheritance 826
 - definition 826
- Result* 179, 210–213, 452
- result of a function, see under function
- resume* 1013, 1014
- resume* (Simula) 1119
- resumption, see retrying
- retain* 1000
- retargeting, see targeting a tool
- retrieved* 252, 1039, 1045, 1047
- retry* 419, 420, 452
- retrying 419–422, 424–425, 452, 1090, 1200
- RETURN* (Fortran) 1104
- return* instruction (not retained) 211
- reusability 7, 15, 16, 67–100, 112–113, 115, 441, 607–609, 666, 735, 740–741, 773, 830, 856, 905, 908, 928–930, 1066, 1146–1147, 1200
 - and methodology 666
 - and the software development process 928–930
 - benefits 68–69
 - goals 68–70
 - in analysis 905, 908
 - in graphical applications 1066
 - nature of reusable components 70–73
 - obstacle
 - non-technical 74–81
 - obstacles
 - technical 81–83
 - of abstracted modules 73
 - of designs and specifications 70–71, 89
 - of implementation 607–609
 - of interface 607–609
 - of non-concurrent software 1031
 - of personnel 70
 - of source code 72–73
 - styles 608–609
- reusability culture 929
- reusable component
 - what form? 70–98
- reusable software component 67–100, 1200
 - distribution formats 79–80
 - how to find and access 77–79
 - indexing 78–79
- reuse
 - consumers 69–70
 - producers 69–70
 - see reusability
- reuse-redo dilemma 82–83, 735
- reuser of the month 929
- reverse assignment, reverse assignment attempt, see assignment attempt
- reverse engineering 1150
- reversibility 919, 930, 931–933, 1150, 1200
 - in BON 919
- reversion 860
- rewind* 1173, 1174
- Rich, Charles 99
- Riehle, Richard 1097
- right* 607
- Riley, David 948
- Rine, David 948
- ring 827, 1173, 1180, 1185
- RING_ELEMENT* 1176, 1177, 1178, 1179, 1180, 1184, 1185, 1188
- Risks forum 8, 125, 315–316
 - Web address 316
- Rist, Robert 35, 948
- Ritchie, Dennis 1107
- robustness 5–6, 16, 331, 332, 389
- roman 900
- ROOM* 629
- roommate* 622
- root
 - as synonym for “origin” (see that word) in memory management 285
 - class 196, 1200
 - creation procedure 196
 - directory 199
 - object 195, 285, 288, 289, 290, 1200
- root* (Lace) 199
- Roscoe, A. William 1033
- Rosenbaum, Sarah xiv
- Ross, Herbert H. 868
- rotate* 1071
- Rousseau, Roger 948
- routine 89–90, 173–176, 1200
 - body 178
 - cannot modify its arguments 446
 - inlining, see inlining
 - once, see once routine
 - overloading, see overloading
- routine grouping 84, 90
- Routine_failure* 432
- RS/6000 1152
- RTF (Microsoft’s Rich Text Format) 391, 1159
- rule

