

INDEX

Symbols

- ! 122
- 122
- 122
- != 125
- #define DEBUG directive 645
- #endregion directive 541
- #region directive 541
- % 123
- * 123
- + 122
- ++ 122
- += operator 304
- .NET Framework
 - downloading 21
 - installing 20–21
- .NET Remoting
 - Singleton mode 468
- .NET remoting 466–477
 - configuration files 472
 - network communication handled by 469
 - passing collection of Person objects between remote object and client 474
 - persisting remote object state 469
 - purpose of 466
 - registering channels 468
 - registering service name 468
 - registering well known service types 468
 - remote object access via interface 470
 - serializing complex objects 474
 - simple example 467
 - SingleCall mode 468
 - SingleCall vs. Singleton remote object modes 469
 - swapping remote objects
 - enabling with interfaces 470
 - three primary channels 467
 - three required components 466
- .NET Remoting Architecture 466
- .NET remoting infrastructure 495
- / 123
- /d
 - DEBUG compiler switch 646
- ; 119

- < 125
- << 124
- <= 125
- = 119
- == 125
- > 125
- >= 125
- ~ 122

A

- abstract
 - classes 267
 - methods 267
- abstract class 257
 - expressing in UML 268
 - purpose of 268
 - term defined 257
- abstract class vs. interface 270, 271
- abstract data types 191
- abstract keyword
 - using to declare classes and methods 269
- abstract methods
 - implementing in derived classes 269
- abstract thinking 9
- abstraction
 - problem 9
 - the art of programming 190
- abstractions
 - selecting the right kinds of 662
- access
 - horizontal 274
 - vertical 274
- Access Control Graph (ACG) 675
- access modifiers 201
 - default/package 201
 - most often used 274
 - private 201
 - protected 201
 - public 201
- address bus 81
- addressing local machine 455
- ADO.NET 494
- aggregation 234, 235, 237, 250
 - aggregate constructors 236
 - composite 236, 250
 - composite example code 239
 - definition 235
 - determining type by who controls object lifetime 236
 - effects of garbage collector 236
 - example
 - engine simulation 242
 - engine simulation class diagram 244
 - simple 236, 237, 250
 - simple example code 238
 - two types of 676
- algorithm
 - running time 85
 - understanding the concept of 76
 - working definition of 83
- algorithm growth rate 85
- algorithms 76, 83
 - good vs. bad 83
- alter statement
 - used to create foreign key constraint 511
- analysis 48, 668
- Ansel Adams 668
- API Framework
 - blessing and curse 94
- API reference documentation
 - class general overview page 96
 - class member page 97
 - obsolete APIs 102
 - Syntax section 101
- API reference information
 - definitive source 94
- application
 - definition 111
 - graceful recovery 46
 - layers 454
 - physical deployment 454
 - physical tier distribution 456
 - simple
 - structure 111
 - tier responsibilities 456
 - tiers 454
- Application class
 - Run() method 293
 - use of to run GUI programs 293
- application distribution 454
 - across multiple computers 455
- application domain 384
- application layer 458

- application layers 495
 - application message loop 293
 - application tiers
 - logical 456
 - separation of concerns 456
 - ApplicationException 367
 - applications
 - multitiered 456
 - architectural diagram
 - multitiered database application 494
 - architecture
 - flexibility 669
 - modularity 669
 - reliability 669
 - stability 669
 - array 339, 342
 - creating with literal values 168
 - declaration syntax 163
 - definition of 162
 - difference between value type and reference type arrays 169
 - dynamic resizing
 - example code 339
 - elements 162
 - functionality provided by array types 164
 - homogeneous elements 162
 - Main() method String parameter 181
 - multidimensional 176, 179
 - of value types 166
 - properties of 165
 - references
 - calling Array class methods on 167
 - single dimensional 166
 - single dimensional in action 171
 - specifying length 163
 - specifying types 163
 - two dimensional
 - example program 179
 - type inheritance hierarchy 164
 - value type
 - memory arrangement 166
 - Array class 182
 - array initializer expression 178
 - array literal 168, 169
 - array of arrays 178
 - array processing 46
 - array-based collection
 - growing on insertion 339
 - arrays 162
 - rectangular 176
 - sorting with Array class 182
 - two-dimensional
 - processing 62
 - using to solve problems 162
 - Ashmore's hash code algorithm 631
 - assembly
 - definition 111
 - Assertion Failed dialog 646
 - association 235, 250
 - definition 235
 - associativity
 - operator 121
 - forcing 121
 - asynchronous method calls 400
 - asynchronous methods
 - EndInvoke() method 402
 - IAAsyncResult interface 403
 - obtaining results from 402
 - providing callback method to BeginInvoke() method 402
 - attribute candidates 47
 - attributes 45
 - automated water tank custom event example 326–331
 - auxiliary storage device 410
- B**
- BackgroundWorker 382
 - BackgroundWorker class 396
 - BackgroundWorker events 396
 - bad software architecture
 - characteristics of 661
 - base class
 - methods
 - overriding 266
 - source code example 259
 - Base Class Libraries (BCL) 100
 - BaseCommand class 698
 - BaseDAO
 - class definition
 - using DatabaseFactory class 523
 - behavior
 - generalized 256
 - behavior contract 615
 - Bertrand Meyer 656, 671
 - Bertrand Meyer's Design by Contract (DbC) 643
 - binary data 420–422
 - BinaryFormatter class 414, 486
 - BinaryReader class 420, 422, 423
 - BinaryWriter class 420, 422
 - bit 80, 81
 - Bitmap class 299
 - using to create Image object 300
 - Bloch's hash code algorithm 631
 - block 479
 - blocking I/O operation 479
 - Bounds
 - data that comprises 298
 - property
 - printing to screen 298
 - Bounds property 298
 - setting example 302
 - boxing 225
 - break 142
 - bridge 451
 - Business Layer 495
 - business object
 - definition 495
 - business objects 494, 495
 - business rules 495
 - creep 495
 - Button 291
 - byte 80, 81
- C**
- C# compile and execute process 86
 - cache memory 80
 - calling base class constructor with base() 260
 - camel case 199, 735
 - cascade delete 501
 - SQL
 - cascade delete
 - testing 515
 - casting 264, 351
 - advice on use of 265
 - chained hash table 345
 - character constants
 - declaring
 - example 55
 - using in switch statement 56
 - Christopher Alexander 688
 - class 111, 257
 - abstract 267
 - expressing in UML 268
 - purpose of 268
 - abstract class 257
 - four categories of members 194
 - non-static fields 195
 - sealed 274
 - static fields 195
 - term definition 257

