

*Нат Тудлар, Роберт Мартин,
Дамазис Спанелис, Леван Хенна и др.*

97



**ЭТЮДОВ
ДЛЯ ПРОГРАММИСТОВ**

16 лет вместе
с профессионалами



всегда к слову

97 Things Every Programmer Should Know

Collective Wisdom from the Experts

*Pete Goodliffe, Robert Martin,
Diomidis Spinellis, Kevlin Henney and others*

Edited by Kevlin Henney

O'REILLY®

ПРОФЕССИОНАЛЬНО

97 ЭТЮДОВ для программистов

Опыт ведущих экспертов

*Пит Гудлиф, Роберт Мартин,
Диомидис Спинеллис, Кевлин Хенни и др.*

под редакцией Кевлина Хенни



*Санкт-Петербург — Москва
2012*

Серия «Профессионально»
Пит Гудлиф, Роберт Мартин,
Диомидис Спинеллис, Кевлин Хенни и др.

97 этюдов для программистов **Опыт ведущих экспертов**

Перевод С. Маккавеева

| | |
|------------------|--------------------|
| Главный редактор | <i>А. Галунов</i> |
| Зав. редакцией | <i>Н. Макарова</i> |
| Науч. редактор | <i>А. Долгушин</i> |
| Редактор | <i>М. Зислис</i> |
| Корректор | <i>О. Макарова</i> |
| Верстка | <i>Д. Орлова</i> |

Гудлиф П., Мартин Р., Спинеллис Д., Хенни К. и др.

97 этюдов для программистов. Опыт ведущих экспертов. – Пер. с англ. – СПб.: Символ-Плюс, 2012. – 256 с., ил.

ISBN 978-5-93286-198-1

Приобщитесь к мудрости экспертов и запомните то, что должен знать каждый программист, с каким бы языком и на какой платформе он ни работал. 97 кратких и очень полезных советов повысят ваш профессионализм посредством новых подходов к старым проблемам, лучших практик и разумных подсказок, предназначенных для оттачивания мастерства.

Авторы этой книги, очень опытные и признанные в отрасли специалисты, передадут вам практические знания и принципы, полезные для проектов любого типа. Статьи касаются разных тем: от рекомендаций по написанию кода до культуры, от выбора алгоритмов до гибкого программирования, от приемов реализации до профессионализма, от стиля до сущности. Новички смогут познакомиться с фундаментальными положениями, а для профессионалов сборник сможет стать отправной точкой для обсуждений.

ISBN 978-5-93286-198-1

ISBN 978-0-596-80948-5 (англ)

© Издательство Символ-Плюс, 2012

Authorized Russian translation of the English edition of 97 Things Every Programmer Should Know ISBN 978-0-596-80948-5 © 2010 O'Reilly Media Inc. This translation is published and sold by permission of O'Reilly Media Inc., the owner of all rights to publish and sell the same.

Все права на данное издание защищены Законодательством РФ, включая право на полное или частичное воспроизведение в любой форме. Все товарные знаки или зарегистрированные товарные знаки, упоминаемые в настоящем издании, являются собственностью соответствующих фирм.

Издательство «Символ-Плюс». 199034, Санкт-Петербург, 16 линия, 7,
тел. (812) 380-5007, www.symbol.ru. Лицензия ЛПН N 000054 от 25.12.98.

Подписано в печать 29.02.2012. Формат 70×90^{1/16}.

Печать офсетная. Объем 16 печ. л.

Отсутствующим друзьям посвящается

Оглавление

| | |
|--|----|
| Статьи по категориям | 13 |
| Предисловие | 19 |
| Будьте благоразумны Себ Роуз | 22 |
| Применяйте принципы функционального программирования Эдвард Гарсон | 24 |
| Выясните, как поступит пользователь (и вы – не пользователь) Жиль Колборн | 26 |
| Автоматизируйте свой стандарт форматирования кода Филип ван Лаенен | 28 |
| Красота – следствие простоты Йорн Ольмхейм | 30 |
| Прежде чем приступать к рефакторингу Раджит Аттапатту | 32 |
| Осторожно: общий код Уди Дахан | 34 |
| Правило бойскаута Роберт Мартин, известный также как «Дядюшка Боб» | 36 |
| Прежде чем пенять на других, проверь собственный код Аллан Келли | 38 |
| Тщательно выбирайте инструменты Джованни Аспрони | 40 |

