



Index

A

Abbreviations

FxCop rules for, 287

in identifier names, 43–44

Abrams, Brad

abstract class design, 84, 85

acronym capitalization conventions,
37–38

acronyms in identifiers, 44

addition through subtraction
principle, 22

API specification, 312

arrays, 209

assemblies, namespaces, 49

Basic Dispose Pattern

implementation, 253

capitalization of compound words/
common terms, 40

choice between enum, Boolean
parameters, 152

choice between properties, methods,
117, 120

enum design, 91, 95, 96

enumeration names, 60

Enum.IsDefined, 153–154

event design, 134

exception error message design, 190

exception handling, 192

exception throwing, 187

exceptions, 181

extensibility, 176

factories, 262

framework design, 9

member overloading, 107, 109

namespace names, 50–51

naming conventions, 34, 35

pointer parameters, 162

protected members, 166

type functionality, 68

type names, 55

types and namespaces, 71, 72

usability studies, 18

Abstract class

contract separation with, 81–82

design guidelines, 83–85

extensibility and, 67

FxCop rules for, 293

Abstractions

base classes for implementation of,
173

choice between class, interface, 77–83

extensibility with, 170–172

layered architecture and, 29–30

limiting for self-documenting object
models, 28–29

Optional Feature Pattern and, 264–
267

AccessViolationException, 199–200

Acronyms

capitalization conventions for, 35,
36–38

capitalization rules for, 284–285

- Acronyms (*continued*)
 - FxCop rules for, 287
 - in identifier names, 43–44
 - Active Server Pages (ASP), 10
 - Addition through subtraction, 22
 - Aggregate components, 235–243
 - component-oriented design, 237–239
 - description of, 235–237
 - design guidelines, 240–243
 - factored types and, 240
 - Alcazar, Mark, 229
 - Alias names, 44–46
 - Allocation, 74–75
 - Anderson, Chris
 - choice between class, interface, 82
 - custom event handler design, 138
 - developer methodologies, 17
 - field design, 140
 - low barrier to entry principle, 20, 21
 - APIs. *See* Application programming interfaces
 - Application development, 1–3
 - Application model namespace, 52
 - Application programming interfaces (APIs)
 - application development, 1, 2
 - design specification, 14–15
 - exceptions for consistency, 179
 - layered architecture principle, 29–31
 - low barrier to entry principle, 19–23
 - naming new versions of existing, 46–48
 - sample API specification, 311–317
 - scenario-driven design principle, 13–19
 - self-documenting object models principle, 23–29
 - well-designed framework, 3
 - Applied Microsoft.NET Framework Programming* (Richter), 43, 44–45
 - Architecture, layered architecture principle, 29–31
 - ARGB value, 38
 - Argument checking, 161
 - Argument exceptions, 304
 - ArgumentException, 198–199
 - ArgumentNullException, 198–199
 - ArgumentOutOfRangeException, 198–199
 - Arguments
 - parameter passing, 155–157
 - validating, 152–155
 - ArrayList, 212
 - Arrays
 - choice between collections and, 218–219
 - choice between properties, methods, 119–120
 - design usage guidelines, 207–209
 - FxCop rules for, 305
 - groupings of types, 67–68
 - members with variable number of parameters, 157–161
 - ASP (Active Server Pages), 10
 - ASP.NET, 30, 31
 - Assembly
 - FxCop rules for, 288
 - naming conventions for, 48–49
 - Async callback, 244
 - Async Pattern, 243–248
 - asynchronous operations, 243–244
 - design guidelines, 244–246
 - elements of, 244
 - implementation example, 247–248
 - Async Result object, 244
 - Asynchronous operations
 - Async Pattern, 244–248
 - function of, 243–244
 - Attributes
 - FxCop rules for, 305–306
 - groupings of types, 67–68
 - System.Attribute, 209–211
 - AttributeUsageAttribute, 210
 - Avalon, 20, 21
- B**
- Base class
 - choice between class, interface, 81
 - in derived class name, 55–56
 - extensibility and, 67, 172–174

- virtual members documentation and, 169
- Base suffix, 174
- Base types, 301
- Basic Dispose Pattern
 - finalizable types and, 256–260
 - guidelines for, 250–251
 - implementation guidelines, 251–256
- Begin method
 - Async Pattern design, 245–246
 - Async Pattern implementation, 248
 - function of, 244
- BinaryReader, 18
- Bitwise operations, 97
- Boolean parameters
 - enum parameters and, 150–152
 - member overloading, 107
- Boolean properties
 - naming of, 62
 - for Optional Feature Pattern, 267
- Box, 75
- Boxing
 - explicit interface member
 - implementation and, 111–112
 - from Object.Equals method, 222
- Brace usage conventions, 274–275
- Brumme, Christopher
 - Async Pattern implementation, 248
 - exception from constructor, 127
 - exception handling, 193, 194
 - fail fast, 185–186
 - hash code, 226
 - standard exception types, 199–201
 - type constructor design, 131, 132
 - unhandled exception handler, 182–183
 - virtual calls, 130
- By-value parameter, 155
- C**
- C++
 - enumeration names in, 60
 - finalizer in, 252
 - static class in, 85
 - subclassing, 10
- varargs methods, 160–161
- virtual calls in, 130
- C#
 - constructor design and, 128
 - finally blocks of, 194
 - indexed property design and, 123–124
 - indexer name in, 123
 - operator overloads and, 143
 - static class in, 85
 - types, namespaces and, 72
- C# coding style conventions, 273–279
 - comments, 277–278
 - file organization, 278–279
 - general style, 274–276
 - goals of, 273
 - naming, 277
- Callbacks
 - event design guidelines, 132–138
 - extensibility with, 166–168
 - virtual members *vs.*, 168
- CamelCasing convention
 - acronym capitalization conventions, 37
 - identifier naming conventions, 34, 35–36
 - for parameter names, 64
- Capitalization conventions, 34–41
 - acronyms, 36–38
 - case sensitivity, 41
 - compound words, common terms, 39–40
 - FxCop rules for, 284–286
 - identifiers, 34–36
- Case insensitivity
 - attribute and, 211
 - support of, 41
- Case sensitivity. *See also* Capitalization conventions
 - FxCop rules for, 286
 - guidelines for, 41
- Catch block
 - for cleanup code, 193–194
 - exception handling, 191
- Catch clauses, 193–194