- class declarations
 - viewed as behavior specifications 644
- class definition
 - adding fields 207
 - adding instance methods 208
 - constructor method 208
 - starting 207
- class invariant 644, 646
 - defined 644
- class invariants 644
- class member access
 - default when omitting access modifier 274
- classes
 - classes vs. structs 225
 - number in an application 234
- Class-Wide Fields 195
- Click event 303
- client 450, 453
 - application 450, 453
 - hardware 450, 453
- client application 466
- client coordinates 299
- client-server applications
 - See also TCP/IP client-server TCP/IP 478
 - with .NET remoting 466
- cloning objects 627
- CloseReader() method 523
- Coad's Inheritance Criteria 672
- code blocks
 - executing in if statements 139
- code library
 - creating 86
- code module
 - creating 86
- code reuse 668
- coding convention
 - adopting 735
- cohesion 15, 203
- collateral roles
 - modeling 674
- collections
 - ArrayList
 - usage example 340
 - casting 351
 - extending ArrayList 352
 - extracting elements into arrays 361
 - general characteristics 338
 - generic
 - example code 354–356
 - KeyedCollection<TKey, TItem> example 355
 - List<T> 354
 - IComparer<T, T> 359, 636
 - implementing IComparable<T> 357, 634
 - interfaces 338
 - linked list node elements 343
 - making an object sortable 357, 634
 - non-generic to generic mapping table 349
 - old-school style 350–353
 - old-school style programming 348
 - performance characteristics
 - arrays 342
 - hashtable 345
 - linked list 343
 - Person list example 351
 - red-black tree node elements 346
 - sorting 357
 - rules for implementing IComparable<T>.CompareTo() method 358, 635
 - specialized 349
 - underlying data structures 349
 - using foreach to iterate over example 351
- Color structure 299
- columns 501
- command console layout properties
 - modifying 28
- command pattern 688, 697
- CommandFactory class 699
- command-line arguments
 - processing 181
- command-line compiler 20
- command-line tools 20
 - why you should learn 20
- Common Language Infrastructure
 - four parts 87
- Common Language Infrastructure (CLI) 86, 87
- Common Language Runtime (CLR) 90
- Common Language Specification (CLS) 88
- Common Log File System 438
- Common Type System (CTS) 88
- compiler errors
 - dealing with 30
 - finding their meaning on MSDN 30
 - fixing 14
- compiling
 - simple application 111
- compiling multiple source files 234
- compiling source file
 - how to 29
- compiling with csc
 - using target switch example 215
- complex application behavior 234
- complex project folder organization 516
- complexity
 - conceptual 14, 234, 235, 250
 - managing physical 15
 - physical 15, 234, 235, 250
 - relationship between physical and conceptual 15
- Component 293
- components
 - adding to Controls collection 302
 - adding to windows 301
 - initializing in separate method 302
- composite aggregation
 - defined 236
- composition 668, 676
 - as force multiplier 676
- compositional design 234, 676
- compositionists 668
- computer
 - architecture
 - feature set 79
 - feature set accessibility 79
 - feature set implementation 79
 - three aspects of 79
 - definition of 76
 - memory
 - organization 79
 - processing cycle 82
 - system 76
 - components of 77
 - hard drive 77
 - keyboard 77
 - main logic board 77
 - memory 77, 80
 - monitor 77
 - mouse 77
 - processor 77
 - speakers 77
 - system unit 77
 - vs. computer system 76
- computer network
 - definition 450
 - purpose 450
- computer program
 - modeling real world problem 190

