



Index

- 16-bit integers, 26, 27, 325
- 16-bit Unicode code, 26, 27
- 32-bit integers, 26, 27, 325
- 64-bit integers, 26, 27, 325

- AboutBox assembly
 - Assembly Linker, 162–163
 - building second version, 173–174
 - embedded resources, 158–160
 - linked resources, 160–162
 - revised makefile, 169
- AboutBox class, 154
- AboutBox component
 - developing into assembly, 153–157
 - version 1, 153–154
- AboutBoxBase base class, 153, 154
- AboutBoxBase.cs file, 153
- AboutBox.cs file, 154–155
- AboutBox.dll file, 154–155, 156, 163
- AboutBoxEmbed directory, 158
- AboutBoxGAC directory, 168
- AboutBoxGACV2 directory, 173
- ..\AboutBox.key file, 168
- AboutBoxLinkedAL subdirectory, 162
- AboutBox.net module, 163
- AboutBoxSample namespace, 154
- Abstract classes, 446
- Abstract data types, 454
- Abstract execution environment, 105
- Abstract record types, 387
- Access methods, 422
- Accessibility, 473
- Accessibility levels, 58–59
- Activation record, 141
- Activator class, 84
- Active Oberon, 441
 - abstract data types, 454
 - active object types, 441
 - active objects, 442–446
 - agents, 442–446
 - aggregation, 441
 - assemblies, 468–469
 - assertion-oriented synchronization, 467
 - AWAIT statement, 445
 - base method calls, 468
 - block statement, 442
 - compiling definitions, 460–461
 - compiling modules, 459–460
 - components produced by foreign
 - languages, 464
 - definitions, 441, 446–454, 458
 - delegates, 468
 - elements of module scope, 456
 - exception handling, 468
 - fields, 468
 - IMPLEMENTS relation, 441, 458
 - IMPORTS relation, 441, 458
 - interoperability, 463–466

492 ■ Programming in the .NET Environment

- Active Oberon *continued*
 - intersecting chains of refinement, 452–453
 - intrinsic behavior, 442
 - language fitting, 467–469
 - lightweight object usage, 450
 - mapping active behavior, 466–467
 - mapping definitions, 461–463
 - mapping modules, 459–461
 - module class, 459
 - modules, 455–456, 458
 - multiple inheritance at compile time, 451
 - mutual exclusion, 467
 - namespaces, 464–465, 468
 - nesting procedures, 469
 - OBJECT keyword, 444
 - object model, 451
 - object type automatically running as thread, 442
 - object types, 458
 - overloading methods, 468
 - packages, 455–456
 - passive objects, 442–445
 - pointer-based object types, 444
 - polymorphic declarations, 454
 - polymorphism, 442
 - record types, 468
 - recursive descent strategy, 457
 - REFINES relation, 441, 458
 - reusable components, 465
 - shared resources, 442–443
 - stack engine model, 457–458
 - static modules, 441, 455–456
 - symmetry, 448
 - syntax, 442
 - type-specific object variable declarations, 454
 - USES clause, 442
 - value types, 468
- Active object types, 441, 467
- Active objects, 442–446
- ActiveState Perl Dev Kit Web site, 380
- ActiveX Test Container, 84
- add instruction, 111
- add_EventName method, 37
- add_OnClick() function, 338–339
- AddRef method, 314, 315
- AddressOf expression, 322
- ADO.NET, 223, 254–256
- Advanced interoperability and unsafe code, 340
 - Agents, 442–446
- Aggregation, 441
- Algol 60, 432
- Allen, Paul, 297
- AndAlso operator, 326
- ANSI C, 5
- API (application program interface), 222, 247
- AppDomain class, 121
- Applet base class, 450
- Application configuration files, 200, 205, 208, 211, 217
- Application directory, 131
- Application Domain policy level, 134
- Application domains, 85, 473
 - assemblies, 120–121
 - creation of, 188–189
 - displaying, 121
 - execution system, 119–121
 - loading assemblies into, 189–192
 - manipulating, 121, 188–189
 - passing object handle between, 191
 - versus processes, 186–187
 - threads, 120–121
 - usage, 187
- Applications, 147, 457, 473
 - concepts and services, 13
 - configuration files, 174–176, 195–205, 197–204
 - debugging, 199
 - deploying, 195
 - existing technologies solving problems, 149–152
 - installing, 217–226
 - internationalization, 176–186

- Internet protocols to send and receive data, 261–263
- locale information, 148
- localization, 176–186
- .NET Framework, 12–13
- non-ASCII string technology, 176
- private assemblies, 164
- retrieving current information, 187–188
- reuse, 13
- search paths, 149–150
- separation of code and user interfaces, 177–178
- simplifying development, 237–238
- string literals, 177
- symbolic names, 150
- text-based configuration files, 196–197
- traditional model for building, 147–148
- Trojan Horse components, 148
- Unicode, 176–177
- version conflicts, 148
- versioning-related technologies, 150–151
- well-known locations, 149
- Windows Registry, 151–152
- apply methods, 408–410
- Array classes, 48
- Array objects, 49
- Array types, 307
- ArrayList class, 250, 253, 335–336
- Arrays, 391, 474
 - bounds for each dimension, 307
 - Component Pascal, 386, 391–394
 - CTS (Common Type System), 392
 - encoding rank within type, 306
 - entire assignment, 393
 - as fixed-size structure, 435
 - lower bounds of each dimension, 307–308
 - reference surrogates, 391–394
 - Visual Basic .NET, 306–309
- ASCII character set, 27, 273–278
- ASP (Application Server Pages), 235–237
- ASP pages, HTML generated for, 215–216
- ASP.NET, 12, 230
 - building applications, 287
 - configuration files, 200
 - configuration options, 199
 - events using standard HTML and ECMAScript, 213
 - functionality of, 287–289
 - server controls, 213
 - Web controls, 213–216
- Assemblies, 14, 32, 152–153, 457, 474
 - acting in type-safe manner, 113
 - Active Oberon, 468–469
 - application domains, 120–121
 - Assembly Linker, 162–163
 - assembly manifest, 152–153
 - authenticity, 165
 - Authenticode signature, 165
 - binding to different version, 174–176
 - building second version, 173–174
 - caches, 165–168
 - checking, 113–114
 - checking types, 114
 - configuration files, 208
 - delayed signing, 172
 - development of AboutBox component as, 153–157
 - domain-neutral, 120
 - domain-specific, 120
 - download cache, 166
 - embedded resources, 157–162
 - GAC (Global Assembly Cache), 165–166
 - generating from type library, 282
 - identities, 172
 - information about, 61, 87
 - isolating, 212
 - linked resources, 157–162
 - linking files and modules into, 162–163
 - loading into application domains, 189–192
 - localization, 178
 - makefile files, 154, 156–157
 - managing caches, 166–168
 - metadata, 113–114
 - nmake utility, 154–155

494 ■ Programming in the .NET Environment

- Assemblies *continued*
 - PerlNET Component Builder, 377
 - permission sets, 133, 134–141
 - permissions, 130–131
 - policy evaluation, 139
 - private, 163–172, 208
 - public, 163–172
 - public and private key pair, 164, 168
 - referencing external, 210
 - referencing with codeBase element, 208–211
 - Registry entries, 99
 - self-describing, 88
 - signed, 161, 165–166
 - strong names, 88, 164–165, 168, 172, 208
 - trust issues, 165
 - type library description, 99
 - verifying for type safety, 113–114
 - version discrimination, 174
 - version number, 88–89
 - visibility, 391
 - Win32 version information, 172
- Assembly accessibility level, 58
- Assembly Linker, 162–163, 168, 172, 184, 209
- Assembly manifests, 86–92, 152–153, 161–162, 474
- assemblyBinding subelement, 198
- AssemblyBuilder.GetManifestResourceStream() function, 160
- assemblyIdentity element, 198, 211
- AssemblyQualifiedName property, 68, 70
- Assert model, 141–142
- Asserting permissions, 141–142
- Assertion-oriented synchronization, 467
- Assertions, 474
- Assignment compatibility, 54–56
- Assignment-compatible types, 40
- Asynchronous execution pattern, 237
- Attribute class, 79
- Attributes, 337–338
- AttributeSample class, 79, 80
- AttributeUsage attribute, 78
- Australian culture, testing, 185
- AuthorAttribute class, 79
- AWAIT statement, 445
- AWT (Abstract Windows Toolkit), 11
- Backing stores, 259–260
- Base classes, 248–250, 304
 - implicit inheritance from, 15
 - overloaded virtual methods, 360
- Base Framework, 8, 10, 26, 229, 474
 - interface types, 48
 - object-oriented design, 232–233
- Base methods, 468
- BaseType property, 68, 70
- BASIC, 297
- Basic type system, 116
- BASIC-A, 297
- BCL (Base Class Library), 26, 127, 221, 229
 - predefined standard attributes, 80–83
- BEGIN blocks, 372
- bin directory, 149
- Binary configuration files, 196
- Binding information, 85
- Binding to different version assemblies, 174–176
- bindingRedirect element, 175, 198
- Block statement, 442
- BOA (Basic Object Adapter), 105
- bool type, 27
- Boolean class, 252
- Boolean type, 26, 27, 43, 252
- Boolean values, 26, 27
- box instruction, 106
- Boxed types, 39–40, 46, 309–310
- Boxing, 332–333, 474–475
 - Hotdog Scheme, 412
- Boxing value types, 39
- Browsing support, 263–266
- BSD UNIX, 220
 - PAL (Platform Adaptation Layer), 222
 - SSCLI (Shared Source Common Language Infrastructure), 224–226
- Buffering stream, 260
- Builder classes, 93