- Certainty value, 283
- Change events, 124–125
- Clark, Jason, 185
- Clarke, Steven
 - Boolean parameters, 151
 - enum design, 91–92
 - exception wrapping, 196
 - explicit interface member implementation, 112
 - method names, 61
 - return codes, 180
- Classes
 - abstract class design, 83–85
 - base classes, extensibility with, 172–174
 - choice between interfaces and, 77–83
 - choice between struct and, 74–76
 - function of, 67
 - FxCop rules for choice between interface and, 293–294
 - FxCop rules for names of, 288–290
 - groupings of types, 68
 - naming conventions for, 54–60
 - sealing, 175–176
 - static class design, 85–86
 - unsealed classes, 164–165
- Cleanup code, 193–194
- Cloning, 221
- Closed-form compound word, 39
- CloseHandle, 181
- CLR. *See* Common Language Runtime
- CLS (Common Language Specification), 43
- Coding styles, 14. *See also* FxCop
- Collections
 - arrays and, 207, 208
 - choice between arrays and, 218–219
 - custom, 219–220
 - design usage guidelines, 211–213
 - FxCop rules for, 306–307
 - groupings of types, 67–68
 - as parameters, 213–214
 - properties, return values, 214–217
 - snapshot *vs.* live, 217–218
- Collection<T>
 - base suffix and, 174
 - for properties, return values, 214
 - implementation of base class, 173
 - return subclass of, 216
- ComException, 201
- Comments, 277–278
- Common Language Runtime (CLR)
 - assemblies, DLLs and, 48–49
 - capitalization conventions, 34
 - case sensitivity, 41
 - default arguments and, 110
 - exception wrapping, 195
 - interface design and, 86
 - language-specific names, 44–46
 - memory management, 248–249
 - standard exception types and, 199–201
 - type constructor design and, 131
 - types in, 67
- Common Language Specification (CLS), 43
- Common terms, 39–40
- Company name, 50
- Comparison operators, 223
- Component-oriented design
 - aggregate components and, 238–239
 - aggregate components for, 235–243
 - Create-Set-Call usage pattern, 237–238
- Compound words
 - capitalization conventions for, 39–40
 - capitalization rules for, 285
- Concrete type
 - abstract class design, 84–85
 - extensibility with abstractions, 171
- Consistency
 - exceptions and, 179
 - of naming conventions, 33
 - of self-documenting object models, 27–28
 - of well-designed framework, 6
- Constant fields, 140
- Constructor parameters
 - for attribute properties, 210–211
 - Booleans for, 152

- Constructors
 - abstract class design, 83–84
 - for aggregate components, 239, 241–242
 - on exceptions, 202
 - factories *vs.*, 261–262
 - struct design, 89–90
 - Constructors, design guidelines
 - conversion operators, 146–147
 - event design, 132–138
 - event handler design, custom, 138
 - field design, 139–141
 - FxCop rules for, 297–298
 - general guidelines, 125–130
 - operator overloads, 141–146
 - overloading operator `==`, 146
 - parameter design, 148–150
 - parameter passing, 155–157
 - parameters, choice between enum/
Boolean, 150–152
 - parameters, members with variable
number of, 157–161
 - pointer parameters, 161–162
 - type constructor design, 131–132
 - validating arguments, 152–155
 - Contract, 80, 81–82
 - Convenience overloads, 21, 23
 - Conversion operators, 146–147
 - Core namespaces, 53
 - Create-Set-Call usage pattern
 - aggregate component design, 241
 - of component-oriented design,
237–238
 - Custom attributes, 209–211
 - Custom collections, 219–220
 - Custom exceptions
 - design guidelines, 202–203
 - exception throwing, 189
 - FxCop rules for, 305
 - Customization, protected members for,
165–166
 - Cwalina, Krzysztof
 - abstractions, 29, 172
 - acronym capitalization conventions,
37
 - aggregate components, 236, 237, 240,
241
 - API specification, 313–314, 315
 - attributes, 211
 - base classes, 173
 - choice between class, interface, 80, 81
 - custom collections, 219
 - delegate calls, 167
 - exception filters, 188
 - exception throwing, 184–185
 - explicit interface member
implementation, 112
 - extensibility limitations, 175
 - factories, 263
 - factorization, 265
 - framework design, 8, 261
 - ICloneable interface, 221
 - low barrier to entry principle, 22
 - naming conventions, 33
 - nested types, 103
 - operator overloads design, 142
 - property names, 62
 - scenario-driven design principle, 14
 - sealing, 176
 - self-documenting object models
 - principle, 24, 25
 - standard exception types, 198
 - subnamespaces, 73
 - type names, 54, 55
 - types and namespaces, 70
 - usability studies, 17
 - virtual members, 169
 - word choice conventions, 42
- D**
- Deallocation, 74–75
 - Debugging
 - exceptions, 194–195
 - with `Object.ToString`, 227–228
 - Deep-copy, cloning, 221
 - Default arguments, 109–110
 - Default constructors. *See also* Instance
constructors
 - design guidelines, 126–130
 - for struct, 89

