

DETAILED CONTENTS

PREFACE

WELCOME – AND THANK YOU!	vii
TARGET AUDIENCE	vii
APPROACH(ES)	vii
PEDAGOGY – I MEAN, HOW THIS BOOK’S ARRANGED	viii
<i>LEARNING OBJECTIVES</i>	<i>viii</i>
<i>INTRODUCTION</i>	<i>viii</i>
<i>CONTENT</i>	<i>viii</i>
<i>QUICK REVIEWS</i>	<i>viii</i>
<i>SUMMARY</i>	<i>viii</i>
<i>SKILL-BUILDING EXERCISES</i>	<i>viii</i>
<i>SUGGESTED PROJECTS</i>	<i>viii</i>
<i>SELF-TEST QUESTIONS</i>	<i>ix</i>
<i>REFERENCES</i>	<i>ix</i>
<i>NOTES</i>	<i>ix</i>
TYPOGRAPHICAL FORMATS	ix
<i>THIS IS AN EXAMPLE OF A FIRST LEVEL SUBHEADING</i>	<i>ix</i>
<i>THIS IS AN EXAMPLE OF A SECOND LEVEL SUBHEADING</i>	<i>ix</i>
<i>SOURCE CODE FORMATTING</i>	<i>ix</i>
SUPPORTSITE™ WEBSITE	ix
PROBLEM REPORTING	x
ABOUT THE AUTHOR	x
ACKNOWLEDGMENTS	x

I AN APPROACH TO THE ART OF PROGRAMMING

INTRODUCTION	4
<i>THE DIFFICULTIES YOU WILL ENCOUNTER LEARNING C#</i>	<i>4</i>
<i>REQUIRED SKILLS</i>	<i>4</i>
<i>THE PLANETS WILL COME INTO ALIGNMENT</i>	<i>4</i>
<i>HOW THIS CHAPTER WILL HELP YOU</i>	<i>5</i>
PERSONALITY TRAITS FOUND IN GREAT PROGRAMMERS	5
<i>CREATIVE</i>	<i>5</i>
<i>TENACIOUS</i>	<i>5</i>
<i>RESILIENT</i>	<i>5</i>
<i>METHODICAL</i>	<i>5</i>
<i>METICULOUS</i>	<i>6</i>
<i>HONEST</i>	<i>6</i>
<i>PROACTIVE</i>	<i>6</i>
<i>HUMBLE</i>	<i>6</i>
<i>BE A GENERALIST AND A JUST-IN-TIME SPECIALIST</i>	<i>6</i>
PROJECT MANAGEMENT	6
<i>THREE SOFTWARE DEVELOPMENT ROLES</i>	<i>6</i>
<i>ANALYST</i>	<i>6</i>
<i>ARCHITECT</i>	<i>7</i>
<i>PROGRAMMER</i>	<i>7</i>
<i>A PROJECT-APPROACH STRATEGY</i>	<i>7</i>

<i>You Have Been Handed A Project – Now What?</i>	7
<i>STRATEGY AREAS OF CONCERN</i>	8
<i>Think Abstractly</i>	9
<i>THE STRATEGY IN A NUTSHELL</i>	10
<i>Applicability To The Real World</i>	10
THE ART OF PROGRAMMING	10
<i>DON'T START AT THE COMPUTER</i>	10
<i>Inspiration Strikes At The Weirdest Time</i>	10
<i>OWN YOUR OWN COMPUTER</i>	11
<i>You Either Have Time And No Money, Or Money And No Time</i>	11
<i>The Family Computer Is Not Going To Cut It!</i>	11
<i>SET THE MOOD</i>	11
<i>LOCATION, LOCATION, LOCATION</i>	11
<i>CONCEPT OF THE FLOW</i>	11
<i>THE STAGES OF FLOW</i>	12
<i>BE EXTREME</i>	12
<i>THE PROGRAMMING CYCLE</i>	12
<i>THE PROGRAMMING CYCLE SUMMARIZED</i>	13
<i>A Helpful Trick: Stubbing</i>	13
<i>Fix The First Compiler Error First</i>	14
MANAGING PROJECT COMPLEXITY	14
<i>CONCEPTUAL COMPLEXITY</i>	14
<i>MANAGING CONCEPTUAL COMPLEXITY</i>	14
<i>THE UNIFIED MODELING LANGUAGE (UML)</i>	15
<i>PHYSICAL COMPLEXITY</i>	15
<i>MANAGING PHYSICAL COMPLEXITY</i>	15
<i>THE RELATIONSHIP BETWEEN PHYSICAL AND CONCEPTUAL COMPLEXITY</i>	15
<i>MAXIMIZE COHESION – MINIMIZE COUPLING</i>	15
SUMMARY	16
SKILL-BUILDING EXERCISES	16
SUGGESTED PROJECTS	16
SELF-TEST QUESTIONS	17
REFERENCES	17
NOTES	17

2 SMALL VICTORIES: CREATING C# PROJECTS

INTRODUCTION	20
CREATING PROJECTS WITH MICROSOFT C#.NET COMMAND-LINE TOOLS	20
<i>DOWNLOADING AND INSTALLING THE .NET FRAMEWORK</i>	20
<i>DOWNLOADING AND INSTALLING NOTEPAD++</i>	22
<i>CONFIGURING YOUR DEVELOPMENT ENVIRONMENT</i>	22
<i>ENVIRONMENT VARIABLES</i>	22
<i>CREATING A PROJECT FOLDER</i>	25
<i>SETTING FOLDER OPTIONS</i>	25
<i>CREATING A SHORTCUT TO THE COMMAND CONSOLE AND SETTING ITS PROPERTIES</i>	26
<i>TESTING THE CONFIGURATION</i>	29
<i>CREATING THE SOURCE FILE</i>	29
<i>COMPILING THE SOURCE FILE</i>	29
<i>EXECUTING THE APPLICATION</i>	30
<i>Quick Review</i>	31
CREATING PROJECTS WITH MICROSOFT VISUAL C# EXPRESS	32
<i>Download and Install Visual C# Express</i>	33
<i>Quick Tour Of Visual C# Express</i>	33
<i>SELECT PROJECT TYPE</i>	33
<i>SAVING THE PROJECT</i>	36
<i>BUILD THE PROJECT</i>	36

<i>LOCATING THE PROJECT EXECUTABLE FILE</i>	36
<i>EXECUTE THE PROJECT</i>	38
<i>WHERE TO GO FOR MORE INFORMATION ABOUT VISUAL C# EXPRESS</i>	38
<i>Quick Review</i>	38
SUMMARY	38
Skill-Building Exercises	39
SUGGESTED PROJECTS	39
Self-Test Questions	39
REFERENCES	40
NOTES	40

