

Index

Special Characters

- (unary) operator, 74
!(not) operator, 79
!= (not equal to) operator, 77
% (modulo) operator, 74, 140
& (and) operator, 79
&& (AND) operator, 79, 102
* (multiplication) operator, 73–74, 81
/ (divide) operator, 74
: (colon), class definitions, 286
\\ (Backslash character), 35
\0 (Null character), 35
\\n (New line character), 35, 220
\\r (Carriage return character), 35
\\t (Tab character), 35
^ (or) operator, 79
_ (underscore), 257
| (and/or) operator, 79
|| (OR) operator, 79, 102
~ (tilde), 305
+ (add) operator, 74
< (less than) operator, 77
<= (less than or equal to) operator, 77
= (equals symbol), 26, 242
== (same value) operator, 77
> (greater than) operator, 77
>= (greater than or equal to) operator, 77
?: (ternary) operator, 90

A

abstract class
 BankAccount class example, 327–328
 defined, 328–330
 determining when to use, 353–354
 factoring, 322–327
 objects and, 330
abstract keyword, C# interface, 335
abstracting away, 208
AbstractInheritance program, 330

abstraction concept
 importance of methods, 234
 object-oriented programming
 overview, 207–208
 preparing, 209
 procedural programming, 208
Academic version, Visual Studio, 485
access control
 classes
 accessor methods, 266
 example, 266–269
 overview, 265–266
 object-oriented programming, 212
 overview, 213
 restricting to class members
 overview, 261–262
 public example, 262–264
 security levels, 264–265
access keyword
 internal, 387–390
 protected, 390–392
 protected internal, 392
 relating namespaces to, 395–397
access methods, 270, 273
AccessControl namespace, 412
accessing collections, 145–149
accessing current object
 defined, 254–255
 explicit, 255–257
 overview, 253–254
 working without this keyword, 257–259
accessing data
 how data classes fit into framework,
 417–418
 process, 418
 System.Data namespace
 connecting to data source, 420–425
 entity framework, 431–433
 overview, 416–417
 sample database schema, 419–420
 visual tools, 425–428
 writing data code, 428–431

- accessing Internet
 - how net classes fit into framework, 457–458
 - System.Net namespace
 - checking network status, 459–460
 - downloading file from internet, 460–462
 - e-mailing status report, 462–465
 - logging network activity, 465–468
 - overview, 456
- AccessorPropertyShortcuts example
 - accessor property access levels, 273
 - compiler-created data members, 272–273
- accessors, 270, 273
- AccountNumber property, 270–271
- AccumulateInterest() method, 295–296
- Activity Designer Library project, 493
- Activity Library project, 493
- Adapter classes, 420
- adapters
 - database-specific, 416
 - defined, 173
- add (+) operator, 74
- Add button, Code Snippets Manager tool, 525
- Add Connection dialog box, Visual Studio, 422–423
- Add Customer command, 621–623
- Add() method
 - adding items to List<T> class, 125
 - collections, 155
 - DictionaryExample program, 127
- Add New Data Source link, Visual Studio, 421
- Add New Item dialog box, 530
- AddCustomerCommand, 623
- Add-in Manager tool, Visual Studio, 514
- adding constraints, 184–186
- AddRange() method, 125
- Address() method, 252–253
- Advanced panel, Visual Studio, 520
- Affected users attribute, DREAD model, 404
- AlignOutput program, 61–62
- All-in-one-folder, 383
- altering content sent to clients, 689–690
- and (&) operator, 79
- AND (&&) operator, 102
- and/or (!) operator, 79
- anonymous methods, 368–369, 374
- ANSI text, 438
- app.config file, 465–466
- Append() method, 70
- Append mode, 442
- application. *See* console application; Windows Presentation Foundation (WPF)
- application code, 142
- Application Wizard, Visual Studio 2010, 14–16
- ApplicationCommands Library, 603
- application-scoped resource, 549–550
- Apply the Following Additional Keyboard Mapping Scheme drop-down list, Visual Studio, 519
- App.xaml file, 548, 551
- arguments
 - passing to base class constructor, 300–301
 - passing to methods, 236–238
- arithmetic operators
 - assignment, 75–76
 - increment, 76–77
 - operating orders, 74–75
 - simple, 73–74
- array bounds, 112
- array of strings, 60
- _arrayElements array, 147–148
- ArrayList, 170
- arrays
 - C#
 - argument for, 110
 - fixed-value array, 110–112
 - initializing, 115
 - Length property, 114–115
 - overview, 109–110
 - variable-length array, 112–114
 - compared to collections, 122
 - defined, 109
- arrays of data, 116–120
- as operator
 - avoiding invalid conversions, 297
 - type-safety, 170
- ASCII text, 438
- ASMX, building Web services
 - building code for SHARP, 739–740
 - consuming services in applications, 743–744
 - creating new service, 735–738
 - deploying, 741–742
 - overview, 731–732
 - SOAP, 732–735

- ASP.NET
 - adding custom controls, 683–684
 - breaking down Web applications, 630–632
 - constructing user controls
 - overview, 680–681
 - Phone Number control, 681–683
 - cookies, 699
 - data binding
 - overview, 670–671
 - setting up markup for, 671–673
 - using code-behind, 673–674
 - using commonly bound controls, 674–676
 - displaying info to user
 - images, 661–663
 - labels versus plain old text, 660–661
 - panels and multiviews, 663
 - tables, 663–664
 - input from user
 - submitting input with Submit buttons, 670
 - using multiple-item selection controls, 668–669
 - using other kinds of input controls, 669–670
 - using single-item selection controls, 666–668
 - using text input controls, 664–666
 - overview, 629
 - questioning client
 - getting information back from client, 633–634
 - overview, 632–633
 - scripting client, 633
 - understanding weaknesses of browser, 634–636
 - securing with .NET framework
 - changing trusts, 691–692
 - fixing problems, 692
 - overview, 690–691
 - site accessibility
 - control features, 680
 - design considerations, 680
 - overview, 679
 - styling controls
 - binding styles with CSS, 678–679
 - setting control properties, 677–678
 - Web Development with, 6
 - Web servers
 - PostBack, 636–639
 - state, 639
 - ASP.NET AJAX Server Control Extender project, 492
 - ASP.NET AJAX Server Control project, 492
 - ASP.NET MVC project, 494
 - ASP.NET Server Control project, 492
 - ASP.NET Web Application project, 492
 - ASP.NET Web Service Application project, 492
 - AsReadOnly() method, List<T> class, 350
 - assemblies, 264, 379–381
 - assembly directive, T4, 541
 - assemblyinfo.cs file, 528
 - assignment operator, 26, 75–76, 221
 - asterisk (*) operator, 73–74
 - Attach to Process tool, Visual Studio, 514
 - authentication, 401, 404–407
 - Authentication class, 457
 - authorization, 401, 404
 - Authorization class, 457
 - Authorization namespace, 412
 - Auto definition, Grid Panel, 563
 - autocompleting, 510–511
 - autoindenting, 89
 - automation pattern, 536
 - auxiliary windows, 512–514
 - Auxiliary windows tool, 509
 - Average() method, 243–244
 - average variable, 104
 - AverageAndDisplay() method
 - example, 237–240
 - returning values, 244
-
- ## B
-
- Backslash character (\\), 35
 - Balance property, 270–271
 - BankAccount class example
 - access control, 265–266, 269–270
 - default constructors, 275–276
 - initializers, 281–282
 - listing, 262–263
 - overview, 288–291

- BankAccount class example (*continued*)
 - SimpleSavingsAccount program, 290–295
 - static properties, 271
 - updated, 302–306
- base class constructor
 - base keyword, 301–302
 - invoking default, 298–299
 - passing arguments to, 300–301
- base class method, hiding
 - accidentally, 312–313
 - general discussion, 309–311
 - making hiding approach better than adding simple test, 312
- base interface, 351
- base keyword, 301–302, 314
- base namespace, 393
- basic input controls, XAML, 572–574
- Batch Build option, Build and Debug menus, 515
- Beck, Kent, 610
- behavior testing, 620
- binary operators, 74
- BinaryReader class, 452
- BinarySearch() method, 126
- BinaryWriter class, 452
- \bin\Debug directory, 450
- \bin\Debug subdirectory, 450
- binding, 606–607. *See also* data binding
- BindingNavigator object, Visual Studio, 426–427
- BindingOperations class, 583
- bitwise operator version, 79
- BL (Business Layer), 613
- black box problem, 417
- blocks. *See also* catch block; iterator block
 - finally, 190, 450
 - streamwriting, 442–445
 - try, 189, 441, 450
- bool type, 34
- bound controls, data binding, 674–676
- boxed value-type, 170
- boxing, 171
- braces, 86
- break command
 - looping, 99–100
 - nesting loops, 107
 - switch statement and, 94

- Breakpoints window, Visual Studio, 513–514
- browser-as-client model, 634–636
- brushes, System.Drawing namespace, 471
- bubble sort algorithm, 117
- buffer, 452
- BufferedStream class, 454
- buffering, 454
- BuildASentence program, 51
- Building menu, 515
- built-in commands
 - WPF, 603–604
 - XAML, 604
- built-in Sort() method, 119
- Business Layer (BL), 613
- button handler, Visual Studio, 375
- Button input control, 573
- byte integer, 28

C

- C#
 - array
 - argument for, 110
 - fixed-value array, 110–112
 - initializing, 115
 - Length property, 114–115
 - overview, 109–110
 - variable-length array, 112–114
 - compiler
 - inferring data types, 42–43
 - writing properties, 272–273
 - constructor, 274–275
 - defined, 12–13
 - defining data binding with, 583–584
 - Dynamic Language Runtime (DLR), 778–780
 - events
 - handling, 374–375
 - how publisher advertises events, 370–371
 - observer design pattern, 369–370
 - passing information to event handler, 372–373
 - publishing, 372
 - Publish/Subscribe pattern, 370
 - raising, 373–374
 - subscribing, 371–372
 - interface, 338