- Default values, 121
 - Delegates
 - events and, 132
 - extensibility with callbacks, events, 166–168
 - groupings of types, 67–68
 - Derived class, 55–56
 - Descriptive identifier names, 25–26
 - Descriptive parameter names, 64–65
 - Design conventions, framework, 2–3
 - Design patterns
 - aggregate components, 235–243
 - Async Pattern, 243–248
 - Dispose Pattern, 248–260
 - factories, 260–263
 - FxCop rules for, 309
 - Optional Feature Pattern, 264–267
 - Template Method Pattern, 267–269
 - timeouts, 269–271
 - Design Patterns* (Gamma et al), 267
 - .Design subnamespace, 73
 - Design usage guidelines. *See also* FxCop
 - arrays, 207–209
 - attributes, 209–211
 - collections, 211–220
 - equality operators, 231–233
 - ICloneable interface, 221
 - IComparable<T>, IEquatable<T>, 222–223
 - IDisposable, 223
 - System.Object, 224–228
 - System.Uri, 228–230
 - System.Xml Usage, 230–231
 - Dictionary suffix, 220
 - Dictionary<TKey,TValue>, 212–213
 - Disposable types, 250
 - Dispose (bool) method
 - Basic Dispose Pattern
 - implementation, 251–256
 - finalizable types and, 257
 - Dispose methods, 250
 - Dispose Pattern, 248–260
 - Basic Dispose Pattern
 - implementation, 251–256
 - finalizable types, 256–260
 - IDisposable known as, 223
 - implementation guidelines, 249–251
 - memory management, 248–249
 - DLL. *See* Dynamic Link Library
 - Documentation
 - of abstractions, 171
 - self-documenting object models
 - principle, 23–29
 - Doer
 - danger of, 204
 - preconditions check with, 186
 - Dot separators, 65
 - Duffy, Joe, 253
 - Dussud, Patrick, 122
 - Dynamic Link Library (DLL)
 - FxCop rules for, 288
 - naming conventions for, 48–49
- E**
- 80/20 rule, 8
 - Elapsed time, 15
 - Encapsulation, 139
 - Enclosing type, 101–102
 - End method
 - Async Pattern design, 245, 246
 - Async Pattern implementation, 248
 - function of, 244
 - Enum parameters
 - choice between Boolean parameters and, 150–152
 - validation of, 153–154
 - Enumerators, 102
 - Enum.IsDefined, 153–154
 - Enums
 - design guidelines, 91–101
 - function of, 67
 - FxCop design rules for, 294–295
 - FxCop rules for names of, 290
 - groupings of types, 68
 - naming conventions, 59–60
 - Equality operators
 - IEquatable<T> and, 223
 - use guidelines, 231–233
 - Error codes

- exception throwing and, 184–185
- exceptions, performance and, 203
- for timeout expiration, 271
- Error handling code, 180
- Error message
 - exception error message design, 189–191
 - exception error message terminology, 196
- Error reporting
 - component-oriented design and, 238
 - exceptions for, 179–183
- Event arguments
 - description of, 132
 - design guidelines, 135
 - naming of, 63–64
- Event handlers
 - custom design, 138
 - design guidelines, 134–135
 - naming of, 63
- Event handling method, 132
- EventArgs, 135
- EventLog component, 236
- Events
 - aggregate components and, 239, 242
 - design guidelines, 132–138
 - extensibility with, 166–168
 - FxCop design rules, 298
 - FxCop name rules, 291–292
 - naming conventions, 63–64
- Ex abbreviation
 - avoidance of, 47
 - use of, 40
- Exception builder methods, 187–188
- Exception filter blocks, 188
- Exception handler, 182–183
- Exception handling, 303
- Exception throwing
 - Async Pattern design and, 245–246
 - choice of exception type, 189
 - dispose (bool) method and, 255–256
 - error message design, 189–191
 - exception handling, 191–195
 - FxCop rules for, 303, 304
 - guidelines for, 183–188
 - standard exception types and, 199
 - for timeouts, 270–271
 - wrapping exceptions, 195–197
- Exception types, standard
 - ArgumentException,
 - ArgumentNullException, and
 - ArgumentOutOfRangeException, 198–199
 - ComException, SEHException, 201
 - for exception throwing, 189
 - ExecutionEngineException, 201
 - FxCop rules for, 303–304
 - InvalidOperationException, 198
 - NullReferenceException,
 - IndexOutOfRangeException, and
 - AccessViolationException, 199–200
 - OutOfMemoryException, 200–201
 - StackOverflowException, 200
 - System.ApplicationException, 197–198
 - System.Exception and
 - System.SystemException, 197
- Exceptions
 - aggregate components and, 238, 239
 - argument validation and, 153
 - benefits of, 179–183
 - from constructor, 127
 - conversion operators and, 147
 - custom, 202–203
 - error message design, 189–191
 - exception handling, 191–195
 - exception message resource names, 65
 - FxCop rules for, 303–305
 - groupings of types, 67–68
 - for low barrier to entry, 23
 - performance and, 203–205
 - in property getter, 121–122
 - for self-documenting object models, 26–27
 - standard exception types, 197–201
 - throwing, 183–188, 189
 - type constructor design and, 131
 - wrapping exceptions, 195–197

