Detailed Contents

Preface To The Second Edition

Welcome To The Second Edition xxxv
Improvements To This Edition xxxv
Where To Send Comments And Suggestions xxxvi
Acknowledgments xxxvi

Preface To The First Edition

Welcome – And Thank You! xxxvii
Target Audience xxxvii
Approach(es) xxxvii
Pedagogy – I Mean, How This Book's Arranged xxxviii
Learning Objectives xxxviii
Introduction xxxviii
Content xxxviii
Quick Reviews xxxviii
Summary xxxix
Skill-Building Exercises xxxix
Suggested Projects xxxix
Self-Test Questions xxxix
References xxxix
Notes xxxix
Typographical Formats xxxix
This Is An Example Of A First Level Subheading xxxix
This Is An Example Of A Second Level Subheading ..... *xxxix*
Source Code Formatting xl
SupportSite™ Website xl
Problem Reporting xl
About The Author xl
Acknowledgments xl

Part I: The C# Student Survival Guide

1 An Approach to the Art of Programing

Introduction 4
The Difficulties You Will Encounter Learning C# 4
Required Skills *4*
The Planets Will Come Into Alignment *5*
How This Chapter Will Help You 5
Personality Traits Found In Great Programmers 5
Creative 5

Tenacious 5
Resilient 6
Methodical 6
Meticulous 6
Honest 6
Proactive 6
Humble 6
Be a Generalist and a Just-in-Time Specialist 6
Project Management 7
Three Software Development Roles 7
Analyst 7
Architect 7
Programmer 8
A Project-Approach Strategy 8
You Have Been Handed A Project – Now What? 8
Strategy Areas of Concern 8
Think Abstractly 10
The Strategy In A Nutshell 10
Applicability To The Real World 11
The Art Of Programming 11
Don't Start At The Computer 11
Inspiration Strikes At The Weirdest Time 11
Own Your Own Computer 12
You Either Have Time and No Money, or Money and No Time 12
The Family Computer Is Not Going To Cut It! 12
Set The Mood 12
Location, Location, Location 12
Concept Of The Flow 12
The Stages of Flow 13
Be Extreme 13
The Programming Cycle 14
The Programming Cycle Summarized 14
A Helpful Trick: Stubbing 15
Fix The First Compiler Error First 15
Managing Project Complexity 15
Conceptual Complexity 15
Managing Conceptual Complexity 16
The Unified Modeling Language (UML) 16
Physical Complexity 16
Managing Physical Complexity 16
The Relationship Between Physical and Conceptual Complexity 17
Maximize Cohesion – Minimize Coupling 17
The Engineer's Notebook 17
Summary 18
Skill-Building Exercises 19
Suggested Projects 19
Self-Test Questions 19
References 20
Notes 20

2 Small Victories: Creating C# Projects

Introduction 22
Creating Projects With Microsoft C#.NET Command-Line Tools 22

Downloading And Installing The .NET Framework 23
Downloading And Installing Notepad++ 23
Configuring Your Development Environment 24
Environment Variables 24
Creating A Project Folder 27
Setting Folder Options 28
Creating A Shortcut To The Command Console And Setting Its Properties 29
Testing The Configuration 31
Creating The Source File 31
Compiling The Source File 32
Executing The Application 33
Quick Review 34
Creating Projects With Microsoft Visual C# Express 35
Download and Install Visual C# Express 36
Quick Tour Of Visual C# Express 36
Select Project Type 36
Saving The Project 38
Build The Project 38
Locating The Project Executable File 40
Execute The Project 41
Where To Go For More Information About Visual C# Express 41
DreamSpark 42
Quick Review 42
Summary 42
Skill-Building Exercises 43
Suggested Projects 43
Self-Test Questions 43
References 44
Notes 44

3 Project Walkthrough

Introduction 46
The Project-Approach Strategy Summarized 46
Development Cycle 47
Project Specification 48
Analyzing The Project Specification 49
Application Requirements Strategy Area 50
Problem-Domain Strategy Area 50
Language-Features Strategy Area 53
Design Strategy Area 55
Development Cycle: First Iteration 56
Plan (First Iteration) 56
Code (First Iteration) 57
Test (First Iteration) 57
Integrate/Test (First Iteration) 57
Development Cycle: Second Iteration 58
Plan (Second Iteration) 58
Code (Second Iteration) 58
Test (Second Iteration) 59
Integrate/Test (Second Iteration) 59
Development Cycle: Third Iteration 59
Plan (Third Iteration) 60
Code (Third Iteration) 61

Integrate/Test (Third Iteration) 63
A Bug In The Program 63
Development Cycle: Fourth Iteration 65
Plan (Fourth Iteration) 65
Implementing State Transition Diagrams 66
Implementing The PrintFloor() Method 67
Code (Fourth Iteration) 67
Putting It All Together 69
Test (Fourth Iteration) 71
Integrate/Test (Fourth Iteration) 71
Development Cycle: Fifth Iteration 72
Plan (Fifth Iteration) 72
Code (Fifth Iteration) 73
Test (Fifth Iteration) 75
Putting It All Together 75
Integrate/Test (Fifth Iteration) 79
Final Considerations 79
Complete RobotRat.cs Source Code Listing 80
An Alternative Main() Method Implementation 86
Summary 88
Skill-Building Exercises 88
Suggested Projects 88
Self-Test Questions 89
References 89
Notes 90

4 Computers, Programs, And Algorithms

Introduction 92
What Is A Computer? 92
Computer vs. Computer System 92
Computer System 92
Processor 94
Three Aspects of Processor Architecture 95
Feature Set 95
Feature Set Implementation 96
Feature Set Accessibility 96
Memory Organization 96
Memory Basics 96
Memory Hierarchy 97
Bits, Bytes, Words 97
Alignment and Addressability 98
What Is A Program? 98
Two Views of a Program 98
The Human Perspective 99
The Computer Perspective 99
The Processing Cycle 100
Fetch 100
Decode 100
Execute 100
Store 101
Why A Program Crashes 101
Algorithms 101
Good vs. Bad Algorithms 101

Don't Reinvent The Wheel! 104
Virtual Machines And The Common Language Infrastructure 104
Virtual Machines 105
The Common Language Infrastructure (CLI) 105
Four Parts Of The Common Language Infrastructure 106
The Cross Platform Promise 107
Summary 108
Skill-Building Exercises 109
Suggested Projects 109
Self-Test Questions 109
References 110
Notes 111

5. Navigating .NET Framework Documentation

Introduction 114
MSDN: The Definitive Source For API Information 114
Discovering Information About Classes 114
General Overview Page 115
Class Member Documentation 117
Getting Information On Other Class Members 119
Quick Review 119
Namespaces Used Heavily In This Book 120
Navigating An Inheritance Hierarchy 120
Quick Review 121
Beware Obsolete APIs 122
Summary 122
Skill-Building Exercises 122
Suggested Projects 123
Self-Test Questions 124
References 124
Notes 125

Part II: Language Fundamentals

6 Simple C# Programs

Introduction 130
What Is A C# Program? 130
A Simple Console Application 131
Definition Of Terms: Application, Assembly, Module, and Entry Point 131
Structure Of A Simple Application 131
Purpose Of The Main() Method 132
Main() Method Signatures 132
Quick Review 133
Identifiers And Reserved Keywords 134
Identifier Naming Rules 135
Quick Review 135
Types 136
Value Type Variables vs. Reference Type Variables 136

Value Type Variables *137*
Reference Type Variables *137*
Maybe Some Pictures Will Help *138*
Mapping Predefined Types To System Structures *138*
Quick Review 140
Statements, Expressions, and Operators **140**
Statement Types 141
Operators And Their Use 142
Operator Precedence and Associativity *144*
Forcing Operator Precedence and Associativity Order With Parentheses *144*
Operators and Operands *144*
Operator Usage Examples 144
Primary Expression Operators *144*
Unary Expression Operators *145*
Multiplicative Expression Operators *146*
Additive Expression Operators *146*
Shift Expression Operators *147*
Relational, Type-Testing, and Equality Expression Operators *148*
Logical AND, OR, and XOR Expression Operators *150*
Conditional AND and OR Expression Operators *152*
Conditional (Ternary) Expression Operator *153*
Assignment Expression Operators *154*
Quick Review 154
Summary **155**
Skill-Building Exercises **155**
Suggested Projects **156**
Self-Test Questions **156**
References **157**
Notes **158**

