

Рауль-Габриэль Урма, Марио Фуско, Алан Майкрофт



Современный язык Java

Лямбда-выражения,
потoki и функциональное
программирование



 MANNING

Modern Java in Action

LAMBDA, STREAMS, FUNCTIONAL
AND REACTIVE PROGRAMMING

RAOUL-GABRIEL URMA, MARIO FUSCO, AND ALAN MYCROFT

2nd Edition



MANNING
SHELTER ISLAND

Рауль-Габриэль Урма,
Марио Фуско,
Алан Майкрофт

Современный язык Java

Лямбда-выражения,
потoki и функциональное
программирование



Санкт-Петербург • Москва • Екатеринбург • Воронеж
Нижний Новгород • Ростов-на-Дону • Самара • Минск

2020

Рауль-Габриэль Урма, Марио Фуско, Алан Майкрофт

Современный язык Java. Лямбда-выражения, поток и функциональное программирование

Серия «Для профессионалов»

Перевел на русский И. Пальти

Руководитель дивизиона
Руководитель проекта
Ведущий редактор
Научный редактор
Художники
Корректоры
Верстка

*Ю. Сергиенко
С. Давид
Н. Гринчик
М. Матвеев
А. Барцевич, С. Заматева
Е. Павлович, Е. Рафалюк-Бузовская
О. Богданович*

ББК 32.973.2-018.1

УДК 004.43

Урма Рауль-Габриэль, Фуско Марио, Майкрофт Алан

У69 Современный язык Java. Лямбда-выражения, поток и функциональное программирование. — СПб.: Питер, 2020. — 592 с.: ил. — (Серия «Для профессионалов»).

ISBN 978-5-4461-0997-5

Преимущество современных приложений — в передовых решениях, включающих микросервисы, реактивные архитектуры и потоковую обработку данных. Лямбда-выражения, потоки данных и долгожданная система модулей платформы Java значительно упрощают их реализацию. Пришло время повысить свою квалификацию и встретить любой вызов во всеоружии!

Книга поможет вам овладеть новыми возможностями современных дополнений, таких как API Streams и система модулей платформы Java. Откройте для себя новые подходы к конкурентности и узнайте, как концепции функциональности улучшают работу с кодом.

В этой книге:

- Новые возможности Java.
- Поток и реактивное программирование.
- Система модулей платформы Java.

16+ (В соответствии с Федеральным законом от 29 декабря 2010 г. № 436-ФЗ.)

ISBN 978-1617293566 англ.
978-5-4461-0997-5

© 2018 by Manning Publications Co. All rights reserved
© Перевод на русский язык ООО Издательство «Питер», 2020
© Издание на русском языке, оформление ООО Издательство «Питер», 2020
© Серия «Для профессионалов», 2020

Права на издание получены по соглашению с Apress. Все права защищены. Никакая часть данной книги не может быть воспроизведена в какой бы то ни было форме без письменного разрешения владельцев авторских прав.

Информация, содержащаяся в данной книге, получена из источников, рассматриваемых издательством как надежные. Тем не менее, имея в виду возможные человеческие или технические ошибки, издательство не может гарантировать абсолютную точность и полноту приводимых сведений и не несет ответственности за возможные ошибки, связанные с использованием книги. Издательство не несет ответственности за доступность материалов, ссылки на которые вы можете найти в этой книге. На момент подготовки книги к изданию все ссылки на интернет-ресурсы были действующими.

Изготовлено в России. Изготовитель: ООО «Прогресс книга». Место нахождения и фактический адрес: 194044, Россия, г. Санкт-Петербург, Б. Сампсониевский пр., д. 29А, пом. 52. Тел.: +78127037373.

Дата изготовления: 09.2019. Наименование: книжная продукция. Срок годности: не ограничен.

Налоговая льгота — общероссийский классификатор продукции ОК 034-2014,
58.11.12 — Книги печатные профессиональные, технические и научные.

Импортер в Беларусь: ООО «ПИТЕР М», 220020, РБ, г. Минск, ул. Тимирязева, д. 121/3, к. 214, тел./факс: 208 80 01.

Подписано в печать 28.08.19. Формат 70×100/16. Бумага офсетная. Усл. п. л. 47,730. Тираж 1500. Заказ 0000.

Краткое содержание

Предисловие.....	19
Благодарности.....	21
Об этой книге.....	23
Об авторах.....	29
Иллюстрация на обложке.....	31
Часть I. Основы.....	33
Глава 1. Java 8, 9, 10 и 11: что происходит?.....	35
Глава 2. Передача кода и параметризация поведения.....	60
Глава 3. Лямбда-выражения.....	77
Часть II. Функциональное программирование с помощью потоков.....	115
Глава 4. Знакомство с потоками данных.....	117
Глава 5. Работа с потоками данных.....	134
Глава 6. Сбор данных с помощью потоков.....	170
Глава 7. Параллельная обработка данных и производительность.....	208
Часть III. Эффективное программирование с помощью потоков и лямбда-выражений.....	235
Глава 8. Расширения Collection API.....	237
Глава 9. Рефакторинг, тестирование и отладка.....	253
Глава 10. Предметно-ориентированные языки и лямбда-выражения.....	277
Часть IV. Java на каждый день.....	311
Глава 11. Класс Optional как лучшая альтернатива null.....	313
Глава 12. Новый API для работы с датой и временем.....	335
Глава 13. Методы с реализацией по умолчанию.....	351
Глава 14. Система модулей Java.....	371

Часть V. Расширенная конкурентность в языке Java	393
Глава 15. Основные концепции класса <code>CompletableFuture</code> и реактивное программирование	395
Глава 16. Класс <code>CompletableFuture</code> : композиция асинхронных компонентов	428
Глава 17. Реактивное программирование.....	457
Часть VI. Функциональное программирование и эволюция языка Java	485
Глава 18. Мыслим функционально	487
Глава 19. Методики функционального программирования	503
Глава 20. Смесь ООП и ФП: сравнение Java и Scala.....	530
Глава 21. Заключение и дальнейшие перспективы Java	547
Приложения	565
Приложение А. Прочие изменения языка.....	566
Приложение Б. Прочие изменения в библиотеках.....	570
Приложение В. Параллельное выполнение нескольких операций над потоком данных	579
Приложение Г. Лямбда-выражения и байт-код JVM	588

Оглавление

Предисловие.....	19
Благодарности	21
Рауль-Габриэль Урма.....	22
Марио Фуско	22
Алан Майкрофт	22
Об этой книге	23
Структура издания	25
О коде.....	27
Форум для обсуждения книги	28
От издательства	28
Об авторах	29
Иллюстрация на обложке	31

Часть I. Основы

Глава 1. Java 8, 9, 10 и 11: что происходит?.....	35
1.1. Итак, о чем вообще речь?.....	35
1.2. Почему Java все еще меняется.....	38
1.2.1. Место языка Java в экосистеме языков программирования	39
1.2.2. Потокковая обработка.....	41
1.2.3. Передача кода в методы и параметризация поведения.....	42
1.2.4. Параллелизм и разделяемые изменяемые данные	43
1.2.5. Язык Java должен развиваться	44
1.3. Функции в Java	45
1.3.1. Методы и лямбда-выражения как полноправные граждане	46
1.3.2. Передача кода: пример	48
1.3.3. От передачи методов к лямбда-выражениям	50

