Дзен

ПЕРЕСТРОЙКА

Как провести рефакторинг с пользой

МОВЧАН ДМИТРИЙ

Обо мне

Мовчан Дмитрий

- Team lead в Дзене
- ПК Podlodka DroidCrew
- Активный спикер
- Участвовал в создании Android Academy MSK / Global
- Пережил десятки крупных рефакторингов

Дзен

О чем будет доклад

- Определимся с понятием
- Поймем зачем нужен рефакторинг
- Когда его пора проводить
- Как его проводить

Другими словами — поговорим в основном про процессы, которые связаны с этим самым рефакторингом

Дзен

Фундамент доклада

- Книга Мартина Фаулера
- Доклады от коллег из разных сфер
- Статьи
- Собственный опыт

Дзен

— Что такое рефакторинг

Рефакторинг

- Рефакторинг представляет собой процесс такого изменения программной системы, при котором не меняется внешнее поведение кода, но улучшается его внутренняя структура

- Это способ систематического приведения кода в порядок, при котором шансы появления новых ошибок минимальны*. В сущности, при проведении рефакторинга кода вы улучшаете его дизайн уже после того, как он написан

Рефакторинг

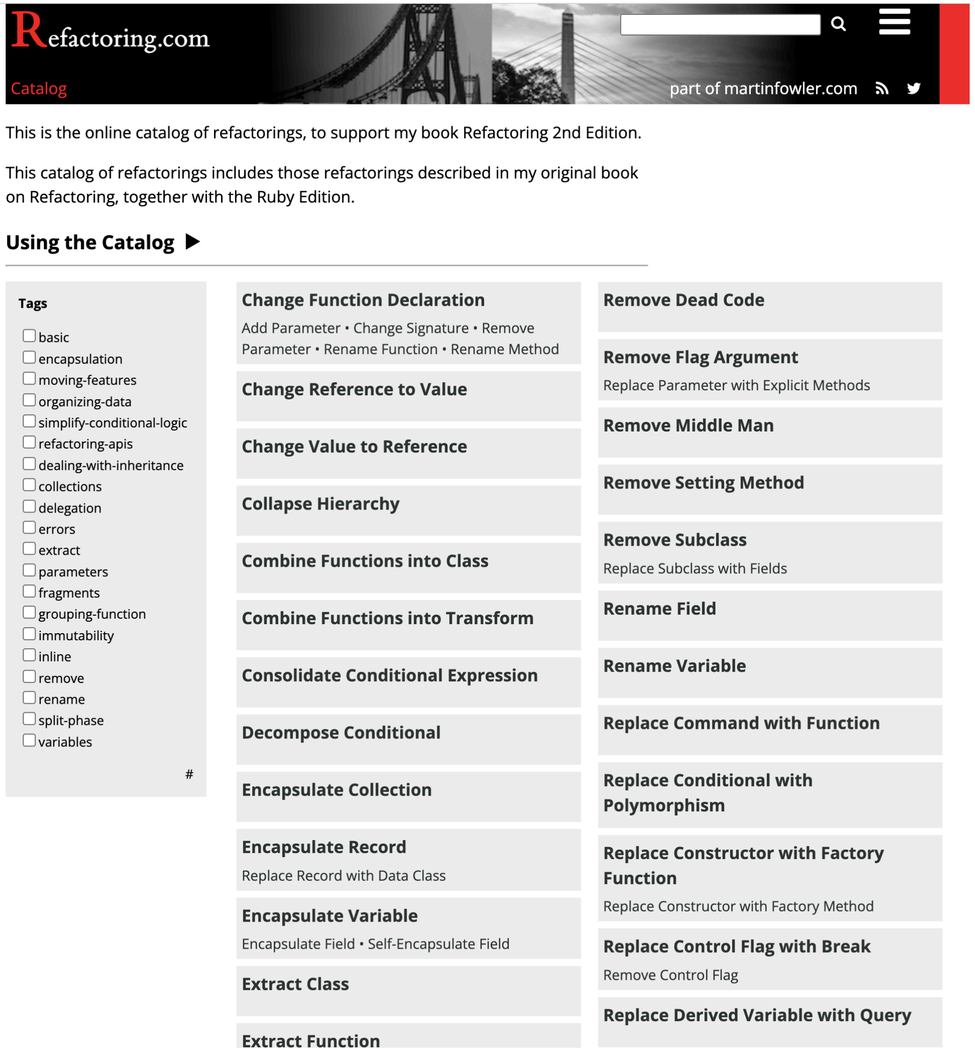
Чем вообще полезен рефакторинг?