- computers 76
 - conceptual complexity 14, 234
 - managing 14
 - taming 14
 - concrete class 260
 - concurrently executing applications 384
 - condition
 - exception 366
 - configuration file
 - example 528
 - configuration files
 - .NET remoting 472
 - configuration-management tool 15
 - connection pooling 495
 - connection string
 - database
 - configuration file setting 499
 - console applications 110–131
 - console text color
 - changing
 - example code 483
 - console text menu
 - processing user commands
 - example 56
 - console text menus
 - example 53
 - const 197
 - constant 47, 195
 - constants 197
 - constraint
 - database 504
 - constructor methods 206
 - constructors 616
 - ContainerControl 293
 - containing aggregate 237
 - containment
 - by reference 236
 - by value 236
 - polymorphic 676
 - contains 237
 - continue 152
 - Control 293
 - control bus 81
 - controller 695
 - controls
 - dynamic layout of 308
 - registering event handler methods 303
 - Controls collection
 - use of 302
 - coordinates
 - client 297
 - origin 298
 - screen 297
 - (x,y) pairs 297
 - origin 298
 - pixel as basic unit of measure 297
 - window 297
 - window placement upon screen 297
 - copy constructor 624
 - coupling 15
 - create tables SQL script 504
 - creativity
 - and problem abstraction 190
 - cross platform
 - promise of 89
 - CRUD operations
 - database
 - CRUD operations 523
 - csc
 - compiling entire source directory 235
 - compiling multiple source files 234
 - csc compiler
 - locating 21
 - current position 48
 - custom event
 - recursive example 327–329
 - custom events 322
 - suggested naming convention 331
 - custom exceptions 374
 - custom serialization 618, 620
- D**
- DAO layer
 - building 519
 - Data Access Layer 495
 - data access object
 - definition 495
 - data access objects 494, 495
 - data base
 - key factor in business rules 495
 - data bus 81
 - Data Control Language 502
 - Data Definition Language 502
 - data link layer 458
 - Data Manipulation Language 502, 506
 - data type 47
 - reference 164
 - value 164
 - data types
 - array 164
 - SQL Server 505
 - database
 - automatically inserting primary key 511
 - cascade delete 501
 - columns 501
 - constraint
 - definition of 504
 - converting binary data into bit-map image 525
 - creating related table with script 511
 - DataBase.AddInParameter()
 - method 526
 - foreign key 501, 511
 - foreign key constraint
 - naming 511
 - inserting image data
 - example code 524
 - inserting test data into related table 512
 - inserting value objects into 526
 - join operation 511
 - primary key 501
 - record 514
 - referential integrity 501
 - rows 501
 - table 501
 - database application
 - compiling 500
 - database connection
 - established via DatabaseFactory 495
 - database connection string 499
 - database connection test application 499–500
 - database management system 501
 - Database object 499, 523
 - database script
 - running
 - example 504
 - DatabaseFactory 495, 523
 - configuration file 499
 - example code 499
 - DataBindingComplete event 580
 - dataConfiguration
 - configuration file section 499
 - datagrams 459
 - DataGridView 562, 564
 - clicking on row to yield row index 562
 - data binding 580
 - DataSource property 562
 - row index value 562

- DateTime structure
 - example of use 310
 - DateTime.Now 310
 - DbC 643
 - DbCommand 526
 - DBMS 501
 - DbType enumeration
 - .NET type mapping table 526
 - Debug.Assert() method 645
 - deep copy 614
 - defined 624
 - default class member access 274
 - default constructor 200
 - delay
 - example code 330
 - delegate 291, 322
 - event subscriber list 322
 - EventHandler 323
 - method signature specification 323
 - delegate object
 - purpose of 322
 - delegate type
 - purpose of 303
 - specification of method signature 304
 - delegates
 - EventHandler 303
 - MouseEventHandler 303
 - PaintEventHandler 303
 - running asynchronous methods with 400
 - delete command
 - SQL
 - commands
 - delete 510
 - delimiter
 - text file 418
 - Department of Defense 452
 - dependencies
 - managed 669
 - dependency 194, 235
 - definition 235
 - effects of dependency relationships between classes 235
 - dependency inversion principle 661
 - dependency relationship 250
 - dependency vs. association 235
 - deprecated members 201
 - derived class
 - source code example 260
 - deserialization
 - object 414
 - deserialize
 - object
 - from XML file 416
 - design 668
 - design by composition 234
 - Design by Contract 643
 - design pragmatists 668
 - development cycle 43
 - application 51
 - applying 43
 - code 43, 728
 - creating feature implementation lists 51
 - deploying 43
 - integrate 43
 - iterative application 51
 - plan 43, 728
 - refactor 43
 - test 43, 728
 - using 43
 - development environment
 - configuring 22
 - device driver 410
 - difference between abstract class and interface 270
 - difference between readonly and const fields 197
 - direct base class 194
 - direction 47
 - directory
 - definition 411
 - Directory class 411
 - example code 412
 - DirectoryInfo class 411
 - disk
 - driver software 410
 - distributed applications 450
 - DockStyle enumeration 309
 - values 309
 - documentation generation 72
 - dominant roles
 - modeling 674
 - Doxygen 72
 - Dr. Barbara Liskov 643
 - Dr. Bertrand Meyer 643
 - drive letters 411
 - driver
 - creating test code 209
 - dynamic class loading
 - example code 699
 - dynamic factory pattern
 - advantages of 695
 - dynamic link library 86
 - dynamic polymorphic behavior 656
 - DynamicArray
 - case study 338
- ## E
- ECMA - 335 87
 - effects of change
 - predicting 669
 - Eiffel 643
 - EmployeeDAO 495
 - empty statement 119
 - Encapsulation 9
 - encapsulation 201
 - EndInvoke() method 402
 - engineering trade-off 668
 - Enterprise Library Configuration tool 499
 - Enterprise Library Data Access Application Block 495
 - entry point 111
 - enumerated type 59
 - environment variable 20
 - environment variables 22–24
 - Erich Gamma 689
 - error checking 46
 - error conditions
 - program
 - handling 137
 - that cause exceptions
 - examples of 366
 - errors
 - compiler 14
 - Ethernet 459
 - event 322
 - event arguments
 - example code 324
 - event consumer 322
 - event driven programs 293
 - event handler
 - explicit call to
 - example 580
 - event handler methods
 - registering 303
 - event handlers
 - located in different objects 305
 - event producer 322
 - event subscriber list 322
 - events 200, 303
 - and their delegate types
 - table of 303
 - BackColorChanged 303
 - BackgroundImageChanged 303
 - Click 303
 - DoubleClick 303

