

УДК 004.971
ББК 32.972
Н72

Маттиас Нобак

Н72 Принципы разработки программных пакетов: Проектирование повторно используемых компонентов / пер. с англ. Д. А. Беликова. – М.: ДМК Пресс, 2020. – 274 с.: ил.

ISBN 978-5-97060-793-0

Существует масса литературы и онлайн-ресурсов, посвященных дизайну классов, но информацию о проектировании программных пакетов найти не так просто. Книга Маттиаса Нобака, профессионального РНР-разработчика, призвана восполнить этот пробел. В ней рассказывается о принципах повторного использования и распространения компонентов, также известных как пакеты, и предлагается ряд полезных техник по организации кода в группы любого размера. Вы узнаете о том, какие классы должны быть внутри пакета, как использовать принципы связности и зацепления, как облегчить поддержку пакета.

Издание адресовано программистам, использующим объектно-ориентированный язык для создания приложений. Представленные в книге примеры кода поясняют отдельные технические моменты и упрощают понимание материала.

УДК 004.971

ББК 32.972

All rights reserved. This work may not be translated or copied in whole or in part without the written permission of the author, except for brief excerpts in connection with reviews or scholarly analysis. Use in connection with any form of information storage and retrieval, electronic adaptation, or computer software is forbidden

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

ISBN 978-1-4842-4118-9 (англ.)
ISBN 978-5-97060-793-0 (рус.)

© 2019 Matthias Noback
© Оформление, издание, перевод,
ДМК Пресс, 2020

Оглавление

Предисловие от издательства	11
Об авторе.....	12
О техническом рецензенте.....	13
Благодарности.....	14
Введение.....	16
ЧАСТЬ I. ПРОЕКТИРОВАНИЕ КЛАССОВ	21
Глава 1. Принцип единственной ответственности.....	25
Класс со множественными ответственностями	25
Ответственности порождают причины для изменения	27
Рефакторинг: использование взаимодействующих классов.....	28
Преимущества единственной ответственности.....	29
Пакеты и принцип единственной ответственности	30
Заключение	31
Глава 2. Принцип открытости/закрытости	32
Класс, который закрыт для расширения.....	32
Рефакторинг: абстрактная фабрика.....	35
Рефакторинг: делаем абстрактную фабрику открытой для расширения.....	39
Замена или декорирование фабрики кодировщика	40
Делаем EncoderFactory открытым для расширения.....	41
Рефакторинг: полиморфизм.....	44
Пакеты и принцип открытости/закрытости.....	47
Заключение	48
Глава 3. Принцип подстановки Барбары Лисков	50
Нарушение: производный класс не имеет реализации всех методов	52
Протекающие абстракции	56
Нарушение: разные замены возвращают вещи разных типов	57

Допустимы более конкретные типы возвращаемых значений	61
Нарушение: производный класс менее снисходителен касательно аргументов метода.....	62
Нарушение: тайное программирование более специфического типа	66
Пакеты и принцип подстановки Барбары Лисков	69
Заключение	70
Глава 4. Принцип разделения интерфейса	72
Нарушение: многократные варианты использования	72
Рефакторинг: отдельные интерфейсы и множественное наследование	74
Нарушение: никакого интерфейса, просто класс.....	77
Неявные изменения в неявном интерфейсе	78
Рефакторинг: добавление интерфейсов заголовка и роли.....	79
Пакеты и принцип разделения интерфейса.....	81
Заключение	82
Глава 5. Принцип инверсии зависимостей	83
Пример инверсии зависимостей: генератор FizzBuzz	83
Делаем класс FizzBuzz открытым для расширения.....	85
Избавляемся от специфичности.....	86
Нарушение: высокоуровневый класс зависит от низкоуровневого.....	89
Рефакторинг: абстракции и конкретные реализации зависят от абстракций.....	92
Нарушение: привязка к поставщику.....	96
Решение: добавляем абстракцию и удаляем зависимость с помощью композиции	100
Пакеты и принцип инверсии зависимостей	103
Зависимость от стороннего кода: всегда ли это плохо?.....	104
Когда публиковать явный интерфейс класса.....	106
Если не все открытые методы предназначены для использования обычными клиентами.....	107
Если класс использует ввод/вывод	108
Если класс зависит от стороннего кода	109

