

Index

Symbols

- 145
- 145
- ; 141
- ! 145
- != 148
- .NET Framework
 - downloading 23
 - installing 23
- .NET Remoting
 - Singleton mode 548
- .NET remoting 546–560
 - configuration files 553
 - network communication handled by 549
 - passing collection of Person objects between remote object and client 555
 - persisting remote object state 550
 - purpose of 546
 - registering channels 548
 - registering service name 548
 - registering well known service types 548
 - remote object access via interface 551
 - serializing complex objects 555
 - simple example 547
 - SingleCall mode 548
 - SingleCall vs. Singleton remote object modes 550
 - swapping remote objects
 - enabling with interfaces 551
 - three primary channels 547
 - three required components 546
- .NET Remoting Architecture 546
- .NET remoting infrastructure 580
- [ContractInvariantMethod] attribute 784
- [Serializable] attribute 484
- * 146
- / 146
- /d
 - DEBUG compiler switch 763
- #endregion directive 636
- #region directive 636
- % 146
- + 145

- ++ 145
- += operator 354
- < 148
- << 147
- <= 148
- = 140
- == 148
- > 148
- >= 148
- ~ 145

A

- abstract
 - classes 311
 - methods 311
- abstract class 300, 311
 - expressing in UML 311
 - purpose of 311
 - term defined 300
- abstract class vs. interface 314, 315
- abstract data types 221
- abstract keyword
 - using to declare classes and methods 313
- abstract methods
 - implementing in derived classes 313
- abstract thinking 10
- abstraction
 - problem 10
 - the art of programming 220
- abstractions
 - selecting the right kinds of 781
- access
 - horizontal 318
 - vertical 318
- Access Control Graph (ACG) 800
- access modifiers 232
 - default/package 233
 - most often used 318
 - private 232
 - protected 232
 - public 232
- address bus 98
- addressing local machine 533
- ADO.NET 580
- aggregation 272, 274, 276, 291
 - aggregate constructors 275
- composite 274, 275, 291
 - composite example code 278
 - definition 274
 - determining type by who controls object lifetime 274
 - effects of garbage collector 275
 - example
 - engine simulation 281
 - engine simulation class diagram 281
 - simple 274, 276, 291
 - simple example code 276
 - two types of 801
- algorithm
 - running time 103
 - understanding the concept of 92
 - working definition of 101
- algorithm growth rate 103
- algorithms 92, 101
 - good vs. bad 101
- alter statement
 - used to create foreign key constraint 601
- analysis 53, 792
- Ansel Adams 792
- API Framework
 - blessing and curse 114
- API reference documentation
 - class general overview page 115
 - class member page 117
 - obsolete APIs 122
 - Syntax section 121
- API reference information
 - definitive source 114
- application
 - definition 131
 - graceful recovery 50
 - layers 532
 - physical deployment 532
 - physical tier distribution 535
 - simple
 - structure 131
 - tier responsibilities 535
 - tiers 532
- Application class
 - Run() method 341
 - running GUI programs with 341
- application distribution 532
 - across multiple computers 534

- application domain 450
 - application layer 536
 - application layers 580
 - application message loop 341
 - application tiers 535
 - logical 534
 - separation of concerns 535
 - ApplicationException 432
 - applications
 - multitiered 534
 - architectural diagram
 - multitiered database application 580
 - architecture
 - flexibility 793
 - modularity 793
 - reliability 794
 - stability 794
 - array 397, 400
 - creating with literal values 195
 - declaration syntax 189
 - definition of 188
 - difference between value type and reference type arrays 196
 - dynamic resizing
 - example code 397
 - elements 188
 - functionality provided by array types 190
 - homogeneous elements 188
 - Main() method String parameter 210
 - multidimensional 203, 207
 - of value types 192
 - properties of 191
 - references
 - calling Array class methods on 194
 - single dimensional 192
 - single dimensional in action 198
 - specifying length 189
 - specifying types 189
 - two dimensional
 - example program 207
 - type inheritance hierarchy 190
 - value type
 - memory arrangement 193
 - Array class 211
 - array initializer expression 205
 - array literal 195, 196
 - array of arrays 206
 - array processing 50
 - array-based collection
 - growing on insertion 397
 - arrays 188
 - rectangular 203
 - sorting with Array class 211
 - two-dimensional
 - processing 69
 - using to solve problems 188
 - Ashmore's hash code algorithm 743
 - assembler 96
 - assembly
 - definition 131
 - assembly language 96
 - assemblymode
 - setting 784
 - Assertion Failed dialog 763
 - association 273, 291
 - definition 273
 - associativity
 - operator 144
 - forcing 144
 - asynchronous method calls 470
 - asynchronous methods
 - EndInvoke() method 472
 - IAAsyncResult interface 473
 - obtaining results from 472
 - providing callback method to BeginInvoke() method 473
 - attribute candidates 51
 - attributes 50
 - automated water tank custom event example 380–387
 - auxiliary storage device 480
- B**
- BackgroundWorker 448
 - events 464
 - BackgroundWorker class 464
 - bad software architecture
 - characteristics of 780
 - base class
 - methods
 - overriding 309
 - source code example 302
 - BaseCommand class 827
 - BaseDAO
 - class definition
 - using DatabaseFactory class 613
 - behavior
 - generalized 298
 - behavior contract 724
 - Bertrand Meyer 774, 795
 - Bertrand Meyer's Design by Contract (DbC) 759
 - binary data 492–494
 - BinaryFormatter class 485, 570
 - BinaryReader class 492, 494, 496
 - BinaryWriter class 492, 494
 - bit 97, 98
 - BitMap class 348
 - using to create Image object 349
 - Bloch's hash code algorithm 743
 - block 562
 - blocking I/O operation 562
 - Bounds
 - data that comprises 347
 - property
 - printing to screen 347
 - Bounds property 347
 - setting example 351
 - boxing 261
 - break 167
 - bridge 529
 - buses 93
 - Business Layer 580
 - business object
 - definition 580
 - business objects 580
 - business rules 580
 - creep 581
 - Button 339
 - byte 97, 98
- C**
- C# compile and execute process 104
 - cache memory 97
 - calling base class constructor with base() 303
 - cascade delete 589
 - SQL
 - cascade delete
 - testing 605
 - casting 307, 411
 - advice on use of 308
 - chained hash table 405
 - character constants
 - declaring
 - example 61
 - chipset 93
 - Christopher Alexander 816
 - class 131, 299
 - abstract 311
 - expressing in UML 311
 - purpose of 311
 - abstract class 300
 - four categories of members 225
 - non-static fields 225

