## Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;&amp; operator</td>
<td>as boolean operator, 26–27</td>
</tr>
<tr>
<td></td>
<td>method-based queries and, 446</td>
</tr>
<tr>
<td></td>
<td>query expressions and, 434, 435</td>
</tr>
<tr>
<td>#define</td>
<td>preprocessor directive, 498, 505</td>
</tr>
<tr>
<td>#elif</td>
<td>preprocessor directive, 498–499</td>
</tr>
<tr>
<td>#else</td>
<td>preprocessor directive, 498–499</td>
</tr>
<tr>
<td>#endif</td>
<td>preprocessor directive, 498–499</td>
</tr>
<tr>
<td>#endregion</td>
<td>preprocessor directive, 500–501</td>
</tr>
<tr>
<td>#error</td>
<td>preprocessor directive, 498–499, 500</td>
</tr>
<tr>
<td>#if</td>
<td>preprocessor directive, 498–499</td>
</tr>
<tr>
<td>#line</td>
<td>preprocessor directive, 500</td>
</tr>
<tr>
<td>#pragma checksum</td>
<td>preprocessor directive, 500–501</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>#pragma</td>
<td>preprocessor directive, 501–502</td>
</tr>
<tr>
<td>#region</td>
<td>preprocessor directive, 500–501</td>
</tr>
<tr>
<td>#undef</td>
<td>preprocessor directive, 498, 505</td>
</tr>
<tr>
<td>#warning</td>
<td>preprocessor directive, 500</td>
</tr>
<tr>
<td>$ character</td>
<td>as anchors, 479, 482–483</td>
</tr>
<tr>
<td>// characters</td>
<td>comment sections and, 22</td>
</tr>
<tr>
<td>@ character</td>
<td>as anchors, 479, 482–483</td>
</tr>
<tr>
<td>[ ] (square brackets)</td>
<td>arrays and, 362</td>
</tr>
<tr>
<td></td>
<td>statement lambdas and, 221</td>
</tr>
<tr>
<td>[DataContract]</td>
<td>attribute, 418</td>
</tr>
<tr>
<td>[XmlIgnore]</td>
<td>attribute, 418</td>
</tr>
<tr>
<td>^ character</td>
<td>as anchors, 479, 482–483</td>
</tr>
<tr>
<td>{ } curly brackets</td>
<td>C# connections and, 381</td>
</tr>
<tr>
<td></td>
<td>complex statements and, 21</td>
</tr>
<tr>
<td></td>
<td>to denote statement blocks, 30</td>
</tr>
<tr>
<td></td>
<td>statement Lambdas and, 352</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>method-based queries and, 446</td>
</tr>
<tr>
<td></td>
<td>query expressions and, 434</td>
</tr>
<tr>
<td></td>
<td>operator</td>
</tr>
<tr>
<td>&lt;T&gt;</td>
<td>generic type parameter, 103, 104</td>
</tr>
<tr>
<td>! (unary logical negation operator)</td>
<td>28</td>
</tr>
<tr>
<td>!= operator</td>
<td>26</td>
</tr>
<tr>
<td>= = operator</td>
<td>26</td>
</tr>
<tr>
<td>=&gt; (goes to operator)</td>
<td>lambda expressions and, 352</td>
</tr>
<tr>
<td></td>
<td>method-based queries and, 445</td>
</tr>
<tr>
<td>\ (backslash)</td>
<td>regular expressions and, 477</td>
</tr>
<tr>
<td>; (semicolons)</td>
<td>do-while loops and, 46</td>
</tr>
<tr>
<td></td>
<td>simple statements and, 21–22</td>
</tr>
<tr>
<td></td>
<td>for statements and, 39</td>
</tr>
<tr>
<td>... (ellipses)</td>
<td>ellipses and circles (real-world scenario), 170–171</td>
</tr>
<tr>
<td></td>
<td>else clause and, 34</td>
</tr>
<tr>
<td>( ) parentheses</td>
<td>in lambda expressions, 220</td>
</tr>
</tbody>
</table>
abstract methods, 90–91
access modifiers, variables and properties and, 96
accessor methods, 72
Action delegates, 216
adding
  assemblies to global assembly cache, 562–564
  event handlers to form’s Paint event, 217–218
  functions to structs, 69–70
  methods to structs, 69–70
  parameters to lambda expressions, 220
  records to tables with DataAdapter (code lab), 385–386
  records using WCF Data Services, 402–403
AddToCategories method, 402
ADO.NET, 377–394
  ADO.NET Entity Framework. See ADO.NET Entity Framework basics of, 377
  DataAdapter to add records to tables (code lab), 385–386
  DataSets, DataTables, and DataAdapter, 384–385
  DbDataAdapter to update and delete records (code lab), 387–388
  System.Data.Common.DBCommand class. See Command object
  System.Data.Common.DBConnection class, 377–379
ADO.NET Entity Framework, 388–394
  basics of, 388
  Entity Framework Models, creating, 388–391
  records, deleting, 393
  records, inserting, 392–393
  records, selecting, 391–392
  records, updating, 393
  stored procedures, calling, 393–394
Advanced Build Settings dialog, throwing overflow exceptions and, 244
aggregate functions, queries and, 455–456
algorithms
  asymmetric, 534
  basics of, 528–529
  hashing, 539
  selecting for encryption, 529, 548
  symmetric, 530
aliases
  comparison of value types and (code lab), 63–64
  value types and, 61
alternation constructs (regular expressions), 481
ancestors (classes), defined, 163
anchors (regular expressions), 479
anonymous methods
  basics of, 351
  defined, 349
  vs. delegates, 351
  delegates and, 217–218
  lambda expressions and, 218, 349
anonymous types
  query expressions and, 438
  querying data and, 16
AnonymousGraph example program, 221–222
application input, validating (as exam objective), 15
applications
  debugging (as exam objective), 15
  improving performance. See threads arguments
  defined, 84
  methods and, 82
  named, 94
ArrayList class, 365–371
basics of, 365–366
Hashtable, 369
vs. List class, 373
methods, 366–368
Populating drop-down list from generic list (real-world scenario), 376–377
properties, 366
Queue collection, 369
SortedList collection, 370
Stack collection, 370–371
when to use, 373
arrays
casting, 119–122
data collections and, 362–365
defined, 362
multidimensional, declaring, 363
of references, 119, 121, 122
of variables, delegate types and, 209
as keyword, casting and, 118
as operator
core, basics of, 118–119
casting arrays and, 120
assemblies
basics of, 551–552
defined, 321, 551
GetCustomAttributes method, 335–336
global assembly cache, adding assemblies to, 562–564
managing (as exam objective), 15, 528
side-by-side versioning, 558–562
signing, strong names and, 555–558
versions, 552–554
Assembly class (System.Reflection namespace), 321–325
assertions
core, basics of, 252–253
data integrity, managing with, 494–497
defined, 494
asymmetric encryption, 15, 529, 534–538
async keyword, 276, 293, 414, 415
async lambdas, 222–223
asynchronous application programming, 293–297
asynchronous I/O operations, 414–415
Asynchronous Pattern Model (APM), 276
asynchronous processing
as exam objective, 13, 266
lambda expressions and, 222
when to make methods asynchronous, 294
AsyncLambda example program (code lab), 223
atomic operation, defined, 297
attribute methods, serialization of
data and, 421
attributes, custom, 335–340
attributes basics, 335
creating, 337–338
read attributes, 335–336
Using (real-world scenario), 338–340
AutoResetEvent class, 301
average aggregate function, 455–456
await keyword
asynchronous I/O operations and, 414, 415
asynchronous programming and, 293, 295, 296, 297
threads and, 276
BackgroundWorker class, 276–279
banker’s rounding, 127
barriers
basics of, 302–304
using with cancellations (code lab), 309–310
base classes, inheriting from, 162–171
base classes, defined, 163
basics of, 162–163
base classes (continued)

constructors that invoke other
constructors (code lab), 166–167
ellipses and circles (real-world scenario),
170–171
parent class constructors, calling,
164–165
same class constructors, calling, 165–166,
167–170
base keyword, calling parent class
constructors and, 164, 168
“Beginners Guide to Performance
Profiling”, 513
binary serialization of data, 416–417
**BinaryFormatter** object
basics of, 416
vs. **System.Xml.Serialization**
namespace, 417
**BinarySearch** method, 368
**BinaryWriter** class, 413–414
binding redirection, 561
block size in cipher block chaining, 529–530
blueprinting process, 9
**bool**
keyword, 28
program flow and, 29
structs and, 61
true and false values and, 27–28
boolean and bitwise operators, 26
boolean expressions, 28–29
**Boolean** types, see **bool**
boxing
custom collections and, 374
value types, 128–130
brain dumps, 4
breakpoints, lambda expressions and, 219
Build number, assembly versions and, 552
Build property, 504, 505
built-in functions, validating data with,
473–474