- улучшает композицию программного обеспечения
- облегчает понимание программного обеспечения
- помогает найти ошибки
- позволяет быстрее писать программы

Антирефакторинг

Антипод рефакторинга: оптимизация производительности отдельной части кода, что в свою очередь может и ухудшить его читаемость

А вот чего не будет в докладе



Refactoring.com

Catalog

part of martinfowler.com

This is the online catalog of refactorings, to support my book Refactoring 2nd Edition.

This catalog of refactorings includes those refactorings described in my original book on Refactoring, together with the Ruby Edition.

Using the Catalog ▶

Tags

- basic
- encapsulation
- moving-features
- organizing-data
- simplify-conditional-logic
- refactoring-apis
- dealing-with-inheritance
- collections
- delegation
- errors
- extract
- parameters
- fragments
- grouping-function
- immutability
- inline
- remove
- rename
- split-phase
- variables

#

Change Function Declaration
Add Parameter • Change Signature • Remove Parameter • Rename Function • Rename Method

Change Reference to Value

Change Value to Reference

Collapse Hierarchy

Combine Functions into Class

Combine Functions into Transform

Consolidate Conditional Expression

Decompose Conditional

Encapsulate Collection

Encapsulate Record
Replace Record with Data Class

Encapsulate Variable
Encapsulate Field • Self-Encapsulate Field

Extract Class

Extract Function

Remove Dead Code

Remove Flag Argument
Replace Parameter with Explicit Methods

Remove Middle Man

Remove Setting Method

Remove Subclass
Replace Subclass with Fields

Rename Field

Rename Variable

Replace Command with Function

Replace Conditional with Polymorphism

Replace Constructor with Factory Function
Replace Constructor with Factory Method