- sealed 319
 - static fields 225
 - term definition 300
 - class declarations
 - viewed as behavior specifications 760
 - class definition
 - adding fields 239
 - adding instance methods 239
 - constructor method 240
 - starting 239
 - class invariant 761, 763
 - defined 761
 - class invariants 760, 761
 - class member access
 - default when omitting access modifier 318
 - classes
 - classes vs. structs 260
 - number in an application 272
 - Class-Wide Fields 225
 - Click event 352
 - client 528, 531
 - application 528, 531
 - hardware 528, 531
 - client application 546
 - client coordinates 347
 - client-server applications
 - See also TCP/IP client-server
 - TCP/IP 560
 - with .NET remoting 546
 - cloning objects 738
 - CloseReader() method 615
 - Coad's Inheritance Criteria 797
 - code blocks
 - executing in if statements 163
 - Code Contracts 760, 782
 - ccrewrite tool 782
 - code library
 - creating 104
 - code module
 - creating 104
 - code reuse 792
 - coding convention
 - adopting 879
 - cohesion 17, 234
 - collaboration
 - peer-to-peer 274
 - collateral roles
 - modeling 799
 - collections
 - ArrayList
 - usage example 399
 - casting 411
 - extending ArrayList 412
 - extracting elements into arrays 422
 - general characteristics 396
 - generic
 - example code 414–417
 - KeyedCollection<TKey, TItem> example 415
 - List<T> 414
 - IComparer<T, T> 421, 751
 - implementing IComparable<T> 418, 747
 - interfaces 396
 - linked list node elements 402
 - making an object sortable 418, 747
 - non-generic to generic mapping table 408
 - old-school style 409–413
 - old-school style programming 407
 - performance characteristics
 - arrays 401
 - hashtable 402
 - linked list 402
 - Person list example 411
 - red-black tree node elements 405
 - sorting 417
 - rules for implementing IComparable<T>.CompareTo() method 419, 750
 - specialized 408
 - underlying data structures 408
 - using foreach to iterate over example 411
- Color structure 348
 - columns 588
 - command console layout properties
 - modifying 31
 - command pattern 816, 827
 - command-line arguments
 - processing 210
 - command-line compiler 22
 - command-line tools 22
 - why you should learn 22
 - Common Language Infrastructure
 - four parts 106
 - Common Language Infrastructure (CLI) 104, 105
 - Common Language Runtime (CLR) 107
 - Common Language Specification (CLS) 107
 - Common Log File System 513
 - Common Type System (CTS) 106
 - compiler errors
 - dealing with 33
 - finding their meaning on MSDN 33
 - fixing 15
 - compiling
 - simple application 131
 - compiling multiple source files 272
 - compiling source file
 - how to 32
 - compiling with csc
 - using target switch example 249
 - complex application behavior 272
 - complex project folder organization 606
 - complexity
 - conceptual 15, 272, 273, 291
 - managing physical 16
 - physical 16, 272, 273, 291
 - relationship between physical and conceptual 17
 - Component 340
 - components
 - adding to Controls collection 351
 - adding to windows 350
 - initializing in separate method 351
 - composite aggregation
 - defined 274
 - composition 792, 801
 - as force multiplier 801
 - compositional design 272, 801
 - compositionists 792
 - computer
 - architecture
 - feature set 95
 - feature set accessibility 95
 - feature set implementation 95
 - three aspects of 95
 - definition of 92
 - memory
 - organization 96
 - processing cycle 100
 - system 92
 - components of 92
 - hard drive 92
 - keyboard 92
 - main logic board 92
 - memory 92, 96
 - monitor 92
 - mouse 92
 - processor 92
 - speakers 92
 - system unit 92
 - vs. computer system 92

