

УДК 004.43Kotlin  
ББК 32.972  
С61

**Сомон. П.-И.**  
С61 Волшебство Kotlin / пер. с англ. А. Н. Киселева. – М.: ДМК Пресс, 2020. – 536 с. : ил.

**ISBN 978-5-97060-801-2**

Kotlin – один из самых новых языков в экосистеме Java, устраняющий многие ограничения Java и гораздо более универсальный. Среди его преимуществ: полная совместимость с Java и возможность интеграции на уровне исходного кода, широкая поддержка парадигмы функционального программирования, помогающая писать надежный и безопасный код, лаконичность синтаксиса, а также, что весьма немало-важно, гарантии поддержки со стороны IT-гиганта Google.

Пьер-Ив Сомон, опытный разработчик на Java, в своей книге подробно освещает нюансы программирования на Kotlin, переходя от общего описания языка к его характерным особенностям и возможностям, включая приемы функционального программирования.

Издание предназначено для разработчиков, знакомых с Java и стремящихся повысить безопасность своих программ, а также упростить их написание, тестирование и сопровождение.

УДК 004.43Kotlin  
ББК 32.972

Original English language edition published by Manning Publications USA, USA. Copyright © 2018 by Manning Publications Co. Russian-language edition copyright © 2020 by DMK Press. All rights reserved.

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.

ISBN 978-1-617-29536-2 (англ.)  
ISBN 978-5-97060-801-2 (рус.)

Copyright © 2018 by Manning Publications Co.  
© Оформление, издание, перевод, ДМК Пресс, 2020

# Оглавление

---

1	■ Создание безопасных программ .....	31
2	■ Функциональное программирование на Kotlin: обзор.....	46
3	■ Программирование с функциями .....	81
4	■ Рекурсия, сорекурсия и мемоизация .....	120
5	■ Обработка данных с использованием списков .....	161
6	■ Необязательные данные .....	197
7	■ Обработка ошибок и исключений.....	222
8	■ Дополнительные операции со списками.....	248
9	■ Ленивые вычисления .....	282
10	■ Обработка данных с использованием деревьев.....	323
11	■ Решение задач с использованием усовершенствованных деревьев ...	363
12	■ Функциональный ввод/вывод .....	392
13	■ Общее изменяемое состояние и акторы.....	420
14	■ Решение типичных проблем функциональным способом.....	448

# Содержание

---

Оглавление .....	5
Предисловие .....	15
Благодарности .....	19
О книге .....	20
Об авторе .....	28
Об иллюстрации на обложке .....	29
<b>1 Создание безопасных программ .....</b>	<b>31</b>
1.1 Программные ловушки .....	33
1.1.1 Безопасная обработка эффектов .....	35
1.1.2 Увеличение безопасности программ за счет ссылочной прозрачности .....	36
1.2 Выгоды безопасного программирования .....	37
1.2.1 Использование подстановочной модели в рассуждениях о программе .....	39
1.2.2 Применение принципов соблюдения безопасности на простом примере .....	40
1.2.3 Максимальное использование абстракций .....	44
Итоги .....	45
<b>2 Функциональное программирование на Kotlin: обзор .....</b>	<b>46</b>
2.1 Поля и переменные в Kotlin .....	47
2.1.1 Тип можно опустить для простоты .....	47
2.1.2 Изменяемые поля .....	47
2.1.3 Отложенная инициализация .....	48
2.2 Классы и интерфейсы в Kotlin .....	49
2.2.1 Еще большее сокращение кода .....	51
2.2.2 Реализация интерфейса или расширение класса .....	51