1.4. Потоки данных	51
1.4.1. Многопоточность — трудная задача	52
1.5. Методы с реализацией по умолчанию и модули Java	55
1.6. Другие удачные идеи, заимствованные из функционального программирования	57
Резюме	59
Глава 2. Передача кода и параметризация поведения	60
2.1. Как адаптироваться к меняющимся требованиям	61
2.1.1. Первая попытка: фильтрация зеленых яблок	61
2.1.2. Вторая попытка: параметризация цвета	62
2.1.3. Третья попытка: фильтрация по всем возможным атрибутам	63
2.2. Параметризация поведения	64
2.2.1. Четвертая попытка: фильтрация по абстрактному критерию	65
2.3. Делаем код лаконичнее	69
2.3.1. Анонимные классы	70
2.3.2. Пятая попытка: использование анонимного класса	70
2.3.3. Шестая попытка: использование лямбда-выражения	72
2.3.4. Седьмая попытка: абстрагирование по типу значений списка	72
2.4. Примеры из практики	73
2.4.1. Сортировка с помощью интерфейса Comparator	73
2.4.2. Выполнение блока кода с помощью интерфейса Runnable	74
2.4.3. Возвращение результатов выполнения задачи с помощью интерфейса Callable	75
2.4.4. Обработка событий графического интерфейса	75
Резюме	76
Глава 3. Лямбда-выражения	77
3.1. Главное о лямбда-выражениях	78
3.2. Где и как использовать лямбда-выражения	81
3.2.1. Функциональный интерфейс	81
3.2.2. Функциональные дескрипторы	83
3.3. Практическое использование лямбда-выражений: паттерн охватывающего выполнения	85
3.3.1. Шаг 1: вспоминаем параметризацию поведения	86
3.3.2. Шаг 2: используем функциональный интерфейс для передачи поведения	86
3.3.3. Шаг 3: выполняем нужный вариант поведения!	87
3.3.4. Шаг 4: передаем лямбда-выражения	87
3.4. Использование функциональных интерфейсов	88
3.4.1. Интерфейс Predicate	89
3.4.2. Интерфейс Consumer	89
3.4.3. Интерфейс Function	90
3.5. Проверка типов, вывод типов и ограничения	94
3.5.1. Проверка типов	94
3.5.2. То же лямбда-выражение, другие функциональные интерфейсы	96

3.5.3. Вывод типов	98
3.5.4. Использование локальных переменных	98
3.6. Ссылки на методы	100
3.6.1. В общих чертах	100
3.6.2. Ссылки на конструкторы.....	103
3.7. Практическое использование лямбда-выражений и ссылок на методы.....	106
3.7.1. Шаг 1: передаем код.....	106
3.7.2. Шаг 2: используем анонимный класс	106
3.7.3. Шаг 3: используем лямбда-выражения	106
3.7.4. Шаг 4: используем ссылки на методы	107
3.8. Методы, удобные для композиции лямбда-выражений	107
3.8.1. Композиция объектов Comparator	108
3.8.2. Композиция предикатов	109
3.8.3. Композиция функций.....	109
3.9. Схожие математические идеи	111
3.9.1. Интегрирование	111
3.9.2. Переводим на язык лямбда-выражений Java 8.....	112
Резюме.....	114

Часть II. Функциональное программирование с помощью потоков

Глава 4. Знакомство с потоками данных.....	117
4.1. Что такое потоки данных.....	118
4.2. Знакомимся с потоками данных	122
4.3. Потоки данных и коллекции	125
4.3.1. Строго однократный проход	126
4.3.2. Внутренняя и внешняя итерация	127
4.4. Потоквые операции	130
4.4.1. Промежуточные операции	130
4.4.2. Завершающие операции	131
4.4.3. Работа с потоками данных.....	132
Резюме.....	133
Глава 5. Работа с потоками данных.....	134
5.1. Фильтрация	135
5.1.1. Фильтрация с помощью предиката	135
5.1.2. Фильтрация уникальных элементов	136
5.2. Срез потока данных.....	136
5.2.1. Срез с помощью предиката.....	137
5.2.2. Усечение потока данных.....	138
5.2.3. Пропуск элементов	139
5.3. Отображение	140
5.3.1. Применение функции к каждому из элементов потока данных	140
5.3.2. Схлопывание потоков данных.....	141

5.4. Поиск и сопоставление с шаблоном.....	144
5.4.1. Проверяем, удовлетворяет ли предикату хоть один элемент	144
5.4.2. Проверяем, удовлетворяют ли предикату все элементы	145
5.4.3. Поиск элемента в потоке	145
5.4.4. Поиск первого элемента	146
5.5. Свертка	147
5.5.1. Суммирование элементов	147
5.5.2. Максимум и минимум	149
5.6. Переходим к практике	153
5.6.1. Предметная область: трейдеры и транзакции.....	154
5.6.2. Решения	155
5.7. Числовые потоки данных	157
5.7.1. Версии потоков для простых типов данных.....	157
5.7.2. Числовые диапазоны	159
5.7.3. Применяем числовые потоки данных на практике: пифагоровы тройки	159
5.8. Создание потоков данных.....	162
5.8.1. Создание потока данных из перечня значений	162
5.8.2. Создание потока данных из объекта, допускающего неопределенное значение	163
5.8.3. Создание потоков данных из массивов	163
5.8.4. Создание потоков данных из файлов.....	163
5.8.5. Потоки на основе функций: создание бесконечных потоков	164
Резюме.....	169
Глава 6. Сбор данных с помощью потоков.....	170
6.1. Главное о коллекторах	172
6.1.1. Коллекторы как продвинутые средства свертки.....	172
6.1.2. Встроенные коллекторы	173
6.2. Свертка и вычисление сводных показателей	174
6.2.1. Поиск максимума и минимума в потоке данных	174
6.2.2. Вычисление сводных показателей	175
6.2.3. Объединение строк.....	176
6.2.4. Обобщение вычисления сводных показателей с помощью свертки.....	177
6.3. Группировка	181
6.3.1. Дальнейшие операции со сгруппированными элементами	182
6.3.2. Многоуровневая группировка	184
6.3.3. Сбор данных в подгруппы.....	186
6.4. Секционирование	190
6.4.1. Преимущества секционирования	190
6.4.2. Секционирование чисел на простые и составные.....	192
6.5. Интерфейс Collector	194
6.5.1. Разбираемся с объявленными в интерфейсе Collector методами	195
6.5.2. Собираем все вместе	199