- computer network
 - definition 528
 - purpose 528
 - computer program
 - modeling real world problem 220
 - computers 92
 - conceptual complexity 15, 272
 - managing 15
 - taming 16
 - concrete class 302
 - concurrently executing applications 450
 - condition
 - exception 430
 - configuration file
 - example 620
 - configuration files
 - .NET remoting 553
 - configuration-management tool 16
 - connection pooling 581
 - connection string
 - database
 - configuration file setting 587
 - console applications 130–155
 - console text menus
 - example 58
 - const 228, 726
 - constant 51, 226
 - constants 228
 - constraint
 - database 592
 - constructor chaining 303
 - constructor methods 238
 - constructors 724
 - ContainerControl 340
 - containing aggregate 276
 - containment
 - by reference 275
 - by value 275
 - polymorphic 801
 - contains 276
 - continue 178
 - Control 340
 - control bus 98
 - controller 825
 - controls
 - dynamic layout of 358
 - registering event handler methods 352
 - Controls collection
 - use of 351
 - coordinates
 - client 345
 - origin 347
 - screen 345
 - (x,y) pairs 346
 - origin 347
 - pixel as basic unit of measure 346
 - window 345
 - window placement upon screen 346
 - copy constructor 735
 - copy constructors 734
 - coupling 17
 - create tables SQL script 592
 - creativity
 - and problem abstraction 220
 - cross platform
 - promise of 107
 - CRUD operations
 - database
 - CRUD operations 615
 - csc
 - compiling entire source directory 273
 - compiling multiple source files 272
 - csc compiler
 - locating 23
 - current position 52
 - custom event
 - recursive example 381–384
 - custom events 376
 - suggested naming convention 388
 - custom exceptions 439
 - custom serialization 726, 729
- D**
- DAO layer
 - building 609
 - Data Access Layer 580
 - data access object
 - definition 581
 - data access objects 580
 - data base
 - key factor in business rules 581
 - data bus 98
 - Data Control Language 589
 - Data Definition Language 589
 - data link layer 536
 - Data Manipulation Language 589, 595
 - data type 51
 - reference 190
 - value 190
 - data types
 - array 190
 - SQL Server 593
 - database
 - automatically inserting primary key 601
 - cascade delete 589
 - columns 588
 - constraint
 - definition of 592
 - creating related table with script 600
 - DataBase.AddInParameter()
 - method 618
 - foreign key 589, 600
 - foreign key constraint
 - naming 601
 - inserting test data into related table 602
 - inserting value objects into 618
 - join operation 600
 - primary key 589
 - record 604
 - referential integrity 589
 - rows 588
 - table 588
 - database application
 - compiling 587
 - database connection
 - established via DatabaseFactory 581
 - database connection string 587
 - database connection test application 586–588
 - database management system 588
 - Database object 586, 615
 - database script
 - running
 - example 592
 - DatabaseFactory 581, 615
 - DatabaseProviderFactory 586, 610
 - configuration file 586
 - DataBindingComplete event 683
 - datagrams 538
 - DataGridView 661, 663
 - clicking on row to yield row index 661
 - data binding 683
 - DataSource property 661
 - row index value 661
 - DateTime structure
 - example of use 360
 - DateTime.Now 360
 - DbC 759
 - DbCommand 586, 618

- DBMS 588
 - DbType enumeration
 - .NET type mapping table 619
 - Debug.Assert() method 761
 - deep copy 722
 - defined 734
 - default class member access 318
 - default constructor 231
 - delay
 - example code 386
 - delegate 339, 376
 - event subscriber list 376
 - EventHandler 377
 - method signature specification 377
 - delegate object
 - purpose of 376
 - delegate type
 - purpose of 353
 - specification of method signature 353
 - delegates
 - EventHandler 352
 - MouseEventHandler 352
 - PaintEventHandler 353
 - running asynchronous methods with 470
 - delete command
 - SQL
 - commands
 - delete 599
 - delimiter
 - text file 490
 - Department of Defense 529
 - dependencies
 - managed 794
 - dependency 224, 273
 - definition 273
 - effects of dependency relationships between classes 273
 - Dependency Inversion Principle 780
 - dependency relationship 291
 - dependency vs. association 273
 - deprecated members 233
 - derived class
 - source code example 302
 - deserialization
 - object 485
 - deserialize
 - object
 - from XML file 487
 - design 792
 - design by composition 272
 - Design by Contract 759
 - design pragmatists 793
 - development cycle 47
 - application 56
 - applying 47
 - code 47, 874
 - creating feature implementation
 - lists 56
 - deploying 47
 - integrate 47
 - iterative application 56
 - plan 47, 874
 - refactor 47
 - test 47, 874
 - using 47
 - development environment
 - configuring 24
 - device driver 481
 - difference between abstract class and interface 314
 - difference between readonly and const fields 228
 - direct base class 224
 - direction 51
 - directory
 - definition 481
 - Directory class 481
 - example code 482
 - DirectoryInfo class 481
 - disk
 - driver software 480
 - distributed applications 528
 - DockStyle enumeration 359
 - values 359
 - documentation generation 86
 - dominant roles
 - modeling 799
 - doubly-linked circular list 402
 - Doxygen 86
 - Doxygen version 1.8.8 for Windows 86
 - Dr. Barbara Liskov 759
 - Dr. Bertrand Meyer 759
 - DreamSpark 42
 - driver
 - creating test code 242
 - dynamic factory pattern
 - advantages of 824
 - dynamic link library 104
 - dynamic polymorphic behavior 774
 - DynamicArray
 - case study 396
- ## E
- ease of maintenance 793
 - ECMA - 335 105
 - effects of change
 - predicting 794
 - Eiffel 759
 - EmployeeDAO 581
 - empty statement 140
 - Encapsulation 9
 - encapsulation 232
 - EndInvoke() method 472
 - engineering trade-off 793
 - Enterprise Library Configuration tool 587
 - Enterprise Library Data Access Application Block 580
 - Enterprise Library Version 6 584
 - EnterpriseLibrary6-binaries.exe 584
 - entry point 131
 - enumerated type 65
 - environment variable 22
 - environment variables 24–27
 - Erich Gamma 817
 - error checking 50
 - error conditions
 - program
 - handling 161
 - that cause exceptions
 - examples of 430
 - errors
 - compiler 15
 - Ethernet 538
 - event 376
 - event arguments
 - example code 378
 - event consumer 376
 - event driven 342
 - event driven programs 341
 - event handler
 - explicit call to
 - example 684
 - event handler methods
 - registering 352
 - event handlers
 - located in different objects 355
 - event producer 376
 - event publisher 376
 - event subscriber 376
 - event subscriber list 376
 - events 231, 352
 - and their delegate types
 - table of 352
 - BackColorChanged 352
 - BackgroundImageChanged 352

