

INDEX

Symbols

- != operator 153
 - overloading 148
- #endregion directive 444
- #region directive 444
- .NET remoting infrastructure 396
- == operator 153
 - overloading 148
- [Serializable] 357
- [XmlElement] 357
- A**
- alter statement
 - used to create foreign key constraint 414
- application domain 214
- application layers 396
- architectural diagram
 - multitiered database application 396
- ArgumentOutOfRangeException 79
- array 48, 79
 - as foundation for list collection 78
 - creating with literal values 30
 - declaration syntax 25
 - definition of 24
 - difference between value type and reference type arrays 31
 - dynamic resizing
 - example code 79
 - elements 24
 - functionality provided by array types 26
 - homogeneous elements 24
 - Main() method String parameter 43
 - multidimensional 38, 41
 - of value types 28
 - properties of 27
 - references
 - calling Array class methods on 29
 - single dimensional 28
 - single dimensional in action 33
 - specifying length 25
 - specifying types 25
 - two dimensional
 - example program 41
 - type inheritance hierarchy 26
 - value type
 - memory arrangement 28

- Array class 44
- array initializer expression 40
- array literal 30, 31
- array of arrays 40
- array-based collection 78
 - growing on insertion 79
- array-based collections
 - how they work 78
- ArrayList 80, 81–88
 - expanded inheritance hierarchy 82
- ArrayList inheritance hierarchy 81
- arrays
 - rectangular 38
 - sorting with Array class 44
- Ashmore's hash code algorithm 150
- Ashmore's Hash Code Generation Algorithm 150
- asynchronous method calls 230
- asynchronous methods
 - EndInvoke() method 232
 - IAsyncResult interface 233
 - obtaining results from 232
 - providing callback method to BeginInvoke() method 232
- automated water tank custom event example 266–271
- auxiliary storage device 298
- B**
- BackgroundWorker 212
- BackgroundWorker class 226
- BackgroundWorker events 226
- BaseDAO
 - class definition
 - using DatabaseFactory class 425
- binary data 308–310
- BinaryFormatter class 302
- BinaryReader class 308, 310, 311
- BinaryWriter class 308, 310
- BindingContext
 - refreshing 288
- BindingList<T> 276, 281
 - accessing protected members 282
 - deserializing 334
 - subclassing 276
 - subclassing example 282
- BitVector32 384, 387
- bitwise equality 148
- Bloch's hash code algorithm 150
- blocking behavior

- Monitor.Enter() 251
- buckets 134
- Business Layer 396
- business object
 - definition 397
- business objects 396
- business rules 397
 - creep 397
- C**
- C# lock keyword 240
- cascade delete 403
 - SQL
 - cascade delete
 - testing 417
- chained hash table 51
- chained hashtable
 - example code 136
- circular array 119
 - example code 120
 - implemented with ordinary array 120
- class derivation constraint 61, 67
- CloseReader() method 426
- Collection<T>
 - as base class 276
- CollectionBase 342
- CollectionChanged event 276
 - handler method 280
- CollectionChangedEventManager 388
- collections
 - ArrayList
 - usage example 80
 - linked list node elements 49
 - non-generic to generic mapping table 19
 - old-school style programming 18
 - performance characteristics
 - arrays 48
 - hashtable 51
 - linked list 49
 - red-black tree node elements 52
 - serializing 301
 - specialized 19
 - underlying data structures 19
- collections and events 276–295
 - BindingList<T> 276, 281
 - DataGridView 289
 - INotifyPropertyChanged interface 291
 - one-way databinding 276