- Built-in data types, 475
- Built-in value types, 25–29
- Button class, 334
- ButtonPenControl abstraction feature, 454
- ButtonSensor abstraction feature, 453
- ByRef variant behavior, simulating, 310–311
- Byte class, 251
- byteBuffer instance, 340
 - header file, 61
 - heap allocation, 122
 - primitive built-in types, 21
 - standard library, 231
 - user-defined types, 21
- C#, 404
 - += operator, 338
 - = operator, 338
 - abstract classes, 446
 - attributes, 337–338
 - boxing, 56, 332–333
 - building assembly, 156
 - code references types, 156
 - combining delegates, 338
 - combining with Mondrian in Sieve of Eratosthenes, 435–437
 - comfortable for C++ programmers, 331
 - compiler, 158
 - component-oriented development, 333–339
 - defining default indexed property, 242–243
 - defining interfaces, 333
 - definition and use of events in value types, 37–38
 - definition of function with params keyword, 343–344
 - delegates, 338
 - design goals, 330–332
 - destructor, 41
 - divide-by-zero exception, 431
 - entry point for program, 33
 - enumerators, 341
 - events, 338–339
 - examining policy levels and permission sets, 139–141
 - fitting into CLR (Common Language Runtime), 331
 - flexibility, 331–332
 - foreach statement, 73, 341–342
 - function value returned, 427
 - future, 347–348
 - general collection classes, 411
 - generating IL (Intermediate Language), 106–112
 - generating user interfaces, 437
 - generics, 347–348
 - GUI executable, 156
 - HelloWorldCS class, 17–18
 - history, 329–330
 - indexers, 335–336
 - /linkresource: option, 160
 - makefile file, 156
 - operator overloading, 336–337
 - parameterized property, 336
 - parameterized types, 347
 - params arrays, 343–344
 - params keyword, 43
 - pointers, 339–340
 - properties, 333–335
 - properties as virtual fields, 335
 - reference types, 332–333
 - remove function, 339
 - removing delegates, 338
 - Sieve of Eratosthenes, 435
 - simplifying C++ model, 331
 - sorting example, 426
 - stack component example, 344–347
 - standardization, 348
 - strings, 46
 - struct, 34
 - supertype, 424
 - supporting component-centric development, 332
 - switch statement based on string values, 342–343
 - type system, 332–333
 - typeof keyword, 70–71

496 ■ Programming in the .NET Environment

- C# *continued*
 - unsafe code, 339–341
 - usage, 316
 - user-defined types, 333
 - using construct, 261–262
 - value types, 332–333
 - XML comments, 344
- C++, 234
 - abstract classes, 446
 - applications running in semi-trusted environments, 232
 - classes, 21
 - header file, 61
 - heap allocation, 122
 - HelloWorldCPP class, 15–16
 - history, 329–330
 - iostream library, 231
 - libraries, 21
 - multiple inheritance at runtime, 451
 - object-oriented design, 232–233
 - RTTI (run-time type information), 3
 - supporting .NET run-time environment, 330
- CalculateValue method, 299
- call instruction, 109–110
- Call-by-name, 432
- Calling
 - .NET objects, 279–282
 - other CLR-hosted languages, 433–434
- callvirt instruction, 106
- Camel casing, 240
- Canonical language, 462
- CAS (Code Access Security, aka Evidence Based Security), 476
- Case-insensitive languages, 240
- Case-lambda special form, 410–411
- Case-sensitive languages, 240
- Casing, 239–241
- Catch handler, 319
- .cctor constructor, 74
- char array overload, 244
- Char class, 252
- CHAR type, 388
- Characters, 27
- CIL (Common Intermediate Language), 25, 106, 384, 476
 - classifying verification status, 114–115
 - executing un-type-safe, 113
 - illegal, 115
 - lack of optimization, 112
 - legal, 115
 - not type-safe, 114
 - type-safe, 114–115
 - verifiable, 115
- cil keyword, 109
- CIL/IL (Common Intermediate Language/Intermediate Language), 103
- Circle type, 387, 399, 400
- Circular references, 314–315
- class keyword, 31, 333
- Class libraries, 221–222, 236, 480–481
 - consistent, 239
 - indexed properties, 242–243
 - member usage, 241–248
 - method usage, 243–245
 - multilanguage, 239
 - naming guidelines, 239–241
 - object-oriented design, 232–233
 - predictable, 239
 - properties versus methods, 241–242
 - secure, 239
 - type usage, 245–248
- class method, 427
- Classes, 13, 40–47, 50, 104
 - base for, 10
 - C++, 21
 - constructors, 74
 - defining cultures, 256–257
 - dotted names, 390
 - identity, 127
 - inheritance, 15, 236
 - in-memory data storage and manipulation, 250, 253–254
 - versus interfaces, 246
 - late binding, 319
 - methods, 66

- naming, 241
 - private, 390
 - public, 390
 - reflection, 65–69
 - as reusable components, 465
 - runtime, 390
 - use as array, 335–336
 - Visual Basic, 298–299
 - Visual Basic .NET, 298–301
- Clear method, 49
- CLI (Common Language Infrastructure), 8, 477
 - assembly boundary, 389
 - configuration files, 197–204
 - non-Windows platforms, 220
- Closure, 407–411, 429, 432
 - environments, 409
 - general implementation, 408–409
 - Hotdog Scheme compiler, 414
 - multiple entry points, 410–411
 - variable argument lists, 409–410
- Closure class, 408–409, 414
 - case-lambda form, 411
- CLR (Common Language Runtime), 2, 8, 404, 406, 477
 - automatic memory management, 121–126
 - delegates, 428
 - execution system, 9, 14
 - field-only class, 422
 - functional languages, 418
 - mapping Component Pascal to, 389–396
 - metadata system, 9
 - parametric types, 424, 425
 - primitive types, 421
 - programming language interaction, 22–23
 - subtyping through class inheritance, 423
 - surrogate array, 393
 - type system, 9, 13–14
 - Windows PE (Portable Executable) file layout, 95
- CLR programs, 116
- CLS (Common Language Specification), 9, 233, 477
 - array types with fixed lower bounds of zero, 308
 - CLS Consumers, 476
 - CLS Extenders, 476
 - CLS Frameworkers, 476
 - CLS-compliant, 475
 - COBOL
 - conversion from string to char array, 244
 - HelloWorldCOB class, 16–17
 - Code
 - reusing, 236
 - separation from user interfaces, 177–178
 - specifying location, 49–50
 - Code Access security (aka Evidence Based Security), 273
 - Code groups, 135–138
 - codeBase element, 208–211, 218
 - CodeBase property, 207
 - Code-based security, 143–144
 - Coercion, 476
 - Collection classes, concrete implementations, 250
 - CollectionBase class, 253
 - COM (Component Object Model), 1, 6–7, 104–105, 297
 - aggregation, 451
 - attributes, 34
 - components, 223
 - concepts mapped to .NET concepts, 300
 - Currency type, 312
 - Date variable, 312
 - Decimal type, 312
 - delegation, 451
 - dynamic type discovery, 84
 - event interfaces, 321
 - events, 37
 - IDL files, 63–64
 - implementation inheritance, 451
 - interoperating with, 281–282
 - IUnknown interface, 39
 - late binding, 317, 319
 - limitations, 7

498 ■ Programming in the .NET Environment

- COM (Component Object Model) *continued*
 - multiple programming languages support, 233
 - objects deterministically finalized, 314
 - one-dimensional arrays, 307
 - proxy objects and implementation code generation at IDL compile time, 105
 - reference counting mechanism, 314
 - Safearrays, 306–307
 - transferring dates from, 313
 - type library, 6, 64, 282
 - unit of identity changes, 299–300
- COM Interop (COM Interoperability), 99–102, 280, 281–282, 476
 - altering assembly name, 282
 - generating assembly from type library, 282
 - mechanisms, 226
 - pointers, 340
- COM objects
 - accessing from .NET objects, 280
 - instance of, 84
- Command object, 255
- Common functionality, 236
- Compact Framework, 220–223
 - _com_params_ variable, 360
- Comparing strings, 44
- Compiler-controlled accessibility level
- Compilers directly generating metadata, 64
- Compiling for the .NET Common Language Runtime, 401
- Component Pascal, 383
 - 8-bit formal parameters, 391
 - Ada-like annotations, 386
 - alias reference, 388
 - arrays, 386, 391–394
 - box instruction, 395
 - built-in types, 386, 387–388
 - choice statements, 388
 - compatibility extensions, 399
 - controlling visibility, 390–391
 - covariant function types, 398
 - dispatched methods, 396
 - enhancing symbol table mechanisms, 400
 - exception-handling facilities, 399
 - extensible records, 394
 - final methods, 387
 - fixed-length arrays, 388
 - hello world program, 385
 - inherited methods, 387
 - invoking constructors with arguments, 400
 - library methods throwing exceptions, 399
 - mapping program structure, 389
 - mapping to CLR, 389–396
 - mapping type system, 391–394
 - methods, 387
 - methods associated with record types, 396
 - module structure, 389
 - modules, 390
 - nested procedures, 388, 390
 - nonlocal addressing, 397–398
 - object-oriented language, 385
 - open array construct, 386
 - optional language extensions, 385
 - performance, 400–401
 - pointers, 386
 - polymorphism, 386
 - procedure variable types, 398
 - record type definitions, 386
 - record types implementing defined interfaces, 399
 - records, 386
 - reference class, 395
 - reference surrogates, 391–394
 - sealed types, 387
 - semantic challenges, 397–398
 - single implementation inheritance, 385
 - statements, 388
 - static procedures, 391
 - static record types, 394–395
 - static variables, 391
 - static variables and local variables of array or record types, 387
 - structural compatibility of delegates, 398
 - synthetic static class, 390, 391