| | |
|---|----|
| Пишите код на языке предметной области | 42 |
| Дэн Норт | |
| Код – это проектирование. | 44 |
| Райан Браш | |
| Важность форматирования кода | 46 |
| Стив Фримен | |
| Рецензирование кода | 48 |
| Маттиас Карлссон | |
| Пиши код с умом | 50 |
| Йехиль Кимхи | |
| Комментарий о комментариях | 52 |
| Кэл Эванс | |
| Комментируйте только то, о чем не скажет код | 54 |
| Кевлин Хенни | |
| Непрерывное обучение | 56 |
| Клинт Шэнк | |
| Удобство – не атрибут качества | 58 |
| Грегор Хоп | |
| Развертывание приложения: раннее и регулярное | 60 |
| Стив Берчук | |
| Отличайте исключения в бизнес-логике от технических | 62 |
| Дэн Берг Джонссон | |
| Больше осознанной практики | 64 |
| Джон Джаггер | |
| Предметно-ориентированные языки | 66 |
| Микаэль Хунгер | |
| Не бойтесь что-нибудь сломать | 68 |
| Майк Льюис | |
| Не прикалывайтесь с тестовыми данными | 70 |
| Род Бегби | |
| Не проходите мимо ошибки! | 72 |
| Пит Гудлиф | |

| | |
|---|-----|
| Не просто учите язык, поймите его культуру | 74 |
| Андерс Норас | |
| Не прибивайте программу гвоздями к стене | 76 |
| Верити Стоб | |
| Не полагайтесь на «автоматические чудеса» | 78 |
| Алан Гриффитс | |
| Не повторяй свой код | 80 |
| Стив Смит | |
| Этот код не трогать! | 82 |
| Кэл Эванс | |
| Инкапсулируйте поведение, а не только состояние | 84 |
| Эйнар Ландре | |
| Числа с плавающей запятой недействительны | 86 |
| Чак Эллисон | |
| Удовлетворяйте свое честолюбие через Open Source | 88 |
| Ричард Монсон-Хейфел | |
| Золотое правило проектирования API | 90 |
| Майкл Фезерс | |
| Миф о гуру | 92 |
| Райан Браш | |
| Тяжелый труд не оправдывает себя | 94 |
| Олве Маудал | |
| Как пользоваться системой отслеживания ошибок | 96 |
| Мэтт Доар | |
| Улучшайте код, удаляя его | 98 |
| Пит Гудлиф | |
| Установи меня! | 100 |
| Маркус Бэйкер | |
| Межпроцессная коммуникация влияет на время отклика приложения | 102 |
| Рэнди Стэффорд | |
| Сборка должна быть чистой | 104 |
| Йоханнес Бродуолл | |

| | |
|---|-----|
| Умей пользоваться утилитами командной строки | 106 |
| Кэрролл Робинсон | |
| Как следует изучи более двух языков программирования | 108 |
| Рассел Уиндер | |
| Знай свою IDE | 110 |
| Хейнц Кабуц | |
| Знай свои возможности | 112 |
| Грег Колвин | |
| Знай, что сохранишь в репозиторий | 114 |
| Дэн Берг Джонссон | |
| Место для больших наборов взаимосвязанных данных – в базе данных | 116 |
| Диомидис Спинеллис | |
| Учите иностранные языки | 118 |
| Клаус Маркардт | |
| Учитесь делать оценки | 120 |
| Джованни Аспрони | |
| Научитесь говорить «Hello, World» | 122 |
| Томас Гест | |
| Пусть ваш проект говорит сам за себя | 124 |
| Дэниэл Линднер | |
| Компоновщик не таит в себе никаких чудес | 126 |
| Уолтер Брайт | |
| Долговечность временных решений | 128 |
| Клаус Маркардт | |
| Интерфейсы должно быть легко использовать правильно и трудно – неправильно | 130 |
| Скотт Мейерс | |
| Пусть невидимое станет более видимым | 132 |
| Джон Джаггер | |
| Передача сообщений улучшает масштабируемость параллельных систем | 134 |
| Рассел Уиндер | |
| Послание потомкам | 136 |
| Линда Райзинг | |