7 Controlling The Flow Of Program Execution

Introduction **160**
Selection Statements **160**
If Statement 160
Handling Program Error Conditions *161*
Executing Code Blocks In If Statements *163*
Executing Consecutive If Statements *164*
If/Else Statement 164
Chained If/Else Statements *166*
Switch Statement 167
Implicit Case Fall-Through *168*
Nested Switch Statement *169*
Quick Review 170
Iteration Statements **170**
While Statement 171
Personality Of The While Statement *171*
Do/While Statement 172
Personality Of The Do/While Statement *172*
For Statement 173
How The For Statement Is Related To The While Statement *173*
Personality Of The For Statement *174*
Nesting Iteration Statements 174
Mixing Selection And Iteration Statements: A Powerful Combination 175
Quick Review 177

Break, Continue, And Goto 177
Break Statement 177
Continue Statement 178
Goto Statement 179
Quick Review 179
Selection And Iteration Statement Selection Table 180
Summary 181
Skill-Building Exercises 182
Suggested Projects 184
Self-Test Questions 185
References 185
Notes 186

8 Arrays

Introduction 188
What Is An Array? 188
Specifying Array Types 189
Quick Review 190
Functionality Provided By C# Array Types 190
Array-Type Inheritance Hierarchy 190
Special Properties Of C# Arrays 191
Quick Review 192
Creating And Using Single-Dimensional Arrays 192
Arrays Of Value Types 192
How Value-Type Array Objects Are Arranged In Memory 193
Finding An Array's Type, Rank, And Total Number of Elements 194
Creating Single-Dimensional Arrays Using Array Literal Values 195
Differences Between Arrays Of Value Types And Arrays Of Reference Types 196
Single-dimensional Arrays In Action 198
Message Array 198
Calculating Averages 200
Histogram: Letter Frequency Counter 201
Quick Review 203
Creating And Using Multidimensional Arrays 203
Rectangular Arrays 203
Initializing Rectangular Arrays With Array Literals 205
Ragged Arrays 206
Multidimensional Arrays In Action 207
Weighted Grade Tool 207
Quick Review 209
The Main() Method's String Array 210
Purpose And Use Of The Main() Method's String Array 210
Manipulating Arrays With The System.Array Class 211
Numeric Formatting 211
Summary 212
Skill-Building Exercises 213
Suggested Projects 213
Self-Test Questions 216
References 216
Notes 217

9 Toward Problem Abstraction: Creating New Data Types

Introduction **220**
Abstraction: Amplify The Essential, Eliminate The Irrelevant **220**
Abstraction Is The Art Of Programming 220
Where Problem Abstraction Fits Into The Development Cycle 221
Creating Your Own Data Types 221
Case-Study Project: Write A People Manager Program 221
Quick Review 223
The UML Class Diagram **223**
Quick Review 225
Overview Of The Class Construct **225**
Eleven Categories Of Class Members 225
Fields *225*
Constants *228*
The Difference Between `const` *and* `readonly`*; Compile-Time vs. Runtime Constants* *228*
Properties *229*
Methods *231*
Instance Constructors *231*
Static Constructors *231*
Events *231*
Operators *231*
Indexers *231*
Nested Type Declarations *232*
Finalizers *232*
Access Modifiers 232
Public *232*
Private *232*
Protected *232*
Internal *232*
Protected Internal *233*
The Concepts Of Horizontal Access, Interface, and Encapsulation 233
Quick Review 234
Methods **234**
Method Naming: Use Action Words That Indicate The Method's Purpose 234
Maximize Method Cohesion 234
Structure Of A Method Definition 235
Method Modifiers (optional) *235*
Return Type Or Void (optional) *236*
Method Name (mandatory) *236*
Parameter List (optional) *236*
Method Body (optional for abstract or external methods) *236*
Method Definition Examples 237
Method Signatures 237
Overloading Methods 237
Constructor Methods 238
Quick Review 238
Building And Testing The Person Class **238**
Start By Creating The Source File And Class Definition Shell 239
Defining Person Instance Fields 239
Defining Person Properties And Constructor Method 239
Adding Properties *240*
Adding A Constructor Method *240*
Testing The Person Class: A Miniature Test Plan 241
Use The PeopleManagerApplication Class As A Test Driver *242*
Adding Features To The Person Class: Calculating Age 242

Adding Features To The Person Class: Convenience Properties 244
Adding Features To The Person Class: Finishing Touches 246
Quick Review 248
Building and Testing The PeopleManager Class 248
Defining The PeopleManager Class Shell 249
Defining PeopleManager Fields 249
Defining PeopleManager Constructor Methods 249
Defining Additional PeopleManager Methods 250
Testing The PeopleManager Class 251
Adding Features To The PeopleManager Class 251
Quick Review 253
More About Methods 254
Value Parameters And Reference Parameters 254
Value Parameters: The Default Parameter Passing Mode 254
Reference Parameters: Using The `ref` Parameter Modifier 255
The `out` Parameter Modifier 257
Parameter Arrays: Using The `params` Modifier 259
Local Variable Scoping 259
Anywhere An Object Of <type> Is Required, A Method That Returns <type> Can Be Used 260
Quick Review 260
Structures vs. Classes 260
Value Semantics vs. Reference Semantics 261
Ten Authorized Members vs. Eleven 261
Default Variable Field Values 261
Behavior During Assignment 261
`this` Behaves Differently 262
Inheritance Not Allowed 262
Boxing And Unboxing 262
When To Use Structures 262
Summary 263
Skill-Building Exercises 264
Suggested Projects 265
Self-Test Questions 267
References 268
Notes 269

10 Compositional Design

Introduction 272
Managing Conceptual And Physical Complexity 272
Compiling Multiple Source Files Simultaneously With csc 272
Quick Review 273
Dependency vs. Association 273
Aggregation 274
Simple vs. Composite Aggregation 274
The Relationship Between Aggregation And Object Lifetime 274
Quick Review 275
Expressing Aggregation In A UML Class Diagram 275
Simple Aggregation Expressed In UML 275
Composite Aggregation Expressed In UML 276
Aggregation Example Code 276
Simple Aggregation Example 276
Composite Aggregation Example 278

Quick Review 279
Sequence Diagrams 279
Message Passing 281
Magic Draw 281
Quick Review 281
The Engine Simulation: An Extended Example 281
The Purpose Of The Engine Class 281
Engine Class Attributes And Methods 281
Engine Simulation Sequence Diagrams 284
Running The Engine Simulation Program 284
Quick Review 284
Complete Engine Simulation Code Listing 286
Summary 291
Skill-Building Exercises 291
Suggested Projects 292
Self-Test Questions 293
References 294
Notes 295

11 Inheritance and Interfaces

Introduction 298
Three Purposes Of Inheritance 298
Implementing The "is a" Relationship 299
The Relationship Between The Terms Type, Interface, and Class 299
Meaning Of The Term Interface 299
Meaning Of The Term Class 300
Quick Review 300
Expressing Generalization And Specialization In The UML 300
A Simple Inheritance Example 300
The UML Diagram 301
BaseClass Source Code 302
DerivedClass Source Code 302
DriverApplication Program 303
Quick Review 304
Another Inheritance Example: Person - Student 304
The Person - Student UML Class Diagram 304
Person - Student Source Code 304
Casting 307
Use Casting Sparingly 308
Quick Review 309
Overriding Base Class Methods 309
Quick Review 311
Abstract Methods and Abstract Base Classes 311
The Primary Purpose Of An Abstract Base Class 311
Expressing Abstract Base Classes In UML 311
Abstract Class Example 312
Quick Review 314
Interfaces 314
The Purpose Of Interfaces 314
Authorized Interface Members 314
The Differences Between An Interface And An Abstract Class 315
Expressing Interfaces In UML 315

Expressing Realization In A UML Class Diagram 315
An Interface Example 316
Quick Review 318
Controlling Horizontal And Vertical Access 318
Quick Review 318
Sealed Classes And Methods 319
Quick Review 319
Polymorphic Behavior 319
Quick Review 319
Inheritance Example: Employee 320
Inheritance Example: Engine Simulation 323
Engine Simulation UML Diagram 323
Simulation Operational Description 325
Compiling The Engine Simulation Code 325
Complete Engine Simulation Code Listing 325
Summary 330
Skill-Building Exercises 331
Suggested Projects 332
Self-Test Questions 333
References 334
Notes 335