6.6. Разрабатываем собственный, более производительный коллектор	201
6.6.1. Деление только на простые числа	201
6.6.2. Сравнение производительности коллекторов	206
Резюме.....	207
Глава 7. Параллельная обработка данных и производительность.....	208
7.1. Параллельные потоки данных.....	209
7.1.1. Превращаем последовательный поток данных в параллельный.....	210
7.1.2. Измерение быстродействия потока данных.....	212
7.1.3. Правильное применение параллельных потоков данных.....	217
7.1.4. Эффективное использование параллельных потоков данных.....	218
7.2. Фреймворк ветвления-объединения.....	220
7.2.1. Работа с классом RecursiveTask	220
7.2.2. Рекомендуемые практики применения фреймворка ветвления-объединения.....	224
7.2.3. Перехват работы	225
7.3. Интерфейс Spliterator	226
7.3.1. Процесс разбиения	227
7.3.2. Реализация собственного сплитератора.....	229
Резюме.....	234

Часть III. Эффективное программирование с помощью потоков и лямбда-выражений

Глава 8. Расширения Collection API.....	237
8.1. Фабрики коллекций	238
8.1.1. Фабрика списков	239
8.1.2. Фабрика множеств.....	240
8.1.3. Фабрики ассоциативных массивов	240
8.2. Работа со списками и множествами	241
8.2.1. Метод removeIf.....	242
8.2.2. Метод replaceAll	243
8.3. Работа с ассоциативными массивами	244
8.3.1. Метод forEach	244
8.3.2. Сортировка.....	244
8.3.3. Метод getOrDefault	245
8.3.4. Паттерны вычисления	246
8.3.5. Паттерны удаления	247
8.3.6. Способы замены элементов	248
8.3.7. Метод merge	248
8.4. Усовершенствованный класс ConcurrentHashMap	250
8.4.1. Свертка и поиск.....	250
8.4.2. Счетчики	251
8.4.3. Представление в виде множества	252
Резюме.....	252

Глава 9. Рефакторинг, тестирование и отладка	253
9.1. Рефакторинг с целью повышения удобочитаемости и гибкости кода	254
9.1.1. Повышаем удобочитаемость кода	254
9.1.2. Из анонимных классов — в лямбда-выражения	254
9.1.3. Из лямбда-выражений — в ссылки на методы	256
9.1.4. От императивной обработки данных до потоков	257
9.1.5. Повышаем гибкость кода	258
9.2. Рефакторинг объектно-ориентированных паттернов проектирования с помощью лямбда-выражений	260
9.2.1. Стратегия	261
9.2.2. Шаблонный метод	263
9.2.3. Наблюдатель	264
9.2.4. Цепочка обязанностей	266
9.2.5. Фабрика	268
9.3. Тестирование лямбда-выражений	270
9.3.1. Тестирование поведения видимого лямбда-выражения	270
9.3.2. Делаем упор на поведение метода, использующего лямбда-выражение	271
9.3.3. Разносим сложные лямбда-выражения по отдельным методам	272
9.3.4. Тестирование функций высшего порядка	272
9.4. Отладка	272
9.4.1. Изучаем трассу вызовов в стеке	273
9.4.2. Журналирование информации	274
Резюме	276
Глава 10. Предметно-ориентированные языки и лямбда-выражения	277
10.1. Специальный язык для конкретной предметной области	279
10.1.1. За и против предметно-ориентированных языков	280
10.1.2. Доступные на JVM решения, подходящие для создания DSL	282
10.2. Малые DSL в современных API Java	286
10.2.1. Stream API как DSL для работы с коллекциями	287
10.2.2. Коллекторы как предметно-ориентированный язык агрегирования данных	289
10.3. Паттерны и методики создания DSL на языке Java	290
10.3.1. Связывание методов цепочкой	293
10.3.2. Вложенные функции	295
10.3.3. Задание последовательности функций с помощью лямбда-выражений	297
10.3.4. Собираем все воедино	300
10.3.5. Использование ссылок на методы в DSL	302
10.4. Реальные примеры DSL Java 8	304
10.4.1. jOOQ	305
10.4.2. Cucumber	306
10.4.3. Фреймворк Spring Integration	308
Резюме	310

Часть IV. Java на каждый день

Глава 11. Класс Optional как лучшая альтернатива null	313
11.1. Как смоделировать отсутствие значения.....	314
11.1.1. Снижение количества исключений NullPointerException с помощью проверки на безопасность	315
11.1.2. Проблемы, возникающие с null	316
11.1.3. Альтернативы null в других языках программирования	316
11.2. Знакомство с классом Optional	318
11.3. Паттерны для внедрения в базу кода опционалов.....	319
11.3.1. Создание объектов Optional	319
11.3.2. Извлечение и преобразование значений опционалов с помощью метода map	320
11.3.3. Связывание цепочкой объектов Optional с помощью метода flatMap.....	321
11.3.4. Операции над потоком опционалов	325
11.3.5. Действия по умолчанию и распаковка опционалов	326
11.3.6. Сочетание двух опционалов.....	327
11.3.7. Отбрасывание определенных значений с помощью метода filter	328
11.4. Примеры использования опционалов на практике	330
11.4.1. Обертывание потенциально пустого значения в опционал.....	331
11.4.2. Исключения или опционалы?.....	331
11.4.3. Версии опционалов для простых типов данных и почему их лучше не использовать	332
11.4.4. Собираем все воедино	332
Резюме.....	334
Глава 12. Новый API для работы с датой и временем.....	335
12.1. Классы LocalDate, LocalTime, LocalDateTime, Instant, Duration и Period.....	336
12.1.1. Работа с классами LocalDate и LocalTime.....	336
12.1.2. Сочетание даты и времени	338
12.1.3. Класс Instant: дата и время для компьютеров	338
12.1.4. Описание продолжительности и промежутков времени	339
12.2. Операции над датами, синтаксический разбор и форматирование дат.....	341
12.2.1. Работа с классом TemporalAdjusters.....	343
12.2.2. Вывод в консоль и синтаксический разбор объектов даты/времени	345
12.3. Различные часовые пояса и системы летоисчисления.....	347
12.3.1. Использование часовых поясов.....	347
12.3.2. Фиксированное смещение от UTC/времени по Гринвичу	348
12.3.3. Использование альтернативных систем летоисчисления.....	349
Резюме.....	350