Replace Control Flag with Break
Remove Control Flag

Replace Derived Variable with Query

www.refactoring.com 1

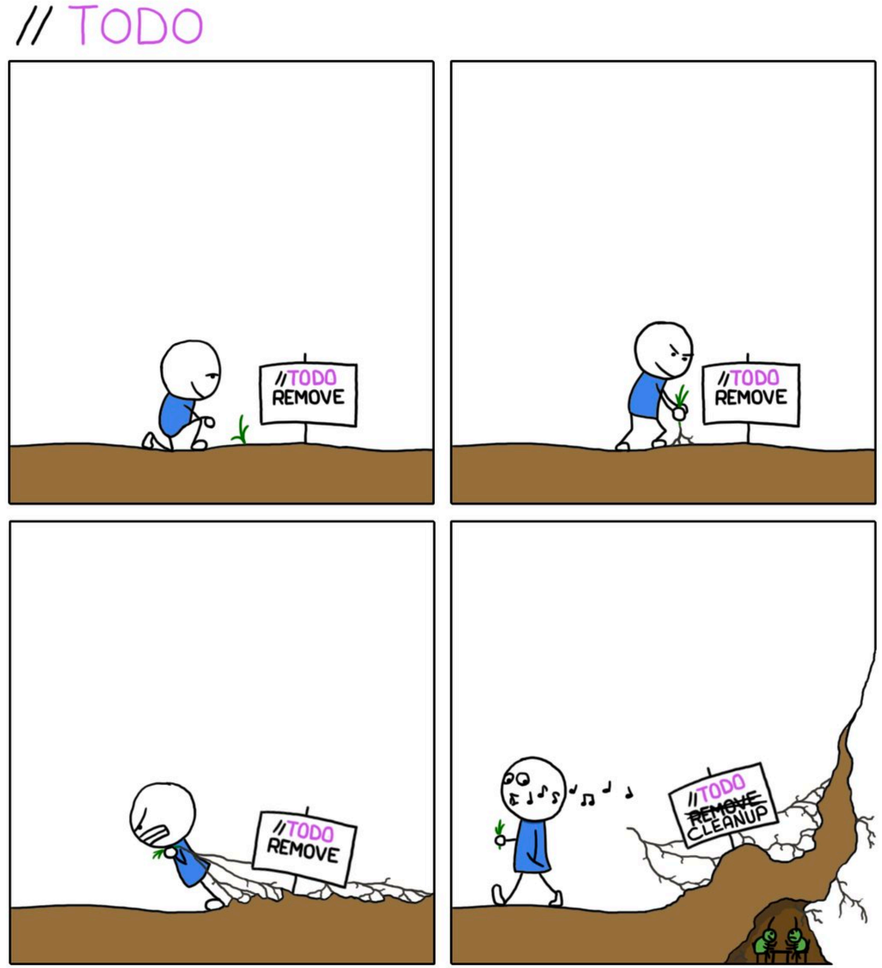
Не будет поднята тема покрытия тестов 2

Откуда это все взялось?

- Формально слово «рефакторинг» появилось только в 1990 году, но то, что оно означает, существовало и сильно раньше
- Часто любят упоминать разработчиков на Smalltalk как тех, кто начал активно применять рефакторинг

В какой момент может понадобится рефакторинг

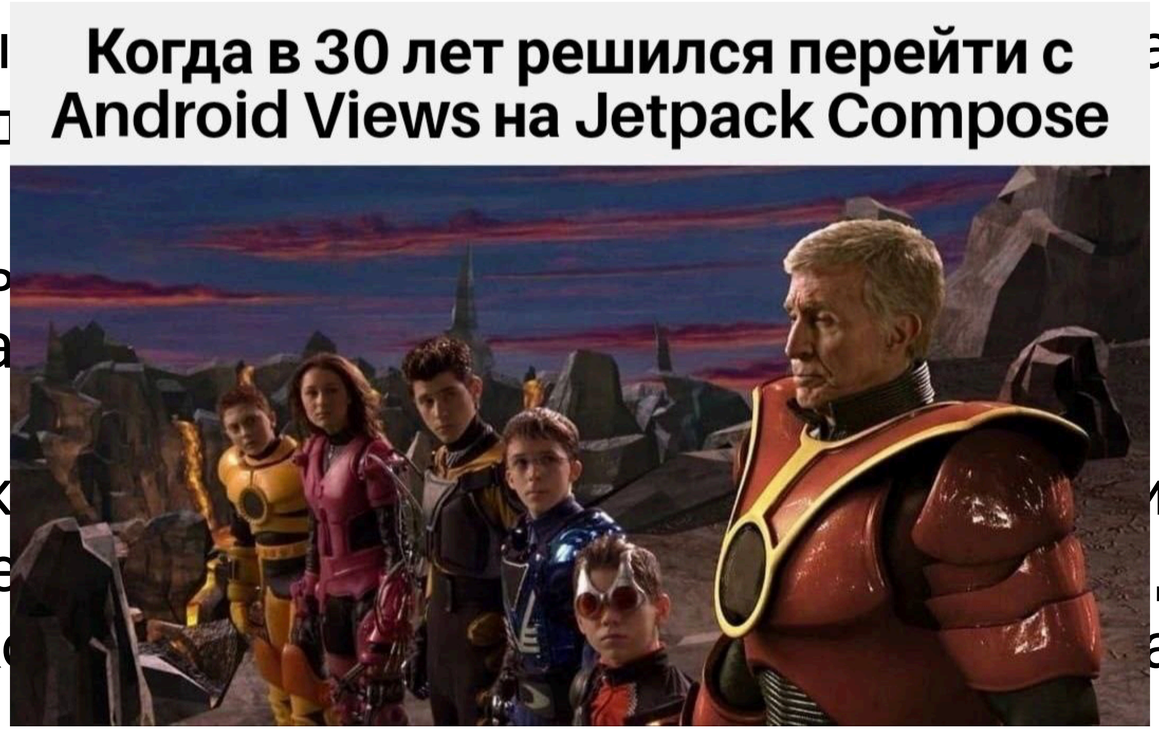
- Техдолг
- Стек технологий устарел
- Слишком часто возникают вопросы:
  «Да кто это писал? Как с этим работать?»
- «Правило трех ударов»
- Слишком сильно поменялись требования



— А можно ли было всего этого избежать?

