CONTENTS IN DETAIL

ACKNOWLEDGMENTS **xvii**

INTRODUCTION **xix**

Who Should Read This Book xx
Organization and Requirements. xxi
Modern Features xxiii
Why Value Types? xxiii

1
MAKING THE MOST OF THE TYPE SYSTEM **1**

The Value of Good Names 2
Adding Clarity Through Types 4
Named Arguments 4
Custom Types 5
Encapsulation 6
Immutability 7
Value Validation 8
Testing 9
Refactoring 9
Replacing Magic Numbers with Named Constants 10
Simplifying Properties and Values 11
Overloading Arithmetic Operators 14
Determining a Need for New Types 15
Encoding Units. 18
Itemizing Units with enums 19
Static Creation Methods. 22
Symmetry in Design. 23
Making Units Explicit. 24
Choosing the Most Natural Usage 25
Returning Types Implied by Units. 27
A Fully Formed Encapsulated Value 28
Deciding Whether to Abstract Types. 29
Summary 30

2
VALUE AND REFERENCE TYPES **31**

User-Defined Types. 32
Structs and Classes 32
Records and Record Structs 33
Inheritance 34
Type Instance Lifetimes. 36
Variables 37
Variables vs. Values. 38
Definite Assignment. 39

Instances and Storage . . . 39
Embedded Values . . . 40
Boxed Values . . . 44
Semantics and Type . . . 45
The Common Type System . . . 45
Copy Semantics . . . 46
Records, Structs, and Value Semantics . . . 48
Construction and Initialization . . . 49
Default Initialization . . . 50
Instance Constructors . . . 51
Field Initializers . . . 58
Object Initializers . . . 58
null Values and Default Values . . . 60
Generics and null . . . 61
Generics and Default Values . . . 62
Nullable Value Types . . . 63
Nullable Reference Types . . . 64
The Null-Forgiving Operator . . . 66
Summary . . . 67

3
REFERENCE AND VALUE PARAMETERS 69

Method Parameters and Arguments . . . 70
Reference Types vs. By-Reference Parameters . . . 70
Value Types and Parameters . . . 71
The Value of a Reference . . . 73
Reference Variables and Aliasing . . . 74
Mutable By-Reference Parameters . . . 76
Passing References by Reference . . . 77
Passing Values by Reference . . . 78
Working with Output Parameters . . . 79
Limitations of By-Reference Parameters . . . 82
Property Values . . . 82
Overloading on By-Reference Parameters . . . 83
Using Fields . . . 84
Extension Methods . . . 87
Side Effects and Direct Effects . . . 88
Mutation vs. Creation . . . 89
Declarative Code and Performance . . . 91
Read-Only References and Returning by Reference . . . 92
Returning Values by Reference . . . 94
Preventing Modifications to Data . . . 95
Keeping By-Reference Variables Within Scope . . . 97
Considering Performance vs. Simplicity . . . 101
Final Word on Mutable By-Reference Parameters . . . 102
Summary . . . 103

4
IMPLICIT AND EXPLICIT COPYING 105

Copying by Simple Assignment . . . 106
Value Copy Behavior . . . 107
Read-Only Properties vs. Immutable Types . . . 109
Creating New Objects . . . 110
Overwriting a Value . . . 111
Constructing Value Types . . . 112
Copying Records Like Value Types . . . 114
Identifying Unnecessary Boxing . . . 115
To an Interface . . . 116
In Method Calls . . . 117
Method Parameters and Arguments . . . 118
Passing and Returning by Value . . . 119
Accessing Properties . . . 120
Using Expressions with Operators . . . 121
Modifying Return Type Instances . . . 123
Reference Type Properties . . . 125
Instance Methods and Mutability . . . 126
Properties as Arguments for Read-Only Parameters . . . 127
Defensive Copies . . . 128
Mutable Value Types and in Parameters . . . 129
Automatic vs. Nonautomatic Properties . . . 130
Read-Only Reference Variables . . . 131
Read-Only Fields . . . 132
Defending Against Mutation . . . 133
Read-Only Accessors and Methods . . . 134
Read-Only Types . . . 135
Summary . . . 136

