

УДК 004.432
ББК 32.972.1
В26

В26 Чейрд ин'т Вейн

Swift. Подробно / Пер. с англ. Д. А. Беликова. – М.: ДМК Пресс, 2020. – 412 с.
ISBN 978-5-97060-780-0

Данная книга знакомит вас с навыками, необходимыми для создания профессионального программного обеспечения для платформ Apple, таких как iOS и MacOS. Вы освоите такие мощные методы, как обобщение, эффективная обработка ошибок, протольно-ориентированное программирование и современные шаблоны Swift.

Издание рассчитано на программистов продвинутого и среднего уровней.

УДК 004.432
ББК 32.972.1

Все права защищены. Любая часть этой книги не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами без письменного разрешения владельцев авторских прав

Материал, изложенный в данной книге, многократно проверен. Но, поскольку вероятность технических ошибок все равно существует, издательство не может гарантировать абсолютную точность и правильность приводимых сведений. В связи с этим издательство не несет ответственности за возможные ошибки, связанные с использованием книги.

Содержание

Предисловие	14
Благодарности	15
Об этой книге	16
Почему эта книга?	16
Подходит ли вам эта книга?	17
Чем эта книга не является	18
Особый акцент на практических сценариях	18
Дорожная карта	18
О коде	24
Книжный форум	25
Об авторе	25
Об иллюстрации на обложке	25
Предисловие от издательства	26
Отзывы и пожелания	26
Список опечаток	26
Нарушение авторских прав	26
Глава 1. Введение	28
1.1. «Золотая середина» SWIFT	28
1.2. Под поверхность	30
1.3. Минусы Swift	30
1.3.1. Стабильность ABI	30
1.3.2. Строгость	31
1.3.3. Сложность протоколов	31
1.3.4. Параллелизм	33
1.3.5. Отход от платформ Apple	34
1.3.6. Время компиляции	34
1.4. Чему вы научитесь	35
1.5. Как извлечь максимум из этой книги	35
1.6. Минимальная квалификация	36
1.7. Версия Swift	36
Глава 2. Моделирование данных с помощью перечислений	37
2.1. OR в сравнении с AND	38
2.1.1. Моделирование данных с помощью структуры	38
2.1.2. Превращаем структуру в перечисление	41
2.1.3. Выбор между структурами и перечислениями	43
2.2. Перечисления для полиморфизма	44
2.2.1. Полиморфизм на этапе компиляции	44
2.3. Перечисления вместо создания подклассов	46
2.3.1. Формирование модели для приложения Workout	47
2.3.2. Создание суперкласса	48
2.3.3. Недостатки подклассов	48
2.3.4. Рефакторинг модели данных с помощью перечислений	49

6 ❖ Содержание

2.3.5. Подклассы или перечисления – что выбрать	51
2.3.6. Упражнения	51
2.4. Алгебраические типы данных	51
2.4.1. Типы-суммы	52
2.4.2. Типы-произведения	53
2.4.3. Распределение суммы в перечислении	53
2.4.4. Упражнение	55
2.5. Более безопасное использование строк	56
2.5.1. Опасность необработанных значений	57
2.5.2. Сопоставление для строк	59
2.5.3. Упражнения	62
2.6. В заключение	62
Глава 3. Написание более чистых свойств	66
3.1. Вычисляемые свойства	66
3.1.1. Моделирование упражнения	67
3.1.2. Преобразование функций в вычисляемые свойства	68
3.1.3. Завершение	70
3.2. Ленивые свойства	70
3.2.1. Создание учебного плана	70
3.2.2. Когда вычисляемые свойства не помогают	71
3.2.3. Использование ленивых свойств	73
3.2.4. Делаем ленивое свойство устойчивым	74
3.2.5. Изменяемые и ленивые свойства	75
3.2.6. Упражнения	77
3.3. Наблюдатели свойств	79
3.3.1. Обрезка пробелов	79
3.3.2. Запуск наблюдателей свойств из инициализаторов	81
3.3.3. Упражнения	82
3.4. В заключение	83
Глава 4. Делаем опционалы второй натурой	87
4.1. Назначение опционалов	88
4.2. Чистое извлечение значений	89
4.2.1. Сопоставление для опционалов	90
4.2.2. Методы извлечения	91
4.2.3. Когда значение вас не интересует	92
4.3. Соккрытие переменной	93
4.3.1. Реализация протокола CustomStringConvertible	93
4.4. Когда опционалы запрещены	94
4.4.1. Добавление вычисляемого свойства	95
4.5. Возврат опциональных строк	96
4.6. Детальный контроль над опционалами	98
4.6.1. Упражнения	99
4.7. Откат назад, если опционал равен nil	99
4.8. Упрощение опциональных перечислений	99
4.8.1. Упражнение	101