Проблемы приходят с течением времени

- Устаревший стек
  - Проекты, запущенные в 2012 году не знали про Kotlin и Compose
- Состав и навы… …а просто нужен свежий взгляд
- Фича делалась… …было времени сдела…
- Со временем к… …истемы, соответствие е… …дизайну постепенно ухудшаются. К… …ания к хакерству.



Рефакторинг <3 проектирование

- Проектирование непрерывно осуществляется во время разработки, а не выполняется целиком заранее. При реализации системы становится ясно, как можно улучшить ее проект.

- Происходящее взаимодействие приводит к созданию программы, качество проекта которой остается высоким по мере продолжения разработки.

— Как доказать необходимость рефакторинга?

Как доказать необходимость рефакторинга?

Дзен

- Время разработки фичей слишком высоко
  - Проблемы масштабируемости
  - Неадекватные оценки на простые доработки
- Слишком много «костылей»
- Качество новых фичей
  - Фича готова, но вы же знаете что мы не можем поворачивать экран на ней?
  - Фича готова, но мы убрали все разделители, которые были на дизайнах

Почему всем не рефакторить?

Дзен

С рефакторингом связан известный риск. Он требует внести изменения в работающий код, что может привести к появлению трудно находимых ошибок в программе. Неправильно осуществляя рефакторинг, можно потерять дни, недели.

Вы начинаете копаться в коде. Вскоре обнаруживаются новые возможности модификации, и вы начинаете копать глубже. Чем больше вы копаете, тем больше вскрывается нового и тем больше изменений вы производите.

— С чего начать?

«Указать цель – лишь одна часть задачи; преобразовать код так, чтобы достичь этой цели, – другая проблема»

План рефакторинга

- А нужен ли тут рефакторинг вообще?

Дзен

90%

План рефакторинга

- А нужен ли тут рефакторинг вообще?
- Выделите ответственных
- Очертите зоны рефакторинга
  - Что будет изменено
  - Какой результат ожидается?
  - Сделайте примерную оценку
  - Декомпозируйте
  - Прототип

Дзен

А кто собственно ответственный?

- Платформенная команда
- Фиче команда
  - Рефакторинг по ходу выполнения новой фичи/доработки старой фичи

Что нужно от ответственных

Требования

- Собрать требования к куску, который рефакторится
  - Вполне вероятно что в старом проекте требования уже очень устарели и часть функционала не нужна
  - Попросите у отдела QA тест кейсы на то, что собираетесь рефакторить

Что нужно от ответственных

По поводу оценки

- Почему этой оценке нельзя доверять?

Задача	Время выполнения (произвольные интервалы)
Анализ требований	A
Создание проектного решения	B
Реализация проектного решения	C
Тестирование программного обеспечения	D
Исправление ошибок	E
Развертывание программного обеспечения	F

Что нужно от ответственных

По поводу оценки

- Почему этой оценке нельзя доверять?

Задача	Время выполнения (произвольные интервалы)
Анализ требований	A
Обсуждение результатов анализа с сотрудниками отдела	B
Создание проектного решения	C
Макетирование проектного решения	D
Оценка макетов	E
Пересмотр проектного решения	F
Реализация высокоуровневых объектов проектного решения	G
Тестирование высокоуровневой интеграции	H
Оценка системы на предмет соответствия требованиям	I
Создание компонентов системы	J
Интеграция и тестирование компонентов	K
Повторная оценка системы на предмет соответствия требованиям	L