- Click 352
- DoubleClick 352
- GotFocus 352
- GUI
 - handling in separate object example 357
- handled in separate objects 355
- MouseClicked 352
- MouseDoubleClick 352
- MouseDown 352
- MouseEnter 352
- MouseLeave 352
- MouseMove 352
- MouseUp 352
- Paint 353
- registering event handler method example of 354
- Exception
 - class hierarchy 432
 - public properties 434
- exception
 - definition 430
- exception information table 431
- exceptions 430–442
 - catch block 430
 - catching multiple exceptions rule of thumb 437
 - catching with try/catch block 162
 - CLR handling mechanism 430
 - custom 439
 - extending Exception class 439
 - using throw keyword 441
 - determining what a method may throw 433
 - documenting 441
 - fault handler code 430
 - low-level to high-level translation 498
 - purpose of 430
 - runtime vs. application 432
 - translating low-level to high-level 441, 498
 - try block 430
 - try/catch/finally blocks 435–439
 - using multiple catch blocks 437
- executing application
 - how to 33
- executing SQL command
 - example code 586
- extension inheritance
 - complications from using 801
 - vs. functional variation 801
- F**
 - façade 816
 - factory 816
 - factory class
 - interfaces involved to employ 799
 - fault handler code 430
 - Fields 225
 - fields
 - readonly 226
 - initializing static readonly fields in static constructor 226
 - readonly vs. const 228
 - file 481
 - definition 480
 - File class 481
 - file I/O 480–520
 - file position pointer 493, 494
 - File Transfer Protocol 537
 - FileDialogs
 - using 516–518
 - FileInfo class 481
 - example code 482
 - files
 - manipulating 481–483
 - FileStream class 485, 494
 - final project considerations
 - checklist 79
 - finalizers 232
 - First-In-First-Out (FIFO) 406
 - fixed-length records 494
 - flexibility 793
 - floor 51
 - flow 12
 - achieving 13
 - concept of 12
 - stages 13
 - flow charts 65
 - FlowDirection enumeration
 - values 360
 - FlowLayoutPanel 339, 358
 - properties
 - AutoSize 359
 - AutoSizeMode 359
 - Dock 359
 - FlowDirection 360
 - WrapContents 359
 - purpose of 358
 - folder 481
 - folder options
 - setting 28
 - foreign key 589, 600
 - foreign key constraint 601
 - Form 339, 340, 342
- class inheritance hierarchy 340
 - properties
 - BackColor 348
 - BackgroundImage 348
 - manipulating 348
 - simple form program 340
 - Text property 341
 - window types created with 340
- formatting
 - numeric strings
 - table 211
 - source code 80, 874
- from clause
 - use to join tables
 - SQL
 - from clause
 - use to join tables 604
- functional decomposition 9
- fundamental language features 50
- G**
 - gate 816
 - gateway 529
 - generalization
 - expressing in UML 300
 - generalized behavior
 - specifying 298
 - GetRegisteredWellKnowClient-Types() method 554
 - good design
 - goals of 793
 - good software architecture
 - characteristics of 781
 - goto 179
 - graphical user interface programming 340–370
 - guarded region
 - of try block 162
 - GUI
 - coding rhythm 368
 - data input dialog design 670
 - loading image in PictureBox
 - example code 620
 - opening image file with OpenFileDialog
 - example code 620
 - separating code from event handlers 355–357
 - using dialogs to enter data 670
 - GUI layout
 - using mock-up sketch to design 658
 - guillemet characters 224

H

hard disk 480
 hardest thing about learning to program 4
 has a 276
 hash code
 algorithm 743
 hash function 404
 hash table 400
 chained 405
 open address 405
 slot probe function 405
 Height property 351
 homogeneous data types 188
 horizontal access 233, 318, 725
 host 531
 HttpChannel 547
 Hypertext Transfer Protocol 537

I

ICloneable 738
 IDataReader 586, 619
 IDE 22
 identifier 134
 constant name examples 880
 method name examples 880
 naming 135, 879
 type name examples 879
 variable name examples 880
 identifiers 135
 forming 135
 if/else statement 164
 Image
 converting to byte array 618
 image
 using to set Form Background-Image property 349
 Image class 348
 IMessageFilter
 implementation example 344
 implementation approach 56
 implicit cast 412
 indexer
 example code 397
 indexers 231
 IndexOutOfRangeException
 handling 64
 infinite loop 172
 Infrastructure Layer 580
 inheritance 792, 794–798
 first purpose of 298
 good reasons for using 794

Meyer's Taxonomy 795
 object-oriented programming
 with 298
 second purpose of 298
 simple example 301
 third purpose of 299
 three purposes of 298
 valid usage checkpoints 797
 inheritance form
 constant 797
 extension 795
 facility 797
 functional variation 796
 implementation 797
 machine 797
 model 795
 reification 796
 restriction 795
 software 796
 structure 797
 subtype 795
 type variation 796
 uneffecting inheritance 796
 variation 796
 view 796
 inheritance hierarchy
 assessing with Coad's criteria 797
 navigating 120
 inheritists 792
 inner join 603
 instance 726
 instance constructors 231
 integral type size
 be aware of 146
 integrated development environment 22
 interface 299
 authorized members 299, 314
 purpose of 314
 reducing dependencies with 799
 role of 798
 term definition 299
 interface members
 mapping to abstract members 321
 interfaces 792
 expressing in UML 315
 Intermediate Language (IL) 104
 internal 232, 300, 304, 318
 Internet Protocol (IP) 538
 Internet protocol layers 536
 Internet Protocols 529
 inter-process communication 547

invariant
 definition 759
 IP 538
 IP address
 parsing with IPAddress.Parse() method 568
 IP addresses 538
 IPAddress.Parse() method 568
 IpcChannel 547
 purpose of 547
 is a relationship
 implementing 299
 iteration
 development 48
 iterative development 47

J

John Vlissides 817
 join operation 600
 just-in-time (JIT) compiler 104

K

keyword
 using as identifier
 example 134
 keywords
 reserved
 listing 134

L

Label 339
 language features 46, 56, 873
 language-features strategy area 53
 Last-In-First-Out (LIFO) 406
 layout managers 358–362
 least intimate collaboration 274
 legacy datafile adapter 496
 library
 creating with compiler
 example 547
 referencing with compiler switch
 example 548
 linked list 400
 reference rewiring 402
 Liskov Substitution Principle
 relationship to Meyer Design by Contract Programming 759
 three rules of 773
 Liskov Substitution Principle (LSP) 759
 List<T>
 example code 399

