# Contents

**Preface**  \hspace{1cm}  xix

**Before You Begin**  \hspace{1cm}  xxvii

## Introduction to Swift and Xcode 6  \hspace{1cm}  1

1.1 Introduction  \hspace{1cm}  2
1.2 Apple’s OS X® and iOS® Operating Systems: A Brief History  \hspace{1cm}  3
1.3 Objective-C  \hspace{1cm}  3
1.4 Swift: Apple’s Programming Language of the Future  \hspace{1cm}  4
   1.4.1 Key Features of Many Popular Languages  \hspace{1cm}  4
   1.4.2 Performance  \hspace{1cm}  6
   1.4.3 Error Prevention  \hspace{1cm}  6
   1.4.4 Swift Standard Library  \hspace{1cm}  6
   1.4.5 Swift Apps and the Cocoa® and Cocoa Touch® Frameworks  \hspace{1cm}  7
   1.4.6 Swift and Objective-C Interoperability  \hspace{1cm}  9
   1.4.7 Other Apple Swift Resources  \hspace{1cm}  9
1.5 Can I Use Swift Exclusively?  \hspace{1cm}  9
   1.5.1 Objective-C Programmers Who Are Developing New iOS and OS X Apps  \hspace{1cm}  10
   1.5.2 Objective-C Programmers Who Are Enhancing Existing iOS and OS X Apps  \hspace{1cm}  10
   1.5.3 Java, C++ and C# Programmers Who Are New to iOS and OS X App Development  \hspace{1cm}  10
   1.5.4 Significant Language Changes Expected  \hspace{1cm}  10
   1.5.5 A Mixture of Swift and Objective-C  \hspace{1cm}  10
1.6 Xcode 6 Integrated Development Environment  \hspace{1cm}  10
1.7 Creating Swift Apps with Xcode 6  \hspace{1cm}  13
1.8 Web Resources  \hspace{1cm}  18

## Introduction to Swift Programming  \hspace{1cm}  20

2.1 Introduction  \hspace{1cm}  21
2.2 A First Swift Program: Printing a Line of Text  \hspace{1cm}  21
2.3 Modifying Your First Program  \hspace{1cm}  23
2.4 Composing Larger Strings with String Interpolation  \hspace{1cm}  25
2.5 Another Application: Adding Integers  \hspace{1cm}  27

## Contents

2.6 Arithmetic 28
  2.6.1 Automatic Arithmetic Overflow Checking 29
  2.6.2 Operator Precedence 29
2.7 Decision Making: The if Conditional Statement and the Comparative Operators 29
2.8 Wrap-Up 32

3 Introduction to Classes, Objects, Methods and Functions 33
3.1 Introduction 34
3.2 Account Class
  3.2.1 Defining a Class 35
  3.2.2 Defining a Class Attribute as a Stored Property 36
  3.2.3 Defining a public Stored Property with a private Setter 37
  3.2.4 Initializing a Class’s Properties with init 37
  3.2.5 Defining a Class’s Behaviors as Methods 39
3.3 Creating and Using Account Objects 40
  3.3.1 Importing the Foundation Framework 40
  3.3.2 Creating and Configuring an NSNumberFormatter to Format Currency Values 41
  3.3.3 Defining a Function—formatAccountString 42
  3.3.4 Creating Objects and Calling an Initializer 42
  3.3.5 Calling Methods on Objects—Depositing into Account Objects 43
  3.3.6 Calling Methods on Objects—Withdrawing from Account Objects 44
3.4 Value Types vs. Reference Types 45
3.5 Software Engineering with Access Modifiers 46
3.6 Wrap-Up 47

4 Control Statements; Assignment, Increment and Logical Operators 48
4.1 Introduction 49
4.2 Control Statements 49
4.3 if Conditional Statement 50
4.4 if...else Conditional Statement 50
4.5 Compound Assignment Operators 52
4.6 Increment and Decrement Operators 53
4.7 switch Conditional Statement
  4.7.1 Using a switch Statement to Convert Numeric Grades to Letter Grades 55
  4.7.2 Specifying Grade Ranges with the Closed-Range Operator (...) 56
  4.7.3 The default Case 56
  4.7.4 Other Patterns in the case Label 57
  4.7.5 No Automatic Fall Through as in Other C-Based Languages 57