2.2.3	Создание экземпляра класса .....	52
2.2.4	Перегрузка конструкторов.....	52
2.2.5	Создание методов <i>equals</i> и <i>hashCode</i> .....	53
2.2.6	Деструктуризация объектов данных.....	55
2.2.7	Реализация статических членов в <i>Kotlin</i> .....	55
2.2.8	Синглтоны .....	56
2.2.9	Предотвращение создания экземпляров служебных классов.....	56
2.3	В <i>Kotlin</i> нет элементарных типов .....	57
2.4	Два типа коллекций в <i>Kotlin</i> .....	57
2.5	Пакеты в <i>Kotlin</i> .....	59
2.6	Видимость в <i>Kotlin</i> .....	60
2.7	Функции в <i>Kotlin</i> .....	61
2.7.1	Объявление функций.....	61
2.7.2	Локальные функции .....	62
2.7.3	Переопределение функций.....	63
2.7.4	Функции-расширения .....	63
2.7.5	Лямбда-выражения .....	64
2.8	Пустое значение <i>null</i> в <i>Kotlin</i> .....	66
2.8.1	Приемы работы с типами, поддерживающими <i>null</i> .....	66
2.8.2	Оператор Элвис и значение по умолчанию.....	67
2.9	Поток выполнения программы и управляющие структуры.....	67
2.9.1	Условная конструкция .....	68
2.9.2	Использование конструкций с несколькими условиями.....	69
2.9.3	Циклы .....	70
2.9.4	Какие проблемы может вызывать поддержка вариантности?.....	76
2.9.5	Когда использовать объявления ковариантности и контрвариантности .....	77
2.9.6	Объявление вариантности в точке определения и точке использования .....	78
	Итоги .....	79

<b>3</b>	<b>Программирование с функциями .....</b>	<b>81</b>
3.1	Что такое функция? .....	82
3.1.1	Отношения между двумя областями функций .....	83
3.1.2	Обратные функции в <i>Kotlin</i> .....	84
3.1.3	Частичные функции .....	85
3.1.4	Композиция функций.....	86
3.1.5	Функции нескольких аргументов .....	86
3.1.6	Каррирование функций .....	87
3.1.7	Частично-примененные функции .....	87
3.1.8	Функции не имеют эффектов.....	88
3.2	Функции в <i>Kotlin</i> .....	89

3.2.1	Функции как данные .....	89
3.2.2	Данные как функции.....	89
3.2.3	Конструкторы объектов как функции .....	89
3.2.4	Использование функций <code>fun</code> в <code>Kotlin</code> .....	90
3.2.5	Объектная и функциональная нотация .....	93
3.2.6	Использование функций как значений .....	94
3.2.7	Ссылки на функции .....	96
3.2.8	Композиция функций <code>fun</code> .....	97
3.2.9	Повторное использование функций .....	98
3.3	<b>Дополнительные особенности функций</b> .....	99
3.3.1	Функции с несколькими аргументами? .....	99
3.3.2	Применение каррированных функций.....	100
3.3.3	Реализация функций высшего порядка.....	100
3.3.4	Полиморфные функции высшего порядка .....	102
3.3.5	Анонимные функции.....	104
3.3.6	Локальные функции .....	106
3.3.7	Замыкания.....	106
3.3.8	Частичное применение функций и автоматическое каррирование .....	107
3.3.9	Перестановка аргументов частично примененных функций .....	112
	<b>Итоги</b> .....	119
<b>4</b>	<b>Рекурсия, сорекурсия и мемоизация</b> .....	120
4.1	<b>Рекурсия и сорекурсия</b> .....	121
4.1.1	Реализация сорекурсии.....	121
4.1.2	Реализация рекурсии .....	123
4.1.3	Различия между рекурсивными и сорекурсивными функциями .....	124
4.1.4	Выбор между рекурсией и сорекурсией.....	125
4.2	<b>Удаление хвостового вызова</b> .....	127
4.2.1	Использование механизма удаления хвостового вызова .....	128
4.2.2	Преобразование циклов в хвостовую рекурсию .....	128
4.2.3	Рекурсивные функции-значения .....	132
4.3	<b>Рекурсивные функции и списки</b> .....	135
4.3.1	Дважды рекурсивные функции .....	137
4.3.2	Абстракция рекурсии в списках .....	140
4.3.3	Обращение списка .....	143
4.3.4	Сорекурсивные списки .....	145
4.3.5	Опасность строгости.....	149
4.4	<b>Мемоизация</b> .....	149
4.4.1	Мемоизация в программировании на основе циклов .....	149
4.4.2	Мемоизация рекурсивных функций .....	150
4.4.3	Неявная мемоизация .....	152
4.4.4	Автоматическая мемоизация .....	154
4.4.5	Мемоизация функций нескольких аргументов .....	157