4.9. Цепочки опционалов	102
4.10. Ограничение опциональных логических типов	103
4.10.1. Сокращение логического типа до двух состояний	104
4.10.2. Откат к значению true	104
4.10.3. Логический тип данных с тремя состояниями	105
4.10.4. Реализация протокола RawRepresentable	106
4.10.5. Упражнение	107
4.11. Рекомендации по принудительному извлечению значения	108
4.11.1. Когда принудительное извлечение значения является «приемлемым»	109
4.11.2. Аварийный сбой со стилем	109
4.12. Приручаем неявно извлекаемые опционалы	110
4.12.1. Как распознать неявно извлекаемый опционал	111
4.12.2. Неявно извлекаемые опционалы на практике	111
4.12.3. Упражнение	114
4.13. В заключение	114
Глава 5. Разбираемся с инициализаторами	117
5.1. Правила инициализаторов структуры	117
5.1.1. Пользовательские инициализаторы	118
5.1.2. Странность инициализатора структуры	120
5.1.3. Упражнения	121
5.2. Инициализаторы и подклассы	121
5.2.1. Создание суперкласса настольной игры	122
5.2.2. Инициализаторы BoardGame	123
5.2.3. Создание подкласса	125
5.2.4. Потеря вспомогательных инициализаторов	126
5.2.5. Возвращение инициализаторов суперкласса	127
5.2.6. Упражнение	129
5.3. Минимизация инициализаторов класса	130
5.3.1. Реализация назначенного инициализатора в качестве вспомогательного с использованием ключевого слова override	130
5.3.2. Деление подкласса на подклассы	132
5.3.3. Упражнение	133
5.4. Требуемые инициализаторы	134
5.4.1. Фабричные методы	134
5.4.2. Протоколы	136
5.4.3. Когда классы являются финальными	137
5.4.4. Упражнения	138
5.5. В заключение	138
Глава 6. Непринужденная обработка ошибок	142
6.1. Ошибки в Swift	143
6.1.1. Протокол Error	144
6.1.2. Генерация ошибок	144
6.1.3. Swift не показывает ошибки	145
6.1.4. Сохранение среды в предсказуемом состоянии	147

8 ❖ Содержание

6.1.5. Упражнения	150
6.2. Распространение ошибок и перехват	151
6.2.1. Распространение ошибок	151
6.2.2. Добавление технических деталей для устранения неполадок	154
6.2.3. Централизация обработки ошибок	158
6.2.4. Упражнения	160
6.3. Создание симпатичных API	161
6.3.1. Сбор достоверных данных в типе	161
6.3.2. Ключевое слово try?	163
6.3.3. Ключевое слово try!	164
6.3.4. Возвращение опционалов	164
6.3.5. Упражнение	165
6.4. В заключение	165
Глава 7. Обобщения	168
7.1. Преимущества обобщений	169
7.1.1. Создание обобщенной функции	170
7.1.2. Рассмотрение обобщений	172
7.1.3. Упражнение	174
7.2. Ограничение обобщений	174
7.2.1. Нужна функция ограничения	175
7.2.2. Протоколы Equatable и Comparable	176
7.2.3. Ограничивать значит специализировать	177
7.2.4. Реализация протокола Comparable	178
7.2.5. Ограничение в сравнении с гибкостью	179
7.3. Ряд ограничений	179
7.3.1. Протокол Hashable	180
7.3.2. Комбинируем ограничения	181
7.3.3. Упражнения	182
7.4. Создание обобщенного типа	182
7.4.1. Желание совместить два типа, соответствующих протоколу Hashable	183
7.4.2. Создание типа Pair	183
7.4.3. Несколько обобщений	184
7.4.4. Соответствие протоколу Hashable	185
7.4.5. Упражнение	187
7.5. Обобщения и подтипы	187
7.5.1. Подтипы и инвариантность	188
7.5.2. Инвариантность в Swift	189
7.5.3. Универсальные типы Swift получают особые привилегии	190
7.6. В заключение	191
Глава 8. Становимся профессионалами в протольно-ориентированном программировании	194
8.1. Время выполнения в сравнении со временем компиляции	195
8.1.1. Создание протокола	195
8.1.2. Обобщения в сравнении с протоколами	196

