

УДК 004.457
ББК 32.972.13
В53

Виссер Дж.
В53 Разработка обслуживаемых программ на языке Java / пер. с англ. Р. Н. Рагимова. – М.: ДМК Пресс, 2017. – 182 с.: ил.

ISBN 978-5-97060-447-2

Данное практическое руководство познакомит вас с 10 простыми рекомендациями, помогающими писать программное обеспечение, которое легко поддерживать и адаптировать. Эти тезисы сформулированы на основании анализа сотен реальных систем.

Написанная консультантами компании Software Improvement Group книга содержит ясные и краткие советы по применению рекомендаций на практике. Примеры для этого издания написаны на языке Java, но существует аналогичная книга с примерами на языке C#.

Издание предназначено программистам на Java, желающим научиться писать качественный и хорошо поддерживаемый код.

УДК 004.457
ББК 32.972.13

Authorized Russian translation of the English edition of 'Building Maintainable Software, Java Edition'.

This translation is published and sold by permission of O'Reilly Media, Inc., which owns or controls all rights to publish and sell the same.

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

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

ISBN 978-1-4919-5352-5 (анг.) © 2016 Software Improvement Group, B.V.
 ISBN 978-5-97060-447-2 (рус.) © Оформление, издание, перевод, ДМК Пресс, 2017

Содержание

Об авторах	11
Предисловие.....	13
Глава 1. Введение	26
1.1. Что такое обслуживаемость?	27
Четыре вида обслуживаемости программного обеспечения	27
1.2. Почему так важна обслуживаемость?	28
Обслуживаемость значительно влияет на деловую сторону вопроса.....	28
Обслуживаемость обеспечивает улучшение других качественных характеристик	29
1.3. Три принципа, на которых основаны рекомендации.....	30
Принцип 1: рекомендации должны быть простыми.....	30
Принцип 2: применение рекомендаций с самого начала и значимость вклада каждого разработчика.....	31
Принцип 3: не все отступления от рекомендаций дают одинаковый отрицательный эффект.....	31
1.4. Заблуждения относительно обслуживаемости	32
Заблуждение: обслуживаемость зависит от языка программирования	32
Заблуждение: обслуживаемость зависит от прикладной области.....	33
Заблуждение: обслуживаемость гарантирует отсутствие ошибок	33
Заблуждение: обслуживаемость оценивается одной из двух альтернатив	34
1.5. Рейтинг обслуживаемости.....	34
1.6. Обзор рекомендаций по улучшению обслуживаемости	36
Глава 2. Пишите короткие блоки кода.....	38
2.1. Мотивация.....	41
Короткие блоки кода проще тестировать	41
Короткие блоки кода проще анализировать.....	41
Короткие блоки кода проще повторно использовать.....	42
2.2. Как применять рекомендацию	42
При написании нового блока кода.....	42
При добавлении в блок новых функциональных возможностей.....	44

Два метода рефакторинга для приведения кода в соответствие с рекомендацией	45
2.3. Типичные возражения против коротких блоков кода	50
Возражение: увеличение количества блоков кода плохо сказывается на производительности	50
Возражение: разделение кода ухудшает читаемость	50
Рекомендация препятствует надлежащему форматированию кода	51
Этот блок кода невозможно разделить	52
Разделение блоков кода не дает заметных преимуществ	53
2.4. Дополнительные сведения	54
Глава 3. Пишите простые блоки кода	56
3.1. Мотивация	62
Простые блоки проще изменять	62
Простые блоки кода проще тестировать	62
3.2. Как применять рекомендацию	63
Цепочки условий	63
Вложенность	65
3.3. Типичные возражения против создания простых блоков кода	67
Возражение: высокая сложность неизбежна	67
Возражение: разделение методов не уменьшает сложности	68
3.4. Дополнительные сведения	68
Глава 4. Не повторяйте один и тот же код	70
Виды дублирования	73
4.1. Мотивация	74
Код с дубликатами сложнее анализировать	74
В дублированный код сложно вносить изменения	75
4.2. Как применять рекомендацию	75
Извлечение суперкласса	77
4.3. Типичные возражения против исключения дублирования	80
Копирование фрагментов из другой базы кода допустимо	80
При незначительных изменениях дублирование неизбежно	81
Этот код никогда не изменится	81
Дублирование всех файлов допустимо в целях создания их резервных копий	82
Модульные тесты защитят меня	82
Дублирование строковых литералов неизбежно и совершенно безвредно	83
4.4. Дополнительные сведения	83