| | |
|---|-----|
| Упущенные возможности применения полиморфизма | 138 |
| Кирк Пеппердин | |
| Невероятно, но факт: тестировщики – ваши друзья | 140 |
| Берк Хафнагель | |
| Один бинарный файл | 142 |
| Стив Фримен | |
| Правду скажет только код | 144 |
| Петер Зоммерлад | |
| Возьмите сборку (и ее рефакторинг) на себя | 146 |
| Стив Берчук | |
| Программируйте парами и входите в поток | 148 |
| Гудни Хаукнес, Кари Россланд и Анн Кэтрин Гэгнат | |
| Предпочитайте примитивам предметно-ориентированные типы данных | 150 |
| Эйнар Ландре | |
| Предотвращайте появление ошибок | 152 |
| Жиль Колборн | |
| Профессиональный программист | 154 |
| Роберт Мартин (Дядюшка Боб) | |
| Держите все в системе управления версиями | 156 |
| Диомидис Спинеллис | |
| Брось мышь и медленно отойди от клавиатуры | 158 |
| Берк Хафнагель | |
| Читайте код | 160 |
| Карианне Берг | |
| Читайте гуманитарные книги | 162 |
| Кейт Брэйтуэйт | |
| Почаще изобретайте колесо | 164 |
| Джейсон П. Сэйдж | |
| Не поддавайтесь очарованию шаблона Singleton | 166 |
| Сэм Сааристе | |
| Путь к повышению эффективности программ заминирован грязным кодом | 168 |
| Кирк Пеппердин | |

| | |
|---|-----|
| Простота достигается сокращением | 170 |
| Пол У. Гомер | |
| Принцип единственной ответственности | 172 |
| Роберт Мартин (Дядюшка Боб) | |
| Сначала скажите «да» | 174 |
| Алекс Миллер | |
| Шаг назад. Теперь автоматизируй, автоматизируй, автоматизируй... | 176 |
| Кэй Хорстман | |
| Пользуйтесь инструментами для анализа кода | 178 |
| Сара Маунт | |
| Тестируйте требуемое, а не случайное поведение. | 180 |
| Кевлин Хенни | |
| Тестируйте точно и конкретно | 182 |
| Кевлин Хенни | |
| Тестируйте во сне (и по выходным) | 184 |
| Раджит Атапатту | |
| Тестирование – это инженерная строгость в разработке программного обеспечения. | 186 |
| Нил Форд | |
| Думайте состояниями | 188 |
| Никлас Нильссон | |
| Одна голова хорошо, но две – часто лучше. | 190 |
| Эдриан Уайбл | |
| Две ошибки могут гасить одна другую (и тогда их трудно исправлять). | 192 |
| Аллан Келли | |
| Написание кода в духе Убунту для друзей | 194 |
| Аслам Хан | |
| Утилиты UNIX – ваши друзья. | 196 |
| Диомидис Спинеллис | |
| Правильно выбирайте алгоритмы и структуры данных. | 198 |
| Ян Кристиаан ван Винкель | |

| | |
|---|-----|
| Многословный журнал лишит вас сна | 200 |
| Йоханнес Бродуолл | |
| WET размазывает узкие места производительности | 202 |
| Кирк Пеппердин | |
| Когда программисты и тестировщики сотрудничают | 204 |
| Джанет Грегори | |
| Пишите код так, как будто вам предстоит сопровождать его всю оставшуюся жизнь. | 206 |
| Юрий Зубарев | |
| Пишите маленькие функции на основе примеров | 208 |
| Кейт Брэйтуэйт | |
| Тесты пишутся для людей | 210 |
| Джерард Мезарос | |
| Нужно заботиться о коде | 212 |
| Пит Гудлиф | |
| Ваши заказчики имеют в виду не то, что говорят | 214 |
| Нэйт Джексон | |
| Авторы. | 216 |
| Алфавитный указатель. | 244 |

Статьи по категориям

Ошибки и исправления

| | |
|--|-----|
| Прежде чем пенять на других, проверь собственный код | 38 |
| Этот код не трогать! | 82 |
| Как пользоваться системой отслеживания ошибок | 96 |
| Две ошибки могут гасить одна другую (и тогда их трудно исправлять) | 192 |