4.5	Являются ли мемоизованные функции чистыми? .....	159
	Итоги .....	159
<b>5</b>	<b>Обработка данных с использованием списков</b> .....	161
5.1	Классификация коллекций данных .....	162
5.2	Разные типы списков .....	162
5.3	Производительность списков .....	164
5.3.1	Время в обмен на объем памяти и сложность .....	165
5.3.2	Отказ от изменения на месте .....	165
5.4	Какие виды списков доступны в Kotlin? .....	167
5.4.1	Использование постоянных структур данных .....	168
5.4.2	Реализация неизменяемого, постоянного, односвязного списка .....	168
5.5	Совместное использование данных в операциях со списками .....	172
5.6	Дополнительные операции со списками .....	174
5.6.1	Преимущества объектной нотации .....	175
5.6.2	Объединение списков .....	178
5.6.3	Удаление элементов с конца списка .....	180
5.6.4	Использование рекурсии для свертки списков с помощью функций высшего порядка .....	181
5.6.5	Вариантность .....	182
5.6.6	Безопасная рекурсивная версия <code>foldRight</code> .....	191
5.6.7	Отображение и фильтрация списков .....	193
	Итоги .....	196
<b>6</b>	<b>Необязательные данные</b> .....	197
6.1	Проблемы с пустым указателем .....	198
6.2	Как пустые ссылки обрабатываются в Kotlin .....	201
6.3	Альтернативы пустым ссылкам .....	202
6.4	Тип <code>Option</code> .....	205
6.4.1	Извлечение значения из <code>Option</code> .....	207
6.4.2	Применение функций к необязательным значениям .....	209
6.4.3	Композиция функций с типом <code>Option</code> .....	210
6.4.4	Примеры использования <code>Option</code> .....	212
6.4.5	Другие способы комбинирования типа <code>Options</code> .....	215
6.4.6	Комбинирование <code>List</code> и <code>Option</code> .....	218
6.4.7	Когда и как использовать тип <code>Option</code> .....	220
	Итоги .....	221
<b>7</b>	<b>Обработка ошибок и исключений</b> .....	222
7.1	Проблемы с отсутствующими данными .....	223
7.2	Тип <code>Either</code> .....	224
7.3	Тип <code>Result</code> .....	227

7.4	Приемы использования типа Result .....	230
7.5	Дополнительные способы использования Result .....	236
7.5.1	<i>Применение предикатов</i> .....	236
7.6	Преобразование ошибок .....	238
7.7	Дополнительные фабричные функции .....	239
7.8	Применение эффектов .....	240
7.9	Дополнительные способы комбинирования с типом Result .....	243
	Итоги .....	247
<b>8</b>	<b><i>Дополнительные операции со списками</i></b> .....	<b>248</b>
8.1	Проблемы функции length .....	249
8.2	Проблема производительности .....	249
8.3	Преимущества мемоизации .....	250
8.3.1	<i>Недостатки мемоизации</i> .....	250
8.3.2	<i>Оценка увеличения производительности</i> .....	252
8.4	Комбинирование List и Result .....	253
8.4.1	<i>Списки, возвращающие Result</i> .....	253
8.4.2	<i>Преобразование List&lt;Result&gt; в Result&lt;List&gt;</i> .....	255
8.5	Абстракции операций со списками .....	257
8.5.1	<i>Упаковка и распаковка списков</i> .....	258
8.5.2	<i>Доступ к элементам по индексам</i> .....	261
8.5.3	<i>Разбиение списков</i> .....	266
8.5.4	<i>Поиск подсписков</i> .....	270
8.5.5	<i>Разные функции для работы со списками</i> .....	271
8.6	Автоматическое распараллеливание операций со списками .....	276
8.6.1	<i>Не все вычисления могут выполняться параллельно</i> .....	276
8.6.2	<i>Деление списка на подсписки</i> .....	276
8.6.3	<i>Параллельная обработка подсписков</i> .....	278
	Итоги .....	280
<b>9</b>	<b><i>Ленивые вычисления</i></b> .....	<b>282</b>
9.1	Строгий и ленивый подходы .....	283
9.2	Строгие вычисления в Kotlin .....	284
9.3	Ленивые вычисления в Kotlin .....	286
9.4	Реализация ленивых вычислений .....	288
9.4.1	<i>Комбинирование ленивых значений</i> .....	290
9.4.2	<i>Преобразование обычных функций в ленивые</i> .....	294
9.4.3	<i>Отображение ленивых значений</i> .....	296
9.4.4	<i>Комбинирование типов Lazy и List</i> .....	298
9.4.5	<i>Обработка исключений</i> .....	299
9.5	Другие способы комбинирования ленивых вычислений .....	302