C

C# programming basics, 19–57
C# program structure, 21
complex statements, 23
exam objectives, 20
loops. See loops
program flow. See decisions in code;
program flow
simple statements, 21–23
CA (Certificate Authority), 543–544
**CalculateGrandTotal** method, 490
**Calculator** class, 344
call stack, throwing exceptions and, 251
callback, creating and implementing (as
exam objective), 13
cancellations, 309–310
case statements, 37
casting. See also converting values,
casting and
basics of, 117–118
cast operators, 119
casting arrays, 119–122
is operator, 118
as operator, 118–119
catch lists, for **try-catch-finally** block, 238
catch sections
sorting exceptions and, 236
of **try-catch-finally** block, 235, 238
catching exceptions, 235, 250
CCIE (Cisco Certified Internetworking
Expert) certification, 2
Certificate Authority (CA), 543–544
Certificate Revocation List (CRL), 544
certificate stores, 545–546
certificates, 542–546
certification. See also Microsoft certifications
defined, 2, 4
class, 4
character classes (regular expressions),
478–479
character escapes (regular expressions), 478
Cheat Sheet, xxiv

checked blocks, throwing overflow exceptions and, 244

checksums, \#pragma checksum directive and, 503

class hierarchies, 161–206

base classes, inheriting from. See base classes, inheriting from

child classes

DbException, 243

defined, 163

cipher block chaining, 529–530
ciphertext, 528

Cisco Certified Internetworking Expert (CCIE) certification, 2
class hierarchies, 161–206

base classes, inheriting from. See base classes, inheriting from
exam objectives, 162

interfaces. See interfaces, designing and implementing

object life cycle. See object life cycle, managing
terminology, 163

classes. See also reference types

character classes (regular expressions), 478–479

child classes, DbException, 243
class files, OOP and, 76
class hierarchy (as exam objective), 14

CodeDOM namespace, 341–344
collections, types to use, 373

concurrent collection classes, 309

creating and using (code lab), 79–80

exception, 240–242

files and directory classes (System.IO

namespace), 405

inherited. See inheritance

methods in (code lab), 83–84

nested, 77

in .NET code, 76–77

new operator and, 63

Reader and Writer, 411

string, 144–148

System.Collections namespace, 365

System.Reflection namespace, common, 320–321. See also specific classes

trace listeners, 506

classes, CodeDOM namespace

CodeCompileUnit class, 344

CodeDOMProvider class, 348–349

CodeMemberField class, 345

CodeMemberMethod class, 347–348

CodeMemberProperty class, 345–347

CodeMethodInvokeExpression class, 348

CodeNamespaceImport class, 344

CodeNamespace class, 344

CodeParameterDeclarationExpression
class, 348

CodeTypeDeclaration class, 345

common, 341–344

client application that uses WCF Service, creating, 400–403

Clone method

ICloneable interface, 183–184

System.Array class, 364

CloneArray example program, 133

clones, 183–184

Close method, 381

closing connections, 381

CLR. See Common Language Runtime (CLR)
code

code branching, 24
data validation code, methods and, 497
duplicated, 175–176
efficiency, 66

generating, using CodeDom. See CodeDOM

namespace

using to subscribe to events, 230–231

Code Document Object Model. See CodeDOM

namespace
code editor (Visual Studio), inserting event

handler names and, 230–231

code generators

CodeDOM and, 340
defined, 340
Code Labs as book convention, xxv–xxvi
CodeBinaryOperatorExpression class, 347–348
CodeConditionStatement class, 347
CodeDOM namespace, 340–349
basics of, 340–341
classes, 341–344
CodeCompileUnit class, 344
CodeDOMProvider class, 348–349
CodeMemberField class, 347–348
CodeMemberMethod class, 345
CodeMemberProperty class, 345–347
CodeMethodInvokeExpression class, 348
CodeNamespaceImport class, 344
CodeNamespace class, 344
CodeParameterDeclarationExpression class, 348
CodeTypeDeclaration class, 345
CodePlex.com, 388
collection classes
ArrayList class. See ArrayList class
basics of, 365
System.Collections namespace, 365
CollectionBase class, 374–375
collections, defined, 362
COM Interop, unmanaged code and, 130, 133
Command object, 379–384
basics of, 379
ExecuteNonQuery method, 379–380
ExecuteReader method, 380–383
ExecuteScalar method, 383
ExecuteXmlReader method, 384
comment section, simple statements and, 22
Common Language Runtime (CLR)
assemblies and, 551, 553–554
interoperability and, 130
managed resources and, 191
common mistakes, infinite loops, 41
CompareTo method, 177, 368
Comparing Cars (code lab), 178–179
comparisons, 27
compiler constants (predefined), debugging and, 503–504
complex statements, 21, 23
composite keys
method-based queries and, 452–453
query expressions and, 441–442
concatenation, queries and, 457–459
concurrent collections, 308–309
conditional instructions, 25–28
conditional operators, 28
conditions, defined, 25
Configuration Manager, 503
Connection property, 379
connection strings, 378
connections, closing, 381
ConnectionString property, 378
constants
compiler constants, debugging and, 503–504
defined, 25
constructors
basics of, 81–82
invoking multiple, 169–170
overloaded methods and, 88
overloading (real-world scenario), 89–90
parent class constructors, calling, 164–165
same class constructors, calling, 165–166, 167–170
String class, 139
string constructors, 138–139
that invoke other constructors (code lab), 166–167
constructs
alternation constructs (regular expressions), 481
character class, 478–479
grouping constructs (regular expressions), 480–481
consume data (as exam objective), 16
consume types (as exam objective), 14
consuming data. See ADO.NET
ContainsKey method, 373
context (assemblies), 324
continuations (TPL), 291–292
covariance
CovarianceAndContravariance example
program (code lab), 214–215
delegates and, 214, 350
data
data access, implementing (as exam objective), 15–16
ADO.NET and. See ADO.NET
authenticity, hashing and, 538
collections for storing. See data
collection classes
compressing, encryption and, 538
consuming. See ADO.NET; WCF Data Services
detecting incorrect input of. See validation of input
encrypting. See encryption
to denote statement blocks, 30
currency
Displaying Currency Values (real-world scenario), 153
NumberStyles.Currency, 136
Parse method and, 124, 126
custom formatting strings, 151
data
control, defined, 4
credentialed, defined, 4
CRL (Certificate Revocation List), 544
cryptography. See also encryption
defined, 528
collection classes
core, 265, 269, 287
count aggregate function, 455
core
CovarianceAndContravariance example
program (code lab), 214–215
delegates and, 214, 350
CPU usage, 512
core
create types (as exam objective), 13
CreateInstance method, 324
core
credential, defined, 4
CRL (Certificate Revocation List), 544
cryptography. See also encryption
defined, 528
curly braces ({}),
complex statements and, 23
data
core
cores, 265, 269, 287
core
count aggregate function, 455
core
data structures. See structs
data types. See also delegates
   basics of, 61–62
   storage needs and, 66
DataAdapter (ADO.NET)
   adding records to tables with (code lab), 385–386
   basics of, 384–385
   DataAdapter class, 385
databases
   database validations, 494
   files, programming, 508
[DataContract] attribute, 418
DataContractJsonSerializer class, 418
DataSets (ADO.NET), 384–385
DataTables (ADO.NET), 384–385
date and time
   OData $filter date functions, 399
   Parse method and, 124
   standard DateTime formatting strings, 152–153
DBConnection class, 377–379
DbDataAdapter, updating and deleting records using (code lab), 387–388
DBDataReader class
   ExecuteReader method and, 380
   methods, 382–383
   properties, 381
DBNull.Value vs. null values, 381
Debug and Trace classes, 504–508, 509
Debug.Assert method, 495, 504
debugging, 497–508
   applications, 14–15, 498
   database files, programming, 508
   Debug and Trace classes, 504–508
   DEBUG symbol, 503
   exam objectives, 470
   predefined compiler constants, 503–504
   preprocessor directives. See preprocessor directives
decimal.Parse method, 124
decision types basics, 30
decisions in code, 29–38
   if statements, 30–32, 34
   if statements, beyond basic (code lab), 35–36
   if statements (code lab), 32–34
   switch statements, 36–38
declarations, variables and, 23
declaring events, 224–225, 226
decrement operators, for statements and, 40
decrementing, loops and, 42
decryption
   with asymmetric encryption, 535
   of chipper text to plain text, 533
   defined, 528
   of streams, 537
deep clones, 184
default constructors, 82
DefaultIfEmpty method, outer joins and, 440
DefaultTraceListener class, 504, 506
deferred execution, defined (query expressions), 434
#define preprocessor directive, 498
defining
   delegates, 208–209
   interfaces, 173–174
delayed processing, 22–23
delegate keyword, 211
delegate variables
   basics of, 209–210, 211
   contravariance and, 214
   covariance and, 214
   series of, 211–212
   storing lambda expressions in, 218
   Using delegate variables (code lab), 210–211
delegates, 208–223
   Action delegate, 216
   anonymous methods, 217–218, 351
   basics of, 77, 208, 349
   covariance and contravariance, 214
DeleteCommand property – encryption