Глава 13. Методы с реализацией по умолчанию.....	351
13.1. Эволюция API	354
13.1.1. Версия 1 API	355
13.1.2. Версия 2 API	355
13.2. Коротко о методах с реализацией по умолчанию	357
13.3. Примеры использования методов с реализацией по умолчанию	360
13.3.1. Необязательные методы	360
13.3.2. Множественное наследование поведения	360
13.4. Правила разрешения конфликтов	364
13.4.1. Три важных правила разрешения неоднозначностей	365
13.4.2. Преимущество — у самого конкретного интерфейса из числа содержащих реализацию по умолчанию	365
13.4.3. Конфликты и разрешение неоднозначностей явным образом.....	367
13.4.4. Проблема ромбовидного наследования.....	368
Резюме.....	370
Глава 14. Система модулей Java.....	371
14.1. Движущая сила появления системы модулей Java: размышления о ПО	372
14.1.1. Разделение ответственности.....	372
14.1.2. Скрытие информации	372
14.1.3. Программное обеспечение на языке Java	373
14.2. Причины создания системы модулей Java	374
14.2.1. Ограничения модульной организации	374
14.2.2. Монолитный JDK.....	376
14.2.3. Сравнение с OSGi.....	376
14.3. Модули Java: общая картина.....	377
14.4. Разработка приложения с помощью системы модулей Java	378
14.4.1. Подготовка приложения	378
14.4.2. Мелкоблочная и крупноблочная модуляризация	381
14.4.3. Основы системы модулей Java	381
14.5. Работа с несколькими модулями	382
14.5.1. Выражение exports	383
14.5.1. Выражение requires.....	383
14.5.3. Именованное	384
14.6. Компиляция и компоновка	385
14.7. Автоматические модули	388
14.8. Объявление модуля и выражения	389
14.8.1. Выражение requires.....	389
14.8.2. Выражение exports	389
14.8.3. Выражение requires transitive	390
14.8.4. Выражение exports ... to.....	390
14.8.5. Выражения open и opens	390
14.8.6. uses и provides	391
14.9. Пример побольше и дополнительные источники информации	391
Резюме.....	392

Часть V. Расширенная конкурентность в языке Java

Глава 15. Основные концепции класса <code>CompletableFuture</code> и реактивное программирование	395
15.1. Эволюция поддержки конкурентности в Java	398
15.1.1. Потоки выполнения и высокоуровневые абстракции	399
15.1.2. Исполнители и пулы потоков выполнения	401
15.1.3. Другие абстракции потоков выполнения: (не) вложенность в вызовы методов	403
15.1.4. Что нам нужно от потоков выполнения	405
15.2. Синхронные и асинхронные API	406
15.2.1. Фьючерсный API	408
15.2.2. Реактивный API	408
15.2.3. Приостановка и другие блокирующие операции вредны	410
15.2.4. Смотрим правде в глаза	412
15.2.5. Исключения и асинхронные API	412
15.3. Модель блоков и каналов	413
15.4. Реализация конкурентности с помощью класса <code>CompletableFuture</code> и комбинаторов	415
15.5. Публикация/подписка и реактивное программирование	419
15.5.1. Пример использования для суммирования двух потоков сообщений	420
15.5.2. Противодействие	425
15.5.3. Простой вариант реализации противодействия на практике	425
15.6. Реактивные системы и реактивное программирование	426
Резюме	427
Глава 16. Класс <code>CompletableFuture</code>: композиция асинхронных компонентов	428
16.1. Простые применения фьючерсов	429
16.1.1. Разбираемся в работе фьючерсов и их ограничениях	430
16.1.2. Построение асинхронного приложения с помощью завершаемых фьючерсов	431
16.2. Реализация асинхронного API	432
16.2.1. Преобразование синхронного метода в асинхронный	433
16.2.2. Обработка ошибок	435
16.3. Делаем код неблокирующим	437
16.3.1. Распараллеливание запросов с помощью параллельного потока данных	438
16.3.2. Выполнение асинхронных запросов с помощью завершаемых фьючерсов	438
16.3.3. Поиск лучше масштабируемого решения	441
16.3.4. Пользовательский исполнитель	442
16.4. Конвейерная организация асинхронных задач	444
16.4.1. Реализация сервиса скидок	444
16.4.2. Использование скидочного сервиса	446
16.4.3. Композиция синхронных и асинхронных операций	446

16.4.4. Сочетание двух завершаемых фьючерсов: зависимого и независимого	449
16.4.5. Сравниваем фьючерсы с завершаемыми фьючерсами.....	450
16.4.6. Эффективное использование тайм-аутов	452
16.5. Реагирование на завершение <code>CompletableFuture</code>	453
16.5.1. Рефакторинг приложения для поиска наилучшей цены	454
16.5.2. Собираем все вместе	455
Резюме.....	456
Глава 17. Реактивное программирование.....	457
17.1. Манифест реактивности.....	458
17.1.1. Реактивность на прикладном уровне.....	460
17.1.2. Реактивность на системном уровне	461
17.2. Реактивные потоки данных и <code>Flow API</code>	462
17.2.1. Знакомство с классом <code>Flow</code>	463
17.2.2. Создаем ваше первое реактивное приложение	466
17.2.3. Преобразование данных с помощью интерфейса <code>Processor</code>	471
17.2.4. Почему Java не предоставляет реализации <code>Flow API</code>	473
17.3. Реактивная библиотека <code>RxJava</code>	474
17.3.1. Создание и использование наблюдаемых объектов.....	475
17.3.2. Преобразование и группировка наблюдаемых объектов	480
Резюме.....	484
 Часть VI. Функциональное программирование и эволюция языка Java	
Глава 18. Мыслим функционально	487
18.1. Создание и сопровождение систем	488
18.1.1. Разделяемые изменяемые данные	488
18.1.2. Декларативное программирование	490
18.1.3. Почему функциональное программирование?	491
18.2. Что такое функциональное программирование	491
18.2.1. Функциональное программирование на языке Java	492
18.2.2. Функциональная прозрачность	494
18.2.3. Объектно-ориентированное и функциональное программирование ..	495
18.2.4. Функциональное программирование в действии	496
18.3. Рекурсия и итерация.....	498
Резюме.....	502
Глава 19. Методики функционального программирования	503
19.1. Повсюду функции	503
19.1.1. Функции высшего порядка.....	504
19.1.2. Каррирование.....	506
19.2. Персистентные структуры данных.....	507
19.2.1. Деструктивные и функциональные обновления	508

19.2.2. Другой пример, с деревьями.....	510
19.2.3. Функциональный подход.....	511
19.3. Отложенное вычисление с помощью потоков данных	513
19.3.1. Самоопределяющиеся потоки данных.....	513
19.3.2. Наш собственный отложенный список	516
19.4. Сопоставление с шаблоном.....	521
19.4.1. Паттерн проектирования «Посетитель»	521
19.4.2. На помощь приходит сопоставление с шаблоном	522
19.5. Разное	525
19.5.1. Кэширование или мемоизация	525
19.5.2. Что значит «вернуть тот же самый объект»?	527
19.5.3. Комбинаторы	527
Резюме.....	529
Глава 20. Смесь ООП и ФП: сравнение Java и Scala	530
20.1. Введение в Scala	531
20.1.1. Приложение Hello beer.....	531
20.1.2. Основные структуры данных Scala: List, Set, Map, Stream, Tuple и Option	533
20.2. Функции	538
20.2.1. Полноправные функции.....	539
20.2.2. Анонимные функции и замыкания.....	540
20.2.3. Каррирование.....	542
20.3. Классы и типы.....	543
20.3.1. Код с классами Scala становится лаконичнее	543
20.3.2. Типы Scala и интерфейсы Java	544
Резюме.....	546
Глава 21. Заключение и дальнейшие перспективы Java	547
21.1. Обзор возможностей Java 8	547
21.1.1. Параметризация поведения (лямбда-выражения и ссылки на методы).....	548
21.1.2. Потоки данных.....	549
21.1.3. Класс CompletableFuture.....	549
21.1.4. Класс Optional	550
21.1.5. Flow API.....	551
21.1.6. Методы с реализацией по умолчанию	551
21.2. Система модулей Java 9	551
21.3. Вывод типов локальных переменных в Java 10	553
21.4. Что ждет Java в будущем?.....	554
21.4.1. Вариантность по месту объявления	554
21.4.2. Сопоставление с шаблоном.....	555
21.4.3. Более полнофункциональные формы обобщенных типов.....	556

