

Contents

| | |
|--|-------------|
| Introduction | xvii |
| Chapter 1: Refactoring: What's All the Fuss About? | 1 |
| A Quick Refactoring Overview | 1 |
| The Refactoring Process | 2 |
| A Look at the Software Situation | 3 |
| The Refactoring Process: A Closer Look | 5 |
| Using Code Smells | 6 |
| Transforming the Code | 6 |
| Automating Refactoring Transformations | 6 |
| The Benefits of Refactoring | 8 |
| Debunking Common Misconceptions | 10 |
| No Programmer Is an Island | 19 |
| Refactoring for Business People | 20 |
| C# and Refactoring | 21 |
| Summary | 21 |
| Chapter 2: A First Taste of Refactoring | 23 |
| Sample Application: Calories Calculator | 23 |
| Calories Calculator Application with Recommended Daily Calories Amount Calculation | 24 |
| Growing Requirements: Calculating Ideal Weight | 27 |
| Growing Requirements: Persisting Patient Data | 30 |
| Refactoring in Action | 31 |
| Decomposing the btnCalculate_Click Method | 32 |
| The Calculate and Display Distance from Ideal Weight Segment | 33 |
| Calculating Calories and Ideal Weight by Gender | 33 |
| The btnCalculate_Click Method after Method Extraction | 34 |
| Discovering New Classes | 35 |
| Narrowing the Patient Class Interface | 38 |
| Restructuring the DistanceFromIdealWeight Method | 41 |
| Creating the Patient Class Hierarchy | 43 |
| Implementing the Persistence Functionality | 48 |
| Saving the Data | 49 |
| Implementing Patient-History Display Functionality | 57 |

Contents

| | |
|--|------------|
| Calories Calculator, Refactored Version | 61 |
| Summary | 63 |
| Chapter 3: Assembling a Refactoring Toolkit | 65 |
| Using an Automated Refactoring Tool | 66 |
| ReSharper from JetBrains | 66 |
| Refactor! Pro from Developer Express | 67 |
| Refactor! for ASP from Developer Express | 68 |
| Visual Studio Refactoring Features | 68 |
| Unit-Testing Basics: The Testing Harness | 70 |
| Why a Unit Testing Framework? | 71 |
| Your First Taste of NUnit | 73 |
| Installing NUnit | 74 |
| Working with the Samples | 74 |
| Implementing Your First Test | 76 |
| The Test-Driven Approach | 84 |
| Other Test Tools to Consider | 85 |
| About Version Control | 87 |
| Version Control As a Backup System | 87 |
| Version Control and Concurrency | 87 |
| Summary | 88 |
| Chapter 4: Application Prototype: Rent-a-Wheels | 89 |
| Interviewing the Client | 90 |
| Interviewing the Manager | 90 |
| Interviewing the Desk Receptionist | 91 |
| Interviewing the Parking Lot Attendant | 91 |
| Interviewing Maintenance Personnel | 92 |
| Taking the Initial Steps in the Rent-a-Wheels Project | 93 |
| Actors and Use Cases | 93 |
| Vehicle States | 95 |
| First Sketch of the Main Application Window | 97 |
| Rent-a-Wheels Team Meeting | 97 |
| Making the Prototype Work | 98 |
| Examining the Database Model | 98 |
| Examining the C# Code | 100 |
| Fast and Furious Approach to Programming | 104 |
| Database-Driven Design | 104 |
| GUI-Based Application | 104 |
| Event-Driven Programming | 105 |

Contents

| | |
|--|------------|
| Rapid Application Development (RAD) | 105 |
| Copy and Paste As a Code Reuse Mechanism | 106 |
| From Prototype to Delivery Through the Refactoring Process | 106 |
| Summary | 106 |
| Chapter 5: Basic Hygiene | 109 |
| Eliminating Dead Code | 109 |
| Types of Dead Code | 111 |
| Common Sources of Dead Code | 112 |
| Reducing the Scope and Access Level of Unduly Exposed Elements | 116 |
| Scope and Access Level | 118 |
| Common Sources of Overexposure | 119 |
| Dealing with Overexposure | 122 |
| Using Explicit Imports | 123 |
| Using Section Depicts Dependencies in Your System | 124 |
| Removing Unused Assembly References | 126 |
| Basic Hygiene in the Rent-a-Wheels Application | 127 |
| Summary | 128 |
| Chapter 6: From Problem Domain to Code: Closing the Gap | 129 |
| Understanding the Problem Domain | 130 |
| Step One: Gathering the Information | 131 |
| Step Two: Agreeing on the Vocabulary | 131 |
| Step Three: Describing the Interactions | 132 |
| Step Four: Building the Prototype | 133 |
| Naming Guidelines | 133 |
| Capitalization Styles | 134 |
| Simple Naming Guidelines | 135 |
| Good Communication: Choosing the Right Words | 136 |
| Rename Refactoring | 140 |
| Rename Refactoring in Visual Studio | 142 |
| Published and Public Interfaces | 144 |
| Self-Contained Applications versus Reusable Modules | 144 |
| Modifying the Published Interfaces | 148 |
| Rename and Safe Rename Refactoring in the Rent-a-Wheels Application | 153 |
| Summary | 154 |
| Chapter 7: The Method Extraction Remedy for Duplicated Code | 155 |
| Encapsulating Code and Hiding the Details | 155 |
| Information and Implementation Hiding | 156 |