3 PROJECT WALKTHROUGH

INTRODUCTION	42
THE PROJECT-APPROACH STRATEGY SUMMARIZED	42
DEVELOPMENT CYCLE	43
PROJECT SPECIFICATION	44
<i>ANALYZING THE PROJECT SPECIFICATION</i>	45
<i>APPLICATION REQUIREMENTS STRATEGY AREA</i>	45
<i>PROBLEM-DOMAIN STRATEGY AREA</i>	46
<i>LANGUAGE-FEATURES STRATEGY AREA</i>	48
<i>DESIGN STRATEGY AREA</i>	50
DEVELOPMENT CYCLE: FIRST ITERATION	51
<i>PLAN (FIRST ITERATION)</i>	51
<i>CODE (FIRST ITERATION)</i>	52
<i>TEST (FIRST ITERATION)</i>	52
<i>INTEGRATE/TEST (FIRST ITERATION)</i>	52
DEVELOPMENT CYCLE: SECOND ITERATION	52
<i>PLAN (SECOND ITERATION)</i>	53
<i>CODE (SECOND ITERATION)</i>	53
<i>TEST (SECOND ITERATION)</i>	53
<i>INTEGRATE/TEST (SECOND ITERATION)</i>	54
DEVELOPMENT CYCLE: THIRD ITERATION	54
<i>PLAN (THIRD ITERATION)</i>	54
<i>CODE (THIRD ITERATION)</i>	55
<i>INTEGRATE/TEST (THIRD ITERATION)</i>	57
<i>A BUG IN THE PROGRAM</i>	57
DEVELOPMENT CYCLE: FOURTH ITERATION	59
<i>PLAN (FOURTH ITERATION)</i>	59
<i>IMPLEMENTING STATE TRANSITION DIAGRAMS</i>	60
<i>IMPLEMENTING THE PRINTFLOOR() METHOD</i>	60
<i>CODE (FOURTH ITERATION)</i>	61
<i>TEST (FOURTH ITERATION)</i>	62
<i>INTEGRATE/TEST (FOURTH ITERATION)</i>	63
DEVELOPMENT CYCLE: FIFTH ITERATION	63
<i>PLAN (FIFTH ITERATION)</i>	63
<i>CODE (FIFTH ITERATION)</i>	64
<i>TEST (FIFTH ITERATION)</i>	65
<i>INTEGRATE/TEST (FIFTH ITERATION)</i>	65
FINAL CONSIDERATIONS	66
COMPLETE ROBOTRAT.CS SOURCE CODE LISTING	67
SUMMARY	73
Skill-Building Exercises	73
SUGGESTED PROJECTS	73
Self-Test Questions	73
REFERENCES	74

NOTES	74
-------------	----

4 COMPUTERS, PROGRAMS, AND ALGORITHMS

INTRODUCTION	76
WHAT IS A COMPUTER?	76
<i>COMPUTER VS. COMPUTER SYSTEM</i>	<i>76</i>
<i>COMPUTER SYSTEM.....</i>	<i>76</i>
<i>PROCESSOR</i>	<i>78</i>
<i>THREE ASPECTS OF PROCESSOR ARCHITECTURE</i>	<i>79</i>
<i>FEATURE SET.....</i>	<i>79</i>
<i>FEATURE SET IMPLEMENTATION.....</i>	<i>79</i>
<i>FEATURE SET ACCESSIBILITY.....</i>	<i>79</i>
MEMORY ORGANIZATION	79
<i>MEMORY BASICS</i>	<i>80</i>
<i>MEMORY HIERARCHY.....</i>	<i>80</i>
<i>Bits, Bytes, Words.....</i>	<i>80</i>
<i>ALIGNMENT AND ADDRESSABILITY</i>	<i>81</i>
WHAT IS A PROGRAM?	82
<i>TWO VIEWS OF A PROGRAM</i>	<i>82</i>
<i>THE HUMAN PERSPECTIVE.....</i>	<i>82</i>
<i>THE COMPUTER PERSPECTIVE.....</i>	<i>82</i>
THE PROCESSING CYCLE	82
<i>FETCH</i>	<i>83</i>
<i>DECODE</i>	<i>83</i>
<i>EXECUTE</i>	<i>83</i>
<i>STORE</i>	<i>83</i>
<i>Why A Program Crashes.....</i>	<i>83</i>
ALGORITHMS	83
<i>Good vs. Bad Algorithms</i>	<i>83</i>
<i>DON'T REINVENT THE WHEEL!</i>	<i>86</i>
VIRTUAL MACHINES AND THE COMMON LANGUAGE INFRASTRUCTURE	86
<i>VIRTUAL MACHINES</i>	<i>87</i>
<i>THE COMMON LANGUAGE INFRASTRUCTURE (CLI)</i>	<i>87</i>
<i>FOUR PARTS OF THE COMMON LANGUAGE INFRASTRUCTURE</i>	<i>87</i>
<i>THE CROSS PLATFORM PROMISE.....</i>	<i>89</i>
SUMMARY	90
SKILL-BUILDING EXERCISES	90
SUGGESTED PROJECTS	91
SELF-TEST QUESTIONS	91
REFERENCES	92
NOTES	92

5. NAVIGATING .NET FRAMEWORK DOCUMENTATION

INTRODUCTION	94
MSDN: THE DEFINITIVE SOURCE FOR API INFORMATION	94
DISCOVERING INFORMATION ABOUT CLASSES	96
<i>GENERAL OVERVIEW PAGE</i>	<i>96</i>
<i>CLASS MEMBER PAGE</i>	<i>97</i>
<i>GETTING INFORMATION ON OTHER CLASS MEMBERS</i>	<i>98</i>
<i>Quick Review</i>	<i>100</i>
THE BASE CLASS LIBRARIES (BCL)	100
<i>Quick Review</i>	<i>101</i>
NAVIGATING AN INHERITANCE HIERARCHY	101

<i>Quick Review</i>	102
BEWARE OBSOLETE APIs	102
SUMMARY	103
SKILL-BUILDING EXERCISES	103
SUGGESTED PROJECTS	104
SELF-TEST QUESTIONS	104
REFERENCES	104
NOTES	105