defining, 208–209
delegate variables. See delegate variables
  Func delegate, 216
lambda expressions and. See lambda
  expressions
static and instance methods, 212
Understanding covariance and
  contravariance (code lab), 214–215
Using delegate variables (code lab),
  210–211
Using static and instance delegates (code
  lab), 212–214
DeleteCommand property, 387–388
deleting
  DbDataAdapter to update and delete
  records (code lab), 387–388
  records (ADO.NET Entity
    Framework), 393
  records, using WCF Data Services, 403
Dequeue method, 369
derived classes, defined, 163
deriving
  new exception classes, 252
  one class from another, defined, 163
descendants (classes), defined, 163
deserialization
  of data (as exam objective), 16
  of exceptions, 252
Deserialize method, 416
destructors
  defined, 191
  finalizers and, 192
  providing, 191–197
diagnostics (as exam objective), 470
dictionaries, defined, 362
Dictionary type, System.Collections.
  Generic namespace, 372–373
directories, working with (I/O), 405–408
Directory and DirectoryInfo classes (I/O),
  406–408
DisplayErrorMessage, order entry forms
  (real-world scenario), 136
Dispose method, 190, 191, 193, 197
Distinct method
  Distinct with custom classes (code lab),
    460–461
  method-based queries and, 459–461
Divide method, creating, 347
DLL files (assemblies), 551
DllImport statement, unmanaged code and,
  130–132, 133
do loop vs. while loops, 47–48
do-while loops vs. while loops, 47–48
do-while statements, loops and, 46–49
drop-down lists, populating (real-world
  scenario), 376–377
dynamic types, 133–137

#elif preprocessor directive, 498–499
ellipses ( . . . )
  ellipses and circles (real-world scenario),
    170–171
  else clause and, 34
#else preprocessor directive, 498–499
Empty field (String class), 139
empty statements, 22
encapsulation, 95–102
  basics of, 60, 95–96
  enforced, using properties, 97–98
  indexed properties, 101–102
  properties, 96–97
  properties, accessing, 100–101
  properties, using (code lab), 98–100
Encrypt data using the ProtectData class
  (real-world scenario), 547–548
encryption, 527–551
  algorithms, selecting, 529, 548
  asymmetric, 15, 534–538
encryption – exam objectives

encryption (continued)
basics of, 528–529
certificates, 542–546
defined, 528
Encrypt data using the ProtectData class
(real-world scenario), 547–548
as exam objective, 15, 528
hashing, 538–542
key management, 547
RSA asymmetric algorithm, using (code
lab), 548–551
stream, 536–538
symmetric. See symmetric encryption
#endif preprocessor directive, 498–499
#endregion preprocessor directive, 500–501
enforcing encapsulation (as exam
objective), 14
Enqueue method, 369
Entity Data Model Wizard, 389, 390, 393
Enumerating Tree Nodes (code lab), 185–188
enumerations
basics of, 72–75
enum keyword and, 61
enums, using (code lab), 75–76
GetEnum methods, and (System.Type
class), 329–330
made easy, 189
naming, 73
equals keyword, join clauses and, 440
Equals method, 182
#error preprocessor directive, 498–499, 500
errors
#warning and #error directives and, 500
error checking vs. exception
handling, 234
escape sequences, defined, 477
event handlers
adding to form’s Paint event, 217–218
anonymous methods and, 217
event logs, 509–511
Event Viewer, 511
Event-based Asynchronous Pattern
(EAP), 276
events, 223–233
basics of, 77, 223–224
creating and implementing (as exam
objective), 13
event publishers, defined, 77
event subscribers, defined, 77
publishing. See publishing events
subscribing and unsubscribing to,
230–233
synchronization events, 298–302
System.ComponentModel.
BackgroundWorker class, 277
EventWaitHandle class, 299–301
Exam 70-483
basics of, 8–10
taking, xxiii
questions in, how written, 9–10
recommended prerequisites for
studying for, 11–12
tips and tricks, 10
exam delivery partners (EDPs), defined, 2
tips
70-483 exam, 12–16
assemblies, managing, 528
asynchronous processing, 266
C# programming basics, 20
class hierarchies, 162
data, working with, 362
debugging, 470
encryption, 528
events and callbacks, 208
exception handling, 208
LINQ, 432
managing program flow, 208
multithreading, 266
security, 470
type system in C#, 60
types, 114, 320
value types, 61
ExcelInterop example program, 133

Exception class
creating custom exceptions and, 251
properties, 246–247

exception handling, 234–253
basics of, 234–235
error checking vs., 234
as exam objective, 13
exception properties, 246–247
exception types, 240–244
ExceptionType class, 235–236
Factorials (real-world scenario), 244–246
throwing and rethrowing exceptions. See exceptions, throwing
try-catch-finally blocks, 235–238
unhandled exceptions, 238–239

exceptions
Dictionary type, 373
protecting code from, 239
return values and, 248–249
sorting, 236
exceptions, throwing
assertions, 252–253
catching, rethrowing and, 249–251
custom exceptions, creating, 251–252
exceptions and return values and, 248–249
invalid integer conversions and, 115
narrowing conversions and, 115–116
throw statement, 250