- see also absolute positive rule, absolute negative rule, advisory rule, methodology, principle
 - self practice 878
 - rule of change 814–816
 - rules
 - full list of methodological rules 1189
 - on library design 674
 - rules on rules 664–671
 - Rumbaugh, James 934
 - run time 1201
 - run-time
 - object structure, see under object
 - see also dynamic, run time, runtime
 - type interrogation, see assignment attempt
 - runtime 30, 291, 293, 294, 297, 304, 1148, 1200
 - definition 304, 1200
- S**
- Sacks, Oliver 167
 - safe_share* 640
 - Sagan, Carl 672
 - Saint-Hilaire, see Geoffroy Saint-Hilaire
 - same_type* 582, 640
 - SAN_FRANCISCO* 497, 729
 - SANTA_BARBARA* 535, 538
 - Santiago de Compostela 152
 - Sarkis, Jean-Pierre 1160
 - Sather 642, 1137, 1139
 - satisfiable call 1026
 - SAVE* (Fortran) 1104
 - Scaife, Ross 277
 - Schaffert, Craig 330, 1188
 - schedule* 909, 910
 - SCHEDULE* 908, 909, 913
 - schedule for a TV station 908–909
 - schema, schema evolution, see schema evolution under persistence
 - Schiller, Friedrich 1141
 - Schmidt, Heinz 1160
 - Schwartz, Laurent 672
 - Schweitzer, Michael 301, 567
 - SCOOP (Simple Concurrent Object-Oriented Programming), see concurrency
 - Scott-Ram, N.R. 868
 - scripting languages 754
 - seamless development, see seamlessness
 - seamlessness 22–23, 506, 919, 930–933, 941, 1150, 1162, 1201
 - in BON 919
 - role in education 941
 - search* 791
 - secondary education 941
 - secret 51
 - secretary-receptionist algorithm 1000–1002
 - security issues 956
 - SEGMENT* 483, 829, 858, 909, 910, 911, 913
 - segment for a TV station 909–911
 - select* 554
 - select* (SQL) 1049
 - selecting a feature under repeated inheritance 553–555
 - selection (in relational databases) 1048
 - selective export 191–194, 209–210, 609, 796–797, 1201
 - and inheritance 609
 - architectural role 209–210
 - Selective Export Inheritance rule 609
 - Self 215, 641, 1137, 1139
 - self* (Smalltalk) 453
 - self practice 878
 - Self-Documentation principle 54–55, 78, 179, 804, 890
 - self-reference, see under reference
 - semantics, see under reference, value
 - semaphore 978, 1009–1012, 1036
 - semicolon
 - use as separator, terminator, or optional element 897–899, 1088
 - Semicolon Style principle 899
 - Semicolon Syntax rule 898
 - sentinel 784, 787, 788, 789, 791, 792–796
 - merging with list header 792–796
 - separate 1201
 - call 967
 - class 967
 - entity 967–968
 - object 967, 982–990
 - reference 967
 - type 967
 - separate* 952, 967
 - Separateness consistency rule 973, 974, 975
 - Separatist 897–899
 - separator 897–899
 - SEQUENCE* 848, 857
 - sequential 102, 1201
 - dependency 44
 - SEQUENTIAL_TABLE* 504, 505, 831
 - set 410
 - SET* 857
 - set_developer_exception_context* 435
 - set_mismatch_information* 1044
 - setjmp* 414
 - setup* 961
 - Shang, David L. 629, 642
 - Shapiro, Marc 1034
 - share* 622, 633
 - shared memory area 742
 - shared object, see shared under object
 - shared value, see shared under object
 - sharing in databases 1047
 - sharing under repeated inheritance 544–548
 - Shaw, Mary xi, 408, 1079, 1097
 - Shelf 1149
 - selfishness, see principle of selfishness
 - Shell 42
 - Shlaer, Sally 918, 922
 - Shlaer-Mellor method 918
 - Shneiderman, Ben 20
 - Shopping List advice 772
 - shopping list approach to class design 111, 770–774
 - short form 176, 204, 389–392, 803–804, 955, 1147, 1158, 1201
 - displaying in the environment 1158
 - side effect 748–764
 - abstract 757
 - concrete 749
 - forms 748–749
 - legitimate 759–764
 - signal* 414
 - signature 1201
 - Silicon Graphics 1152
 - Simons, Anthony J. H. 629, 642
 - SIMPLE* 201
 - Simpson, G.G. 866
 - Simula 35, 49, 57, 81, 209, 211, 215, 272, 509, 517, 732, 988, 1080, 1099, 1113–1126, 1138, 1139, 1140, 1167
 - previously known as Simula 67