- SortableViewList<T> 276
- two-way databinding 276, 289
- collision
 - hashtable 134
- columns 403
- command-line arguments
 - processing 43
- Common Log File System 326
- complex project folder organization 418
- ComVisibleAttribute(true) 84
- concurrently executing applications 214
- configuration file
 - example 430
- connection pooling 397
- connection string
 - database
 - configuration file setting 401
- constraint
 - database 406
- create tables SQL script 406
- creating a custom list collection
 - approaches 85
- CRUD operations
 - database
 - CRUD operations 426
- custom collections 340–379
 - creating strongly-typed non-generic 341
 - extending ArrayList 341
 - extending CollectionBase 342
 - extending existing collection 340
 - implementation considerations 340
 - named iterators 356
- custom event
 - recursive example 267–269
- custom events 262
 - suggested naming convention 271
- custom ordering 148, 159
- custom serialization 347
- D**
- DAO layer
 - building 421
- Data Access Layer 396
- data access object
 - definition 397
- data access objects 396
- data base
 - key factor in business rules 397
- Data Control Language 404
- Data Definition Language 404
- Data Manipulation Language 404, 408
- data type
 - reference 26
 - value 26
- data types
 - array 26
 - SQL Server 407
- database
 - automatically inserting primary key 414
 - cascade delete 403
 - columns 403
 - constraint
 - definition of 406
 - converting binary data into bitmap image 428
 - creating related table with script 413
 - DataBase.AddInParameter() method 428
 - foreign key 403, 413
 - foreign key constraint naming 414
 - inserting image data
 - example code 427
 - inserting test data into related table 414
 - inserting value objects into 428
 - join operation 413
 - primary key 403
 - record 416
 - referential integrity 403
 - rows 403
 - table 403
- database application
 - compiling 402
- database connection
 - established via DatabaseFactory 397
- database connection string 401
- database connection test application 400–403
- database management system 403
- Database object 400, 425
- database script
 - running
 - example 406
- DatabaseFactory 397, 425
 - configuration file 401
 - example code 401
- databinding
 - one-way 281
 - two-way 281
- DataBindingComplete event 483
- DataGridView 465, 467
 - BindingContext
 - refreshing 288
 - clicking on row to yield row index 465
- data binding 483
- DataSource property 465
- initializing 293
- manipulating Columns property after DataBindingComplete event fires 293
- row index value 465
- DbCommand 428
- DBMS 403
- DbType enumeration
 - .NET type mapping table 429
- default constructor constraint 61, 62
- delay
 - example code 270
- delegate 262
 - event subscriber list 262
 - EventHandler 263
 - method signature specification 263
- delegate object
 - purpose of 262
- delegates
 - running asynchronous methods with 230
- delete command
 - SQL
 - commands
 - delete 412
- delimiter
 - text file 306
- deserialization
 - object 302
- deserialize
 - object
 - from XML file 304
- deserializing
 - BindingList<T> collection
 - example code 334
- device driver 298
- dictionaries 134–145
- Dictionary<TKey, TValue> collection 143
- directory
 - definition 299
- Directory class 299
 - example code 300
- DirectoryInfo class 299
- disk
 - driver software 298
- drive letters 299
- dynamic array resizing
 - example code 79
- dynamic growth capability
 - of array-based lists 78
- DynamicArray
 - case study 78