9.5.1	Ленивое применение эффектов .....	302
9.5.2	Вычисления, невозможные без ленивых значений .....	304
9.5.3	Создание ленивого списка .....	305
9.6	Работа с потоками .....	308
9.6.1	Свертка потоков .....	314
9.6.2	Трассировка вычислений и применение функций .....	317
9.6.3	Использование потоков для решения конкретных задач .....	319
	Итоги .....	322

<b>10</b>	<b>Обработка данных с использованием деревьев .....</b>	<b>323</b>
10.1	Бинарное дерево .....	324
10.2	Сбалансированные и несбалансированные деревья .....	325
10.3	Размер, высота и глубина дерева .....	325
10.4	Пустые деревья и рекурсивное определение .....	326
10.5	Лиственные деревья .....	327
10.6	Упорядоченные бинарные деревья, или бинарные деревья поиска .....	327
10.7	Порядок вставки и структура дерева .....	329
10.8	Рекурсивный и нерекурсивный обход дерева .....	330
10.8.1	Рекурсивный обход дерева .....	330
10.8.2	Нерекурсивный обход дерева .....	332
10.9	Реализация бинарного дерева поиска .....	333
10.9.1	Деревья и вариантность .....	334
10.9.2	Об абстрактных функциях в классе Tree .....	336
10.9.3	Перегрузка операторов .....	336
10.9.4	Рекурсия в деревьях .....	336
10.9.5	Удаление элементов из дерева .....	340
10.9.6	Слияние произвольных деревьев .....	341
10.10	О свертке деревьев .....	347
10.10.1	Свертка с двумя функциями .....	348
10.10.2	Свертка с одной функцией .....	350
10.10.3	Выбор реализации свертки .....	350
10.11	О преобразовании элементов деревьев .....	353
10.12	О балансировке деревьев .....	354
10.12.1	Вращение деревьев .....	354
10.12.2	Алгоритм Дея/Стоута/Уоррен .....	357
10.12.3	Самобалансирующиеся деревья .....	361
	Итоги .....	362

<b>11</b>	<b>Решение задач с использованием усовершенствованных деревьев .....</b>	<b>363</b>
11.1	Улучшение производительности и безопасности деревьев добавлением самобалансировки .....	364
11.1.1	Структура красно-черных деревьев .....	365

11.1.2	Добавление элемента в красно-черное дерево .....	367
11.1.3	Удаление элементов из красно-черного дерева .....	373
11.2	Практические примеры использования красно-черных деревьев: ассоциативные массивы .....	373
11.2.1	Реализация Map .....	373
11.2.2	Расширение ассоциативных массивов .....	376
11.2.3	Использование ключей, не поддерживающих сравнение .....	377
11.3	Реализация функциональной приоритетной очереди .....	380
11.3.1	Протоколы доступа к приоритетной очереди .....	380
11.3.2	Варианты использования приоритетных очередей .....	380
11.3.3	Требования к реализации .....	381
11.3.4	Левосторонняя куча .....	381
11.3.5	Реализация левосторонней кучи .....	382
11.3.6	Реализация интерфейса, характерного для очередей .....	385
11.4	Элементы и сортированные списки .....	386
11.5	Приоритетная очередь для несопоставимых элементов ..	388
	Итоги .....	391

<b>12</b>	<b>Функциональный ввод/вывод .....</b>	<b>392</b>
12.1.	Что означает «эффект внутри контекста»? .....	393
12.1.1	Обработка эффектов .....	394
12.1.2	Реализация эффектов .....	394
12.2	Чтение данных .....	397
12.2.1	Чтение данных с клавиатуры .....	398
12.2.2	Чтение из файла .....	402
12.3	Тестирование программ с вводом .....	404
12.4	Полностью функциональный ввод/вывод .....	405
12.4.1	Как сделать ввод/вывод полностью функциональным .....	405
12.4.2	Реализация чисто функционального ввода/вывода .....	406
12.4.3	Комбинирование ввода/вывода .....	407
12.4.4	Обработка ввода с IO .....	409
12.4.5	Расширение типа IO .....	411
12.4.6	Добавление в IO защиты от переполнения стека .....	414
	Итоги .....	419