Part III: GUI Programming & Custom Events

12 Windows Forms Programming

Introduction 340
The Form Class 340
Form Class Inheritance Hierarchy 340
A Simple Form Program 340
Quick Review 342
Application Messages, Message Pump, Events, And Event Loop 342
Message Categories 343
Messages In Action: Trapping Messages With IMessageFilter 344
Final Thoughts On Messages 345
Quick Review 345
Screen And Window (Client) Coordinate System 345
Quick Review 347
Manipulating Form Properties 348
Quick Review 349
Adding Components To Windows: Button, TextBox, And Label 350
Quick Review 351
Registering Event Handlers With GUI Components 352
Delegates And Events 352
Quick Review 354
Handling GUI Component Events In Separate Objects 355
Quick Review 357
Layout Managers 358
FlowLayoutPanel 358
TableLayoutPanel 361

Quick Review 362
Menus 363
Quick Review 366
A Little More About TextBoxes 366
Quick Review 368
The Rhythm Of Coding GUIs 368
Summary 368
Skill-Building Exercises 370
Suggested Projects 371
Self-Test Questions 371
References 372
Notes 373

13 Custom Events

Introduction 376
C# Event Processing Model: An Overview 376
Quick Review 377
Custom Events Example: Minute Tick 378
Custom Events Example: Automated Water Tank System 380
Naming Conventions 388
Final Thoughts On Extending The EventArgs Class 388
Summary 388
Skill-Building Exercises 388
Suggested Projects 389
Self-Test Questions 389
References 390
Notes 391

Part IV: Intermediate Concepts

14 Collections

Introduction 396
Case Study: Building A Dynamic Array 396
Evaluating DynamicArray 398
The ArrayList Class To The Rescue 399
A Quick Peek At Generics 399
Quick Review 400
Data Structure Performance Characteristics 400
Array Performance Characteristics 401
Linked List Performance Characteristics 402
Hash Table Performance Characteristics 402
Chained Hash Table vs. Open-Address Hash Table 405
Red-Black Tree Performance Characteristics 405
Stacks And Queues 406
Quick Review 406
Navigating The .NET Collections API 407
System.Collections 407
System.Collections.Generic 407

System.Collections.ObjectModel 408
System.Collections.Specialized 408
Mapping Non-Generic To Generic Collections 408
Quick Review 409
Using Non-Generic Collection Classes - Pre .NET 2.0 409
Objects In – Objects Out: Casting 101 411
Extending ArrayList To Create A Strongly-Typed Collection 412
Using Generic Collection Classes – .NET 2.0 and Beyond 414
List<T>: Look Ma, No More Casting! 414
Implementing KeyedCollection<TKey, TItem> 415
Quick Review 417
Special Operations On Collections 417
Sorting A List 417
Implementing System.IComparable<T> 418
Extending Comparer<T> 421
Converting A Collection Into An Array 422
Quick Review 423
Summary 423
Skill-Building Exercises 424
Suggested Projects 425
Self-Test Questions 425
References 426
Notes 427

15 Exceptions: Writing Fault-Tolerant Software

Introduction 430
What Is An Exception 430
.NET CLR Exception Handling Mechanism 430
Unhandled Exceptions 431
The Exception Information Table 431
Quick Review 431
Exception Class Hierarchy 432
Application vs. Runtime Exceptions 432
Runtime Exception Listing 433
Determining What Exceptions A .NET Framework Method Throws 433
Quick Review 433
Exception Class Properties 434
Quick Review 435
Creating Exception Handlers: Using Try/Catch/Finally Blocks 435
Using A Try/Catch Block 435
First Line of Defense: Use Defensive Coding 436
Using Multiple Catch Blocks 437
Using A Finally Block 438
Quick Review 439
Creating Custom Exceptions 439
Extending The Exception Class 439
Manually Throwing An Exception With The throw Keyword 441
Translating Low-Level Exceptions Into High-Level Exceptions 441
Quick Review 441
Documenting Exceptions 441
Summary 442
Skill-Building Exercises 443

Suggested Projects **443**
Self-Test Questions **443**
References **444**
Notes **445**

16 Multithreaded Programming

Introduction **448**
Multithreading Overview: The Tale Of Two Vacations **448**
Single-Threaded Vacation 448
Multithreaded Vacation 448
The Relationship Between A Process And Its Threads 449
Vacation Gone Bad 451
Quick Review 451
Creating Managed Threads With The Thread Class **452**
Single-Threaded Vacation Example 452
Multithreaded Vacation Example 454
Thread States 455
Creating And Starting Managed Threads 456
ThreadStart Delegate *456*
ParameterizedThreadStart Delegate: Passing Arguments To Threads *457*
Blocking A Thread With Thread.Sleep() 458
Blocking A Thread With Thread.Join() 459
Foreground vs. Background Threads 461
Quick Review 463
Creating Threads With The BackgroundWorker Class **464**
Quick Review 467
Thread Pools **468**
Quick Review 470
Asynchronous Method Calls **470**
Obtaining Results From An Asynchronous Method Call 472
Providing A CallBack Method To BeginInvoke() 473
Quick Review 474
When To Employ The Various Thread Types **474**
Summary **474**
Skill-Building Exercises **476**
Suggested Projects **477**
Self-Test Questions **477**
References **477**
Notes **478**

17 File I/O

Introduction **480**
Manipulating Directories And Files **480**
Files, Directories, And Paths 481
Manipulating Directories And Files 481
Verbatim String Literals 482
Quick Review 483
Serializing Objects To Disk **483**
Serializable Attribute 484
Serializing Objects With BinaryFormatter 485
Serializing Objects With XmlSerializer 487

Quick Review 489
Working With Text Files 489
Some Issues You Must Consider 490
Saving Dog Data To A Text File 490
Quick Review 492
Working With Binary Data 492
Quick Review 494
Random Access File I/O 494
Towards An Approach To The Adapter Project 496
Start Small And Take Baby Steps 496
Other Project Considerations 497
Locking A Record For Updates And Deletes 497
Monitor.Enter()/Monitor.Exit() vs. The lock Keyword 497
Translating Low-Level Exceptions Into Higher-Level Exception Abstractions 498
Where To Go From Here 498
Complete DataFile Adapter Project Source Code Listing 498
Quick Review 513
Working With Log Files 513
Quick Review 516
Using FileDialogs 516
Quick Review 518
Summary 518
Skill-Building Exercises 520
Suggested Projects 521
Self-Test Questions 521
References 522
Notes 523

Part V: Network & Database Programming

18 Network Programming Fundamentals

Introduction 528
What Is A Computer Network? 528
Purpose Of A Network 528
The Role Of Network Protocols 528
Homogeneous Vs. Heterogeneous Networks 529
The Unifying Network Protocols: TCP/IP 529
What's So Special About The Internet? 529
Quick Review 531
Servers & Clients 531
Server Hardware And Software 531
Client Hardware And Software 531
Quick Review 532
Application Distribution 532
Physical Distribution On One Computer 532
Running Multiple Clients On The Same Computer 533
Addressing The Local Machine 533
Physical Distribution Across Multiple Computers 534
Quick Review 534
Multilayered Applications 534

Logical Application Layers 534
Physical Tier Distribution 535
Quick Review 536
Internet Networking Protocols: Nuts & Bolts 536
The Internet Protocols: TCP, UDP, And IP 536
The Application Layer 537
Transport layer 537
Network Layer 538
Data Link And Physical Layers 538
Putting It All Together 538
What You Need To Know 538
Quick Review 539
Summary 540
Skill-Building Exercises 541
Suggested Projects 541
Self-Test Questions 541
References 542
Notes 543

19 Networked Client -Server Applications

Introduction 546
Building Client-Server Applications With .NET Remoting 546
The Three Required Components of A .NET Remoting Application 546
A Simple .NET Remoting Application 547
SingleCall vs. Singleton 550
Accessing A Remote Object Via An Interface 551
Using Configuration Files 553
Passing Objects Between Client And Server 555
Quick Review 560
Client-Server Applications With TcpListener And TcpClient 560
TCP/IP Client-Server Overview 560
A Simple Client-Server Application 561
Building A Multithreaded Server 564
Listening On Multiple IP Addresses 565
Sending Objects Between Client And Server 569
Quick Review 573
Summary 574
Skill-Building Exercises 575
Suggested Projects 576
Self-Test Questions 576
References 577
Notes 577