EXE files (assemblies), 551
ExecuteNonQuery method, 379–380
ExecuteReader method, 380–383
ExecuteScalar method, 383
ExecuteXmlReader method, 384
exit conditions for loops, 41
explicit conversions, 116
Exponent method, creating, 348
expression Lambdas

basics of, 218–221
defined, 352
expressions, defined, 25
extension methods, 92–93

Factorial method, 244, 245, 249
Factorials (real-world scenario), 244–246
false values
if statements and, 34
type bool and, 27

fields. See also text box fields
defined, 67, 330
instance fields (code lab), 79–80
in .NET code, 77
reference types, basics of defining, 79
string fields, 139–140

File and FileInfo classes (System.IO namespace), 366
FileAccess enumeration, 409
FileIOPermissionAccess enumeration, 410
FileNotFoundException, 561
 FileMode enumeration values, 409
FileNotNotFoundException, 561
files
database files, programming, 508
defined, 404
 FileMode, FileAccess, or FileShare enumerations, 409
I/O operations and, 405–408
working with (I/O), 405–408
files classes (System.IO namespace), 405
FileStream constructor, 408–409
FillBrush property, 198
filtering
method-based queries and, 445–446
query expressions and, 434–435

finalization, defined, 190
finalization queues, destructors and, 193
Finalize method, 190, 192
finalizers, destructors and, 192
finally section, of try-catch-finally block, 235, 237, 238
First method, queries and, 456–457
FlagsAttribute, 285
floating point conversion, overflows and, 116 floating point values
  casting, 117
  floating point, 117
  methods, 244
for loops
  nested, 42–43
  vs. while loops, 46
for statements, loops and, 39–42
foreach loops, 44–45
  complex statements and, 23
  read-only indexer in, 140
foreach statements
  looping over string characters and, 140
  loops and, 43–44
fork-join pattern, defined, 267
formatting
  strings, type conversion and, 151–153
  values, type conversion and, 149–153
forms, order entry forms (real-world scenario), 135–137
FreeResources method, 194, 195
from clause syntax, query expressions and, 433
Fully Qualified Name (FQN), 558
Func delegates, 216
Functional Groups, 11–12
functions
  adding to structs, 69–70
  built-in, validating data with, 473–474
  OData $filter OData, 398–400
G
GAC (Global Assembly Cache), 558, 562–564
gacutil.exe, 561–562
garbage collection, defined, 190
garbage collector (GC), unreachable memory and, 190
generic collection classes, IComparable and, 183
generic methods, 103–105
generic types
  basics of, 102–103
  generic delegate types, 215–216
  vs. non-generic, 371
generic vs. nongeneric versions of IComparable, 178
generic vs. nongeneric versions of IComparer, 180
get method, properties and, 97
GetArrayRank method (System.Type class), 328
GetConstructors method (System.Type class), 328–329
GetCustomAttributes method (assemblies), 335–336
GetDataAsync method, 296
GetEnum methods (System.Type class), 329–330
GetEnumerator method, 185, 186, 188
GetExecutingAssembly method (Assembly class), 322, 323
GetExportedTypes and GetTypes methods (Assembly class), 322, 323
GetField/GetFields methods (System.Type class), 330–332
GetLoadedModules, GetModules, and
  GetModule methods (Assembly class), 323
GetMethod/GetMethods methods (System.Type class), 332–333
GetObjectData method, 420
GetProperty/GetProperties methods (System.Type class), 332
GetReferencesAssemblies method (Assembly class), 325
GetStringDelegate delegate type, 212
GetType methods, (System.Data.Common.DBDataReader), 382
Global Assembly Cache (GAC), 558, 562–564
goes to operator (=>)
   lambda expressions and, 352
   method-based queries and, 445
GraphFunction example program code lab, 210–211
   lambda expressions and, 221
GroupBy method, 452
grouping
   method-based queries and, 454–455
   query expressions and, 443–445
grouping constructs, regular expressions for data validation, 480–481
GroupJoin method, 450–452
GTE CyberTrust Global Root, 544

H
hash bucket, 538
hashing data, 538–542
Hashtable, 369, 373
HasRows property, 381, 383
heaps, in .NET code, 76
“How to: Create Custom Performance Counters”, 514

ICloneable class, 183–184
ICloneablePerson example program, 184
IComparable
   IComparableCars example program (code lab), 178–179
   implementing interfaces and, 177–179
   sorting and, 367–368
IComparer
   IComparerCars example program, 181
   implementing interfaces and, 179–182
IDisposable interface, implementing, 176, 190–191
IDisposableClass example program, 193–196, 197
IEnumerable
   defined, 44
   implementing interfaces and, 185
IEnumerableTree example program (code lab), 185–188, 189
IEquatable, 182–183
if, else if statements, decisions in code and, 34
#if preprocessor directive, 498–499
if statements
   basics of, 32–34
   beyond basic if statements, 35–36
   code lab, 35–36
   decisions and, 30–32, 34
   ignoring properties, 418
IGrouping<TKey, TElement> collection, 443, 445, 454
ildasm.exe, 559–560
immutability of strings, 138
implicit conversions, 116
in keyword, 216
increment operators, for statements and, 40
incrementing, loops and, 39–42
indexers
  basics of, 101–102
  CollectionBase class and, 375
  read-only indexer (String class), 139–140
indexes, array, 367
indexing data, hashing and, 538
inheritance
  basics of, 91–92
  event inheritance, 227–228
inherited classes, defined, 163
infinite loops, 41
initialization vector (IV) in cipher block chaining, 530
initializer, for statements and, 39
InitializeService method, 402–403
initializing
  defined, for statements and, 39–41
  String variables, 138
inline options (regular expressions), 480
inner sequences, joining and, 450
inserting records (ADO.NET Entity Framework), 392–393
instance fields
  code lab, 79–80
  defined, 79
instance method, delegates and, 212
instances
  of classes defined in assemblies, creating, 324
  creating, 80
  of int variable, creating, 326–327
instrumentation, 509–517
  defined, 509
  logging and event logs, 509–511
  profiling. See profiling
  tracing, 509
int datatype, 62
integer overflows, 115–116
integer types, math vs. programming integer types, 38
integer values, enums and, 73
IntelliSense
  exam and, 10, 21
  new types and, 74–75
  optional and named parameters and, 94
  showing values in persons array, 121
  static methods and, 93
  Student class properties and, 100
interface inheritance, 172–174
interfaces, designing and implementing
  basics of, 171–172
  defining interfaces, 173–174
  delegating interfaces, 175–176
  Enumerating Tree Nodes (code lab), 185–188
  enumerations made easy, 189
  ICloneable, 183–184
  IComparable, 177–179
  IComparer, 179–182
  IEnumeratable, 185
  IEquatable, 182–183
  implementing, 174–175
Interlocked class, 306, 308
Intermediate Language Disassembler
  (ildasm.exe) application, 559–560
Intermediate Language (IL), assemblies and, 551
intern pools, String variables and, 138
interoperability, unmanaged code and, 130–133
into clause, grouping and, 444
intrinsic data types, 61
int.TryParse method, 234
InvokeMember method of the System.Type class, 333
I/O operations, 405–415
  asynchronous, 414–415
  basics of, 405
as exam objective, 15
files and directories, 405–408
readers and writers, 410–414
streams, 408–410
is operator
basics of, 118
casting arrays and, 120
ISerializable interface, 420–421
IsLetter method, 140
iteration
iterators in loops, defined, 39
read-only indexer as source of, 140
JavaScript Object Notation (JSON)
request data in JSON format (WCF Data Services), 403–404
serialization of data, 418–419
WCF Data Services and, 394
join keyword, outer joins and, 440
joining
method-based queries and, 449–452
query expressions and, 438–441
JSON. See JavaScript Object Notation (JSON)
Just In Time (JIT) compiler, assemblies and, 551
key management
asymmetric encryption and, 535
basics of, 547
hashing algorithms and, 539
key secrecy, 533–534
RSA asymmetric algorithm, using (code lab), 548–551
key parameter, SortedList and, 370
Key Terms, xxiv