- Simula 1 1114
- Standards Group 1114
- simulated time 1123
- simulation 732, 1122–1126, 1139, 1140
 - continuous 1122
 - discrete-event 1122–1126, 1139, 1140
 - see also modeling
- SIMULATION** (Simula) 1123–1126, 1139, 1140
- sinecure 336
- Single Choice principle 61–63, 65, 592, 1175
- Single Name rule 549
- Single Target principle 184, 185, 215
- SIS (Swedish Standards Institute) 1138
- Sisyphus syndrome 59
- ski team 621
- SKIER** 622, 631, 641
- SKIERI** 625, 628
- skip 716
- sleeping car 522
- slice (of an array) 383
- small classes 714–715
- Small Interfaces, see under interface
- Smalltalk 35, 208, 320, 453, 517, 585, 611, 734, 1050, 1056, 1057, 1075, 1099, 1114, 1126–1130, 1132, 1138, 1140, 1161, 1162
 - assessment 1129–1130
 - efficiency issues 1129
 - Smalltalk 72 1138
 - Smalltalk 76 1138
- SmallVDM 408
- Smith, Glen 1160
- sneaky** 750
- Sneath, Peter H.A. 866, 868
- Snyder, Alan 610
- software
 - and reality 230–231
 - as operational model 732
 - companies 76–77
 - component, see reusable software component
 - fault tolerance, see fault tolerance
 - four worlds of software development 229–230
 - lifecycle, see lifecycle
 - object, see under object
 - process, see lifecycle
 - quality, see quality
- Software Correctness property 333
- software developer as arsonist 201–202
- Software Engineering Institute 55
- software IC™ 672
- software inheritance 825
- SOFTWARE_ENGINEER** 813
- SOFTWARE_ENGINEER_1** 815, 816
- SOFTWARE_ENGINEER_2** 815
- SOFTWARE_ENGINEER_3** 815
- Sokal, Robert P. 866, 868
- SOL (Société des Outils du Logiciel) xi
- Solaris 1152
- solve** 765
- SOMA (Semantic Object Modeling Approach) 919
- Sommerville, Ian 415, 809
- SORTABLE_LIST** 1186
- SORTED_LIST** 1186
- soundness
 - of a garbage collector 305
- source of an attachment 262
- space, see memory management
- spaces, use in software texts 895–896
- special service 998–1002
- specialization and extension 499–500
- specialization versus abstraction 858–859
- SPECIALTY_EMPLOYEE** 853
- specification 5, 150, 1201
 - completeness 153–159
 - reusability 70–71
 - see also: short form; flat-short form; interface under class; requirements document
- Speer, Brian R. 868
- spiral lifecycle 713, 925
- Spivey, J. Michael 160
- sponsor** 909, 912
- SQL 1049, 1050
 - embedded 1049
- SQUARE** 467, 483, 826
- st_number** 692
- stable times of an object's lifecycle 364
- stack 123–162, 280–281, 285, 286, 287, 329, 338–340, 348–364, 500–502, 530–532, 576–577, 610, 857, 1081–1091, 1150
 - bounded 576–577
 - complete abstract data type specification 139
 - package in Ada 1081–1091
 - protected 359–363, 610
 - representations 123–124
- STACK** 166, 318, 329, 500, 502, 530, 540, 576, 610, 857, 882, 989
- stack-based memory management 280–287
- STACK_OF_LINKABLES** 299, 300
- stackexp** 140
- STACKS** (Ada) 1086
- STACK1** 339
- STACK2** 349, 532
- STACK3** 359, 610
- STACK4** 364
- stalactite development process 925, 930
- Stallman, Richard 100
- standard names for features 882–883
- standard_clone** 584
- standard_copy** 583
- Standish, Thomas A. 99
- STARS program (Software Technology for Adaptable, Reliable Systems) 81
- start** 782, 789
- starting_time** 909
- state 1073–1076
 - abstract 756–758
 - as a class 684–686
 - concrete 756–758
 - diagram, see transition diagram
 - in data structures 782–783
- STATE** 685, 686, 689, 690, 694, 731, 734, 1075
- Simula 1117
- state-based concurrency model 979
- STATE_INSTITUTIONS** 834
- static 1201
 - binding, see static under binding
 - memory management 280–287
 - typing, see static under typing
- static-dynamic type consistency 475
- statically typed language 612
- statism 682–683
- Steel, Thomas B. 79
- Stein, Jacob 1061
- Stendhal 121, 148, 269
- step** 961, 962
- Stephan, Philippe 1160
- Stephenson, Ian 294, 305
- Stevens, Peter F. 843, 868