8.1.3. Находим компромисс.	197
8.1.4. Переход ко времени выполнения.	198
8.1.5. Выбор между временем компиляции и временем выполнения.	199
8.1.6. Когда обобщение – лучший вариант	200
8.1.7. Упражнения	201
8.2. Зачем нужны ассоциированные типы	202
8.2.1. Недостатки протоколов	203
8.2.2. Попытка превратить все в протокол	204
8.2.3. Разработка обобщенного протокола	205
8.2.4. Моделирование протокола с ассоциированными типами.	206
8.2.5. Реализация протокола с ассоциированными типами	207
8.2.6. Протоколы с ассоциированными типами в стандартной библиотеке.	209
8.2.7. Другие случаи использования ассоциированных типов	209
8.2.8. Упражнение	210
8.3. Передача протокола с ассоциированными типами	211
8.3.1. Использование оператора where с ассоциированными типами	212
8.3.2. Типы, ограничивающие ассоциированные типы	213
8.3.3. Очистка API и наследование протокола.	215
8.3.4. Упражнения	216
8.4. В заключение.	217
Глава 9. Итераторы, последовательности и коллекции	221
9.1. Итерация.	222
9.1.1. Циклы и метод makeIterator	222
9.1.2. IteratorProtocol	223
9.1.3. Протокол Sequence.	224
9.1.4. Посмотрим поближе.	224
9.2. Сила Sequence.	226
9.2.1. Метод filter	226
9.2.2. Метод forEach.	226
9.2.3. Метод enumerated	227
9.2.4. Ленивая итерация	228
9.2.5. Метод reduce	229
9.2.6. Метод reduce into	230
9.2.7. Метод zip.	232
9.2.8. Упражнения	233
9.3. Создание обобщенной структуры данных с помощью Sequence	233
9.3.1. Bag в действии.	233
9.3.2. Создаем BagIterator	236
9.3.3. Реализация AnyIterator	238
9.3.4. Реализация ExpressibleByArrayLiteral.	239
9.3.5. Упражнение	240
9.4. Протокол Collection	241
9.4.1. Ландшафт Collection.	242
9.4.2. MutableCollection	242

10 ❖ Содержание

9.4.3. RangeReplaceableCollection	244
9.4.4. BidirectionalCollection	245
9.4.5. RandomAccessCollection	245
9.5. Создание коллекции	246
9.5.1. Создание плана поездки	247
9.5.2. Реализация Collection	248
9.5.3. Пользовательские сабскрипты	249
9.5.4. ExpressibleByDictionaryLiteral	250
9.5.5. Упражнение	251
9.6. В заключение	252
Глава 10. map, flatMap и compactMap	255
10.1. Знакомство с map	256
10.1.1. Создание конвейера с помощью метода map	258
10.1.2. Использование метода map для словарей	260
10.1.3. Упражнения	261
10.2. Последовательности	262
10.2.1. Упражнение	263
10.3. Использование метода map для опционалов	263
10.3.1. Когда использовать метод map с опционалами	264
10.3.2. Создание обложки	265
10.3.3. Более короткий вариант нотации	267
10.3.4. Упражнение	269
10.4. map – это абстракция	269
10.5. Овладеваем методом flatMap	270
10.5.1. В чем преимущества flatMap?	270
10.5.2. Когда метод map не подходит	271
10.5.3. Борьба с пирамидой гибели	273
10.5.4. Использование метода flatMap с опционалом	275
10.6. flatMap и коллекции	279
10.6.1. flatMap и строки	280
10.6.2. Сочетание flatMap и map	281
10.6.3. compactMap	282
10.6.4. Вложенность или цепочки	283
10.6.5. Упражнения	285
10.7. В заключение	285
Глава 11. Асинхронная обработка ошибок с помощью типа Result	290
11.1. Зачем использовать тип Result?	291
11.1.1. Как раздобыть Result	292
11.1.2. Result похож на Optional, но с изюминкой	293
11.1.3. Преимущества Result	294
11.1.4. Создание API с использованием типа Result	295
11.1.5. Из Cocoa Touch в Result	297
11.2. Распространение Result	299
11.2.1. Создание псевдонимов типов	299