4.8 while Loop Statement  
4.9 do...while Loop Statement  
4.10 for...in Loop Statement and the Range Operators  
  4.10.1 Iterating Over Collections of Values with Closed Ranges, Half-Open Ranges and the Global stride Function  
  4.10.2 Compound-Interest Calculations with for...in  
  4.10.3 Formatting Strings with Field Widths and Justification  
  4.10.4 Performing the Interest Calculations  
  4.10.5 A Warning about Displaying Rounded Values  
4.11 for Loop Statement  
  4.11.1 General Format of a for Statement  
  4.11.2 Scope of a for Statement’s Control Variable  
  4.11.3 Expressions in a for Statement’s Header Are Optional  
4.12 break and continue Statements  
  4.12.1 break Statement Example  
  4.12.2 continue Statement Example  
4.13 Logical Operators  
  4.13.1 Logical AND (&&) Operator  
  4.13.2 Logical OR (||) Operator  
  4.13.3 Short-Circuit Evaluation of Complex Conditions  
  4.13.4 Logical NOT (!) Operator  
4.14 Wrap-Up  

5 Functions and Methods: A Deeper Look; enums and Tuples  
5.1 Introduction  
5.2 Modules in Swift  
5.3 Darwin Module—Using Predefined C Functions  
5.4 Multiple-Parameter Function Definition  
5.5 Random-Number Generation  
5.6 Introducing Enumerations and Tuples  
  5.6.1 Introducing Enumeration (enum) Types  
  5.6.2 Tuples and Multiple Function Return Values  
  5.6.3 Tuples as Function Arguments  
  5.6.4 Accessing the Raw Value of an enum Constant  
5.7 Scope of Declarations  
5.8 Function and Method Overloading  
5.9 External Parameter Names  
5.10 Default Parameter Values  
5.11 Passing Arguments by Value or by Reference  
5.12 Recursion  
5.13 Nested Functions  
5.14 Wrap-Up  

6 Arrays and an Introduction to Closures

6.1 Introduction
6.2 Arrays
6.3 Creating and Initializing Arrays
6.4 Iterating through Arrays
6.5 Adding and Removing Array Elements
6.6 Subscript Expressions with Ranges
6.7 Sorting Arrays; Introduction to Closures
  6.7.1 Closures and Closure Expressions
  6.7.2 Array Methods sort and sorted
  6.7.3 Sorting with Function ascendingOrder
  6.7.4 Using a Fully Typed Closure Expression
  6.7.5 Using a Closure Expression with Inferred Types
  6.7.6 Using a Closure Expression with Inferred Types and an Implicit return
  6.7.7 Using a Closure Expression with Shorthand Argument Names
  6.7.8 Using an Operator Function as a Closure Expression
  6.7.9 Reversing an Array's Elements
6.8 Array Methods filter, map and reduce
  6.8.1 Filtering an Array
  6.8.2 Mapping an Array's Elements to New Values
  6.8.3 Reducing an Array's Elements to a Single Value
  6.8.4 Combining Filtering, Mapping and Reducing
6.9 Card Shuffling and Dealing Simulation; Computed Properties; Optionals
  6.9.1 Class Card
  6.9.2 Class DeckOfCards
  6.9.3 DeckOfCardsInitializer
  6.9.4 DeckOfCards Method shuffle
  6.9.5 DeckOfCards Method deal1Card and Optional Return Values
  6.9.6 Shuffling and Dealing Cards
  6.9.7 Unwrapping Optional Values with Optional Binding and the if or while Statements
6.10 Passing Arrays to Functions
  6.10.1 Passing an Entire Array By Value
  6.10.2 Passing One Array Element By Value
  6.10.3 Passing an Entire Array By Reference
  6.10.4 Passing One Array Element By Reference
6.11 Notes on Pass-By-Value and Pass-By-Reference
6.12 Multidimensional Arrays
6.13 Variadic Parameters
6.14 Wrap-Up

