


100

ОШИБОК JAVA

и как их избежать

Тагир
Валеев

 **БОМБОРА**
ИЗДАТЕЛЬСТВО
Москва

УДК 004.42
ББК 32.973.2-018.1
В15

Tagir Valeev

100 JAVA MISTAKES AND HOW TO AVOID THEM

© Limited company «Publishing house «Eksmo» 2025. Authorized translation of the English edition © 2023 Manning Publications.

This translation is published and sold by permission of Manning Publications, the owner of all rights to publish and sell the same.

Валеев, Тагир.
В15 100 ошибок Java и как их избежать / Тагир Валеев ; [перевод с английского В. В. Краснянской, Ю. В. Войтко]. — Москва : Эксмо, 2025. — 480 с. : ил. — (Manning: профессиональные книги для ИТ-специалистов).

ISBN 978-5-04-207668-8

Эта книга научит создавать высококачественный и надежный код на Java, используя инструменты статистического анализа. Вы узнаете, как обнаруживать и предотвращать распространенные ошибки программирования, оптимизировать процесс тестирования и отладки, а также повысить качество ПО еще на этапе разработки. Полезная информация представлена живым языком со множеством примеров и практических рекомендаций.

УДК 004.42
ББК 32.973.2-018.1

ISBN 978-5-04-207668-8

© Краснянская В. В., перевод на русский язык, 2025
© Оформление. ООО «Издательство «Эксмо», 2025

Моей матери, Наде

Оглавление

<i>Предисловие</i>	14
<i>Вступление</i>	16
<i>Благодарности</i>	18
<i>Об этой книге</i>	22
Кому следует прочесть эту книгу	22
Как структурирована эта книга (общая схема)	22
Замечание о коде	23
Дискуссионный форум liveBook	24
<i>Об авторе</i>	25
<i>Об иллюстрации на обложке</i>	26
1. Управление качеством кода	27
1.1. Анализ кода и парное программирование	29
1.2. Стиль кода	30
1.3. Статический анализ	33
1.3.1. Инструменты статического анализа для языка Java	33
1.3.2. Использование статических анализаторов	35
1.3.3. Ограничения статического анализа	39
1.3.4. Отключение нежелательных предупреждений	41
1.4. Автоматическое тестирование	44
1.5. Мутационное покрытие	47
1.6. Динамический анализ	49

1.7. Операторы контроля	49
1.8. Итоги	50
2. Выражения	52
2.1. Ошибка № 1. Неправильные представления о последовательности числовых операций	52
2.1.1. Двоичный сдвиг	53
2.1.2. Побитовые операторы	56
2.2. Ошибка № 2. Отсутствие скобок в условных выражениях	58
2.2.1. Порядок выполнения && и 	58
2.2.2. Условный оператор и сложение	61
2.2.3. Условный оператор и проверка на null	64
2.3. Ошибка № 3. Случайная конкатенация вместо сложения	67
2.4. Ошибка № 4. Многострочные строковые литералы	68
2.5. Ошибка № 5. Унарный плюс	70
2.6. Ошибка № 6. Неявное преобразование типов в условных выражениях	72
2.6.1. Числовые объектные переменные в условных выражениях	73
2.6.2. Вложенные условные выражения	75
2.7. Ошибка № 7. Использование логических операторов, не выполняющихся по укороченной схеме	77
2.8. Ошибка № 8. Смешение && и 	84
2.9. Ошибка № 9. Неправильное использование вызовов с переменным количеством аргументов	88
2.9.1. Неоднозначные вызовы с переменным количеством аргументов	88
2.9.2. Смешение массива и коллекции	90
2.9.3. Использование массива примитивов в вызове с переменным количеством аргументов	92
2.10. Ошибка № 10. Условные операторы и вызовы с переменным количеством аргументов	93
2.11. Ошибка № 11. Пропуск важного возвращаемого значения	96
2.12. Ошибка № 12. Не использовать только что созданный объект	104
2.13. Ошибка № 13. Связать ссылку на метод с неверным методом	105