Contents

| | |
|--|------------|
| Decomposing Methods | 159 |
| Circumference Calculation — Long Method Example | 159 |
| Extracting Circumference Length Calculation Code | 162 |
| Extracting the Radius Calculation Code | 165 |
| Extracting the “Wait for User to Close” Code | 166 |
| Extracting the Read Coordinates Code | 166 |
| Extract Method Refactoring in Visual Studio | 169 |
| Inlining Methods | 171 |
| The Duplicated Code Smell | 175 |
| Sources of Duplicated Code | 176 |
| Copy-Paste Programming | 177 |
| Magic Literals | 178 |
| Extract Method and Replace Magic Literal Refactoring in the Rent-a-Wheels Application | 179 |
| Summary | 180 |
| Chapter 8: Method Consolidation and Extraction Techniques | 183 |
| Dealing with Temporary Variables | 183 |
| Move Declaration Near Reference Refactoring | 184 |
| Move Initialization to Declaration Refactoring | 187 |
| Split Temporary Variable Refactoring | 188 |
| Inline Temp Refactoring | 192 |
| Replace Temp with Query Refactoring | 194 |
| Introducing Explaining Temporary Variables | 197 |
| Dealing with Long and Nested Conditionals | 198 |
| Method Reorganization and Rent-a-Wheels | 201 |
| Removing Duplication in Rent-a-Wheels | 203 |
| Summary | 211 |
| Chapter 9: Discovering Objects | 213 |
| A Brief Object-Oriented Programming Overview | 214 |
| Objects in OOP | 214 |
| Encapsulation and Objects | 214 |
| Encapsulate Field Refactoring in Visual Studio | 216 |
| Object State Retention | 218 |
| Classes | 218 |
| Object Identity | 219 |
| Objects As Basic Building Blocks | 220 |
| Root Object | 221 |
| Object Lifetime and Garbage Collection | 221 |

Contents

| | |
|---|------------|
| Designing Classes | 223 |
| Using Analysis Artifacts | 223 |
| Classes Are Nouns, Operations Are Verbs | 226 |
| Classes, Responsibilities, and Collaborators | 231 |
| Employing Cards in Brainstorming Sessions | 235 |
| Entities and Relationships | 239 |
| Discovering Hidden Classes | 240 |
| Dealing with Database-Driven Design | 241 |
| Moving from Procedural to Object-Oriented Design | 244 |
| Separating Domain, Presentation, and Persistence | 252 |
| Discovering Objects and the Rent-a-Wheels Application | 257 |
| Summary | 265 |
| Chapter 10: Advanced Object-Oriented Concepts and Related Refactorings | 267 |
| Inheritance, Polymorphism, and Genericity | 268 |
| Inheritance | 268 |
| Class versus Interface Inheritance | 271 |
| Polymorphism | 273 |
| Genericity | 276 |
| Inheritance Abuse and Refactoring Solutions | 277 |
| Composition Mistaken for Inheritance and Other Misuses | 281 |
| Refactoring for Inheritance — Print-System Example | 288 |
| Delegation Instead of Inheritance Inside the Print System | 294 |
| Making Use of Generics | 308 |
| Inheritance and Generic Types in the Rent-a-Wheels Application | 312 |
| Extracting Super | 312 |
| Employing Generics | 313 |
| Extracting the DataObjectsProvider Class | 314 |
| Summary | 318 |
| Chapter 11: Code Organization on a Large Scale | 319 |
| Namespaces | 319 |
| Naming Guidelines and Namespace Organization | 320 |
| Nested Namespaces | 320 |
| Changing the Default Namespace Name | 321 |
| Employing Using Directives | 321 |
| Assemblies | 323 |
| Binary Reuse | 323 |
| Namespace Organization Guidelines | 325 |
| Dependency Considerations | 330 |