- Execution failure
 - definition of, 183
 - report by exception throwing, 185
- ExecutionEngineException, 201
- Expense, framework, 3–4
- Experimentation, 19–23
- Explicit conversion operators, 146–147
- Explicit interface member
 - implementation, 111–115
- Extensibility
 - abstractions, 170–172
 - base/abstract classes and, 67
 - base classes, 172–174
 - choice of mechanisms, 163–164
 - events, callbacks, 166–168
 - extensibility mechanisms, 163–172
 - protected members, 165–166
 - unsealed classes, 164–165
 - virtual members, 168–170
- FxCop rules for, 302–303
 - sealing, 174–177
 - Template Method Pattern for, 268–269
- Extensible Markup Language (XML)
 - FxCop rules for System.XML, 308
 - System.XML, 230–231
- F**
- Factored design
 - Optional Feature Pattern and, 266
 - for optional features, 264–265
- Factored types
 - of aggregate components, 240
 - design guidelines, 243
- Factories
 - description of, 260–261
 - guidelines for, 261–263
- Factory method
 - description of, 260–261
 - guidelines for, 263
 - instead of constructor, 126
- Factory types
 - description of, 260
 - function of, 261
- naming, 263
- Fail fast, 185–186
- Failures
 - exception benefits and, 180–183
 - exception throwing, 183–188
 - execution failure, 183
 - reporting, 179
- Fanning, Mike
 - argument exceptions rule, 304
 - exception filters, 303
 - freeware version of FxCop, 284
 - FxCop analysis capabilities, 281
 - FxCop capitalization rules, 286
 - FxCop constructor design rules, 298
 - FxCop, custom rules with, 282
 - FxCop evolution, 283
 - ID issue, 285
 - root cause analysis of FxCop, 292
 - spelling checks, 287
 - Visual Studio and FxCop rules, 290
- Features, Optional Feature Pattern, 264–267
- Fields
 - design guidelines, 139–141
 - FxCop design rules, 298–300
 - FxCop name rules, 292
 - naming conventions, 64
- File organization, 278–279
- Filters, 188
- Finalizable types
 - definition of, 249
 - implementation guidelines, 256–260
- Finalize method
 - constructor design, 127
 - Dispose Pattern and, 250
 - release of unmanaged resources, 249
- Finalizers
 - Dispose Pattern implementation, 251–256
 - finalizable types, 256–260
 - release of unmanaged resources, 249
- Finally blocks
 - for cleanup code, 194
 - exception throwing and, 188
- Flag enums

- design guidelines, 96–100
- design of, 91–92
- FxCop design rules, 295
- Framework. *See also* .NET Framework
 - API specification sample, 311–317
 - application development history, 1–3
 - well-designed framework, 3–6
- Framework design fundamentals. *See also* FxCop
 - layered architecture principle, 29–31
 - low barrier to entry principle, 19–23
 - progressive frameworks, 9–12
 - for range of developers, 8–9
 - scenario-driven design principle, 13–19
 - self-documenting object models
 - principle, 23–29
 - simplicity/power, 7–8
 - usability of, 12–13
- FxCop
 - definition of, 281–282
 - design patterns rules, 309
 - evolution of, 282–283
 - exceptions rules, 303–305
 - extensibility design rules, 302–303
 - member design rules, 296–302
 - naming guidelines rules, 284–293
 - process of, 283
 - type design guidelines rules, 293–295
 - usage guidelines rules, 305–309
- G**
- Garbage Collector (GC), 248–249
- GC.SuppressFinalize method
 - Basic Dispose Pattern
 - implementation, 253–254
 - function of, 249–250
- Generator, 215
- Generic type parameters, 56–57
- George, Kit, 48
- Get-only properties
 - property design, 120
 - for required arguments, 210
- Good practices, 5
- Gray, Jan, 169
- Grep test, 44
- Grunkemeyer, Brian
 - Async Pattern design, 246
 - Async Pattern implementation, 248
 - static class, 86
- Gunnerson, Eric, 136–137
- H**
- Handle property, 118
- Harrison, Sheridan, 281–309
- Hash, 225–226
- Hashtable, 212–213
- Hejlsberg, Anders
 - explicit interface member
 - implementation, 111
 - multiframework design, 10–11
 - parameter passing, 156
 - value of enum, 99
- Hierarchy
 - interface for polymorphic hierarchy, 82
 - types, namespaces and, 70–74
- High-level APIs, 29–31
- Hungarian naming convention, 42
- I**
- I, 55, 56
- IAsyncResult.CompletedSynchronously, 246
- ICloneable interface
 - design usage guidelines, 220, 221
 - documentation for, 171
- ICollection, 213–214
- ICollection<T>
 - explicit interface member
 - implementation, 113
 - for properties, return values, 214–215
 - use guidelines, 213–214
- IComparable<T>, 222–223
- Identifier names
 - abbreviations, acronyms in, 43–44
 - acronym capitalization conventions, 36–38
 - capitalization rules for, 34–36, 284
 - conventions for, 276
 - FxCop word choice rules, 286–287

- Identifier names (*continued*)
 - language-specific names, avoidance of, 44–46
 - for self-documenting object models, 25–26
 - word choice conventions, 42–43
 - IDisposable, 223
 - IEnumerable, 211, 213–214
 - IEnumerable<T>
 - collection parameters, 213–214
 - collections implementation of, 211
 - on custom collection, 219
 - as generator, 215
 - for properties, return values, 214–215
 - IEnumerator, 213
 - IEnumerator<T>, 213
 - IEquatable<T>
 - design usage guidelines, 222–223
 - Object.Equals and, 225
 - struct design, 90
 - IList<T>, 214–215
 - Implementation
 - Async Pattern, 247–248
 - Basic Dispose Pattern, 251–256
 - Implicit conversion operators, 146–147
 - Indent usage conventions, 276
 - Indexed property, 122–124
 - IndexOutOfRangeException, 199–200
 - Infrastructure namespace, 52–53
 - Inheritance
 - base classes and, 173
 - unsealed classes and, 164–165
 - Inheritance hierarchy
 - interface design and, 86–87
 - type names and, 54–55
 - Initialization
 - for aggregate components, 241
 - low barrier to entry principle and, 20–21, 22–23
 - Insert method, 180
 - Instance constructors. *See also* Default constructors
 - constructor design guidelines, 126–130
 - type constructors and, 125–126
 - Instrumentation, 183
 - Int32, 95, 96
 - Integer timeouts, 270
 - Integration, of framework, 5–6
 - Intellisense
 - API names and, 47
 - Boolean property names and, 62
 - factories and, 261
 - identifier names and, 26
 - types, namespaces and, 71
 - Interface members, 296
 - Interfaces
 - choice between classes and, 77–83, 293–294
 - design guidelines, 86–89
 - explicit interface member
 - implementation, 111–115
 - function of, 67
 - FxCop design rules, 294
 - FxCop name rules, 288–290
 - groupings of types, 68
 - ICloneable interface, 221
 - naming conventions for, 54–60
 - nested types and, 103
 - Internal constructors, 84
 - .Interop subnamespace, 73–74
 - Invalid states, 238–239
 - InvalidOperationException
 - Async Pattern design and, 246
 - component-oriented design and, 238
 - use guidelines, 198
 - It-Just-Works concept, 236
 - Item, 123
 - IXPathNavigable, 230–231
- J**
- Jagged arrays, 208–209
- K**
- Kay, Alan, 7–8
 - Keyed collections, 216–217
 - Keywords, 43
- L**
- Language-specific names
 - avoidance of, 44–46