- type case statement, 388
- type system, 386–388
- type-bound procedures, 387, 396
 - as type-safe language, 384
- value classes, 395
- value semantics, 388
- verifiable code, 384
- XHR (explicit heap-allocated records), 397–398
- Component Pascal Web site, 401
- Component-oriented development, 333–339
- Components
 - consumers and producers of, 458
 - internationalization, 176–186
 - local representation of, 99
 - localization, 176–186
 - packaging for other programs, 86–92
 - reusing, 31
 - Trojan Horse, 148
 - version conflicts, 148
- Compound operators, 326
- _com_return_type_variable, 15, 360
- Concatenate method, 45
- Concatenating strings, 45
- Concrete classes, 446
- Concurrency support, 419–420
- Concurrent Hope+C, 420
- Conditional expressions, 135
- .config file extension, 175, 200
- /configSections element, 201
- configuration element, 198
- Configuration files
 - application, 200
 - applying different versioning policies, 174–176
 - ASP.NET, 200
 - assemblies, 208
 - binary, 196
 - choosing, 204–205
 - CLI, 197–204
 - codeBase element, 208, 218
 - custom configuration sections, 200–204
 - format, 197–198
 - GAC, 211
 - machine, 199–200
 - schema, 198–199
 - security, 200
 - testing, 176
 - text-based, 196–197
 - XML files, 197
- Connection object, 255
- Connection point, 321
- const string, 271
- ConstructorInfo class, 66, 73
- ConstructorName type, 66
- Constructors, 50, 66, 74
 - arguments, 85
 - strings, 44
- Content hosted on slow resource, 166
- Contracts, 114, 477
- Control object type, 448
- Conversion method, 45
- Copying files to computer, 217
- COR namespace, 359
- CORBA (Common Object Request Broker Architecture), 1
 - attributes, 34
 - dynamic type discovery, 84
 - Event Service, 37
 - events, 37
 - IDL files, 63–64
 - interface repository, 6, 64
 - interoperability, 105
 - limitations, 7
 - multiple programming languages support, 233
 - Object interface, 39
 - platform independence, 105
 - POA (Portable Object Adapter), 5
 - proxy objects and implementation code generation at IDL compile time, 105
- CORBA specification, 99
- cordbg, 116–119, 121, 191
- Core DOM Level 2, 293

500 ■ Programming in the .NET Environment

- corhdr.h header file, 96
- Covariant function types, 398
- CPAN (Comprehensive Perl Archive Network), 371
- CPython, 353, 358
 - extension modules, 357
 - parser, 355, 359
 - python modules, 357
 - type object, 356
- CPython API, 357
- create construct, 434
- CreateChildControls method, 213
- CreateComInstanceFrom method, 84
- CreateFontIndirect method, 280
- CreateInstance method, 84, 85, 190
- CreateInstanceFrom method, 84
- CreateNewInstance method, 301
- Cross-language
 - compatibility, 18
 - inheritance, 236
 - method invocation, 4
- cryptographySettings element, 199
- C-style APIs, 234
- C-style enumerations, 247
- .ctor instance constructor, 74
- CTS (Common Type System), 389, 391, 457
 - arrays, 392
 - value classes, 392, 394
- Culture tags, 178
- CultureInfo class, 182, 185, 256
- Culture-neutral resources, 180, 185
- Culture-neutral strings, 182
- Cultures
 - classes defining, 256–257
 - country/region code, 178
 - information, 85
 - language code, 178
 - printing out date, 256–257
 - valid and invalid strings, 185
- Culture-specific data, 257–259
- Currency type, 312
- Current method, 48
- CurrentUICulture, 258
- Custom attributes, 19, 67, 104, 477
 - as classes, 77
 - compilers not understanding, 77
 - definition and use of, 77–80
 - inheritance, 77
- Custom configuration sections, 200–204
- Custom protocols, 289
- Customizing
 - encryption providers, 199
 - virtual methods, 248
- Data, separate and distinct from data stores, 255
- DataAdapter class, 255
- DataReader class, 255
- DataSet class, 255, 293
- Date data type, 313
- Date variable, 312
- Dates, 312–313
- DateTime class, 313
- DCE (Distributed Computing Environment's) RPC (Remote Procedure Call), 6
- DCOM (Distributed Component Object Model), 6
- Debugging, 116–119, 199, 477
- Decimal class, 252
- Decimal type, 312
- Declarative security check, 477–478
- Declarative security constraints, 144–145
- Deep copy, 42
- Def statements, 325
- Default constructor, 32
- Default indexed properties, 243
- Definitions, 96–97, 441
 - Active Oberon, 458
 - compiling, 460–461
 - derived from foreign classes, 465
 - implementing, 447
 - mapping, 461–463
 - refining, 447–448

- units of service, 447
- Delayed signing, 172
- Delegates, 37, 49, 468
 - C#, 338
 - declaration, 52
 - multicast, 321
 - returning of void, 52
 - structured compatibility of, 398
 - Visual Basic .NET, 321–324
- Delphi, 234
- Demand method, 144
- Demand set, 478
- Demands, 478
- Denial of permission, 478
- Deny operation, 141
- Denying permissions, 142
- dependentAssembly element, 175, 198
- Deploying applications, 195, 205–207
- Derived class, 304
- Design guidelines, 238–248
- Destructor, 41
- Deterministic finalization, 314–316
- DHTML (Dynamic Hypertext Markup Language), 235
- DII (Dynamic Invocation Interface), 84
- Directories, symbolic names for, 150
- DirectXWrapper assembly, 282
- Dispatched methods, 396
- DISPLAY keyword, 17
- Dispose method, 287, 315–316
- Divide-by-zero exception, 431
- .DLL file extension, 22
- DLL hell, 87, 148, 164
- DllImportAttribute standard attribute, 83
- DLLs (dynamically linked libraries), 61, 87, 280
- /DOC compiler, 344
- DOM (Document Object Model)-level XML reader and writer, 237
- DOM node subtree, 293
- Domain-neutral assemblies, 120
- Domain-specific assemblies, 120
- Dotted names and classes, 390
- Double class, 252
- Downcasting, 56
- Download cache, 166
 - downloaded assemblies, 218
 - listing contents, 212
 - managing, 166–168
- Downloading
 - files to computer, 217–218
 - Web content, 205–216
- DTD schema validation, 293
- dx7vb.dll assembly, 282
- DxVBLib namespace, 282
- Dynamic discovery of types, 84–86
- Dynamic function, 427
- Dynamic invocation, 266
- Dynamic language support, 364–365
- Dynamic type checking, 20, 412–413
- Dynamic types, 92–95
- Dynamically emitting new types at runtime, 269–271
- Dynamically type-checked language, 21
- ECMA CLI standards, 220–222
- ECMA Web site, xxvi – xxvii
- Eiffel
 - multiple inheritance at runtime, 451
 - object-oriented design, 232–233
- Embedded resources, 157–162
- Embedded strings, replacing with string resources, 177
- EmployeeCollection class, 253
- EmptyTypes field, 69
- en-AU directory, 184
- en-AU.resources.dll assembly, 184
- Encapsulation of privileged operations, 229
- Encrypting stream, 260
- Encryption providers, customizing, 199
- Enforced asymmetry, 450
- Enqueue method, 359
- Enterprise policy, 137
 - configuration files, 200

502 ■ Programming in the .NET Environment

- Enterprise policy *continued*
 - levels, 134
- enterprisesec.config file, 200
- EntryPoint class, 36, 52
- EntryPoint object type, 53
- enum keyword, 31
- Enumerating over objects, 341–342
- Enumeration namespace, 31
- Enumerations, 21, 233, 246
 - base class, 248
 - defining, 31
 - usage, 247–248
 - user-defined, 30–31
- EnumerationSample value type, 31
- Equality, 24
- Equals method, 41
- Error handling, 4
 - structured, 319–321
- ETH programming languages history, 442
- eval statement, 375
- Event interfaces, 321
- EventClass class, 38
- EventClass value type, 37
- EventInfo class, 66
- EventInfo object, 67
- Events, 14, 22, 50, 66, 233, 238, 478
 - adding and removing handler, 322
 - C#, 338–339
 - declaring, 322
 - defining methods, 37
 - delegate types, 322
 - delegates, 321–324
 - interface types, 47
 - Visual Basic .NET, 321–324
- Evidence, 131
 - passing to security system, 132–133
 - permission sets based on, 134–141
- Evidence-based security (aka Code Access Security)
 - evidence, 131–132
 - permissions, 130–131
- Exact type, 40, 478
- Exception handling
 - Active Oberon, 468
 - Mondrian, 419
 - PerlNET, 379
- Exceptions, 5, 115, 248, 478
- .EXE file extension, 22
- exec method, 409, 410
- Executable files, 62, 478
- Execution engine, 63
- Execution environments, 105–106, 113
- Execution issues, 4
- Execution permission set, 139
- Execution system, 9, 14, 116, 479
 - application domains, 119–121
 - erroneous or destructive behavior, 113
 - layout of objects in memory, 28
 - making decisions about run-time values, 28
 - versus other component models, 104–106
 - platform independence, 103
 - security, 19
 - versioning on types, 19
- export statement, 389
- Extensible records, 394
- Extension modules, 357
- Family accessibility level, 58
- Family and Assembly accessibility level, 58
- Family or Assembly accessibility level, 58
- FCL (Framework Class Library), 29, 229–230
 - garbage collection, 232
 - namespaces, 248–295
 - object-oriented design concepts, 236
 - simplifying application development, 237–238
- FieldInfo class, 66
- Field-only class, 422
- Fields, 14, 50, 66
 - Active Oberon, 468
 - instance, 33
 - interface types, 47
 - properties synonymous with, 241

- static, 33
- WithEvents modifier, 322
- FieldType property, 66
- Figure abstract base type, 399
- Figure abstraction, 448
- Figure type, 387, 396, 445
- File handles, 236
- File system
 - reference types, 24–25
 - value types, 24–25
- FileIOPermission, 273
- Files
 - copying to computer, 217
 - downloading, 217–218
 - symbolic links, 150
- FileStream class, 259, 260, 273
- filter function, 436
- Finalization methods, 124
- Finalize method, 41, 42
- FindInterfaces method, 70
- FindMembers method, 70
- fixed statement, 340
- Fixup, 479
- Floating-point types, 28–29
- for loop, 388
- foreach statement, 48, 341–342
- Format method, 45
- Format strings, 45
- Form_Load method, 287
- from function, 436
- FullName property, 68
- FullTrust permission set, 139
- Fully qualified string names, 263, 264
- Function pointers, unmanaged, 49
- Functional languages, 417–418, 433
- Functions
 - compiled into class with well-known method, 426–427
 - Mondrian, 426–433
- GAC (Global Assembly Cache), 113–114, 165, 168
 - configuration files, 211
 - fully signed assemblies, 166
 - managing, 166–168
 - modifying message.txt file, 170–171
- gacutil, 168–170
 - /l option, 173
 - /ldl option, 212
- Garbage, 232
- Garbage collector, 40, 123–124, 226
- Gardens Point Component Pascal compiler, 383–385
- Gates, Bill, 297
- GC (Garbage Collection), 14, 479
 - generation-based, 125–126
 - long-lived objects, 124
 - object's generations, 124
 - Visual Basic .NET, 314–316
- __gc keyword, 15
- gdi32.dll, 280
- General-purpose configuration handlers, 201
- Generation-based garbage collection, 125–126
- Generic C#, 419
- GenericIdentity class, 127
- GenericPrincipal class, 127
- get method, 34, 45, 52, 68
- Get procedure, 458
- GetAboutText() method, 153, 154
- GetAddMethod method, 66
- GetAverage Web service, 291–292
- GetBaseDefinition method, 66
- GetConstructors method, 70
- GetCustomAttributes method, 67, 77, 80
- GetEnumerator() function, 341
- GetHashCode method, 41
- GetLength method, 48, 49
- GetManifestResourceStream() function, 162
- GetMembers method, 67, 71
- GetMethod method, 68
- get_methods, 35
- GetMethods method, 70
- GetNestedTypes method, 70