20 Database Access & Multilayered Applications

Introduction 580
What You Are Going To Build 580
Preliminaries 581
Installing SQL Server Express Edition 581
Installing Microsoft SQL Server Management Studio Express 583
Installing Microsoft Enterprise Library 584
A Simple Test Application 586

Introduction To Relational Databases And SQL 588
Terminology 588
Structured Query Language (SQL) 589
Data Definition Language (DDL) 589
Creating The EmployeeTraining Database 590
Creating A Database With A Script 591
Creating Tables 592
SQL Server Database Types 593
Data Manipulation Language (DML) 595
Using The Insert Command 595
Using The Select Command 596
Using The Update Command 598
Using The Delete Command 599
Quick Review 599
Complex SQL Queries 600
Creating A Related Table With A Foreign Key 600
Inserting Test Data Into The tbl_employee_training Table 602
Selecting Data From Multiple Tables 603
Join Operations 603
Testing The Cascade Delete Constraint 605
Quick Review 605
The Server Application 605
Project Folder Organization 606
Using Microsoft Build To Manage And Build the Project 606
First Iteration 609
Coding The EmployeeVO And EmployeeDAO 610
Application Configuration File 620
Creating Test Application 620
Second Iteration 625
Testing The Code - Second Iteration 638
Reality Check 648
Third Iteration 648
The Client Application 654
Third Iteration (continued) 654
Fourth Iteration 657
Fifth Iteration 663
Sixth Iteration 669
Compiling And Running The Modified EmployeeTrainingClient Project 684
Where To Go From Here 686
Summary 687
Skill-Building Exercises 687
Suggested Projects 688
Self-Test Questions 689
References 689
Notes 690

Part VI: Advanced Topics

21 Operator Overloading

Introduction 694

Operator Overloading **694**
Overloadable Operators 694
Quick Review 695
Overloading Unary Operators **695**
+, - Operators 696
! Operator 697
true, false Operators 698
++, -- Operators 700
Quick Review 702
Overloading Binary Operators **702**
+, - Operators 702
*, / Operators 704
&, | Operators 707
Quick Review 710
Overloading Comparison Operators **710**
==, !=, <, >, <=, >= Operators 710
Quick Review 713
Implicit And Explicit Cast Operators **713**
Implicit vs. Explicit Cast 714
Overloaded Cast Operators Example 714
Quick Review 717
The Assignment Operators: Things You Get For Free **717**
Quick Review 717
Summary **718**
Skill-Building Exercises **718**
Suggested Projects **718**
Self-Test Questions **719**
References **719**
Notes **720**

22 Well-Behaved Objects

Introduction **722**
Object Behavior Defined **722**
Fundamental Behavior 722
Copy/Assignment Behavior 723
Equality Behavior 723
Comparison/Ordering Behavior 723
Seven Object Usage Scenarios 723
Fundamental Behavior **724**
Object Creation – Constructors 724
Default Constructor *724*
Private Constructors *724*
Overloaded Constructors *725*
Member Accessibility 725
Horizontal Member Access *725*
Vertical Member Access *725*
Overriding Object.ToString() 726
Static vs. Instance Members 726
Serialization 726
Custom Serialization Example *726*
Quick Review 733
Copy/Assignment Behavior **733**
Value Object vs. Reference Object Assignment 734

Rule Of Thumb – Favor The Class Construct For Complex Types *734*
Shallow Copy vs. Deep Copy 734
Copy Constructors 735
System.ICloneable vs. Object.MemberwiseClone() 738
Quick Review 740
Equality Behavior **741**
Reference Equality vs. Value Equality 741
Rules For Overriding The Object.Equals() Method 741
Overriding The Object.GetHashCode() Method 742
Bloch's Hash Code Generation Algorithm *743*
Ashmore's Hash Code Generation Algorithm *743*
Overriding Object.Equals() and Object.GetHashCode() Methods In The PersonVO Class 743
Quick Review 746
Comparison/Ordering Behavior **747**
Implementing System.IComparable<T> 747
Rules For Implementing The CompareTo(T other) Method *750*
Extending The Comparer<T> Class 751
Quick Review 753
Summary **753**
Skill-Building Exercises **753**
Suggested Projects **753**
Self-Test Questions **754**
References **754**
Notes **755**

23 Three Design Principles

Introduction **758**
The Preferred Characteristics Of An Object-Oriented Architecture **758**
Easy To Understand: How does this thing work? 758
Easy To Reason About: What are the effects of change? 758
Easy To Extend: Where do I add functionality? 759
The Liskov Substitution Principle & Design by Contract **759**
Reasoning About The Behavior Of Supertypes And Subtypes 759
Relationship Between The LSP And DbC *759*
The Common Goal Of The LSP And DbC *760*
C# Support For The LSP And DbC *760*
Designing With The LSP/DbC In Mind 760
Class Declarations Viewed As Behavior Specifications *760*
Quick Review 760
Preconditions, Postconditions, And Class Invariants **761**
Class Invariant 761
Precondition 761
Postcondition 761
An Example 761
A Note On Using The Debug.Assert() Method To Enforce Pre- and Postconditions *763*
Using Incrementer As A Base Class 763
Changing The Preconditions Of Derived Class Methods *765*
Changing The Postconditions Of Derived Class Methods 769
Special Cases Of Preconditions And Postconditions 770
Method Argument Types *770*
Method Return Types *773*
Three Rules Of The Substitution Principle 773
Signature Rule *773*

Methods Rule *773*
Properties Rule *774*
Quick Review 774
The Open-Closed Principle **774**
Achieving The Open-Closed Principle 774
An OCP Example 775
Quick Review 780
The Dependency Inversion Principle **780**
Characteristics Of Bad Software Architecture 780
Characteristics Of Good Software Architecture 781
Selecting The Right Abstractions Takes Experience 781
Quick Review 781
Code Contracts **782**
Downloading And Installing Code Contracts Extensions 782
Code Contracts Example 783
Quick Review 784
Terms and Definitions **785**
Summary **786**
Skill-Building Exercises **787**
Suggested Projects **787**
Self-Test Questions **787**
References **788**
Notes **789**

24 Inheritance, Composition, Interfaces, Polymorphism

Introduction **792**
Inheritance Vs. Composition: The Great Debate **792**
What's The End Game? 793
Flexible Application Architectures *793*
Modularity And Reliability *793*
Architectural Stability Via Managed Dependencies *794*
Knowing When To Accept A Design That's Good Enough 794
Quick Review 794
Inheritance-Based Design **794**
Three Good Reasons To Use Inheritance 794
As A Means To Reason About Code Behavior *795*
To Gain A Measure Of Code Reuse *795*
To Facilitate Incremental Development *795*
Forms Of Inheritance: Meyer's Inheritance Taxonomy 795
Coad's Inheritance Criteria 797
Person - Employee Example Revisited 797
Quick Review 798
The Role Of Interfaces **798**
Reducing Or Limiting Intermodule Dependencies 799
Modeling Dominant, Collateral, and Dynamic Roles 799
Dominant Roles *799*
Collateral Roles *800*
Dynamic Roles *800*
Quick Review 800
Applied Polymorphism **800**
Quick Review 801
Composition-Based Design As A Force Multiplier **801**
Two Types Of Aggregation 801

Polymorphic Containment 801
An Extended Example 802
Quick Review 810
Summary 810
Skill-Building Exercises 811
Suggested Projects 812
Self-Test Questions 813
References 813
Notes 814

25 Helpful Design Patterns

Introduction 816
Software Design Patterns And How They Came To Be 816
What Exactly Is A Software Design Pattern? 816
Origins 816
Pattern Specification 817
Applying Software Design Patterns 817
Quick Review 818
The Singleton Pattern 818
Quick Review 822
The Factory Pattern 822
The Dynamic Factory 822
Advantages Of The Dynamic Factory Pattern 824
Quick Review 824
The Model-View-Controller Pattern 825
Quick Review 827
The Command Pattern 827
Quick Review 835
A Comprehensive Pattern-Based Example 835
Complete Code Listing 835
Com.PulpFreePress.Exceptions 835
Com.PulpFreePress.Common 836
Com.PulpFreePress.Utils 843
Com.PulpFreePress.Commands 847
Com.PulpFreePress.Model 851
Com.PulpFreePress.View 853
Com.PulpFreePress.Controller 863
MSBuild Project File 864
Running The Application 866
Summary 867
Skill-Building Exercises 867
Suggested Projects 868
Self-Test Questions 868
References 869
Notes 869