L
lambda expressions, 218–223
async lambdas, 222–223
basics of, 349–352
expression lambdas, 218–221
statement lambdas, 221–222
Language Integrated Query (LINQ). See LINQ
Last method, queries and, 456–457
left joins, query expressions and, 441
Legacy Certifications, defined, 5
Length property (string class), 139
Length property (System.Array class), 363–364
library references, COM Interop and, 133
line numbers, #line directive and, 500
LINQ, 431–467. See also method-based LINQ queries
basics of, 431, 432
data and objects and (as exam objective), 16
exam objectives, 432
LINQ to XML, 461–462
queries, writing, 392
query expressions. See query expressions
list box fields, data validation and, 473
List class (System.Collections.Generic namespace), 373–374
listeners, Debug and Trace classes, 506–508
lists, defined, 362
literals, defined, 25
load context (assemblies), 324
load-from context (assemblies), 324
LoadFrom or LoadFile methods (Assembly class), 323
locale-aware parsing, 126
localization, 555
locks
alternatives to locking, 306–307
dealing with, 307
locking mechanisms, 304–308
logging, 509–511
loops, 38–51
  basics of, 38–39
do-while statements, 46–49
foreach loops (code lab), 44–45
foreach statements, 43–44
infinite loops, 41
looping structures, 39
loops (code lab), 49–50
nested for loops, 42–43
parallel, 289–290
for statements, 39–42
while statements, 45–46
lottery program, nested loops for (real-world scenario), 42–43

M
Major number, assembly versions and, 552
Makecert.exe (Certificate Creation Tool), 545
managed resources
  CLR and, 191
  vs. unmanaged resources, 193
ManualResetEvent class, 301
many-core processors, 269
MarshalAs attributes, 132–133
MaskedTextBox control, 471
math
  OData $filter math functions, 399–400
  vs. programming integer types, 38
max aggregate function, 456
MCM (Microsoft Certified Master) certification, 2
MCP (Microsoft Certified Professional) certifications, 5. See also Microsoft certifications
MCPD (Microsoft Certified Professional Developer) certification, 6
MCSD (Microsoft Certified Solutions Developer) certification, 5
MCSE (Microsoft Certified Systems Engineer) certification, 5
MCSM (Microsoft Certified Solutions Master) certification, 2
MCTS (Microsoft Certified Technology Specialist) credential, 6
MDM, 2
memory
  of CPU usage, 512
  memory address, in .NET code, 76–77
Message Authentication Code (MACs) algorithms, 529
Message Digest (MD) (hash algorithms), 540
method-based LINQ queries, 445–459
  aggregate functions, 455–456
  basics of, 432, 445
  composite keys, 452–453
  concatenation, 457–459
  Distinct method, 459–461
  filtering, 445–446
  First method, 456–457
  grouping, 454–455
  joining, 449–450
  Last method, 456–457
  ordering, 446
  outer joins, 450–452
  projection, 446–449
  vs. query expressions, 432, 445
  Skip or Take methods, 459
methods. See also constructors; specific
abstract and overridden, 90–92
adding to structs, 69–70
anonymous, 217
ArrayList class, 366–368
Assembly class, 322–325
asynchronous, 295, 297
attribute methods, serialization of data and, 421
Barrier class, 302–303
BitConverter class, 128
CountdownEvent class, 301–302
Data validation code and, 497
DBDataReader class, 382–383
Debug and Trace classes, 505
defining, 82–83
delegates and, 208
Directory and DirectoryInfo classes
(I/O), 407
EventWaitHandle class, 299
extension, 92–93
File and FileInfo classes, 406
floating point special value, 244
generic, 103–105
hash algorithms, 541
incoming values and, 84
Interlocked class, 308
making asynchronous, 293, 294, 295
method-based queries and, 445
methods in classes (code lab), 83–85
Monitor class, 305–306
objects and, 77
overloaded, 88
Parallel class, 288–289
private, in structs, 72
public, in structs, 72
Regex class, 476
rethrowing exceptions and, 249–250, 251
rules that apply to destructors, 192
StreamReader class, 411
string, 140–143, 474–475
StringBuilder class, 146
StringReader class, 148
StreamWriter class, 147
symmetric encryption, 531–532
System.Collections
.CollectionBase, 374
System.Collections.Concurrent
namespace, 308
System.Collections.Generic
.Dictionary, 372–373
System.ComponentModel
.BackgroundWorker class, 276–277
System.Convert class, 127
System.Data.Common.DBConnection
class, 378
SymmetricAlgorithm class, 531–532
System.Threading.ThreadPool class,
273–274
System.Type class, 327–328
Task class, 282
TaskFactory, 283
validating data with string methods,
474–475
validation methods, 489–490
value types, passing to (code lab), 85–87
Microsoft certifications, 1–8
basics of, 4–7
certification, defined, 2
exam. See Exam 70-483
MDM, MCSM basics, 2
reasons for, 2–4
various, 2
Microsoft Certified Master (MCM)
certification, 2
Microsoft Certified Professional Developer
(MCPD), 6
Microsoft Certified Solutions Developer
(MCSD) certification, 5
Microsoft Certified Solutions Master
(MCSM) certification, 2
Microsoft Certified Systems Engineer (MCSE)
certification, 5
Microsoft Certified Technology Specialist
(MCTS) credential, 6
Microsoft Learning Web site, 12
Microsoft Technology Associate (MTA), 7
min aggregate function, 456
Minor number, assembly versions and, 552
Model (Entity Framework), creating,
388–391
modifiers, 77–79
modules
    defined, 323
    System.Data assembly, 323
Modules property (Assembly class), 323
modulus, 28
monitors, 305–306, 307
Moore, Gordon, 265
MoveNext method, 185, 188
MTA (Microsoft Technology Associate), 7
multicore processors, 269
multiple class constructors, 165, 169–170
multiple inheritance, interfaces to simulate, 172–174
Multiply method, 332
multithreading
    barriers, 302–304
    barriers code labs, 303–304, 310–311
    basics of, 267
    cancellations, 309–310
    concurrent collections, 308–309
    difficulty of, 297
    disadvantages of, 270
    exam objectives, 266
    lambda expressions and, 222
    lock-free alternatives, 306–307
    managing (as exam objective), 13
    monitors, 305–306, 307
    multithreaded Windows Forms applications, 279–280
    multithreaded WPF applications, 280–281
    mutual exclusion, 304–308
    sharing data and, 298
    synchronization events, 298–302
    mutual exclusion, 304–308
    myFirstDelegate variable, 349–350
    MyLongRunningProcess method, 414

names, signing assemblies and, 555–558
naming
    custom attribute classes, 337
    delegate types, 209
    enumerations, 73
    extension methods, 93
    interfaces, 173
    methods, 88
    when values are the same as names, 330
NaN, overflow exceptions and, 244
narrowing conversions, 114–116, 117
negativeInfinity static property, 244
nested for loops, 42–43
nested if statements, 31, 34
New Generation of Certifications, defined, 5, 6
new operator, creating classes and, 63
New order form (real-world scenario), 484–493
nondeterministic finalization, 190
null vs. DBNull.Value, 381
NumberFormatInfo class, 126
numbers
    NumberStyles enumeration, values defined by, 124–126
    numeric types, defined, 61
    standard numeric formatting strings, 152