2.14. Ошибка № 14. Использование неверного метода в ссылке на метод	110
2.15. Итоги	112
3. Структура программы	114
3.1. Ошибка № 15. Некорректная цепочка <code>if-else</code>	114
3.2. Ошибка № 16. Условие, которое доминирует над последующим условием	121
3.4. Ошибка № 18. Некорректный классический цикл <code>for</code>	130
3.5. Ошибка № 19. Неиспользование переменной цикла	133
3.6. Ошибка № 20. Неправильное направление цикла	135
3.7. Ошибка № 21. Переполнение цикла	137
3.8. Ошибка № 22. Идемпотентное тело цикла	138
3.9. Ошибка № 23. Неправильный порядок инициализации	143
3.9.1. Статические поля	143
3.9.2. Поля подклассов	146
3.9.3. Порядок инициализации классов	149
3.9.4. Цикл инициализации констант <code>enum</code>	154
3.10. Ошибка № 24. Отсутствие вызова суперметода	158
3.11. Ошибка № 25. Случайное объявление статического поля	160
3.12. Итоги	166
4. Числа	167
4.1. Ошибка № 26. Случайное использование восьмеричного литерала	167
4.2. Ошибка № 27. Числовое переполнение	169
4.2.1. Переполнение в Java	169
4.2.2. Присваивание результата умножения чисел типа <code>int</code> переменной типа <code>long</code>	177
4.2.3. Размер файла, время и финансовые расчеты	178
4.3. Ошибка № 28. Округление при целочисленном делении	180
4.4. Ошибка № 29. Абсолютное значение <code>Integer.MIN_VALUE</code> ..	182
4.5. Ошибка № 30. Проверка на нечетность и отрицательные числа	185

4.6. Ошибка № 31. Расширяющее преобразование типа с потерей точности	187
4.7. Ошибка № 32. Не ограниченное условиями сужающее преобразование типов	188
4.8. Ошибка № 33. Отрицательные шестнадцатеричные значения	190
4.9. Ошибка № 34. Неявное преобразование типов при составном присваивании	191
4.10. Ошибка № 35. Деление и составное присваивание	194
4.11. Ошибка № 36. Использование типа <code>short</code>	195
4.12. Ошибка № 37. Написание алгоритмов для манипуляции с битами вручную	196
4.13. Ошибка № 38. Забыть об отрицательных значениях типа <code>byte</code>	199
4.14. Ошибка № 39. Неправильный порядок ограничения значения	200
4.15. Ошибка № 40. Неправильное использование особых чисел с плавающей точкой	202
4.15.1. Знаковый ноль: <code>+0.0</code> и <code>-0.0</code>	202
4.15.2. Не число: значение <code>NaN</code>	205
4.15.3. <code>Double.MIN_VALUE</code> не является минимальным значением	206
4.16. Итоги	208
5. Основные исключения	209
5.1. Ошибка № 41. <code>NullPointerException</code>	210
5.1.1. Отказ от <code>null</code> и защитные проверки	211
5.1.2. Использование типа <code>Optional</code> вместо <code>null</code>	219
5.1.3. Аннотации для обозначения нуллабельности	220
5.2. Ошибка № 42. <code>IndexOutOfBoundsException</code>	225
5.3. Ошибка № 43. <code>ClassCastException</code>	229
5.3.1. Явное преобразование типов	230
5.3.2. Параметризованные типы и неявное приведение	235
5.3.3. Различные загрузки классов	241
5.4. Ошибка № 44. <code>StackOverflowError</code>	243
5.4.1. Глубокая, но конечная рекурсия	244
5.4.2. Бесконечная рекурсия	248
5.5. Итоги	252