Задача	Время выполнения (произвольные интервалы)
Тестирование комплектной системы	M
Исправление неисправностей системы в преддверии альфа-тестирования	N
Начало альфа-тестирования	O
Исправление ошибок, выявленных на этапе альфа-тестирования	P
Начало бета-тестирования	Q
Разработка стратегии развертывания	R
Исправление ошибок, выявленных на этапе бета-тестирования	S
Тестирование стратегии развертывания	T
Тестирование конечного продукта	U
Развертывание программного обеспечения	V

— Конечность

У рефакторинга есть цель

Подойдите к рефакторингу как к обычной разработке

- Сделайте необходимую декомпозицию
- Определите временные рамки, спринты

Дзен

— Прототип

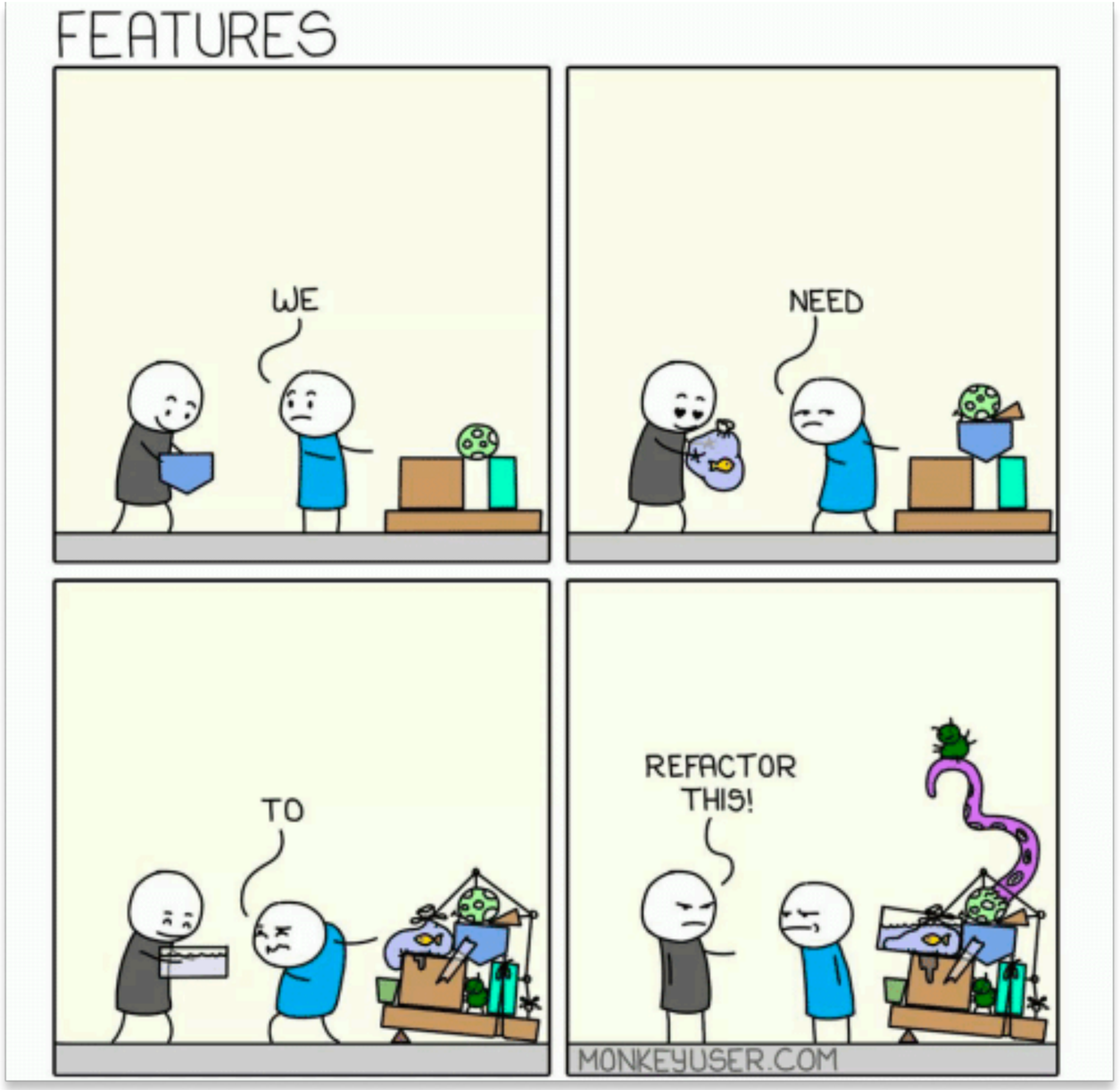
Прототип

Дзен

Попробуйте сделать изменения в отрыве от проекта
Создайте прототип, который необходим для проверки основных гипотез