504 ■ Programming in the .NET Environment

- GetObject method, 84, 258
- GetParameters method, 66
- GetRaiseMethod method, 66
- GetRemoveMethod method, 66
- GetString method, 258
- GetType method, 41–42, 64, 65, 70, 71, 74, 263
- GetValue method, 66
- get_X method, 36
- GIOP (General Inter-ORB Protocol), 105
- GJ [8] for Java, 419
- Glasgow Haskell Compiler, 439
- Got Dot Net Web site, 349
- Grades class, 292
- grades class, 291
- Grades Web service, 292
- Grades.asmx file, 290
- Grant set, 479
- Grouping related types, 87
- GUI (graphical user interface) applications, 11
- GUIDs (globally unique identifiers), 5
- GW-BASIC for DOS, 297

- Hammond, Mark, 353
- Handles clause, 322
- Hashes, 131
- Hashtable class, 250
- Haskell
 - function declaration, 426
 - homogeneous list, 424
 - list of integers, 422–423
 - type for coordinate, 421
- Header file, 61
- Heap-allocated memory, 122–123
- Heaps, 95–96, 232
- Hello World program, 14–18
- HelloATL component, 99–101
- hello.exe file, 359
- hello.py file, 358
- HelloWorldCOB class, 16–17
- HelloWorldCPP class, 15–16
- HelloWorldCS class, 17–18
- HelloWorldForm class, 380
- HelloWorldPY class, 15
- HelloWorldVB class, 16
- Heterogeneous generic types, 424
- Hide by name, 302
- hidebysig keyword, 108
- Homogeneous list, 424
- Hotdog Scheme compiler, 403–404, 406, 415
 - boxing, 412
 - call/cc, 413
 - case-lambda special form, 410–411
 - closure, 414
 - closure arguments, 408
 - implementations, 407–413
 - interactive environments, 414
 - JIT compiler, 407
 - performance improvements, 414–415
 - type checks, 414–415
- HTML (Hypertext Markup Language), 235, 236
- HTML generated for ASP pages, 215–216
- httpHandlers section, 201
- Hungarian notation, 240

- IChanged interface, 52
- ICloneable interface, 42
- ICollection interface, 253, 344
- IComparable interface, 49, 55
- Identifiers, 240
- Identities, 24, 126–128
- Identity object, 127
- IDictionary interface, 202
- IDispatch interface, 84, 317, 318
- IDisposable interface, 315
- IDL (Interface Definition Language), 6–7, 63–64, 105
 - extensions, 76
 - interfaces, 7
- IDL (Interface Definition Language) files, 63–64
- IDL-based systems, 6–7
- IDL-to-language mappings, 105

- ID-to-strong mapping, 177
- IEEE floating-point standard, 28
- IEnumerable contract, 250, 253
- IEnumerable design pattern, 341
- IEnumerable interface, 341–342
- IEnumerator interface, 48
- IIdentity interface, 127
- IIOP (Internet Inter-ORB Protocol), 105
- IL (Intermediate Language), 480
 - generating, 106–112
 - machine independence, 106
 - object-oriented, 106
 - verification, 112–116
 - viewing code, 271
- ilasm, 384
- ILDASM (Intermediate Language Disassembler), 75, 94, 271
- IList interface, 344
- Illegal CIL, 115
- Imperative security check, 479
- Imperative style security constraints, 144–145
- Implementation repositories, 63
- IMPLEMENTS relation, 441, 446, 454, 458, 461
- import statement, 357, 389
- Importing types, 22–23
- IMPORTS relation, 441, 458
- INamingContainer interface, 213
- Indexed properties, 242–243
- Indexers, 335–336
- IndexOf method, 45
- IndexOfAny method, 244
- inetd process, 196
- inetd.conf file, 196
- Info classes, 67
- Inheritance
 - classes, 15
 - cross-language, 236
 - custom attributes, 77
 - interface types, 47
 - name hiding, 302
 - value types, 39
 - Visual Basic .NET, 301–304
- Inheritance demand, 480
- Inherits keyword, 16
- __init__ constructor method, 15
- Initializers, 326
- Inner type, 56
- Insert method, 45, 85
- Installing applications
 - copying files to computer, 217
 - downloading files to computer, 217–218
 - installation programs, 218–219
- Installing .NET Framework, 219–220
- InstallShield, 219
- Instance fields, 14, 33, 480
- instance method, 427
- Instance methods, 32–33, 47–48, 480
- Instances, 14
- int type, 110
- Int16 type, 26, 27, 251
- Int32 type, 26, 27, 29, 43, 251
- Int64 type, 26, 27, 251
- Integer array type, 307–308
- INTEGER type, 396
- Integer type, 325
- Integer variables, storage locations, 109
- Integers, 3
 - native, 28
 - signed, 9, 28
 - size, 27
 - unsigned, 9, 28
- interface keyword, 333
- Interface reference, 53
- Interface repositories, 63
- Interface types, 13, 39, 47–49, 48, 480
- Interfaces, 7
 - versus classes, 246
 - contracts for collections, 250, 253
 - immutability, 5
 - in-memory data storage and manipulation, 250, 253–254
 - instance methods, 47–48

506 ■ Programming in the .NET Environment

- Interfaces *continued*
 - late binding, 319
 - as reusable components, 465
 - use on value types, 53–54
- Internationalization, 176–186
- Internet Explorer
 - deploying applications, 205–207
 - security system, 207–208
- Internet permission set, 137
- Internet protocols to send and receive data, 261–263
- Internet zone, 132
- Interoperability, 2, 99–102, 456–458, 463–466, 480
- Intersect method, 130
- Interworking Specification, 99
- IntList class, 423
- IntPtr class, 26, 252
- invo construct, 434
- Invocation off subclasses, 268–269
- Invocation support, 266–269
- Invoke method, 66, 266
- InvokeEvent method, 38
- InvokeMember method, 70
- I/O (input/output), 259–261
- IPermission interface, 130, 273
- IPoint interface, 52, 53
- IPrincipal interface, 127
- IPyType interface, 356–357, 367–368
- Is methods, 68
- Is properties, 68
- IsAbstract property, 66, 70
- IsAutoLay property, 68
- IsAutoLayout property, 68
- IsConstructor property, 66
- IsDefined property, 67
- IsExplicitLayout property, 68
- IsIn property, 67
- IsInRole method, 129
- IsInterface method, 70
- IsLayoutSequential property, 68
- IsMatch method, 276
- IsOut property, 67
- IsPublic method, 70
- IsSealed method, 70
- IsSynchronized method, 49
- IsValueType method, 70
- IUnknown interface, 298
- Java, 2
 - abstract classes, 446
 - building on abstraction, 448
 - class library, 233
 - general collection classes, 411
 - inheritance of interfaces, 21
 - object-oriented language, 233
 - object-oriented type system, 21
 - primitive built-in types, 21
 - sand-box, 233
 - semi-trusted execution, 232
 - single-implementation inheritance model, 22
 - supertype, 424
- JIT (just-in-time) compilation, 105
 - optimization, 112
 - verification and, 115
- JIT (just-in-time) compiler, 385, 406–407, 480
- JIT (just-in-time) evaluation, 418, 430–433, 436
- JPython, 352, 353
- JSP (Java Server Pages), 235–237
- JVM (Java Virtual Machine), 105, 113, 404, 406
- Kernel profile, 221
- Language
 - implementers, 418
 - interoperability, 6
- Language-specific metadata facilities, 61
- Late binding, 317–319
- Late-bound invocation, 267–268
- Launcher type, 465
- ldstr instruction, 110
- Legal CIL, 115
- length function, 423
- Length method, 45

- Length property, 46
- Let assignment, 316–317
- lfaceName string, 280
- Lifetime, 481
- Link type, 444
- Linked resources, 157–162
- Link-time demand, 481
- LISP, 232
- List value, 425
- Load instructions, 481
- Local Intranet zone, 131
- Local variable array, 109
- Local variables, 109, 122
- Locale, 256
- Localization
 - assemblies, 178
 - concepts, 178–180
 - cultures, 178–179
 - resource fallback process, 179–180
 - Windows, 178
- Localized application, 181–182
- Locals, 481
- .locals directive, 109
- Locations, 40, 481
- LOGFONT type, 280
- Logical backing store, 241
- Logical fields, 34
- Long type, 325
- Long-lived objects, 124–125