- new features, 2–3, 6
- shifting toward dynamic typing, 770–772
- support for object-oriented programming, 212–213
- versus XAML, 552–553
- Cache function, `System.Net` namespace, 458
- CalculateInterest program, 87–88, 95
- CalculateInterestTable program, 96–97, 229
- CalculateInterestTableMore Forgiving program, 100–103
- CalculateInterestTableWithBreak program, 100
- CalculateInterestTableWith Methods program, 230–231
- CalculateInterestWithEmbedded Test program, 91
- CalculateRatio() method, 234
- CalculateSin() method, 234
- CalculateWordWrap() method, 234
- calculating operation type
 - explicit type conversion, 82
 - implicit type conversion, 81–82
- Calendar input control, 573
- call chain, unwinding, 190
- callback method, 357–358, 360–362
- camel-casing, 38
- CanExecute event handler, 611
- CanExecute property, 607
- CanExecuteChanged event, 602–603
- Canvas Panel, WPF, 560–561
- Carriage return character (`\r`), 35
- case
 - class names, 216
 - strings
 - converting to upper- or lowercase, 52–53
 - distinguishing between all-uppercase and all-lowercase strings, 52
- case values, `switch` statement, 94
- cast, 41–42, 82, 295–296
- catch block
 - Directory of Files, 139
 - exceptions, 189, 196–197
 - intercepting exceptions, 204
 - Main() method, 451
 - StreamReader class, 450
 - catch keyword, 187–188
- Change Data Source dialog box, Visual Studio, 423
- char variable type, 34–35
- character types
 - char variable type, 34–35
 - special chars, 35
 - string type, 35–36
- chatty Web services, 724–725
- CheckBox input control, 572
- Chonoles, Michael Jesse, 323
- Choose Toolbox Items tool, Visual Studio, 514
- Choose Your Data Connection screen, Visual Studio, 422
- Choose Your Database Objects screen, Visual Studio
 - Data Source Configuration Wizard, 423–425
 - Entity Data Model Wizard, 432
- chunky Web services, 724–725
- class constraint, 185
- Class Designer
 - Data View, 501–502
 - overview, 500–502
- class diagrams, 432–433
- class inheritance, 213
- class libraries, 199, 380–381
- Class Library project, 491
- class members
 - defined, 216, 250
 - naming, 217
 - as static members, 225
- class methods, 228
- class properties, 224
- classes
 - abstract
 - BankAccount class example, 327–328
 - defined, 328–330
 - factoring, 322–327
 - objects and, 330
 - access control
 - accessor methods, 266
 - to data members, 212–213
 - example, 266–269
 - overview, 265–266
 - const data member, 225–226

classes (continued)

constructors

- C#-provided constructor, 274–275
- objects and, 273
- replacing default constructor, 275–284

containing classes, 223–224

creating for library, 384–385

defining, 215–217

defining properties

- accessors with access levels, 273
- letting compiler write properties, 272–273
- overview, 270–271
- side effects, 272
- static properties, 271

generating static in class members, 224–225

hierarchy of, 299

inheritance, 286–287

naming, 216

objects

- accessing members of, 218
- defining, 217–218
- discriminating between, 220–221
- program example, 218–220

objects versus, 273

overview, 215–216

putting into libraries

- adding second project to existing solution, 383
- creating classes for library, 384–385
- creating projects for class library, 382
- creating stand-alone class library, 382–383
- using class library from program, 386
- using driver program to test library, 385–386

putting into namespaces

- declaring namespace, 394–395
- fully qualified names, 397–398
- overview, 392–393
- relating namespaces to access keyword, 395–397

readonly data member, 225–226

references, 221–222

restricting access to class members

- overview, 261–262
- public example, 262–264
- security levels, 264–265

sealed, 330–331

Unified Modeling Language (UML), 323

visibility and accessibility in

- namespace, 396

classification concept, object-oriented

- programming, 209–211

Clean option, Build and Debug

- menus, 515

client

altering content sent to, 689–690

questioning

- getting information back from client, 633–634

overview, 632–633

scripting client, 633

weaknesses of browser, 634–636

software, Visual Studio, 487

client-side storage, 697–698

CLR (Common Language Runtime),

- 3, 380, 545

code. *See also* writing secure code

- building based on outside data, 536
- changing Stack Panel Orientation, 558
- guidelines for error handling, 199–200
- reusing, 23–24

reusing from Toolbox, 23–24

saving, 23

saving in Toolbox, 23

WPF ViewModel

Add Customer command, 621–623

model, 615–617

model repositories, 619–621

model unit tests, 617–618

overview, 614–615

testing, 624

View, 624–626

Code element, Visual Studio, 524

code region, Visual Studio, 18

Code Snippets Manager tool, Visual Studio,

- 514, 525

- Code View, Visual Studio
 - adding functionality using C#, 649–650
 - interface
 - autocompleting with IntelliSense, 510–511
 - exploring auxiliary windows, 512–514
 - outlining, 511–512
 - overview, 509–510
 - overview, 647–649
 - Page.Load, 650–651
- code-oriented namespace, 458
- coding behind
 - data binding, 673–674
 - event handler, 601
 - overview, 652–653
- collections
 - accessing, 145–149
- C# array
 - argument for array, 110
 - fixed-value array, 110–112
 - initializing, 115
 - Length property, 114–115
 - overview, 109–110
 - variable-length array, 112–114
- compared to arrays, 122
- defined, 417
- dictionaries, 126–128
- flexibility, 121–122
- initializing, 129–130
- initializing arrays, 128
- iterating
 - iterators, 141–145
 - through directory of files, 135–141
- lists, 124–126
- loop made `foreach` array, 115–116
- looping around iterator block, 150–167
- old-fashioned, 134
- sets, 130–133
- sorting arrays of data, 116–120
- syntax
 - `<T>` notation, 123
 - generic, 124
 - overview, 122
- using `var` for arrays, 120–121
- colon (:), class definitions, 286
- COM (Component Object Model), 243
- `Combine()` method, 444
- ComboBox control
 - WPF, 546
 - XAML, 574–575, 626
- Command Binding expressions, ViewModel
 - pattern, 625
- command pattern, WPF
 - built-in commands, 603–604
 - custom commands
 - bindings, 606–607
 - ICommand and, 608–610
 - parameters, 607–608
 - UI commands, 605–606
 - focus, 605
 - ICommand, 602
 - overview, 601–602
 - routed commands, 602–603
 - testing, 610–611
- Command Prompt window, console
 - application, 20
- Command property, Window Command Bindings XAML, 607
- CommandBase class, 621
- comments
 - method names versus, 229, 232
 - reviewing console application, 21
 - stripping out, 229
- Common Intermediate Language (IL), 380
- Common Language Runtime (CLR), 3, 380, 545
- Communication Foundation Projects, Visual Studio, 493
- `Compare()` method
 - converting string cases, 53
 - strings, 48–51
- `Compare()` operation, 49
- `CompareTo()` method, `IComparable<T>`
 - interface, 342
- comparing
 - decimal type, 33
 - floating-point numbers, 78–79
 - integers, 33
 - strings
 - `Compare()` method, 48–51
 - letter case, 51–52

- compiler
 - inferring data types, 42–43
 - type-safety, 170
 - writing properties, 272–273
- compiling program, 15
- Component Object Model (COM), 243
- composing streams, 454
- compound logical operations, 79–80
- computer languages, 11–14
- Concatenate() method
 - controlling string output manually, 63–64
 - requiring two strings, 236
- concrete class, 353–354
- concrete classes, 329
- concrete object, 339
- condition expression, 104
- Condition option, Visual Studio, 513
- conditional expression, if statement, 86
- Configuration function, System.Net namespace, 458
- Configuration Manager option, Build and Debug menus, 515
- Configuration Manager tool, Solution Explorer, 503
- configuring WCF service, 752–756
- Connect to Database tool, Visual Studio, 514
- Connect to Server tool, Visual Studio, 514
- console application
 - .NET language, 13–14
 - C# language, 12–13
 - computer languages, 11–14
 - creating source program, 15–18
 - making it do something, 19–20
 - overview, 14–15
 - reviewing
 - comments, 21
 - meat of program, 21–22
 - program framework, 20–21
 - taking it out for test drive, 18–19
 - Toolbox
 - overview, 22
 - reusing code, 23–24
 - saving code, 23
 - Visual C#, 14
- consoleapplication.csproj file, 528
- Console.WriteLine() method, 22, 67
- const data member, 225–226
- constants
 - naming, 226
 - numeric, 40
- constraints, adding, 184–186
- constructors
 - C#-provided, 274–275
 - objects and, 273
 - replacing default constructor
 - example, 276–278
 - executing constructor from debugger, 278–281
 - initializing object directly with initializer, 281–282
 - initializing object without constructor, 283–284
 - overview, 275–276
- ConstructorSavingsAccount program, 302–304
- consuming
 - ASMX service, 743–744
 - WCF service, 757–758
- container[n] command, 145
- containers, database-generic, 416
- Contains() method, 55, 126
- ContainsKey() method, Dictionary Example program, 127
- Content control, WPF, 556
- Content value, XAML, 551
- continue command, 99
- contract driven Web services, 722–724
- contravariance, 796–798
- controls, defined, 556
- conventions used in book, 7–8
- conversions
 - explicit type, 41, 81–82
 - invalid
 - avoiding with as operator, 297
 - avoiding with is operator, 296–297
- Convert() method
 - IValueConverter interface, 592–593
 - parsing numeric input, 56
- ConvertBack method, IValueConverter interface, 592–593
- converting data, WPF, 592–599
- ConvertTemperatureWithFloat program, 31

ConvertTemperatureWithRoundOff
 directory, 29
 Convert.ToDecimal() command,
 CalculateInterest program, 88
 Cookie class, 457
 cookies
 ASP.NET manages, 699
 coding for client-side storage, 697–698
 overview, 696–697
 on server, 698–699
 Copy Web design surface, Web
 construction, 714–715
 Corneliussen, Lars, 533
 Count property
 dictionaries, 128
 List<T> class, 126
 counting number, converting, 41
 coupling
 loose, 408, 721–722
 tight, 183–184
 covariance, 798
 CPU processor fault, 245
 Create Directory for Solution check
 box, 489
 Create GUID tool, Visual Studio, 514
 Create mode, 442
 CreateNew mode, 442
 CreatePackage() method, 183
 CreateRecorder() method, 338
 Cryptography namespace, 413
 CryptoStream class, 454
 .cs source files
 compiling, 379
 creating separate for each class, 220
 csConsoleApplication.vstemplate
 file, ConsoleApplication, 528
 CSS, binding styles with, 678–679
 culture option, Template directive, 540
 current object, accessing
 defined, 254–255
 explicit, 255–257
 overview, 253–254
 working without this keyword, 257–259
 Current property, 143
 currentNode member, LinkedList
 class, 163

custom commands, WPF
 bindings, 606–607
 ICommand and, 608–610
 parameters, 607–608
 UI commands, 605–606
 custom controls, 683–684
 custom delegate, 368
 Customer Maintenance Window, 615
 Customer Model Class, 616–617
 Customize tool, Visual Studio, 515
 customizing Visual Studio
 hacking project types
 item templates, 530–532
 project templates, 527–530
 overview, 517–518
 setting options
 additional options, 520–521
 Environment section, 518–519
 Language, 519–520
 snippets
 deploying, 525
 making, 523–525
 overview, 521–522
 sharing, 526
 surround, 522–523
 CustomSaveCommand, 609–610

D

DAL (Data Access Layer), 613
 Damage potential attribute, DREAD
 model, 404
 data, accessing
 how data classes fit into framework,
 417–418
 process, 418
 System.Data namespace, 416–417,
 419–433
 Data Access Layer (DAL), 613
 data binding
 additional concepts, 599–600
 binding object
 defining with XAML, 581–583
 defining with C#, 583–584
 converting data, 592–599
 dependency properties, 579–580