- Заведите отдельный проект
- Заведите отдельный модуль
- Создайте отдельный экран
- Попробуйте добавить изменения в текущий код

— Сделка


FEATURES
WE
NEED
TO
REFACTOR THIS!
MONKEYUSER.COM

Почему нужна сделка

Дзен

Вы можете наломать дров если:

- Попытаетесь сделать рефакторинг в тайне от всех
- Попытаетесь заложить рефакторинг в стандартное выполнение фичи
- Начнете рефакторинг без должной подготовки
- Рефакторинг может провалиться на первой итерации

Сделка

Договоритесь с менеджерами

- Если рефакторинг серьезный - очень тяжело будет проводить его параллельно с разработкой фич в зоне рефакторинга
- Донесите пользу рефакторинга.
  - Отталкивайтесь не от ваших проблем (не хочу работать с Java, писать на xml), говорите как это повлияет на качество продукта и скорость разработки.

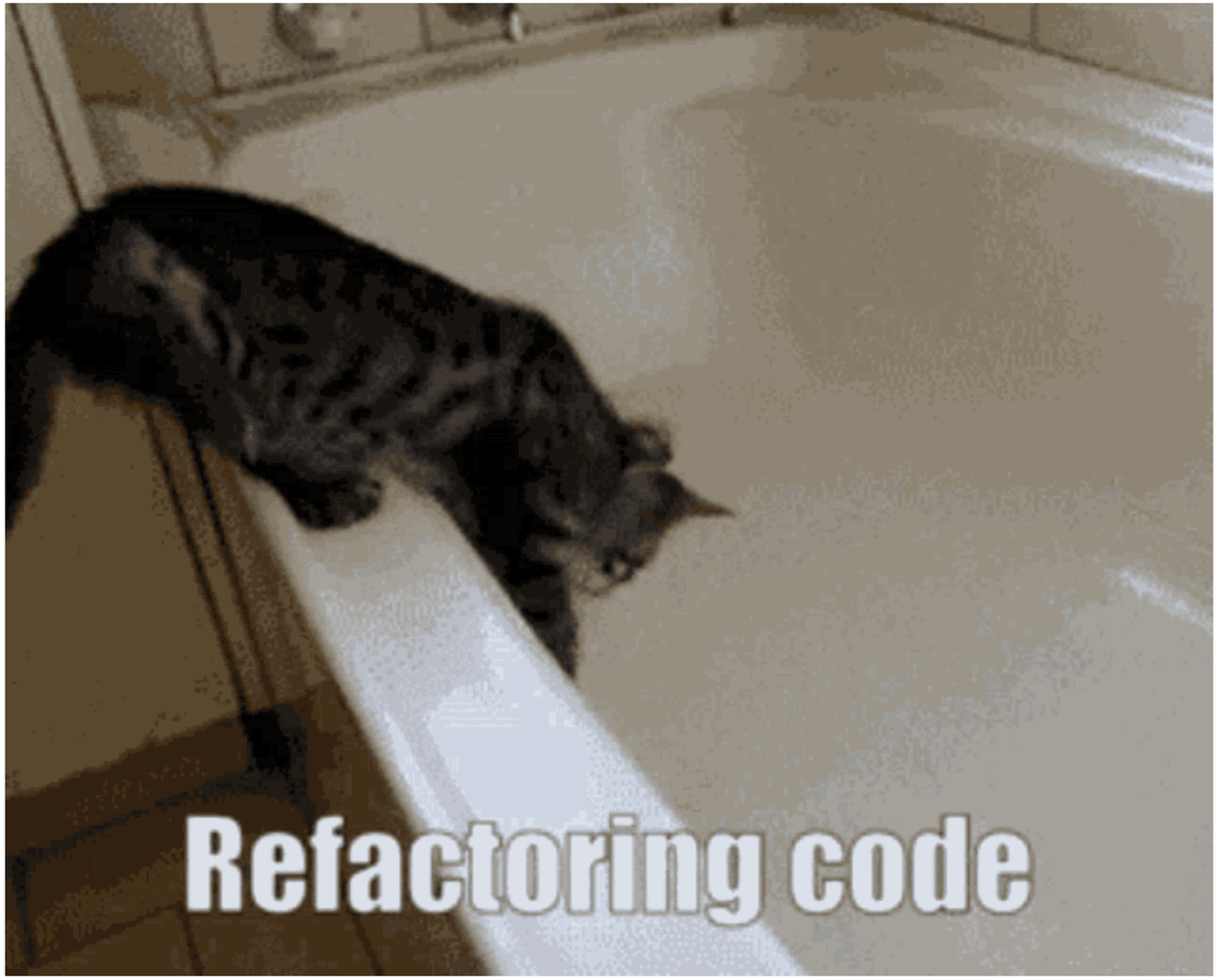
— И самое главное


Did you refactor?
Yes
What did it cost?
96 Errors
126 Warnings
0 of 12 Messages
Description
Everything

У вас всегда есть рабочий вариант

- Вы всегда можете откатиться на тот вариант, который работал
- Ведите разработку под фиче флагом, в отдельной ветке

— Почему может что-то пойти не так


Refactoring code

Основные причины

- Плохая подготовка
- Плохое понимание зоны рефакторинга
  - Обилие костылей в конкретном месте
  - Нехватка опыта в конкретных технологиях
- Нехватка «рук»
- Несогласованность с менеджментом

Дзен

«Незавершенный рефакторинг как залезание в долги. Большинству компаний для нормальной работы нужны кредиты. Однако вместе с долгами появляются и проценты, то есть дополнительная стоимость обслуживания и расширения, обусловленная чрезмерной сложностью кода. Выплату каких-то процентов можно вытерпеть, но если платежи слишком велики, вы разоритесь. Важно управлять своими долгами, выплачивая их часть посредством рефакторинга.»