- Machine configuration files, 199–200, 205, 208
- Machine Policy configuration files, 200
- Machine policy level, 134
- machine.config file, 199, 201
- Main method, 18, 33, 80, 106–110
- make tool, 154
- makefile files, 154, 156–157, 173, 214
- Managed APIs, 95
- Managed code, 14, 481
 - calling unmanaged functions, 279–281
 - COM objects, 281–282
- Managed data, 14
- Managed Extensions for C++, 15
- Managed heap, 123
- managed keyword, 109
- Managed pointers, 49, 481–482
- Managed types, 229
- Match objects, 277
- MatchCollection object, 277
- .maxstack value, 109
- Mechanics, 239
- MemberInfo class, 65, 66, 67, 266, 268
- MemberInfo-derived classes, 71
- Members, 14, 482
 - accessibility attributes, 391
 - accessibility levels, 58–59
 - annotating, 67, 76–80
 - usage, 241–248
- MemberwiseClone method, 42
- Memory
 - allocation algorithm, 123–126
 - forgetting to free, 122
 - garbage collection, 123–126
 - heap-allocated, 122–123
 - managed heap, 123
 - management, 121–126
 - reclaiming, 122–123
 - reference types, 121–123
 - unreachable objects, 123
 - value types, 121–123
- MemoryStream class, 259
- Message instance member, 15
- MessageBox method, 83
- MessageBoxA method, 83
- message.txt file, 158–159, 162–163, 169
 - modifying, 170–171
 - renaming or removing, 160
- Metadata, 22, 61, 482
 - assemblies, 113–114
 - custom attributes, 76–80
 - description, 99
 - executable files, 62
 - execution engine, 63
 - extensibility, 76–83
 - file format, 95–98

508 ■ Programming in the .NET Environment

- Metadata *continued*
 - files, 7
 - header file, 61
 - identifying, 96
 - information about assemblies, 61
 - information about types, 61–62
 - inspecting, 64–75
 - issues, 3, 62–63
 - language-specific files, 62
 - locating, 96
 - managed APIs, 95
 - multi-language types, 63
 - primitive facilities, 63
 - reading and writing, 63, 75
 - saving, 63–64
 - standard attributes, 80–83
 - storing, 62–63
 - tokens, 96
 - tools and extensions, 75–83
 - unmanaged APIs, 95
 - verification, 113–114
- Metadata system, 9, 14, 61–62
 - accessing, 10
 - custom attributes, 19, 104
- Meta-programming, 92–95
- .method keyword, 108
- Method variables, 468
- MethodBase class, 66
- MethodInfo object, 66, 68, 85
- Methods, 14, 32–33, 50, 66, 238, 482
 - body, 109–110
 - calling, 109–110
 - Component Pascal, 387
 - default values, 244–245
 - .entrypoint annotation, 108
 - formatting and parsing enumeration values, 248
 - hiding methods, 108
 - instance, 32–33
 - interface types, 47
 - managed, 109
 - name of, 108
 - not returning value, 108
 - overloading, 244
 - parameter list, 109
 - private, 108
 - properties, 34
 - versus properties, 241–242
 - return value, 110
 - static, 32–33, 108
 - usage, 243–245
 - variable numbers of parameters, 245
- MFCs (Microsoft Foundation Classes), 11, 234
- Microsoft Web site, xxvii, 220, 349
- Middle type, 56
- MMC (Microsoft Management Console)
 - snap-in, 138, 168
- Mobile devices, 221, 222–223
- Modula-2, 385, 442
- module class, 459
- Module construct, 455
- Modules, 87, 152, 457, 482
 - Active Oberon, 455–456, 458
 - compiling, 459–460
 - mapping, 459–461
- Monadic I/O, 433–434
- Mondrian, 417, 418
 - access methods, 422
 - calling other CLR-hosted languages, 433–434
 - class construct, 422
 - class method, 427
 - class/subclass formulation, 423
 - combining with C# in Sieve of Eratosthenes, 435–437
 - concurrency support, 419–420
 - create construct, 434
 - dynamic function, 427
 - exception handling, 419
 - functions, 426–433
 - instance method, 427
 - invo construct, 434
 - JIT evaluation, 430–433, 436
 - minimalist language, 420
 - monadic I/O, 433–434

- monomorphic functions, 426–428
- parametric polymorphism, 419
- parametric types, 424–425
- partial applications of functions, 429–430
- pattern matching, 420, 423
- polymorphic functions, 428
- primitive types, 421
- product type, 421–422
- semantics, 418–419
- statically defined function, 427
- syntax, 420
- type casts, 425
- type system, 419
- types, 421–425
- union types, 422–423
- Mondrian, Pieter Cornelis, 420
- Monomorphic functions, 426–428
- Month enumeration, 31
- MoveNext method, 48
- Movie type, 445
- mscorcfg.msc, 168
- mscorlib assembly, 89, 110, 132, 190, 469
- mscorlib.dll assembly, 116, 121
- mscorlib.wks.dll, 116
- MSIL, 457–457
- Multi-language types, 63
- Multiple inheritance, 362
- Multiple programming languages support, 233
- Multiple-implementation inheritance, 22
- Multithreaded code, 278–279
- Mutual exclusion, 467
- mxcorsvr.dll, 116
- My Computer zone, 131
- MyControl class, 213
- MyDomain application domain, 188
- MyMethod() method, 94, 373
- Name hiding
 - hide by name, 302
 - hide by name and signature, 303–304
 - shadow by name, 302
- Name property, 66, 73
- Named constants, 247
- Named delegate types, 322
- Named permission sets, 139
- Namespaces, 5, 248–295, 457
 - Active Oberon, 464–465, 468
 - naming, 241
 - root, 248–250
 - runtime, 389–390
 - System namespace, 248–250
 - System.Collections namespace, 250, 253–254
 - System.Data namespace, 254–256
 - System.Globalization namespace, 256–257
 - System.IO namespace, 259–261
 - System.Net namespace, 261–263
 - System.Reflection namespace, 263–271
 - System.Resources namespace, 257–259
 - System.Runtime.InteropServices namespaces, 279–282
 - System.Security namespace, 272–273
 - System.Text namespace, 273–278
 - System.Threading namespace, 278–279
 - System.Web namespace, 287–289
 - System.Web.Services namespace, 289–292
 - System.Windows.Forms namespace, 283–287
 - System.Xml namespace, 292–295
 - Visual Basic .NET, 306
- NameValueCollection sections, 203
- Naming, 4
- Naming guidelines, 239–241
- Native code, 296, 482
- native int type, 29
- Native integers, 28
- NDR (Network Data Representation), 5
- Nested types, 56–57
- Nesting procedures, 469
- .NET
 - COM concepts mapped to .NET concepts, 300
 - compile-time determination of attributes, 365
 - runtime behavior, 198–199
 - unit of identity changes, 299–300

510 ■ Programming in the .NET Environment

- .NET Framework, 1–2
 - applications, 12–13
 - Base Framework, 8, 10
 - CLI (Common Language Infrastructure), 8
 - CLR (Common Language Runtime), 8, 9–10
 - CLS (Common Language Specification), 233
 - compared with IDL-based systems, 6–7
 - compile-time naming hierarchy, 457
 - component packaging model, 457
 - consumers and producers of components, 458
 - cross-language compatibility, 18
 - CTS (Common Type System), 457
 - elements, 8–10
 - enumerations, 233
 - events, 233
 - exposing, 10–11
 - factored and extensible, 235–236
 - goals, 234–238
 - infrastructure for building applications, 295
 - installing, 219–220
 - integrating with Web standards and practices, 236–237
 - interoperability, 456–458
 - language implementer tasks, 458
 - library design guidelines, 238–248
 - operator overloading, 233
 - properties, 233
 - protecting resources, 233
 - run-time deployment hierarchy, 457
 - stack engine model, 457–458
 - standard I/O functions, 231
 - structures, 233
 - unified type system, 232
 - unifying programming models, 234–235
- .NET objects
 - accessing unmanaged code, 279–282
 - Python for .NET, 359
 - using .NET objects, 359
- .NET SDK, 358
- .netmodule file extension, 162
- new() function, 400
- newobj instruction, 106
- nmake utility, 154–155, 209
- Nondefault indexed properties, 243
- Nonlocal addressing, 397–398
- Non-object-oriented languages and type systems, 22
- Nonstrict evaluation, 430
- Nonvirtual methods, 243–244
- Non-Windows platforms and CLI, 220
- Nothing permission set, 137, 139
- N.T. type, 32
- Number type, 405
- Oberon, 385, 441, 468
- Oberon Microsystems Web site, 401
- Object class, 10, 15, 39, 40–44, 52, 70, 89
- Object files, 61
- OBJECT keyword, 444
- Object references, 55, 483
- Object types, 13, 40–47, 57–58, 132, 245, 409, 425, 483
 - active, 441
 - Active Oberon, 458
 - API (application programming interface), 447
 - ByRef parameters, 310–311
 - deterministically finalized, 315
 - facets, 447
 - functionally equivalent to variant, 310
 - garbage collected, 315
 - implementing definitions, 447
 - interface types, 47
 - roles, 447
 - Visual Basic .NET, 309–312
- Object variable, assigning value to, 310
- ObjectHandle class, 191
- Object-oriented design, 232–233
- Object-oriented languages, 426, 433
 - Java, 233
 - versus semi-trusted languages, 232–233
- type hierarchy, 413

- Object-oriented prime generator, 438
- Object-oriented programming, 482
- Object-oriented VMs (virtual machines), 406
- Objects
 - accessing metadata, 41–42
 - allocating on heap, 122
 - deep copy, 42
 - destroying, 41
 - deterministic finalization, 314–316
 - deterministically finalized, 314
 - equal, 41
 - exact type, 42
 - hash code, 41
 - instance of, 84
 - lifetime, 122
 - long-lived, 124–125
 - managing, 63
 - pinned in memory, 49
 - referring to same, 42
 - returning handle to, 84
 - returning reference to, 84
 - returning type, 263
 - shallow copy, 42
 - string representing, 42
 - type object for, 41–42
 - values as, 39
- Obsolete attribute, 81
- ObsoleteAttribute attribute, 80–81
- ODL (Object Definition Language), 63
- On Error statement, 319
- On Error-style error handling, 319–321
- OnClick delegate, 338
- OnClick event, 338
- open array construct, 386
- Operand procedure, 458
- Operating systems
 - processes, 186
 - well-known directories, 149
- Operator overloading, 336–337
- Operator procedure, 458
- OrElse operator, 326
- os module, 357
- Outer type, 56
- /out:UseAboutBox.exe file, 157
- Overloaded methods, 343
- Overloading methods, 244, 363
 - Active Oberon, 468
 - PerlNET Component Builder, 378–379
 - Visual Basic .NET, 304–305
- Overloads keyword, 303, 304
- Overriding, 43, 483
- Pad method, 45
- Paint() method, 334
- Pair type, 405
- PAL (Platform Adaptation Layer), 222
- Parameter areas, 483
- ParameterInfo class, 67, 73
- Parameters
 - allocating and freeing memory, 122
 - Hungarian notation, 240
- Parametric polymorphism, 418, 419
- Parametric types, 424–425
- params arrays, 343–344
- Partial applications of functions, 429–430
- partialTimes class, 430
- Pascal, 385, 442
- Pascal casing, 240
- PATH variables, 149–150, 151
- Pattern matching, 422–423
- PE (Portable Execution) files, 479, 483
- PenSensor abstraction feature, 453
- Perl
 - annotating source code with C#-like interface declaration, 373
 - B module, 372
 - B::CC module, 372–373
 - CPAN (Comprehensive Perl Archive Network), 371
 - modules, 376
 - scope stack, 379
 - syntax, 372
 - typed variables, 374
 - untyped language, 373
- Windows Forms application, 380–382
- XS code, 376