Appendix A: Helpful Checklists And Tables

Project-Approach Strategy Check-off List 873
Development Cycle 874
Final Project Review Checklist 874

Appendix B: ASCII Table

ASCII Table **875**

Appendix C: Identifier Naming: Writing Self-Commenting Code

Identifier Naming: Writing Self-Commenting Code **879**
Benefits of Self-Commenting Code 879
Coding Convention 879
Type Names (Classes, Structures, Delegates, Enumerations) 879
Constant Names 880
Variable Names 880
Method Names 880
Property Names 881

List of Tables

Table 3-1: Project Approach Strategy 46
Table 3-2: Development Cycle. 47
Table 3-3: Project Specification 48
Table 3-4: Robot Rat Project Nouns and Verbs 51
Table 3-5: Language Feature Study Check-Off List For Robot Rat Project 53
Table 3-6: First Iteration Design Considerations 57
Table 3-7: Second Iteration Design Considerations 58
Table 3-8: Third Iteration Design Considerations 60
Table 3-9: Fourth Iteration Design Considerations 65
Table 3-10: Fifth Iteration Design Considerations 72
Table 3-11: Final Project Review Checklist 80
Table 5-1: .NET Framework Class Libraries Used Heavily In This Book 120
Table 6-1: C# Reserved Keywords 134
Table 6-2: Predefined Type Mappings, Default Values, and Value Ranges 139
Table 6-3: C# Statement Types 141
Table 6-4: Operator Categories by Precedence 142
Table 6-5: Comparison Operator Behavior 148
Table 6-6: Logical Operator Behavior 150
Table 7-1: C# Selection And Iteration Statement Selection Guide 180
Table 8-1: C# Array Properties. 191
Table 8-2: Numeric Formatting. 212
Table 8-3: EISCS Machine Instructions 214
Table 9-1: People Manager Program Class Responsibilities 222
Table 9-2: Method Modifiers 235
Table 11-1: Differences Between Abstract Classes and Interfaces 315
Table 12-1: System Message Categories and their Prefixes. 343
Table 12-2: Partial Listing of Control Events. 352
Table 14-1: Mapping Non-Generic Collections to Their Generic Counterparts 409
Table 14-2: Rules For Implementing IComparable<T>.CompareTo(T other) Method 419
Table 15-1: Runtime Exceptions – Partial Listing. 433
Table 15-2: Exception Class Public Properties 434
Table 20-1: SQL Server Data Types 593
Table 20-2: Project Folder Descriptions 606
Table 20-3: Employee Training Server Application – First Iteration Design Considerations & Decisions 609
Table 20-4: .NET to DbType to SQL Server Type to IDataReader Method Mapping 619
Table 20-5: Employee Training Server Application – Second Iteration Design Considerations And Decisions. 625
Table 20-6: Employee Training Server Application – Third Iteration Design Considerations And Decisions 648
Table 20-7: Employee Training Client Application – Third Iteration Design Considerations And Decisions (Continued) . . . 654
Table 20-8: Employee Training Client Application – Fourth Iteration Design Considerations And Decisions. 657
Table 20-9: Employee Training Client Application – Fifth Iteration Design Considerations And Decisions. 664
Table 20-10: Employee Training Client Application – Sixth Iteration Design Considerations And Decisions 669
Table 21-1: Overloadable Operators 694
Table 22-1: Object Usage Scenario Evaluation Checklist 723
Table 22-2: Rules for Overriding Object.Equals() method. 741
Table 22-3: The GetHashCode() General Contract 742
Table 22-4: Rules For Implementing IComparable<T>.CompareTo(T other) Method 750
Table 23-1: Terms and Definitions Used in this Chapter. 785
Table 24-1: Inheritance Form Descriptions 795

Table 25-1: Pattern Specification Template 817
Table Appendix A-1: Project Approach Strategy 873
Table Appendix A-2: Development Cycle 874
Table Appendix A-3: Final Project Review Checklist 874
Table Appendix B-1: ASCII Table 875
Table Appendix C-1: Type Naming Examples 879
Table Appendix C-2: Constant Naming Examples 880
Table Appendix C-3: Variable Naming Examples 880
Table Appendix C-4: Method Naming Examples 881
Table Appendix C-5: Property Naming Examples 881

List of Figures

Figure 1-1: Isomorphic Mapping Between Problem Domain and Design Domain 10
Figure 1-2: Engineer's Notebook: College-Ruled Composition Book with Sample Pages 18
Figure 2-1: Microsoft.NET Framework Installation Directory 23
Figure 2-2: Partial Directory Listing of the v4.0.30319 Folder 24
Figure 2-3: Creating an Environment Variable 26
Figure 2-4: Editing the Path User Environment Variable 26
Figure 2-5: Command Console Window 27
Figure 2-6: Testing the DOT_NET_FRAMEWORK_HOME Environment Variable 27
Figure 2-7: Testing the Path Environment Variable by Running the C# Compiler 27
Figure 2-8: Creating a New Folder 28
Figure 2-9: Projects Folder Before Setting Folder Options 28
Figure 2-10: Folder Options Dialog Window 29
Figure 2-11: Projects Folder After Setting Folder Options 29
Figure 2-12: Default Command Console Window 30
Figure 2-13: Command Console Properties Dialog 30
Figure 2-14: Setting the Start in Property 31
Figure 2-15: Setting Command Console Layout Properties 32
Figure 2-16: Directory Listing of the Chapter2 Directory Showing the HelloWorld.cs File 32
Figure 2-17: Compiling HelloWorld.cs Using the `csc` C# Compiler Command 33
Figure 2-18: Running the HelloWorld Program 33
Figure 2-19: Compiler Output Showing Compiler Error on Line 6 at Position 39 33
Figure 2-20: C# Language Compiler Errors 34
Figure 2-21: C# Compiler Error CS1002 "; expected" 35
Figure 2-22: Visual C# Express Installation Window 36
Figure 2-23: Visual Studio Express 2010 Initial Start-Up Screen 37
Figure 2-24: New Project Dialog Showing Console Application Selected 37
Figure 2-25: HelloWorld Project View 38
Figure 2-26: IntelliSense Pop-Up Window Showing Available Console Object Methods and Properties 39
Figure 2-27: Updated HelloWorld Visual C# Project 39
Figure 2-28: Saving the HelloWorld Project 40
Figure 2-29: Building HelloWorld Project 40
Figure 2-30: Results of Running the `tree /f` Command from the Command Prompt 41
Figure 3-1: Spiral Development Cycle Deployment 47
Figure 3-2: Robot Rat Viewed as a Collection of Attributes 52
Figure 3-3: Robot Rat Floor Sketch 52
Figure 3-4: Complete Robot Rat Attributes 52
Figure 3-5: RobotRat UML Class Diagram 55
Figure 3-6: Compiling and Testing RobotRat – First Iteration 58
Figure 3-7: Compiling and Testing RobotRat - Second Iteration 59
Figure 3-8: Testing Menu Commands 63
Figure 3-9: A Disturbing Error Message 64
Figure 3-10: Unhandled IndexOutOfRangeException Error Message 64
Figure 3-11: _penPosition State Transition Diagram 66
Figure 3-12: State Transition Diagram for the _direction Variable 67
Figure 3-13: Testing the PrintFloor() Method 72
Figure 3-14: Testing Robot Rat Movement in All Directions 76
Figure 3-15: Robot Rat HTML Documentation Generated with Doxygen 86
Figure 4-1: Typical Apple Mac Pro Computer System 93