- Local Area Network 528
 - local method variables
 - local scope 236
 - local scope 236
 - local variable
 - example 63
 - local variable scoping rules 254
 - local variables 259
 - localhost 533
 - Location property 351
 - lock keyword 497
 - compared to `Monitor.Wait()`/
`Monitor.Exit()` 497
 - log files 513–516
 - loops 171
 - LSP 759
 - LSP & DbC
 - C# support for 760
 - common goals 760
 - designing with 760
- M**
- machine code 96, 104
 - machine instructions 96
 - Magic Draw UML Design Tool 281
 - Main method 130
 - main method
 - purpose 132
 - signatures 132
 - Main thread 450
 - managed code 107
 - managed threads 452
 - MarshalByRefObject 340
 - use to create remotable object
547
 - marshaling
 - remoting method calls 547
 - MemberwiseClone() 738
 - memory
 - address bus 98
 - alignment 98
 - bit 97, 98
 - byte 97, 98
 - cache 97
 - control bus 98
 - data bus 98
 - hierarchy 97
 - non-volatile 96
 - organization 96
 - RAM 96
 - ROM 96
 - volatile 96
 - word 97, 98
 - menu 51, 658
 - menus 363–366
 - adding submenu items to menus
365
 - item naming conventions 364
 - menu item separator
 - adding 365
 - menuitems
 - registering event handlers with
364
 - MenuStrip
 - docking to window 365
 - importance of adding last 365
 - MenuStrip class 363
 - ToolStripMenuItem 363
 - MenuStrip 363
 - declaring and creating 364
 - message categories 343
 - message filters
 - adding 344
 - message loop
 - window 342
 - Message Passing 281
 - message passing 272
 - message pump 343
 - message queue 342
 - message routing
 - windows 343
 - messages
 - system
 - how they are generated 343
 - Metadata 106
 - method
 - cohesion 234
 - definition structure 235
 - parameter list 131
 - sealed 319
 - signature
 - definition 132
 - method stubbing 15, 59
 - methods 51, 231, 234
 - abstract 311
 - body 236
 - constructors 238
 - example definitions 237
 - local variable scoping 259
 - modifiers 235
 - name 236
 - naming 234
 - overloading 237
 - parameter behavior 254
 - parameter list 236
 - passing arguments to 254
 - return types 236
 - signatures 237
 - using return values as arguments
260
 - methods rule 773
 - Microsoft Build 273
 - Microsoft Developer Network
(MSDN) 23, 114
 - Microsoft Enterprise Library
 - installation 584–585
 - support for application layers 580
 - Microsoft Enterprise Library Appli-
cation Blocks 580
 - Microsoft Intermediate Language
(MSIL) 22, 104, 105
 - Microsoft SQLServer Express Edi-
tion 580
 - Microsoft Visual C# Express 22
 - MinuteTick custom event example
378–380
 - model 50, 825
 - modeling 50
 - collateral roles 800
 - dominant roles 799
 - dynamic roles 800
 - model-view-controller 816
 - model-view-controller (MVC) 825
 - modularity 793
 - module
 - creating with compiler 131
 - definition 131
 - Monitor class
 - synchronizing thread access with
497
 - usage 497
 - monoalphabetic substitution 201
 - MSBuild 273, 606
 - <Csc> task 609
 - <ItemGroup> tag 608
 - <Project> tag 608
 - <PropertyGroup> tag 608
 - <Target> tag 609
 - compiling value object target 613
 - default target 609
 - items
 - referencing 608
 - project file
 - example 607
 - properties
 - referencing 608
 - targets
 - defining 608
 - tasks 608
 - using to manage and build proj-
ect 606

- MSDN 23, 114
 - MSIL Disassembler 105
 - multilayer projects
 - recommended approach 609
 - multilayered 528
 - multilayered database application
 - design 580
 - multilayered database applications 580–687
 - multithreaded programming 448
 - multithreaded server 564
 - multithreaded server application 533
 - multithreaded TCP/IP server 564–565
 - multithreaded vacation 448
 - multitiered applications 528, 534
 - MVC 825, 827
 - simple example of 825
- N**
- namespaces 7
 - naming conventions
 - for custom events 388
 - nested type 232
 - nested type declarations 232
 - network
 - definition 528
 - homogeneous vs. heterogeneous 529
 - purpose 528
 - network application
 - layers 532
 - physical deployment 532
 - tiers 532
 - network applications 528
 - network clients
 - running multiple on same machine 533
 - network layer 536
 - network stream
 - flushing after writing serialized object 570
 - network streams
 - StreamWriter.Flush() method 562
 - StreamWriter.WriteLine() method 562
 - networking 528
 - networking protocols
 - role of 528
 - NetworkStream 570
 - NonSerialized 726
 - non-static 726
 - NotePad++ 23
- noun 51
 - noun lists
 - suggesting possible application objects 51
 - nouns 51
 - mapping to data structures 51
 - numeric formatting 211
- O**
- Object 340
 - object
 - cloning 738
 - their associated type 299
 - object attributes 51
 - object behavior
 - comparison/ordering 723, 747
 - copy/assignment 723, 733
 - defined 722
 - equality 723, 741
 - fundamental 722, 724
 - object collaboration 272
 - object creation
 - with System.Activator.GetObject() method 549
 - object equality 722
 - object usage scenario evaluation checklist 723
 - Object.Equals() method
 - rules for overriding 741
 - Object.GetHashCode() method
 - general contract 742
 - object-oriented analysis 792
 - object-oriented architecture
 - extending 759
 - preferred characteristics 758
 - reasoning about 758
 - understanding 758
 - object-oriented design approach 9
 - object-oriented programming 220
 - object-oriented programming enablers 792
 - object-oriented programming patterns 357
 - objects
 - operations upon 299
 - value vs. reference assignment 734
 - well-behaved 722
 - obsolete Thread methods 456
 - OCP 775
 - defined 775
 - example 775
 - octets 537
- OnDeserialized 726
 - OnDeserializing 726
 - OnSerialized 726
 - OnSerializing 726
 - open address hash table 405
 - open-closed principle 774
 - achieving 774
 - operands 144
 - operating system
 - file management services 480
 - operator associativity 144
 - operator overloading 231, 694–718
 - assignment operators 717
 - binary * / operators 704
 - binary + - operators 702
 - binary operators 702
 - bitwise & | operators 707
 - comparison operators 710
 - implicit and explicit cast 713
 - in the context of your design 694
 - purpose for 694
 - table of overloadable operators 694
 - true false operators 698
 - unary - operator 696
 - unary ! operator 697
 - unary + operator 696
 - unary ++ -- operators 700
 - unary operators 695
 - operator precedence 144
 - operator semantics 694
 - operators 142–155, 231
 - additive 146
 - assignment 154
 - conditional AND 152
 - conditional OR 152
 - equality 148
 - logical AND 150
 - logical OR 150
 - logical XOR 150
 - modulus 146
 - multiplicative 146
 - overloading 694
 - primary 144
 - relational 148
 - shift 147
 - ternary 153
 - type testing 148
 - unary 145
 - OptionalField attribute 726
 - origin 347
 - overloaded operators
 - leading to cleaner code 694
 - overloading 231