object life cycle, managing, 190–199
    basics of, 190
    destructors, providing, 191–197
    as exam objective, 14
    IDisposable interface, implementing, 190–191
    Shape Resources (real-world scenario), 198–199
    using statement, 197–198
object-oriented programming (OOP), 76
Object-Relational Mapping tool, 388
objects
  assigning value to members of. See constructors
  basics of, 77
  Debug and Trace classes, 505
reusing, 191
OData (Open Data Protocol)
  $filter date functions, 399
  $filter math functions, 399–400
  $filter query options, 398
  $filter string functions, 398–399
  $filter type functions, 400
  basics of, 393, 394
  OData ATOM Format, 394
  query options, 397
OnDeserializingAttribute and
  OnDeserializedAttribute, 418, 421
OnSerializedAttribute and
  OnSerializingAttribute, 418, 421
operators. See also specific operators
  basics of, 25–26
  boolean and bitwise, 26
  cast, 119
  conditional, 28
  decrement operators, for statements and, 40
  increment operators, for statements and, 40
  relational, 25
  ternary, 28
options, regular expressions, 480
order entry forms (real-world scenario), 135–137
ordering
  method-based queries and, 446
  query expressions and, 436–437
ORM tool, 388
outer joins
  method-based queries and, 450–452
  query expressions and, 440–441
outer sequences, joining and, 450
OutlinePen property, 198
Overdraft account (real-world scenario), 228–229
overflow exceptions, 244
overridden methods, 90, 91
overriding, defined, 73
oversubscribing to events, 231

P
P2P forums, xxviii
Parallel class, 288–290
Parallel Linq (PLinq), 290
Parallel.For method, 289
ParallelLoopState parameter, 289
ParamArrays, LINQ to XML and, 462
parameters
  adding to lambda expressions, 220
  defined, 84
  events best practices and, 225–226
  optional and named, 94–95
  prefixes on names, 131
  subscribing to events and, 230
parent class constructors, calling, 164–165
parent classes, defined, 163
parentheses ( ) in lambda expressions, 220
BigInt parsing methods
  converting values and, 122–126
  errors and, 234
passwords
  signing assemblies and, 557
  storing, hashing and, 538–539
patterns, defined, 475
PDB file, 508
percentage values, handling (real-world scenario), 144
performance counters, 514–517
Performance Monitor, 514, 516–517
Performance Wizard, 512
Person class (ThisAndBase example program), 167–168
piracy, exams and, 4
PKI (Public Key Infrastructure), 543
Platform invoke (P/invoke), unmanaged code and, 130–132
PLinq, 290
Pop method, Stack collection and, 370–371
positional arguments, named arguments and, 94
PositiveInfinity static property, 244
practice questions basics, 12
#pragma checksum preprocessor directive, 503
#pragma warning preprocessor directive, 501–502
predefined event types, 225
predicate, defined (query expressions), 434
prefixes, on parameter names, 131
prep guide for exam, 9, 11, 12
preprocessor directives
#define and #undef, 498
#error, 500
#if, #elif, #else, and #endif, 498–499
#line, 500
#pragma, 501–502
#pragma checksum, 503
#pragma region and #endregion, 500–501
#warning, 500
primary key values, specifying, 400
PrintGrades method, 175
private keys, 534
private member variables, 96
private methods and properties in structs, 72
probing, defined, 324
ProcessImageFile method, 516
processor architecture, assemblies and, 561–562
profiling
“Beginners Guide to Performance Profiling”, 513
by hand, 513–514
performance counters, using, 514–517
using profilers, 511–513
program flow, 24–51
basics of, 24
bool (code lab), 29
boolean expressions, 28
conditional instructions, 25–28
decisions in code. See decisions in code exam objective, 13
program structure, C#, 21
programming basics. See C#
programming basics
programming integer types vs. math integers, 38
projection
method-based queries and, 446–449
query expressions and, 437–438
properties
accessing (code lab), 100–101
ArrayList class, 366
Assembly class, 321–322
Barrier class, 302–303
DBDataReader class, 381
Directory and DirectoryInfo classes (I/O), 407
encapsulation and, 96–97
enforced encapsulation using, 97–100
exception properties, 246–247
fields and, 77
FileInfo class (I/O), 405
hash algorithms, 539, 542
ignoring, 418
indexed, 101–102
private, in structs, 72
public, in structs, 72
reflection to map table columns to class properties (real-world scenario), 333–334
SQL exceptions, 243
SQLException class, 242–243
static, floating point types, 244
string, 139–140
StringBuilder class, 146
symmetric encryption, 530–531
System.Collections
  .CollectionBase, 374
System.Data.Common.DBConnection class, 378
System.Exception class, 246–247
  SymmetricAlgorithm class, 530–531
System.Type class, 326–327
Task class, 283
using reflection to map table columns to class properties (real-world scenario), 333–334
ProtectData class, encrypting with (real-world scenario), 547–548
ProtectedData class, 547
psychometrics, exams and, 8, 9
public key encryption. See also asymmetric encryption
defined, 529
Public Key Infrastructure (PKI), 543
Public Key Token, 555
public keys, 534
public methods and properties in structs, 72
publishing events
declaring events, 224–225
event best practices, 225–227
event inheritance, 227–228
Overdraft account (real-world scenario), 228–229
predefined event types, 225
publishers, defined, 224
Push method, Stack collection and, 370–371
quad cores, defined, 287
quantifiers (regular expressions), 481
query expressions, 432–445
  aggregate functions and, 455–456
  basics of, 432–434
  composite keys, 441–442
  filtering, 434–435
  First and Last methods, 456–457
  grouping, 443–445
  joining, 438–440
  vs. method-based queries, 432
  ordering, 436–437
  outer joins, 440–441
  projection, 437–438
  syntax, exam and, 433
query options (OData), 397–398
questions in exams, how written, 9–10
Queue collection, 369, 373
queues
defined, 362
Enqueue and Dequeue methods and, 369
QueueUserWorkItem method, 273, 274, 276
race conditions, 279, 297
RACE Integrity Primitives Evaluation
  Message Digest (RIPEMD), hash algorithms and, 540
Rank property, System.Array class, 363–364
reachable memory, 190
ReadDataFromIO method, 293, 294–296
readers
  I/O operations and, 410–414
  System.IO namespace, 410–414
read-only indexer (String class), 139
Real-World Case Scenarios as book convention, xxv
records
adding to tables (code lab), 385–386
adding/deleting/updating, using WCF Data Services, 402–403
selecting/inserting/updating/deleting (ADO.NET Entity Framework), 391–393
reference types, 76–95
basics of, 59–60, 76–77
constructors, 81–82
constructors, overloading (real-world scenario), 89–90
extension methods, 92–93
fields, basics of defining, 79
instance fields (code lab), 79–80
methods, abstract and overridden, 90–92
methods, overloaded, 88
methods basics, 82–83
methods in classes (code lab), 83–85
modifiers, 77–79
parameters, optional and named, 94–95
value types, passing to methods (code lab), 85–87
references
to arrays, 120, 121, 122
using with monitors, 307
reflection. See also System.Reflection
namespace
basics of, 320–321
boxing and unboxing and, 129
reflection-only context (assemblies), 324
using to map table columns to class properties (real-world scenario), 333–334
using with types at runtime (as exam objective), 14
Regex class
data validation and, 475–476
power of, 477
#region preprocessor directive, 500–501
regular expressions for data validation,
475–483
alternation constructs, 481
anchors, 479
basics of, 475–478
character classes, 478–479
character escapes, 478
grouping constructs, 480–481
options, 480
quantifiers, 481
useful, 481–483
relational operators, 25
resource management, 190–193
rethrowing exceptions, 249–251
retrieving data from collections (as exam objective), 16
return statements
methods and, 82–83
statement lambdas and, 221
return values, exceptions and, 248–249
Revision, assembly versions and, 552
RIPEMD (RACE Integrity Primitives Evaluation Message Digest), hash algorithms and, 540
root certificates, 544
RSA asymmetric algorithm, using (code lab), 548–551
RunSequential method, 286
RunTasks method, 286
runtime. See also Common Language Runtime (CLR)
types at, as exam objective, 14
sanity checks
SanityCheckRow method, 493
validating data with, 483
scheduler (TPL), 288
schedulers, threads and, 268
Secure Socket Layer (SSL), web security and, 542, 545
Secured Hash Algorithms (SHAs), 529, 540
security, implementing (as exam objective), 14–15, 470
select clause
  grouping and, 444
  projections and, 437, 438
Select method
  outer joins and, 452
  projection and, 446
SelectMany method, projection and, 448
semicolons (;)
  do-while loops and, 46
  simple statements and, 21–22
  for statements and, 39
sentinels, 48–49
Serializable attribute, 335
serialization of data
  basics of, 416
  binary, 416–417
  custom, 419–421
  as exam objective, 16
  JSON, 418–419
  XML, 417–418
serialization of exceptions, 252
Serialize method, 416
series, of delegate variables, 211–212
set method, properties and, 97
SetEntitySetAccessRule, 403
SetMinThread method, 274
sets, defined, 362
SHA (Secured Hash Algorithms), 529
SHA-1, 540
SHA-2, 540
shallow clones, 184
shallow copies of arrays, 364
Shape Resources (real-world scenario), 198–199
shared secret encryption. See symmetric encryption
sharing data, 298
sibling classes, defined, 163
side-by-side versioning, assemblies and, 555, 558–562
signaling mechanisms, 298
signatures, of methods, 88
signing assemblies, 555–558
simple statements, 21–23
Skip method, queries and, 459
Solutions Associate level, 7
Solutions Expert level, 7
Solutions Master level, 7
Sort method (arrays), 367
SortedList collection, 370, 373
spaghetti, defined, 24
SQL exceptions, 242–243
SSL (Secure Socket Layer), web security and, 542, 545
Stack collection, 370–371, 373
stacks, in .NET code, 76
standard DateTime formatting strings, 152–153
standard formatting strings, 151–152
standard numeric formatting strings, 152
statement lambdas
  basics of, 221–222
  defined, 352
statements
  complex, 23
  methods and, 82
  simple, 21–23
static keyword, declaring properties and, 96
static methods
  delegates and, 212
  IntelliSense and, 93
  Regex class, 476
  static String methods, 140
StaticAndInstanceDelegates example program (code lab), 212–214
Stopwatch class, 513–514
stored procedures – symmetric encryption