Глава 5. Стремитесь к уменьшению размеров интерфейсов	86
5.1. Мотивация.....	89
Интерфейсы небольшого размера упрощают понимание и повторное использование кода	89
В методы с компактным интерфейсом проще вносить изменения	89
5.2. Как применять рекомендацию	90
5.3. Типичные возражения против сокращения размеров интерфейсов	94
Возражение: объекты параметров требуют определения конструкторов с большим количеством параметров.....	95
Преобразование интерфейсов большого размера не улучшает ситуацию	95
Фреймворки или библиотеки предоставляют интерфейсы с длинными списками параметров.....	95
5.4. Дополнительные сведения	96
 Глава 6. Разделяйте задачи на модули.....	 98
6.1. Мотивация.....	102
Небольшие слабо связанные модули позволяют разработчикам иметь дело с надежно изолированными частями системы.....	102
Небольшие слабо связанные модули упрощают навигацию по коду	103
Небольшие слабо связанные модули делают все области кода более понятными новым разработчикам	103
6.2. Как применять рекомендацию	103
Разделение классов по решаемым задачам	103
Соккрытие подробностей реализации за интерфейсами	104
Замена пользовательского кода библиотеками или фреймворками от сторонних производителей.....	107
6.3. Типичные возражения против разделения задач.....	107
Возражение: слабые связи вступают в конфликт с возможностью повторного использования	107
Возражение: интерфейсы языка Java не предназначены для ослабления связей	108
Возражение: высокая нагрузка на служебные классы неизбежна.....	108
Возражение: не все слабо связанные решения улучшают обслуживаемость.....	109
 Глава 7. Избегайте тесных связей между элементами архитектуры.....	 111
7.1. Мотивация.....	112

8 ❖ Содержание

Слабая зависимость между компонентами обеспечивает изолированность его обслуживания	115
Слабая зависимость компонентов способствует разделению ответственности за обслуживание	115
Слабая зависимость компонентов упрощает тестирование.....	116
7.2. Как применять рекомендацию	116
Абстрактная фабрика как шаблон проектирования	117
7.3. Типичные возражения против устранения тесных связей компонентов.....	119
Возражение: зависимости между компонентами невозможно смягчить из-за тесного переплетения компонентов	119
Возражение: нет времени на исправление	119
Возражение: транзитный код просто необходим.....	120
7.4. Дополнительные сведения	120

Глава 8. Стремитесь к сбалансированности архитектуры компонентов

8.1. Мотивация.....	124
Хороший баланс компонентов упрощает поиск и анализ кода	124
Хорошо сбалансированные компоненты улучшают изолированность обслуживания	124
Хорошо сбалансированные компоненты позволяют распределять ответственность при обслуживании	125
8.2. Как применять рекомендацию	125
Выбор правильного концептуального уровня при распределении функциональных возможностей по компонентам	125
8.3. Типичные возражения против стремления к сбалансированности компонентов.....	127
Возражение: системы с дисбалансом компонентов отлично работают	127
Возражение: запутанность связей между компонентами не позволяет их сбалансировать	127
8.4. Дополнительные сведения	128

Глава 9. Следите за размером базы кода

9.1. Мотивация.....	131
Проект с большой базой кода, скорее всего, обречен на неудачу.....	131
Большие базы кода труднее обслуживать.....	131
Большие системы отличаются высокой плотностью дефектов.....	133
9.2. Как применять рекомендацию	134
Функциональные меры.....	134
Технические меры	134

9.3. Типичные возражения против уменьшения размеров базы кода	136
Возражение: сокращение размера базы кода снижает	
производительность разработки.....	137
Возражение: выбранный язык программирования препятствует	
уменьшению объема кода.....	137
Возражение: сложность системы заставляет дублировать код.....	138
Возражение: разделение базы кода невозможно	
из-за архитектуры платформы	138
Возражение: разделение кода приводит к дублированию.....	139
Возражение: разделение базы кода невозможно из-за тесной	
связанности	139

Глава 10. Автоматизируйте тестирование

10.1. Мотивация.....	143
Автоматизация делает тестирование повторяемым	143
Автоматизированное тестирование увеличивает эффективность	
разработки	143
Автоматизированное тестирование делает код предсказуемым	143
Тесты документируют тестируемый код	144
Разработка тестов улучшает качество кода	144
10.2. Как применять рекомендацию	145
Начало работы с jUnit	146
Общие принципы разработки хороших модульных тестов	149
Оценка охвата для определения достаточности количества	
тестов.....	154
10.3. Типичные возражения против автоматизации тестов	155
Возражение: нам нужно и ручное тестирование.....	155
Возражение: мне не разрешается писать модульные тесты	156
Возражение: зачем тратить время на модульные тесты	
при низком текущем охвате ими?	156
10.4. Дополнительные сведения	157

Глава 11. Пишите чистый код.....

11.1. Не оставляйте следов	158
11.2. Как применять рекомендацию	159
Правило 1: не оставляйте после себя грязи на уровне блоков	
кода.....	159
Правило 2: не оставляйте после себя неудачных комментариев.....	160
Правило 3: не оставляйте после себя закомментированного кода	162
Правило 4: не оставляйте после себя неиспользуемого кода	163
Правило 5: не оставляйте после себя длинных идентификаторов	163
Правило 6: не оставляйте после себя таинственных констант	164

Правило 7: не оставляйте после себя плохую обработку исключений	165
11.3. Типичные возражения против написания чистого кода	166
Возражение: комментарии являются документацией	166
Возражение: обработка исключений увеличивает объем кода	167
Возражение: почему выбраны именно эти правила?	167
Глава 12. Дальнейшие действия	168
12.1. Применение рекомендаций на практике	168
12.2. Низкоуровневые (блоки кода) рекомендации имеют более высокий приоритет, если они противоречат высокоуровневым (компоненты) рекомендациям	169
12.3. Помните, что учитывается каждое действие	169
12.4. Передовой опыт разработки будет рассмотрен в следующей книге	170
Приложение А. Как в SIG оценивается обслуживаемость	171
Предметный указатель	174