FxCop rules for, 287–288
Languages
 exception message and, 190–191
 scenario code samples and, 14, 16
Layered architecture principle, 29–31
Learning curve
 of multiframework platform, 9–10
 of progressive framework, 11–12
List<T>, 212
Live collections, 217–218
Lossy conversions, 147
Low barrier to entry principle
 description of, 19–21
 guidelines for, 21–23
Low-level APIs, 29–31

M

Managed memory, 248
Mariani, Rico
 argument validation, 152, 155
 choice between properties, methods,
 116, 117, 120
 enum design, 94, 95
 event design, 136
 exception wrapping, 196
 explicit interface member
 implementation, 112, 114, 115
 extensibility, 169
 indexed property design, 122
 interfaces, 83
 marker interface, 88
 member overloading, 106, 110
 Object.ToString, 228
 operator overloads, 144
 parameter design, 148, 149
 parameter passing, 156
 params array parameter, 158, 160
 pit of success, 12
 pointer parameters, 161
 property change notification events,
 124–125
 property design, 121
 reference/value types, 75–76
 Tester-Doer pattern, 204
 ToString, 202
 TryParse pattern, 205

 type design, 69
 types and namespaces, 71, 72
 varargs methods, 161
Marker interfaces, 87–88
MarshalByRefObject, 81
Member design
 constructors, general guidelines,
 125–130
 constructors, type constructor
 guidelines, 131–132
 conversion operators, 146–147
 event design, 132–138
 event handler design, custom, 138
 explicit interface member
 implementation, 111–115
 field design, 139–141
 FxCop rules for, 296–302
 indexed property design, 122–124
 member overloading, 105–110
 operator overloads, 141–146
 overloading operator ==, 146
 parameter design, 148–150
 parameter passing, 155–157
 parameters, choice between enum/
 Boolean, 150–152
 parameters, members with variable
 number of, 157–161
 pointer parameters, 161–162
 properties/methods, choice between,
 115–120
 property change notification events,
 124–125
 property design, 120–122
 validating arguments, 152–155
Member overloading
 design guidelines for, 105–110
 FxCop rules for, 296
Members
 naming new versions of existing
 APIs, 46–48
 protected members, extensibility
 with, 165–166
 sealing, 175, 176–177
 virtual members, extensibility with,
 168–170

- Memory management
 - Dispose Pattern, 249–260
 - Garbage Collector, 248–249
- Messages, 189–191
- Method parameters, 269
- Methods
 - for aggregate components, 239
 - choice between properties and, 115–120
 - exception throwing and, 185
 - FxCop rules for choice between properties and, 296–297
 - naming conventions, 60–61
 - operator overloads and, 144–146
- Microsoft
 - FxCop and, 282–283
 - FxCop capitalization rules, 285, 286
 - multiframework design, 9
 - namespace name, 50
- Microsoft Office, 182, 190–191
- Microsoft Windows, 10–11
- Modes, 242–243
- Moore, Anthony, 151, 152
- Morrison, Vance, 100
- Multiframework platform, 9–11
- Multiple inheritance, 86–87
- Mutable types
 - FxCop rules for field design, 298–299
 - readonly fields and, 141
- N**
- Name collisions, 71–72
- Names
 - for aggregate components, 240, 241
 - for indexed property, 123
- Namespace
 - assemblies, DLLs and, 48
 - FxCop rules for type design guidelines, 293
 - in layered architecture, 30–31
 - low barrier to entry principle and, 20, 21
 - type design guidelines and, 69–74
- Namespace, naming conventions
 - FxCop rules for namespace names, 288
 - guidelines for, 49–51
 - type name conflicts, 51–53
- Naming
 - c# coding style conventions, 277
 - custom collections, 220
 - factories, 263
 - parameters, 149–150
 - for self-documenting object models, 24–26
- Naming guidelines
 - capitalization conventions, 34–41
 - classes, structs, interfaces, 54–60
 - FxCop rules for, 284–293
 - general naming conventions, 41–49
 - namespaces, 49–53
 - parameters, 64–65
 - reasons to use, 33
 - summary, 66
 - type members, 60–64
- Nested types
 - design guidelines, 101–103
 - FxCop rules for, 295
- .NET Framework
 - acronym capitalization conventions, 37–38
 - API specification sample, 311–317
 - base classes of, 173
 - capitalization of compound words/ common terms, 40
 - choice between class, interface, 77–83
 - choice between properties, methods, 117, 120
 - exception types, standard, 197–201
 - low barrier to entry principle and, 22–23
 - memory management, 248–249
 - namespace names, 50–51
 - naming conventions, 25, 33
 - as progressive framework, 11–12
 - self-documenting object models principle, 28
 - static class in, 86