Сборка и развертывание

| | |
|--|-----|
| Развертывание приложения: раннее и регулярное. | 60 |
| Этот код не трогать! | 82 |
| Установи меня! | 100 |
| Сборка должна быть чистой | 104 |
| Пусть ваш проект говорит сам за себя. | 124 |
| Один бинарный файл | 142 |
| Возьмите сборку (и ее рефакторинг) на себя | 146 |

Правила написания и форматирования кода

| | |
|---|-----|
| Автоматизируйте свой стандарт форматирования кода | 28 |
| Важность форматирования кода | 46 |
| Рецензирование кода | 48 |
| Комментарий о комментариях | 52 |
| Комментируйте только то, о чем не скажет код | 54 |
| Пользуйтесь инструментами для анализа кода | 178 |

Принципы проектирования и техника написания кода

| | |
|--|----|
| Применяйте принципы функционального программирования | 24 |
| Выясните, как поступит пользователь (и вы – не пользователь) | 26 |
| Красота – следствие простоты | 30 |

| | |
|---|-----|
| Тщательно выбирайте инструменты | 40 |
| Пишите код на языке предметной области | 42 |
| Код – это проектирование | 44 |
| Пиши код с умом | 50 |
| Удобство – не атрибут качества | 58 |
| Отличайте исключения в бизнес-логике от технических. | 62 |
| Не повторяй свой код | 80 |
| Инкапсулируйте поведение, а не только состояние | 84 |
| Золотое правило проектирования API | 90 |
| Межпроцессная коммуникация влияет на время отклика приложения | 102 |
| Интерфейсы должно быть легко использовать правильно и трудно – неправильно | 130 |
| Передача сообщений улучшает масштабируемость параллельных систем | 134 |
| Упущенные возможности применения полиморфизма | 138 |
| Правду скажет только код | 144 |
| Предпочитайте примитивам предметно-ориентированные типы данных | 150 |
| Предотвращайте появление ошибок | 152 |
| Не поддавайтесь очарованию шаблона Singleton | 166 |
| Принцип единственной ответственности | 172 |
| Думайте состояниями | 188 |
| WET размывает узкие места производительности | 202 |

Предметно-ориентированное мышление

| | |
|--|-----|
| Пишите код на языке предметной области | 42 |
| Предметно-ориентированные языки | 66 |
| Учите иностранные языки | 118 |
| Предпочитайте примитивам предметно-ориентированные типы данных | 150 |
| Читайте гуманитарные книги | 162 |
| Думайте состояниями | 188 |
| Пишите маленькие функции на основе примеров | 208 |

Ошибки, их обработка и исключения

| | |
|--|-----|
| Отличайте исключения в бизнес-логике от технических. | 62 |
| Не проходите мимо ошибки! | 72 |
| Не прибивайте программу гвоздями к стене | 76 |
| Предотвращайте появление ошибок | 152 |
| Многословный журнал лишит вас сна | 200 |

Обучение, мастерство и опыт

| | |
|--|-----|
| Непрерывное обучение | 56 |
| Больше осознанной практики | 64 |
| Не просто учите язык, поймите его культуру | 74 |
| Удовлетворяйте свое честолюбие через Open Source | 88 |
| Миф о гуру | 92 |
| Тяжелый труд не оправдывает себя | 94 |
| Читайте код | 160 |
| Читайте гуманитарные книги | 162 |
| Почаще изобретайте колесо | 164 |

Ночное и волшебное

| | |
|--|-----|
| Не полагайтесь на «автоматические чудеса» | 78 |
| Этот код не трогать! | 82 |
| Миф о гуру | 92 |
| Умей пользоваться утилитами командной строки | 106 |
| Компоновщик не таит в себе никаких чудес | 126 |
| Тестируйте во сне (и по выходным) | 184 |
| Многословный журнал лишит вас сна | 200 |
| Пишите код так, как будто вам предстоит сопровождать его всю оставшуюся жизнь | 206 |