7 Dictionary

7.1 Introduction
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.1.1 What Is a Dictionary?</td>
<td>132</td>
</tr>
<tr>
<td>7.1.2 Dictionary Examples</td>
<td>133</td>
</tr>
<tr>
<td>7.1.3 Dictionary is a Generic Type</td>
<td>133</td>
</tr>
<tr>
<td>7.1.4 Dictionary Is a Value Type</td>
<td>133</td>
</tr>
<tr>
<td>7.1.5 Dictionary Is Implemented as a Hash Table</td>
<td>134</td>
</tr>
<tr>
<td>7.1.6 Dictionary Is Type Safe</td>
<td>134</td>
</tr>
<tr>
<td>7.2 Declaring a Dictionary: Key–Value Pairs and Dictionary Literals</td>
<td>134</td>
</tr>
<tr>
<td>7.2.1 Dictionary Key–Value Pairs and Dictionary Literals</td>
<td>135</td>
</tr>
<tr>
<td>7.2.2 Declaring a Dictionary with Generics and Explicit Typing</td>
<td>136</td>
</tr>
<tr>
<td>7.2.3 Declaring a Dictionary with Type Inference</td>
<td>136</td>
</tr>
<tr>
<td>7.2.4 Invoking Dictionary's description Property Explicitly and Implicitly</td>
<td>136</td>
</tr>
<tr>
<td>7.3 Declaring and Printing Empty Dictionary Objects</td>
<td>136</td>
</tr>
<tr>
<td>7.4 Iterating through a Dictionary with for...in</td>
<td>137</td>
</tr>
<tr>
<td>7.5 General-Purpose Generic Dictionary Printing Function</td>
<td>139</td>
</tr>
<tr>
<td>7.6 Dictionary Equality Operators == and !=</td>
<td>140</td>
</tr>
<tr>
<td>7.7 Dictionary count and isEmpty Properties</td>
<td>141</td>
</tr>
<tr>
<td>7.8 Dictionary Whose Values Are Arrays</td>
<td>142</td>
</tr>
<tr>
<td>7.9 Dictionary's keys and values Properties</td>
<td>143</td>
</tr>
<tr>
<td>7.10 Inserting, Modifying and Removing Key–Value Pairs with Subscripting</td>
<td>145</td>
</tr>
<tr>
<td>7.10.1 Updating the Value of an Existing Key–Value Pair</td>
<td>147</td>
</tr>
<tr>
<td>7.10.2 Adding a New Key–Value Pair</td>
<td>147</td>
</tr>
<tr>
<td>7.10.3 Removing a Key–Value Pair</td>
<td>147</td>
</tr>
<tr>
<td>7.10.4 Subscripting Returns an Optional Value</td>
<td>147</td>
</tr>
<tr>
<td>7.10.5 Processing an Optional Value</td>
<td>148</td>
</tr>
<tr>
<td>7.10.6 Inserting a New Key–Value Pair in an Empty Dictionary</td>
<td>148</td>
</tr>
<tr>
<td>7.11 Inserting, Removing and Modifying Key–Value Pairs</td>
<td>148</td>
</tr>
<tr>
<td>7.11.1 Inserting a Key–Value Pair with Dictionary Method updateValue</td>
<td>150</td>
</tr>
<tr>
<td>7.11.2 Updating a Key–Value Pair with Dictionary Method updateValue</td>
<td>151</td>
</tr>
<tr>
<td>7.11.3 Removing a Key–Value Pair with Dictionary Method removeValueForKey</td>
<td>151</td>
</tr>
<tr>
<td>7.11.4 Attempting to Remove a Nonexistent Key–Value Pair with Method removeValueForKey</td>
<td>151</td>
</tr>
<tr>
<td>7.11.5 Emptying a Dictionary with Method removeAll</td>
<td>151</td>
</tr>
<tr>
<td>7.12 Building a Dictionary Dynamically: Word Counts in a String</td>
<td>151</td>
</tr>
<tr>
<td>7.13 Bridging Between Dictionary and Foundation Classes</td>
<td>153</td>
</tr>
<tr>
<td>7.14 Hash Tables and Hashing</td>
<td>154</td>
</tr>
<tr>
<td>7.15 Wrap-Up</td>
<td>155</td>
</tr>
</tbody>
</table>

