CHAPTER 3: **Let Me Say This about this** 287
Passing an Object to a Method 288
Comparing Static and Instance Methods 290
Employing static properties and methods effectively 291
Employing instance properties and methods effectively 293
Expanding a method's full name 295
Accessing the Current Object 296
What is the this keyword? 298
When is the this keyword explicit? 299
Using Local Functions 300
Creating a basic local function 300
Using attributes with local functions 301

CHAPTER 4: **Holding a Class Responsible** 303
Restricting Access to Class Members 303
A public example of public BankAccount 304
Jumping ahead — other levels of security 306
Why You Should Worry about Access Control 307
Accessor methods 308
Working with init-only setters 309
Access control to the rescue — an example 311
Defining Class Properties 313
Static properties 315
Properties with side effects 315
Accessors with access levels 316
Using Target Typing for Your Convenience 316
Dealing with Covariant Return Types 319
Getting Your Objects Off to a Good Start — Constructors 320
The C#-Provided Constructor 321
Replacing the Default Constructor 322
Constructing something 324
Initializing an object directly with an initializer 326
Seeing that construction stuff with initializers 326
Initializing an object without a constructor 327
Using Expression-Bodied Members 329
Creating expression-bodied methods 329
Defining expression-bodied properties 329
Defining expression-bodied constructors and destructors 330
Defining expression-bodied property accessors 330
Defining expression-bodied event accessors 331

CHAPTER 5: **Inheritance: Is That All I Get?** 333
Why You Need Inheritance 334
Inheriting from a BankAccount Class (a More Complex Example) 335
Working with the basic update 336
Tracking the BankAccount and SavingsAccount classes features 339
IS_A versus HAS_A — I'm So Confused_A 342
The IS_A relationship 342
Gaining access to BankAccount by using containment 343
The HAS_A relationship 345
When to IS_A and When to HAS_A 346
Other Features That Support Inheritance 346
Substitutable classes 346
Invalid casts at runtime 347
Avoiding invalid conversions with the is operator 348
Avoiding invalid conversions with the as operator 349

CHAPTER 6: **Poly-what-ism?** 353
Overloading an Inherited Method 354
It's a simple case of method overloading 354
Different class, different method 355
Peek-a-boo — hiding a base class method 355
Polymorphism 361
Using the declared type every time (Is that so wrong?) 362
Using is to access a hidden method polymorphically 364
Declaring a method virtual and overriding it 365
Getting the most benefit from polymorphism 368
C# During Its Abstract Period 368
Class factoring 369
The abstract class: Left with nothing but a concept 373
How do you use an abstract class? 374
Creating an abstract object — not! 377
Sealing a Class 377

CHAPTER 7: **Interfacing with the Interface** 379
Introducing CAN_BE_USED_AS 379
Knowing What an Interface Is 381
How to implement an interface 382
Using the newer C# 8.0 additions 383
How to name your interface 386
Why C# includes interfaces 386
Mixing inheritance and interface implementation 387
And he-e-e-re's the payoff 387

Using an Interface 388
As a method return type 389
As the base type of an array or collection 389
As a more general type of object reference 390
Using the C# Predefined Interface Types 390
Looking at a Program That CAN_BE_USED_AS an Example. 391
Creating your own interface at home in your spare time 391
Implementing the incomparable IComparable<T> interface 392
Creating a list of students. 394
Testing everything using Main(). 395
Unifying Class Hierarchies 396
Hiding Behind an Interface 399
Inheriting an Interface 401
Using Interfaces to Manage Change in Object-Oriented Programs 402
Making flexible dependencies through interfaces 403
Abstract or concrete: When to use an abstract class and when to use an interface 404
Doing HAS_A with interfaces 405

CHAPTER 8: **Delegating Those Important Events**. 407
E.T., Phone Home — The Callback Problem 408
Defining a Delegate. 408
Pass Me the Code, Please — Examples 411
Delegating the task 411
First, a simple example. 412
Considering the Action, Func, and Predicate delegate types 413
A More Real-World Example 415
Putting the app together 416
Setting the properties and adding event handlers. 418
Looking at the workhorse code. 419
Shh! Keep It Quiet — Anonymous Methods 421
Defining the basic anonymous method. 421
Using static anonymous methods. 422
Working with lambda discard parameters 424
Stuff Happens — C# Events. 424
The Observer design pattern. 425
What's an event? Publish/Subscribe. 425
How a publisher advertises its events 426
How subscribers subscribe to an event. 427
How to publish an event. 427
How to pass extra information to an event handler 428
A recommended way to raise your events 429
How observers "handle" an event. 430

CHAPTER 9: **Can I Use Your Namespace in the Library?** 433
Dividing a Single Program into Multiple Source Files 434
Working with Global using Statements 435
Dividing a Single Program into Multiple Assemblies 437
Executable or library? 437
Assemblies 437
Executables 438
Class libraries 439
Putting Your Classes into Class Libraries 439
Creating the projects for a class library 439
Creating a stand-alone class library 440
Adding a second project to an existing solution 442
Creating the code for the library 445
Using a test application to test a library 446
Going Beyond Public and Private: More Access Keywords 448
Internal: For CIA eyes only 448
Protected: Sharing with subclasses 451
Putting Classes into Namespaces 453
Declaring a namespace 454
Using file-scoped namespaces 456
Relating namespaces to the access keyword story 456
Using fully qualified names 458
Working with partial classes 459
Working with Partial Methods 463
Defining what partial methods do 463
Creating a partial method 464

CHAPTER 10: **Improving Productivity with Named and Optional Parameters** 465
Exploring Optional Parameters 466
Working with optional value parameters 466
Avoiding optional reference types 468
Looking at Named Parameters 470
Using Alternative Methods to Return Values 470
Output (out) parameters 471
Working with out variables 471
Returning values by reference 472
Dealing with null Parameters 473

CHAPTER 11: **Interacting with Structures** 475
Comparing Structures to Classes 476
Considering struct limits 476
Understanding the value type difference 477
Determining when to use struct versus class 477

Creating Structures 478
Defining a basic struct 478
Including common struct elements 479
Using supplemental struct elements 482
Working with Read-only Structures 485
Working with Reference Structures 487
Using Structures as Records 489
Managing a single record 489
Adding structures to arrays 489
Overriding methods 490
Using the New Record Type 491
Comparing records to structures and classes 491
Working with a record 492
Using the positional syntax for property definition 493
Understanding value equality 494
Creating safe changes: Nondestructive mutation 494
Using the field keyword 495

BOOK 3: DESIGNING FOR C# 497

CHAPTER 1: Writing Secure Code 499
Designing Secure Software 500
Determining what to protect 500
Documenting the components of the program 501
Decomposing components into functions 502
Identifying potential threats in functions 502
Building Secure Windows Applications 503
Authentication using Windows logon 503
Encrypting information 507
Deployment security 507
Using System.Security 508

CHAPTER 2: Accessing Data 509
Getting to Know System.Data 510
How the Data Classes Fit into the Framework 512
Getting to Your Data 512
Using the System.Data Namespace 513
Setting up a sample database schema 513
Creating the data access project 514
Connecting to a data source 514
Working with the visual tools 519
Writing data code 521