21.4.4. Расширение поддержки неизменяемости	558
21.4.5. Типы-значения	559
21.5. Ускорение развития Java.....	562
21.6. Заключительное слово.....	564

Приложения

Приложение А. Прочие изменения языка.....	566
А.1. Аннотации	566
А.1.1. Повторяющиеся аннотации	567
А.1.2. Аннотации типов	568
А.2. Обобщенный вывод целевых типов	569
Приложение Б. Прочие изменения в библиотеках.....	570
Б.1. Коллекции	570
Б.1.1. Добавленные методы	570
Б.1.2. Класс Collections	572
Б.1.3. Интерфейс Comparator	572
Б.2. Конкурентность	572
Б.2.1. Пакет Atomic	573
Б.2.2. ConcurrentHashMap.....	574
Б.3. Массивы	575
Б.3.1. Использование метода parallelSort	575
Б.3.2. Использование методов setAll и parallelSetAll	576
Б.3.3. Использование метода parallelPrefix.....	576
Б.4. Классы Number и Math.....	576
Б.4.1. Класс Number	576
Б.4.2. Класс Math	577
Б.5. Класс Files	577
Б.6. Рефлексия.....	577
Б.7. Класс String	578
Приложение В. Параллельное выполнение нескольких операций над потоком данных	579
В.1. Ветвление потока данных.....	580
В.1.1. Реализация интерфейса Results на основе класса ForkingStreamConsumer	581
В.1.2. Разработка классов ForkingStreamConsumer и BlockingQueueSpliterator	583
В.1.3. Применяем StreamForker на практике	585
В.2. Вопросы производительности.....	587
Приложение Г. Лямбда-выражения и байт-код JVM	588
Г.1. Анонимные классы.....	588
Г.2. Генерация байт-кода	589
Г.3. Invokedynamic спешит на помощь	590
Г.4. Стратегии генерации кода	591

Предисловие

В 1998 году, когда мне было всего восемь лет, я открыл свою первую книгу по программированию — она была посвящена JavaScript и HTML. Я и не подозревал, что это простое действие перевернет всю мою жизнь — я открыл для себя мир языков программирования и всех тех потрясающих вещей, которые можно сделать с их помощью. Я попался на крючок. Время от времени я по-прежнему узнаю о новых возможностях языков программирования, которые воскрешают во мне то чувство восхищения, поскольку позволяют писать более аккуратный и лаконичный код, причем вдвое быстрее. Надеюсь, обсуждаемые в данной книге нововведения Java версий 8, 9 и 10, привнесенные из функционального программирования, будут так же вдохновлять вас.

Наверное, вам интересно, как появилась эта книга вообще и второе ее издание в частности?

Что ж, в 2011 году Брайан Гётц (Brian Goetz) — архитектор языка Java в Oracle — публиковал различные предложения по добавлению лямбда-выражений в Java, желая вовлечь в этот процесс сообщество разработчиков. Они вновь разожгли мой интерес, и я начал продвигать эти идеи, организуя семинары по Java 8 на различных конференциях разработчиков, а также читать лекции студентам Кембриджского университета.

К апрелю 2013 года мой издатель из Manning прислал мне по электронной почте письмо, где спросил, не хочу ли я написать книгу о лямбда-выражениях в Java 8. На тот момент я был всего лишь скромным соискателем степени PhD на втором году обучения, и это предложение показалось мне несвоевременным, поскольку могло негативно отразиться на моем графике подачи диссертации. С другой стороны, *carpe diem*¹. Мне казалось, что написание небольшой книги требует не так уж

¹ В переводе с лат. «лови день». — *Примеч. пер.*

много времени (и только позднее я понял, как сильно ошибался!). В итоге я обратился за советом к своему научному руководителю профессору Алану Майкрофту (Alan Mycroft), который, как оказалось, был не против поддержать меня в такой авантюре (даже предложил помочь с этой не относящейся к моей диссертации работой, за что я навеки его должник). А несколько дней спустя мы встретили знакомого евангелиста Java 8 Марио Фуско (Mario Fusco), обладавшего богатым опытом и хорошо известного своими докладами по функциональному программированию на крупных конференциях разработчиков.

Мы быстро поняли, что, объединив наши усилия и столь разнородный опыт, можем написать не просто небольшую книгу по лямбда-выражениям Java 8, а книгу, которая, надеемся, будет полезной сообществу Java-разработчиков и через пять и даже десять лет. У нас появилась уникальная возможность подробно обсудить множество вопросов, важных для Java-разработчиков, и распахнуть двери в совершенно новую вселенную: функциональное программирование.

Наступил 2018 год, и оказалось, что было продано 20 000 экземпляров первого издания, версия Java 9 только появилась, вот-вот должна была появиться версия Java 10, и время притупило воспоминания о множестве долгих ночей редактирования книги. Итак, вот оно — второе издание «*Современного языка Java*», охватывающее Java 8, Java 9 и Java 10! Надеемся, оно вам понравится!

*Рауль-Габриэль Урма (Raoul-Gabriel Urma),
Cambridge Spark*

Благодарности

Эта книга не появилась бы на свет без поддержки множества замечательных людей, в числе которых:

- ❑ наши близкие друзья и просто люди, на добровольных началах дававшие ценные отзывы и предложения: Ричард Уолкер (Richard Walker), Ян Сагановски (Jan Saganowski), Брайан Гётц (Brian Goetz), Стюарт Маркс (Stuart Marks), Джем Редиф (Cem Redif), Пол Сандоз (Paul Sandoz), Стивен Колбурн (Stephen Colebourne), Иньиго Медиавилла (Íñigo Mediavilla), Аллабакш Асадулла (Allahbaksh Asadullah), Томаш Нуркевич (Tomasz Nurkiewicz) и Михаэль Мюллер (Michael Müller);
- ❑ участники программы раннего доступа от издательства Manning (MEAP), публиковавшие свои комментарии на онлайн-форуме автора;
- ❑ рецензенты, дававшие полезные отзывы во время написания книги: Антонио Маньяни (Antonio Magnaghi), Brent Стэйнс (Brent Stains), Франциска Мейер (Franziska Meyer), Фуркан Камачи (Furkan Kamachi), Джейсон Ли (Jason Lee), Джорн Динкла (Jörn Dinkla), Лочана Меникараччи (Lochana Menikarachchi), Маюр Патил (Mayur Patil), Николаос Кайнтанцис (Nikolaos Kaintantzis), Симоне Борде (Simone Bordet), Стив Роджерс (Steve Rogers), Уилл Хейворт (Will Hayworth) и Уильям Уилер (William Wheeler);
- ❑ Кевин Харрелд (Kevin Harreld) — наш редактор-консультант из издательства Manning, терпеливо отвечавший на все наши вопросы и решавший наши проблемы. Он подробно консультировал нас при написании книги и вообще поддерживал, как только мог;
- ❑ Дэннис Селинджер (Dennis Selinger) и Жан-Франсуа Морен (Jean-François Morin), вычитавшие рукопись перед сдачей в верстку, а также Эл Шерер (Al Scherer), консультировавший нас по техническим вопросам во время работы над книгой.