- override
 - keyword used to override base class methods 310
 - overriding
 - base class methods
 - enabling with virtual keyword 310
 - overriding Object.GetHashCode()
 - checklist 742
- P**
- packet 529
 - packet-switched network 536
 - parameter 131
 - parameter arrays
 - example 259
 - ParameterizedThreadStart delegate
 - 457
 - used in multithreaded server 565
 - parameters
 - behavior of reference types 255
 - behavior of value-types 255
 - how arguments are passed to methods 255
 - out parameter modifier 257
 - parameter arrays 259
 - passing ref arguments 254
 - ref keyword 254
 - params keyword 259
 - part objects 274, 275
 - Pascal case 879
 - pascal case 230
 - pass by reference 254
 - pass by value 254
 - PATH 22
 - path
 - absolute 481
 - definition 481
 - relative 481
 - Path class 481
 - patterns
 - command 816
 - façade 816
 - factory 799, 816
 - MVC 816
 - singleton 799, 816
 - peer-to-peer collaboration 274
 - pen 51
 - Peter Coad 797
 - physical complexity 16, 272
 - physical layer 536
 - Point structure 350
 - using to place components 350
 - polymorphic behavior
 - example of 310
 - polymorphic containment 801
 - polymorphic substitution 799
 - polymorphism 792
 - applied 800
 - defined 319, 800
 - goal of programming with 800
 - planning for proper use of 800
 - port 548
 - postcondition 763
 - defined 761
 - postconditions 760, 761
 - changing in derived class methods 769
 - PowerShell
 - Set-ExecutionPolicy command 584
 - precondition 253, 763
 - defined 761
 - preconditions 760, 761
 - changing preconditions of derived class methods 765
 - weakening 766
 - predefined types 136
 - preempted 450
 - PreFilterMessage() method 344
 - prepared statements 618
 - primary key 589
 - automatically incrementing integer 601
 - private 300, 318
 - problem abstraction 10, 220
 - and the development cycle 221
 - end result of 221
 - mantra 220
 - performing problem analysis 221
 - process of 220
 - problem domain 9, 46, 50, 56, 873
 - procedural-based design approach 9
 - process 448
 - definition 449, 450
 - multithreaded
 - definition 450
 - single-threaded
 - definition 450
 - processing cycle 100
 - decode 100
 - execute 100
 - fetch 100
 - store 100, 101
 - processor
 - block diagram 94
 - CISC 94
 - machine code 96
 - RISC 94
 - processors 93
 - production coders vs. design theorists 793
 - program
 - computer perspective 99
 - definition of 98
 - human perspective 99
 - two views of 98
 - what is a C# 130
 - program control flow statements 160
 - programming 4
 - challenges & frustrations 4
 - skills required 4
 - programming as art 4
 - programming cycle 14
 - code 14
 - integrate 14
 - plan 14
 - refactor 14
 - repeating 14
 - summarized 14
 - test 14
 - programs 92
 - why they crash 101
 - project approach strategy 8
 - application requirements 8
 - design 9
 - in a nutshell 10
 - language features 9
 - problem domain 9
 - strategy areas 8
 - project complexity
 - managing 15
 - project folder
 - creating 27
 - project objectives 50
 - project requirements 8, 56
 - project specification 51
 - properties 229
 - creating a calculated property 242
 - example 240
 - get accessors 229
 - instance 229
 - read-only 229
 - read-write 229
 - set accessors 229
 - static 229
 - properties rule 774
 - protected 300, 304, 318
 - protected block 430
 - protected code 435

protected internal 233, 300, 304, 318
 protocol stack 536
 proxy
 used by remoting client 547
 pseudocode 65, 66
 public 131
 public interface 233
 publisher
 responsibilities 376

Q

quality without a name 816
 queue 406
 FIFO characteristic 406
 QWAN 816

R

ragged array 206
 Ralph Johnson 817
 random access file I/O 494–513
 calculating fixed-length record
 count 494
 random access file operations 494
 RDBMS 588
 readonly fields 226
 readonly instance fields 226
 readonly static fields 226
 readonly vs. const fields 228
 realization 315
 expanded form 315
 expressing in UML 315
 lollipop diagram 315
 simple form 315
 record 604
 record locking 497
 rectangular arrays 203
 recursion
 example 384
 red-black binary tree 400
 refactor 299
 refactoring a design 299
 reference equality vs. value equality
 741
 reference parameters 254
 reference semantics 261
 reference to object combinations 304
 reference types 136
 referential integrity 589
 regression testing
 example 65
 relational database 580, 588
 relational database management sys-

tem 588
 relationships
 between database tables 589
 reliable object-oriented software
 creating 759
 remotable object 546
 how to create 547
 remote object
 creating for multitiered applica-
 tion 648
 Remoting exception
 problem sending bitmap across
 application domains 661
 remoting infrastructure 546, 549
 requirements 8, 46, 873
 gaining insight through pictures
 51
 requirements gathering 8
 resource sharing 528
 ResumeLayout() method
 purpose of 359
 reusability 793
 Richard Helm 817
 Robert's Rules of Order 528
 robot rat project specification 48
 analyzing 49
 root directory
 definition 481
 routing tables 538
 rows 588