- data binding (*continued*)
 - editing, 584–599
 - modes, 580–581
 - overview, 670–671
 - setting up markup for, 671–673
 - using code-behind, 673–674
 - using commonly bound controls, 674–676
 - validating data, 589–592
- data code
 - basic, 429–431
 - output of visual tools, 428–429
- Data Connections panel, Visual Studio, 425
- data connections, Server Explorer panel, 507–508
- Data Encryption Standard (DES), 407
- data entry form, WPF, 567–569
- Data property, `LLNode`, 163
- Data Source Configuration Wizard, Visual Studio, 420–424
- Data Sources panel, Visual Studio, 425–426
 - connecting to data sources, 420–421, 424
 - parent/child forms, 427
- data type `T`
 - determining null value, 186
 - letting C# compiler infer, 42–43
- Data View, Class Designer, 501–502
- DataGrid control, 577
- DataGridCheckBoxColumn, 577
- DataGridComboBoxColumn, 577
- DataGridHyperlinkColumn, 577
- DataGridTemplateColumn, 577
- DataGridTextColumn, 577
- DataReader container, 418
- DataRow container, 418
- DataSet class, 350
- DataSet container
 - adapters, 416
 - defined, 418
 - overview, 415
- DataTable container
 - defined, 418
 - overview, 415
- DataTemplateSelector base class, 599
- DataRow container
 - defined, 418
 - overview, 415
- DatePicker input control, 573
- DateTime type, 38–40
- debug option, `Template` directive, 540
- debugger, executing constructor, 278–281
- Debugging menu, 515
- decide at runtime, 316. *See also* polymorphism
- decimal type
 - comparing, 33
 - declaring, 33
 - overview, 32–33
- decimal variable, 33
- DecimalBankAccount program, 268–269
- `decimal.Round()` statement, 98
- declaration, `int` variable, 26
- Declarative Flowchart Service Library project, 493
- Declarative Sequential Service library project, 493
- declared type, polymorphism, 316–317
- declaring `bool` variable, 34
- declaring namespace, 394–395
- declaring value-type variables
 - `bool` type, 34
 - calculating leap years, 38–40
 - cast, 41–42
 - character types
 - `char` variable type, 34–35
 - special chars, 35
 - string type, 35–36
 - comparing string and `char`, 37–38
 - decimal type, 32–33
 - declaring numeric constants, 40
 - floating-point variables
 - converting some more temperatures, 31
 - declaring floating-point variable, 30–31
 - examining some limitations of floating-point variables, 31–32
 - overview, 29–30
- `int`
 - overview, 26–27
 - rules for declaring variables, 27
 - types of, 27–28

- letting C# compiler infer data types, 42–43
 - overview, 25–26
 - representing fractions, 28–29
 - value type, 36–37
 - default arguments, 240–243
 - default base class constructor, 298–299
 - default constructor
 - C#-provided, 274–275
 - replacing
 - example, 276–278
 - executing constructor from debugger, 278–281
 - initializing objects, 281–284
 - overview, 275–276
 - delegates
 - anonymous methods, 368–369
 - C# events
 - handling, 374–375
 - how publisher advertises events, 370–371
 - Observer design pattern, 369–370
 - passing information to event handler, 372–373
 - publishing, 372
 - Publish/Subscribe pattern, 370
 - raising, 373–374
 - subscribing, 371–372
 - callback method
 - examples, 360–362
 - overview, 357–358
 - code, 365–366
 - defined, 358–360
 - more real-world example, 362–368
 - putting app together, 363–364
 - tracking life cycle, 366–368
 - when to use, 374
- delimiter, 58
- DemonstrateCustomConstructor program
 - executing constructor from debugger, 278–281
 - initializers, 282–283
 - listing, 276–277
 - process, 277–278
- demotion, 81
- denial of service, 403
- dependencies, 183
- Dependency Injection (DI), 625
- DependencyObject target property, 579
- DependencyProperty target property, 579–580
- deployment
 - security, 407–408
 - WCF service, 756–757
 - Web construction
 - Copy Web design surface, 714–715
 - options, 713–714
 - Package/Publish tab, 715–716
 - XML Web services, 741–742
- Deposit() method
 - BankAccount class example, 263–264
 - SimpleSavingsAccount program, 290
- Dequeue() method
 - defined, 173
 - PriorityQueue class, 181
- DES (Data Encryption Standard), 407
- DescendingEvens() method, 159
- design mode, 363
- Design view, HTML designer, 499
- designer, Visual Studio
 - Class Designer
 - Data View, 501–502
 - overview, 500–502
 - general discussion, 642–647
 - overview, 495–496
 - Web Forms, 499
 - Windows Forms, 498
 - Windows Presentation Foundation (WPF), 496–498
- designing secure software
 - decomposing components into functions, 403
 - determining what to protect, 402
 - documenting components of program, 402
 - identifying potential threats in functions, 403
 - rating risk, 404
- destructors, 305
- Details View, Visual Studio, 425–426

- development, ASP.NET
 - adding custom controls, 683–684
 - constructing user controls
 - overview, 680–683
 - Phone Number control, 681–683
- data binding
 - overview, 670–671
 - setting up markup for, 671–673
 - using code-behind, 673–674
 - using commonly bound controls, 674–676
- displaying info to user
 - images, 661–663
 - labels versus plain old text, 660–661
 - panels and multiviews, 663
 - tables, 663–664
- input from user
 - submitting input with Submit buttons, 670
 - using multiple-item selection controls, 668–669
 - using other kinds of input controls, 669–670
 - using single-item selection controls, 666–668
 - using text input controls, 664–666
- site accessibility
 - control features, 680
 - design considerations, 680
 - overview, 679
- styling controls
 - binding styles with CSS, 678–679
 - setting control properties, 677–678
- Device Independent Units (DIUs), 556
- DI (Dependency Injection), 625
- dictionaries, 126–128
- Dictionary<TKey,TValue> class, 123
- DictionaryExample program, 127
- DirectDeposit() method,
 - SimpleSavingsAccount program, 292–294
- Directives, NotePad, 537
- Directory class, 139, 444
- Directory.GetCurrentDirectory() method, 139
- DirectoryInfo object, 139
- Discoverability attribute, DREAD model, 404
- display only controls, XAML, 570–572
- DisplayArray() method
 - creating own interface, 341
 - foreach loop, 346
- DisplayRatio() method, 245
- DisplayRoundedDecimal() methods, 241
- Dispose() method
 - destructors versus, 305
 - using statement, 447
- DIUs (Device Independent Units), 556
- divide (/) operator, 74
- DLR. *See* Dynamic Language Runtime (DLR)
- DNS class, 457
- do . . . while loop, 99
- Dock Panel, WPF, 559–560
- Documents directory, 17
- dot operator, 221
- Dotfuscator Community Edition tool, Visual Studio, 515
- “do-to-each” trick, 321
- double variable, 30
- DoubleBankAccount program, 266–267
- Double.Epsilon constant, 78
- doublesArray array, 114
- doublesArray variable, 111
- downcast, 82
- Download class, 457
- DownloadFile method, 461
- Draw() method, 321
- drawing board, System.Drawing namespace, 476–478
- DREAD model, 403–405
- driver, 381–382
- driver program, testing library with, 385–386
- DSL Tools, 534
- DumpBuffer() method, 140
- DumpHex() method, 139–140
- duration, defined, 39
- duration variable, CalculateInterestTable program, 97
- Dynamic Data Entities Web Application project, 492
- Dynamic Data Linq to SQL Web Application project, 492
- dynamic import, Interop, 790–791
- dynamic keyword, 43

Dynamic Language Runtime (DLR)
 C#, 778–780
 overview, 776–777
 Ruby, 777–778
 dynamic programming
 DLR
 C#, 778–780
 overview, 776–777
 Ruby, 777–778
 examples, 774–775
 making static operations dynamic, 775
 overview, 772–774
 shifting C# toward dynamic typing, 770–772
 dynamic type, 43
 dynamic typing, 2

E

e parameter, event handler, 372
 early binding, 316
 editing data, WPF, 584–599
 element binding, 589
 elevation of privilege, 403
 else statement, 89–90
 e-mailing status report, 462–465
 embedded statement, 91
 Empty Project, 491
 empty string, 36
 Encapsulate Field tool, Visual Studio, 516
 encodings, Unicode, 438
 encrypting information, 407
 EndPoint class, 457
 Enqueue() method, 173, 180–181
 Enroll() method, 257
 Entity Data Model Wizard, Visual Studio, 432
 entity framework
 generating, 432–433
 overview, 431–432
 writing code, 433
 enumerator, 141. *See also* iterators
 Environment Design Time Environment
 (EnvDTE), 534
 Environment section, Visual Studio
 changing security settings, 536–537
 creating template from text file, 537–539
 keyboard commands, 519
 overview, 518–519
 start page, 519
 Epsilon constant, 78
 equals symbol (=), 26, 242
 Error List window, 312–313
 errors
 code guidelines for handling, 199–200
 runtime, 187, 296
 event handler
 coding behind, 601
 passing information to, 372–373
 Event Viewer, 507
 events, C#
 handling, 374–375
 how publisher advertises, 370–371
 Observer design pattern, 369–370
 passing information to event handler,
 372–373
 publishing, 372
 Publish/Subscribe pattern, 370
 raising, 373–374
 subscribing, 371–372
 Excel 2007 Add-in project, 493
 Excel 2007 Template project, 493
 Excel 2007 Workbook project, 492
 exceptions
 assigning multiple catch blocks, 196–197
 example, 193–196
 exception mechanism
 catch blocks, 189
 finally blocks, 190
 overview, 187–188
 thrown exceptions, 190–192
 try blocks, 189
 handling strategy
 analyzing method for possible
 exceptions, 200–202
 guidelines for code that handles errors,
 199–200
 planning guide, 198
 which methods throw which
 exceptions, 203
 “last chance” exception handler, 203–204
 purpose of, 192–193
 throwing, 192
 ExceptWith() method, 132
 executable assembly, 379, 381
 executable program, 12
 Execute event handler, 602
 Executed property, Window Command
 Bindings XAML, 607

- explicit constraint, 184
- explicit demotion, 82
- explicit type conversion, 82
- Exploitability attribute, DREAD model, 404
- Express version, Visual Studio, 4, 482–483
- expression types, matching
 - assigning types, 82–83
 - calculating type of operation
 - explicit type conversion, 82
 - implicit type conversion, 81–82
 - overview, 80–81
- extending classes, 288, 391–392
- extension methods, 71, 158
- external methods, 263
- External Tools tool, Visual Studio, 515
- Extract Interface tool, Visual Studio, 516
- Extract Method refactoring, Visual Studio, 232–233
- Extract Method tool, Visual Studio, 516

F

- `FactorialException` program, 193
- factories
 - defined, 179
 - generics, 183–184
- factoring, abstract class, 322–327
- factory methods, 338, 355
- FAQ (Frequently Asked Questions) list, 8
- file access, 435
- File Transfer Protocol (FTP), 455
- `FileInfo` class, 139
- files
 - downloading from Internet, 460–462
 - Solution Explorer panel, 504
- `FileStream` class
 - exploring streams, 453–454
 - readers, 436–437, 452–453
 - `StreamReader`, 448–452
 - streams, 435–436
 - streamwriting
 - example, 439–440
 - methods and blocks, 442–445
 - overview, 438–439
 - `StreamWriter`, 441–442
 - using statement, 445–448
 - writers, 436–437, 452–453