6. <i>Строки</i>	254
6.1. Ошибка № 45. Предполагать, что значение <code>char</code> является символом	254
6.2. Ошибка № 46. Неожиданные преобразования регистра символов	259
6.3. Ошибка № 47. Использование <code>String.format</code> с языковой средой по умолчанию	261
6.4. Ошибка № 48. Расхождение аргументов форматирования	264
6.5. Ошибка № 49. Использование обычных строк вместо регулярных выражений	268
6.6. Ошибка № 50. Случайное использование метода <code>replaceAll</code>	272
6.7. Ошибка № 51. Случайное использование экранированных последовательностей	274
6.8. Ошибка № 52. Сравнение строк в разных регистрах	277
6.9. Ошибка № 53. Не проверять результат метода <code>indexOf</code>	279
6.10. Ошибка № 54. Ошибочное использование аргументов <code>indexOf</code>	283
6.11. Итоги	284
7. <i>Сравнение объектов</i>	286
7.1. Ошибка № 55. Использование эквивалентности ссылок вместо метода <code>equals</code>	287
7.2. Ошибка № 56. Предполагать, что <code>equals()</code> сравнивает содержимое	291
7.3. Ошибка № 57. Использование <code>URL.equals()</code>	294
7.4. Ошибка № 58. Сравнение <code>BigDecimal</code> в разных масштабах ..	296
7.5. Ошибка № 59. Использование <code>equals()</code> на объектах несвязанных типов	297
7.6. Ошибка № 60. Неправильная реализация метода <code>equals()</code> ..	301
7.7. Ошибка № 61. Неправильный <code>hashCode()</code> с полями-массивами	307
7.8. Ошибка № 62. Несовпадение между <code>equals()</code> и <code>hashCode()</code>	310

7.9. Ошибка № 63. Опирается на конкретное возвращаемое значение метода <code>compare()</code>	315
7.10. Ошибка № 64. При сравнении равных объектов не возвращается нуль	318
7.11. Ошибка № 65. Использование вычитания при сравнении чисел	321
7.12. Ошибка № 66. Игнорировать возможные значения NaN в методах сравнения	327
7.13. Ошибка № 67. Не представлять объект как цепочку значений в методе сравнения	329
7.14. Ошибка № 68. Возвращать из компаратора случайные числа	335
7.15. Итоги	336
8. Коллекции и ассоциативные массивы <i>Map</i>	338
8.1. Ошибка № 69. Поиск объекта несвязанного типа	338
8.2. Ошибка № 70. Одновременное использование единичного объекта и коллекции	342
8.3. Ошибка № 71. Поиск значения <code>null</code> в коллекции, не принимающей <code>null</code>	347
8.4. Ошибка № 72. Использование значения <code>null</code> в ассоциативных массивах	349
8.5. Ошибка № 73. Попытка изменить неизменяемую коллекцию	353
8.6. Ошибка № 74. Использование изменяемых объектов как ключей	358
8.7. Ошибка № 75. Полагаться на порядок появления элементов <code>HashMap</code> и <code>HashSet</code>	362
8.8. Ошибка № 76. Модификация коллекции во время обхода	364
8.9. Ошибка № 77. Путаница с перегрузками <code>List.remove</code>	369
8.10. Ошибка № 78. Перескакивать через следующий элемент после выполнения <code>List.remove</code>	370
8.11. Ошибка № 79. Чтение коллекции внутри <code>Collection.removeIf</code>	375

8.12. Ошибка № 80. Изменения ассоциативного массива внутри лямбды в <code>computeIfAbsent</code>	377
8.13. Ошибка № 81. Нарушение контракта итератора	382
8.14. Итоги	387
9. Библиотечные методы	389
9.1. Ошибка № 82. Передача символа в конструктор <code>StringBuilder</code>	389
9.2. Ошибка № 83. Побочные эффекты цепочки операций <code>Stream</code> <code>API</code>	391
9.3. Ошибка № 84. Повторное использование потока обработки данных	396
9.4. Ошибка № 85. Использование <code>null</code> в потоке обработки данных тогда, когда это недопустимо	400
9.5. Ошибка № 86. Нарушение контракта операций <code>Stream API</code> ...	401
9.6. Ошибка № 87. Использование <code>getClass()</code> вместо <code>instanceof</code>	406
9.7. Ошибка № 88. Использование <code>getClass()</code> на константах <code>enum</code> , аннотациях или других классах	408
9.8. Ошибка № 89. Неправильное преобразование из строки в булево значение	411
9.9. Ошибка № 90. Неправильные спецификаторы формата при форматировании даты	413
9.10. Ошибка № 91. Случайная инвалидация слабой или мягкой ссылки	415
9.11. Ошибка № 92. Полагать, что ничего никогда не изменится	417
9.12. Ошибка № 93. Доступ к конкурентным структурам данных должен быть атомарным	421
9.13. Итоги	424
10. Модульное тестирование	425
10.1. Ошибка № 94. Побочные эффекты в операторах контроля	426

