

Содержание

Предисловие	17
1. Всё, что вам нужно, — это Java	21
<i>Андерс Норас</i>	
2. Тестирование на одобрение	23
<i>Эмили Бач</i>	
3. Усильте Javadoc AsciiDoc'ом	25
<i>Джеймс Эллиотт</i>	
4. Будьте внимательны к контейнерному окружению	27
<i>Давид Делабассе</i>	
5. Поведение — это «легко»; состояние — это сложно	29
<i>Эдсон Янага</i>	
6. Бенчмаркинг — это сложно, JMН поможет	31
<i>Майкл Хангер</i>	
7. Преимущества систематизации и проверки архитектурного качества	34
<i>Дэниел Брайант</i>	
8. Разбивайте проблемы и задачи на небольшие фрагменты	37
<i>Жанна Боярски</i>	
9. Создавайте разнообразные команды	39
<i>Икшель Руис</i>	

10. Сборки не обязательно должны быть медленными и ненадежными	42
<i>Дженн Стретер</i>	
11. «Но на моем компьютере это работает!»	44
<i>Бенджамин Мушко</i>	
12. Дело против fat JAR	47
<i>Дэниел Брайант</i>	
13. Реставратор кода	49
<i>Авраам Марин-Перез</i>	
14. Параллелизм в JVM	51
<i>Марио Фуско</i>	
15. CountdownLatch — друг или враг?	53
<i>Алексей Сошин</i>	
16. Декларативное выражение — вот путь к параллелизму	56
<i>Рассел Уиндер</i>	
17. Поставляйте качественное ПО быстрее	58
<i>Бурк Хуфнагель</i>	
18. Не знаете, который час?	60
<i>Кристин Горман</i>	
19. Не скрывайте от себя все инструменты, используя IDE	63
<i>Гейл Оллис</i>	
20. Не меняйте свои переменные	65
<i>Стив Фримен</i>	

21. Научитесь использовать SQL-мышление по максимуму	69
<i>Дин Уэмплер</i>	
22. События между компонентами Java	71
<i>А. Махди Абдель-Азиз</i>	
23. Циклы обратной связи	74
<i>Лиз Кио</i>	
24. На всю катушку	76
<i>Майкл Хангер</i>	
25. Следуйте скучным стандартам	78
<i>Адам Биен</i>	
26. Частые релизы снижают риск	80
<i>Крис О'Делл</i>	
27. От головоломок к продуктам	82
<i>Джессика Керр</i>	
28. «Разработчик полного цикла» — это образ мышления	84
<i>Мацей Валковяк</i>	
29. Сборщик мусора — ваш друг	86
<i>Холли Камминс</i>	
30. Называйте вещи своими именами	88
<i>Питер Хилтон</i>	
31. Эй, Фред, не мог бы ты передать мне HashMap?	90
<i>Кирк Пеппердайн</i>	

32. Как избежать Null	92
<i>Карлос Обрегон</i>	
33. Как вывести из строя JVM	95
<i>Томас Ронзон</i>	
34. Улучшение повторяемости и контролируемости посредством непрерывной поставки	97
<i>Билли Корандо</i>	
35. В языковых войнах Java может за себя постоять	99
<i>Дженнифер Райф</i>	
36. Встроенное мышление	102
<i>Патриция Аас</i>	
37. Взаимодействие с Kotlin	104
<i>Себастьяно Поджи</i>	
38. Дело сделано, но... ..	106
<i>Жанна Боярски</i>	
39. Сертификаты Java: пробирный камень технологий	108
<i>Мала Гупта</i>	
40. Java — дитя 90-х	110
<i>Бен Эванс</i>	
41. Программирование на Java в аспекте производительности JVM	112
<i>Моника Беквит</i>	
42. Java должна приносить радость	115
<i>Холли Камминс</i>	

43. Неуказываемые типы Java	117
<i>Бен Эванс</i>	
44. JVM — мультипарадигмальная платформа. Используйте это, чтобы повысить свой уровень программирования	120
<i>Рассел Уиндер</i>	
45. Держите руку на пульсе	122
<i>Триша Джи</i>	
46. Виды комментариев	124
<i>Николай Парлог</i>	
47. Знай flatMap свой	127
<i>Дэниел Инохоса</i>	
48. Знайте свои коллекции	130
<i>Никхил Нанивадекар</i>	
49. Обратите внимание на Kotlin	132
<i>Майк Данн</i>	
50. Изучайте идиомы Java и храните их в памяти	136
<i>Жанна Боярски</i>	
51. Учитесь создавать kata и создавайте kata, чтобы учиться	138
<i>Дональд Рааб</i>	
52. Научитесь любить ваш устаревший код	141
<i>Уберто Барбини</i>	
53. Научитесь использовать новые функции Java	143
<i>Гейл С. Андерсон</i>	