stored procedures, calling (ADO.NET), 393–394

storing
  anonymous methods in delegate variables, 217
data in collections (as exam objective), 16
delegate values in variables, 209
passwords, hashing and, 538–539
series of data. See data collections
stream encryption, 536–538

StreamReader class
  ReadToEndAsync method, 415
  basics of, 411–413

streams, 404, 408–410

StreamWriter class, 413

string functions, OData $filter, 398–399

string keyword, 138

string methods
  basics of, 140–141
  string instance methods, 141–143
  validating data with, 474–475

StringBuilder class
  ToString method, 145
  basics of, 145–147
  Using (real-world scenario), 148–149

String.Format method, 150–151

StringReader class
  basics of, 147–148
  vs. StreamReader class, 413

strings, 137–149
  assembly versions and, 552
  basics of, 138
  basics of manipulating, 137–138
  connection strings, 378
  as exam objective, 14
  formatting, type conversion and, 151–153
  methods. See string methods
  percentage values (real-world scenario), 144
  string classes, 137–138, 144–148

string constructors, 138–139
string fields and properties, 139–140
string processing classes, 145

StringBuilder, Using (real-world scenario), 148–149

StreamWriter class, 147

strong names, signing assemblies and, 555–558

strongly typed collections, 374–376

structs
  basics of, 61, 66–70
  creating (real-world scenario), 70–72
delegate types and, 209
illegal use of, 69
instances of, creating, 68
user-defined, 61
working with, 68

studying for exam, 11–12
  Functional Groups, 11–12
  practice questions, 12
  prep guide, 11

subclasses
  covariance and, 214
defined, 163

subclassing, defined, 163

subexpression, regular expressions and, 480

subscribers (events)
defined, 224
  subscribing to events, 230–233

sum aggregate function, 456

superclasses
  contravariance and, 214
defined, 163

swap methods, 104–105

switch statements, decisions and, 36–38

symbols, preprocessor directives and, 498–499

symmetric encryption
  algorithms implemented in .NET, 530
  basics of, 529–530
deciphering chipper text, 533
encrypting text into chipper text, 
532–533
as exam objective, 15, 528
methods, 531–532
properties, 530–531
synchronization events
CountdownEvent class, 301–302
EventWaitHandle class, 299–301
synchronizing resources. See barriers;
synchronization events
System idle process, 268
system log files, writing event information 
into, 510
System.Array class, 363
System.Attribute abstract class, 337
System.BitConverter class, converting 
values and, 128
System.Collections namespace, 365
System.Collections.CollectionBase 
methods, 374
properties, 374
System.Collections.Concurrent 
namespace, 365
System.Collections.Generic namespace, 
371–374
System.ComponentModel.BackgroundWorker 
class. See BackgroundWorker class
System.Convert class, converting values 
and, 127
System.Data namespace, 377
System.Data.Common.DBCommand class, 379
System.Data.Common.DBConnection class, 
377–379
System.Data.Common.DBDataReader, 
381–382
System.Data.SqlClient namespace, 377
System.Data.SqlClient.SqlCommand 
class, 379
System.Data.SqlClient.SqlException 
class, 242–243
System.Diagnostics.Debug class, 252
System.Diagnostics.PerformanceCounter 
object, 514
System.Enum type, 74
System.Exception class properties, 246–247
System.IO namespace, 405
System.IO.FileSystemInfo object, 405
System.Reflection namespace, 320–335
Assembly class, 321–325
common classes of, 320–321
reflection basics, 320–321
System.Type class. See System.Type class
Using reflection (real-world scenario), 
333–334
System.Reflection .ProcessorArchitecture, 561
Formatters.Binary namespace, 416
System.Runtime.Serialization.Json 
namespace, 418
System.Security.Cryptography namespace 
basics of, 530
hashing and, 540
key management and, 547
X509Certificates namespace, 544
System.String, string keyword and, 138
System.Text.RegularExpressions.Regex 
class, 475–476
System.Threading.Thread class. See 
threads
System.Threading.ThreadPool class, 
273–275
System.Type class, 325–333
basics of, 325–326
details about classes or variables and, 320
GetArrayRank method, 328
GetConstructors method, 328–329
System.Type class (continued) – threads

- GetEnum methods, 329–330
- GetField/GetFields methods, 330–332
- GetMethod/GetMethods methods, 332–333
- GetProperty/GetProperties methods, 332
- properties, 327–328

System.ValueType, 61

System.Xml.Linq namespace, 462
System.Xml.Serialization namespace, 417

T

<T>, generic type parameter, 103

T4 templates, 390

- tables
  - adding new records to (code lab), 385–386
  - reflection to map table columns to class properties (real-world scenario), 333–334

- Take method, queries and, 459

- targets, of attributes, defined, 335

- Task Parallel Library (TPL), 281–297
  - asynchronous application programming, 293–297
  - basics of, 276
  - continuations, 291–292
  - methods, 282
  - Parallel class, 288–290
  - properties, 283
  - scheduler, 288
  - Task class, 282
  - Task return type, 293
  - Task<TResult> class, 282, 288
  - Task<TResult> return type, 293
  - TaskCreationOptions enumeration options, 284–285
  - TaskFactory methods, 283

- tasks, creating, 284–288
- tasks, defined, 282
- task scheduler, 288
- Task-based Asynchronous Pattern Model (TAP), 276
- tasks, defined, 282
- templates, T4, 390
- terminology of class hierarchies, 163
- ternary operators, 28