512 ■ Programming in the .NET Environment

- Perl for .NET Research compiler
 - code generator, 372–374
 - compatibility with existing extensions, 376
 - execution speed, 375
 - full language support, 375
 - interface specification, 373
 - parser, 372
 - problems, 375–376
 - run-time library, 374
 - status, 374
 - typed variables, 374
- Perl Reference Manual, 330
- PerlNET, 378–380
- PerlNET Component Builder, 376–379
- Permission class, 483
- Permission demand, 130
- Permission grant, 130
- Permission objects, 144, 483
- Permission sets, 133
 - based on evidence, 134–141
 - code groups, 135
 - named, 139
- Permissions
 - assemblies, 130–131
 - asserting, 141–142
 - collection of, 130–131
 - denying, 142
 - security check, 141
- Permissions-based security system, 272–273
- PermissionSet, 141
- PermitOnly, 141
- Permit-only restriction, 483–484
- PInvoke (Platform Invoke), 279–281, 378, 484
- Pizza, 419
- Platform invocation services, 484
- Plug-and-play components, 236
- POA (Portable Object Adapter), 5, 105
- Pocket PCs, 220
- Point class, 33, 52
- Point value type, 34, 35, 50, 53
- Pointer types, 13, 25, 49–50, 484
 - reference surrogates, 393–394
- Pointer-based object types, 444
- Pointers
 - C#, 339–340
 - COM Interop, 340
 - Component Pascal, 386
 - to user-defined types, 25
- Policy assemblies, 139
- Policy levels
 - code groups, 135–138
 - named permission sets, 139
 - policy assemblies, 139
- Policy Manager
 - code groups, 135–138
 - policy levels, 134–135
- Polymorphic declarations, 454
- Polymorphic functions, 428
- Polymorphism
 - Active Oberon, 442
 - Component Pascal, 386
- Pop instructions, 106
- Port type, 405
- POST event, 216
- PowerBuilder, 234
- pred function, 436
- Prime numbers, 434–438
- primes function, 437
- Primitive types, 25, 246, 421
- Principal objects, 127
- PrincipalIdentity object, 127
- PrincipalPermissionAttribute attribute, 128–129
- PrintAssemblies function, 189
- PrintItems method, 253
- PrintPrimes method, 437
- PrintValue method, 299
- Private accessibility level
- Private assemblies, 163–172, 217
- Private assembly, 208
- Private classes, 390
- private keyword, 108
- Private methods, 108
- Privileges, 484

- Procedure type, 405
- Processes
 - application domains, 119–121
 - versus application domains, 186–187
 - isolation, 186
- Profiles, 221–222, 484
- Profiling, 484
- Program IDs, 151–152
- Programmer's Introduction to C#, A, 349
- Programming, 3–6
- Programming in the large, 4–5
- Programming in the small, 2–4
- Programming issues, 2–5
- Programming languages
 - combining to solve problem, 434–438
 - consumer perspective, 463
 - importing types, 22–23
 - interaction with CLR, 22–23
 - passing types between, 3
 - producer perspective, 463
 - producing verifiable code, 384
 - support for multiple, 13
 - type systems, 20–23
- Programming language-specific type systems, 21
- Programs
 - controlled execution, 103
 - debugging, 116–119
 - execution environments, 113
 - forgetting to free memory, 122
 - performing multiple tasks at same time, 278–279
- Properties, 14, 22, 50, 66, 68, 233, 238, 484–485
 - C#, 333–335
 - definition of, 35
 - hints to JIT (just-in-time) compiler, 36
 - indexed, 242–243
 - interface types, 47
 - logical backing store, 241
 - methods, 34
 - versus methods, 241–242
 - returning arrays, 242
- Property design pattern, 334
- PropertyInfo class, 66
- Proxy objects, 7, 84, 92–93, 99
- Public accessibility level, 58, 391
- Public assemblies, 163–172
- Public classes, 390
- public key token, 88, 211
- PUBLIC modifier, 456
- Public static constants, 247
- Publisher certificates, 131, 132
- Push instructions, 106
- Puzzle module, 465
- PuzzleForm type, 465
- .py files, 358
- PyObject structure, 357, 367
- Python
 - code not verifiable, 366
 - dynamic features, 356
 - extensibility, 352
 - extension modules, 358, 361
 - HelloWorldPY class, 15
 - implementations, 352–353
 - library, 357
 - modules, 357
 - .NET tools not supported, 366
 - overview, 351–353
 - standard library, 361, 362
 - type declarations, 362, 363
 - type interference, 363
 - unable to inherit from objects, 366
- Python for .NET, 353
 - alternative implementation strategy, 365–368
 - API for use by compiler, 357
 - architecture, 355–358
 - class and instance semantics, 362
 - closed world syndrome, 361–362
 - compiler, 355–356, 358–361, 366–367
 - contractual obligations, 354
 - current status, 354–355
 - different intermediate language, 366–367

514 ■ Programming in the .NET Environment

- Python for .NET *continued*
 - dynamic language support, 364–365
 - full integration with .NET, 354
 - Hello World program, 358–359
 - help option, 358
 - leveraging of existing Python code, 362
 - library, 357–358, 368
 - limitations, 361–364
 - method overloading, 363
 - method signatures and overloads, 359–360
 - .NET interface, 356
 - no support for some features of .NET, 354
 - no support for some Python features, 355
 - performance, 361
 - runtime, 356–357, 367–368
 - speed penalty, 355–356
 - speed problem, 354
 - type declarations for semantics, 363–364
 - unmanaged emit API, 367
 - usage, 358–361
- Python for Win32 extensions, 358
- Python Web site, 351, 358
- Python2C project, 355

- Queue class, 250, 359

- /R:AboutBox.dll file, 156
- RAD (rapid application development) paradigm, 234
- Reachable objects and garbage collector, 123–124
- ReadCustomData2.cs file, 203–204
- ReadCustomData2.exe.config file, 203
- ReadCustomData.cs file, 202–203
- ReadCustomData.exe file, 201
- ReadCustomData.exe.config file, 202
- Reader class, 260
- REAL type, 387
- Record types, 468
- Records, 386
- Recordsets, 255
- RefAny type, 311
- Reference classes, 391, 395
- Reference surrogates, 391–394
- Reference types, 13, 23, 485
 - accessing, 40
 - allocated on garbage collected heap, 25
 - allocating and freeing memory, 121–123
 - allocating on garbage-collected heap, 40
 - assignment compatibility, 54–56
 - C#, 332–333, 333
 - directly inherit from, 24
 - interface types, 47–49
 - lifetimes, 122
 - object types, 40–47
 - pointer types, 49–50
 - prohibiting subtyping, 39
 - sealed, 39
 - SmallTalk, 231–232
 - String type, 44–47
 - strongly typed references, 25
- ReferenceEquals method, 42
- References, 96, 97
- REFINES relation, 441, 446, 458
- Reflection, 64–75, 485
 - class hierarchy, 93
 - classes, 65–69
 - invocation support, 266–269
 - late binding, 317–318
 - late-bound helper, 318
 - late-bound nature, 264
 - PerlNET, 379
 - usage, 69–74
- Reflection APIs browsing support, 263–266
- Reflection elements, reading custom attributes, 249
- Reflection.Emit, 269–271, 276
- Reflection.Emit, 92, 223
- Reflection/Emit API, 373
- Reflection::Emit library, 355–356, 366–367
- regedit, 196
- Regex object, 277
- RegionInfo class, 256
- @Register directive, 215
- Registry, 63, 105, 151–152, 196–197
- Registry Assembly, 99

- Regular expression engine, 269, 275
- Regular expressions, 276, 277
- Release method, 314, 315
- Remote assemblies
 - public key token, 211
 - security settings, 218
- Remote objects, managing, 199
- RemoteAssembly assembly, 210, 213
- RemoteAssembly.cs file, 208–209
- RemoteAssembly.dll file, 209
- RemoteAssemblyfile, 209, 210
- RemoteAssembly.netmodule file, 209
- RemoteHelloWorld, 213
- RemoteHelloWorld.cs source code, 206
- RemoteHelloWorld.exe file, 206
- Remoting services, 223
- Remove method, 253
- remove_EventName method, 37
- remove_OnClick() function, 338–339
- Replace method, 278
- require, 375
- requiredRuntime, 198
- Reset method, 48
- Resource constrained, 222
- Resource fallback process, 179–180
- Resource Generator, 182, 184
- Resource manager, 258–259
- ResourceManager class, 179, 182, 258
- Resources, 152, 257
 - accessing and controlling, 258
 - authentication, 199
 - culture-neutral, 180, 185
 - defining, 182
 - embedded, 157–162
 - linked, 157–162
 - usage, 199
- ResourceSet class, 179
- Resume Next statement, 321
- Resume statement, 321
- ReturnType property, 66
- Reusable components, 465
- Reusing components, 31
- RevertAll, 142
- RevertPermitOnly, 142
- RIL (Runtime Infrastructure Library), 221
- Role-based security, 126–130
- Roles, 126–128
- Root namespaces, 248–250
- RTTI (run-time type information), 3
- Run method, 278
- runat=server directives, 215
- run() method, 466
- Runnable interface, 450
- Runtime
 - classes, 390
 - creation of types, 92–95
 - namespaces, 389–390
- Runtime checks, 428–429
- runtime element, 198–199
- Runtime environment
 - Python language semantics faithfully implemented, 356–357
 - reference types unioned with other types, 309
 - version, 116, 198
 - versioning policy, 172–173
- Runtime libraries, historical perspective, 231–233
- runtime subelement, 198
- Runtime systems, 413
- Runtime type libraries, 229–230
- Runtime type system as single-root type system, 309
- Safearrays, 306–307
- Sample value type, 33
- SampleIL.EntryPoint class, 110
- SampleIL.EntryPoint type, 112
- SampleIL.exe application domain, 121
- SampleIL.exe assembly, 121
- Satellite assemblies, 179–180
 - defining, 183–185
- Satellite data files, 257
- Saving metadata, 63–64
- SayHello method, 15, 16, 17
- SayHelloClass method, 33