### 8 Classes: A Deeper Look and Extensions

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.1 Introduction</td>
<td>158</td>
</tr>
<tr>
<td>8.2 Time Class: Default Initializers and Property Observers</td>
<td>160</td>
</tr>
<tr>
<td>8.2.1 Stored Property Initialization and the Default Initializer</td>
<td>162</td>
</tr>
</tbody>
</table>

## Contents

8.2.2 willSet and didSet Property Observers for Stored Properties  162
8.2.3 Computed Read-Only Properties universalDescription and description  163
8.2.4 Using Class Time  164
8.3 Designated and Convenience Initializers in Class Time  166
8.3.1 Class Time with Overloaded Initializers  166
8.3.2 Designated Initializers  167
8.3.3 Convenience Initializers and Initializer Delegation with self  168
8.3.4 Using Class Time’s Designated and Convenience Initializers  169
8.4 Failable Initializers in Class Time  170
8.4.1 Failable Designated Initializers  172
8.4.2 Failable Convenience Initializers  172
8.4.3 Implicitly Unwrapped Failable Initializers  173
8.4.4 Invoking Failable Initializers  173
8.5 Extensions to Class Time  174
8.5.1 Class Time with Extensions  175
8.5.2 Testing Class Time’s Extensions  177
8.5.3 Extensions and Access Modifiers  178
8.6 Read-Write Computed Properties  178
8.7 Composition  181
8.7.1 Class Employee  181
8.7.2 Testing Class Employee  183
8.8 Automatic Reference Counting, Strong References and Weak References  184
8.9 Deinitializers  185
8.10 Using NSDecimalNumber for Precise Monetary Calculations  185
8.11 Type Properties and Type Methods  187
8.11.1 Type Scope  188
8.11.2 Motivating Type Properties  188
8.11.3 Creating Type Properties and Type Methods in Classes  189
8.11.4 Using Type Properties and Type Methods  190
8.12 Lazy Stored Properties and Delayed Initialization  191
8.13 Wrap-Up  192

9 Structures, Enumerations and Nested Types  194
9.1 Introduction  195
9.2 Structure Definitions  196
9.2.1 Time struct Definition with Default and Memberwise Initializers  198
9.2.2 Custom Initializers extension to struct Time  198
9.2.3 Computed Properties extension to struct Time  199
9.2.4 Mutating Methods extension to struct Time  199
9.2.5 Testing the Time struct  200
9.3 Enumerations and Nested Types  202
9.3.1 Card struct with Nested Suit and Face enum Types  202
9.3.2 DeckOfCards struct  205