- text box fields
  - data validation and, 473, 481–482
  - New order form (real-world scenario), 484–493
  - validating data and, 472

- text styles in this book, xxvii

Text Transformation Template Toolkit files, 390

- TextReader, StringReader and, 147
- TextReader abstract class, 411
-TextWriter class, 411
-TextWriterTraceListener, 506, 507
- TheFunction variable, 209–210, 211
- ThenBy method, 446

- this keyword, calling parent class constructors and, 166, 168

- ThisAndBase example program, 166–167

ThreadPool class
  - basics of, 273–275
  - Using (real-world scenario), 275–276

- threads, 267–281
  - anonymous methods and, 217–218
  - basics of, 267–271
  - defined, 267
  - manually created vs. thread pool threads, 275–276
  - multithreading. See multithreading
  - reason for creating, 267
  - shortcomings of, 281
  - thread pool, 273–275

- UI, unblocking. See UI, unblocking

- Using (code lab), 271–273
Using the thread pool (real-world scenario), 275–276
throwing exceptions. See exceptions, throwing
tiers in certification programs, 7
time
Parse method and, 124
standard DateTime formatting strings, 152–153
TLS (Transport Layer Security), web security and, 542
tools
Makecert.exe (Certificate Creation Tool), 545
Object-Relational Mapping (ORM) tool, 388
Text Transformation Template Toolkit files, 390
Visual Studio tool for implementing interfaces, 174
ToString method, 147, 150, 151
Trace class
debugging, and, 504–508
tracing and, 509
TRACE symbol, 503, 504
“TraceListener Class”, 508
tracing, 509
Transport Layer Security (TLS), web security and, 542
typeof() keyword, 325
types, 113–159
casting. See casting
CodeDOM namespace. See CodeDOM namespace
converting between types basics, 114, 161
creating and using, as exam objective, 13–14
defined, 320
dynamic, 133–137
exam objectives, 114, 320
generic delegate types, 215–216
implicit and explicit conversions, 116–117
interoperability, unmanaged code and, 130–133
lambda expressions. See lambda expressions
reflection, using with at runtime (as exam objective), 14
reflection and. See System.Reflection namespace
strings, manipulating. See strings
System.Collections namespace, 365
System.Collections.Generic namespace, 371
unhandled exceptions and, 238
TryParse methods
data types providing, 123–124
data validation and, 473
order entry forms (real-world scenario) and, 136
type system in C#, 59–112
encapsulation. See encapsulation
exam objectives, 60
generic methods, 103–105
generic types, 102–103
reference types (classes). See reference types
value types. See value types
types (continued) – value types

value types, boxing and unboxing, 128–130
values, converting. See converting values, casting and
values, formatting, 149–153
widening and narrowing conversions, 114–116

UI, unblocking

BackgroundWorker class, 276–279
multithreaded Windows Forms applications, 279–280
multithreaded WPF applications, 280–281
updating the UI, 281
unary logical negation operator (!), 28
unboxing
custom collections and, 374
value types, 128–130
#endif preprocessor directive, 498
undersubscribing, 233
unhandled exceptions, 238–239
Unicode version UTF-16, 138
unmanaged code, interoperability and, 130–133
unmanaged resources
CLR and, 191
vs. managed resources, 193
unreachable memory, 190
unsubscribing from events, 230–233
UpdateCommand property of DbDataAdapter object, 387–388
updating
records (ADO.NET Entity Framework), 393
records using WCF Data Services, 403

user input, parsing String class methods
and, 143
user-defined structs, defined, 61
using keyword, 198
using statements
for closing connections, 381
interoperability with unmanaged code
and, 130–131
managing object life cycle and, 197–198
as a try-finally sequence, 235–238

ValidateRequiredTextBox, order entry forms
(real-world scenario) and, 136
ValidateRow, order entry forms (real-world scenario) and, 136, 137
validating data, 472–493
with built-in functions, 473–474
New order form (real-world scenario),
484–493
with regular expressions. See regular
expressions for data validation
with sanity checks, 483
with string methods, 474–475
validation of input
avoiding, 470–471
basics of, 470
data integrity, managing, 494–497
triggering, 471–472
validating data. See validating data
value types, 61–76
alias comparison and (code lab), 63–64
allowable in enums, 73–74
basics of, 59, 61–63
enumerations basics, 72–75
collections, 57
...
structs, creating (real-world scenario), 70–72
structs basics, 61, 66–70
types of, 61
using (code lab), 64–66
value_type_passing (code lab), 85–87
values. See also converting values, casting and assigning to enumerators, 74
assigning to members of objects. See constructors
defined by NumberStyles enumeration, 124–126
delegate values, storing in variables, 209
Displaying Currency Values (real-world scenario), 153
FileIOPermissionAccess enumeration, 409
FileMode enumeration, 409–410
floating point operations and, 244
formatting, type conversion and, 149–153
incoming, methods and, 84
null vs. DBNull.Value, 381
percentage values (real-world scenario), 144
true values, 27, 34
of variables, GetFields method and, 331
var, variables defined as, 433
variables
array variables, casting and, 121
contravariant, 214
covariant, 214
declaring as private, 331
defined, 25
defined as var, 433
delegate, 209–210
of delegate type, anonymous method stored in, 217
exception catching and, 237
implicitly typed, 433
values of, GetFields method and, 331
verifying hash for data, 542
versions, assemblies, 552–554, 558–562
virtual methods, 90–91
Visual Studio
code editor, inserting event handler names and, 230–231
debugging and, 41, 50
error messages, 238–239
event handlers and, 231–232
handling dynamic types and, 133–135
IntelliSense and, 94. See also IntelliSense
Server Explorer, 514–515
tool for implementing interfaces, 174
using code to subscribe to events and, 230–231
void return type, 293
von Newmann, John, 267

#warning preprocessor directive, 500
warnings
#warning and #error directives, 500
#pragma warning preprocessor directive, 502
WCF Data Services, 394–404
basics of, 394–395
client application that uses, creating, 400–403
creating, 395–397
OData $filter
OData $filter date functions, 399
OData $filter
OData $filter math functions, 399–400
OData $filter
OData $filter string functions, 398–399
OData $filter
OData $filter type functions, 400
OData $filter query options, 398
OData query options, 397
request data in JSON format, 403–404
websites for downloading
   chapter downloads, xxv
   Northwinds database, 388
   ShortPathNames example program, 132
websites for further information
   BitConverter class, 128
   DllImport statements, 132
   “How to: Create Custom Performance Counters”, 514
   P2P forums, xxviii
   prep guide, 11
   strings, 378
   “TraceListener Class”, 508
where clause
   multiple, 435
   query expressions and, 433, 434, 435
while loops
   vs. do-while loops, 47–48
   vs. for loops, 46
while statements, loops and, 45–47
while keyword, 45
widening conversions, 114, 116, 117
Window Designer, subscribing/unsubscribing to events and, 232
Windows Forms
   event inheritance and, 227
   multithreaded applications, 279–280
   vs. XAML applications, 232
Windows Forms form designer
   subscribing to events and, 231–233
   undersubscribing and, 233
wizards
   Entity Data Model Wizard, 389, 390, 393
   Performance Wizard, 512
   WPF, multithreaded applications, 280–281
   wrapper classes, implementing encryption algorithms and, 529
writers
   I/O operations and, 410–414
   System.IO namespace, 410–414
   WriteToConsoleForward method, 349, 350
   WriteToEventLog program, 510
X
   XAML applications, vs. Windows Forms applications, 232
   XElement class, 462
   XML
      converting LINQ to, 461–462
      serialization of data, 417–418
   [XmlIgnore] attribute, 418
   XmlReader object, 384
   XmlSerializer class, 417
Z
   zero based arrays, 363