Рауль-Габриэль Урма

Прежде всего я хотел бы поблагодарить родителей за их бесконечную любовь и поддержку на протяжении всей моей жизни. Моя маленькая мечта о написании книги стала реальностью! Кроме того, хотелось бы выразить бесконечную признательность Алану Майкрофту, моему соавтору и научному руководителю, за его доверие и поддержку. Хотел бы также поблагодарить моего соавтора Марио Фуско, разделившего со мной это приключение. Наконец, спасибо моим друзьям и наставникам за их полезные советы и моральную поддержку: Софии Дроссопулу (Sophia Drossopoulou), Эйдену Рошу (Aidan Roche), Алексу Бакли (Alex Buckley), Хаади Джабадо (Haadi Jabado) и Гаспару Робертсону (Jaspar Robertson). Вы крутые!

Марио Фуско

Я хотел бы в первую очередь поблагодарить свою жену Марилену, благодаря безграничному терпению которой мне удалось сосредоточиться на написании этой книги, и нашу дочь Софию, поскольку создаваемый ею бесконечный хаос помогал мне творчески отвлекаться от книги. Как вы узнаете во время чтения, София преподавала нам один урок — о различиях между внутренней и внешней итерацией, — как это может сделать только двухлетняя девочка. Хотел бы также поблагодарить Рауля-Габриэля Урму и Алана Майкрофта, с которыми я разделил (большую) радость и (маленькие) огорчения от написания этой книги.

Алан Майкрофт

Я хотел бы выразить благодарность моей жене Хилари и остальным членам моей семьи, терпеливо выносившим мою постоянную занятость. Я хотел бы также поблагодарить своих коллег и студентов, которые на протяжении многих лет учили меня, как нужно учить, Марио и Рауля за эффективное соавторство и, в частности, Рауля — за умение столь обходительно требовать «следующий кусочек текста к пятнице».

Об этой книге

Грубо говоря, появление новых возможностей Java 8 вместе с менее явными изменениями в Java 9 — самые масштабные изменения в языке Java за 21 год, прошедший с момента выпуска Java 1.0. Ничего не было удалено, так что весь существующий код на Java сохранит работоспособность, но новые возможности предлагают мощные новые идиомы и паттерны проектирования, благодаря которым код становится чище и лаконичнее. Сначала у вас может возникнуть мысль: «Ну зачем они опять меняют мой язык?» Но потом, после начала работы с новыми функциями, вы обнаружите, что благодаря им пишете более краткий и чистый код вдвое быстрее, чем раньше, — и осознаете, что уже не хотите возвращаться к «старому Java».

Второе издание этой книги, «Современный язык Java. Лямбда-выражения, потоки и функциональное программирование», рассчитано на то, чтобы помочь вам преодолеть этап «в принципе, звучит неплохо, но все такое новое и непривычное» и начать ориентироваться в новых функциях как рыба в воде.

«Может, и так, — наверное, думаете вы, — но разве лямбды и функциональное программирование не из того разряда вещей, о которых бородатые теоретики рассуждают в своих башнях из слоновой кости?» Возможно, но в Java 8 предусмотрен ровно такой баланс идей, чтобы извлечь выгоду способами, понятными обычному Java-программисту. И эта книга как раз описывает Java с точки зрения обычного программиста, лишь с редкими отступлениями в стиле «Откуда это взялось?».

«Лямбда-выражения» — звучит как какая-то тарабарщина! Да, но они позволяют писать лаконичные программы на языке Java. Многие из вас сталкивались с обработчиками событий и функциями обратного вызова: когда регистрируется объект, содержащий метод, который должен вызываться в случае какого-либо события. Лямбда-выражения значительно расширяют сферу использования подобных

функций в Java. Попросту говоря, лямбда-выражения и их спутники — ссылки на методы — обеспечивают возможность максимально лаконичной их передачи в качестве аргументов для выполнения кода или методов без отрыва от какой-либо совершенно иной деятельности. В книге вы увидите, что эта идея встречается гораздо чаще, чем вы могли себе представить: от простой параметризации метода сортировки кодом для сравнения и до сложных запросов к коллекциям данных с помощью новых Stream API (API обработки потоков данных).

«Потоки данных — что это такое?» Это замечательная новая возможность Java 8. Потоки ведут себя подобно коллекциям, но обладают несколькими преимуществами перед последними, позволяя использовать новые стили программирования. Во-первых, если вам доводилось программировать на языках запросов баз данных, например SQL, то вы знаете, что они позволяют описывать запросы несколькими строками вместо множества строк на Java. Поддержка потоков данных в Java 8 дает возможность использовать подобный сжатый код в стиле запросов баз данных — но с синтаксисом Java и без необходимости каких-либо знаний о базах данных! Во-вторых, потоки устроены так, что их данные не обязательно должны находиться в памяти (или даже обрабатываться) одновременно. А значит, появляется возможность обработки потоков данных настолько больших, что они не помещаются в памяти компьютера. Кроме того, Java 8 умеет оптимизировать операции над потоками данных лучше, чем операции над коллекциями: например, может группировать несколько операций над одним потоком так, что данные требуется обходить только один раз, а не несколько. Более того, Java может автоматически параллелизовать потоковые операции (в отличие от операций над коллекциями).

«А функциональное программирование? Что это такое?» Это еще один стиль программирования, как и объектно-ориентированное программирование, но ориентированный на применение функций в качестве значений, как мы упоминали выше, говоря про лямбда-выражения.

Версия Java 8 хороша тем, что включает в привычный синтаксис Java множество замечательных идей из функционального программирования. Благодаря удачным проектным решениям можно рассматривать функциональное программирование в Java 8 как дополнительный набор паттернов проектирования и идиом, позволяющих быстрее писать более чистый и лаконичный код. Можете рассматривать его как расширение своего арсенала разработчика.

И да, помимо этих функциональных возможностей, основанных на глобальных концептуальных новшествах в Java, мы расскажем о множестве других удобных функций и новинок Java 8, например о методах с реализацией по умолчанию, новых классах `Optional` и `CompletableFuture`, а также о новом API для работы с датой и временем (Date and Time API).

Вдобавок Java 9 привносит еще несколько новшеств: новую систему модулей, поддержку реактивного программирования с помощью Flow API и другие разнообразные усовершенствования.

Но погодите, это же только введение, дальнейшие подробности вы узнаете из самой книги.