- GotFocus 303
 - GUI
 - handling in separate object
 - example 307
 - handled in separate objects 305
 - MouseClicked 303
 - MouseDoubleClick 303
 - MouseDown 303
 - MouseEnter 303
 - MouseLeave 303
 - MouseMove 303
 - MouseUp 303
 - Paint 303
 - registering event handler method
 - example of 304
 - Exception
 - class hierarchy 367
 - public properties 370
 - exception
 - definition 366
 - exception information table 367
 - exceptions 366–376
 - catch block 366
 - catching multiple exceptions
 - rule of thumb 372
 - catching with try/catch block 138
 - CLR handling mechanism 366
 - custom 374
 - extending Exception class 374
 - using throw keyword 375
 - determining what a method may
 - throw 369
 - documenting 376
 - fault handler code 366
 - low-level to high-level translation 425
 - purpose of 366
 - runtime vs. application 368
 - translating low-level to high-level 375, 425
 - try block 366
 - try/catch/finally blocks 371–374
 - using multiple catch blocks 372
 - executing application
 - how to 30
 - executing SQL command
 - example code 499
 - extension inheritance
 - complications from using 675
 - vs. functional variation 675
- F**
- façade 688
 - factory 688
 - factory class
 - interfaces involved to employ 674
 - fault handler code 366
 - Fields 195
 - fields
 - readonly
 - initializing static readonly
 - fields in static constructor 195
 - readonly vs. const 197
 - file
 - definition 410
 - File class 411
 - file I/O 410–443
 - file position pointer 421, 422
 - File Transfer Protocol 458
 - FileDialogs
 - using 440–442
 - FileInfo class 411
 - example code 412
 - files
 - manipulating 411–413
 - FileStream class 414, 422
 - final project considerations
 - checklist 66
 - finalizers 200
 - First-In-First-Out (FIFO) 347
 - fixed-length records 422
 - reading
 - example code 429
 - floor 48
 - flow 11
 - achieving 12
 - concept of 11
 - stages 12
 - flow charts 59
 - FlowDirection enumeration
 - values 309
 - FlowLayoutPanel 291, 308
 - properties
 - AutoSize 309
 - AutoSizeMode 309
 - Dock 309
 - FlowDirection 309
 - WrapContents 309
 - purpose of 308
 - folder 411
 - folder options
 - setting 25
 - foreign key 501, 511
 - foreign key constraint 511, 512
 - Form 291, 292, 294
 - class inheritance hierarchy 292
 - properties
 - BackColor 299
 - BackgroundImage 299
 - manipulating 299
 - simple form program 293
 - Text property 293
 - window types created with 292
 - formatting
 - numeric strings
 - table 183
 - source code 66, 728
 - from clause
 - use to join tables
 - SQL
 - from clause
 - use to join tables 514
 - functional decomposition 8
 - fundamental language features 46
- G**
- gate 688
 - gateway 451
 - generalization
 - expressing in UML 258
 - generalized behavior
 - specifying 256
 - GetRegisteredWellKnowClientTypes() method 473
 - good design
 - goals of 669
 - good software architecture
 - characteristics of 662
 - goto 153
 - graphical user interface programming 292–318
 - guarded region
 - of try block 138
 - GUI
 - coding rhythm 317
 - data input dialog design 569
 - loading image in PictureBox
 - example code 528
 - opening image file with OpenFileDialog
 - example code 528
 - separating code from event handlers 305–307
 - using dialogs to enter data 569
 - GUI layout
 - using mock-up sketch to design 559

guillemet characters 194

H

hard disk 410
 hardest thing about learning to program 4
 has a 237
 hash code
 algorithm 631
 hash function 345
 hash table 342
 chained 345
 open address 345
 slot probe function 345
 Height property 302
 homogeneous data types 162
 horizontal access 201, 274, 616
 host 453
 HttpChannel 467
 Hypertext Transfer Protocol 458

I

ICloneable 627
 IDataReader 526
 IDE 20
 identifier 114
 class name examples 735
 constant name examples 736
 method name examples 736
 naming 114, 735
 variable name examples 736
 identifiers 115
 forming 114
 if/else statement 140
 Image
 converting to byte array 526
 image
 using to set Form Background-Image property 300
 Image class 299
 Image data
 storing and transferring as byte array 564
 IMessageFilter
 implementation example 296
 implementation approach 51
 implicit cast 352
 indexer
 example code 339
 indexes 200
 IndexOutOfRangeException

 handling 57
 infinite loop 146
 inheritance 668, 670–673
 first purpose of 256
 good reasons for using 670
 Meyer’s Taxonomy 671
 object-oriented programming with 256
 second purpose of 257
 simple example 259
 third purpose of 257
 three purposes of 256
 valid usage checkpoints 672
 inheritance form
 constant 672
 extension 671
 facility 672
 functional variation 672
 implementation 672
 machine 672
 model 671
 reification 672
 restriction 671
 software 672
 structure 672
 subtype 671
 type variation 672
 uneffecting inheritance 672
 variation 672
 view 672
 inheritance hierarchy
 assessing with Coad’s criteria 673
 navigating 101
 inheritists 668
 inner join 514
 instance constructors 199
 integral type size
 be aware of 123
 integrated development environment 20
 interface 257
 authorized members 257, 270
 purpose of 270
 reducing dependencies with 674
 role of 674
 term definition 257
 interface members
 mapping to abstract members 275
 interfaces 668
 expressing in UML 271
 Intermediate Language (IL) 86
 internal 201, 258, 261, 274

Internet Protocol (IP) 459
 Internet protocol layers 457
 Internet Protocols 452
 inter-process communication 467
 IP 459
 IP address
 parsing with IPAddress.Parse() method 484
 IP addresses 459
 IPAddress.Parse() method 484
 IpcChannel 467
 purpose of 467
 is a relationship
 implementing 257
 iteration
 development 43
 iterative development 43

J

John Vlissides 689
 join operation 511
 Just-In-Time (JIT) compiler 86

K

keyword
 using as identifier example 114
 keywords
 reserved listing 113

L

Label 291
 language features 42, 51, 727
 language-features strategy area 48
 Last-In-First-Out (LIFO) 347
 layout managers 307–312
 legacy datafile adapter 422
 library
 creating with compiler example 467
 referencing with compiler switch example 468
 linked list 342
 Liskov Substitution Principle
 relationship to Meyer Design by Contract Programming 643
 three rules of 654
 Liskov Substitution Principle (LSP) 643
 List<T>