6 Simple C# Programs

INTRODUCTION	110
WHAT IS A C# PROGRAM?	110
A Simple Console Application	111
<i>Definition Of Terms: Application, Assembly, Module, and Entry Point</i>	111
<i>Structure Of A Simple Application</i>	111
<i>Purpose Of The Main() Method</i>	112
<i>Main() Method Signatures</i>	112
<i>Quick Review</i>	113
Identifiers And Reserved Keywords	113
<i>Identifier Naming Rules</i>	114
<i>Quick Review</i>	115
Types	115
<i>Value Type Variables vs. Reference Type Variables</i>	116
<i>Value Type Variables</i>	116
<i>Reference Type Variables</i>	116
<i>Maybe Some Pictures Will Help</i>	117
<i>Mapping Predefined Types To System Structures</i>	118
<i>Quick Review</i>	119
STATEMENTS, EXPRESSIONS, AND OPERATORS	119
<i>Statement Types</i>	119
<i>Operators And Their Use</i>	120
<i>Operator Precedence And Associativity</i>	121
<i>Forcing Operator Precedence And Associativity Order With Parentheses</i>	121
<i>Operators And Operands</i>	121
<i>Operator Usage Examples</i>	122
<i>Primary Expression Operators</i>	122
<i>Unary Expression Operators</i>	122
<i>Multiplicative Expression Operators</i>	123
<i>Additive Expression Operators</i>	124
<i>Shift Expression Operators</i>	124
<i>Relational, Type-Testing, and Equality Expression Operators</i>	125
<i>Logical AND, OR, and XOR Expression Operators</i>	126
<i>Conditional AND and OR Expression Operators</i>	129
<i>Conditional (Ternary) Expression Operator</i>	129
<i>Assignment Expression Operators</i>	130
<i>Quick Review</i>	131
SUMMARY	131
SKILL-BUILDING EXERCISES	131
SUGGESTED PROJECTS	132
SELF-TEST QUESTIONS	132
REFERENCES	133
NOTES	133

7 CONTROLLING THE FLOW OF PROGRAM EXECUTION

INTRODUCTION	136
SELECTION STATEMENTS	136
<i>If STATEMENT</i>	136
<i>Handling PROGRAM ERROR Conditions</i>	137
<i>EXECUTING Code Blocks In If STATEMENTS</i>	139
<i>EXECUTING CONSECUTIVE If STATEMENTS</i>	139
<i>If/Else STATEMENT</i>	140
<i>CHAINED If/Else STATEMENTS</i>	141
<i>Switch STATEMENT</i>	142
<i>Implicit CASE Fall-Through</i>	143
<i>NESTED Switch STATEMENT</i>	144
<i>Quick Review</i>	145
ITERATION STATEMENTS	145
<i>While STATEMENT</i>	145
<i>PERSONALITY Of THE While STATEMENT</i>	145
<i>Do/While STATEMENT</i>	146
<i>PERSONALITY Of THE Do/While STATEMENT</i>	147
<i>FOR STATEMENT</i>	148
<i>HOW THE FOR STATEMENT IS RELATED TO THE While STATEMENT</i>	148
<i>PERSONALITY Of THE FOR STATEMENT</i>	148
<i>NESTING ITERATION STATEMENTS</i>	149
<i>MIXING SELECTION AND ITERATION STATEMENTS: A POWERFUL COMBINATION</i>	150
<i>Quick Review</i>	151
BREAK, CONTINUE, AND GOTO	151
<i>BREAK STATEMENT</i>	152
<i>CONTINUE STATEMENT</i>	152
<i>GOTO STATEMENT</i>	153
<i>Quick Review</i>	153
SELECTION AND ITERATION STATEMENT SELECTION TABLE	154
SUMMARY	155
SKILL-BUILDING EXERCISES	155
SUGGESTED PROJECTS	157
SELF-TEST QUESTIONS	158
REFERENCES	159
NOTES	159

8 ARRAYS

INTRODUCTION	162
WHAT IS AN ARRAY?	162
<i>Specifying ARRAY Types</i>	163
<i>Quick Review</i>	164
FUNCTIONALITY PROVIDED BY C# ARRAY TYPES	164
<i>ARRAY-TYPE INHERITANCE HIERARCHY</i>	164
<i>SPECIAL PROPERTIES Of C# ARRAYS</i>	165
<i>Quick Review</i>	165
CREATING AND USING SINGLE-DIMENSIONAL ARRAYS	166
<i>ARRAYS Of VALUE Types</i>	166
<i>HOW VALUE-TYPE ARRAY OBJECTS ARE ARRANGED IN MEMORY</i>	166
<i>FINDING AN ARRAY'S TYPE, RANK, AND TOTAL NUMBER Of ELEMENTS</i>	167
<i>CREATING SINGLE-DIMENSIONAL ARRAYS USING ARRAY LITERAL VALUES</i>	168
<i>DIFFERENCES BETWEEN ARRAYS Of VALUE Types AND ARRAYS Of REFERENCE Types</i>	169
<i>SINGLE-DIMENSIONAL ARRAYS IN ACTION</i>	171
<i>MESSAGE ARRAY</i>	171

<i>Calculating Averages</i>	173
<i>Histogram: Letter Frequency Counter</i>	173
<i>Quick Review</i>	175
CREATING AND USING MULTIDIMENSIONAL ARRAYS	176
<i>Rectangular Arrays</i>	176
<i>Initializing Rectangular Arrays With Array Literals</i>	178
<i>Ragged Arrays</i>	178
<i>Multidimensional Arrays In Action</i>	179
<i>Weighted Grade Tool</i>	179
<i>Quick Review</i>	181
THE MAIN() METHOD'S STRING ARRAY	181
<i>Purpose And Use Of The Main() Method's String Array</i>	181
MANIPULATING ARRAYS WITH THE SYSTEM.ARRAY CLASS	182
NUMERIC FORMATTING	183
SUMMARY	183
SKILL-BUILDING EXERCISES	184
SUGGESTED PROJECTS	184
SELF-TEST QUESTIONS	187
REFERENCES	188
NOTES	188

9 TOWARD PROBLEM ABSTRACTION: CREATING NEW DATA TYPES

INTRODUCTION	190
ABSTRACTION: AMPLIFY THE ESSENTIAL, ELIMINATE THE IRRELEVANT	190
<i>Abstraction Is The Art Of Programming</i>	190
<i>Where Problem Abstraction Fits Into The Development Cycle</i>	191
<i>Creating Your Own Data Types</i>	191
<i>Case-Study Project: Write A People Manager Program</i>	191
<i>Quick Review</i>	193
THE UML CLASS DIAGRAM	193
<i>Quick Review</i>	194
OVERVIEW OF THE CLASS CONSTRUCT	194
<i>Eleven Categories Of Class Members</i>	194
<i>Fields</i>	195
<i>Constants</i>	197
<i>The Difference Between const and readonly; Compile-Time vs. Runtime Constants</i>	197
<i>Properties</i>	198
<i>Methods</i>	199
<i>Instance Constructors</i>	199
<i>Static Constructors</i>	200
<i>Events</i>	200
<i>Operators</i>	200
<i>Indexers</i>	200
<i>Nested Type Declarations</i>	200
<i>Finalizers</i>	200
<i>Access Modifiers</i>	201
<i>Public</i>	201
<i>Private</i>	201
<i>Protected</i>	201
<i>Internal</i>	201
<i>Protected Internal</i>	201
<i>The Concepts Of Horizontal Access, Interface, And Encapsulation</i>	201
<i>Quick Review</i>	202
METHODS	202
<i>Method Naming: Use Action Words That Indicate The Method's Purpose</i>	203