Figure 4-2: System Unit Components 93
Figure 4-3: Mac Pro Main Logic Board Block Diagram 94
Figure 4-4: Intel Xeon 5520 Quad-Core Processor 94
Figure 4-5: Two Intel Xeon 5520 Quad Core Microprocessors 95
Figure 4-6: Memory Hierarchy 97
Figure 4-7: Simplified Memory Subsystem Diagram 97
Figure 4-8: Simplified Main Memory Diagram 99
Figure 4-9: Processing Cycle 100
Figure 4-10: Dumb Sort Results 1 102
Figure 4-11: Dumb Sort Results 2 103
Figure 4-12: Dumb Sort Results 3 103
Figure 4-13: Algorithmic Growth Rates 103
Figure 4-14: The C# Compile and Execution Process Overview 104
Figure 4-15: MSIL Disassembler Session Showing Main() Method IL Instructions 105
Figure 4-16: The Common Language Infrastructure Architecture 106
Figure 4-17: Managed Assemblies can be Executed on any System that Implements the Common Language Infrastructure 107
Figure 4-18: Chapter 3's Robot Rat Program Running in the Mono Environment on Apple OS X 108
Figure 4-19: Microsoft .NET Architecture 108
Figure 5-1: .NET Framework Class Library Reference Page 115
Figure 5-2: .NET Development Link Expanded and Class Library Link Highlighted 115
Figure 5-3: Class Library Link Expanded and System Namespace Highlighted 116
Figure 5-4: String Class API Reference Overview Page 116
Figure 5-5: String Class's Constructors Section Partial Listing 117
Figure 5-6: String Class's Methods Section Partial Listing 118
Figure 5-7: String.SubString Page with Collapsed Subheadings 118
Figure 5-8: String.SubString Examples Section Expanded Showing Example Code 119
Figure 5-9: String Class Inheritance Hierarchy 121
Figure 5-10: Obsolete .NET Framework Version 2.0 API Partial Listing by Namespace 122
Figure 6-1: Results of Running Example 6.1 132
Figure 6-2: Results of Compiling Example 6.3 with Improper Main() Method Signature 133
Figure 6-3: Errors Produced when Attempting to Reintroduce a Reserved Keyword 135
Figure 6-4: C# Type Hierarchy 136
Figure 6-5: Results of Running Example 6.6 137
Figure 6-6: The Results of Running Example 6.7 138
Figure 6-7: Value Type Memory Allocation 138
Figure 6-8: Reference Type Memory Allocation 138
Figure 6-9: Results of Calling the Append() Method via the sb1 Variable 139
Figure 6-10: Results of Running Example 6.9 145
Figure 6-11: Results of Running Example 6.10 146
Figure 6-12: Results of Running Example 6.11 147
Figure 6-13: Results of Running Example 6.12 147
Figure 6-14: Results of Running Example 6.13 149
Figure 6-15: Logical AND, OR, and XOR Truth Tables 150
Figure 6-16: Results of Running Example 6.14 151
Figure 6-17: Results of Running Example 6.15 151
Figure 6-18: Results of Running Example 6.16 152
Figure 6-19: Results of Running Example 6.17 153
Figure 6-20: Compiler Warning due to Unreachable Code 154
Figure 6-21: Results of Running Example 6.18 154
Figure 6-22: Results of Running Example 6.19 155
Figure 7-1: if Statement Execution Diagram 160
Figure 7-2: Results of Running Example 7.1 161
Figure 7-3: Typical .NET Error Message Dialog Window 161
Figure 7-4: Unhandled IndexOutOfRangeException Message 162

Figure 7-5: FormatException Error Message 162
Figure 7-6: Results of Running Example 7.2 163
Figure 7-7: Results of Running Example 7.3 164
Figure 7-8: Results of Running Example 7.4 165
Figure 7-9: if/else Statement Execution Diagram 165
Figure 7-10: Results of Running Example 7.5 166
Figure 7-11: Results of Running Example 7.6 166
Figure 7-12: `switch` Statement Execution Diagram 167
Figure 7-13: Results of Running Example 7.7 168
Figure 7-14: Results of Running Example 7.8 169
Figure 7-15: Results of Running Example 7.9 170
Figure 7-16: while Statement Execution Diagram 171
Figure 7-17: Results of Running Example 7.10 172
Figure 7-18: do/while Statement Execution Diagram 172
Figure 7-19: Results of Running Example 7-11 173
Figure 7-20: for Statement Execution Diagram 173
Figure 7-21: Results of Running Example 7.12 174
Figure 7-22: Results of Running Example 7.13 175
Figure 7-23: Results of Running CheckBookBalancer 177
Figure 7-24: Results of Running Example 7.15 178
Figure 7-25: Results of Running Example 7.16 179
Figure 7-26: Results of Running Example 7.17 179
Figure 8-1: Array Elements are Contiguous and Homogeneous 189
Figure 8-2: Declaring a Single-Dimensional Array 189
Figure 8-3: Array-Type Inheritance Hierarchy 191
Figure 8-4: Results of Running Example 8.1 193
Figure 8-5: Memory Representation of Value Type Array int_array Showing Default Initialization 193
Figure 8-6: Results of Running Example 8.2 194
Figure 8-7: Element Values of int_array After Initialization Performed by Second `for` Loop 194
Figure 8-8: Results of Running Example 8.3 195
Figure 8-9: Results of Running Example 8.4 196
Figure 8-10: Results of Running Example 8.5 197
Figure 8-11: State of Affairs After Line 5 of Example 8.5 Executes 197
Figure 8-12: State of Affairs After Line 10 of Example 8.5 Executes. 198
Figure 8-13: State of Affairs After Line 14 of Example 8.5 Executes 198
Figure 8-14: Final State of Affairs: All object_array Elements Point to an Object object 199
Figure 8-15: Results of Running Example 8.6 200
Figure 8-16: Results of Running Example 8.7 201
Figure 8-17: Results of Running Example 8.8 203
Figure 8-18: Rectangular Array Declaration Syntax 204
Figure 8-19: Accessing Two-Dimensional Array Elements 204
Figure 8-20: Results of Running Example 8.9 205
Figure 8-21: Results of Running Example 8.10 206
Figure 8-22: Array Declaration Syntax for a Two-Dimensional Ragged Array 206
Figure 8-23: Results of Running Example 8.11 207
Figure 8-24: Results of Running Example 8.12 209
Figure 8-25: Results of Running Example 8.13 211
Figure 8-26: Results of Running Example 8.14 211
Figure 9-1: People Management Program Project Specification 222
Figure 9-2: Class Diagram for People Manager Classes 224
Figure 9-3: Static and Non-Static Fields 226
Figure 9-4: Results of Running Example 9.1 226
Figure 9-5: Error Resulting from an Attempt to Assign to a Readonly Field 227
Figure 9-6: Results of Running Example 9.3 228

Figure 9-7: Results of Running Example 9.4 228
Figure 9-8: Results of Running Example 9.5 230
Figure 9-9: Horizontal Access Controlled via Access Modifiers public and private 233
Figure 9-10: Method Definition Structure 235
Figure 9-11: Results of Running Example 9.10 242
Figure 9-12: Results of Running Example 9.12 244
Figure 9-13: Results of Running Example 9.14 246
Figure 9-14: Results of Running Example 9.16 248
Figure 9-15: Results of Running Example 9.21 251
Figure 9-16: Results of Running Example 9.23 253
Figure 9-17: Default Value Parameter Behavior 255
Figure 9-18: Reference Parameter Behavior – Using `ref` Modifier 255
Figure 9-19: Results of Running Example 9.24 257
Figure 9-20: Results of Running Example 9.25 258
Figure 9-21: Results of Running Example 9.26 258
Figure 9-22: Results of Running Example 9.27 259
Figure 9-23: Structures vs. Value Types 261
Figure 9-24: Results of Running Example 9.28 262
Figure 9-25: Circular Linked List with Three Nodes 267
Figure 10-1: UML Diagram Showing Simple Aggregation 275
Figure 10-2: Part Class Shared Between Simple Aggregate Classes 276
Figure 10-3: UML Diagram Showing Composite Aggregation 276
Figure 10-4: Simple Aggregation Example 277
Figure 10-5: Results of Running Example 10.3 278
Figure 10-6: Composite Aggregation Example 278
Figure 10-7: Results of Running Example 10.6 279
Figure 10-8: Sequence Diagram – Simple Aggregation 280
Figure 10-9: Sequence Diagram – Composite Aggregation 280
Figure 10-10: Engine Simulation Project Specification 282
Figure 10-12: Engine Class Diagram 282
Figure 10-11: Engine Simulation Class Diagram 283
Figure 10-13: Create Engine Object Sequence 285
Figure 10-14: Result of Running Example 10.7 286
Figure 10-15: Simple Aggregation Class Diagram 292
Figure 10-16: Composite Aggregation Class Diagram 292
Figure 11-1: Inheritance Hierarchy Illustrating Generalized and Specialized Behavior 299
Figure 11-2: UML Class Diagram Showing DerivedClass Inheriting from BaseClass 301
Figure 11-3: UML Diagram of BaseClass and DerivedClass Showing Fields, Properties, and Methods 301
Figure 11-4: Results of Running Example 11.3 303
Figure 11-5: UML Diagram Showing Student Class Inheritance Hierarchy 305
Figure 11-6: Results of Running Example 11.6 308
Figure 11-7: Results of Running Example 11.7 308
Figure 11-8: UML Class Diagram For BaseClass & DerivedClass 309
Figure 11-9: Results of Running Example 11.3 with Modified Versions of BaseClass and DerivedClass 310
Figure 11-10: Expressing an Abstract Class in the UML 312
Figure 11-11: UML Class Diagram Showing the AbstractClass and DerivedClass Inheritance Hierarchy 312
Figure 11-12: Results of Running Example 11.12 314
Figure 11-13: Two Types of UML Interface Diagrams 315
Figure 11-14: UML Diagram Showing the Simple Form of Realization 316
Figure 11-15: UML Diagram Showing the Expanded Form of Realization 316
Figure 11-16: UML Diagram Showing the MessagePrinter Class Implementing the IMessagePrinter Interface 317
Figure 11-17: Results of Running Example 11.15 318
Figure 11-18: Employee Class Inheritance Hierarchy 320
Figure 11-19: Results of Running Example 11.20 323