- example code 341
 - Local Area Network 450
 - localhost 455
 - Location property 302
 - lock keyword 424
 - compared to Monitor.Wait()/Monitor.Exit() 424
 - log files 438–440
 - loops 145
 - LSP 643
 - LSP & DbC
 - C# support for 643
 - common goals 643
 - designing with 644
- M**
- machine code 79, 86
 - Magic Draw UML Design Tool 241
 - Main method 110
 - main method
 - purpose 112
 - signatures 112
 - managed code 89
 - managed threads 385
 - MarshalByRefObject 293
 - use to create remotable object 466
 - marshaling
 - remoting method calls 467
 - MemberwiseClone() 627
 - memory
 - address bus 81
 - alignment 81
 - bit 80, 81
 - byte 80, 81
 - cache 80
 - control bus 81
 - data bus 81
 - hierarchy 80
 - non-volatile 80
 - organization 79
 - RAM 80
 - ROM 80
 - volatile 80
 - word 80, 81
 - menu 47, 559
 - menus 312–315
 - adding submenu items to menus 313
 - item naming conventions 313
 - menu item separator
 - adding 313
 - menuitems
 - registering event handlers with 313
 - MenuStrip
 - docking to window 313
 - importance of adding last 313
 - MenuStrip class 312
 - ToolStripMenuItem 312
 - MenuStrip 312
 - declaring and creating 313
 - message categories 295
 - message filters
 - adding 296
 - message loop
 - window 294
 - message pump 294
 - message queue 294
 - message routing
 - windows 294
 - messages
 - system
 - how they are generated 294
 - Metadata 88
 - method
 - cohesion 203
 - definition structure 203
 - parameter list 111
 - sealed 274
 - signature
 - definition 112
 - method stubbing 13
 - methods 46, 199, 202
 - abstract 267
 - body 205
 - constructors 206
 - example definitions 205
 - local variable scoping 224
 - modifiers 203
 - name 205
 - naming 203
 - overloading 206
 - parameter behavior 219
 - parameter list 205
 - passing arguments to 219
 - return types 204
 - signatures 206
 - using return values as arguments 224
 - methods rule 655
 - Microsoft Build 235
 - Microsoft Developer Network (MSDN) 20, 94
 - Microsoft Enterprise Library
 - installation 498
 - support for application layers 495
 - Microsoft Enterprise Library Application Blocks 494
 - Microsoft Intermediate Language (MSIL) 87
 - Microsoft SQLServer Express Edition 494
 - Microsoft Visual C# Express 20
 - MinuteTick custom event example 323–325
 - model 45, 695
 - modeling 45
 - collateral roles 675
 - dominant roles 674
 - dynamic roles 675
 - model-view-controller 688
 - model-view-controller (MVC) 695
 - module
 - creating with compiler 111
 - definition 111
 - monalphabetic substitution 173
 - Monitor class
 - synchronizing thread access with 424
 - usage 424
 - MSBuild 235, 516
 - <Csc> task 518
 - <ItemGroup> tag 518
 - <project> tag 518
 - <PropertyGroup> tag 518
 - <Target> tag 518
 - compiling value object target 522
 - default target 518
 - items
 - referencing 518
 - project file
 - example 517
 - properties
 - referencing 518
 - targets
 - defining 518
 - using to manage and build project 517
 - MSDN 20, 94
 - MSIL Disassembler 87
 - multithreaded programming 382
 - multithreaded server 480
 - multithreaded server application 454
 - multithreaded TCP/IP server 480–482
 - multithreaded vacation 382
 - multi-tier projects
 - recommended approach 519
 - multitiered applications 450, 456
 - multitiered database application design 494