- E**
- EmployeeDAO 397
- EndInvoke() method 232
- enumerator
 - read-only sequence of elements 83
- Enterprise Library Configuration tool 401
- Enterprise Library Data Access Application Block 396
- enumerator
 - purpose of 82
- equality operations
 - programming for 148
- event 262
- event arguments
 - example code 264
- event consumer 262
- event handler
 - explicit call to
 - example 483
- event producer 262
- event subscriber list 262
- exceptions
 - low-level to high-level translation 313
 - translating low-level to high-level 313
- executing SQL command
 - example code 400
- extension methods 84
 - using on a List<T> object 89
- F**
- façade software design pattern
 - example 123
- façade software pattern 85
- file
 - definition 298
- File class 299
- file I/O 298–338
- file position pointer 309, 310
- FileDialogs
 - using 328–330
- FileInfo class 299
 - example code 300
- files
 - manipulating 299–301
- FileStream class 302, 310
- First-In-First-Out (FIFO) 53
- fixed-length records 310
 - reading
 - example code 317
- folder 299
- foreach statement
 - example explained 83
- foreign key 403, 413
- foreign key constraint 414
- formatting
 - numeric strings
 - table 489
- from clause
 - use to join tables 416
- G**
- generic collection types 57
- generic method 60
 - creating with multiple type parameters 60
 - definition 60
- generic type 58
 - constraints 61
 - listed 61
 - creating 58–60
 - creating with multiple type parameters 59
 - creating with single type parameter 58
 - definition 58
- generic type constraints 57
 - class derivation constraint 67
 - interface implementation constraint 66
 - most useful 61
 - naked constraint 70
 - reference semantics vs. value semantics 64
 - reference type constraint 64
 - table of 72
- generic type inference 61
- generic type parameters 57
- generic types
 - benefits of use 72
- Globally Unique Identifier (Guid)
 - using in program 152
- GUI
 - data input dialog design 472
 - loading image in PictureBox
 - example code 431
 - opening image file with OpenFileDialog
 - example code 431
 - using dialogs to enter data 472
- GUI layout
 - using mock-up sketch to design 462
- H**
- hard disk 298
- hash code
 - algorithm 150
- hash function 51, 134
- hash functions 134
- hash table 48
 - chained 51
 - open address 51
 - slot probe function 51
- HashSet<T> 194
 - inheritance class diagram 194
- hashtable
 - buckets 134
 - calculating load limit 138
 - collision resolution
 - chaining 134
 - double hashing 140
 - collisions 134
 - example code 136
 - growth factors 134
 - keys 134
 - load limit formula 138
 - operation 134
 - values 134
- Hashtable collection 140
- hashtables 134–145
- homogeneous data types 24
- HybridDictionary 386
- I**
- ICancelAddNew interface
 - functionality of 282
- ICloneable 84
- ICollection 84
- IComparable interface
 - implementing 156
- IComparable.CompareTo() method
 - rules for implementing 156
- IComparable<T> interface
 - implementing 156
- IComparer interface
 - implementing 159
- IComparer<T> interface
 - implementing 159
- IDataReader 429
- IDictionary<KeyValuePair<TKey, TValue>> 356
- IDictionary<TKey, TValue> 356
- IEnumerable 82, 356
- IEnumerable<T> 356
- IEnumerator 82
- IList 84
- IList interface 81
- IList methods
 - non supported in System.Array class 45
- Image
 - converting to byte array 428
- Image data
 - storing and transferring as byte array 468
- immutability 162
- immutable object
 - creating 162

- indexer
 - example 79
 - example code 78, 79
- inner join 416
- INotifyCollection 383
- INotifyCollectionChanged interface
 - functionality of 277
- INotifyPropertyChanged interface 291
 - functionality of 277
 - implemented on Person class properties 289
- interface implementation constraint 61, 66
- IRaiseItemChangedEvents interface
 - functionality of 282
- IsProperSupersetOf() method 203
- iterator
 - purpose of 82
- iterator properties vs. enumerator methods 374
- iterator software design pattern 82
- IXmlSerializable interface 357
- J**
- join operation 413
- K**
- key 134
 - hashing to bucket value 134
- keys
 - coding rules 161
 - creating objects for use as 148
 - using objects as 161
- KeyValuePair
 - example code 135
- L**
- Last-In-First-Out (LIFO) 53
- legacy datafile adapter 310
- linked list 48
 - circular list 91
 - diagram 91
- linked list node
 - example code 91
- linked lists 90–99
- linked lists vs. array-based lists 90
- LinkedList<T> 78, 96–99
 - characteristics 97
- List<T> 78, 88–90
- List<T> vs. ArrayList 89
- ListBox
 - binding datasource to 282–288
- ListDictionary 386
- Lists 78
- lock keyword 313
 - compared to Monitor.Wait()/Monitor.Exit() 313
- log files 326–328
- M**
- managed threads 215
- memory leak 388
- menu 463
- Microsoft Enterprise Library
 - installation 399–400
 - support for application layers 396
- Microsoft Enterprise Library Application Blocks 396
- Microsoft Intermediate Language
 - tracing execution example 105
- Microsoft SQL Server Express Edition 396
- MinuteTick custom event example 263–265
- monalphabetic substitution 35
- Monitor class 246
 - synchronizing thread access with 312
 - usage 312
- MSBuild 418
 - <Csc> task 421
 - <ItemGroup> tag 421
 - <project> tag 420
 - <PropertyGroup> tag 420
 - <Target> tag 421
 - compiling value object target 424
 - default target 421
 - items
 - referencing 421
 - project file
 - example 419
 - properties
 - referencing 420
 - targets
 - defining 421
 - using to manage and build project 419
- MSIL
 - directives
 - maxstack 105
 - evaluation stack 106
 - local variables 106
- multithreaded programming 212
- multithreaded vacation 212
- multi-tier projects
 - recommended approach 421
- multitiered database application
 - design 396
- multitiered database applications 396–486
- N**
- naked constraint 61, 70
 - limited usefulness of 71
- named iterators 356
- NameValueCollection 384
- naming conventions
 - for custom events 271
- natural ordering 148, 156
- NotifyCollectionChangedEventArgs properties 277
- numeric formatting 489
- O**
- object immutability 148
- Object.Equals() method
 - overriding 148, 149
 - rules for overriding 149
- Object.GetHashCode() method
 - general contract 149
 - overriding 148
 - rules for overriding 149
- ObservableCollection<T> 276
 - class diagram 276
 - example program 277
 - responding to events 276
- obsolete Thread methods 219
- one-way databinding 276
 - example 282
- open address hash table 51
- operating system
 - file management services 298
- overloaded operators
 - example code 67
- overriding Object.GetHashCode()
 - checklist 149
- P**
- palindrome 126
- palindrome checker program 126
- ParameterizedThreadStart delegate 220
- path
 - absolute 299
 - definition 299
 - relative 299
- Path class 299
- preempted 214
- prepared statements 428
- primary key 403
 - automatically incrementing integer 414
- process 212
 - definition 213, 214
 - multithreaded
 - definition 214
 - single-threaded
 - definition 214
- PropertyChangedEventArgs
 - extracting data from 291
- publisher 262
 - responsibilities 262