Figure 11-20: Engine Simulation UML Class Diagram 324
Figure 11-21: Results of Running the EngineTestApp 325
Figure 12-1: Form Class Inheritance Hierarchy 341
Figure 12-2: Results of Running Example 12.1 342
Figure 12-3: A Standard Window can be Resized by Dragging the Lower Right Corner 342
Figure 12-4: Windows Message Routing (Message Pump) 343
Figure 12-5: Results of Running Example 12.2 345
Figure 12-6: Screen Coordinate System 346
Figure 12-7: Window Coordinates 346
Figure 12-8: Results of Running Example 12.3 347
Figure 12-9: Running Example 12.4 via the Command Line with the Name of the Image WCC_2.jpg 349
Figure 12-10: Running Example 12.4 with No Image 350
Figure 12-11: Results of Running Example 12.5 351
Figure 12-12: Results of Running Example 12.6 with Different Text in the TextBox 354
Figure 12-13: UML Class Diagram Showing Separate GUI and Application/Event Handler Classes 355
Figure 12-14: Results of Running Example 12.8 – GUI Events Handled in Separate Object 358
Figure 12-15: Results of Running Example 12.10 – Buttons Adjust when Window is Resized 360
Figure 12-16: Results of Running Example 12.12 After Several Buttons have been Clicked 362
Figure 12-17: Window and Menu Structure of Menu Demo Program 363
Figure 12-18: Results of Running Example 12.14 and Adding Several Buttons and Text Boxes 366
Figure 12-19: Results of Running Example 12.16 – Double-clicking the First Line 368
Figure 13-1: Event Publisher and Subscriber 376
Figure 13-2: Event Publisher and Subscriber 377
Figure 13-3: Minute Tick UML Class Diagram 378
Figure 13-4: Results of Running Example 13.5 380
Figure 13-5: Water Tank System UML Class Diagram 381
Figure 13-6: Results of Running Example 13.11 387
Figure 14-1: Results of Testing DynamicArray 398
Figure 14-2: Results of Running Example 14.3 399
Figure 14-3: Results of Running Example 14.4 400
Figure 14-4: Array of Object References Before Insertion 401
Figure 14-5: New Reference to be Inserted at Array Element 3 (index 2) 401
Figure 14-6: Array After New Reference Insertion 402
Figure 14-7: Linked List Node Organization 402
Figure 14-8: Linked List Before New Element Insertion 403
Figure 14-9: New Reference Being Inserted Into Second Element Position 403
Figure 14-10: References of Previous, New, and Next List Elements must be Manipulated 403
Figure 14-11: Linked List Insertion Complete 404
Figure 14-12: A Hash Function Transforms a Key Value into an Array Index 404
Figure 14-13: Hash Table Collisions are Resolved by Linking Nodes Together 404
Figure 14-14: Red-Black Tree Node Data Elements 405
Figure 14-15: Red-Black Tree After Inserting Integer Values 9, 3, 5, 6, 7, 8, 4, 1 405
Figure 14-16: A Stack After Several Push and Pop Operations 406
Figure 14-17: A Queue After Several Enqueue and Dequeue Operations 406
Figure 14-18: Results of Running Example 14.6 412
Figure 14-19: Results of Running Example 14.8 414
Figure 14-20: Results of Running Example 14.9 415
Figure 14-21: Results of Running Example 14.11 417
Figure 14-22: Results of Running Example 14.13 420
Figure 14-23: Results of Running Example 14.15 422
Figure 14-24: Results of Running Example 14.16 423
Figure 15-1: Exception Information Table 431
Figure 15-2: Exception Class Hierarchy 432
Figure 15-3: Getting Exception Information from MSDN 434

Figure 15-4: Results of Running Example 15.1 436
Figure 15-5: Results of Running Example 15.2 437
Figure 15-6: Results of Running Example 15.3 438
Figure 15-7: Results of Running Example 15.4 438
Figure 15-8: Results of Running Example 15.7 441
Figure 16-1: List of Running Applications 449
Figure 16-2: Partial List of Processes Running on the Same Computer 449
Figure 16-3: Processes and their Threads Executing in a Single-Processor Environment 450
Figure 16-4: Processes and their Threads Executing in a Multiprocessor Environment 451
Figure 16-5: SingleThreadedVacation Program Output 453
Figure 16-6: MultiThreadedVacation Program Output - Partial Listing 455
Figure 16-7: Thread States and Transition Initiators 456
Figure 16-8: Results of Running Example 16.3 457
Figure 16-9: Results of Running Example 16.4 458
Figure 16-10: Results of Running Example 16.5 459
Figure 16-11: Results of Running Example 16.6 460
Figure 16-12: Results of Running Example 16.7 461
Figure 16-13: Results of Running Example 16.8 462
Figure 16-14: Results of Running Example 16.9 462
Figure 16-15: One Particular Result of Running Example 16.10 468
Figure 16-16: Partial Result of Running Example 16.11 470
Figure 16-17: Results of Running Example 16.12 472
Figure 16-18: Results of Running Example 16.13 473
Figure 16-19: Results of Running Example 16.14 474
Figure 17-1: Simplified View of Service Layers 480
Figure 17-2: Typical Directory Structure 481
Figure 17-3: The Absolute Path to the Reports\East\Q2.xls File 482
Figure 17-4: Results of Running Example 17.1 483
Figure 17-5: Results of Running Example 17.3 487
Figure 17-6: Results of Running Example 17.4 489
Figure 17-7: Results of Running Example 17.6 492
Figure 17-8: Results of Running Example 17.8 494
Figure 17-9: Legacy Datafile Adapter Project Specification 495
Figure 17-10: Header and Record Length Analysis 496
Figure 17-11: Monitor.Enter()/Monitor.Exit() vs. the lock Keyword 497
Figure 17-12: Results of Running Example 17.16 Once 513
Figure 17-13: Results of Running Example 17.19 516
Figure 17-14: Results of Running Example 17.21 and Selecting Three Files 519
Figure 18-1: A Local Area Network (LAN). Network Resources Connected via a Switch 529
Figure 18-2: Local Area Network Connected to the Internet 530
Figure 18-3: The Internet – A Network of Networks Communicating via Internet Protocols 530
Figure 18-4: Client and Server Hardware and Applications 532
Figure 18-5: Client and Server Applications Physically Deployed to the Same Computer 533
Figure 18-6: Running Multiple Clients on Same Hardware 533
Figure 18-7: Client and Server Applications Deployed on Different Computers 534
Figure 18-8: A Multilayered Application 535
Figure 18-9: Physically Deploying Logical Application Tiers on Same Computer 535
Figure 18-10: Logical Application Tiers Physically Deployed to Different Computers 536
Figure 18-11: TCP/IP Protocol Stack 537
Figure 18-12: Internet Protocol Stack Operations 539
Figure 19-1: .NET Remoting Architecture 546
Figure 19-2: RemotingServer Waiting for Something to do 549
Figure 19-3: Results of Running RemotingServer and RemotingClient with a SingleCall Mode Remote Object 550
Figure 19-4: Results of Hosting TestClass Remote Object in Singleton Mode 551