multitiered database applications
494–583

MVC 695, 697

Controller

using factory pattern 698

simple example of 696

N

namespaces 7

naming conventions

for custom events 331

nested type 200

nested type declarations 200

network

definition 450

homogeneous vs. heterogeneous
451

purpose 450

network application

layers 454

physical deployment 454

tiers 454

network applications 450

network clients

running multiple on same ma-
chine 454

network layer 458

network stream

flushing after writing serialized
object 486

network streams

StreamWriter.Flush() method
480

StreamWriter.WriteLine() meth-
od 480

networking 450

networking protocols

role of 451

NetworkStream 486

NonSerialized 618

NotePad++ 22

noun 47

noun lists

suggesting possible application
objects 46

nouns 46, 47

mapping to data structures 47

numeric formatting 183

O

Object 293

object

cloning 627

their associated type 257

object attributes 46

object behavior

comparison/ordering 615, 633

copy/assignment 614, 623

defined 614

equality 615, 629

fundamental 614, 616

object creation

with System.Activator.GetOb-
ject() method 469

object equality 614

object usage scenario evaluation

checklist 615

Object.Equals() method

rules for overriding 629

Object.GetHashCode() method

general contract 630

object-oriented analysis 668

object-oriented architecture

extending 642

preferred characteristics 642

reasoning about 642

understanding 642

object-oriented design approach 9

object-oriented programming 190

object-oriented programming en-
ablers 668

object-oriented programming pat-
terns 307

objects

operations upon 257

value vs. reference assignment
624

well-behaved 614

obsolete Thread methods 389

obsolete Thread methods 389

obsolete Thread methods 389

obsolete Thread methods 389

obsolete Thread methods 389

obsolete Thread methods 389

obsolete Thread methods 389

obsolete Thread methods 389

obsolete Thread methods 389

obsolete Thread methods 389

obsolete Thread methods 389

obsolete Thread methods 389

obsolete Thread methods 389

obsolete Thread methods 389

obsolete Thread methods 389

obsolete Thread methods 389

obsolete Thread methods 389

obsolete Thread methods 389

obsolete Thread methods 389

obsolete Thread methods 389

obsolete Thread methods 389

obsolete Thread methods 389

obsolete Thread methods 389

obsolete Thread methods 389

obsolete Thread methods 389

obsolete Thread methods 389

binary * / operators 599

binary + - operators 597

binary operators 597

bitwise & | operators 601

comparison operators 603

implicit and explicit cast 607

in the context of your design 590

purpose for 590

table of overloadable operators

590

true false operators 593

unary - operator 591

unary ! operator 592

unary + operator 591

unary ++ -- operators 595

unary operators 591

operator precedence 121

operator semantics 590

operators 120–131, 200

additive 124

assignment 130

conditional AND 129

conditional OR 129

equality 125

logical AND 126

logical OR 126

logical XOR 126

modulus 123

multiplicative 123

overloading 590

primary 121, 122

relational 125

shift 124

ternary 129

type testing 125

unary 122

OptionalField attribute 618

origin 298

overloaded operators

leading to cleaner code 590

overloading 199

override

keyword used to override base

class methods 267

overriding

base class methods

enabling with virtual keyword

266

overriding Object.GetHashCode()

checklist 630

P

packet 452

- packet-switched network 457
 - parameter 111
 - parameter arrays
 - example 223
 - ParameterizedThreadStart delegate 390
 - used in multithreaded server 482
 - parameters
 - behavior of reference types 220
 - behavior of value-types 220
 - how arguments are passed to methods 220
 - out parameter modifier 223
 - parameter arrays 223
 - passing ref arguments 219
 - ref keyword 219
 - params keyword 223
 - part objects 236
 - pass by reference 219
 - pass by value 219
 - PATH 20
 - path
 - absolute 411
 - definition 411
 - relative 411
 - Path class 411
 - patterns
 - command 688
 - façade 688
 - factory 674, 688
 - MVC 688
 - singleton 674, 688
 - pen 47
 - Peter Coad 672
 - physical complexity 15, 234
 - physical layer 458
 - Point structure 301
 - using to place components 301
 - polymorphic behavior
 - example of 267
 - polymorphic containment 676
 - polymorphic substitution 674
 - polymorphism 668
 - applied 675
 - defined 275, 675
 - goal of programming with 675
 - planning for proper use of 675
 - port 468
 - postcondition 646
 - defined 645
 - postconditions 644
 - changing in derived class methods 652
 - precondition 218, 646
 - defined 644
 - preconditions 644
 - changing preconditions of derived class methods 648
 - weakening 648
 - predefined types 115
 - preempted 384
 - PreFilterMessage() method 296
 - prepared statements 526
 - primary key 501
 - automatically incrementing integer 511
 - private 258, 274
 - problem abstraction 9, 190
 - and the development cycle 191
 - end result of 191
 - mantra 190
 - performing problem analysis 191
 - process of 190
 - problem domain 8, 42, 46, 51, 727
 - procedural-based design approach 8
 - process 382
 - definition 383, 384
 - multithreaded
 - definition 384
 - single-threaded
 - definition 384
 - processing cycle 82
 - decode 82, 83
 - execute 82, 83
 - fetch 82, 83
 - store 82, 83
 - processor
 - block diagram 78
 - CISC 78
 - machine code 79
 - RISC 78
 - production coders vs. design theorists 669
 - program
 - computer perspective 82
 - definition of 82
 - human perspective 82
 - two views of 82
 - what is a C# 110
 - program control flow statements 136
 - programming 4
 - challenges & frustrations 4
 - skills required 4
 - programming as art 4
 - programming cycle 12
 - code 12
 - integrate 12
 - plan 12
 - refactor 13
 - repeating 13
 - summarized 13
 - test 12
 - programs 76
 - why they crash 83
 - project approach strategy 7
 - application requirements 8
 - design 8
 - in a nutshell 10
 - language features 8
 - problem domain 8
 - strategy areas 8
 - project complexity
 - managing 14
 - project folder
 - creating 25
 - project objectives 45
 - project requirements 8, 51
 - project specification 47
 - properties 198
 - creating a calculated property 210
 - example 208
 - get accessors 198
 - instance 198
 - read-only 198
 - read-write 198
 - set accessors 198
 - static 198
 - properties rule 655
 - protected 258, 261, 274
 - protected block 366
 - protected code 371
 - protected internal 201, 258, 261, 274
 - protocol stack 457
 - proxy
 - used by remoting client 467
 - pseudocode 59, 60
 - public 111
 - public interface 201
 - publisher 322
 - responsibilities 322
- ## Q
- quality without a name 688
 - queue 347
 - FIFO characteristic 347
 - QWAN 688
- ## R
- ragged array 178