Производительность, оптимизация и представление

| | |
|--|-----|
| Применяйте принципы функционального программирования | 24 |
| Числа с плавающей запятой недействительны | 86 |
| Улучшайте код, удаляя его | 98 |
| Межпроцессная коммуникация влияет на время отклика приложения | 102 |
| Знай свои возможности | 112 |
| Место для больших наборов взаимосвязанных данных – в базе данных | 116 |
| Передача сообщений улучшает масштабируемость параллельных систем | 134 |
| Путь к повышению эффективности программ заминирован грязным кодом | 168 |
| Правильно выбирайте алгоритмы и структуры данных | 198 |
| WET размазывает узкие места производительности | 202 |

Профессионализм, мировоззрение и позиция

| | |
|--|----|
| Непрерывное обучение | 56 |
| Больше осознанной практики | 64 |
| Тяжелый труд не оправдывает себя | 94 |

| | |
|--|-----|
| Долговечность временных решений | 128 |
| Профессиональный программист | 154 |
| Брось мышь и медленно отойди от клавиатуры | 158 |
| Тестирование – это инженерная строгость в разработке программного обеспечения | 186 |
| Пишите код так, как будто вам предстоит сопровождать его всю оставшуюся жизнь | 206 |
| Нужно заботиться о коде | 212 |

Языки и парадигмы программирования

| | |
|--|-----|
| Применяйте принципы функционального программирования | 24 |
| Предметно-ориентированные языки | 66 |
| Не просто учите язык, поймите его культуру | 74 |
| Как следует изучи более двух языков программирования | 108 |
| Учите иностранные языки | 118 |

Рефакторинг и забота о коде

| | |
|--|-----|
| Будьте благоразумны | 22 |
| Прежде чем приступить к рефакторингу | 32 |
| Правило бойскаута | 36 |
| Комментируйте только то, о чем не скажет код | 54 |
| Не бойтесь что-нибудь сломать | 68 |
| Улучшайте код, удаляя его | 98 |
| Сборка должна быть чистой | 104 |
| Знай, что сохранишь в репозиторий | 114 |
| Долговечность временных решений | 128 |
| Послание потомкам | 136 |
| Правду скажет только код | 144 |
| Возьмите сборку (и ее рефакторинг) на себя | 146 |
| Профессиональный программист | 154 |
| Путь к повышению эффективности программ заминирован грязным кодом | 168 |
| Простота достигается сокращением | 170 |
| Написание кода в духе Убунту для друзей | 194 |
| Нужно заботиться о коде | 212 |

Повторное использование и повторение кода

| | |
|--|----|
| Осторожно: общий код | 34 |
| Удобство – не атрибут качества | 58 |
| Больше осознанной практики | 64 |

| | |
|--|-----|
| Не повторяй свой код | 80 |
| Почаще изобретайте колесо | 164 |
| Правильно выбирайте алгоритмы и структуры данных | 198 |
| WET размывает узкие места производительности | 202 |

Графики, сроки и оценки

| | |
|--|-----|
| Будьте благоразумны. | 22 |
| Код – это проектирование | 44 |
| Знай, что сохранишь в репозиторий | 114 |
| Учитесь делать оценки. | 120 |
| Пусть невидимое станет более видимым | 132 |

Простота

| | |
|---|-----|
| Красота – следствие простоты | 30 |
| Научитесь говорить «Hello, World» | 122 |
| Послание потомкам | 136 |
| Простота достигается сокращением | 170 |

Командная работа и сотрудничество

| | |
|--|-----|
| Рецензирование кода | 48 |
| Учите иностранные языки | 118 |
| Программируйте парами и входите в поток. | 148 |
| Сначала скажите «да» | 174 |
| Одна голова хорошо, но две – часто лучше | 190 |
| Написание кода в духе Убунту для друзей | 194 |
| Когда программисты и тестировщики сотрудничают | 204 |

Тесты, тестирование и тестировщики

| | |
|--|-----|
| Применяйте принципы функционального программирования | 24 |
| Код – это проектирование | 44 |
| Не прикалывайтесь с тестовыми данными | 70 |
| Золотое правило проектирования API | 90 |
| Интерфейсы должно быть легко использовать правильно и трудно – неправильно. | 130 |
| Пусть невидимое станет более видимым | 132 |
| Невероятно, но факт: тестировщики – ваши друзья | 140 |
| Тестируйте требуемое, а не случайное поведение | 180 |
| Тестируйте точно и конкретно | 182 |
| Тестируйте во сне (и по выходным) | 184 |

| | |
|---|-----|
| Тестирование – это инженерная строгость в разработке программного обеспечения | 186 |
| Когда программисты и тестировщики сотрудничают | 204 |
| Пишите маленькие функции на основе примеров | 208 |
| Тесты пишутся для людей | 210 |