9.3.3 Testing the struct Types Card and DeckOfCards, and the enum Types Suit and Face 207

9.4 Choosing Among Structures, Enumerations and Classes in Your Apps 209

9.5 Associated Values for enums 210

9.6 Wrap-Up 212

10 Inheritance, Polymorphism and Protocols 214

10.1 Introduction 215

10.1.1 Superclasses and Subclasses 215

10.1.2 Polymorphism 216

10.1.3 Implementing for Extensibility 216

10.1.4 Programming in the Specific 216

10.1.5 Protocols 217

10.2 Superclasses and Subclasses 217

10.3 An Inheritance Hierarchy: CommunityMembers 218

10.4 Case Study: Using Inheritance to Create Related Employee Types 218

10.4.1 Superclass CommissionEmployee 220

10.4.2 Subclass BasePlusCommissionEmployee 221

10.4.3 Testing the Class Hierarchy 224

10.5 Access Modifiers in Inheritance Hierarchies 226

10.6 Introduction to Polymorphism: A Polymorphic Video Game Discussion 227

10.7 Case Study: Payroll System Class Hierarchy Using Polymorphism 228

10.7.1 Base Class Employee 229

10.7.2 Subclass SalariedEmployee 231

10.7.3 Subclass CommissionEmployee 232

10.7.4 Indirect Subclass BasePlusCommissionEmployee 233

10.7.5 Polymorphic Processing 235

10.8 Case Study: Creating and Using Custom Protocols 238

10.8.1 Protocol Capabilities Must Be Defined in Each Conforming Type 238

10.8.2 Protocols and Is-a Relationships 238

10.8.3 Relating Disparate Types Via Protocols 238

10.8.4 Accounts-Payable Application 239

10.8.5 Developing a Payable Hierarchy 239

10.8.6 Declaring Protocol Payable 240

10.8.7 Creating Class Invoice 241

10.8.8 Using extensions to Add Printable and Payable Protocol Conformance to Class Employee Polymorphically 242

10.8.9 Using Protocol Payable to Process Invoices and Employees Polymorphically 244

10.9 Additional Protocol Features 246

10.9.1 Protocol Inheritance 246

10.9.2 Class-Only Protocols 246

10.9.3 Optional Capabilities in Protocols 246

10.9.4 Protocol Composition 247

10.9.5 Common Protocols in Swift 247

## Contents

10.10 Using `final` to Prevent Method Overriding and Inheritance  248  
10.11 Initialization and Deinitialization in Class Hierarchies  248  
  10.11.1 Basic Class-Instance Initialization  248  
  10.11.2 Initialization in Class Hierarchies  249  
  10.11.3 Initialization of a `BasePlusCommissionEmployee` Object  250  
  10.11.4 Overriding Initializers and Required Initializers  250  
  10.11.5 Deinitialization in Class Hierarchies  251  
10.12 Wrap-Up  251  

## 11 Generics  253  
11.1 Introduction  254  
11.2 Motivation for Generic Functions  254  
11.3 Generic Functions: Implementation and Specialization  255  
11.4 Type Parameters with Type Constraints  258  
11.5 Overloading Generic Functions  259  
11.6 Generic Types  259  
11.7 Note About Associated Types for Protocols  263  
11.8 Wrap-Up  263  

## 12 Operator Overloading and Subscripts  264  
12.1 Introduction  265  
12.2 String Operators and Methods  266  
  12.2.1 String Variables and Constants  268  
  12.2.2 String Comparative Operators  268  
  12.2.3 Custom String Unary Prefix Operator `!`  269  
  12.2.4 String Concatenation with Operators `+` and `+=`  269  
  12.2.5 String Subscript `([])` Operator for Creating Substrings  270  
  12.2.6 Other String Methods  270  
  12.3 Custom `Complex` Numeric Type with Overloaded Arithmetic Operators  271  
  12.3.1 Overloaded Operator Functions `+`, `-`, and `*`  272  
  12.3.2 Overloading the Arithmetic Assignment Operator `+=`  272  
  12.3.3 Performing Arithmetic with `Complex` Numbers  273  
12.4 Overloading Arithmetic Operators for Class `NSDecimal1Number`  274  
  12.4.1 Overloading the Multiplication Operator `(*)`  275  
  12.4.2 Overloading the Addition Operator `+`  276  
  12.4.3 Using the Overloaded Operators  276  
  12.4.4 Overloading the `*=` Multiplication Assignment Operator  276  
12.5 Overloading Unary Operators: `++` and `--`  276  
  12.5.1 Overloading Unary Prefix Operators That Modify Their Operands  278  
  12.5.2 Overloading Unary Postfix Operators That Modify Their Operands  278  
  12.5.3 Swift's `AnyObject` Type—Bridging Between Objective-C and Swift  278  
12.6 Overloading Subscripts  279
12.6.1 Box Type with Custom Subscripts 279
12.6.2 Subscript Syntax 281
12.6.3 Type Box’s Int Subscript and the precondition Function 281
12.6.4 Type Box’s String Subscript 282
12.6.5 Using Type Box’s Subscripts 282
12.7 Custom Operators 283
12.7.1 Precedence and Associativity 283
12.7.2 Symbols Used in Custom Operators 284
12.7.3 Defining a Custom Exponentiation Operator for Type Int 285
12.8 Custom Generic Operators 286
12.9 Wrap-Up 287