Contents

| | |
|--|------------|
| C# Project File Structure Organization | 338 |
| Partial Classes | 339 |
| Namespace Organization and Windows Forms Inheritance in Rent-a-Wheels | 345 |
| Extracting the Parent Administration Form Using the Abstract Form Helper Pattern | 345 |
| Namespace and Assembly Reorganization | 353 |
| Summary | 355 |
| Chapter 12: Refactoring to Patterns | 357 |
| Design Patterns: What's All the Fuss About? | 358 |
| Defining Design Patterns | 358 |
| Classifying Patterns | 359 |
| Pattern Elements | 359 |
| Weighing the Benefits of Design Patterns | 360 |
| Using Patterns | 360 |
| Example Design Pattern: Abstract Factory | 361 |
| Using Abstract Factory | 361 |
| Solution | 371 |
| Consequences | 374 |
| Dependency Injection Pattern | 376 |
| Problem Using Dependency Injection | 376 |
| Solution | 379 |
| Constructor-Based Injection vs. Property-Based Injection | 380 |
| What Service Implementation to Inject | 380 |
| Benefits of the DI Pattern | 382 |
| Refactoring to DI | 385 |
| Refactoring to Patterns and the Rent-a-Wheels Application | 385 |
| Eliminating Code That Duplicates .NET Framework Functionality | 385 |
| Injecting Data Classes to GUI Classes via Dependency Injection | 386 |
| CRUD Persistence Pattern | 389 |
| Summary | 389 |
| Chapter 13: LINQ and Other C# 3.0 Enhancements | 391 |
| Type Inference for Local Variables | 391 |
| Auto-implemented Properties | 393 |
| Extension Methods | 395 |
| Object and Collection Initializers | 402 |
| Querying Objects with LINQ | 404 |
| Old Example in New Robes | 407 |
| Object-Relational Mapping with LINQ to SQL | 414 |
| LINQ and the Rent-a-Wheels Application | 418 |
| Summary | 427 |

Contents

| | |
|--|------------|
| Chapter 14: A Short History of the Web for Developers and ASP.NET Tooling | 429 |
| Refactor! for ASP.NET | 430 |
| Invoking Refactor! for ASP.NET | 431 |
| Exploring the Refactor! for ASP.NET User Interface | 433 |
| HTML History and Its Legacy | 438 |
| HTML History | 438 |
| Catching Up with the Web | 446 |
| Visual Studio and XHTML | 446 |
| XML and Encoding | 447 |
| Visual Studio DTD Validation for HTML | 448 |
| Serving Strict XHTML | 449 |
| Summary | 451 |
| Chapter 15: Refactoring ASP.NET Applications | 453 |
| Refactoring HTML | 454 |
| Well-Formed XHTML Documents | 454 |
| XHTML Validity | 456 |
| Tool Support for Upgrading Legacy, XHTML Non-Compliant Markup | 458 |
| Pretty-Printing Your HTML Documents | 459 |
| Separating Structure from Presentation | 460 |
| Using HTTP to Your Advantage with REST | 467 |
| Refactoring ASP.NET Code | 472 |
| ASP.NET Code Model: Single-file vs. Code-behind | 472 |
| Master Pages | 476 |
| Web User and Custom Server Controls | 480 |
| Rent-a-Wheels and ASP.NET Refactorings | 486 |
| Summary | 489 |
| Appendix A: Rent-a-Wheels Prototype Internals and Intricacies | 491 |
| Hand Over Button Click Event-Handling Code | 491 |
| Receive Button Click Event-Handling Code | 492 |
| Charge Button Click Event-Handling Code | 492 |
| Change Branch Button Click Event-Handling Code | 493 |
| To Maintenance and From Maintenance | 497 |
| Administer Fleet Form | 498 |
| Delete Button Click Event-Handling Routine | 498 |
| New Button Click Event-Handling Routine | 499 |
| Reload Button Click Event-Handling Routine | 499 |
| Form Load Event-Handling Routine | 499 |

Contents

| | |
|---|-------------------|
| Administer Fleet Form Class Code: Fields | 501 |
| Navigation Buttons Click Event-Handling Routine | 502 |
| Save Button Click Event-Handling Routine | 503 |
| Display Button Click Event-Handling Routine | 505 |
| Summary | 509 |
| <u>Appendix B: Unleash Refactor! for ASP.NET</u> | <u>511</u> |
| Index | 515 |