Оглавление

10.2. Ошибка № 95. Неправильное использование методов подтверждения	428
10.3. Ошибка № 96. Неправильный тест исключения	430
10.4. Ошибка № 97. Преждевременный выход из тестового метода	432
10.5. Ошибка № 98. Игнорировать <code>AssertionError</code> в модульных тестах	434
10.6. Ошибка № 99. Использование <code>assertNotEquals</code> для проверки контракта равенства	438
10.7. Ошибка № 100. Некорректные методы тестирования	440
10.8. Итоги	442
<i>Приложение А. Аннотации статического анализа</i>	<i>444</i>
А.1. Пакеты аннотаций	444
А.2. Виды аннотаций	447
<i>Приложение Б. Расширение инструментов статического анализа</i>	<i>452</i>
Б.1. Плагины для Error Prone	453
Б.2. Плагины для SpotBugs	457
Б.3. Плагин для IntelliJ IDEA	464
Б.4. Использование структурного поиска и замены в IntelliJ IDEA ..	469

Предисловие

В августе 2023-го мне на почту упало вот что: «Здравствуйте, Кей! Меня зовут Тагир Валеев, возможно, вы помните меня по кое-каким обсуждениям в групповых рассылках. И еще мы лично встречались на одной конференции...»

Письмо показалось мне странным. Разумеется, я знал Тагира — и не только по тем «обсуждениям в групповых рассылках». Я автор классической книги «Java. Библиотека профессионала»¹, цель которой со времен Java 1.0 заключается в том, чтобы разъяснять этот язык квалифицированным программистам. Пока я обновляю текст (для 13-го издания), мне нужно рыться во множестве малоизвестных и каверзных аспектов самого ЯП и его библиотеки. Периодически я захожу в тупик с каким-нибудь изошренным вопросом, запускаю мою любимую поисковую систему и нахожу вдумчивый разбор, проведенный не кем иным, как Тагиром. Этот парень по-настоящему разбирается в Java. Ему точно нужно написать книгу...

О чем и было его сообщение. «Я работаю над своей самой первой книгой», — извещал Тагир и просил меня написать предисловие. Тему он преподносил так: «В тексте я акцентирую внимание на типичных и регулярно возникающих ошибках, которые разработчики допускают в программах на Java, и стараюсь дать советы о том, как их избежать».

Подобная идея не нова. Много лет назад мне пришлось научиться программировать на C, и меня ужасало, что люди вообще используют столь очевидно опасный язык, но я ничего не мог поделать. Только в этом ЯП имелись привязки к библиотекам, которые мне требовалось применить. Я совершил все распространенные ошибки и лишь позднее наткнулся на чудесную книгу Эндрю

¹ Хорстманн К. Java. Библиотека профессионала. Том 1. Основы. М.: Диалектика, 2020.

Книга «Ловушки и подводные камни С»¹. Моя жизнь протекала бы гораздо безмятежнее, если бы я тогда опирался на его текст, а не набивал шишки, тыкаясь наугад.

Конечно же, Java — это не С, а весьма тщательно разработанный ЯП, и его библиотека тоже отлично продумана. Правда, если Java 1.0 была несложной, то текущая версия возлежит на слоях хитросплетений, которые накапливались четверть века. И мы за минувшие годы тоже кое-чему научились... В общем, не все из того, что двадцать пять лет назад выглядело как хорошая мысль, выдержало проверку временем.

Соответственно, можно с легкостью рассмотреть сотню «подводных камней» в языке и программном интерфейсе Java, чем Тагир и занялся в своей книге. Каждый ее подраздел основан на примере кода, увиденного автором на практике. А кода он повидал немало, поскольку трудится в JetBrains — компании, создавшей IntelliJ IDEA, самую популярную (с большим отрывом) интегрированную среду разработки для Java.