Если вы хотите ввести абстракцию для множества конкретных вещей	111
Если вы предвидите, что пользователь захочет заменить часть иерархии объектов.....	112
Для всего остального: придерживайтесь финального класса	114
Заключение	115
ЧАСТЬ II. РАЗРАБОТКА ПАКЕТОВ.....	117
Глава 6. Принцип эквивалентности повторного использования и выпуска.....	127
Держите свой пакет в системе управления версиями	129
Добавьте файл определений.....	129
Семантическое версионирование	130
Проектирование для обратной совместимости	132
Практические правила	133
Ничего не выбрасывайте	134
Когда вы что-то переименовываете, добавьте посредника.....	134
Добавляйте параметры только в конце и со значением по умолчанию	136
Методы не должны иметь скрытых побочных эффектов.....	137
Версии зависимостей не должны быть очень строгими.....	138
Используйте объекты вместо значений примитивного типа.....	139
Используйте объекты для инкапсуляции состояния и поведения	142
Используйте фабрики объектов	144
И так далее... ..	145
Добавьте метафайлы	146
Файл README и документация	146
Установка и настройка	147
Применение	147
Точки расширения (не обязательно).....	147
Ограничения (не обязательно)	147
Лицензия.....	147

Журнал изменений (не обязательно).....	148
Примечания касательно обновлений (не обязательно)	149
Руководство по содействию (не обязательно)	150
Контроль качества	150
Качество с точки зрения пользователя	150
Что нужно сделать человеку, отвечающему за поддержку пакета	152
Статический анализ	152
Добавьте тесты	152
Настройте непрерывную интеграцию	153
Заключение	154
Глава 7. Принцип совместного повторного использования.....	156
Пласты функций	158
Очевидное расслоение	158
Скрытое расслоение	160
Классы, которые можно использовать, только когда установлен.....	162
Предлагаемый рефакторинг	166
Пакет должен быть «компонуемым»	167
Чистые релизы	169
Бонусные функции	173
Предлагаемый рефакторинг	175
Наводящие вопросы	177
Когда применять этот принцип.....	178
Когда нарушать принцип	179
Почему не следует нарушать этот принцип	179
Заключение	180
Глава 8. Принцип общей закрытости.....	181
Изменение в одной из зависимостей	182
Assetic	183
Изменение на уровне приложения	185
FOSUserBundle.....	186
Изменения, продиктованные бизнесом	189
Sylius	190
Бизнес-логика	191

Треугольник принципов связности.....	193
Заключение	195
Глава 9. Принцип ацикличности зависимостей.....	196
Зацепление: выявление зависимостей.....	196
Различные способы зацепления пакетов	197
Композиция	198
Наследование	198
Реализация	199
Использование	199
Инстанцирование	199
Использование глобальной функции.....	200
Что не следует учитывать: глобальное состояние.....	201
Визуализация зависимостей	201
Принцип ацикличности зависимостей.....	203
Проблемные циклы	204
Решения, позволяющие разорвать циклы	207
Псевдоциклы и избавление от них.....	207
Реальные циклы и избавление от них.....	210
Инверсия зависимостей.....	211
Инверсия управления	213
Посредник.....	214
Цепочка обязанностей	217
Сочетание Посредника и Цепочки обязанностей: система событий.....	218
Заключение	223
Глава 10. Принцип устойчивых зависимостей.....	224
Устойчивость.....	225
Не каждый пакет может быть высокостабильным.....	228
Нестабильные пакеты должны зависеть только от более стабильных.....	229
Оценка устойчивости	230
Снижение нестабильности и повышение устойчивости	231
Вопрос: следует ли учитывать все пакеты, какие есть во вселенной?	233
Нарушение: ваш стабильный пакет зависит от стороннего нестабильного пакета	234

Решение: используйте инверсию зависимостей.....	237
Пакет может быть как ответственным, так и наоборот	240
Заключение	242
Глава 11. Принцип устойчивых абстракций.....	243
Устойчивость и абстрактность.....	243
Как определить, является ли пакет абстрактным	246
А-метрика	246
Абстрактные вещи в стабильных пакетах	247
Абстрактность возрастает по мере роста устойчивости.....	248
Главная последовательность.....	249
Типы пакетов	251
Странные пакеты.....	252
Заключение	254
Глава 12. Заключение.....	256
Создание пакетов – это сложно	256
Повторное использование «в малом»	256
Повторное использование «в большом»	257
Разнородность программного обеспечения.....	258
Повторное использование компонентов возможно, но требует больше работы	259
Создание пакетов – выполнимая задача	259
Уменьшение воздействия первого из трех правил	260
Уменьшение воздействия второго из трех правил	261
Создание пакетов – это легко?	262
Приложение А. Полный вариант класса Page	263
Предметный указатель	273