Инструменты, автоматизация, среды разработки

| | |
|--|-----|
| Автоматизируйте свой стандарт форматирования кода | 28 |
| Прежде чем пенять на других, проверь собственный код | 38 |
| Тщательно выбирайте инструменты | 40 |
| Не повторяй свой код | 80 |
| Как пользоваться системой отслеживания ошибок | 96 |
| Умей пользоваться утилитами командной строки | 106 |
| Знай свою IDE | 110 |
| Место для больших наборов взаимосвязанных данных – в базе данных | 116 |
| Научитесь говорить «Hello, World» | 122 |
| Пусть ваш проект говорит сам за себя. | 124 |
| Компоновщик не таит в себе никаких чудес | 126 |
| Держите все в системе управления версиями | 156 |
| Шаг назад. Теперь автоматизируй, автоматизируй, автоматизируй... | 176 |
| Пользуйтесь инструментами для анализа кода | 178 |
| Тестируйте во сне (и по выходным) | 184 |
| Утилиты UNIX – ваши друзья | 196 |

Пользователи и заказчики

| | |
|--|-----|
| Выясните, как поступит пользователь (и вы – не пользователь) | 26 |
| Предметно-ориентированные языки. | 66 |
| Интерфейсы должно быть легко использовать правильно и трудно – неправильно | 130 |
| Невероятно, но факт: тестировщики – ваши друзья | 140 |
| Предотвращайте появление ошибок | 152 |
| Читайте гуманитарные книги | 162 |
| Ваши заказчики имеют в виду не то, что говорят | 214 |

Предисловие

Новейший компьютер способен лишь с большей скоростью усложнить древнейшую проблему отношений между людьми, и в конечном итоге участнику общения по-прежнему придется решать, что и как говорить.

Эдвард Р. Мэрроу (Edward R. Murrow)

Программистам есть, над чем думать. Языки программирования, приемы программирования, среды разработки, стили написания кода, инструменты, процессы разработки, планы работ, совещания, архитектуры программ, шаблоны проектирования, динамика командного взаимодействия, код, технические требования, дефекты, качество кода. И другое. Много чего еще.

Здесь мы находим искусство, ремесло и науку, которые простираются далеко за рамки программы. Деятельность программиста объединяет дискретный мир компьютеров и текучий мир человеческих занятий. Программисты служат связующим звеном между бизнесом с его расплывчатыми договорными истинами и выверенной, бескомпромиссной областью, где царят биты, байты и построенные на их основе пользовательские типы.

Учитывая объемы знаний, работы и разнообразие способов ее выполнения, никакой человек или источник не может претендовать на знание «истинного пути». Поэтому, опираясь на народную мудрость и накопленный опыт, книга «97 этюдов для программистов» предлагает не столько упорядоченную общую картину, сколько пеструю мозаику мнений о том, что должно быть известно каждому программисту. Она касается разных тем: от рекомендаций по написанию кода до культуры, от выбора алгоритмов до гибкого программирования, от приемов реализации до профессионализма, от стиля до сущности.

Отдельные статьи не стыкуются между собой, да и цель ставилась скорее противоположная. Ценность отдельной статьи здесь как раз в том, что она не похожа на другие. А ценность сборника в целом состоит в том, что статьи дополняют, подтверждают одна другую и даже противоречат друг другу. Они не связаны общим сюжетом: читатель сам может оценить материал, поразмышлять над ним и увязать прочитанное, сравнив новое с собственными контекстом, знаниями и опытом.