<i>Maximize Method Cohesion</i>	203
<i>Structure Of A Method Definition</i>	203
<i>Method Modifiers (optional)</i>	203
<i>Return Type Or Void (optional)</i>	204
<i>Method Name (Mandatory)</i>	205
<i>Parameter List (optional)</i>	205
<i>Method Body (optional for abstract or external methods)</i>	205
<i>Method Definition Examples</i>	205
<i>Method Signatures</i>	206
<i>Overloading Methods</i>	206
<i>Constructor Methods</i>	206
<i>Quick Review</i>	206
Building And Testing The Person Class	207
<i>Start By Creating The Source File And Class Definition Shell</i>	207
<i>Defining Person Instance Fields</i>	207
<i>Defining Person Properties And Constructor Method</i>	208
<i>Adding Properties</i>	208
<i>Adding A Constructor Method</i>	208
<i>Testing The Person Class: A Miniature Test Plan</i>	209
<i>Use The PeopleManagerApplication Class As A Test Driver</i>	209
<i>Adding Features To The Person Class: Calculating Age</i>	210
<i>Adding Features To The Person Class: Convenience Properties</i>	211
<i>Adding Features To The Person Class: Finishing Touches</i>	213
<i>Quick Review</i>	214
Building And Testing The PeopleManager Class	215
<i>Defining The PeopleManager Class Shell</i>	215
<i>Defining PeopleManager Fields</i>	215
<i>Defining PeopleManager Constructor Methods</i>	215
<i>Defining Additional PeopleManager Methods</i>	216
<i>Testing The PeopleManager Class</i>	217
<i>Adding Features To The PeopleManager Class</i>	217
<i>Quick Review</i>	219
More About Methods	219
<i>Value Parameters And Reference Parameters</i>	219
<i>Value Parameters: The Default Parameter Passing Mode</i>	219
<i>Reference Parameters: Using The ref Parameter Modifier</i>	220
<i>The out Parameter Modifier</i>	223
<i>Parameter Arrays: Using The params Modifier</i>	223
<i>Local Variable Scoping</i>	224
<i>Anywhere An Object Of <type> Is Required, A Method That Returns <type> Can Be Used</i>	224
<i>Quick Review</i>	225
Structures vs. Classes	225
<i>Value Semantics vs. Reference Semantics</i>	225
<i>Ten Authorized Members vs. Eleven</i>	226
<i>Default Variable Field Values</i>	226
<i>Behavior During Assignment</i>	226
<i>this Behaves Differently</i>	226
<i>Inheritance Not Allowed</i>	226
<i>Boxing And Unboxing</i>	226
<i>When To Use Structures</i>	227
Summary	227
Skill-Building Exercises	228
Suggested Projects	229
Self-Test Questions	231
References	232
Notes	232

10 Compositional Design

INTRODUCTION	234
MANAGING CONCEPTUAL AND PHYSICAL COMPLEXITY	234
<i>Compiling Multiple Source Files Simultaneously With csc</i>	234
<i>Quick Review</i>	235
DEPENDENCY VS. ASSOCIATION	235
AGGREGATION	235
<i>Simple vs. Composite Aggregation</i>	236
<i>The Relationship Between Aggregation And Object Lifetime</i>	236
<i>Quick Review</i>	236
EXPRESSING AGGREGATION IN A UML CLASS DIAGRAM	236
<i>Simple Aggregation Expressed In UML</i>	237
<i>Composite Aggregation Expressed In UML</i>	237
AGGREGATION EXAMPLE CODE	237
<i>Simple Aggregation Example</i>	238
<i>Composite Aggregation Example</i>	239
<i>Quick Review</i>	240
SEQUENCE DIAGRAMS	240
<i>Magic Draw</i>	241
<i>Quick Review</i>	241
THE ENGINE SIMULATION: AN EXTENDED EXAMPLE	242
<i>The Purpose Of The Engine Class</i>	243
<i>Engine Class Attributes And Methods</i>	244
<i>Engine Simulation Sequence Diagrams</i>	244
<i>Running The Engine Simulation Program</i>	244
<i>Quick Review</i>	246
COMPLETE ENGINE SIMULATION CODE LISTING	246
SUMMARY	250
SKILL-BUILDING EXERCISES	250
SUGGESTED PROJECTS	252
SELF-TEST QUESTIONS	252
REFERENCES	253
NOTES	253

11 Inheritance and Interfaces

INTRODUCTION	256
THREE PURPOSES OF INHERITANCE	256
<i>Implementing The “is A” Relationship</i>	257
<i>The Relationship Between The Terms Type, Interface, and Class</i>	257
<i>Meaning Of The Term Interface</i>	257
<i>Meaning Of The Term Class</i>	257
<i>Quick Review</i>	258
EXPRESSING GENERALIZATION AND SPECIALIZATION IN THE UML	258
A SIMPLE INHERITANCE EXAMPLE	259
<i>The UML Diagram</i>	259
<i>BaseClass Source Code</i>	259
<i>DerivedClass Source Code</i>	260
<i>DriverApplication Program</i>	260
<i>Quick Review</i>	261
ANOTHER INHERITANCE EXAMPLE: PERSON - STUDENT	261
<i>The Person - Student UML Class Diagram</i>	261
<i>Person - Student Source Code</i>	262
<i>CASTING</i>	264
<i>Use Casting Sparingly</i>	265

<i>Quick Review</i>	265
OVERRIDING BASE CLASS METHODS	266
<i>Quick Review</i>	267
ABSTRACT METHODS AND ABSTRACT BASE CLASSES	267
<i>The Primary Purpose Of An Abstract Base Class</i>	268
<i>Expressing Abstract Base Classes In UML</i>	268
<i>Quick Review</i>	270
INTERFACES	270
<i>The Purpose Of Interfaces</i>	270
<i>Authorized Interface Members</i>	270
<i>The Differences Between An Interface And An Abstract Class</i>	271
<i>Expressing Interfaces In UML</i>	271
<i>Expressing Realization In A UML Class Diagram</i>	271
<i>An Interface Example</i>	272
<i>Quick Review</i>	274
CONTROLLING HORIZONTAL AND VERTICAL ACCESS	274
<i>Quick Review</i>	274
SEALED CLASSES AND METHODS	274
<i>Quick Review</i>	274
POLYMORPHIC BEHAVIOR	275
<i>Quick Review</i>	275
INHERITANCE EXAMPLE: EMPLOYEE	275
INHERITANCE EXAMPLE: ENGINE SIMULATION	278
<i>Engine Simulation UML Diagram</i>	278
<i>Simulation Operational Description</i>	278
<i>Compiling The Engine Simulation Code</i>	280
COMPLETE ENGINE SIMULATION CODE LISTING	280
SUMMARY	284
SKILL-BUILDING EXERCISES	285
SUGGESTED PROJECTS	285
SELF-TEST QUESTIONS	287
REFERENCES	287
NOTES	288