S

Sandcastle 88
 screen coordinates 345, 347
 ScrollableControl 340
 sealed class 319
 sealed method 319
 segments 537
 select command 596
 selection statements 160
 self-commenting code
 writing 879
 semantics
 pre- and postfix increment and
 decrement operators 702
 value vs. reference 261
 sentinel variable 60
 sequence diagram 272
 serializable attribute 484
 serialization
 custom 726, 729
 object 483
 serializing

List<People> to NetworkStream
 570
 objects
 as XML 487
 serializing objects 483–489
 steps to 485
 server 528, 531
 application 528, 531
 hardware 528, 531
 multithreaded 533
 treated as capital equipment 531
 server application 546
 service 449
 shallow copy 722
 defined 734
 shallow vs. deep copy 734
 Short circuiting AND 143
 Short circuiting OR 143
 short-circuiting logical operators 152
 shortcut
 creating 29
 modifying properties 29
 modifying start-up folder 30
 signature
 method 231
 signature rule 773
 simple aggregation
 defined 274
 simple vs. composite aggregation 274
 simplification
 of real-world problems 220
 SingleCall 548, 550
 single-threaded vacation 448
 Singleton 548, 550
 singleton 816
 socket 561
 software design 223
 software design patterns 816
 abstract factory 822
 background 816
 command 827
 definition 816
 dynamic factory 822
 factory 822
 factory method 822
 Singleton 818
 specification template 817
 Software Development Kit (SDK)
 105
 software development roles 7
 analyst 7
 architect 7
 programmer 8
 sorting

- arrays with Array class 211
 - collections 418, 747
 - source code
 - file header 80, 874
 - formatting 80, 874
 - specialization
 - expressing in UML 300
 - SplitContainer
 - example code 517
 - SQL 588–605
 - AND operator 604
 - commands
 - alter 589
 - create 589, 590
 - delete 595
 - drop 589
 - insert 595
 - select 595, 596
 - update 595
 - use 589
 - constraint
 - definition of 592
 - creating tables 592
 - Data Control Language 589
 - Data Definition Language 589
 - Data Manipulation Language 589, 595
 - database script
 - dropping and creating tables with 591
 - database scripts
 - using 591
 - executing commands with go 591
 - from clause 597
 - inner join 603
 - join operation 603
 - order by clause
 - example 604
 - prepared statements 618
 - three sub languages 589
 - where clause 597
 - SQL command parameter strings
 - difference from verbatim strings 618
 - SQL command parameters 618
 - SQL command parameters and prepared statements
 - generalized steps 618
 - SQL command utility
 - use of 590
 - W switch 604
 - SQL query string constants 618
 - SQL Server
 - changing to master database 591
 - data types 593
 - four default databases 590
 - identity operator 601
 - newid() function 603
 - use of 597
 - SQL Server Management Studio 601
 - installation 583
 - SQLServer Express
 - installation 581–583
 - stack 406
 - LIFO characteristic 406
 - state transition diagrams 66
 - statement
 - for
 - personality of 174
 - statements 140–155
 - break 177
 - chained if/else 166
 - continue 177, 178
 - control flow 160
 - do/while 172
 - personality of 172
 - eighteen kinds of 141
 - empty 140
 - executing consecutive if 164
 - for 173
 - relationship to while 173
 - goto 179
 - if 160
 - if/else 160, 164
 - iteration 170
 - nesting 174
 - mixing selection and iteration 175
 - nineteen kinds of 141
 - selection statements 160
 - switch 160, 167
 - condition expression types 167
 - nested 169
 - using break in 167
 - table of 180
 - try/catch 162
 - while 171
 - personality of 171
 - state-transition diagrams 65
 - static 134, 726
 - static constructor 226
 - static constructors 231
 - strategy
 - project approach 8
 - StreamReader class 489
 - StreamReaders
 - use in network programming 528
 - StreamWriter class 487, 489
 - strengthening precondition 768
 - strengthening preconditions 767
 - String
 - array of 199
 - string 134
 - string characters
 - accessed using array notation 202
 - string formatting 211
 - structs
 - advice on when to use 262
 - authorized members 261
 - behavior during assignment 261
 - behavior of this 262
 - boxing and unboxing 262
 - default field values 261
 - Structured Query Language 588–605
 - structures
 - structures vs. classes 260
 - stubbing 15
 - subdirectory 481
 - subfolder 481
 - subject matter experts 9
 - subscriber
 - responsibilities 376
 - subscriber notification process 377
 - supertypes & subtypes
 - reasoning about 759
 - SuspendLayout() method
 - purpose of 359
 - switch
 - implicit case fall-through 168
 - switch statement 167
 - system message queue 343
 - System namespace
 - exploring 114
 - System.Activator.GetObject()
 - example code 549
 - System.Collections 407
 - System.Collections.Generic 407
 - System.Collections.ObjectModel 408
 - System.Collections.Specialized 408
 - System.Diagnostics namespace 761
 - System.Drawing.Point 351
 - System.Drawing.Rectangle 351
 - System.EventHandler delegate 354
 - System.Guid
 - use of as primary key 613
 - System.ValueType class
 - direct base class for all value types 138
 - SystemException 432
- T**
- table 588