Q

queue 53
 FIFO characteristic 53
 Queue class 125
 Queue class inheritance hierarchy 125
 Queue<T> 127
 used in store simulation 128
 Queue<T> inheritance hierarchy 127
 queues 118–130
 characteristic queue operations 118
 dequeue operation 118
 enqueue operation 118
 implemented with circular array 119

R

ragged array 40
 random access file I/O 310–325
 calculating fixed-length record count 310
 RDBMS 403
 record 416
 record locking 312
 rectangular arrays 38
 recursion
 example 269
 red-black tree 48, 170–184
 benefits over ordinary binary trees 170
 example code 174
 link to java animation applet 172
 node 170
 operation 171
 properties 172
 RotateLeft diagram 173
 RotateRight diagram 173
 tree fix-up cases 172
 reference equality 148
 reference equality vs. value equality 148
 reference type constraint 61, 63
 referential integrity 403
 relational database 403
 relational database management system 403
 relationships
 between database tables 403
 remote object
 creating for multitiered application 454
 Remoting exception
 problem sending bitmap across application domains 465
 ring buffer 119

root directory
 definition 299
 rows 403
S
 select command 409
 sensor
 multimode
 example 267
 serializable attribute 301
 SerializableAttribute 84
 serialization
 BindingList<T>
 automatically upon ListChanged event 331
 object 301
 serializing
 objects
 as XML 304
 serializing collections 301
 serializing objects 301–306
 steps to 302
 service 213
 Sets 194–209
 HashSet<T> 194
 set operations 196
 IntersectWith() method 196
 IsProperSubsetOf() method 200
 IsSubsetOf() method 198
 IsSupersetOf() method 201
 Overlaps() method 205
 SymmetricExceptWith() method 208
 UnionWith() method 197
 SortedSet<T> 194
 single-threaded vacation 212
 SortableBindingList<T> 276, 282
 SortedCollections 170–192
 SortedDictionary<TKey, TValue> 170, 185
 performance compared to SortedList<TKey, TValue> 190
 SortedList<TKey, TValue> 170, 190
 SortedSet<T> 195
 inheritance class diagram 195
 sorting
 arrays with Array class 44
 SplitContainer
 example code 329
 SQL 403–417
 AND operator 416
 commands
 alter 404