- subnamespaces, 73
- type design, 69
- type names, 55, 57–59
- usability of, 12–13
- usability studies, 17–18
- Nonspecific exceptions
 - catching/wrapping, 196
 - swallowing, 191–192
- NotSupportedException, 267
- Null
 - argument validation, 153
 - collection properties and, 217
 - event design and, 136–137
 - member overloading and, 109
 - as params array parameter, 160
- NullReferenceException, 199–200
- O**
- Obasanjo, Dare, 231
- Object-oriented programming (OOP)
 - for application development, 2
 - exceptions and, 179–180
 - extensibility and, 163
 - naming conventions, 42–43
- Object.Equals
 - equality operators and, 232
 - Object.GetHashCode and, 226
 - override, 222, 224–225
- Object.GetHashCode
 - Object.Equals and, 224–225
 - overriding, 225–226
- Objects
 - factories and, 260–263
 - FxCop rules for, 307
 - self-documenting object models
 - principle, 23–29
- Object.ToString, 227–228
- Office, Microsoft, 182, 190–191
- OOP. *See* Object-oriented programming
- Operator!=
 - IEquatable<T> and, 223
 - use guidelines, 231–233
- Operator overloads
 - design guidelines, 141–146

- FxCop rules for, 300
- Operators
 - conversion operators, design
 - guidelines, 146–147
 - FxCop rules for, 308–309
 - overloading operator ==, 146
- Optional Feature Pattern
 - guidelines for, 265–267
 - modeling requirements, 264–265
- Optional properties
 - of attributes, 210
 - constructor parameters and, 211
- Out parameters
 - Async Pattern design and, 245
 - design guidelines, 149
 - member overloading and, 108
 - passing, 156
- OutOfMemoryException, 200–201
- Overloading operator ==
 - complexity of, 146
 - design usage guidelines, 231–233
 - IEquatable<T> and, 223
- Overloads
 - choice between properties, methods
 - and, 116
 - equality operators, 231–233
 - FxCop rules for operator overloads, 300
 - FxCop rules for operators, 308
 - member overloading, 105–110
 - operator overloads, design
 - guidelines, 141–146
- Override
 - Object.Equals, 224–225
 - sealing, 177
- OverrideAt method, 265
- P**
- Parameter passing, 155–157, 301
- Parameters
 - Async Pattern design and, 245
 - capitalization conventions for, 35
 - choice between enum, Boolean, 150–152
 - collection, 213–214

340 ■ INDEX

- Parameters (*continued*)
 - design guidelines, 148–150
 - FxCop design rules, 301–302
 - FxCop name rules, 292
 - indexed property design and, 123
 - member overloading, 105–110
 - members with variable number of, 157–161
 - naming guidelines, 64–65
 - passing, 155–157
 - pointer parameters, 161–162
- Parametized constructor, 128
- Params array parameter
 - design guidelines, 157–159
 - member overloading and, 108
- Parsing
 - factory for, 262–263
 - TryParse pattern, 204–205
- PascalCasing convention
 - for acronyms, 284–285
 - for field names, 64
 - identifier naming conventions, 34–36
 - for namespace names, 51
 - for property names, 61
 - for resource names, 65
 - for type names, 55
- Passing, parameter, 155–157, 301
- Patterns. *See* Design patterns
- Pepin, Brian
 - Basic Dispose Pattern
 - implementation, 254
 - choice between properties, methods, 118
 - contract in abstract class, 82
 - Dispose Pattern, 250
 - event design, 135
 - Intellisense, 71
 - interfaces, 83
 - member overloading, 109
 - parameter passing, 156
 - params array parameter, 158
 - property design, 121
 - virtual members, 168
- Performance
 - exception throwing and, 186
 - exceptions and, 203–205
 - properties, methods and, 116, 117
- Period, 190
- .Permissions subnamespace, 73
- Pincus, Jon, 151
- Pointer parameters, 161–162
- Post-events, 133–134
- Power
 - framework design for, 7–8
 - simplicity and, 3
- PowerCollections project, 172
- Pre-events, 133
- Preconditions, 186
- Predefined object instances, 140
- Prefix
 - for custom collection name, 220
 - enumeration names and, 60
 - in namespace name, 50
 - type names and, 55, 56
- Productivity, 6
- Programmers, 8
- Programming languages, 184. *See also* Object-oriented programming
- Progressive frameworks, 9–12
- Properties
 - for aggregate components, 239, 242
 - of attributes, 209–211
 - choice between methods and, 115–120, 296–297
 - collection return from, 214–217
 - design guidelines, 120–122
 - field design and, 139–140
 - FxCop name rules, 291
 - indexed property design, 122–124
 - naming conventions, 61–62
 - property change notification events, 124–125
- Property bags, 27
- Property change notification events, 124–125
- Property getter
 - arrays for properties and, 219
 - exception from, 121–122
 - snapshot collection and, 217
- Property setter, 120–121