13 iOS 8 App Development: Welcome App 288
13.1 Introduction 289
13.2 Technologies Overview 290
13.2.1 Xcode and Interface Builder 290
13.2.2 Labels and Image Views 290
13.2.3 Asset Catalogs and Image Sets 291
13.2.4 Running the App 291
13.2.5 Accessibility 291
13.2.6 Internationalization 291
13.3 Creating a Universal App Project with Xcode 291
13.3.1 Xcode Projects and App Templates 291
13.3.2 Creating and Configuring a Project 292
13.4 Xcode Workspace Window 293
13.4.1 Navigator Area 294
13.4.2 Editor Area 294
13.4.3 Utilities Area and Inspectors 295
13.4.4 Debug Area 295
13.4.5 Xcode Toolbar 295
13.4.6 Project Navigator 296
13.4.7 Keyboard Shortcuts 296
13.5 Storyboarding the Welcome App’s UI 296
13.5.1 Configuring the App for Portrait and Landscape Orientations 297
13.5.2 Providing an App Icon 297
13.5.3 Creating an Image Set for the App’s Image 299
13.5.4 Overview of the Storyboard and the Xcode Utilities Area 300
13.5.5 Adding an Image View to the UI 302
13.5.6 Using Inspectors to Configure the Image View 302
13.5.7 Adding and Configuring the Label 304
13.5.8 Using Auto Layout to Support Different Screen Sizes and Orientations 306
13.6 Running the Welcome App 308
13.6.1 Testing on the iOS Simulator 308
# Contents

13.6.2 Testing on a Device (for Paid Apple iOS Developer Program Members Only)  311
13.7 Making Your App Accessible  311
  13.7.1 Enabling Accessibility for the Image View  311
  13.7.2 Confirming Accessibility Text with the Simulator’s Accessibility Inspector  312
13.8 Internationalizing Your App  313
  13.8.1 Locking Your UI During Translation  314
  13.8.2 Exporting Your UI’s String Resources  315
  13.8.3 Translating the String Resources  316
  13.8.4 Importing the Translated String Resources  316
  13.8.5 Testing the App in Spanish  317
13.9 Wrap-Up  318

## 14 iOS 8 App Development: Tip Calculator App  319

14.1 Introduction  320
14.2 Test-Driving the Tip Calculator App in the iPhone and iPad Simulators  321
14.3 Technologies Overview  322
  14.3.1 Swift Programming  322
  14.3.2 Swift Apps and the Cocoa Touch® Frameworks  322
  14.3.3 Using the UIKit and Foundation Frameworks in Swift Code  323
  14.3.4 Creating Labels, a Text Field and a Slider with Interface Builder  324
  14.3.5 View Controllers  324
  14.3.6 Linking UI Components to Your Swift Code  324
  14.3.7 Performing Tasks After a View Loads  325
  14.3.8 Bridging Between Swift and Objective-C Types  325
14.4 Building the App’s UI  325
  14.4.1 Creating the Project  325
  14.4.2 Configuring the Size Classes for Designing a Portrait Orientation iPhone App  327
  14.4.3 Adding the UI Components  327
  14.4.4 Adding the Auto Layout Constraints  334
14.5 Creating Outlets with Interface Builder  337
14.6 Creating Actions with Interface Builder  340
14.7 Class ViewController  341
  14.7.1 import Declarations  342
  14.7.2 ViewController Class Definition  342
  14.7.3 ViewController’s @IBOutlet Properties  342
  14.7.4 Other ViewController Properties  343
  14.7.5 Overridden UIViewController method viewDidLoad  344
  14.7.6 ViewController Action Method calculateTip  345
  14.7.7 Global Utility Functions Defined in ViewController.swift  347
14.8 Wrap-Up  349