- TableLayoutPanel 339, 361
 - adding multidimensional array
 - of controls to 361
 - properties
 - ColumnCount 361
 - RowCount 361
- TCP 537
 - octet sequencing 537
- TCP/IP 528, 529, 536–539
 - application layer 537
 - data link layer 538
 - network layer 538
 - physical layer 538
 - transport layer 537
- TCP/IP client server programming 560–574
- TCP/IP client-server
 - binding TcpListener object to machine IP address and port 562
 - calling TcpListener.AcceptTcpClient() method 562
 - calling TcpListener.Start() method 562
 - connection process illustrated 560
 - listening on multiple IP addresses 565
 - multithreaded server
 - building 564
 - serializing complex objects between 569
 - simple example code 561
- TcpChannel 547, 548, 549
- TcpClient 560
- TcpListener 560
- TELNET 537
- test data
 - inserting into database with script 595
- test driver program 241
- testing 242
 - user-defined type 241
- text files
 - delimiter 490
 - issues to consider before creating 490
 - procedure to read 491
 - reading and writing 489–492
- Text property
 - effects on different controls 351
- TextBox 339
 - multiline
 - selecting line of text by double-clicking 366
 - property
 - MultiLine 367
 - WrapContents 367
- TextWriter 488
- the art of programming 4, 11
 - inspiration 11
 - money but no time 12
 - mood setting 12
 - time but no money 12
 - where not to start 11
 - your computer 12
- thinking outside the box 220
- this()
 - called from constructor 250
- thread
 - execution context 450
- thread context 450
- thread queue 450
- ThreadPool 448
- ThreadPool class 468
 - number of default worker threads 468
 - starting threads with 469
- threads 448–476
 - asynchronous method calls 470
 - BackgroundWorker
 - events 464
 - BackgroundWorker class 464
 - blocking with Thread.Join() 459
 - blocking with Thread.Sleep() 458
 - creating managed threads 452
 - executing on single-processor system 450
 - foreground vs. background 461
 - ParameterizedThreadStart delegate 457
 - passing ThreadStart delegate to Thread constructor 456
 - preempted 450
 - running asynchronous methods with delegates 470
 - setting Thread.IsBackground property 462
 - starting managed threads 456
 - thread state 455
 - ThreadPool class 468
 - ThreadStart delegate 456
 - time-slicing 450
- threads of execution 450
- ThreadStart delegate 455
- throwing an exception 430
- timeless way 816
- time-slicing 450

- TimeSpan
 - passing to Thread.Sleep() method 459
- TimeSpan structure
 - example use of 360
- title bar
 - window 341
- ToolStripMenuItem
 - constructor usage 364
- ToolStripMenuItems
 - declaring and creating 364
- transitivity
 - exhibited by inheritance hierarchies 299
- Transmission Control Protocol (TCP) 537
 - transport layer 536
- tree command 40
- try/catch statement 162
- type 299
 - diagram 136
 - value type
 - behavior 137
- type coercion 308
- types 136–140
 - array 190
 - predefined 136
 - mapping to system namespace structures 138
 - reference 136
 - behavior 137
 - value 136
 - value range table 139

U

- UDP 528, 537
- UML 16, 220, 272, 291
 - class diagram 223
 - composite aggregation 276, 278
 - expressing abstract class 311
 - expressing inheritance 300
 - expressing interfaces 315
 - expressing realization 315
 - expression aggregation 275
 - realization
 - diagram
 - expanded form 316
 - simple form 316
 - sequence diagram
 - engine object creation 284
 - sequence diagrams 279, 280
 - simple aggregation 275, 276
 - stereotype 224
 - using to tame conceptual com-

- plexity 272
 - UML class diagram
 - purpose of 223
 - UML design tool
 - Magic Draw 281
 - UnhandledException 431
 - Unified Modeling Language (UML) 16
 - uniqueidentifier
 - use as primary key
 - example 592
 - unmanaged code 107
 - unreachable code 154
 - update command
 - SQL
 - commands
 - update 598
 - URI 528
 - URL 528
 - User Datagram Protocol (UDP) 537
 - user-defined types 221
 - using 134
 - using directive 131
 - utility methods
 - definition of 232
- V**
- value objects 580
 - value parameters 254
 - value semantics 261
 - value types 136
 - ValueType class 138
 - variable 51
 - definition 137
 - variable length records 493
 - verb phrases 51
 - verbatim string literals 482
 - Verbatim strings
 - formulation 483
 - verbs 51
 - vertical access 318, 725
 - view 825
 - virtual
 - keyword to allow method overriding 310
 - Virtual Execution System 105
 - Virtual Execution System (VES) 104, 107
 - virtual machine 104
 - and the common language infrastructure 104
 - virtual machines 104, 105
 - visible region
 - of a window 341
 - Visual C# Express Edition
 - building project 38
 - creating project with 36
 - creating projects with 35
 - locating project executable file after build 40
 - void 134
- W**
- well-behaved objects 722
 - WellKnownObjectMode.SingleCall mode 550
 - WellKnownObjectMode.Singleton mode 550, 551
 - whole object 274
 - whole objects 275
 - whole/part class relationship 274
 - Width property 351
 - window
 - basic functionality provided by 341
 - message categories 343
 - message prefixes 343
 - parts of 341
 - title bar 341
 - visible region 341
 - window application
 - execution thread 343
 - window coordinates 345, 347
 - window coordinates diagram 347
 - window message routing 343
 - window messages
 - trapping with IMessageFilter interface 344
 - window types
 - dialog boxes 340
 - floating 340
 - multiple-document interface (MDI) 340
 - standard 340
 - tool 340
 - windows
 - messages
 - WM_CHAR 344
 - WM_KEYDOWN 344
 - WM_KEYUP 344
 - WM_MOUSEMOVE 344
 - WM_MOUSEWHEEL 344
 - events
 - processing 341
 - executable
 - compiler switch 341
 - creating 341
 - Windows Task Manager
 - using to show applications and processes 449
 - word 97, 98
 - world
 - imperfect understanding of 793
- X**
- Xamarin 107
 - XML documentation
 - generating from command line 84
 - XML serialization 484
 - XmlSerializer 488
 - XmlSerializer class 487