5
TYPES OF EQUALITY 139

Built-in Equality . . . 140
Whole Numbers . . . 140
Floating-Point Values . . . 141
Reference Equality . . . 145
Strings and Value Equality . . . 148
Custom Equality for Classes . . . 149
Defining Equality Operators . . . 150
Handling Comparisons with null . . . 151
Making Type-Safe Comparisons . . . 152
Working with Hash Codes . . . 152
Structs and Equality . . . 156
Overriding Equals for Structs . . . 156
Boxing Values and Comparing by Identity . . . 158
Comparing Generic Variables . . . 162
Generic Code and the Equals Method . . . 163
The IEquatable<T> Interface . . . 164

Compiler-Generated Equality 165
Records and Record Structs 165
Equality for Nullable Values 168
Value Tuples and Equality 170
Summary 172

6
THE NATURE OF VALUES 175

Value vs. Reference Semantics 176
Copying and Equality Comparison Behavior 177
Mutability 181
Mechanics vs. Semantics 182
Object Relationships 183
Kinds of Objects 184
Object Characteristics 185
Design Refinement to Model Object Roles 191
Abstraction and Vocabulary 191
Encapsulation and Cohesion 192
Eliminating Duplication 193
Establishing Class Invariants 194
Clarifying with Symmetry 196
Encapsulation and the Public Interface 196
Extending the Interface 197
Reducing the Internal Interface 198
Composing Abstractions 199
Choosing Between Value and Reference Semantics 199
Avoiding the Pitfalls of Default Variables 200
Implementing Custom vs. Generated Behavior 201
Overriding Generated Methods 202
Comparison for Ordering 203
Equivalence vs. Equality 204
The Contract for Comparisons 205
Other Kinds of Ordering 206
The Perils of Uniformity and Consistency 207
Arithmetic and Nonarithmetic Types 208
Nonstandard Operator Behavior 209
Summary 209

7
VALUE TYPES AND POLYMORPHISM 211

Why Value Types Are Sealed 212
Implementation Inheritance 213
Value-Based Equality for Classes 214
Equality Behavior in Derived Classes 218
Equality Comparisons and Type Substitution 220
Inclusion Polymorphism and Subtyping 222
Working with Input and Output Types of Virtual Methods 223
Upholding a Type's Contract 224
Inheriting Record Types 225
Avoiding Implementation Inheritance 230
Containing Instead of Inheriting Types 231

Parametric Polymorphism with Generics 233
Generic Constraints and Protocol Interfaces 233
Generic Method Parameters and Type Deduction 236
Parameterized Types 237
Ad Hoc Polymorphism with Overloading 238
Symbolic Polymorphism with Overloaded Operators 240
Generic Delegates for Polymorphism 241
Coercion Polymorphism Using Conversions 242
Widening vs. Narrowing Conversions 244
For Representation 244
For Purpose 245
Summary 247

8
PERFORMANCE AND EFFICIENCY 249

Measuring and Optimizing Performance 250
The JIT Compiler 250
Performance Benchmarks 251
The Profiler 252
Measuring Basic Performance with Equals 253
Hidden Costs of Simplicity 255
The ValueType.Equals Method 257
The ValueType.GetHashCode Method 259
The HashCode.Combine Method 259
Optimizing Equality 261
The Effect of IEquatable<T> 263
Property Accesses 265
The Equality Operators 266
How Type Affects Performance 270
Measuring the Cost of Copying 270
Copying Large Instances 272
Weighing Object Construction Costs 273
Measuring the Compiler-Generated Equals Method 277
How Common Idioms and Practices Affect Performance 279
Looping and Iteration 279
Pattern Matching and Selection 284
Summary 286

AFTERWORD 287

APPENDIX: FURTHER READING 289

INDEX 301