Могу вас заверить, что вы многое узнаете, читая об этих ловушках: поймете, почему они опасны, и научитесь обходить их. Я программировал на Java со времен версии 1.0 и, пока изучал книгу, не раз ловил себя на том, что согласно киваю. В иных случаях у меня вырывалось: «Ого, я пользовался Java со времен версии 1.0, но о таком и не подозревал!» Негативные моменты, выбранные Тагиром, действительно важны. Они встречаются в реальном коде, поэтому всем, кто работает с Java, необходимо держать в уме такие проблемы и то, как с ними справляться.

Лучший путь к успеху — учиться на неудачах, а наименее затратный путь к успеху — учиться на неудачах других. Чтобы стать более умелым и уверенным в себе программистом на Java, идите вслед за Тагиром по дороге, проложенной через подобные ошибки, и извлекайте пользу из преподанных им уроков! О более сведущем провожатом и мечтать не нужно.

*Кей Хорстманн,
почетный профессор Университета штата Калифорния в Сан-Хосе*

¹ Koenig A. C Traps and Pitfalls. Addison-Wesley, 1989.

Вступление

Я уже около двадцати лет создаю коммерческое ПО, прежде всего на Java, хотя у меня есть опыт работы и с другими языками. Мною написаны сотни тысяч строк кода, а пока я изучал программы, составленные другими людьми, то прочел даже больше. Если на протяжении моей карьеры подтвердилась хотя бы одна азбучная истина, то звучит она так: «Люди неидеальны». Любой программист допускает ошибки. Самые умные разработчики ПО тоже способны внести баги, со стороны кажущиеся очевидными.

За ошибками же всегда следуют трудности. Программа не ведет себя так, как ожидалось, сбоит и выбрасывает исключения, пользователи жалуются, а вы тратите часы на то, чтобы воспроизвести, проанализировать и решить проблему. Даже для того, чтобы отыскать крошечный промах вроде опечатки, могут потребоваться серьезные усилия.

Примерно десять лет назад я обнаружил статический анализатор Java под названием FindBugs, что стало для меня настоящим откровением. В нашей компании раньше не применяли такие инструменты, и он за пару минут отыскал десятки реальных ошибок в проекте, над которым я тогда работал. FindBugs точно указывал на неправильный фрагмент кода, поэтому мне не пришлось ни ждать, пока мой промах затронет пользователей, ни выявлять причины проблемы. Я смог просто исправить ее.

Так я стал завзятым фанатом статического анализа. Однако же вскоре выяснилось, что возможности FindBugs не безграничны: он часто выдает ложные срабатывания и указывает на довольно мелкие недостатки, из-за чего затягивается изучение отчета. С другой стороны, он не сообщает о многих настоящих багах, которые следовало бы учесть. Я начал участвовать в работе над FindBugs, стараясь улучшить его, а через пару лет примкнул к потрясающей команде создателей IntelliJ IDEA — среды с великолепным статическим

Вступление

анализатором. С тех пор моя основная задача состоит в том, чтобы помогать разработчикам уменьшить количество багов.

Примерно тогда же я изменил свой подход к исправлению ошибок. Я уже не просто решал проблемы, а постоянно спрашивал себя: «Почему они возникли? Возможно ли было избежать их в принципе? Есть ли риск появления схожих багов в будущем? Что способна сделать моя команда, чтобы нам больше не встречались такие трудности?»

Я выяснил, что не существует панацеи от ошибок. Иногда вам может помочь статический анализ, а порой лучше положиться на тестирование или использование утверждений. В некоторых случаях нужно усовершенствовать ваш стиль программирования, автоматизировать создание кода или же применять методы из библиотек, вместо того чтобы подготавливать что-либо вручную. Впрочем, многие недочеты встречаются регулярно, а значит, поддаются классификации, и есть стандартные способы их предотвратить. Вот почему я решил написать книгу, чтобы обобщить свой опыт и перечислить наиболее типичные ошибки, появляющиеся в программах на Java. Надеюсь, что другие разработчики, прочитав ее, будут готовы решать проблемы, с которыми еще не сталкивались лично.