- Stop Point 1159
stop_requested 962
STORABLE 252–253, 592, 1038, 1039, 1045, 1047, 1152
store 252, 253
 Store library 1149, 1152
 Store library for object-oriented-relational combination 1050
store_ignore 1040
 storing object structures, see under object
str_swap 1169
 Strether, Lambert 301
 string 456–457
 constant 653–654
STRING 220, 456, 565, 583, 657, 1186
STRING_COMPARABLE 1185
STRING_swap 1168
 strong condition 335–337, 573
 Stroustrup, Bjarne 35, 557, 1132, 1138
 structural detection policy (persistence) 1042
 structural inheritance, see structure inheritance
 structure inheritance 532, 727, 831–832
 definition 831
 structure type (C) 737, 1109
 structured design 64
 structured programming 334, 1131
 control structures 277
 Stuessy, Tod F. 868
 style 180, 875–902
 and creativity 878
 applying rules in practice 180, 875–876
 assertions 899
 for comments and indexing clauses 886–891
 for constants 884–886
 importance of rules 877–878
 naming rules 879–884
 self practice in this book 878
 terseness and explicitness 876–877
 text layout and presentation 891–901
 subcluster, see under cluster
 subcontract, subcontracting 332, 576, 1201
 subobject 255–261
 no references to subobjects 260–261
 see also composite under object
 subroutine, see routine
 Fortran 1104–1106
 multiple-entry 1104–1106
 subtype 835–836
 subtype inheritance 825–826, 835–843
 definition 825
 subtyping
 enforcing the subtype view 836–837
 versus other forms of inheritance 833–835
 subversion of software architecture 682, 683
 subwindow 524–525, 1070
 success 412
 successive approximation 382, 384
 sufficient completeness 156–159
sufficient_duration 910
 SunOS 1152
super (Smalltalk) 517, 1128, 1140
 super-module, see under module
 supplier 182–183, 862, 1201
 definition 182
 suspended process 1123
 Suzuki, Norihisa 270, 277
 Swanson, E. Burton 17–18, 20, 125
swap 1168
 Swift, Jonathan 104, 672, 750
 Switzer, Robert 948, 1034
 see Strether, Lambert
 Sybase 820, 1152
 Sylla, Karl-Heinz 642
 symbolic constant 45, 644, 884–886
 see also constant under attribute
 Symbolic Constant principle 644, 884
 synchronization 977–982
 versus communication 977
 synchronous call 966, 1202
 Syntax method 918
 system 196–197, 688–690, 1202
 assembling 198–200
 closure 196
 execution 195
 no main program in object-oriented development 197–198
 putting together 194–202
 retrieving 1041
 storing 1041
 validity, see system validity
 system parameter 650
 System Tool 1153, 1156
 system validity 627, 628–641
 error 627
 System Validity rule 634
 system-valid 627, 636
- ## T
- tab characters 894
 table 1150
 in relational databases 1048
TABLE 82, 504
 table searching 74, 81–98
TABLE_HANDLING 97
 Tabourier, Yves 120
 tabular form for requirements 914, 920
tag_name 432
tagged (Ada 95) 1092
tagged_out 582
 Tang, Terry 1033, 1160
tangent_from 887
TAPE 1173, 1174, 1175
 Tardieu, Hubert 120
 target
 of a call 184
 of a tool in the object-oriented environment 1154–1158
 of an attachment 262
 single 184
target 849
 targeting a tool 1154–1159
 task
 in Ada 980, 1091–1092
 to emulate classes 1098
 see also process
 tasks of object-oriented development 922
 taxomania 728, 743, 820–821
 taxon (pl. taxa) 866, 867
 taxonomy
 bibliography 868
 history 864–867
 in the biological sciences 864–868
 limitations 841–843
 of inheritance 824–825
 premature 728–729
 taxonomy exception 837
 yoyo approach 860
 teaching object technology 935–948
 industrial training 935–937
 introductory courses 937–940