- `FileWeb` class, 457
- Filter option, Breakpoints window, 514
- finally block
 - exceptions, 190
 - `StreamReader` class, 450
- finally keyword, 187–188
- `Find()` method, 148
- `FindEmpty()` method, 148
- fire hose, 418
- fixed sizing, 562
- `FixedArrayAverage` program, 112
- fixed-value array, 110–112
- flexible dependencies, 353
- flexible lists, 109
- float variable, 30
- floating-point variables
 - comparing numbers, 78–79
 - converting temperatures, 31
 - declaring, 30–31
 - limitations
 - calculation speed, 32
 - comparing numbers, 31–32
 - counting, 31
 - range, 32
 - overview, 29–30
- flow control, program, 86
- Flowchart Workflow Console application
 - project, 493
- flushing streams, 445
- focus, WPF, 605
- font, character variables, 35
- `FontSize` property, XAML, 552
- for loop
 - iterating through `numElements` items, 114
 - looping specified number of times, 104–106
 - searching string values, 52
 - `VariableArrayAverage` program, 114
- foreach loop
 - `Draw()` method, 321
 - `IEnumerator` interface, 142
- iterating `StringChunks` collection, 157
- iterating through collections, 115–116
- iterating through subarrays, 60
- looping specified number of times, 105
- looping through string, 53

form class, 364
 Format() method, 65
 formatCommand variable, 68
 Formatting panel, Visual Studio, 520
 formatting strings, 65–69
 FORTRAN, 234
 Fowler, Martin, 612
 fractions, representing, 28–29
 Frequently Asked Questions (FAQ) list, 8
 FTP (File Transfer Protocol), 455
 FtpWeb class, 457
 fully qualified name, 397–398
 functional programming, 2

G

Gamma, Erich, 601
 garbage collector, 222, 305
 generic constraints, 185
 generics
 contravariance, 796–798
 covariance, 798
 efficiency, 171
 overview, 795–796
 type-safety, 170–171
 variance, 796
 writing
 adding constraints, 184–186
 code, 179–180
 determining null value for data
 type T, 186
 factories, 183–184
 Main() method, 178–179
 OOPs, 172–176
 overview, 171–172
 Package class, 177–178
 PackageFactory, 183
 PriorityQueue class, 180–182
 get() accessor, 160–161
 get operation, 272
 GetAccountNumber() method,
 BankAccount class example, 266
 GetBalance() method, BankAccount
 class example, 266
 GetEnumerator() method, 144, 162
 GetFileList() method, 139

Get...List() method, 139
 GetNext() method, 142
 GetString() method, 321
 GetString() method, BankAccount
 class example, 266
 GetType() method, 43
 GetX() method, BankAccount class
 example, 270
 global namespace, 393
 Go to Definition option, Visual Studio, 428
 goto statement, 107–108
 Graphic Development Environment, 495
 graphical programming, 363
 graphical user interface (GUI), 14
 graphics, System.Drawing
 namespace, 470
 greater than (>) operator, 77
 greater than or equal to (>=) operator, 77
 Grid Panel, WPF
 content alignment within Content
 Controls, 565
 horizontal and vertical alignment within
 parent container's layout slot, 565
 margin versus padding, 565
 RowSpan and ColumnSpan, 564–565
 shared size groups, 565–566
 sizing rows and columns, 562–564
 Grid.Column property, XAML, 552
 Grid.Row property, XAML, 552
 GUI (graphical user interface), 14

H

hacking project types
 item templates, 530–532
 Project templates, 527–530
 handling strategy, exceptions
 analyzing methods for exceptions,
 200–202
 guidelines for code that handles errors,
 199–200
 planning guide, 198
 which methods throw which
 exceptions, 203
 hand-out-an-interface technique, 350
 Hanselman, Scott, 519, 533

- hard-coding, 92
- hardware, Visual Studio, 487
- HAS_A relationship
 - determining when to use, 293–294
 - gaining access to BankAccount using containment, 292–293
 - with interfaces, 354–355
 - versus IS_A relationship, 291–292
 - making flexible dependencies through interfaces, 353
 - overview, 293
- hash, computing, 127
- HashSet<T> class, 123, 131
- hashtable, 417
- Header element, Visual Studio, 523–524
- Helm, Richard, 601
- helper classes, 392
- hex dump, 136, 138
- hexadecimal system, 138
- hiding base class method
 - accidentally hiding, 312–313
 - general discussion, 309–311
 - making hiding approach better than adding simple test, 312
- HidingWithdrawalPolymorphically program, 315
- hierarchy, operator, 75
- high-level computer language, 11
- Hit Count option, Breakpoints window, 513
- Holzner, Steve, 369
- horizontal split, WPF designer, 497
- Horizontal Stack Panel, 558
- HorizontalAlignment property, 565
- hostspecific option, Template directive, 540
- Howard, Michael, 464
- HTML designer, 499
- Http class, 457
- human-readable data, 438
- I**
- ICommand, WPF, 602, 608–610
- IComparable interface, 340
- icons used in book, 6–7
- id argument, Init() method, 255
- IDataErrorInfo interface, 589–591
- IDE (Integrated Development Environment), 481, 495
- IDisplayable interface, 340–341
- IDisposable interface, 447
- IEnumerable interface, 340
- IEnumerator interface, 141
- if clause, 89
- if statement
 - else statement, 89–90
 - nesting, 90–92
 - overview, 86–89
- IL (Common Intermediate Language), 380
- IList interface, 149
- Image control, XAML, 570–571
- images
 - displaying info to user, 661–663
 - how drawing classes fit into framework, 472–473
 - System.Drawing namespace
 - brushes, 471
 - drawing board, 476–478
 - getting started, 473–475
 - graphics, 470
 - pens, 470–471
 - setting up project, 475–476
 - text, 471–472
- Immediate window, Visual Studio, 512–513
- immutability of strings, 47
- implicit promotion, 81
- implicit type conversion, 41, 81–82
- Import and Export Settings tool, Visual Studio, 515
- import directive, T4, 541
- IMyInterface constraint, 185
- include directive, T4, 540
- increment operator, 76–77
- index, fixed-value array, 111
- indexers, accessing collection, 145–149
- IndexOf() method, 54, 126
- IndexOfAny() method, 54, 63–64
- indirection, defined, 213
- indivisibility, string, 46–47
- infinite loop, 98
- InfoPath 2007 Add-in project, 493

- information disclosure, 403
- inheritance
 - BankAccount class example
 - overview, 288–291
 - updated, 302–306
 - base class constructor
 - base keyword, 301–302
 - invoking default, 298–299
 - passing arguments to, 300–301
 - class, 286–287
 - databases and, 431
 - HAS_A relationship
 - determining when to use, 293–294
 - gaining access to BankAccount using
 - containment, 292–293
 - versus IS_A relationship, 291–292
 - overview, 293
 - importance of, 287–288
 - invalid casts at run time, 295–296
 - invalid conversions
 - avoiding with as operator, 297
 - avoiding with is operator, 296–297
 - IS_A relationship
 - determining when to use, 293–294
 - gaining access to BankAccount using
 - containment, 292–293
 - versus HAS_A relationship, 293
 - overview, 291–292
 - object class, 297–298
 - overview, 285
 - substitutable classes, 294–295
- inheritance of convenience, 325
- InheritanceExample program, 286–287
- inherited method, overloading
 - calling back to base, 313–314
 - hiding base class method, 309–313
 - overview, 308–309
- InheritingAConstructor program, 298–299
- inherits option, Template directive, 540
- Init() method
 - id argument, 255
 - name argument, 255–256
- InitBankAccount() method
 - BankAccount class example, 263, 275
 - SimpleSavingsAccount program, 291
- initExpression expression, 104
- initializing
 - arrays, 128
 - C# array, 115
 - collections, 129–130
 - variables, 26
- InitSavingsAccount() method,
 - SimpleSavingsAccount
 - program, 291
- InkCanvas layout panel, 570
- inlining code, 448
- inner loop, 106
- INotifyCollectionChanged interface,
 - WPF, 587
- INotifyPropertyChanged interface,
 - WPF, 587
- input, string
 - handling series of numbers, 58–60
 - joining array of strings into one
 - string, 60
 - parsing numeric input, 56–58
 - trimming excess white space, 55–56
- input controls, XAML, 572–574
- input from user
 - submitting input with Submit
 - buttons, 670
 - using multiple-item selection controls,
 - 668–669
 - using other kinds of input controls,
 - 669–670
 - using single-item selection controls,
 - 666–668
 - using text input controls, 664–666
- InputInterestData() method
 - function of, 232
 - refactoring, 235
- InputPositiveDecimal()
 - method, 235
- inserting element in array, 121
- install directory, Visual Studio, 524
- installing
 - sample schemas, 420
 - Visual Studio, 486–487
- instance members
 - defined, 250
 - as nonstatic members, 225

- instance methods
 - defined, 227
 - general discussion, 250–252
 - invoking, 228
- instances, defined, 217
- instantiating
 - defined, 123, 217
 - new, empty list for `string` type, 125
- `int` integer, 28
- `int` type, strings, 45
- `int` variable
 - averaging, 110
 - overview, 26–28
 - rules for declaring variables/nl, 27
 - types of, 27–28
- integers
 - comparing, 33
 - truncation, 29
- Integrated Development Environment (IDE), 481, 495
- IntelliSense
 - autocompleting with, 510–511
 - tool tips, 203
 - Visual Studio, 520
- intercepting request
 - digging up request data, 686–689
 - using information from requests, 689
- interface, C#. *See also names of specific interfaces*
 - CAN_BE_USED_AS relationship, 333–334
 - creating at home, 340–341
 - defined, 338
 - hiding behind, 348–350
 - `IComparable<T>` interface, 341–343
 - implementing, 335–336
 - importance of, 336
 - inheriting, 351–352
 - `Main()` method, 346
 - managing changes in object-oriented programs
 - abstract class, 353–354
 - concrete class, 353–354
 - HAS_A relationship, 354–355
 - making flexible dependencies, 353
 - overview, 352–353
 - mixing inheritance and interface
 - implementation, 336–337
 - naming, 336
 - unifying class hierarchies, 346–348
 - usefulness, 337–338
 - using as base type of array or collection, 339
 - using as method return type, 338
 - using as more general type of object
 - reference, 339
 - using predefined interface types, 339–340
- interface, Visual Studio
 - Building menu, 515
 - Code View
 - autocompleting with IntelliSense, 510–511
 - exploring auxiliary windows, 512–514
 - outlining, 511–512
 - overview, 509–510
 - Debugging menu, 515
 - designer
 - Class Designer, 500–502
 - overview, 495–496
 - Web Forms, 499
 - Windows Forms, 498
 - Windows Presentation Foundation (WPF), 496–498
 - panels
 - Class View, 508–509
 - Properties, 504–505
 - Server Explorer, 506–508
 - Solution Explorer, 502–504
 - Toolbox, 505–506
 - Refactor menu, 515–516
 - Tools menu, 514–515
- internal access keyword, 387–390
- internal keyword, 380
- internal members, 264–265
- internal method, 616
- internal protected member, 264–265
- Internet
 - how net classes fit into framework, 457–458
 - `System.Net` namespace, 456, 459–468
- Interop
 - overview, 789–790
 - Ref statement, 793–794
 - using dynamic import, 790–791
 - working without Primary Interop Assemblies (PIAs), 791–793

intersection operation, 130
 IntersectWith() method, 133
 intrinsic variable, 36
 invalid cast, 295–296
 invalid conversion
 avoiding with as operator, 297
 avoiding with is operator, 296–297
 InvokeBaseConstructor program,
 301–302
 I/O request, 437
 IP class, 457
 IPrioritizable interface, 177–178
 IrDA class, 457
 IRecordable interface, 335–336
 IRONPython, 494
 IRONRuby, 494
 is operator
 as operator versus, 297
 avoiding invalid conversions, 296–297
 avoiding runtime errors, 296
 polymorphism, 318
 type-safety, 170
 IS_A relationship
 determining when to use, 293–294
 gaining access to BankAccount using
 containment, 292–293
 versus HAS_A relationship, 293
 inheritance and, 287
 overview, 291–292
 IS_NOT_A relationship, 287
 IsAllDigits() method, 57–58
 IsChecked property, 572
 IsSharedSizeScope property, 565
 item templates, hacking, 530–532
 ItemsControls, WPF, 599
 iterating
 days of month, 154–155
 through collections, 135
 through directory of files, 135–141
 through numElements items, 114
 iterator block
 looping around
 days of month, 154–155
 general discussion, 150–154
 syntax, 156–158
 shapes and sizes, 158–161

iterators
 accessing collections, 141–143
 letting C# access data foreach container,
 143–145
 placing, 161–167
 IteratorX naming convention, 142–143
 IValueConverter interface, 592–593