54. Изучите свою IDE, чтобы уменьшить когнитивную нагрузку	146
<i>Триша Джи</i>	
55. Давайте заключим контракт: искусство разработки Java API	148
<i>Марио Фуско</i>	
56. Делайте код простым и читабельным	150
<i>Эмили Цзян</i>	
57. Добавьте в вашу Java немного Groovy	153
<i>Кен Коузен</i>	
58. Минимизируйте конструкторы	156
<i>Стив Фримен</i>	
59. Назовите дату	159
<i>Кевлин Хенни</i>	
60. Необходимость технологий промышленной прочности	161
<i>Пол У. Гомер</i>	
61. Создавайте только те части, которые изменяются, и повторно используйте остальные	163
<i>Дженн Стретер</i>	
62. Проекты с открытым кодом — это не волшебство	165
<i>Дженн Стретер</i>	
63. Optional — монада, нарушающая закон, но это хороший тип	167
<i>Николай Парлог</i>	

64. Упаковка по функциям с модификатором доступа по умолчанию	170
<i>Марко Билен</i>	
65. Продакшн — самое радостное место на земле	172
<i>Джош Лонг</i>	
66. Программируйте с GUT	175
<i>Кевлин Хенни</i>	
67. Ежедневно читайте OpenJDK	178
<i>Хайнц М. Кабуц</i>	
68. По-настоящему заглянуть «под капот»	180
<i>Рафаэль Беневидес</i>	
69. Возрождение Java	182
<i>Сандер Мак</i>	
70. Заново откройте для себя JVM с помощью Clojure	184
<i>Джеймс Эллиотт</i>	
71. Преобразование логических значений в перечисления	186
<i>Питер Хилтон</i>	
72. Рефакторинг для ускорения чтения	188
<i>Бенджамин Мушкала</i>	
73. Простые объекты значений	191
<i>Стив Фримен</i>	
74. Позаботьтесь о своих объявлениях модулей	194
<i>Николай Парлог</i>	

75. Заботьтесь о создаваемых зависимостях	197
<i>Брайан Вермеер</i>	
76. Принимайте разделение ответственности всерьез	199
<i>Дэйв Фарли</i>	
77. Техническое интервьюирование — это навык, который стоит развивать	202
<i>Триша Джи</i>	
78. Разработка на основе тестирования	204
<i>Дэйв Фарли</i>	
79. В вашем каталоге bin/ отличные инструменты	207
<i>Род Хилтон</i>	
80. Вылезайте из песочницы Java	209
<i>Иэн Ф. Дарвин</i>	
81. Мысли о сопрограммах	211
<i>Доун и Дэвид Гриффитс</i>	
82. Потоки — это инфраструктура, относитесь к ним соответственно	214
<i>Рассел Уиндер</i>	
83. Три черты по-настоящему отличных разработчиков	216
<i>Джанна Патчей</i>	
84. Компромиссы в архитектуре микросервисов	218
<i>Кенни Бустани</i>	
85. Не проверяйте свои исключения	220
<i>Кевлин Хенни</i>	

86. Как высвободить потенциал интеграционного тестирования с использованием контейнеров 223
Кевин Виттек
87. Необъяснимая эффективность фаззинга 225
Нэт Прайс
88. Используйте покрытие, чтобы улучшить ваши модульные тесты 228
Эмили Бач
89. Свободно применяйте нестандартные идентификационные аннотации 230
Марк Ричардс
90. Тестируйте, чтобы разрабатывать более качественное ПО быстрее 233
Марит ван Дейк
91. Применение объектно-ориентированных принципов в тестовом коде 235
Энджи Джонс
92. Как развивать карьеру, опираясь на силы сообщества 238
Сэм Хепберн
93. Что такое программа JCP и как в ней участвовать 240
Хизер Ванчура
94. Почему я не придаю никакого значения сертификации 242
Колин Випурс

95. Пишите к документации комментарии в одно предложение	244
<i>Питер Хилтон</i>	
96. Пишите «читаемый код»	247
<i>Дэйв Фарли</i>	
97. Молодые, старые и мусор	250
<i>Мария Ариас де Рейна</i>	
Об авторах	252
Предметный указатель	282

Предисловие

Ум — это не сосуд, который нужно наполнить, а факел, который нужно зажечь.