Структура издания

Книга делится на шесть частей: «Основы», «Функциональное программирование с помощью потоков», «Эффективное программирование с помощью потоков и лямбда-выражений», «Java на каждый день», «Расширенная конкурентность в языке Java» и «Функциональное программирование и эволюция языка Java». Мы настоятельно советуем вам прочитать сначала главы из первых двух частей (причем по порядку, поскольку многие из обсуждаемых понятий основываются на материале предыдущих глав), оставшиеся же четыре части можно читать относительно независимо друг от друга. В большинстве глав есть несколько контрольных заданий для лучшего усвоения вами материала.

В первой части книги приводятся основные сведения, необходимые для знакомства с новыми идеями, появившимися в Java 8. К концу первой части вы будете знать, что такое лямбда-выражения, и научитесь писать код, лаконичный и в то же время достаточно гибкий для адаптации к меняющимся требованиям.

- ❑ В главе 1 мы резюмируем основные изменения в Java (лямбда-выражения, ссылки на методы, потоки данных и методы с реализацией по умолчанию) и подготовим фундамент для остальной части книги.
- ❑ Из главы 2 вы узнаете о параметризации поведения — важном для Java 8 паттерне разработки программного обеспечения, лежащем в основе лямбда-выражений.
- ❑ Глава 3 подробно объясняет понятия лямбда-выражений и ссылок на методы, здесь приводятся примеры кода и контрольные задания.

Вторая часть книги представляет собой углубленное исследование нового Stream API, позволяющего писать мощный код для декларативной обработки коллекций данных. К концу второй части вы полностью разберетесь, что такое потоки данных и как их использовать в своем коде для лаконичной и эффективной обработки коллекций данных.

- ❑ В главе 4 вы познакомитесь с понятием потока данных и увидите, чем он отличается от коллекции.
- ❑ Глава 5 подробно описывает потоковые операции, с помощью которых можно выражать сложные запросы для обработки данных. Вы встретите в ней множество паттернов, касающихся фильтрации, срезов, сопоставления с шаблоном, отображения и свертки.
- ❑ Глава 6 посвящена сборщикам данных — функциональной возможности Stream API, с помощью которой можно выражать еще более сложные запросы для обработки данных.
- ❑ Из главы 7 вы узнаете, как организовать автоматическое распараллеливание потоков данных и в полной мере использовать многоядерную архитектуру своей машины. Кроме того, мы расскажем вам, как избежать многочисленных подводных камней и применять параллельные потоки правильно и эффективно.

В третьей части книги рассмотрены различные функции Java 8 и Java 9, которые помогут вам эффективнее использовать язык и обогатить вашу базу кода

современными идиомами. Поскольку мы нацелены на использование более продвинутых концепций программирования, то для понимания изложенного далее в книге материала не требуется знаний этих методик.

- ❑ Глава 8 написана специально для второго издания, в ней мы изучаем коллекцию API дополнений (Collection API Enhancements) Java 8 и Java 9. Глава охватывает вопросы использования фабрик коллекции и новых идиоматических паттернов для работы со списками и множествами (коллекциями `List` и `Set`), а также идиоматических паттернов для ассоциативных массивов (коллекций `Map`).
- ❑ Глава 9 посвящена вопросам усовершенствования существующего кода с помощью новых возможностей Java 8 и включает несколько полезных примеров. Кроме того, в ней мы исследуем такие жизненно важные методики разработки программного обеспечения, как применение паттернов проектирования, рефакторинг, тестирование и отладка.
- ❑ Глава 10 также написана специально для второго издания. В ней обсуждается использование API на основе предметно-ориентированного языка (DSL). Это многообещающий метод проектирования API, популярность которого неуклонно растет. Его уже можно встретить в таких интерфейсах Java, как `Comparator`, `Stream` и `Collector`.

Четвертая часть книги посвящена различным новым возможностям Java 8 и Java 9, касающимся упрощения написания кода и повышения его надежности. Мы начнем с двух API, появившихся в Java 8.

- ❑ В главе 11 описывается класс `java.util.Optional`, позволяющий проектировать более совершенные API, а заодно и снизить количество исключений, связанных с указателями на `null`.
- ❑ Глава 12 посвящена изучению Date and Time API (API даты и времени), который существенно лучше предыдущих API для работы с датами/временем, подверженных частым ошибкам.
- ❑ В главе 13 вы прочитаете, что такое методы с реализацией по умолчанию, как развивать с их помощью API с сохранением совместимости, а также познакомиться с некоторыми паттернами и узнаете правила эффективного использования методов с реализацией по умолчанию.
- ❑ Глава 14 написана специально для второго издания, в ней изучается система модулей Java — масштабное нововведение Java 9, позволяющее разбивать огромные системы на хорошо документированные модули вместо «беспорядочного набора пакетов».

В пятой части книги исследуются способы структурирования параллельных программ на языке Java — более продвинутые, чем простая параллельная обработка потоков данных, с которой вы к тому времени уже познакомитесь в главах 6 и 7.

- ❑ Глава 15 появилась во втором издании и демонстрирует идею асинхронных API «с высоты птичьего полета» — включая понятие будущего (Future) и протокол публикации-подписки (Publish-Subscribe), лежащий в основе реактивного программирования и включенный в Flow API Java 9.

- ❑ Глава 16 посвящена классу `CompletableFuture`, с помощью которого можно выражать сложные асинхронные вычисления декларативным образом — аналогично `Stream API`.
- ❑ Глава 17 тоже появилась только во втором издании, в ней подробно изучается `Flow API Java 9` с упором на пригодный для практического использования реактивный код.

В шестой, заключительной, части этой книги мы немного отступим от темы, чтобы предложить вам руководство по написанию эффективных программ в стиле функционального программирования на языке Java, а также сравнить возможности Java 8 с возможностями языка Scala.

- ❑ Глава 18 предлагает полное руководство по функциональному программированию, знакомит с его терминологией и объясняет, как писать программы в функциональном стиле на языке Java.
- ❑ Глава 19 охватывает более продвинутые методики функционального программирования, включая каррирующие неизменяемые структуры данных, ленивые списки и сопоставление с шаблоном. Материал этой главы представляет собой смесь практических методик, которые можно использовать в своем коде, и теоретической информации, направленной на расширение ваших познаний как программиста.
- ❑ Глава 20 продолжает тему сравнения возможностей Java 8 с возможностями языка Scala — языка, который, как и Java, основан на использовании JVM. Scala быстро развивается, угрожая занять часть ниши Java в экосистеме языков программирования.
- ❑ В главе 21 мы еще раз обсудим наш путь изучения Java 8 и свою аргументацию в пользу функционального программирования. В дополнение мы поговорим о возможных будущих усовершенствованиях и замечательных новых возможностях Java, которые могут появиться после Java 8, Java 9 и небольших добавлений в Java 10.