- Ralph Johnson 689
 - random access file I/O 422–437
 - calculating fixed-length record count 422
 - RDBMS 501
 - Readonly Fields 195
 - readonly instance fields 195
 - readonly static fields 195
 - readonly vs. const fields 197
 - realization 271
 - expanded form 271
 - expressing in UML 271
 - lollipop diagram 271
 - simple form 271
 - record 514
 - record locking 424
 - Rectangle structure 301
 - rectangular arrays 176
 - recursion
 - example 329
 - red-black binary tree 342
 - refactor 257
 - refactoring a design 257
 - reference equality vs. value equality 629
 - reference parameters 219
 - reference semantics 225
 - reference to object combinations 261
 - reference types 115
 - referential integrity 501
 - regression testing
 - example 58
 - relational database 494, 501
 - relational database management system 501
 - relationships
 - between database tables 501
 - reliable object-oriented software
 - creating 643
 - remotable object 466
 - how to create 466
 - remote object
 - creating for multitiered application 551
 - Remoting exception
 - problem sending bitmap across application domains 562
 - remoting infrastructure 466, 469
 - requirements 8, 42, 727
 - gaining insight through pictures 47
 - requirements gathering 8
 - resource sharing 450
 - modifying start-up folder 27
 - signature
 - method 199
 - signature rule 655
 - simple aggregation
 - defined 236
 - simple vs. composite aggregation 236
 - simplification
 - of real-world problems 190
 - SingleCall 468, 469
 - single-threaded vacation 382
 - Singleton 468, 469
 - singleton 688
 - socket 478
 - software design 192
 - software design patterns 688
 - abstract factory 693
 - background 688
 - command 697
 - definition 688
 - dynamic factory 693
 - factory 693
 - factory method 693
 - Singleton 690
 - specification template 689
 - Software Development Kit (SDK) 87
 - software development roles 6
 - analyst 6
 - architect 7
 - programmer 7
 - sorting
 - arrays with Array class 182
 - collections 357, 634
 - source code
 - file header 66, 728
 - formatting 66, 728
 - specialization
 - expressing in UML 258
 - SplitContainer
 - example code 441
 - SQL 501–515
 - AND operator 514
 - commands
 - alter 502
 - create 502
 - delete 506
 - drop 502
 - insert 506
 - select 506, 507
 - update 506
 - use 502
 - constraint
 - definition of 504
 - creating tables 504
- ResumeLayout() method
 - purpose of 309
 - Richard Helm 689
 - Robert’s Rules of Order 451
 - robot rat project specification 44
 - analyzing 45
 - root directory
 - definition 411
 - routing tables 459
 - rows 501
- S**
- screen coordinates 297, 299
 - ScrollableControl 293
 - sealed class 274
 - sealed method 274
 - segments 458
 - select command 507
 - selection statements 136
 - self-commenting code
 - writing 735
 - semantics
 - pre and postfix increment and decrement operators 596
 - value vs. reference 225
 - sensor
 - multimode
 - example 327
 - serializable attribute 413
 - serialization
 - custom 618, 620
 - object 413
 - serializing
 - List<People> to NetworkStream 486
 - objects
 - as XML 416
 - serializing objects 413–418
 - steps to 414
 - server 450, 453
 - application 450, 453
 - hardware 450, 453
 - multithreaded 454
 - treated as capital equipment 453
 - server application 466
 - service 383
 - shallow copy 614
 - defined 624
 - shallow vs. deep copy 624
 - shortcut
 - creating 26
 - modifying properties 27

- Data Control Language 502
 - Data Definition Language 502
 - Data Manipulation Language
 - 502, 506
 - database script
 - dropping and creating tables with 503
 - database scripts
 - using 503
 - executing commands with go 503
 - from clause 508
 - inner join 514
 - join operation 514
 - order by clause
 - example 514
 - prepared statements 526
 - three sub languages 502
 - where clause 508
 - SQL command parameters 526
 - SQL command parameters and prepared statements
 - generalized steps 526
 - SQL command utility
 - use of 502
 - W switch 514
 - SQL query string constants 526
 - SQL Server
 - changing to master database 503
 - data types 505
 - four default databases 502
 - identity operator 511
 - newid() function 513
 - use of 508
 - SQL Server Management Studio 512
 - installation 496–497
 - SQLServer Express
 - installation 495–496
 - stack 347
 - LIFO characteristic 347
 - state transition diagrams 60
 - statement
 - for
 - personality of 148
 - nineteen kinds of 119
 - statements 119–131
 - break 151
 - chained if/else 141
 - continue 151, 152
 - control flow 136
 - do/while 146
 - personality of 147
 - empty 119
 - executing consecutive if 139
 - for 148
 - relationship to while 148
 - goto 153
 - if 136
 - if/else 136, 140
 - iteration 145
 - nesting 149
 - mixing selection and iteration 150
 - nineteen kinds of 120
 - selection statements 136
 - switch 136, 142
 - condition expression types 142
 - nested 144
 - using break in 142
 - table of 154
 - try/catch 138
 - while 145
 - personality of 145
 - state-transition diagrams 59
 - static 114
 - static constructor 195
 - static constructors 200
 - strategy
 - project approach 7
 - StreamReader class 418
 - StreamReaders
 - use in network programming 450
 - StreamWriter class 416, 418
 - strengthening preconditions 650
 - String
 - array of 172
 - string 114
 - string characters
 - accessed using array notation 175
 - string formatting 183
 - structs
 - advice on when to use 227
 - authorized members 226
 - behavior during assignment 226
 - behavior of this 226
 - boxing and unboxing 226
 - default field values 226
 - Structured Query Language 501–515
 - structures
 - structures vs. classes 225
 - stubbing 13
 - subfolder 411
 - subject matter experts 8
 - subscriber 322
 - responsibilities 322
 - subscriber notification process 323
 - supertypes & subtypes
 - reasoning about 643
 - SuspendLayout() method
 - purpose of 309
 - switch
 - implicit case fall-through 143
 - switch statement 142
 - system message queue 294
 - System namespace
 - exploring 96
 - System.Activator.GetObject()
 - example code 469
 - System.Collections 348
 - System.Collections.Generic 348
 - System.Collections.ObjectModel 349
 - System.Collections.Specialized 349
 - System.Diagnostics namespace 645
 - System.Guid
 - use of as primary key 522
 - System.ValueType class
 - direct base class for all value types 118
 - SystemException 367
- ## T
- table 501
 - TableLayoutPanel 291, 310
 - adding multidimensional array of controls to 311
 - properties
 - ColumnCount 311
 - RowCount 311
 - TCP 458
 - octet sequencing 458
 - TCP/IP 450, 451, 452, 457–460
 - application layer 458
 - data link layer 459
 - network layer 459
 - physical layer 459
 - transport layer 458
 - TCP/IP client server programming 478–489
 - TCP/IP client-server
 - binding TcpListener object to machine IP address and port 479
 - calling TcpListener.AcceptTcpClient() method 479
 - calling TcpListener.Start() method 479
 - connection process illustrated 478
 - listening on multiple IP addresses 482
 - multithreaded server
 - building 480