- non-introductory courses 940–942
- pedagogy 942–946
- plan 946–947
- TEACHING_ASSISTANT** 520
- template (C++) 1134, 1202
- TEMPORARY_EMPLOYEE** 852
- tenuring 308
- terminated process 1123
- Terminatist 897–899
- terminator 897–899
- terseness 328, 876–877
- Tesler, Larry 715
- test** 849
- TEST** 533
- testing 392–398
- TEX 108, 391, 1159
- text editor 715
- TEXT_FILE** 1174
- textual form for requirements 920
- Thatcher, J. W. 160
- then** 448, 450
- Theophrastus of Eresos 864
- theorem prover 578
- theory, role in software methodology 665
- theta** 763
- Thomas, Pete 948
- thrashing 292
- thread 969, 970, 972, 1012, 1032
- threshold model for object-oriented databases 1053
- time
 - computing 1123
 - simulated 1123
- TIME** 910
- time** (Simula) 1123
- time-sharing 954
- TIMED** 1019
- timeliness 14, 15, 16, 68
 - garbage collector property 305
- token sequence 115
- TOKENIZER** 848
- Tokoro, Mario 1033, 1034
- tolerance, see fault tolerance
- tolerant style for preconditions 354–357, 359–363
- tool
 - in an object-oriented environment 1153–1159
- tool developer 1064, 1067
- toolkit 1064–1065, 1073
- TOOLKIT** 818, 819
- TOOLS conferences (Technology of Object-Oriented Languages and Systems) 641, 642, 674, 1033, 1034
- tools for object-oriented software construction 1143, 1148–1150
- top 679
 - finding 107–108
- top-down design 41, 43, 89, 103–114, 678–684, 1086
 - as a technique for describing rather than producing system architectures 114
- topmost function 41, 104, 116
- total correctness 337
- total function 138
- total order 523
- Toy Story* (movie) 953
- train car 522
- training in object technology, see teaching object technology
- Training Topics principle 936
- traitor 973–976, 977, 989, 1032, 1202
- transaction 1047, 1054–1057, 1061
 - in Versant 1057
 - long 1054–1055
- transient object, see transient under object
- transition** 689, 692
- transition diagram 676–678, 1075, 1076
- transition function 678–679
- transition graph, see transition diagram
- transition label 1072, 1076
- translate** 172, 181, 1071
- transmission, see data transmission
- transparencies containing software texts, choice of layout and font 901
- transparency, see referential transparency
- TRAVERSABLE** 857
- tree 524–527, 1150
 - abstract syntax tree 1038
 - definition 526
 - see also binary tree, binary search tree
- TREE** 525, 567, 607
- tree of life 859, 863
- Trellis 330, 1137, 1188
- TRIANGLE** 467, 483
- triangle of computation 101–103, 964
- trigger** 434
- troff 391, 1159
- Trojan War 671
- True** 220, 452
- trying a type 592
- tuple
 - in relational databases 1048
- Turing, Alan 407
- TV station programming (example) 907–913
- twin**, not needed 274–275
- TWO_WAY_LIST** 596, 599, 600, 625, 710, 880, 1166
- TWO_WAY_TREE** 796, 1070, 1166
- type 24, 167, 497, 1202
 - anchored, see anchored declaration and inheritance 497
 - and module 170, 185, 1095–1096
 - application to development environments 1157
 - as a semantic concept 170
 - base class 325, 602–603
 - basic 93, 171, 190, 191, 220, 222, 226, 228, 234, 254, 256, 257, 258, 263, 265, 272, 276, 616, 636, 650, 656, 881
 - checking, see typing
 - conformance, see conformance, Type Conformance rule
 - consistency 472–473
 - enumerated, see unique value
 - expanded 254–261, 263–265, 412, 445, 470, 616, 975–976
 - and analysis 907
 - and concurrency 975–976
 - definition 255
 - effect on attachment and comparison 263–265, 412, 445
 - properties 258–261
 - role 256–258
- forcing 478, 591–595
- generalization 317–318
- interrogation, see assignment attempt
- parameterization 318–320
- protected (Ada 95) 1094
- redefinition 621–641
- reference 254, 256
 - definition 256
- role 319–320
- rule 323, 613
- separate 967
- static, dynamic 475–476
- tagged (Ada 95) 1094
- trying 592
- type system 171–172
- variable 629–630, 642