12 Windows Forms Programming

INTRODUCTION	292
THE FORM CLASS	292
<i>Form Class Inheritance Hierarchy</i>	292
<i>A Simple Form Program</i>	293
<i>Quick Review</i>	294
APPLICATION MESSAGES, MESSAGE PUMP, EVENTS, AND EVENT LOOP	294
<i>Message Categories</i>	295
<i>Messages In Action: Trapping Messages With IMessageFilter</i>	296
<i>Final Thoughts On Messages</i>	296
<i>Quick Review</i>	297
SCREEN AND WINDOW (CLIENT) COORDINATE SYSTEM	297
<i>Quick Review</i>	299
MANIPULATING FORM PROPERTIES	299
<i>Quick Review</i>	301
ADDING COMPONENTS TO WINDOWS: BUTTON, TEXTBOX, AND LABEL	301
<i>Quick Review</i>	302
REGISTERING EVENT HANDLERS WITH GUI COMPONENTS	303
<i>Delegates And Events</i>	303
<i>Quick Review</i>	305
HANDLING GUI COMPONENT EVENTS IN SEPARATE OBJECTS	305

<i>Quick Review</i>	307
LAYOUT MANAGERS	307
<i>FlowLayoutPanel</i>	308
<i>TableLayoutPanel</i>	310
<i>Quick Review</i>	311
MENUS	312
<i>Quick Review</i>	315
A LITTLE MORE ABOUT TEXTBOXES	315
<i>Quick Review</i>	316
THE RHYTHM OF CODING GUIs	317
SUMMARY	317
SKILL-BUILDING EXERCISES	318
SUGGESTED PROJECTS	319
SELF-TEST QUESTIONS	319
REFERENCES	320
NOTES	320

13 CUSTOM EVENTS

INTRODUCTION	322
C# EVENT PROCESSING MODEL: AN OVERVIEW	322
<i>Quick Review</i>	323
CUSTOM EVENTS EXAMPLE: MINUTE TICK	323
CUSTOM EVENTS EXAMPLE: AUTOMATED WATER TANK SYSTEM	326
NAMING CONVENTIONS	331
FINAL THOUGHTS ON EXTENDING THE EVENTARGS CLASS	332
SUMMARY	332
SKILL-BUILDING EXERCISES	333
SUGGESTED PROJECTS	333
SELF-TEST QUESTIONS	333
REFERENCES	334
NOTES	334

14 COLLECTIONS

INTRODUCTION	338
CASE STUDY: BUILDING A DYNAMIC ARRAY	338
<i>Evaluating DynamicArray</i>	340
<i>The ArrayList Class To The Rescue</i>	340
<i>A Quick Peek At Generics</i>	341
<i>Quick Review</i>	341
DATA STRUCTURE PERFORMANCE CHARACTERISTICS	342
<i>Array Performance Characteristics</i>	342
<i>Linked List Performance Characteristics</i>	343
<i>Hash Table Performance Characteristics</i>	345
<i>Chained Hash Table vs. Open-Address Hash Table</i>	345
<i>Red-Black Tree Performance Characteristics</i>	346
<i>Stacks And Queues</i>	347
<i>Quick Review</i>	347
NAVIGATING THE .NET COLLECTIONS API	348
<i>System.Collections</i>	348
<i>System.Collections.Generic</i>	348
<i>System.Collections.ObjectModel</i>	349
<i>System.Collections.Specialized</i>	349

<i>Mapping Non-Generic To Generic Collections</i>	349
<i>Quick Review</i>	350
Using Non-Generic Collection Classes - Pre .NET 2.0	350
<i>Objects In – Objects Out: Casting 101</i>	351
<i>Extending ArrayList To Create A Strongly-Typed Collection</i>	352
Using Generic Collection Classes – .NET 2.0 and Beyond	354
<i>List<T>: Look Ma, No More Casting!</i>	354
<i>Implementing KeyedCollection<TKey, TItem></i>	355
<i>Quick Review</i>	356
Special Operations On Collections	357
<i>Sorting A List</i>	357
<i>Implementing System.IComparable<T></i>	357
<i>Extending Comparer<T></i>	359
<i>Converting A Collection Into An Array</i>	361
<i>Quick Review</i>	361
SUMMARY	362
Skill-Building Exercises	362
SUGGESTED PROJECTS	363
SELF-TEST QUESTIONS	363
REFERENCES	364
NOTES	364

15 EXCEPTIONS: WRITING FAULT-TOLERANT SOFTWARE

INTRODUCTION	366
WHAT IS AN EXCEPTION	366
.NET CLR EXCEPTION HANDLING MECHANISM	366
<i>Unhandled Exceptions</i>	366
<i>The Exception Information Table</i>	367
<i>Quick Review</i>	367
EXCEPTION CLASS HIERARCHY	367
<i>Application vs. Runtime Exceptions</i>	368
<i>Runtime Exception Listing</i>	368
<i>Determining What Exceptions A .NET Framework Method Throws</i>	369
<i>Quick Review</i>	369
EXCEPTION CLASS PROPERTIES	370
<i>Quick Review</i>	370
CREATING EXCEPTION HANDLERS: USING TRY/CATCH/FINALLY BLOCKS	371
<i>Using A Try/Catch Block</i>	371
<i>First Line of Defense: Use Defensive Coding</i>	372
<i>Using Multiple Catch Blocks</i>	372
<i>Using A Finally Block</i>	373
<i>Quick Review</i>	374
CREATING CUSTOM EXCEPTIONS	374
<i>Extending The Exception Class</i>	374
<i>Manually Throwing An Exception With The Throw Keyword</i>	375
<i>Translating Low-Level Exceptions Into High-Level Exceptions</i>	375
<i>Quick Review</i>	376
DOCUMENTING EXCEPTIONS	376
SUMMARY	377
Skill-Building Exercises	377
SUGGESTED PROJECTS	378
SELF-TEST QUESTIONS	378
REFERENCES	378
NOTES	379

16 MULTITHREADED PROGRAMMING

INTRODUCTION	382
MULTITHREADING OVERVIEW: THE TALE OF TWO VACATIONS	382
<i>Single-Threaded Vacation</i>	382
<i>MultiThreaded Vacation</i>	382
<i>The Relationship Between A Process And Its Threads</i>	383
<i>Vacation Gone Bad</i>	384
<i>Quick Review</i>	385
CREATING MANAGED THREADS WITH THE THREAD CLASS	385
<i>Single-Threaded Vacation Example</i>	386
<i>MultiThreaded Vacation Example</i>	386
<i>Thread States</i>	389
<i>Creating And Starting Managed Threads</i>	389
<i>ThreadStart Delegate</i>	389
<i>ParameterizedThreadStart Delegate: Passing Arguments To Threads</i>	390
<i>Blocking A Thread With Thread.Sleep()</i>	391
<i>Blocking A Thread With Thread.Join()</i>	392
<i>Foreground vs. Background Threads</i>	394
<i>Quick Review</i>	395
CREATING THREADS WITH THE BACKGROUNDWORKER CLASS	396
<i>Quick Review</i>	399
THREAD POOLS	399
<i>Quick Review</i>	400
ASYNCHRONOUS METHOD CALLS	400
<i>Obtaining Results From An Asynchronous Method Call</i>	402
<i>Providing A Callback Method To BeginInvoke()</i>	402
<i>Quick Review</i>	403
SUMMARY	404
SKILL-BUILDING EXERCISES	405
SUGGESTED PROJECTS	405
SELF-TEST QUESTIONS	406
REFERENCES	406
NOTES	407