- serializing complex objects between 484
 - simple example code 479
 - TcpChannel 467, 468, 469
 - TcpClient 478
 - TcpListener 478
 - TELNET 458
 - test data
 - inserting into database with script 506
 - test driver program 209
 - testing 209
 - user-defined type 209
 - text files
 - delimiter 418
 - issues to consider before creating 418
 - procedure to read 419
 - Text property
 - effects on different controls 302
 - TextBox 291
 - multiline
 - selecting line of text by double-clicking 315
 - property
 - MultiLine 316
 - WrapContents 316
 - textfiles
 - reading and writing 418–420
 - TextWriter 417
 - the art of programming 4, 10
 - inspiration 10
 - money but no time 11
 - mood setting 11
 - time but no money 11
 - where not to start 10
 - your computer 11
 - thinking outside the box 190
 - this()
 - called from constructor 216
 - thread
 - execution context 384
 - thread context 384
 - thread queue 384
 - ThreadPool 382
 - ThreadPool class 399
 - number of default worker threads 399
 - starting threads with 400
 - threads 382–405
 - asynchronous method calls 400
 - BackgroundWorker class 396
 - BackgroundWorker events 396
 - blocking with Thread.Join() 392
 - blocking with Thread.Sleep() 391
 - creating managed threads 385
 - executing on single-processor system 384
 - foreground vs. background 394
 - ParameterizedThreadStart delegate 390
 - passing ThreadStart delegate to Thread constructor 389
 - preempted 384
 - running asynchronous methods with delegates 400
 - setting Thread.IsBackground property 394
 - starting managed threads 389
 - thread state 389
 - ThreadPool class 399
 - ThreadStart delegate 389
 - time-slicing 384
 - ThreadStart delegate 388
 - timeless way 688
 - time-slicing 384
 - TimeSpan
 - passing to Thread.Sleep() method 391
 - TimeSpan structure
 - example use of 310
 - title bar
 - window 293
 - ToolStripMenuItem
 - constructor usage 313
 - ToolStripMenuItems
 - declaring and creating 313
 - transitivity
 - exhibited by inheritance hierarchies 257
 - Transmission Control Protocol (TCP) 458
 - transport layer 458
 - tree command 36
 - try/catch statement 138
 - type 257
 - diagram 115
 - value type
 - behavior 116
 - type coercion 265
 - types 115–119
 - array 164
 - predefined 115
 - mapping to system namespace structures 118
 - reference 115
 - behavior 116
 - value 115
 - value range table 118
- ## U
- UDP 450, 458
 - UML 15, 190, 234, 250
 - class diagram 193
 - composite aggregation 237, 239
 - expressing abstract class 268
 - expressing inheritance 258
 - expressing interfaces 271
 - expressing realization 271
 - expression aggregation 236
 - realization
 - diagram
 - expanded form 272
 - simple form 272
 - sequence diagram
 - engine object creation 244
 - sequence diagrams 240, 241
 - simple aggregation 237
 - stereotype 194
 - using to tame conceptual complexity 234
 - UML class diagram
 - purpose of 193
 - UML design tool
 - Magic Draw 241
 - Unified Modeling Language (UML) 15
 - uniqueidentifier
 - use as primary key
 - example 504
 - unmanaged code 89
 - update command
 - SQL
 - commands
 - update 509
 - URI 450
 - URL 450
 - User Datagram Protocol (UDP) 458
 - user-defined types 191
 - using 114
 - using directive 111
 - utility methods
 - definition of 201
- ## V
- value objects 494, 495
 - spanning application layers 495
 - value parameters 219
 - value semantics 225
 - value types 115

- ValueType class 118
 - variable 47
 - definition 116
 - verb phrases 46
 - verbatim string literals 412
 - verbs 46
 - vertical access 274, 617
 - view 695
 - virtual
 - keyword to allow method overriding 266
 - Virtual Execution System 87
 - Virtual Execution System (VES) 86, 89
 - virtual machine 86
 - and the common language infrastructure 86
 - virtual machines 86, 87
 - visible region
 - of a window 293
 - Visual C# Express Edition
 - building project 36
 - creating project with 33
 - creating projects with 32
 - installing 33
 - locating project executable file after build 36
 - void 114
- W**
- well-behaved objects 614
 - WellKnownObjectMode.SingleCall mode 469
 - WellKnownObjectMode.Singleton mode 469, 470
 - whole object 235
 - whole objects 236
 - whole/part class relationship 235
 - Width property 302
 - window
 - basic functionality provided by 293
 - message categories 295
 - message prefixes 295
 - parts of 293
 - title bar 293
 - visible region 293
 - window application
 - execution thread 294
 - window coordinates 297, 299
 - window coordinates diagram 298
 - window message routing 294
 - window messages
 - trapping with IMessageFilter interface 296
 - window types
 - dialog boxes 292
 - floating 292
 - multiple-document interface (MDI) 292
 - standard 292
 - tool 292
 - windows
 - messages
 - WM_CHAR 296
 - WM_KEYDOWN 296
 - WM_KEYUP 296
 - WM_MOUSEMOVE 296
 - WM_MOUSEWHEEL 296
 - windows events
 - processing 293
 - windows executable
 - compiler switch 293
 - creating 293
 - Windows Task Manager
 - using to show applications and processes 383
 - word 80, 81
 - world
 - imperfect understanding of 668
- X**
- XML documentation
 - generating from command line 71
 - XML serialization 413
 - XMLSerializer 417
 - XMLSerializer class 416