Figure 19-5: Results of Accessing a Remote Object via an Interface 553
Figure 19-6: Results of Running RemotingServer and RemotingClient with Configuration Files 555
Figure 19-7: Results of Sending a Collection of Person Objects to a Remoting Client 559
Figure 19-8: Server Application Listens on a Host and Port for Incoming TcpClient Connections 561
Figure 19-9: TcpListener Accepts Incoming TcpClient Connection 561
Figure 19-10: TcpClients Communicate via a NetworkStream using StreamReader and StreamWriter Objects 561
Figure 19-11: Results of Running the EchoClient and EchoServer Applications 563
Figure 19-12: Two Clients Connected to MultiThreadedClientServer 565
Figure 19-13: Results of Running MultiIPEchoServer and EchoClient (Mod 1) Applications 568
Figure 19-14: Results of Running SurrealistEchoServer and EchoClient (Mod 2) 573
Figure 20-1: Employee Training Server Application Architecture 581
Figure 20-2: Installing New SQL Server Instance or Updating Existing Instance 582
Figure 20-3: SQL Express 2014 Feature Selection Dialog 582
Figure 20-4: Results of Testing SQL Server Express Edition Installation 583
Figure 20-5: Management Studio Login Dialog 584
Figure 20-6: SQL Management Studio Main Window 584
Figure 20-7: Enterprise Library Installation Directory – Need to Run install-packages.ps1 with PowerShell 585
Figure 20-8: Enterprise Library Directory After Running install-packages.ps1 585
Figure 20-9: Enterprise Library Configuration File Creation Tool 587
Figure 20-10: Contents of the SimpleConnection Project Directory Before Compiling 587
Figure 20-11: Results of Running the SimpleConnection Application 588
Figure 20-12: The Primary Key of One Table Can Serve as the Foreign Key in a Related Table 589
Figure 20-13: SQL Server's Default Databases 590
Figure 20-14: Creating EmployeeTraining Database with SQL Command Utility 590
Figure 20-15: Checking on the Existence of the EmployeeTraining Database 591
Figure 20-16: Results of Executing the create_database.sql Script 592
Figure 20-17: Results of Executing create_tables.sql Database Script 593
Figure 20-18: Results of Running create_test_data.sql Database Script 596
Figure 20-19: Results of Executing a Simple Select Statement 596
Figure 20-20: Selecting Specific Rows with select Statement 597
Figure 20-21: Inserting More Test Data with the create_test_data.sql Database Script 598
Figure 20-22: Results of Limiting Data Returned from select Statement with where Clause 598
Figure 20-23: Results of Executing the Previous Query 598
Figure 20-24: Changing Coralie Powell's Last Name to Miller with the Update Statement 599
Figure 20-25: Deleting all Employees whose Last Names = "Miller" 600
Figure 20-26: Verifying the Creation of the tbl_employee_training Table 601
Figure 20-27: Selecting EmployeeIDs from tbl_employee 602
Figure 20-28: Results of Running the Previous SQL Query 604
Figure 20-29: Results of Running the Previous SQL Query 604
Figure 20-30: Results of Executing a Cascade Delete and Checking the Results 605
Figure 20-31: Employee Training Project Folder Arrangement 606
Figure 20-32: EmployeeVO and EmployeeDAO Class Diagram 610
Figure 20-33: Results of Running the CompileVO Target using the MSBuild Utility 613
Figure 20-34: Build Warnings From Conflicting Type Declarations 614
Figure 20-35: Initial State of the EmployeeTrainingServer Application Window 623
Figure 20-36: Employee Picture Loaded and Create Button Enabled 624
Figure 20-37: Testing with More Employee Pictures 624
Figure 20-38: Testing the Insertion and Retrieval of a Large Image 624
Figure 20-39: TrainingDAO and TrainingVO Class Diagram 625
Figure 20-40: EmployeeAdminBO UML Class Diagram 626
Figure 20-41: Collapsed Code Regions in Notepad++ 636
Figure 20-42: Modified Test Application 639
Figure 20-43: EmployeeTrainingRemoteObject UML Class Diagram 649
Figure 20-44: EmployeeTrainingServer Running and Ready For Remote Connections 654

Figure 20-45: Client Project Directory Structure 655
Figure 20-46: Running Client Application via the MSBuild Project's Run Target 657
Figure 20-47: EmployeeTrainingClient UML Class Diagram 658
Figure 20-48: Mock-up Sketch of the EmployeeTrainingApplication GUI 658
Figure 20-49: EmployeeTrainingClient Initial Display on Startup – Something's Not Quite Right! 662
Figure 20-50: Employee's Related Training Shown in Training DataGridView 662
Figure 20-51: Results of Clicking on a Employee with a Picture - a RemotingException is Thrown 663
Figure 20-52: Bitmap Class Usage Note 663
Figure 20-53: EmployeeTrainingClient Application with Employee's Picture Displayed in the PictureBox 669
Figure 20-54: Employee Form Mock-up 670
Figure 20-55: Training Form Mock-up 674
Figure 20-56: Main Application Window with Edit Menu Open to Reveal Revised Menu Structure 685
Figure 20-57: Edit Menu Items Disabled 685
Figure 20-58: Empty Employee Data Entry Form 686
Figure 20-59: Employee Form Fully Populated and Submit Button Enabled 686
Figure 20-60: Training Form Empty and Filled 686
Figure 21-1: Method Signature for Overloaded Unary Operator 695
Figure 21-2: Method Signature for Overloaded Unary Logical Operator 695
Figure 21-3: Results of Running Example 21.2 697
Figure 21-4: Results of Running Example 21.4 698
Figure 21-5: Results of Running Example 21.6 699
Figure 21-6: Results of Running Example 21.9 701
Figure 21-7: Overloaded Binary + Operator Signature that Operates on Two Objects of Type MyType 702
Figure 21-8: Overloaded Binary + Operator Signature that Operates on Objects of MyType and Integer 702
Figure 21-9: Results of Running Example 21.11 705
Figure 21-10: Results of Running Example 21.13 707
Figure 21-11: Results of Running Example 21.15 709
Figure 21-12: Method Signature for Overloaded Equality Operator 710
Figure 21-13: Compiler Warning – == and != Operators Need Special Attention 713
Figure 21-14: Results of Running Example 21.17 713
Figure 21-15: Method Signatures for Implicit and Explicit Cast Operators 714
Figure 21-16: Results of Running Example 21.19 717
Figure 21-17: Results of Running Example 21.20 717
Figure 22-1: Horizontal and Vertical Member Accessibility 725
Figure 22-2: Running Example 22.2 Several Times 729
Figure 22-3: Running MainApp in the Read Mode 733
Figure 22-4: Results of Running MainApp Several More Times in the Append Mode then Read Mode 733
Figure 22-5: Concept of a Shallow Copy 735
Figure 22-6: Concept of a Deep Copy 735
Figure 22-7: Results of Running Example 22.6 738
Figure 22-8: Results of Running Example 22.8 740
Figure 22-9: Results of Running Example 22.10 747
Figure 22-10: Results of Running Example 22.12 751
Figure 22-11: Results of Running Example 22.14 752
Figure 23-1: Results of Running Example 23.2 763
Figure 23-2: Results of Running Example 23.4 765
Figure 23-3: Results of Running Example 23.6 768
Figure 23-4: Results of Running Example 24.8 770
Figure 23-5: Strong vs. Weak Types 771
Figure 23-6: Results of Running Example 23.12 773
Figure 23-7: Naval Fleet Class Inheritance Hierarchy 776
Figure 23-8: Results of Running Example 23.22 780
Figure 23-9: Traditional Top-Down Functional Dependencies 781
Figure 23-10: Code Contracts for .NET Page 782

Figure 23-11: Results of Compiling, Calling ccrewrite, and Executing Examples 23.25 and 23.26 785
Figure 24-1: Meyer's Inheritance Taxonomy 796
Figure 24-2: Person-Employee Inheritance Diagram 798
Figure 24-3: Revised Person - Employee Example 802
Figure 24-4: Results of Running Example 24.11 810
Figure 25-1: Results of Running Example 25.4 821
Figure 25-2: Results of Running Example 25.8 824
Figure 25-3: Model-View-Controller Pattern 825
Figure 25-4: Results of Running Example 25.14 and Clicking the "Next Message" Button Several Times 827
Figure 25-5: Command Pattern Version of Inspirational Messages 834
Figure 25-6: EmployeeMVC Project Directory Structure 836
Figure 25-7: Interacting with the Employee Management Application 866