516 ■ Programming in the .NET Environment

- SayHelloInstance method, 33
- SByte class, 251
- Scalability, 5
- Scheme
 - call/cc (call-with-current-continuation), 405
 - closures, 407–411, 413
 - CLR and JVM runtimes, 406
 - continuations, 405
 - dynamic type checking, 412–413
 - essential primitives, 404–405
 - first-class continuations, 413–414
 - global names, 407
 - latently typed language, 405
 - limitations, 413–414
 - number type, 405
 - objects wrapped in special class, 415
 - pair type, 405
 - port type, 405
 - primitive value types, 405
 - procedure type, 405
 - run-time systems, 413
 - run-time tricks, 409
 - simple model for computation, 404
 - small values for pointers, 411–412
 - special optimizations, 409
 - symbol type, 405
 - type checking, 405
 - variable argument lists, 409–410
 - vector type, 405
- Schwartz, Randal, 372
- Screen scrapping, 289
- SDE (Smart Device Extensions), 223–224
- Sealed reference types, 39
- Sealed types, 44
- Sealed value types, 39
- Search paths, 149–150
- sectionGroup element, 201
- Security, 4, 19
 - code-based, 143–144
 - evidence-based, 130–133
 - Framework Class Library, 229–230
 - overriding, 483
 - permissions-based system, 272–273
 - requirements, 85
 - role-based, 126–130
 - strong names, 164–165, 170–172
- Security configuration files, 200
- Security constraints, 144–145
- Security hole, 485
- Security manager, 485
- Security model for Windows, 128–129
- Security objects, 485
- Security policies, 485–486
- Security systems, 486
 - Internet Explorer, 207–208
 - passing evidence to, 132–133
 - stack walks, 141–144
 - XML (eXtended Markup Language), 133
- security.config file, 200
- SecurityException exception, 273
- SecurityPermission.Assertion permission, 474
- Select statement, 321
- Semi-trusted code, 229
- Sensor abstraction, 448, 453–454
- Server-side HTML (Hypertext Markup Language) generation, 235
- Set assignment, 316–317
- Set functions, 334
- set methods, 34, 35, 52
- SetCaption() function, 334
- set_X method, 36
- Shadow by name, 302
- Shadows keyword, 302, 304
- Shallow copy, 42
- Shared resources, 442–443
- SHORTREAL type, 387
- side-by-side component sharing, 150
- Sieve of Eratosthenes, 434
 - C#, 435
 - combining Mondrian and C#, 435–437
- Signed assemblies, 165–166, 170
- Signed integers, 9, 27, 28

- Simonyi, Charles, 240
- Simple types, 25
- sin() function, 372
- Single class, 252
- Single type system for multiple languages, 21–22
- SingleTabSectionHandler class, 202
- Sites, 131
- SkipVerification permission set, 139
- SmallTalk
 - common base type, 231
 - library, 231–232
 - object-oriented design, 232–233
 - reference types, 231–232
 - single-implementation inheritance model, 22
 - type system, 21
 - unified type system, 231
- Smart clients, 11
- sn.exe, 168, 172
- SOAP, 236, 237, 290
- SOAP-based Web services, 290–292
- Sort method, 49
- source interface, 37
- SSCLI (Shared Source Common Language Infrastructure), 220, 224–226
- STA (Single Threaded Apartment) model, 104–105
- Stack
 - activation record, 141
 - importance of size, 111
 - maximum depth, 109
 - rules for use, 111
 - underflow / overflow, 109
- Stack class, 250, 253
- stack engine model, 457–458
- Stack frames, 486
- Stack walks, 141–144, 486
- Stacktrace property, 379
- Standard attributes, 80–83
- standard C++, 15
- Standard execution environment, 105
- startup element, 198
- Stateless programming model, 237
- Static fields, 33, 68, 69, 486
- static keyword, 108
- Static methods, 32–33, 108, 486
- Static module construct, 441
- Static modules, 455–456
- Static procedures, 391
- Static record types, 394–395
- Static variables, 391
- Statically defined function, 427
- Stein, Greg, 353, 355
- STL (Standard Template Library), 21
- Store instructions, 109, 486
- Stream classes, 260
- Streams, 260–261
- String[] args keyword, 109
- String class, 10, 39, 42, 43, 44–47, 74, 89, 90
- string interning, 342–343
- String literals, 177
- String objects, 48, 110, 273
- String pattern, replacing with string, 275
- String reference, 55
- String replacement functions, 277–278
- String resource, 177
- String type, 40, 43, 57–58, 132
- String.Append() method, 244
- StringBuilder class, 44
- String.IndexOf() method, 244
- String.IsInterned() function, 343
- Strings, 10, 44–47
 - adding to strings, 46
 - comparing, 44
 - comparing to pattern, 275–276
 - concatenating, 45
 - constructors, 44
 - converting to uppercase or lowercase, 45
 - fully qualified names, 263
 - immutable, 44
 - inserting strings in, 45
 - length, 45

518 ■ Programming in the .NET Environment

- Strings *continued*
 - location of character or string within, 45
 - padding characters inserted or removed, 45
 - representing objects, 42
 - retrieving portions of formatted, 275
 - temporary, 44, 46–47
- Strings class, 253
- strings.en.AU.resources module, 184
- strings.resources file, 183
- strings.txt file, 184
- Strong names, 88, 131–132, 164–165, 167–168, 208
- Strongly typed collections, 253
- struct keyword, 31, 33, 333
- structs, 394
- Structural compatibility of delegates, 398
- Structured exception handling, 319–321
- Structures, 21, 233
 - events, 36–38
 - fields, 33
 - methods, 32–33
 - Visual Basic .NET, 312
- Subclasses
 - exact signatures, 360
 - invocation off, 268–269
- Subprocesses, 119
- Subsystems, 236
- Subtypes, 236
- Subtyping, 234, 423
- Swap-stack-top instruction, 458
- switch expression, 423
- Symbol type, 405
- Symbolic links, 150, 151
- Symbolic names, 150
- SyncRoot method, 49
- Synthetic static class, 390, 391
- syslogd process, 196
- System namespace, 248–252, 469
- System32 directory, 149
- system32 directory, 164
- System.Attribute base class, 77, 80, 249, 337
- System.Collections namespace, 250, 253–254, 359
- System.Collections.Specialized.Name.Value Collection, 203
- System.Configuration class library, 197
- System.Configuration.ConfigurationSettings class, 202
- System.Configuration.NameValueSection-Handler class, 201, 203
- System.Configuration.SingleTagSection-Handler class, 201
- System.Console::WriteLine method, 110
- System.Data namespace, 254–256
- System.DateTime type, 29
- System.Decimal type, 29
- system.diagnostics element, 199
- System.Drawing namespace, 230
- System.EnterpriseServices namespace, 280
- System.Enum base class, 24, 248
- System.Exception base class, 248
- System.Globalization namespace, 256–257
- System.Guid type, 29
- System.Int32 type, 55
- System.IO namespace, 259–261
- System.IO.StreamReader object, 158
- system.net element, 199
- System.Net namespace, 261–263
- System.Object base class, 40–44, 64, 69, 117, 248
- System.Random, 434
- System.Reflection namespace, 263–271
- System.Reflection.Assembly.GetManifestResourceStream() function, 158
- System.Resources namespace, 257–259
- System.Runtime.InteropServices namespaces, 279–282
- System.Runtime.InteropServices.GCHandle type, 378
- system.runtime.remoting, 199
- System.Security namespace, 272–273
- System.Security.PermissionSet class, 130–131
- System.Stream base class, 259
- System.String object, 378

- System.String type, 110, 421
- System.Text namespace, 273–278
- System.Text.RegularExpressions namespace, 275
- System.Threading namespace, 278–279
- System.TimeSpan type, 29
- System.Type class, 263–266, 312
- System.ValueType base class, 24, 248
- system.web element, 199
- System.Web namespace, 201, 230, 287–289
- System.Web.Configuration.HttpHandlers-SectionHandler, 201
- System.Web.Services namespace, 12, 289–292
- System.Web.Services.WebServiceAttribute attribute, 290
- System.Web.UI.Control class, 213
- System.Web.UI.Page class, 12
- System.Windows.Forms namespace, 11–12, 230, 283–287
- System.Windows.Forms.Form base class, 11, 380
- System.Windows.Forms.Form, 465
- System.Xml namespace, 283, 292–295

- Tables, 95–96
- TCB (trusted computing base), 486
- Temporary strings, 44, 46–47
- Text-based configuration files, 196–197
- TextReader class, 260
- TextWriter class, 260
- this object, 396
- this pointer, 486
- Thread class, 450
- Thread pool implementation, 237
- Threads, 278–279
 - application domains, 120–121
 - Principal objects, 127
- ThreadStartDelegate method, 278
- THROWEXCEPTION procedure, 468
- Thunks, 429
- Time, 29, 313
- Time spans, 29
- time() function, 372
- TINAC, 63
- /t:library option, 156
- Tokens, 96–97
- ToLower method, 46
- ToString method, 42, 43, 79, 191
- ToUpper method, 47
- Trim Strings method, 45
- Trojan Horse components, 148
- Trusted Sites zone, 132
- Try statement, 319
- try-finally block, 340
- /t:winexe option, 156
- Type checks, 414–415
- Type class, 10, 41–42, 65–70, 74–75
- Type coercion, 409
- Type declarations, 364
- Type erasure, 398, 412
- Type fields, 14
- Type interference, 363
- Type keyword, 312
- Type libraries, 63, 99
- Type Library Exporter, 99, 282
- Type Library Importer, 99, 101, 281, 282
- Type loader, 487
- Type object, 43, 67, 69, 90, 271
- Type safety, 113, 114, 487
- Type systems, 9, 13–14, 487
 - C#, 332–333
 - Component Pascal, 386–388
 - elements, 23–25
 - events, 22
 - evolution, 20
 - incompatibility, 22
 - issues, 3
 - Mondrian, 419
 - for multiple languages, 21–22
 - non-object-oriented languages, 22
 - programming languages, 20–23
 - programming language-specific, 21
 - programming styles and languages, 19