17 FILE I/O

INTRODUCTION	410
MANIPULATING DIRECTORIES AND FILES	410
<i>Files, Directories, And Paths</i>	411
<i>Manipulating Directories And Files</i>	411
<i>Verbatim String Literals</i>	412
<i>Quick Review</i>	413
SERIALIZING OBJECTS TO DISK	413
<i>Serializable Attribute</i>	413
<i>Serializing Objects With BinaryFormatter</i>	414
<i>Serializing Objects With XmlSerializer</i>	416
<i>Quick Review</i>	418
WORKING WITH TEXT FILES	418
<i>Some Issues You Must Consider</i>	418
<i>Saving Dog Data To A Text File</i>	418
<i>Quick Review</i>	420
WORKING WITH BINARY DATA	420
<i>Quick Review</i>	421
RANDOM ACCESS FILE I/O	422
<i>Towards An Approach To The Adapter Project</i>	422

<i>START SMALL AND TAKE BABY STEPS</i>	423
<i>OTHER PROJECT CONSIDERATIONS</i>	424
<i>LOCKING A RECORD FOR UPDATES AND DELETES</i>	424
<i>MONITOR.ENTER()/MONITOR.EXIT() VS. THE LOCK KEYWORD</i>	424
<i>TRANSLATING LOW-LEVEL EXCEPTIONS INTO HIGHER-LEVEL EXCEPTION ABSTRACTIONS</i>	425
<i>WHERE TO GO FROM HERE</i>	425
<i>COMPLETE RANDOMACCESSFILE LEGACY DATAFILE ADAPTER SOURCE CODE LISTING</i>	425
<i>Quick Review</i>	437
Working With Log Files	438
<i>Quick Review</i>	440
Using FileDialogs	440
<i>Quick Review</i>	442
SUMMARY	443
Skill-Building Exercises	444
SUGGESTED PROJECTS	444
SELF-TEST QUESTIONS	444
REFERENCES	445
NOTES	445

18 NETWORK PROGRAMMING FUNDAMENTALS

INTRODUCTION	450
WHAT IS A COMPUTER NETWORK?	450
<i>PURPOSE OF A NETWORK</i>	450
<i>THE ROLE OF NETWORK PROTOCOLS</i>	451
<i>HOMOGENEOUS VS. HETEROGENEOUS NETWORKS</i>	451
<i>THE UNIFYING NETWORK PROTOCOLS: TCP/IP</i>	451
<i>WHAT'S SO SPECIAL ABOUT THE INTERNET?</i>	452
<i>Quick Review</i>	452
SERVERS & CLIENTS	453
<i>SERVER HARDWARE AND SOFTWARE</i>	453
<i>CLIENT HARDWARE AND SOFTWARE</i>	453
<i>Quick Review</i>	454
Application Distribution	454
<i>Physical Distribution ON ONE COMPUTER</i>	454
<i>RUNNING MULTIPLE CLIENTS ON THE SAME COMPUTER</i>	454
<i>ADDRESSING THE LOCAL MACHINE</i>	455
<i>Physical Distribution ACROSS MULTIPLE COMPUTERS</i>	455
<i>Quick Review</i>	455
MULTITIERED APPLICATIONS	456
<i>Logical Application TIERS</i>	456
<i>Physical Tier Distribution</i>	456
<i>Quick Review</i>	456
INTERNET NETWORKING PROTOCOLS: NUTS & BOLTS	457
<i>THE INTERNET PROTOCOLS: TCP, UDP, AND IP</i>	457
<i>THE APPLICATION LAYER</i>	458
<i>TRANSPORT LAYER</i>	458
<i>NETWORK LAYER</i>	459
<i>DATA LINK AND PHYSICAL LAYERS</i>	459
<i>PUTTING IT ALL TOGETHER</i>	459
<i>WHAT YOU NEED TO KNOW</i>	460
<i>Quick Review</i>	460
SUMMARY	460
Skill-Building Exercises	461
SUGGESTED PROJECTS	462
SELF-TEST QUESTIONS	462

REFERENCES	462
NOTES	463

19 NETWORKED CLIENT -SERVER Applications

INTRODUCTION	466
Building CLIENT-SERVER Applications With .NET REMOTING	466
<i>THE THREE REQUIRED COMPONENTS OF A .NET REMOTING APPLICATION</i>	466
<i>A SIMPLE .NET REMOTING APPLICATION</i>	467
<i>SINGLECALL VS. SINGLETON</i>	469
<i>ACCESSING A REMOTE OBJECT VIA AN INTERFACE</i>	470
<i>USING CONFIGURATION FILES</i>	472
<i>PASSING OBJECTS BETWEEN CLIENT AND SERVER</i>	474
<i>QUICK REVIEW</i>	477
CLIENT-SERVER Applications With TcpListener And TcpClient	478
<i>TCP/IP CLIENT-SERVER OVERVIEW</i>	478
<i>A SIMPLE CLIENT-SERVER APPLICATION</i>	479
<i>BUILDING A MULTITHREADED SERVER</i>	480
<i>LISTENING ON MULTIPLE IP ADDRESSES</i>	482
<i>SENDING OBJECTS BETWEEN CLIENT AND SERVER</i>	484
<i>QUICK REVIEW</i>	488
SUMMARY	489
SKILL-BUILDING EXERCISES	490
SUGGESTED PROJECTS	491
SELF-TEST QUESTIONS	491
REFERENCES	492
NOTES	492