 create 404, 405
 delete 409
 drop 404
 insert 409
 select 409
 update 409
 use 404
 constraint
 definition of 406
 creating tables 406
 Data Control Language 404
 Data Definition Language 404
 Data Manipulation Language 404, 408
 database script
 dropping and creating tables with 406
 database scripts
 using 405
 executing commands with go 406
 from clause 410
 inner join 416
 join operation 416
 order by clause
 example 416
 prepared statements 428
 three sub languages 404
 where clause 410
 SQL command parameters 428
 SQL command parameters and prepared statements
 generalized steps 429
 SQL command utility
 use of 405
 -W switch 416
 SQL query string constants 428
 SQL Server
 changing to master database 406
 data types 407
 four default databases 404
 identity operator 414
 newid() function 416
 use of 410
 SQL Server Management Studio 414
 installation 398–399
 SQLServer Express
 installation 397–399
 stack 53
 balanced symbol checker 109
 boxing and unboxing of items 103
 command line postfix calculator 113
 custom stack example code 107
 disassembled MSIL example 103
 LIFO characteristic 53
 MSIL evaluation stack 106
 operation 102–107

- peek operation 102
- pop operation 102
- push operation 102
- pushing reference types 103
- pushing value types 103
- Stack class 109
 - inheritance hierarchy 109
- Stack<T> class 112
 - inheritance hierarchy 112
- Store Simulation 128
- StreamReader class 306
- StreamWriter class 304, 306
- String
 - array of 34
- string characters
 - accessed using array notation 37
- string formatting 489
- StringCollection 383
- strong reference 389
- Structured Query Language 403–417
 - subfolder 299
 - subscriber 262
 - responsibilities 262
 - subscriber notification process 263
- SynchronizedCollection<T> 255
- SynchronizedKeyedCollection<T> 255
- SynchronizedReadOnlyCollection<T> 255
- System.Collections 18
- System.Collections.Generic 18
- System.Collections.ObjectModel 19
- System.Collections.Specialized 19, 382–392
 - namespace members table 382
- System.Guid
 - use of as primary key 424
- T**
- table 403
- test data
 - inserting into database with script 409
- text files
 - delimiter 306
 - issues to consider before creating 306
 - procedure to read 308
- textfiles
 - reading and writing 306–308
- TextWriter 305
- thread
 - execution context 214
- Thread class 215
- thread context 214
- thread queue 214
- thread synchronization 238–259
 - .NET VM implementation 242
 - C# lock keyword 240
 - creating synchronized collection with Synchronized() method 242
 - critical code sections 238
 - IsSynchronized property 242
 - MethodImplOptions.Synchronized attribute 254
 - Monitor class 246
 - Monitor.Enter() & Monitor.Exit() 248
 - Monitor.Enter() blocking behavior 251
 - Monitor.TryEnter() 251
 - old-school collection classes 242
 - overloaded Monitor.Enter() 249
 - recommendations for usage 255
 - shared code segments 238
 - shared resources or objects 238
 - synchronizing on same object 241
 - SyncRoot property 242
 - the need for 238
 - usage table 256
- ThreadPool 212
- ThreadPool class 229
 - number of default worker threads 229
 - starting threads with 230
- threads 212–235
 - asynchronous method calls 230
 - BackgroundWorker class 226
 - BackgroundWorker events 226
 - blocking with Thread.Join() 222
 - blocking with Thread.Sleep() 221
 - creating managed threads 215
 - executing on single-processor system 214
 - foreground vs. background 224
 - ParameterizedThreadStart delegate 220
 - passing ThreadStart delegate to Thread constructor 219
 - preempted 214
 - running asynchronous methods with delegates 230
 - setting Thread.IsBackground property 224
 - starting managed threads 219
 - thread queue 214
 - thread state 219
 - ThreadPool class 229
 - ThreadStart delegate 219
 - time-slicing 214
- ThreadStart delegate 218
- time-slicing 214
- TimeSpan
 - passing to Thread.Sleep() method 221
- transactional capability 282
- two-way databinding 276, 289
- type argument 58
- type parameters 58
- types
 - array 26
- U**
- unbounded type parameters 57, 58, 60
- uniqueidentifier
 - use as primary key example 406
- update command
 - SQL commands update 412
- V**
- value equality 148
- value objects 396
 - spanning application layers 396
- value type constraint 61, 63, 65
- verbatim string literals 300
- W**
- weak event listener pattern 388, 389
- WeakEventManager 389
- Windows Task Manager
 - using to show applications and processes 213
- X**
- XML serialization 301
- XMLSerializer 305
- XMLSerializer class 304