Наконец, в книге есть четыре приложения, охватывающие несколько дополнительных тем, связанных с Java 8. В приложении А рассматриваются несколько небольших функций языка Java 8, которые не обсуждаются в основной части книги. В приложении Б приведен обзор остальных важных дополнений библиотеки Java, которые могут быть вам полезны. Приложение В продолжает часть II и посвящено продвинутым возможностям использования потоков данных. Приложение Г изучает вопросы реализации лямбда-выражений внутри компилятора Java.

О коде

Весь исходный код в этой книге, как в листингах, так и в представленных фрагментах, набран таким моноширинным шрифтом, позволяющим отличить его от остального текста. Большинство листингов снабжено пояснениями, подчеркивающими важные понятия.

Все примеры из книги и инструкции по их запуску доступны в репозитории GitHub (<https://github.com/java-manning/modern-java>), а также на сайте издателя по адресу <http://www.manning.com/books/modern-java-in-action>.

Форум для обсуждения книги

На странице <https://forums.manning.com/forums/modern-java-in-action> вы найдете информацию о том, как попасть на форум издательства Manning после регистрации. Там можно оставлять свои комментарии по поводу данной книги, задавать технические вопросы и получать помощь от авторов и других пользователей. Узнать больше о форумах издательства Manning и правилах поведения на них можно на странице <https://forums.manning.com/forums/about>.

Издательство Manning взяло на себя обязательство по предоставлению места, где читатели могут конструктивно пообщаться как друг с другом, так и с авторами книги. Но оно не может гарантировать присутствия на форуме авторов, участие которых в обсуждениях остается добровольным (и неоплачиваемым). Мы советуем вам задавать авторам интересные и трудные вопросы, чтобы их интерес не угас! Как форум, так и архивы предыдущих обсуждений будут доступны на сайте издательства до тех пор, пока книга находится в продаже.

От издательства

Ваши замечания, предложения, вопросы отправляйте по адресу comp@piter.com (издательство «Питер», компьютерная редакция).

Мы будем рады узнать ваше мнение!

На веб-сайте издательства www.piter.com вы найдете подробную информацию о наших книгах.

Об авторах



Рауль-Габриэль Урма — генеральный директор и один из основателей компании Cambridge Spark (Великобритания), ведущего образовательного сообщества для исследователей данных и разработчиков. Рауль был избран одним из участников программы Java Champions в 2017 году. Он работал в компаниях Google, eBay, Oracle и Goldman Sachs. Защитил диссертацию по вычислительной технике в Кембриджском университете. Кроме того, он имеет диплом магистра технических наук Имперского колледжа Лондона, окончил его с отличием и получил несколько премий за рационализаторские предложения. Рауль представил более 100 технических докладов на международных конференциях.



Марио Фуско — старший инженер-разработчик программного обеспечения в компании Red Hat, участвующий в разработке ядра Drools — механизма обработки правил JBoss. Обладает богатым опытом разработки на языке Java, участвовал (зачастую в качестве ведущего разработчика) во множестве корпоративных проектов в различных отраслях промышленности, начиная от СМИ и заканчивая финансовым сектором. В числе его интересов функциональное программирование и предметно-ориентированные языки. На основе этих двух увлечений он создал библиотеку `lambdaj` с открытым исходным кодом, желая разработать внутренний DSL Java для работы с коллекциями и сделать возможным применение некоторых элементов функционального программирования в Java.



Алан Майкрофт — профессор на факультете компьютерных наук Кембриджского университета, где он преподает с 1984 года. Он также сотрудник колледжа Робинсона, один из основателей Европейской ассоциации языков и систем программирования, а также один из основателей и попечителей Raspberry Pi Foundation. Имеет ученые степени в области математики (Кембридж) и компьютерных наук (Эдинбург). Алан — автор более 100 научных статей. Был научным руководителем для более 20 кандидатов на соискание PhD. Его исследования в основном касаются сферы языков програм-

мирования и их семантики, оптимизации и реализации. Какое-то время он работал в AT&T Laboratories и научно-исследовательском отделе Intel, а также участвовал в основании компании Codemist Ltd., выпустившей компилятор языка C для архитектуры ARM под названием Norcroft.

Иллюстрация на обложке

Рисунок на обложке называется «Одеяния военного мандарина в Китайской Тартарии, 1700 г.». Одеяния мандарина причудливо разукрашены, он носит меч, а также лук и колчан со стрелами. Если присмотреться к его поясу, можно заметить там греческую букву «лямбда» (намек на один из рассматриваемых в этой книге вопросов), аккуратно добавленную нашим дизайнером. Эта иллюстрация позаимствована из четырехтомника Томаса Джеффериса (Thomas Jefferys) *A Collection of the Dresses of Different Nations, Ancient and Modern* («Коллекция платья разных народов, старинного и современного»), изданного в Лондоне в 1757–1772 годах. На титульном листе говорится, что это гравюра, раскрашенная вручную с применением гуммиарабика.

Томаса Джеффериса (1719–1771) называли географом при короле Георге III. Он был английским картографом и ведущим поставщиком карт своего времени. Он чертил и печатал карты для правительства и других государственных органов. На его счету целый ряд коммерческих карт и атласов, в основном Северной Америки. Занимаясь картографией в разных уголках мира, он интересовался местными традиционными нарядами, которые впоследствии были блестяще переданы в данной коллекции. В конце XVIII века заморские путешествия для собственного удовольствия были относительно новым явлением, поэтому подобные коллекции пользовались популярностью, позволяя получить представление о жителях других стран как настоящим туристам, так и «диванным» путешественникам.

Разнообразие иллюстраций в книгах Джеффериса — яркое свидетельство уникальности и оригинальности народов мира в то время. С тех пор тенденции в одежде сильно изменились, а региональные и национальные различия, такие значимые 200 лет назад, постепенно сошли на нет. В наши дни бывает сложно отличить друг от друга жителей разных континентов. Если взглянуть на это с оптимистической точки

зрения, мы пожертвовали культурным и внешним разнообразием в угоду более насыщенной личной жизни или более разнообразной и интересной интеллектуальной и технической деятельности.

В то время как большинство компьютерных изданий мало чем отличаются друг от друга, компания Manning выбирает для своих книг обложки, основанные на богатом региональном разнообразии, которое Джефферис воплотил в иллюстрациях два столетия назад. Это ода находчивости и инициативности современной компьютерной индустрии.

Часть I
Основы

В первой части книги приводятся основы, необходимые для понимания новых идей, появившихся в Java 8. К концу первой части вы будете знать, что такое лямбда-выражения, и научитесь писать код, лаконичный и в то же время достаточно гибкий для адаптации к меняющимся требованиям.

- ❑ В главе 1 мы резюмируем основные изменения в Java (лямбда-выражения, ссылки на методы, потоки данных и методы с реализацией по умолчанию) и подготовим фундамент для материала, изложенного в остальной части книги.
- ❑ Из главы 2 вы узнаете о параметризации поведения — важном для Java 8 паттерне разработки программного обеспечения, лежащем в основе лямбда-выражений.
- ❑ Глава 3 подробно рассказывает про понятия лямбда-выражений и ссылок на методы, приводятся примеры кода и контрольные задания.