20 DATABASE ACCESS & MULTITIERED Applications

INTRODUCTION	494
WHAT YOU ARE GOING TO BUILD	494
PRELIMINARIES	495
<i>INSTALLING SQL SERVER EXPRESS EDITION</i>	495
<i>INSTALLING MICROSOFT SQL SERVER MANAGEMENT STUDIO EXPRESS</i>	496
<i>INSTALLING MICROSOFT ENTERPRISE LIBRARY</i>	498
<i>A SIMPLE TEST APPLICATION</i>	499
INTRODUCTION TO RELATIONAL DATABASES AND SQL	501
<i>TERMINOLOGY</i>	501
<i>STRUCTURED QUERY LANGUAGE (SQL)</i>	502
<i>DATA DEFINITION LANGUAGE (DDL)</i>	502
<i>CREATING THE EMPLOYEETRAINING DATABASE</i>	502
<i>CREATING A DATABASE WITH A SCRIPT</i>	503
<i>CREATING TABLES</i>	504
<i>SQL SERVER DATABASE TYPES</i>	505
<i>DATA MANIPULATION LANGUAGE (DML)</i>	506
<i>USING THE INSERT COMMAND</i>	506
<i>USING THE SELECT COMMAND</i>	507
<i>USING THE UPDATE COMMAND</i>	509
<i>USING THE DELETE COMMAND</i>	510
<i>QUICK REVIEW</i>	510
Complex SQL QUERIES	511
<i>CREATING A RELATED TABLE WITH A FOREIGN KEY</i>	511
<i>INSERTING TEST DATA INTO THE tbl_EMPLOYEE_TRAINING TABLE</i>	512
<i>SELECTING DATA FROM MULTIPLE TABLES</i>	514

Join OPERATIONS	514
TESTING THE CASCADE DELETE CONSTRAINT	515
Quick Review	515
The SERVER Application	516
PROJECT FOLDER ORGANIZATION	516
Using Microsoft Build To MANAGE And Build the PROJECT	517
FIRST ITERATION	519
Coding THE EMPLOYEEVO And EMPLOYEEDAO	520
Application CONFIGURATION FILE.....	528
CREATING TEST Application.....	528
SECOND ITERATION	532
TESTING THE CODE - SECOND ITERATION.....	543
Reality CHECK.....	551
Third ITERATION	551
The CLIENT Application	556
Third ITERATION (CONTINUED)	556
FOURTH ITERATION	558
Fifth ITERATION	564
Sixth ITERATION	569
Compiling And RUNNING THE Modified EMPLOYEETRAININGCLIENT PROJECT	580
WHERE TO GO FROM HERE	582
SUMMARY	583
SKILL-BUILDING EXERCISES	583
SUGGESTED PROJECTS	584
SELF-TEST QUESTIONS	584
REFERENCES	585
NOTES	585

21 OPERATOR OVERLOADING

INTRODUCTION	590
OPERATOR OVERLOADING	590
Overloadable OPERATORS	590
Quick Review	591
OVERLOADING UNARY OPERATORS	591
+, - OPERATORS	591
! OPERATOR	592
TRUE, FALSE OPERATORS	593
++ -, OPERATORS	595
Quick Review	597
OVERLOADING BINARY OPERATORS	597
+, - OPERATORS	597
*, / OPERATORS	599
&, OPERATORS	601
Quick Review	603
OVERLOADING COMPARISON OPERATORS	603
==, !=, <, >, <=, >= OPERATORS	604
Quick Review	607
CREATING IMPLICIT AND EXPLICIT CAST OPERATORS	607
Implicit vs. Explicit CAST	607
Overloaded CAST OPERATORS EXAMPLE	607
Quick Review	610
THE ASSIGNMENT OPERATORS: THINGS YOU GET FOR FREE	610
Quick Review	610
SUMMARY	610
SKILL-BUILDING EXERCISES	611

SUGGESTED PROJECTS	611
SELF-TEST QUESTIONS	611
REFERENCES	612
NOTES	612

22 Well-Behaved Objects

INTRODUCTION	614
OBJECT BEHAVIOR DEFINED	614
<i>FUNDAMENTAL BEHAVIOR</i>	614
<i>COPY/ASSIGNMENT BEHAVIOR</i>	614
<i>EQUALITY BEHAVIOR</i>	615
<i>COMPARISON/ORDERING BEHAVIOR</i>	615
<i>SEVEN OBJECT USAGE SCENARIOS</i>	615
FUNDAMENTAL BEHAVIOR	616
<i>OBJECT CREATION – CONSTRUCTORS</i>	616
<i>DEFAULT CONSTRUCTOR</i>	616
<i>PRIVATE CONSTRUCTORS</i>	616
<i>OVERLOADED CONSTRUCTORS</i>	616
<i>MEMBER ACCESSIBILITY</i>	616
<i>HORIZONTAL MEMBER ACCESS</i>	616
<i>VERTICAL MEMBER ACCESS</i>	617
<i>OVERRIDING OBJECT.TOSTRING()</i>	617
<i>STATIC VS. INSTANCE MEMBERS</i>	617
<i>SERIALIZATION</i>	618
<i>CUSTOM SERIALIZATION EXAMPLE</i>	618
<i>Quick Review</i>	623
COPY/ASSIGNMENT BEHAVIOR	623
<i>VALUE OBJECT VS. REFERENCE OBJECT ASSIGNMENT</i>	624
<i>RULE OF THUMB – FAVOR THE CLASS CONSTRUCT FOR COMPLEX TYPES</i>	624
<i>SHALLOW COPY VS. DEEP COPY</i>	624
<i>COPY CONSTRUCTORS</i>	624
<i>SYSTEM.ICLONEABLE VS. OBJECT.MEMBERWISECLONE()</i>	627
<i>Quick Review</i>	629
EQUALITY BEHAVIOR	629
<i>REFERENCE EQUALITY VS. VALUE EQUALITY</i>	629
<i>RULES FOR OVERRIDING THE OBJECT.EQUALS() METHOD</i>	629
<i>OVERRIDING THE OBJECT.GETHASHCODE() METHOD</i>	630
<i>BLOCH'S HASH CODE GENERATION ALGORITHM</i>	631
<i>ASHMORE'S HASH CODE GENERATION ALGORITHM</i>	631
<i>OVERRIDING OBJECT.EQUALS() AND OBJECT.GETHASHCODE() METHODS IN THE PERSONVO CLASS</i>	631
<i>Quick Review</i>	633
COMPARISON/ORDERING BEHAVIOR	633
<i>IMPLEMENTING SYSTEM.ICOMPARABLE<T></i>	634
<i>RULES FOR IMPLEMENTING THE COMPARETO(T OTHER) METHOD</i>	635
<i>EXTENDING THE COMPARER<T> CLASS</i>	636
<i>Quick Review</i>	637
SUMMARY	637
SKILL-BUILDING EXERCISES	638
SUGGESTED PROJECTS	638
SELF-TEST QUESTIONS	638
REFERENCES	639
NOTES	639