11.2.2. Функция search	300
11.3. Преобразование значений внутри Result	302
11.3.1. Упражнение	304
11.3.2. Использование метода flatMap для типа Result	304
11.3.3. Упражнения	306
11.4. Смешивание Result с функциями, генерирующими ошибку	307
11.4.1. От генерации ошибки к типу Result	307
11.4.2. Преобразование генерирующей функции внутри flatMap	309
11.4.3. Пропускаем ошибки через конвейер	310
11.4.4. Подходя к концу	312
11.4.5. Упражнение	312
11.5. Несколько ошибок внутри Result	313
11.5.1. Знакомство с AnyError	313
11.6. Невообразимый провал и Result	316
11.6.1. Когда протокол определяет Result	316
11.7. В заключение	319
Глава 12. Расширения протоколов.	325
12.1. Наследование классов в сравнении с наследованием протоколов	326
12.1.1. Моделирование данных по горизонтали, а не по вертикали	327
12.1.2. Создание расширения протокола	328
12.1.3. Несколько расширений	329
12.2. Наследование в сравнении с композицией	330
12.2.1. Протокол Mailer	330
12.2.2. Наследование протокола	331
12.2.3. Композиционный подход	332
12.2.4. Высвобождаем энергию пересечения	334
12.2.5. Упражнение	336
12.3. Переопределение приоритетов	336
12.3.1. Переопределение реализации по умолчанию	336
12.3.2. Переопределение и наследование протоколов	337
12.3.3. Упражнение	338
12.4. Расширение в двух направлениях	339
12.4.1. Выбор расширений	339
12.4.2. Упражнение	341
12.5. Расширение с использованием ассоциированных типов	341
12.5.1. Специализированное расширение	343
12.5.2. Недостаток расширения	344
12.6. Расширение с конкретными ограничениями	344
12.7. Расширение протокола Sequence	346
12.7.1. Заглянем внутрь метода filter	346
12.7.2. Создание метода take (while ;)	348
12.7.3. Создание метода Inspect	350
12.7.4. Упражнение	351
12.8. В заключение	352

12 ❖ Содержание

Глава 13. Шаблоны Swift	354
13.1. Внедрение зависимости	355
13.1.1. Замена реализации	355
13.1.2. Передача пользовательской версии Session	356
13.1.3. Ограничение ассоциированного типа	357
13.1.4. Замена реализации	358
13.1.5. Модульное тестирование и мокирование с использованием ассоциированных типов	360
13.1.6. Использование типа Result	362
13.1.7. Упражнение	363
13.2. Условное соответствие	364
13.2.1. Бесплатная функциональность	364
13.2.2. Условное соответствие для ассоциированных типов	365
13.2.3. Делаем так, чтобы Array условно соответствовал пользовательскому протоколу	366
13.2.4. Условное соответствие и обобщения	367
13.2.5. Условное соответствие для типов	368
13.2.6. Упражнение	372
13.3. Что делать с недостатками	372
13.3.1. Как избежать использования перечисления	374
13.3.2. Стирание типов	375
13.3.3. Упражнение	379
13.4. Альтернатива протоколам	380
13.4.1. Чем мощнее, тем хуже код	380
13.4.2. Создание обобщенной структуры	382
13.4.3. Эмпирические правила полиморфизма	383
13.5. В заключение	384
Глава 14. Написание качественного кода на языке Swift	387
14.1. Документация по API	388
14.1.1. Как работает Quick Help	388
14.1.2. Добавление выносок в Quick Help	389
14.1.3. Документация в качестве HTML с использованием Jazzy	391
14.2. Комментарии	392
14.2.1. Объясняем причину	392
14.2.2. Объясняем только непонятные элементы	393
14.2.3. Код несет истину	393
14.2.4. Комментарии – не повязка для неудачных имен	393
14.2.5. Зомби-код	394
14.3. Правила стиля	394
14.3.1. Согласованность – это ключ	395
14.3.2. Обеспечение соблюдения правил с помощью линтера	395
14.3.3. Установка SwiftLint	396
14.3.4. Настройка SwiftLint	397
14.3.5. Временное отключение правил SwiftLint	398
14.3.6. Автозамена правил SwiftLint	399

14.3.7. Синхронизация SwiftLint.	399
14.4. Избавляемся от менеджеров.	400
14.4.1. Ценность менеджеров.	400
14.4.2. Атака на менеджеров.	401
14.4.3. Прокладываем дорогу для обобщений.	401
14.5. Именованние абстракций.	402
14.5.1. Обобщенное или конкретное.	403
14.5.2. Хорошие имена не меняются.	403
14.5.3. Именованние обобщений.	404
14.6. Контрольный список.	404
14.7. В заключение.	405
Глава 15. Что дальше?	407
15.1. Создавайте фреймворки, предназначенные для Linux.	407
15.2. Изучите диспетчер пакетов Swift.	407
15.3. Изучайте фреймворки.	408
15.4. Бросьте себе вызов.	408
15.4.1. Присоединяйтесь к эволюции Swift.	409
15.4.2. Заключительные слова.	409
Предметный указатель.	410