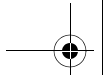
- Protected accessibility, 170
- Protected members
 - explicit interface member
 - implementation, 114–115
 - extensibility with, 165–166
 - sealing, 176–177
 - unsealed classes and, 164
- Protected virtual method, 135–136
- Public accessibility, 170
- Public constructors
 - abstract class design, 83–84
 - default, explicitly declared, 128
- Public members, 187
- Publicly accessible APIs. *See also*
 - Design usage guidelines
 - arrays in, 207–209
 - collections in, 211–220
- R**
- RAD (Rapid Application Development), 10
- Raise
 - event design and, 137–138
 - protected virtual method to raise
 - event, 135–136
 - use of term, 134
- Rapid Application Development (RAD), 10
- Read-only collection
 - for array, 208
 - ReadOnly prefix for, 220
- Read-only fields
 - arrays and, 207–208
 - design guidelines, 140–141
- ReadOnlyCollection<T>
 - for properties, return values, 214–215
 - return subclass of, 216
- Rector, Brent
 - catch blocks, 194
 - exception throwing, 189
 - exceptions and application, 182
 - framework with multiple languages,
 - 16
 - identifier names, 26
 - usability studies, 18
- Ref parameter, passing, 155, 156, 157
- Reference equality, 224
- Reference parameters, 245
- Reference test, 172
- Reference types
 - choice between class, struct, 74–76
 - equality operators on, 232–233
 - groupings of types, 67–68
 - Object.Equals on, 224, 225
- Reflection, 197
- Renaming, 113–114
- Required properties, 209–210
- Reserved enum values, 93
- Reserved parameters, 148–149
- Resources
 - FxCop rules for names of, 292–293
 - naming conventions, 65
- Return codes
 - error handling, 181
 - exception benefits for, 180–181
- Return-value-based error reporting,
 - 179–180
- Return values
 - collection, 214–217
 - of Object.GetHashCode, 226
- Richter, Jeffrey
 - abstractions, 171, 172
 - assembly names, 48–49
 - asynchronous operations, 244
 - Basic Dispose Pattern
 - implementation, 253, 254
 - case sensitivity, 41
 - choice between class, interface, 81, 83
 - choice between properties, methods,
 - 118
 - constructor design, 127
 - default arguments, 110
 - enum design, 92, 97
 - event design, 137
 - exception handling, 192, 193, 194, 195
 - exception message, 190–191
 - exception throwing, 189
 - exception wrapping, 197
 - exceptions for error reporting, 180

- Richter, Jeffrey (*continued*)
- exceptions, program failures, 181–182
 - explicit interface member implementation, 112
 - field design, 140
 - finalizable types, 259
 - language-specific names, 44–45
 - standard exception types, 197–198, 199
 - static class, 86
 - syntax for exceptions, 184
 - Tester-Doer pattern, 186, 204
 - TryParse pattern, 205
 - type design, 69
 - type names, 55, 56
 - types and namespaces, 70, 71, 72
 - unhandled exception handler, 183
 - value types, 76
 - word choice conventions, 43
- Rules. *See* FxCop
- S**
- Scenario code samples, 13–16
- Scenario-driven design principle
- description of, 13–14
 - guidelines for, 14–17
 - usability studies, 17–19
- Sealing
- custom attributes, 211
 - explicit interface member implementation and, 114
 - extensibility with, 174–177
 - FxCop rules for, 302–303
- Security
- exception message and, 190
 - explicit interface member implementation and, 114
- Security-sensitive information
- custom exceptions and, 202–203
 - Object.ToString and, 228
- SEHException, 201
- Self-documenting object models
- principle
 - abstractions, limiting, 28–29
 - consistency, 27–28
 - description of, 23–24
 - exceptions, 26–27
 - naming, 24–26
 - strong typing, 27
- Sells, Chris
- finalizable types, 258
 - IEnumerable<T>, 215
 - Object.ToString, 227
- Sender parameter, 63
- Sentinel values, 93–94
- Set-only properties, 120–121
- Settable properties
- collections and, 214
 - for optional properties, 210
- Shallow-copy, cloning, 221
- Simple enums
- example of, 91
 - value of zero on, 95
- Simple overloads, 21
- Simplicity, 3, 7–8
- Snapshot collections, 217–218
- Space usage conventions, 275–276
- Specification sample, API, 311–317
- StackOverflowException, 200
- Standard design methodologies, 16–17
- Standard exception types. *See* Exception types, standard
- Starck, Steve, 266
- State object, 244
- Static class
- design guidelines, 85–86
 - function of, 67
 - FxCop rules for, 294
 - groupings of types, 68
- Static constants, 92
- Static constructors. *See* Type constructors
- Static fields, 131–132
- Stopwatch class, 312–317
- StreamReader
- choice between class, interface, 78–80
 - usability studies, 18
- String-based overloads, 229–230

- StringReader, 18
 - Strong typing
 - enums for, 92
 - FxCop rules for, 306
 - for self-documenting object models, 27
 - Structs
 - choice between class and, 74–76
 - design guidelines, 89–90
 - explicit default constructors on, 128–129
 - function of, 67
 - FxCop design rules, 294
 - FxCop name rules, 288–290
 - groupings of types, 68
 - naming conventions for, 54–60
 - Style conventions, 274–276
 - Subnamespace names, 73–74
 - Suffix
 - for attributes, 210
 - for collections, 220
 - enumeration names and, 59–60
 - naming new versions of existing APIs, 47
 - type names and, 57–59
 - Sutter, Herb
 - Basic Dispose Pattern
 - implementation, 252
 - finalizable types, 259
 - finalizers, 257
 - Swallow, exceptions, 191–192
 - Syntax, 184
 - System resources
 - Dispose Pattern for, 249–260
 - memory management, 248–249
 - System.ApplicationException, 197–198
 - System.ArgumentException, 153
 - System.Attribute, 209–211
 - System.ComponentModel.IComponent, 172
 - System.Data, 22–23
 - System.DateTime, 106
 - System.Decimal, 141–142
 - System.Enum, 96–97
 - System.Environment class, 86
 - System.Environment.FailFast, 185–186
 - System.EventHandler<T>, 134–135
 - SystemEvents, 168
 - System.Exception
 - throwing, 189
 - use guidelines, 197
 - System.FlagsAttribute, 97
 - System.IDisposable interface
 - Basic Dispose Pattern
 - implementation, 251–256
 - release of unmanaged resources
 - with, 249
 - System.InvalidOperationException, 198
 - System.IO namespace, 17–18
 - System.IO.Stream abstract class, 77–80
 - System.IO.Stream class
 - Basic Dispose Pattern and, 251
 - Optional Feature Pattern and, 266
 - System.Messaging.MessageQueue, 23
 - System.Net, 21
 - System.Net.WebClient, 236
 - System.Object
 - Object.Equals, 224–225
 - Object.GetHashCode, 225–226
 - Object.ToString, 227–228
 - release of unmanaged resources, 249
 - System.Runtime.Serialization
 - namespace, 70
 - System.ServiceProcess namespace, 237
 - System.SystemException, 197
 - System.TimeoutException, 270–271
 - System.Uri
 - FxCop rules for, 307–308
 - implementation guidelines, 229–230
 - use guidelines, 228–229
 - System.Web namespace, 30, 31
 - System.XML
 - APIs, 61
 - FxCop rules for, 308
 - use guidelines, 230–231
 - Szyperski, Clemens, 101
- T**
- T, 57