- variation (reusability issue) 84, 97
 - versus class 324–325
 - violation 612, 613, 614, 626, 627, 628, 630, 632, 634, 635, 636, 639, 641
 - Type Conformance rule 474, 591, 613
 - Type Redeclaration rule 624
 - type variable, see variable under type
 - type variation inheritance, see type under variation inheritance
 - typed pick-and-throw 1157
 - typeset 634, 635
 - typing 322–323, 595–604, 611–642
 - a little bit typed 618–619
 - and efficiency 616
 - and inheritance 472–480
 - and readability 615
 - and redeclaration 595–598
 - and reliability 615
 - application to development environments 1157
 - becoming obnoxious? 591–592
 - dynamic 612–613, 616, 1128, 1195
 - arguments 616
 - definition 612–613
 - in Smalltalk 1128
 - pessimism 636–638
 - realism 613–614, 632
 - role 319–320
 - static 25–26, 595–604, 612–613, 615–642, 835, 1202
 - basic definitions 612–613
 - definition and advantages 615–621
 - the issue 611–614
 - versus binding 619–621
- U**
- UCSD Pascal 947
 - UML (Unified Modeling Language) 919
 - UNARY_TREE** 604
 - UNCOL 79
 - unconstrained genericity, see under genericity
 - undefine** 552
 - undefinition 485, 551–553
 - see also uneffecting inheritance
 - undergraduate courses 941
 - underline the nouns 700, 720–724
 - understandability 43–44, 48, 50, 54
 - undo** 699
 - UNDOABLE** 717
 - undoing 695–718, 932, 991
 - implementation 705–707, 707–711
 - multi-level 696, 704–707, 716
 - practical issues 696–697
 - purpose and usefulness 695–698
 - requirements on an undo-redo mechanism 697–698
 - user interface 711–712
 - uneffecting inheritance 830
 - definition 830
 - Ungar, David 316, 641, 1139
 - Unified Modeling Language 919
 - uniform access 55–57, 64, 175–176, 203–208, 775, 779, 888
 - Uniform Access principle 57
 - unique value 654–655, 657–659
 - compared to enumerated types 657–659
 - emulated by once function 660
 - UniSQL 1055
 - United Kingdom
 - university curriculum 941
 - unity** 1180
 - Universal Class rule 580
 - universal class, see universal under class
 - universal feature, universal operation, see universal under feature
 - universal machine 79, 1137
 - universe 198
 - Università degli Studi di Milano xi
 - University of Technology, Sydney xi
 - UNIVERSITY_PERSON** 520
 - Unix 12, 72, 199, 414–415, 533, 737, 802, 818, 1063, 1064, 1074, 1152
 - and exceptions 414–415
 - folklore 802, 1063
 - pipe 44
 - Shell 42
 - Unixware 1152
 - unqualified call, see under call
 - unreachable object, see under object
 - unstructured information 1060
 - until** 451
 - untyped, see dynamic under typing
 - US Department of Defense (DoD) 55, 81, 1079–1080
 - US_DRIVER** 544, 545, 546
 - usability, see ease of use
 - usable language 613–614
 - use** (Ada) 1083, 1085
 - use case 112, 738–740
 - Use Case principle 739
 - useful language 613–614
 - Usenet 35, 674
 - user interface 109–110, 711–712, 713, 1152–1159
 - interaction mechanisms 1071–1076, 1152–1159
 - object 1072
 - see also GUI (Graphical User Interface)
- V**
- Valente, Dino 1160
 - Valéry, Paul 725
 - validity
 - see class validity, system validity
 - value
 - semantics 261–265
 - van Ommeren, Robin 1160
 - van Wijngaarden, Aad 100
 - variable, see entity, attribute
 - variant** 451
 - variant, see loop variant
 - variation inheritance 828–830
 - definition 829
 - functional 829
 - type 829
 - uneffecting, see uneffecting inheritance
 - Vax 1152
 - VBX 67
 - VDM 160, 400, 408, 831
 - VDM++ 408
 - vector 585–587
 - VECTOR** 585, 588, 589, 610
 - VEHICLE** 522
 - VENTILATION_SYSTEM** 831
 - verb 720
 - verifiability 14
 - Versant 1055, 1057, 1061
 - versioning, see under class, object
 - Vi editor (Unix) 696, 1074–1075, 1076
 - view inheritance 851–858
 - criteria 856–858
 - not a beginner's mechanism 854
 - when appropriate 854–855
 - violation
 - see under type
 - VIP client 999, 1031
 - virtual
 - function (C++) 514, 620, 1133

- machine 956, 1136
- reality 231
- routine (Simula) 1115
- virtual**
 - C++ 514, 620
 - Simula 1115
- Vision library 820, 1039, 1067, 1146, 1149, 1150, 1152
- Visual Basic 67, 1130
- Vladivostok 976
- VMS 199, 1152
- VOCATION** 815
- Void** 244, 582
- void, see under reference
- Voltaire 498