Лицензионные права

Все статьи публикуются по свободной лицензии. Они свободно доступны в Интернете под лицензией Creative Commons Attribution 3.0 License, что означает возможность использования отдельных статей в собственной работе при условии ссылки на их авторов:

<http://creativecommons.org/licenses/by/3.0/us/>

Контакты

На веб-странице книги перечислены найденные ошибки и приводятся дополнительные сведения:

<http://www.oreilly.com/catalog/9780596809485/>

Сопроводительный сайт, где опубликованы все статьи, биографии авторов и другие данные, находится по адресу:

<http://programmer.97things.oreilly.com>

Вы также можете следить за новостями и исправлениями книги в Twitter:

<http://twitter.com/97TEPSK>

Комментарии и технические вопросы, касающиеся этой книги, можно отправить электронной почтой:

bookquestions@oreilly.com

Дополнительная информация о наших книгах, Центрах ресурсов и сети O'Reilly Network приведена на нашем веб-сайте:

<http://www.oreilly.com/>

Safari® Books Online



Safari Books Online – цифровая библиотека, которая дает возможность быстро находить ответы на ваши вопросы в 7500 технических книг, справочников и видеозаписей.

Подписка Safari дает право читать любую страницу и смотреть любое видео в режиме онлайн. Читайте книги на сотовых телефонах и мобильных устройствах. Получайте доступ к новым изданиям до выхода их из печати. Получайте эксклюзивный доступ к рукописям в процессе работы и отправляйте замечания авторам. Копируйте текст примеров кода, загружайте главы, создавайте закладки и заметки, печатайте страницы – вот лишь некоторые из множества функций, экономящих ваше время.

O'Reilly Media опубликовала эту книгу в Safari Books Online. Чтобы получить полный цифровой доступ к этой книге и книгам схожей тематики, выпущенным

О’Reilly и другими издательствами, оформите бесплатную подписку на <http://my.safaribooksonline.com>.

Благодарности

Проекту «97 этюдов для программистов» прямо или косвенно отдали свое время и знания многие люди. Все они заслуживают благодарности.

Ричард Монсон-Хейфел (Richard Monson-Haefel) – редактор серии «97 Things» и редактор первой книги из этой серии, «97 Things Every Software Architect Should Know»¹, в написании которой я принимал участие. Спасибо Ричарду за идею серии и за ее открытость для потенциальных участников, а также за то, что он так энергично поддерживал мои предложения по данной книге.

Хочу поблагодарить всех тех, кто отдал свое время и силы, участвуя в создании текстов для этого проекта: как тех, чьи статьи опубликованы в этой книге, так и тех, чьи тексты не попали в нее, но опубликованы на веб-сайте. Большое количество и высокое качество материала весьма затруднили процесс окончательного отбора – жестко фиксированное в названии книги число не оставило места для дополнительных статей.

Я также благодарен за дополнительные отзывы, комментарии и предложения, авторами которых были Джованни Аспрони (Giovanni Asproni), Пол Колин Глостер (Paul Colin Gloster) и Микаэль Хунгер (Michael Hunger).

Спасибо О’Reilly за предоставленную этому проекту поддержку – от вики-хостинга, сделавшего книгу реальностью, до обеспечения всех стадий процесса публикации в бумажном виде. Сотрудники О’Reilly, которых я хотел бы особо поблагодарить: Майк Лукидес (Mike Loukides), Лорел Акерман (Laurel Ackerman), Эди Фридман (Edie Freedman), Эд Стивенсон (Ed Stephenson) и Рейчел Монахан (Rachel Monaghan).

Дело не только в том, что текст книги рождался в среде Веб: через Веб проект также приобрел известность и популярность. Спасибо всем, кто распространял сведения о нем через социальные сети, блоги и прочими путями.

Хочу также поблагодарить свою жену Кэролин за то, что привносит порядок в мой хаос, и двух моих сыновей, Стефана и Янника, за то, что часть этого порядка они вновь превращают в хаос.

Надеюсь, что эта книга станет для вас источником информации, открытий и вдохновения.

Приятного чтения!

Кевлин Хенни (Kevlin Henney)

¹ Нил Форд, Майкл Найгард, Билл де Ора и др. «97 этюдов для архитекторов программных систем». – Пер. с англ. – СПб.: Символ-Плюс, 2010.

Будьте благоразумны

Себ Роуз



В любом деле будь благоразумен и думай о последствиях.