- Technology namespace groups, 53
 - Template Method Pattern, 267–269
 - Tester
 - danger of, 204
 - preconditions check with, 186
 - Tester-Doer pattern
 - exception throwing and, 203–204
 - preconditions check with, 186
 - TextReader, 18
 - ThreadPool.QueueUserWorkItem, 248
 - Throwing. *See* Exception throwing
 - Timeouts
 - guidelines for, 269–271
 - System.IO.Stream and, 77–80
 - TimeSpan, 270
 - ToString, 202
 - Trademark, 38
 - Try-catch, 193
 - Try-finally, 193, 194
 - TryParse pattern
 - guidelines for, 204–205
 - Tester-Doer pattern and, 186
 - Two-character acronyms, 37
 - Type constructors
 - design guidelines, 131–132
 - FxCop design rules, 297–298
 - instance constructors and, 125–126
 - Type design guidelines
 - abstract class design, 83–85
 - categories of, 67–68
 - class, interface, choice between, 77–83
 - class, structs, choice between, 74–76
 - enum design, 91–101
 - FxCop rules for, 293–295
 - interface design, 86–89
 - namespaces and, 69–74
 - nested types, 101–103
 - static class design, 85–86
 - struct design, 89–90
 - summary, 104
 - Type members, naming guidelines, 60–64
 - events, 63–64
 - fields, 64
 - FxCop rules for, 291–292
 - methods, 60–61
 - properties, 61–62
 - Type parameters, 56–57
 - Types
 - constructor design guidelines, 126–130
 - factored types, 240
 - FxCop name rules, 289–290
 - FxCop type design rules, 293
 - low barrier to entry principle and, 20, 21–22
 - namespace, type name conflicts, 51–53
 - naming, 24–26
 - naming conventions for classes, structs, interfaces, 54–60
 - naming new versions of existing APIs, 46–48
 - nested types, 101–103
 - Types, design usage guidelines
 - arrays, 207–209
 - attributes, 209–211
 - collections, 211–220
 - equality operators, 231–233
 - ICloneable interface, 221
 - IComparable<T>, IEquatable<T>, 222–223
 - IDisposable, 223
 - System.Object, 224–228
 - System.Uri, 228–230
 - System.Xml Usage, 230–231
- U**
- UEF (unhandled exception filter), 182–183
 - Underlying type, 95–96
 - Unhandled exception, 190–191
 - Unhandled exception filter (UEF), 182–183
 - Unhandled exception handler, 182–183
 - Uniform Resource Identifiers (URIs)
 - FxCop rules for, 307–308
 - System.Uri for, 229–230
 - Unmanaged resources

- Dispose Pattern for, 249–260
 - finalizers for, 248–249
 - Unsealed classes, 164–165
 - UrtCop. *See* FxCop
 - Usability
 - framework design for, 12–13
 - naming conventions for, 33
 - of scenario-driven framework design, 17–19
 - Usage guidelines, 305–309
- V**
- Validation, argument, 152–155
 - Value equality, 224
 - Value types
 - choice between class, struct, 74–76
 - enum design, 91–101
 - equality operators on, 232
 - explicit interface member
 - implementation, 111–115
 - groupings of types, 67–68
 - IEquatable<T> on, 222
 - interface design and, 87
 - interface for polymorphic hierarchy, 82
 - Object.Equals on, 224, 225
 - struct design, 89–90
 - Values
 - to enums, 100–101
 - of properties, 116
 - property change notification events and, 125
 - Varargs methods, 160
 - VB. *See* Visual Basic
 - Vick, Paul
 - .NET Framework unification, 11
 - case sensitivity, 41
 - framework design, 8
 - framework with multiple languages, 16
 - layered architecture principle, 29
 - low barrier to entry principle, 20
 - parameters in VB, 155
 - Virtual members
 - callbacks *vs.*, 166–168
 - constructor design and, 129–130
 - explicit interface member
 - implementation and, 114–115
 - extensibility with, 168–170
 - member overloading, 108–109
 - sealing, 176–177
 - Template Method Pattern and, 267–269
 - unsealed classes and, 164
 - Visibility, 240, 241
 - Visible instance fields, 300
 - Visual Basic (VB)
 - case sensitivity and, 41
 - framework design for programmers, 8
 - layered architecture, 29
 - libraries, design of, 9
 - low barrier to entry, 20
 - parameters in, 155
 - Rapid Application Development, 10
 - word choice conventions, 43
 - Visual Basic.NET
 - attribute in, 211
 - finally blocks of, 194
 - usability problems of, 13
 - Visual Studio
 - aggregate component integration with, 237
 - FxCop rules and, 286
 - Intellisense, 71
 - Visual Studio Designers, 243
- W**
- Well-designed framework, 3–6
 - Win32
 - CloseHandle, 181
 - failure reporting of, 179
 - framework design of, 9
 - Windows, Microsoft, 10–11
 - Word choice
 - FxCop rules for, 286–287
 - naming conventions for, 42–43
 - Wrapping, 195–197



X

XML. *See* Extensible Markup Language
XMLDataDocument, 231
XmlReader, 230–231

Z

Zero, value of
flag enum design, 99–100
on simple enum, 95