Плутарх

Что должен знать каждый Java-программист? Зависит от обстоятельств. От того, зачем, кого и когда вы спрашиваете. Ответов по крайней мере столько же, сколько точек зрения. В языке, платформе, экосистеме и сообществе, которые влияют на программное обеспечение (ПО) и жизни многих людей и делали так из двадцатого века в двадцать первый, от одного ядра ко многим, от мегабайт к гигабайтам, ответ зависит от большего количества факторов, чем возможно охватить в одной книге единственным автором.

Вместо этого в данной книге мы опираемся на некоторые из этих многочисленных точек зрения, чтобы дать вам поперечный срез и представление о типах мышления во вселенной Java. Это не *все* мнения, но 97 из них от 73 участников. Прочитайте предисловие «97 вещей, которые должен знать каждый программист» (O'Reilly):

Когда так много нужно знать, так много нужно сделать и есть так много способов сделать это, ни один человек или один источник не может утверждать, что его путь — «единственный истинный». Идеи не совпадают, как модульные части, и никто к этому не стремится — пожалуй, даже наоборот. Ценность каждого вклада проистекает из его уникальности. Ценность коллекции заключается в том, как материалы дополняют, подтверждают и даже противоречат друг другу. Здесь нет всеобъемлющего повествования: вы должны прочувствовать идеи, поразмыслить над ними и связать воедино, примеряя их к своей ситуации, знаниям и опыту.

Что должен знать каждый Java-программист? 97 тем, которые мы выбрали в качестве ответа, охватывают язык, JVM, методы тестирования, JDK, сообщество, историю, гибкое мышление, ноу-хау реализации, профессионализм, стиль, суть, парадигмы программирования, программистов как людей, архитектуру ПО, навыки за пределами программирования, инструментарий, механику GC, языки JVM, отличные от Java... и многое другое.

Разрешения

В духе первой книги из серии «97 вещей...» каждый интеллектуальный вклад в данный том внесен на основе неограничивающей модели с открытым исходным кодом. Каждый вклад автора лицензируется в соответствии с Creative Commons Attribution 4.0 license (<https://oreil.ly/zPsKK>). Многие материалы также впервые появились в публикации «97 вещей...» на странице Medium (<https://medium.com/97-things>).

Все это — топливо и огонь для ваших мыслей и кода.

Онлайн-обучение O'Reilly

Более сорока лет *O'Reilly Media* помогает компаниям добиваться успеха, предоставляя им технологии и бизнес-тренинги, сведения и наработки.

Наше уникальное сообщество экспертов и новаторов делится знаниями и опытом с помощью книг, статей, конференций и нашей платформы онлайн-обучения O'Reilly. Последняя предоставляет вам доступ по запросу к практическим подготовительным курсам, методам углубленного обучения, интерактивным средам программирования и обширной коллекции текстов и видео от O'Reilly и более 200 других издателей. Чтобы узнать подробности, посетите <http://oreilly.com>.

Как с нами связаться

Вы можете получить доступ к веб-странице этой книги, где найдете примеры, обнаруженные неточности и прочую дополнительную информацию, по адресу <https://oreil.ly/97Tejpsk>.

Чтобы прокомментировать или задать технические вопросы об этой книге, напишите письмо на bookquestions@oreilly.com.

Для получения новостей и информации о наших книгах и курсах посетите <http://oreilly.com>.

Присоединяйтесь к нам в Twitter <http://twitter.com/oreillymedia>, а также посмотрите http://twitter.com/97_Things.

Смотрите нас на YouTube: <http://youtube.com/oreillymedia>.

Благодарности

Многие люди вложили свои знания и время, как прямо, так и косвенно, в проект «97 вещей, которые должен знать каждый Java-программист». Все они заслуживают похвалы.

Мы хотели бы поблагодарить всех тех, кто потратил время и силы, чтобы внести свой вклад в эту книгу. Мы также благодарны Брайану Гетцу за дополнительные отзывы, комментарии и предложения.

Спасибо O'Reilly за поддержку, которую они оказали этому проекту, в том числе Зан Маккуэйд и Корбину Коллинзу за их руководство и заботу об участниках и содержании, а также Рэйчел Румелиотис, Сьюзан Конант и Майку Лукидесу за их вклад в проделанную работу.