23 THREE DESIGN PRINCIPLES

INTRODUCTION	642
THE PREFERRED CHARACTERISTICS OF AN OBJECT-ORIENTED ARCHITECTURE	642
<i>EASY TO UNDERSTAND: HOW DOES THIS THING WORK?</i>	642
<i>EASY TO REASON ABOUT: WHAT ARE THE EFFECTS OF CHANGE?</i>	642
<i>EASY TO EXTEND: WHERE DO I ADD FUNCTIONALITY?</i>	642
THE LISKOV SUBSTITUTION PRINCIPLE & DESIGN BY CONTRACT	643
<i>REASONING ABOUT THE BEHAVIOR OF SUPERTYPES AND SUBTYPES</i>	643
<i>RELATIONSHIP BETWEEN THE LSP AND DbC</i>	643
<i>THE COMMON GOAL OF THE LSP AND DbC</i>	643
<i>C# SUPPORT FOR THE LSP AND DbC</i>	643
<i>DESIGNING WITH THE LSP/DbC IN MIND</i>	644
<i>CLASS DECLARATIONS VIEWED AS BEHAVIOR SPECIFICATIONS</i>	644
<i>QUICK REVIEW</i>	644
PRECONDITIONS, POSTCONDITIONS, AND CLASS INVARIANTS	644
<i>CLASS INVARIANT</i>	644
<i>PRECONDITION</i>	644
<i>POSTCONDITION</i>	645
<i>AN EXAMPLE</i>	645
<i>A NOTE ON USING THE DEBUG.ASSERT() METHOD TO ENFORCE PRE- AND POSTCONDITIONS</i>	646
<i>USING INCREMENTER AS A BASE CLASS</i>	646
<i>CHANGING THE PRECONDITIONS OF DERIVED CLASS METHODS</i>	648
<i>CHANGING THE POSTCONDITIONS OF DERIVED CLASS METHODS</i>	652
<i>SPECIAL CASES OF PRECONDITIONS AND POSTCONDITIONS</i>	652
<i>METHOD ARGUMENT TYPES</i>	653
<i>METHOD RETURN TYPES</i>	654
<i>THREE RULES OF THE SUBSTITUTION PRINCIPLE</i>	654
<i>SIGNATURE RULE</i>	655
<i>METHODS RULE</i>	655
<i>PROPERTIES RULE</i>	655
<i>QUICK REVIEW</i>	655
THE OPEN-CLOSED PRINCIPLE	656
<i>ACHIEVING THE OPEN-CLOSED PRINCIPLE</i>	656
<i>AN OCP EXAMPLE</i>	656
<i>QUICK REVIEW</i>	661
THE DEPENDENCY INVERSION PRINCIPLE	661
<i>CHARACTERISTICS OF BAD SOFTWARE ARCHITECTURE</i>	661
<i>CHARACTERISTICS OF GOOD SOFTWARE ARCHITECTURE</i>	662
<i>SELECTING THE RIGHT ABSTRACTIONS TAKES EXPERIENCE</i>	662
<i>QUICK REVIEW</i>	662
TERMS AND DEFINITIONS	663
SUMMARY	663
SKILL-BUILDING EXERCISES	664
SUGGESTED PROJECTS	664
SELF-TEST QUESTIONS	664
REFERENCES	665
NOTES	666

24 INHERITANCE, COMPOSITION, INTERFACES, POLYMORPHISM

INTRODUCTION	668
INHERITANCE VS. COMPOSITION: THE GREAT DEBATE	668
<i>WHAT'S THE END GAME?</i>	669
<i>FLEXIBLE APPLICATION ARCHITECTURES</i>	669
<i>MODULARITY AND RELIABILITY</i>	669

<i>Architectural Stability Via Managed Dependencies</i>	669
<i>Knowing When To Accept A Design That's Good Enough</i>	670
<i>Quick Review</i>	670
INHERITANCE-BASED DESIGN	670
<i>THREE GOOD REASONS TO USE INHERITANCE</i>	670
<i>As A Means To Reason About Code Behavior</i>	670
<i>To Gain A Measure Of Code Reuse</i>	670
<i>To Facilitate Incremental Development</i>	670
<i>FORMS OF INHERITANCE: MEYER'S INHERITANCE TAXONOMY</i>	671
<i>COAD'S INHERITANCE CRITERIA</i>	672
<i>PERSON - EMPLOYEE EXAMPLE REVISITED</i>	673
<i>Quick Review</i>	673
THE ROLE OF INTERFACES	674
<i>Reducing Or Limiting Intermodule Dependencies</i>	674
<i>Modeling Dominant, Collateral, And Dynamic Roles</i>	674
<i>Dominant Roles</i>	674
<i>Collateral Roles</i>	675
<i>Dynamic Roles</i>	675
<i>Quick Review</i>	675
Applied Polymorphism	675
<i>Quick Review</i>	676
Composition-Based Design As A Force Multiplier	676
<i>Two Types Of Aggregation</i>	676
<i>Polymorphic Containment</i>	676
AN EXTENDED EXAMPLE	677
<i>Quick Review</i>	682
SUMMARY	682
Skill-Building Exercises	683
SUGGESTED PROJECTS	684
SELF-TEST QUESTIONS	685
REFERENCES	685
NOTES	686

25 Helpful Design Patterns

INTRODUCTION	688
SOFTWARE DESIGN PATTERNS AND HOW THEY CAME TO BE	688
<i>What Exactly Is A Software Design Pattern?</i>	688
<i>Origins</i>	688
<i>PATTERN SPECIFICATION</i>	689
<i>Applying Software Design Patterns</i>	689
<i>Quick Review</i>	689
THE SINGLETON PATTERN	690
<i>Quick Review</i>	693
THE FACTORY PATTERN	693
<i>The Dynamic Factory</i>	693
<i>Advantages Of The Dynamic Factory Pattern</i>	695
<i>Quick Review</i>	695
THE MODEL-VIEW-CONTROLLER PATTERN	695
<i>Quick Review</i>	697
THE COMMAND PATTERN	697
<i>Quick Review</i>	702
A COMPREHENSIVE PATTERN-BASED EXAMPLE	702
<i>Complete Code Listing</i>	702
<i>Com.PulpFreePress.Exceptions</i>	702
<i>Com.PulpFreePress.Common</i>	702

<i>Com.PulpFreePress.Utils</i>	707
<i>Com.PulpFreePress.Commands</i>	710
<i>Com.PulpFreePress.Model</i>	713
<i>Com.PulpFreePress.View</i>	714
<i>Com.PulpFreePress.Controller</i>	720
<i>Running The Application</i>	721
SUMMARY	722
Skill-Building Exercises	722
SUGGESTED PROJECTS	723
SELF-TEST QUESTIONS	723
REFERENCES	723
NOTES	724

Appendix A: Helpful Checklists And Tables

Project-Approach Strategy Check-off List	727
Development Cycle	728
Final Project Review Checklist	728

Appendix B: ASCII Table

ASCII Table	731
--------------------------	------------

Appendix C: Identifier Naming: Writing Self-Commenting Code

Identifier Naming: Writing Self-Commenting Code	735
<i>Benefits of Self-Commenting Code</i>	735
<i>Coding Convention</i>	735
<i>Class Names</i>	735
<i>Constant Names</i>	736
<i>Variable Names</i>	736
<i>Method Names</i>	736
<i>Property Names</i>	737