<b>13</b>	<b>Общее изменяемое состояние и акторы .....</b>	<b>420</b>
13.1	Модель акторов .....	422
13.1.1	Асинхронный обмен сообщениями .....	422
13.1.2	Параллельное выполнение .....	422
13.1.3	Управление изменением состояния актора .....	423
13.2	Реализация инфраструктуры акторов .....	424
13.2.1	Обзор ограничений .....	425
13.2.2	Интерфейсы инфраструктуры акторов .....	425
13.3	Реализация AbstractActor .....	427

13.4	Включение акторов в работу .....	428
13.4.1	Реализация примера игры в пинг-понг.....	429
13.4.2	Параллельное выполнение вычислений .....	431
13.4.3	Переупорядочение результатов .....	437
13.4.4	Оптимизация производительности .....	440
Итого	.....	447

## **14** Решение типичных проблем функциональным способом..... 448

14.1	Утверждения и проверка данных .....	449
14.2	Повторный вызов функций и эффектов .....	453
14.3	Чтение свойств из файла .....	456
14.3.1	Загрузка файла со свойствами .....	457
14.3.2	Чтение свойств как строк .....	458
14.3.3	Вывод более информативных сообщений об ошибках .....	459
14.3.4	Чтение свойств как списков .....	462
14.3.5	Чтение значений перечислений .....	463
14.3.6	Чтение свойств произвольных типов .....	464
14.4	Преобразование императивной программы: чтение файлов XML .....	467
14.4.1	Шаг 1: императивное решение .....	468
14.4.2	Шаг 2: превращаем императивную программу в функциональную .....	470
14.4.3	Шаг 3: делаем программу еще более функциональной .....	473
14.4.4	Шаг 4: исправление проблемы с аргументами одного типа .....	477
14.4.5	Шаг 5: передача функции обработки элемента в параметре .....	478
14.4.6	Шаг 6: обработка ошибок в именах элементов .....	479
14.4.7	Шаг 7: дополнительные улучшения в прежде императивном коде .....	481
Итого	.....	483

## **Приложение А. Смешивание кода на Kotlin и Java..... 484**

Создание и управление смешанными проектами.....	485
Создание простого проекта в GRADLE .....	485
Импортирование Gradle-проекта в IntelliJ .....	487
Добавление зависимостей в проект .....	488
Создание проектов с несколькими модулями .....	488
Добавление зависимостей в проект с несколькими модулями.....	489
Вызов Java-методов из Kotlin.....	490
Использование примитивов Java .....	490
Использование числовых типов-объектов Java .....	491
Быстрый отказ со значениями null .....	492
Использование строковых типов Kotlin и Java .....	492

Преобразование других типов.....	493
Вызов Java-методов с переменным числом параметров .....	494
Управление поддержкой null в Java .....	494
Доступ к свойствам в JAVA с зарезервированными именами .....	497
Вызов контролируемых исключений .....	497
SAM-интерфейсы .....	498
Вызов Kotlin-функций из Java .....	498
Преобразование свойств Kotlin.....	498
Использование общедоступных полей Kotlin.....	499
Статические поля.....	499
Вызов функций Kotlin из методов Java.....	500
Преобразование типов Kotlin в типы Java .....	503
Типы функций .....	504
Характерные проблемы смешанных проектов на Kotlin/Java.....	504
Итоги .....	505
<b>Приложение В. Тестирование на основе свойств .....</b>	<b>507</b>
Зачем нужно тестирование на основе свойств? .....	508
Интерфейс .....	509
Тест .....	509
Что такое тестирование на основе свойств? .....	510
Абстракция и тестирование на основе свойств.....	511
Зависимости для модульного тестирования на основе свойств .....	513
Разработка тестов на основе свойств .....	514
Создание своих генераторов .....	517
Использование своих генераторов .....	518
Упрощение кода дальнейшим абстрагированием.....	522
Итоги .....	524
 Предметный указатель .....	 526