Кроме того, Кевлин хотел бы поблагодарить свою жену Каролин за то, что она понимает смысл его бреда, и своих сыновей Стефана и Янника за то, что они понимают своих родителей.

Триша желала бы добавить благодарность своему мужу Исре (за то, что он подсказал ей: пребывание в стрессе из-за того, что она делает слишком мало, мешает ей делать вообще что-либо) и своим дочерям Эви и Эми за бескорыстную любовь и обнимашки.

Мы надеемся, что эта книга станет для вас информативной, поучительной и вдохновляющей. Наслаждайтесь!

Всё, что вам нужно, — это Java*

Андерс Норас



Работая над первой крупной редакцией Visual Studio, команда Microsoft представила миру трех персонажей-разработчиков: Морта, Элвиса и Эйнштейна.

Морт, приспособливающийся работник, быстро исправлял и придумывал что-то по ходу дела. Элвис, прагматичный сотрудник, создавал решения на века, обучаясь на рабочем месте. Эйнштейн, программист-параноик, как одержимый разрабатывал наиболее эффективное решение и старался во всем разобраться, прежде чем писать код.

Языки программирования (ЯП) разделены, как религиозные касты. Мы со стороны Java смеялись над Мортами и хотели быть Эйнштейнами, которые создают фреймворки, чтобы вынудить Элвисов писать свой код «правильно».

Это происходило на заре эры фреймворков, и, если вы не владели новейшим, лучшим объектно-реляционным мэппером и фреймворком с инверсией управления, вы не были настоящим Java-программистом. Библиотеки превратились в фреймворки с установленной архитектурой. И по мере того, как они становились технологическими экосистемами, многие из нас забыли о маленьком, но очень способном языке — Java.

Java — отличный ЯП, и в его библиотеке классов есть инструменты на все случаи жизни. Нужно работать с файлами? `java.nio` тебя прикроет. Базы данных? `java.sql` — вам сюда. Почти каждый дистрибутив Java даже оснащен полноценным HTTP-сервером, однако вам придется перейти от модуля с именем Java на `com.sun.net.httpserver`.

По мере того как наши приложения переходят к бессерверным архитектурам, где единицами развертывания могут быть отдельные функции, преимущества, которые мы получаем от фреймворков, уменьшаются. Это связано с тем,

* В оригинале All You Need Is Java — отсылка к All You Need Is Love, классическому хиту Beatles 1967 года. — *Прим. ред.*

что мы, вероятно, будем тратить не так много времени на решение технических и инфраструктурных проблем, сосредоточив усилия по программированию на бизнес-возможностях, которые реализуют наши приложения.

Как выразился Брюс Джойс, автор книг про образование и педагогику:

Нам приходится время от времени изобретать колесо, и не потому, что нам нужно много колес, а потому, что нам нужно много изобретателей.

Многие задалась целью создания универсальных фреймворков для бизнес-логики, чтобы максимально использовать их повторно. Большинство потерпели неудачу, поскольку на самом деле нет никаких универсальных бизнес-проблем. Чем один бизнес отличается от другого? Тем, что делает нечто особенное уникальным, специфичным для него образом. Вот почему вы гарантированно будете писать бизнес-логику практически для каждого проекта. Возможно, у вас возникнет соблазн ввести механизм правил или что-то подобное ради того, чтобы создать нечто универсальное и многократно. Но в конечном счете настройка механизма правил — это программирование, часто на языке, уступающем Java. Почему бы вместо этого не попробовать просто написать на Java? Вы с удивлением обнаружите, что конечный результат легко читаем, а это, в свою очередь, упрощает поддержку кода даже для тех программистов, которые на Java не работают.

Довольно часто можно заметить, что библиотека классов Java немного ограничена, и в таких случаях требуется нечто, способное сделать чуть более комфортной работу с датами, сетевое взаимодействие или что-то еще. Это прекрасно. Задействуйте любую библиотеку. Разница в том, что теперь вы будете применять ее, потому что возникла конкретная необходимость, а не потому, что она входила в стек, который вы всегда использовали.

В следующий раз, когда вам в голову придет идея небольшой программы, выведите свои знания о библиотеке классов Java из спящего режима, вместо того чтобы лезть в дебри JHipster. Хипстеризм — это уже не комильфо; жить простой жизнью — вот что актуально сейчас. Бьюсь об заклад, Морту нравилась простая жизнь.