J

Java programming language, 13
 Johnson, Ralph, 601
 Join() method, 60

K

key, 148
 KeyedArray virtual array class, 146–147
 _keys array, 147–148
 Keys property, dictionaries, 128
 kludge, 191

L

Label control, 571, 589
 labels, 660–661
 lambdas, 374
 Language Integrated Query, 415
 Language option
 Template directive, 540
 Visual Studio, 519–520
 “last chance” exception handler, 203–204
 last in wins problem, 416
 LastChildFill setting, Dock Panel, 560
 LastIndexOf() method, 54
 LastIndexOfAny() method, 54
 late binding, 316. *See also* polymorphism
 layout panels, WPF
 Canvas, 560–561
 data entry form, 567–569
 Dock, 559–560
 Grid, 562–566
 specialized, 569–570
 Stack, 557–558
 Uniform Grid, 561–562
 Wrap, 559

- leap years, calculating, 38–40
 - LeBlanc, David C., 464
 - left-hand operator, 83
 - Length property, C# array, 114–115
 - less than (<) operator, 77
 - less than or equal to (<=) operator, 77
 - level of abstraction, 208
 - libraries
 - access keywords
 - internal, 387–390
 - protected, 390–392
 - protected internal, 392
 - putting classes into
 - adding second project to existing solution, 383
 - creating classes for library, 384–385
 - creating projects for class library, 382
 - creating stand-alone class library, 382–383
 - using class library from program, 386
 - using driver program to test library, 385–386
 - single program
 - dividing into multiple assemblies, 379–381
 - dividing into multiple source files, 378–379
 - life cycle, delegates, 366–368
 - like-typed variables, bundling, 215
 - LinkedList<T> class, 123
 - list collection, 124–126
 - list node, 163
 - List<T> class, 123, 169, 350
 - list-based controls, XAML, 574–577
 - listeners, 370
 - Literal element, Visual Studio, 525
 - local variables, constructor, 274
 - Location box, Visual Studio, 383, 489
 - Location option, Breakpoints window, 513
 - logging network activity, 465–468
 - logical comparison operators
 - comparing floating-point numbers, 78–79
 - compound logical operations, 79–80
 - overview, 77–78
 - long integer, 27, 28
 - LookUpWordBreak() method, 234
 - looping
 - around iterator block
 - days of month, 154–155
 - general discussion, 150–154
 - shapes and sizes, 158–161
 - syntax, 156–158
 - break command, 99–100
 - do . . . while loop, 99
 - nesting, 106–107
 - placing iterators, 161–167
 - scope rules, 103–104
 - specified number of times with for loop, 104–106
 - through strings, 53–54
 - until you get it right, 100–103
 - while, 95–99
 - LoopThroughFiles program, 135–139
 - loosely coupled
 - Web Forms application, 408
 - Web services, 721–722
-
- ## M
-
- machine language, 11
 - Macros tool, Visual Studio, 514
 - Mail function, System.Net namespace, 458
 - Main() method
 - AlignOutput program, 62
 - catch block, 451
 - demonstrating custom LinkedList class, 150
 - exceptional version, 194
 - finding exceptions, 203–204
 - generics, 178–179
 - Indexer class, 149
 - iterating collections using foreach loop, 154
 - iterating StringChunks collection, 156
 - nested loops, 167
 - MainWindow.xaml file, 548, 551
 - malicious SQL code, 409
 - managed services, Server Explorer panel, 507
 - markup, setting for data binding, 671–673

- Master Pages
 adding content, 710–711
 making master page, 709–710
- maximum calculation, 89–90
- MbUnit, Gallio, 610
- MemoryStream stream class, 453
- MenuItem.Click function, 472
- metadata, 380
- method call, 88
- methods. *See also* polymorphism
 analyzing for possible exceptions, 200–202
 anonymous, 368–369
 declaring virtual and overriding it, 319–321
 defining, 227–229
 determining which throw which exceptions, 203
 example, 229–236
 expanding full name, 252–253
 implementing default arguments, 240–243
 importance of, 234
 instance, 250–252
 matching argument definitions with usage, 238–239
 naming, 229, 232
 with no value, 244–246
 overloading, 239–240
 passing arguments to, 236–237
 passing multiple arguments to, 237–238
 passing object to, 247–249
 returning value via return postage, 243–244
 static, 249–250
 streamwriting, 442–445
 supplying multiple versions of, 240–241
 visibility and accesibility in namespace, 396–397
- Microsoft Developer Network (MSDN)
 version, Visual Studio, 484–485
- Microsoft Message Queue, 507
- Microsoft Report Viewer, 472
- Mime function, System.Net namespace, 458
- MixingStaticAndInstanceMethods program, 257–259
- mocking framework, 618
- Model, ViewModel pattern, 613
- Model View Controller development for Windows Presentation Foundation (MVC4WPF), 494
- Model View Controller (MVC), 494
- module, 264, 379–381
- modulo (%) operator, 74, 140
- Mono, 1
- moreColors array, 131–132
- MoveNext() method, IEnumerator interface, 143
- MSDN (Microsoft Developer Network)
 version, Visual Studio, 484–485
- multidimensional arrays, 116
- multiple assemblies, dividing single program into, 379–381
- multiple inheritance, 336
- multiple source files, dividing single program into, 378–379
- multiple-item selection controls, 668–669
- multiplication (*) operator, 73–74, 81
- multiviews, panels, 663
- MVC (Model View Controller), 494
- MVC4WPF (Model View Controller development for Windows Presentation Foundation), 494
- MyBaseClass constraint, 185

N

- name argument, Init() method, 255–256
- Name box, New Project dialog box, 489
- named iterator, 158
- named parameters, 786–787
- namespaces
 putting classes into
 declaring namespace, 394–395
 fully qualified names, 397–398
 overview, 392–393
 relating namespaces to access keyword story, 395–397
 System.Security, 412–413

- naming conventions, 38
- nesting
 - if statement, 90–92
 - namespaces, 394
- .NET Common Language Runtime (CLR), 3
- .NET dictionaries, 127
- .NET framework
 - cookies
 - ASP.NET manages, 699
 - coding for client-side storage, 697–698
 - overview, 696–697
 - on server, 698–699
 - managing files, 695–696
 - navigating with site maps
 - adding site map, 692–694
 - navigating site with SiteMap, 694–695
 - overview, 685
 - securing with ASP.NET
 - changing trusts, 691–692
 - fixing problems, 692
 - overview, 690–691
 - surfing Web streams
 - altering content sent to clients, 689–690
 - intercepting request, 686–689
 - tracing with TraceContext, 699–701
- .NET language, 13–14
- .NET Software Development Kit (SDK), 3, 380
- network activity, logging, 465–468
- network status, checking, 459–460
- NetworkCredential class, 457
- NetworkInformation function,
 - System.Net namespace, 458, 460
- networking, 455. *See also* Internet
- NetworkStream class, 454
- new() constraint, 185
- New line character (\n), 35
- new operator, 217
- New Project dialog box, Visual Studio, 383, 395, 488–489, 548
- New Project window, Visual Studio 2010, 15
- newline character (\n), 220
- NextAccountNumber property, 271
- nondeterministic destruction, 305
- noninteger index, 148

- nonrecursive algorithm, 196
- nonstatic methods, 228
- nonvoid methods, 244
- not (!) operator, 79
- not equal to (!=) operator, 77
- notational C#, 48
- NotifyPropertyChanged
 - method, 616
- n-tier code, 655–656
- Null character (\0), 35
- null keyword, 186
- null object, 221
- null value
 - determining for data type T, 186
 - strings, 55
- num variable, 217
- numberInput variable, 68
- numeric constants, 40
- numeric input, parsing, 56–58
- NUnit testing tool, 7

O

- obfuscation, 401
- object class, 297–298
- object properties, 224
- object role modelers, 431
- Object-based arrangement, 134
- object-based languages, 317
- ObjectInitializers program
 - example, 284
- object-oriented (OO) programming
 - abstraction concept
 - overview, 207–208
 - preparing object-oriented programming, 209
 - preparing procedural programming, 208
 - access control concept, 212
 - C# support, 212–213
 - classification concept, 209–210
 - classifying, 210–211
 - managing changes with interface
 - abstract class, 353–354
 - concrete class, 353–354

- HAS_A relationship, 354–355
 - making flexible dependencies, 353
 - overview, 352–353
 - polymorphism and, 316–317
 - usable interfaces concept, 211
 - objects
 - abstract class and, 330
 - accessing members of, 218
 - classes versus, 273
 - constructors and, 273
 - defining, 217–218
 - discriminating between, 220–221
 - initializing in constructors, 281–284
 - passing to method, 247–249
 - program example, 218–220
 - ObservableCollection interface, WPF, 587
 - Observer design pattern, 369–370
 - Office projects, Visual Studio, 492–493
 - On the Web icon, 7
 - oneth element, 111
 - OneTime binding mode, 580
 - OneWay binding mode, 580–581
 - OneWayToSource binding mode, 581
 - OO programming. *See* object-oriented (OO) programming
 - operating orders, 74–75
 - operating systems, Visual Studio, 486–487
 - operators. *See also* names of specific operators
 - simple, 73–74
 - smooth
 - arithmetic, 73–77
 - logical comparisons, 77–80
 - matching expression types, 80–83
 - optional parameters
 - output, 785–786
 - overview, 782–784
 - reference types, 784–785
 - options, Visual Studio
 - additional options, 520–521
 - Environment section, 518–519
 - Language, 519–520
 - Options dialog box, Visual Studio, 518
 - Options tool, Visual Studio, 515
 - or (^) operator, 79
 - OR (||) operator, 102
 - Oracle, 350
 - order of precedence, 75
 - outlining code, 511–512
 - Outlook 2007 Add-in project, 493
 - output, string
 - controlling with Concatenate() method, 63–64
 - controlling with Split() method, 64–65
 - controlling with Trim() and Pad() methods, 61–62
 - output directive, T4, 539–540
 - Output() method
 - abstract classes, 329–330
 - example using, 236–237
 - output parameters, 785–786
 - Output window, Visual Studio, 512
 - OutputBanner() method, 259
 - OutputBannerAndName() method, 259
 - OutputFormatControls program, 67–68
 - OutputInterestTable() method, 232–233
 - OutputName() method, 248, 250, 259
 - overloading
 - inherited method
 - calling back to base, 313–314
 - general discussion, 308–313
 - methods, 239–240
 - resolution, 787
 - override keyword, 319, 335
-
- p
-
- Package class
 - implementing IPrioritizable interface, 177–178
 - specifying possible priorities, 177
 - PackageFactory, generics, 183
 - Package/Publish tab, Web construction, 715–716