- W**
- Wagner, E. W. 160
- wait by necessity 952, 987–988, 989, 990, 1006, 1007–1009, 1015, 1027, 1033
- wait condition 990–998
- wait_for_job** 963
- wait_turn** 1000
- Waldén, Kim 34, 271, 277, 517, 642, 745, 772, 901, 919, 921, 922, 934, 1034, 1061
- walking menu 568
- War of the Semicolons 897–899
- Warnier, Jean-Dominique 120
- watch-calculator 522
- WATCHDOG** 1020
- watchdog mechanism 1019–1021
- waterfall model of the software lifecycle 924–925
- Waterloo Fortran 947
- Waters, Richard C. 99
- weak condition 335–337, 573
- Weak Coupling 48
- wean** 311
- weaning an object 311
- Web browser 3, 954, 955, 968, 1134
- Web Crawler 1060
- Web library 1149, 1152
- Weber, Franz 628, 629, 642
- Weber, Mats 1096, 1097
- Weedon, Ray 948
- Wegner, Peter 642, 1100
- weight 157
- Weight Consistency rule 156
- Weiser, Mark 100

- WEL library 442, 818, 1067, 1068, 1149, 1150
- well-formed expression 153–155
- Welsh, Jim 660
- Wexelblat, Richard L. 1112
- Wheeler, David A. 1095, 1097
- when** 450
 - Ada 1089
- widget (for graphical applications) 1066
- Wiener, Richard S. xi, 35, 901, 948, 1138
- Wiley, E.O. 868
- WIMP (Windows, Icons, Menus, Pointing device) 1063
- window 524–525, 1070–1071
 - operations 1070–1071
 - window system 1064–1065
- WINDOW** 511, 512, 524, 560, 561, 567, 817, 818, 1070
- WINDOW_WITH_BORDER** 558, 560, 561
- WINDOW_WITH_BORDER_AND_MENU** 560, 561
- WINDOW_WITH_MENU** 559, 560, 561
- Windows 199, 201, 442, 533, 1064, 1065, 1067, 1068, 1150, 1152
 - Windows NT xiv, 1067, 1152
 - Windows 3.1 1152
 - Windows xiv, 95 1152
- Wing, Jeannette M. 160
- Wintel 11
- wipe_out** 298, 882
- Wirfs-Brock, Rebecca 744
- Wirth, Niklaus xi, 64, 119, 408, 1079, 1097, 1111, 1137, 1139
- with** (Ada) 91, 1171
- withdrawals_list** 364, 368, 369, 1046
- Word, see under Microsoft
- WORK_CONTRACT** 855
- World-Wide Web 35, 954, 955–956, 968, 1032, 1060, 1147
 - replacing databases? 1060
- wrapper 441, 1144, 1178–1187
 - class 1178–1187
 - object 1178–1187
- wrapup** 962
- wristwatch 522
- writable** 883
- write-back 1042

- writer 222
- Wulf, William 1079
- Wyatt, Barbara B. 1033
- WYSIWYG (What You See Is What You Get) 1063, 1065
- x** 763

- X**
- X Window system 947
- Xerox PARC (Palo Alto Research Center) 1126, 1129, 1131
- Xlib 1064
- XmPushButtonCallbackStruct** (example of C/Motif name) 1065
- xor** 454

- Y**
- Yacc 75
- Yahoo 78, 1060
- Yates, Warren 948
- Yehudai, Amiram 642
- yield** 1000, 1035
- Yokote, Yasuhiko 1034
- Yonezawa, Akinori 999, 1033, 1034
- YOCC (Yes! An Object-Oriented Compiler-Compiler) 1151, 1160
- Yourdon, Edward Nash 64, 120, 917, 922
- yoyo approach to classification 860, 863
- Yuksel, Deniz 1160

- Z**
- Z 100, 160, 330, 400, 408, 1167
- Z++ 408
- Zdonik, Stanley B. 1061
- Zen 342–345
- zero** 522, 1180
- Zero Weight rule 157
- Zilles, Stephen 160
- zooming 1150

- 2167 standard (US Department of Defense) 55
- 3E (Environment's Execution Engine) 1147