Уорд Каннингем

— Рефакторинг готов

Тщательно проверьте все

- Обратитесь к помощи QA
- Воспользуйтесь тестами, если это возможно

Сделайте сессию шеринга знаний

- Проведите митап для внутренних разработчиков
- Сделайте документацию по тому, что вы сделали

Следите за метриками

- В первую очередь креш рейт
- Дальше конверсии и тд
- Производите поэтапную раскатку
- В идеале посчитайте «выхлоп» этого рефакторинга

Дзен

— Итог

В чем плюс проекта?

- Вы упростили онбординг новых разработчиков
  - У вас есть документация или записанная сессия шаринга знаний
  - Ваши фичи теперь делать проще и быстрее
  - Вы избавились от старый костылей

Что поменялось для ответственных?

- Вы большой молодец, возьмите отпуск, попросите не кидать на горящие задачи
- Вы определенно сделали проект лучше, что должно сказаться на вашей оценке перформанс ревью

Краткое резюме

- Не бойтесь рефакторить
- Договаривайтесь о том, что собираетесь делать
- Доверьте работу ответственным
- Хорошие программисты, конечно, всегда хотя бы какое-то время посвящают приведению своего кода в порядок. Они занимаются этим, поскольку поняли, что аккуратный код проще модифицировать, чем сложный и запутанный, а хорошим программистам известно, что сразу написать хороший код удается редко

Где почерпнуть информацию и вдохновение

Дзен

- Статья про рефакторинг в Яндекс музыке —

- Книги про рефакторинг:
  Мартин Фаулер «Рефакторинг. Улучшение существующего кода»
  Роберт Мартин «Чистый код: создание, анализ и рефакторинг»

Дзен

На этом все, спасибо

Контакты:

Telegram: @v1sar