- Pad() method, 61–62
- PadRight() method, AlignOutput program, 62
- PaintBoard method, 477
- paired brackets [], 110–111
- panels
 - multiviews and, 663
 - Toolbox, 505–506
 - Visual Studio
 - Class View, 508–509
 - Properties, 504–505
 - Server Explorer, 506–508
 - Solution Explorer, 502–504
 - WPF, 556
- parameters
 - defined, 236–237
 - named, 235, 786–787
 - optional
 - output parameters, 785–786
 - overview, 782–784
 - reference types, 784–785
 - overload resolution, 787
 - overview, 781
 - WPF custom commands, 607–608
- parent Grid, IsSharedSizeScope property, 565
- parent/child form, 426–428
- parentheses
 - constructors, 284
 - operating orders, 75
- ParseSequenceWithSplit program, 59–60
- parsing numeric input, 56–58
- Pascal-cased names, 38
- PasswordBox input control, 572
- Path class, 444
- Path.Combine() method, 444
- pens, System.Drawing namespace, 470–471
- Permissions namespace, 413
- persisting objects, 438
- Phone Number control, 681–684
- PIAs (Primary Interop Assemblies), 791–793
- Pixel definition, Grid Panel, 562
- plain old text, 660–661
- Policy namespace, 413
- PolymorphicInheritance program, 319–321
- polymorphism
 - abstract class
 - BankAccount class example, 327–328
 - defined, 328–330
 - factoring, 322–327
 - objects and, 330
 - declaring method virtual and overriding it, 319–321
 - “do-to-each” trick, 321
 - generics, 171
 - overloading inherited method
 - calling back to base, 313–314
 - hiding base class method, 309–313
 - overview, 308–309
 - overview, 213, 314–316
 - sealed class, 330–331
 - ToString() method, 321–322
 - using declared type, 316–317
 - using is keyword to access hidden method polymorphically, 318
- Pop() method, 142
- popping operation, 133
- Portable .NET, 1
- PostBack, Web servers, 636–639
- postincrement operator, 76, 106
- PowerPoint 2007 Add-in project, 493
- predefined delegate, 368
- preincrement operator, 76, 106
- preventing
 - script exploits, 410–411
 - SQL Injection attacks, 410
- Preview Changes window, Visual Studio, 232
- Preview Method Signature box, Visual Studio, 232
- Primary Interop Assemblies (PIAs), 791–793
- Principal namespace, 413
- PrintForm component, Toolbox, 472
- Priority enum, Package class, 177

- PriorityQueue class
 - Dequeue() method, 181
 - Enqueue() method, 180–181
 - remaining members, 182
 - TopQueue() utility method, 182
 - underlying queues, 180
- PriorityQueue's Enqueue() method, 180–181
- private access specifier, 335
- private members
 - access to, 264–265
 - BankAccount class example, 263–264
 - Unified Modeling Language (UML), 323
- procedural programming
 - object-oriented programming versus, 210–211
 - overview, 208
- ProcessAmount() method,
 - SimpleSavingsAccount program, 295–296
- processor upchuck, 245
- Professional version, Visual Studio, 483
- program
 - single, 378–381
 - using class library from, 386
- program flow
- goto statement, 107–108
- if statement
 - else statement, 89–90
 - nesting, 90–92
 - overview, 86–89
- looping
 - break command, 99–100
 - do . . . while loop, 99
 - scope rules, 103–104
 - specified number of times with for loop, 104–106
 - until you get it right, 100–103
 - while, 95–99
- nesting loops, 106–107
- overview, 85–86
- switch statement, 92–94
- program framework, console application, 20–21
- program.cs file, ConsoleApplication project, 528
- programming by intention, 230
- ProgrammingToAnInterface
 - program, 186
- ProgressBar control, 572
- Project 2007 Add-in project, 493
- Project Dependencies tool, Solution Explorer, 503
- project file
 - defined, 379
 - Solution Explorer, 504
- projects
 - creating for class library, 382
 - defined, 15
 - hacking
 - item templates, 530–532
 - project templates, 527–530
 - Visual Studio
 - categories, 491–494
 - New Project dialog box, 488–489
 - Solution Explorer panel, 503–504
 - solutions and, 489–491
- Promote Local Variable to Parameter
 - refactoring, Visual Studio, 233
- properties
 - classes
 - accessors with access levels, 273
 - letting compiler write properties, 272–273
 - overview, 270–271
 - side effects, 272
 - static properties, 271
 - syntax, 271
- Properties panel, Visual Studio, 504–505
- Property Element tag, 550
- Property Pages tool, Solution Explorer, 503
- PropertyChanged event, INotify
 - PropertyChanged interface, 587
- protected access keyword, 390–392
- protected internal access keyword, 392
- protected keyword, 380
- protected methods, 616
- ProtectedInternalLimitsAccess
 - example, 392
- protocols, 455
- pseudocode, 48

- public access specifier, 335
- public interface, 338
- public members
 - access to, 264–265
 - Unified Modeling Language (UML), 323
- public method, 390
- public modifier, 217
- publishing C# events, 372
- Publish/Subscribe pattern, 370
- pulling strings. *See* strings
- Push() method, 142
- pushing operation, 133

Q

- questioning client
 - getting information back from client, 633–634
 - overview, 632–633
 - scripting client, 633
 - understanding weaknesses of browser, 634–636
- Queue class, 173
- queue data structure, 172
- Queue<T> class, 123
- queues, 109

R

- RAD (Rapid Application Development)
 - data tools, 425
- RadioButton input control, 573
- raising events, 372
- Random class, 183
- Rapid Application Development (RAD)
 - data tools, 425
- reachable objects, 222
- read() call, synchronous I/O, 437
- Read() method, 220
- readers, FileStream class, 436–437, 452–453
- ReadFileToConsole() method, FileRead program, 451
- ReadLine() command, Calculate Interest program, 88

- ReadLine() method
 - comparing strings, 50
 - parsing numeric input, 56
 - Read() method versus, 220
 - StreamReader class, 437
- readonly data member, 225–226
- ReadToEnd() method, StreamReader class, 437
- real number, 29
- Rebuild option, Build and Debug menus, 515
- recursive algorithm, 196, 313
- RedisplayDocument() method, 234
- ref keyword, 248
- Ref statement, Interop, 793–794
- Refactor menu, Visual Studio, 232, 515–516
- refactoring
 - defined, 229
 - InputInterestData() method, 235
 - OutputInterestTable() method, 232–233
- reference types
 - generics, 170
 - optional parameters, 784–785
- references
 - class, 221–222
 - defined, 36
- ReferencingThisExplicitly program, 256–257
- Reflector tool, 7, 380
- regions, 18
- registration service, WCF, 750–752
- relational databases, 350
- relative values, WPF layout, 556
- relaying, 462
- Remember icon, 6
- Remove button, Visual Studio, 525
- Remove Parameters tool, Visual Studio, 516
- RemoveWhiteSpace sample program, 63–64
- removing
 - element from array, 121
 - white space, 121
- Rename tool, Visual Studio, 515

- Reorder Parameters refactoring, Visual Studio, 233
 - Reorder Parameters tool, Visual Studio, 516
 - Reparagraph() method, 234
 - repetitive coding, replacing, 535
 - Replace() method
 - removing white space, 63
 - replacing characters, 63
 - Replace() operation, 69
 - Reports Application project, 494
 - Repository interface, 619
 - Representational State Transfer (ReST)
 - changing WCF service to use, 762–765
 - guiding principles, 760–761
 - overview, 759–760
 - representing fractions, 28–29
 - Reproducibility attribute, DREAD model, 404
 - repudiation of action, 403
 - request, intercepting, 686–689
 - Request For Comments (RFC), 456
 - Reset() method, IEnumerator interface, 142
 - resources, scoped, 548
 - ReST. *See* Representational State Transfer (ReST)
 - restricting access to class members
 - overview, 261–262
 - public example, 262–264
 - security levels, 264–265
 - reviewing console application
 - comments, 21
 - meat of program, 21–22
 - program framework, 20–21
 - RFC (Request For Comments), 456
 - RhinoMocks tool, 618–619
 - Richter, Jeffrey, 199
 - right-hand operator, 83
 - Round() method, DecimalBankAccount program, 269
 - rounding, 29
 - routed commands, WPF, 602–603
 - RoutedCommand, 603
 - RoutedCommand class, ICommand interface, 602
 - RoutedUICommand, 603
 - Ruby, 777–778
 - runtime error, 187, 296
-
- ## S
-
- SalesRegion Model Class, 617
 - same value (==) operator, 77
 - SAO (Software Architecture Overview) diagram, 402
 - SavingsAccount class
 - ConstructorSavingsAccount program, 304
 - SimpleSavingsAccount program, 290–296
 - sbyte integer, 28
 - Schardt, James A., 323
 - scope rules, looping, 103–104
 - scoped resources, 548
 - script exploits
 - overview, 410
 - preventing, 410–411
 - scripting, 653–655
 - scripting client, 633
 - SDK (Software Development Kit), 3, 380
 - SDLC (Software Development Life Cycle), 402
 - sealed class, 330–331
 - sealed keyword, 330–331
 - sealing, 391
 - searching strings, 54–55
 - security. *See also* writing secure code classes, 264–265
 - text template environment, 536–537
 - Security function, System.Net namespace, 458
 - Security namespace, 412
 - Select the Data Objects screen Visual Studio, 421
 - Sequential Workflow Console Application project, 493
 - serialization, 438