Неизвестный

Как бы успокаивающе ни выглядел график работы в начале итерации, в какой-то момент неизбежно возникает нехватка времени. Если приходится разрываться между «сделать правильно» и «сделать быстро», часто возникает соблазн «сделать быстро» с оговоркой, что вы исправите решение позже, когда появится время. Вы совершенно искренне даете обещание именно так и поступить – даете самому себе, команде или заказчику. Но очень часто на следующей итерации возникают уже другие проблемы, которым и приходится посвящать свое внимание. Такую отложенную работу называют *техническим долгом*, и хорошего от него не жди. В своей классификации технических долгов Мартин Фаулер называет такой вид долга *умышленным техническим долгом*, и его не следует путать с *непреднамеренным техническим долгом*.¹

Технический долг подобен кредиту: в краткосрочной перспективе он выгоден, но по нему приходится выплачивать проценты до полного погашения займа. Срезая углы при написании кода, вы затрудняете как разработку новой функциональности, так и рефакторинг. Это создает благоприятную почву для появления ошибок и нестабильных *тестовых сценариев (test cases)*. Чем дольше долг существует, тем тяжелее последствия. К тому времени, как дойдут руки внести запланированные исправления, может оказаться, что на основе изначального сомнительного кода уже выстроена целая гора не вполне верных с точки зрения проектирования решений, а это значительно усложнит рефакторинг и исправление этого кода. На самом деле, к решению изначальной проблемы часто возвращаются лишь тогда, когда *уже нет выбора*, кроме как вернуться и все исправить. И зачастую к этому моменту исправление оказывается уже настолько

¹ <http://martinfowler.com/bliki/TechnicalDebtQuadrant.html>

сложным, что вы просто не можете себе позволить потерять так много времени или пойти на подобный риск.

Бывают ситуации, когда приходится идти на создание технического долга, если необходимо уложиться в срок или частично реализовать некую функциональность. Старайтесь не оказываться в таких ситуациях, однако если положение совершенно безвыходное, действуйте. Но (и это увесистое *но*) вы должны вести учет своего технического долга и гасить его как можно скорее, иначе проблемы растут, как снежный ком. И если уж вы пошли на такой компромисс, составьте карточку с заданием (task card) или создайте запись в системе учета дефектов, чтобы не забыть о проблеме.

Если вы планируете погасить свой долг на следующей итерации, потери будут минимальными. На непогашенный долг капают проценты, за которыми нужно постоянно следить, чтобы видеть реальную конечную цену. Это подчеркивает влияние технического долга проекта на его бизнес-стоимость и позволяет разумно расставлять акценты в вопросах погашения такого долга. Способ начисления и отслеживания процентов зависит от конкретного проекта, но отслеживать их должны вы.

Гасите технические долги как можно скорее. Поступать иначе – неблагоразумно.

Применяйте принципы функционального программирования

Эдвард Гарсон



Функциональное программирование недавно снова обратило на себя внимание большинства в сообществе программистов. Отчасти благодаря тому, что *эмерджентные свойства* функциональной парадигмы созвучны решению задач, возникающих в нашей отрасли в связи с ростом значимости многоядерных архитектур. И хотя данное применение, несомненно, важно, однако не оно является главным основанием для моего наставления *познать функциональное программирование*.

Овладев парадигмой функционального программирования, программист может значительно повысить качество кода, создаваемого в других контекстах. Глубокое понимание парадигмы функционального программирования и ее применение на практике помогут вам проектировать системы, обладающие гораздо большей степенью *ссылочной прозрачности (referential transparency)*.

Ссылочная прозрачность является качеством очень желательным: она предполагает, что функции неизменно дают одинаковые результаты на одинаковых входных данных независимо от места и времени обращения к этим функциям. Вычисление функции, таким образом, слабо зависит от побочных эффектов изменяющегося (*mutable*) состояния – в идеале не зависит от них вообще.

Один из главных источников дефектов в коде на императивном языке программирования – изменяемые (*mutable*) переменные. Каждому читателю наверняка приходилось разбираться, почему в каком-либо конкретном случае некоторое значение не соответствовало ожидаемому. Семантика областей видимости может препятствовать появлению таких коварных ошибок или, по крайней мере, значительно сужать возможную область их появления. Но истинной причиной их возникновения может быть сама концепция проектирования такого кода, который беспорядочно полагается на изменяемость (*mutability*).

И в этом отношении нам определенно не стоит ждать особой помощи от собственной отрасли. Вводные тексты по объектно-ориентированному программированию скрыто пропагандируют подобные конструкции. В них часто приводятся