520 ■ Programming in the .NET Environment

- Type systems *continued*
 - properties, 22
 - reference types, 23
 - union of, 22
 - value types, 23
 - Visual Basic.NET additions, 298–306
- Type type, 64–65
- Type-bound procedures, 396
- TypeConstructorName type, 66
- Typed references, 487
- Typed variables, 374
- Type.GetType() static method, 264
- Type.InvokeMember function, 267
- Type.InvokeMember() method, 266
- typeof() operator, 263–264
- TYPEOF() primitive, 395
- Types, 13, 14, 487
 - annotating, 76–80
 - base class, 248
 - ComVisible() attribute, 282
 - constructors, 66
 - contracts, 114
 - describing, 14
 - as documentation, 20
 - dynamic, 92–95
 - dynamic checking, 20
 - dynamic discovery of, 84–86
 - dynamically emitting at runtime, 269–271
 - events, 36–38, 66
 - fields, 33, 66
 - fully qualified string name, 264
 - grouping related, 87
 - importing, 22–23
 - information about, 10, 61–62
 - instances, 25
 - logical fields, 34
 - Mondrian, 421–425
 - nested, 56–57
 - printing members of, 264–266
 - properties, 66
 - runtime creation of, 92–95
 - sealed, 44
 - subtyping from, 7
 - uniquely identifying, 32
 - usage, 245–248
 - user-defined, 29
 - visibility, 57
- Type-safe, 111
- Type-safe CIL, 115
- Type-safe code, 44
- UInt16 class, 26, 251
- UInt32 class, 26, 251
- UInt64 class, 26, 251
- UIntPtr class, 26, 252
- Unnamed delegate types, 322
- unbox instruction, 106
- Unboxing, 487
- Unboxing value types, 39
- Unicode, 176–177, 273–278
 - converting string to byte array, 273–275
 - strings, 10
- Unifying programming models, 234–235
- Union, 109
- Union construct, 386
- Union method, 130
- Union types, 422–423
- UNIX
 - bin directory, 149
 - make tool, 154
 - text-based configuration files, 196–197
 - well-known directories, 149
- Unmanaged APIs, 95
- Unmanaged code, 14, 279, 488
- Unmanaged function pointers, 49
- Unmanaged pointers, 49
- Unreachable objects, 123
- Unsafe code, 339–341
- Unsigned integers, 9, 27, 28
- Untrusted Sites zone, 132
- Updating Registry, 152
- URLs, 131
- use statement, 372
- UseAboutBox.exe file, 153, 154–155, 170, 176

- UseAboutBox.exe.config file, 175, 176
- UseCulture.cs file, 181, 183
- UseCulture.exe assembly, 183, 184
- UseCulture.resources.dll assembly, 184
- User interfaces, separation of code from, 177–178
- User Policy configuration files, 200
- User policy level, 134
- user32.dll, 83
- User-defined
 - attribute class, 80
 - enumerations, 30–31
 - methods, 47–48
 - object types example, 50–53
 - objects exposed to COM, 282
- User-defined types, 29, 312
 - C#, 333
 - pointers to, 25
- User-defined value types
 - boxed types, 39
 - enumerations, 30–31
 - structures, 31–38
- UseRemoteAssembly.cs file, 210
- UseRemoteAssembly.exe.config file, 211
- USES clause, 442
- using directives, 32, 77
- using System directive, 30, 35
- UTF-7 character set, 273–278
- UTF-8 character set, 273–278
- UTF-16 character set, 27

- Value classes, 391–392, 394
- Value types, 13, 23, 24, 245, 488
 - Active Oberon, 468
 - allocated on garbage-collected heap, 25
 - allocated on stack, 25, 36, 121–122
 - allocating and freeing memory, 121–123
 - base class, 248
 - boolean values, 27
 - boxed form, 309–310
 - boxing, 39, 332–333
 - built-in, 25–28
 - C#, 332–333
 - characters, 27
 - default constructor, 32
 - directly inheriting from, 24
 - fields, 34–36
 - floating-point types, 28–29
 - inheritance, 39
 - integers, 27–28
 - properties, 34–36
 - sealed, 39
 - signed integers, 27
 - unboxed form, 309
 - unboxed value types, 309
 - unboxing, 39
 - unsigned integers, 27
 - usage, 246
 - use of interfaces, 53–54
 - user-defined, 29–38
- Values, 13, 25, 488
 - as classes, 39
 - specifying location, 49–50
- Variable argument lists, 409–410
- Variables
 - Hungarian notation, 240
 - read-only export, 391
- Variant data type, 309–312
- Variant variables, assigning value to, 310
- Variants, 309
- Vector type, 405
- Verifiable CIL, 115
- Verifiable code, 44
- Verification, 488
- Versioning, 5
- Versioning policy, 172–173
- Versioning-related technologies, 150–151
- Virtual machines, 105
- Virtual methods, 243–244, 248, 488
- Viruses, 113
- Visibility, 57, 488
- Visual Basic, 234
 - classes, 298–299
 - COM (Common Object Model), 297
 - COM coclass, 298
 - default interfaces, 298

522 ■ Programming in the .NET Environment

- Visual Basic *continued*
 - excluding inheritance, 301
 - HelloWorld VB class, 16
 - language extensions, 2
 - namespaces, 306
 - Obsolete method, 82
 - optional parameters, 304
 - reference assignment, 316–317
 - time, 313
 - typeless language, 309
 - user-defined types, 312
 - value assignment, 316–317
 - Win32 APIs, 297
- Visual Basic .NET, 297–298
 - AndAlso operator, 326
 - array types, 308
 - array types with fixed lower bounds of zero, 308
 - arrays, 306–309
 - classes, 298–301
 - compound operators, 326
 - Def statements, 325
 - delegates, 321–324
 - deterministic finalization, 314–316
 - End While, 325
 - On Error-style error handling, 319–321
 - events, 321–324
 - future, 327
 - garbage collection, 314–316
 - Gosub, 325
 - inheritance, 301–304
 - initializers, 326
 - Integer type, 325
 - keywords, 301
 - language cleanup, 324–326
 - late binding, 317–319
 - Long type, 325
 - namespaces, 306
 - new features, 326
 - nonzero-lower-bound arrays, 308
 - Object types, 309–312
 - OrElse operator, 326
 - overloading methods, 304–305
 - overloading resolution rules, 305
 - parameterized property, 336
 - platform changes, 313–324
 - programming model changes, 324
 - strongly typed language, 305
 - structured exception handling, 319–321
 - structures, 312
 - type system additions, 298–306
 - type system modifications, 306–313
 - typeless ability, 305
 - user-defined value types, 312
 - variant data type, 309–312
 - Wend, 325
- Visual Perl plug-in, 380
- Visual Studio, 223–224
- VMs (virtual machines), 6, 403, 406
- void keyword, 108
- W3C DOM Level 1 Core, 293
- W3C XPath 1.0 data model, 293
- W3C XSD specification, 293
- W3C XSLT 1.0-compliant XSLT processor, 293
- Wall, Larry, 371
- Web application services, 287
- Web content, downloading, 205–216
- Web controls, 213–216
- Web Forms, 230, 287
- Web pages, displaying Web controls, 215
- Web servers, uploading application to, 206
- Web services, 12, 230, 290–292
- Web standards and practices, 236–237
- Web-based applications, 12
- web.config file, 200
- WebRequest class, 261
- WebService class, 12
- Well-known directories, 149
- Well-known locations, 149
- while loop, 48
- Win32 APIs, 280, 297
- Win32API class, 280
- %windir%/assemblies directory, 167
- Windows

- clients, 11–12
- COM, 104–105
- localization, 178
- Registry, 151–152
- security model, 128–129
- System32 directory, 149
- Windows 98 SE, 150
- Windows 2000, 150
- Windows applications
 - developing, 283–287
 - traditional model for building, 147–148
- Windows Explorer, 167
- Windows Forms, 226, 230, 238, 283–287
- Windows Forms library, 223
- Windows handles, 236
- Windows Installer, 219
- Windows XP, 220, 224–226
- Windows-based platforms and PAL
(Platform Adaptation Layer), 222
- Wise, 219
- WithEvents modifier, 322
- Word choice, 239
- Wrapper classes, 438
- WriteLine method, 52, 54, 271
- Writer class, 260
- WSDL tool, 292

- X Windows, 150
- xcopy, 205
- XDR (eXternal Data Representation), 5
- XDR schema validation, 293
- XHR (explicit heap-allocated records),
397–398
- XML (eXtended Markup Language), 223,
236, 237
 - built-in support, 255–256
 - C# and comments, 344
 - data support services, 229
 - forward-only way of generating, 293
 - processing functionality, 292–295
 - reading data, 293
 - security system, 133
 - transforming documents, 293
 - viewing and manipulating, 293
- XML namespace, 293
- XML serialization, 237
- XmlDataDocument class, 293
- XmlDocument class, 293
- XmlNodeReader class, 293
- XmlReader, 293
- XmlSchema object model classes, 293
- XmlSchemaCollection class, 293
- XmlTextReader class, 293
- XmlTextWriter class, 293
- XmlValidatingReader class, 293
- XPathDocument class, 293
- XPathNavigator class, 293
- XS code, 376
- XSD schema validation, 293
- XSLT, 293
- XsltTransform class, 293

- Zones, 131