- series of numbers, 58–60
- server
 - cookies, 698–699
 - Web
 - PostBack, 636–639
 - state, 639
- Server Explorer panel
 - data connections, 507–508
 - managed services, 507
 - overview, 506–507
- Server.HTMLEncode method, 410
- Service class, 457
- service-oriented application, 6, 726–728
- set[string] indexer, 148
- SetAccountNumber() method,
 - BankAccount class example, 266
- SetFirstName() method, 254
- SetName() method
 - example using, 248, 250, 253–254
 - this keyword, 255
- sets, collections, 130–133
- setting options, Visual Studio
 - additional options, 520–521
 - Environment section
 - keyboard commands, 519
 - overview, 518–519
 - start page, 519
 - Language, 519–520
- SetX() method, BankAccount class
 - example, 270
- SharePoint, 3, 484
- SHARP, 739–740
- SharpDevelop utility, 4, 7
- short integer, 28
- short-circuit evaluation, 80
- Show All Files button, Solution Explorer, 467, 504
- Side-by-side folder, 383
- signed integer variable, 28
- Silverlight project, 494
- Simonyi, Charles, 38
- Simple Object Access Protocol (SOAP)
 - size and speed, 734–735
 - standards, 732
 - WS-* standards, 733
- simple operators, 73–74
- SimpleDelegateExample program, 361
- SimpleSavingsAccount program,
 - 288–290
- Sin() method, 243
- single program
 - dividing into multiple assemblies
 - assemblies, 380
 - class libraries, 381
 - executables assemblies, 379, 381
 - dividing into multiple source files,
 - 378–379
- single-item selection controls, 666–668
- single-item variables, 120
- site accessibility
 - control features, 680
 - design considerations, 680
 - overview, 679
- site maps, navigating .NET framework
 - adding site map, 692–694
 - navigating site with SiteMap, 694–695
- Slider input control, 573
- Smart Indenting, 89
- SmartTags, 426
- smooth operators
 - arithmetic
 - assignment operator, 75–76
 - increment operator, 76–77
 - operating orders, 74–75
 - simple operators, 73–74
 - logical comparisons
 - comparing floating-point numbers,
 - 78–79
 - compound logical operations, 79–80
 - overview, 77–78
 - matching expression types
 - assigning types, 82–83
 - calculating type of operation, 81–82
 - overview, 80–81
- snippets
 - deploying, 525
 - making, 523–525
 - overview, 521–522
 - sharing, 526
 - surround, 522–523

- SOAP. *See* Simple Object Access Protocol (SOAP)
- SOAP envelope, 465
- Socket class, 457
- Sockets function, `System.Net` namespace, 458
- software
 - decomposing components
 - into functions, 403
 - determining what to protect, 402
 - documenting components of program, 402
 - identifying potential threats
 - in functions, 403
 - rating risk, 404
- Software Architecture Overview (SAO) diagram, 402
- Software Development Kit (SDK), 3, 380
- Software Development Life Cycle (SDLC), 402
- Solution box, New Project dialog box, 489
- Solution Explorer panel
 - files, 504
 - overview, 502–503
 - projects, 503–504
 - solutions, 503
- solutions
 - defined, 379
 - Solution Explorer panel, 503
- `SomeMethod()` method, 188
- `Sort()` method
 - arrays, 119
 - collection classes, 342
- sorted-Names array, 119
- `SortInterface` program, 340–341
- source file, 12
- source program, 15–18
- Source view, HTML designer, 499
- Special Projects, Visual Studio, 488, 494
- specialization, C# support, 213
- specialized panels, WPF, 569–570
- specifiers, format, 66
- `Split()` method, 58
 - controlling string output manually, 64–65
- Split view, HTML designer, 499
- spoofing identity, 403
- SQL (Structured Language Queries), 415
- SQL Injection attacks
 - overview, 409
 - preventing, 410
- SQL Server database, 350
- SQL Server Project, 494
- Stack Panel, WPF, 557–558
- stack trace, 195
- `Stack<T>` class, 123
- stacks
 - defined, 109
 - tracing, 195–196
- stand-alone class library, 382–383
- Star definition, Grid Panel, 563
- Start Page News Channel field, Visual Studio, 519
- Start Without Debugging command, Visual Studio 2008, 20
- start-up project, Visual Studio, 381
- `StartupUri` value, XAML file, 549
- state, application, 639
- state-based testing, 620
- static
 - generating in class members, 224–225
 - making dynamic, 775
- `static` keyword, 224–225
- static members, accessing, 225
- static methods, 228, 249–250, 366, 386
- static properties, 271
- status report, e-mailing, 462–465
- `StreamReader` class, 437, 448–452
- streams, `FileStream` class, 435–436
- `StreamWriter` class, 437, 441–442
- streamwriting
 - example, 439–440
 - methods and blocks, 442–445
 - overview, 438–439
 - `StreamWriter`, 441–442
 - using statement, 445–448
- STRIDE acronym, 403

- string literal, 35
- string type, 35–38
- StringBuilder, 69–71
- StringChunks collection, 156
- String.Concat() method, 51
- StringReader class, 452
- strings
 - case
 - converting to upper- or lowercase, 52–53
 - distinguishing between all-uppercase and all-lowercase strings, 52
 - comparing
 - Compare() method, 48–51
 - letter case, 51–52
 - controlling output manually
 - Concatenate() method, 63–64
 - overview, 60–61
 - Split() method, 64–65
 - Trim() and Pad() methods, 61–62
 - formatting precisely, 65–69
 - getting input from command line
 - handling series of numbers, 58–60
 - joining array of strings into one string, 60
 - parsing numeric input, 56–58
 - trimming excess white space, 55–56
 - indivisibility, 46–47
 - looping through, 53–54
 - overview, 45–46
 - performing common operations on, 47–48
 - searching, 54–55
 - StringBuilder, 69–71
 - String.Split() method, 58
 - StringWriter class, 452
 - struct constraint, 185
 - Structured Language Queries (SQL), 415
 - Student class, 47
 - StudentClassWithMethods program, 249–250
- styling controls
 - binding styles with CSS, 678–679
 - setting control properties, 677–678
- subclasses
 - defined, 209–210
 - inheritance, 307
- Submit buttons, ASP.NET, 670
- subscribing, C# events, 371–372
- subscript, 145
- substitutable classes, 294–295
- Substring() method, 53, 55
- Substring() operation, 69
- surfing Web streams
 - altering content sent to clients, 689–690
 - intercepting request, 686–689
- SUT (System Under Test) variable, 618
- swapping two objects, 117
- sw.Close() expression, 442
- switch statement, 52, 92–94, 108
- SymmetricExceptWith() method, 133
- synchronous I/O, 437
- Syndication Service Library project, 493
- syntax
 - collections
 - <T> notation, 123
 - generic, 124
 - overview, 122
 - iterator
 - overview, 156
 - yield break, 157–158
 - yield return, 157
- system absolute value method, 78
- System Under Test (SUT) variable, 618
- System.Collections namespace, 134
- System.Collections.Specialized namespace, 134
- System.Data namespace
 - connecting to data source, 420–425
 - database connectivity, 415
 - entity framework
 - generating, 432–433
 - overview, 431–432
 - writing code, 433
 - getting data, 418
 - overview, 416–417

- purpose of, 417
- sample database schema, 419–420
- visual tools, 425–428
- writing data code
 - basic, 429–431
 - output of visual tools, 428–429
- `System.Data.Common` namespace, 417
- `System.Data.OleDb` namespace, 417
- `System.Data.OracleClient` namespace, 417
- `System.Data.SqlClient` namespace, 417
- `System.Data.SqlTypes` namespace, 417
- `System.Drawing` namespace
 - brushes, 471
 - drawing board, 476–478
 - getting started, 473–475
 - graphics, 470
 - pens, 470–471
 - setting up project, 475–476
 - text, 471–472
- `System.Drawing.2D` namespace, 469
- `System.Drawing.Imaging` namespace, 469
- `System.Drawing.Text` namespace, 469
- `System.IO` namespace, 436, 453–454
- `System.Net` namespace
 - checking network status, 459–460
 - downloading file from internet, 460–462
 - e-mailing status report, 462–465
 - functions, 458
 - logging network activity, 465–468
 - overview, 456
- `System.Security`, 412–413
- `System.Web` namespace, 416
- `System.Web.Forms.Control`
 - `CreateGraphics` method, Windows Forms application, 470
- `System.Windows` namespace, 416
- `System.Windows.Forms.Control`
 - `CreateGraphics` method, 473

T

- `<T>` notation, collections syntax, 123
- T4. *See* Text Template Transformation Toolkit (T4)
- Tab character (`\t`), 35
- Table Options drop-down list, Visual Studio, 425–426
- tables, displaying info to user, 663–664
- `TabPanel` layout panel, 569
- Tabs panel, Visual Studio, 520
- tampering threat, 403
- Tasks window, Visual Studio, 514
- Team Foundation Server, 483
- Team System version, Visual Studio, 483–484
- Technical Stuff icon, 6
- template directive, T4, 540
- templates, hacking, 527–530
- temporary variable, storing events, 373
- ternary (`?:`) operator, 90
- Test Project, 494
- testing
 - Web applications, 711–712
 - WPF command pattern, 610–611
- text
 - creating template from, 537–539
 - input controls, 664–666
 - `System.Drawing` namespace, 471–472
- text boxes, New Project dialog box, 489
- Text Template Transformation Toolkit (T4)
 - building code based on outside data, 536
 - directives
 - assembly, 541
 - import, 541
 - include, 540
 - output, 539–540
 - template, 540
 - DSL Tools, 534
 - overview, 533–534
 - replacing repetitive coding, 535

- text templates. *See* Text Template Transformation Toolkit (T4)
- TextBlock control, 571
- TextBox control, 589
- TextBox input control, 572
- TextReader class, 436
- TextWriter class, 436
- this keyword
 - defined, 254–255
 - overview, 253–254
 - when explicit, 255–257
 - working without, 257–259
- throw keyword, 187–188
- throwing exceptions, 192
- tight coupling, 183–184
- tilde (~), 305
- Tip icon, 6
- TKey class, 126
- ToArray() method, 125
- ToBankAccountString() method, SimpleSavingsAccount program, 290
- ToBoolean() method, 56
- ToDecimal() command, Calculate Interest program, 88
- //TODO: comment, 450
- ToDouble() method, 56
- ToFloat() method, 56
- ToLower() method, 53
- ToNameString() method, 259
- ToolBarOverflowPanel layout panel, 570
- ToolBarPanel layout panel, 569
- Toolbox, Visual Studio
 - overview, 22, 505–506
 - reusing code, 23–24
 - saving code, 23
- Tools menu, Visual Studio, 514–515
- Tools/Options box, 537
- TopQueue() utility method, PriorityQueue class, 182
- ToString() method, 43, 70, 194, 321–322
- ToUpper() method
 - converting string cases, 52–53
 - overview, 47–48
 - uppercasing first string characters, 70–71
- tracing stack, 195–196
- TrackDownAMate.com, matching expression types at, 80–83
- transforming text templates
 - setting up environment
 - changing security settings, 536–537
 - creating template from text file, 537–539
- Text Template Transformation Toolkit (T4)
 - building code based on outside data, 536
 - directives, 539–540
 - DSL Tools, 534
 - overview, 533–534
 - replacing repetitive coding, 535
- Treat Warnings As Errors section, Build pane, 312–313
- TreeView control, XAML, 576
- tree-view selector, New Project dialog box, 488
- Trim() method
 - AlignOutput program, 62
 - controlling string output manually, 61–62
 - white space, 55
- TrimEnd() method, 56
- TrimFront() method, 56
- trimming white space, 55–56
- try block
 - exceptions, 189
 - file I/O activity, 441
 - StreamReader class, 450
- try keyword, 187–188
- TValue> class, 126–127
- TwoWay binding mode
 - defined, 581
 - editing data, 586
 - with INotifyPropertyChanged interface, 588
- type-safety, generics, 170–171
- typing, dynamic, 770–772

U

- UI commands, 605–606
- UIElements, 556
- uint integer, 28
- Ulong integer, 28
- UML (Unified Modeling Language), 322–323
- unary (-) operator, 74
- unboxed value-type, 170
- underscore (_), 257
- Unicode characters, 141
- Unicode file format, 438
- Unified Factoring Theory, 327
- Unified Modeling Language (UML), 322–323
- Uniform Grid Panel, WPF, 561–562
- union operation, 130
- UnionWith() method, 131
- unit testing, 610
- UnmanagedMemoryStream class, 454
- unreachable memory, 305
- UpdateSourceTrigger interface, 591
- Upload class, 457
- usable interfaces concept, object-oriented programming, 211
- user controls, 680–683
- users, 241. *See also* client; input from user
- UseTheDel() method, Simple DelegateExample program, 362
- ushort integer, 28
- using statement, streamwriting, 445–448
- UsingVarWithArraysAndCollections sample program, 121
- UTF8 format, 438

V

- ValidatesOnDataErrors interface, 591
- ValidatesOnExceptions interface, 591
- validating data, WPF, 589–592
- value variable types, 36–37

- Values property, dictionaries, 128
- value-type variables, declaring
 - bool type, 34
 - calculating leap years, 38–40
 - cast, 41–42
 - character types, 34–36
 - comparing string and char, 37–38
 - decimal type, 32–33
 - declaring numeric constants, 40
 - floating-point variables, 29–32
 - int variable, 26–28
 - letting C# compiler infer data types, 42–43
 - overview, 25–26
 - representing fractions, 28–29
 - value type, 36–37
- var keyword, 43, 120–121, 128
- variability, declaring value-type variables
 - bool type, 34
 - calculating leap years, 38–40
 - cast, 41–42
 - character types, 34–36
 - comparing string and char, 37–38
 - decimal type, 32–33
 - declaring numeric constants, 40
 - floating-point variables, 29–32
 - int, 26–28
 - letting C# compiler infer data types, 42–43
 - overview, 25–26
 - representing fractions, 28–29
 - value type, 36–37
- VariableArrayAverage program, 112–114
- variable-length array, 112–114
- variables. *See also* floating-point variables; names of specific variables; value-type variables, declaring
 - declaring inside loop, 103
 - defined, 22
 - pointing to different objects, 222
- variance, 2, 796
- Variant data types, 43
- VCR Bar, Visual Studio, 426–427

- VehicleAndMotor sample program, 223–224
 - VehicleDataOnly program example, 218–220
 - vertical split, WPF designer, 497
 - Vertical Stack Panel, 557–558
 - VerticalContentAlignment
 - property, 565
 - View Controller, modeling, 656–657
 - ViewModel, WPF
 - Business Layer (BL), 613
 - code
 - Add Customer command, 621–623
 - model, 615–617
 - model repositories, 619–621
 - model unit tests, 617–618
 - overview, 614–615
 - testing, 624
 - View, 624–626
 - Data Access Layer (DAL), 613
 - importance of, 612–613
 - Model, 613
 - overview, 614
 - View, 613
 - virtual keyword
 - abstract method, 330
 - C# interface, 335
 - methods, 321
 - Virtual PC, 487
 - VirtualizingStackPanel layout panel, 570
 - visibility, namespace, 395–396
 - Visio 2007 Add-in project, 493
 - Vissides, John, 601
 - Visual Basic Express, 483
 - Visual C# Express version, 482–483
 - Visual C# version, 14
 - Visual Studio
 - 2010, 14
 - button handler, 375
 - customizing
 - hacking project types, 527–532
 - setting options, 518–521
 - using snippets, 521–526
 - determining exceptions of methods, 203
 - implementing interface, 336
 - installing, 486–487
 - New Project dialog box, 395
 - projects
 - categories, 491–494
 - New Project dialog box, 488–489
 - solutions and, 489–491
 - Solution Explorer, 379
 - start-up project, 381
 - Task List window, 450
 - Team System, 402
 - testing Web applications, 711–712
 - unit testing framework, 386
 - versions
 - Academic, 485
 - Express, 4, 482–483
 - features, 485–486
 - Microsoft Developer Network (MSDN), 484–485
 - Professional, 483
 - Team System, 483–484
 - Web application
 - additional file types, 651–652
 - coding in Code View, 647–651
 - working in designer, 642–647
 - visual tools
 - output, 428–429
 - System.Data namespace, 425–428
 - Visual Web Developer (VWD), 483
 - void keyword, 244
 - void methods, 133, 244–245
 - .vscontent file, 526
 - .vsi file, 526
 - VWD (Visual Web Developer), 483
-
- W**
- Wagner, Bill, 350
 - Warning icon, 6
 - WCF. *See* Windows Communication Foundation (WCF)
 - WCF Service Application project, 492

- WCF Service Configuration Manager tool, Visual Studio, 515
- WCF Service Library project, 493
- Web application
 - breaking down, 630–632
 - developing with style
 - building in n-tier, 655–656
 - coding behind, 652–653
 - modeling View Controller, 656–657
 - scripting, 653–655
 - testing with Visual Studio, 711–712
 - working in Visual Studio
 - additional file types, 651–652
 - coding in Code View, 647–651
 - designer, 642–647
- Web class, 457
- Web construction
 - deployment
 - Copy Web design surface, 714–715
 - options, 713–714
 - Package/Publish tab, 715–716
 - managing files
 - organizing, 708
 - reviewing file types, 706–707
 - reviewing project types, 704–706
- Master Pages
 - adding content, 710–711
 - making master page, 709–710
 - testing Web applications with Visual Studio, 711–712
- Web Development with ASP.NET, 6
- Web forms applications
 - best practices for securing, 411–412
 - overview, 408–409
 - script exploits
 - overview, 410
 - preventing, 410–411
 - SQL Injection attacks
 - overview, 409
 - preventing, 410
- Web Forms designer, 499
- Web projects, Visual Studio, 488, 492
- Web servers
 - PostBack, 636–639
 - state, 639
- Web services
 - building service-oriented applications, 726–728
 - building with ASMX
 - building code for SHARP, 739–740
 - consuming services in applications, 743–744
 - creating new service, 735–738
 - deploying, 741–742
 - overview, 731–732
 - SOAP, 732–735
 - building with ReST, 759–765
 - changing WCF service to use ReST, 762–765
 - getting to know ReST, 759–760
 - understanding guiding principles of ReST, 760–761
 - building with WCF
 - breaking it down, 748–750
 - configuring, 752–756
 - consuming, 757–758
 - deploying, 756–757
 - making registration service, 750–752
 - overview, 745–747
 - chunky versus chatty, 724–725
 - contract driven, 722–724
 - loosely coupled, 721–722
 - overview, 719–721
 - providing XML Web services, 728
 - sample apps, 728–729
- Web-knowledgeable program, 14
- WebRequest class, `System.Net` namespace, 461
- WebResponse class, `System.Net` namespace, 461
- WhenHit option, Breakpoints window, 514
- where clause, 185
- while loop
 - file-manipulation program, 441
 - incrementing counting variable, 98
 - overview, 95–99
- white space, trimming, 55–56, 121

- Windows applications
 - authentication using Windows login, 404–407
 - deployment security, 407–408
 - encrypting information, 407
- Windows Communication Foundation (WCF)
 - building Web services
 - breaking it down, 748–750
 - configuring, 752–756
 - consuming, 757–758
 - deploying, 756–757
 - making registration service, 750–752
 - overview, 745–747
 - changing service to use ReST, 762–765
- Windows Forms (WinForms)
 - Application project, 491
 - control library project, 492
 - designer, 498
 - terminology, 556
- Windows login, 404–407
- Windows Presentation Foundation (WPF)
 - Application project, 491
 - arranging elements with layout panels
 - Canvas, 560–561
 - data entry form, 567–569
 - Dock, 559–560
 - Grid, 562–566
 - specialized, 569–570
 - Stack, 557–558
 - Uniform Grid, 561–562
 - Wrap, 559
- Browser Application project, 491
- command pattern
 - built-in commands, 603–604
 - custom commands, 605–610
 - focus, 605
 - ICommand, 602
 - overview, 601–602
 - routed commands, 602–603
 - testing, 610–611
- creating application
 - declaring application-scoped resource, 549–550
 - overview, 547–549
 - running application, 550–552
- Custom Control Library project, 491
- data binding
 - converting your data, 592–599
 - dependency properties, 579–580
 - finding out more about WPF data
 - binding, 599–600
 - modes, 580–581
 - objects, 581–584
 - overview, 584–588
 - validating data, 589–592
- designer, 496–498
- general discussion, 545–546
- laying out application, 555–556
- style tree viewer, 488
- User Control Library project, 492
- ViewModel
 - Business Layer (BL), 613
 - code, 614–626
 - Data Access Layer (DAL), 613
 - importance of, 612–613
 - Model, 613
 - overview, 614
 - View, 613
- Window class, 556
- XAML
 - basic input controls, 572–574
 - versus C#, 552–553
 - display only controls, 570–572
 - list-based controls, 574–577
 - overview, 547
- Windows progress bar, 357
- Windows projects, Visual Studio, 488
- Windows Security Application Code, 406–407
- Windows Security tab, 407–408
- Windows Services project, 491
- WinForms. *See* Windows Forms (WinForms)
- Withdraw() method
 - BankAccount class example, 263
 - SimpleSavingsAccount program, 290
- Word 2007 Add-in project, 493
- Word 2007 Document project, 493
- Word 2007 Template project, 493
- Workflow Projects, Visual Studio, 493

WPF. *See* Windows Presentation Foundation (WPF)

Wrap Panel, WPF, 559

wrapper, defined, 173

wrapping

- overview, 443
- StreamWriter class, 442

write modes, 442

WriteFileFromConsole() method, StreamWriter class, 442

WriteLine() command, Calculate Interest program, 88

WriteLine() method

- newline character (\n) versus, 220
- overview, 246
- StreamWriter class, 437

writers, FileStream class, 436–437, 452–453

writing data code

- basic data code, 429–431
- output of visual tools, 428–429

writing secure code

- designing secure software
 - decomposing components into functions, 403
 - determining what to protect, 402
 - documenting components of program, 402
 - identifying potential threats in functions, 403
 - rating risk, 404
- System.Security, 412–413

Web forms applications

- best practices for securing, 411–412
- overview, 408–409
- script exploits, 410–411
- SQL Injection attacks, 409–410

Windows applications

- authentication using Windows login, 404–407
- deployment security, 407–408
- encrypting information, 407

X

XAML

- versus C#, 552–553
- controls
 - basic input controls, 572–574
 - display only controls, 570–572
 - list-based controls, 574–577
- defining data binding with, 581–583
- overview, 547
- Windows Presentation Foundation (WPF), 496

x:Class attribute, XAML, 549

XML

- comments, 203
- System.Data namespace, 416
- Web services, 728

x:Name attribute, XAML, 551

Y

yield break syntax, 157–158

yield return syntax, 155, 157

Z

zeroth element, 111

