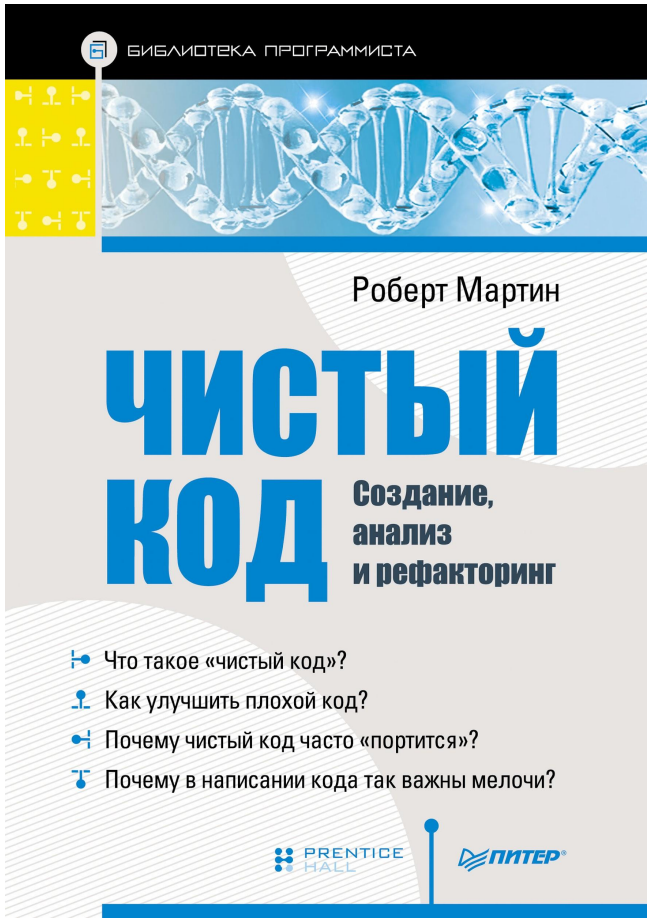
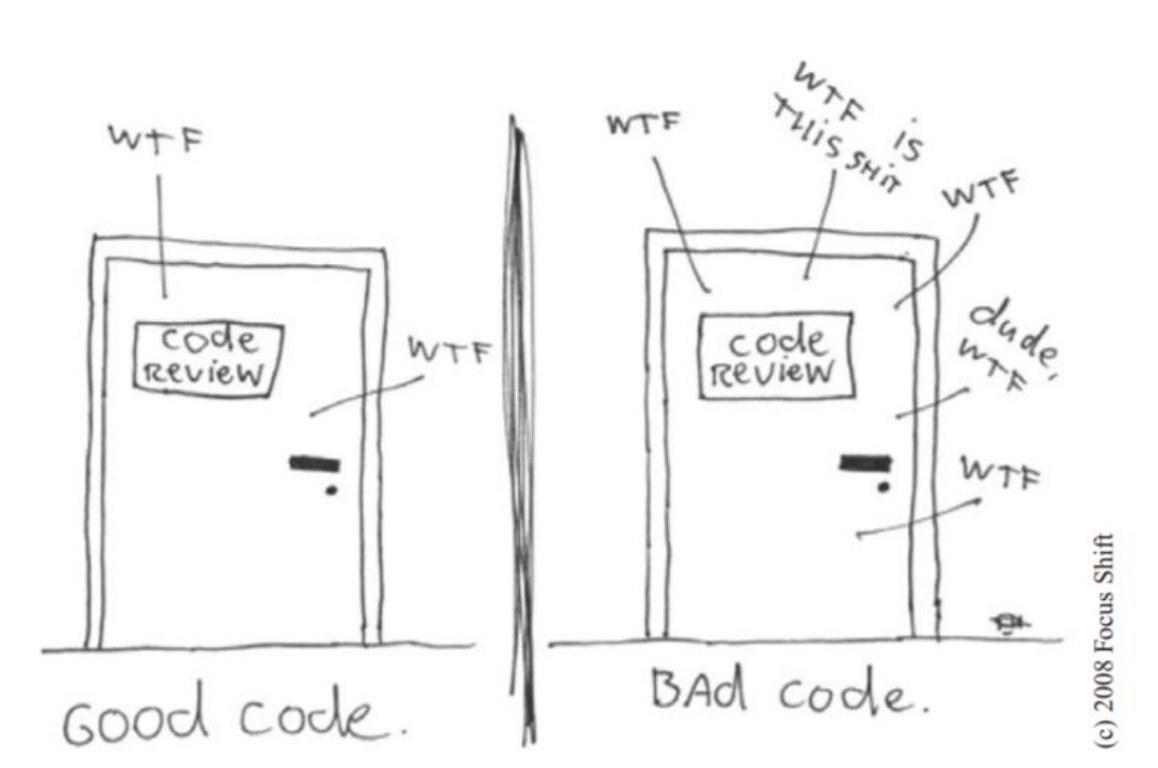

JS

Clean Code


БИБЛИОТЕКА ПРОГРАММИСТА
Роберт Мартин
ЧИСТЫЙ КОД
Создание, анализ и рефакторинг
Что такое «чистый код»?
Как улучшить плохой код?
Почему чистый код часто «портится»?
Почему в написании кода так важны мелочи?
PRENTICE HALL
ПИТЕР

What and Why



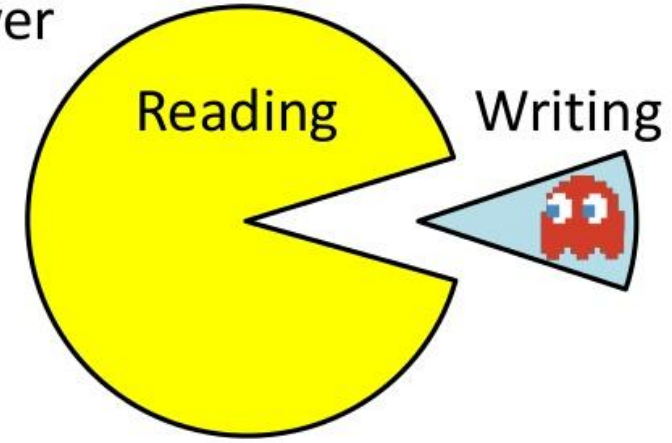
"The ratio of time spent

reading vs. writing

is well over

10:1"

– Clean Code



Stanford University

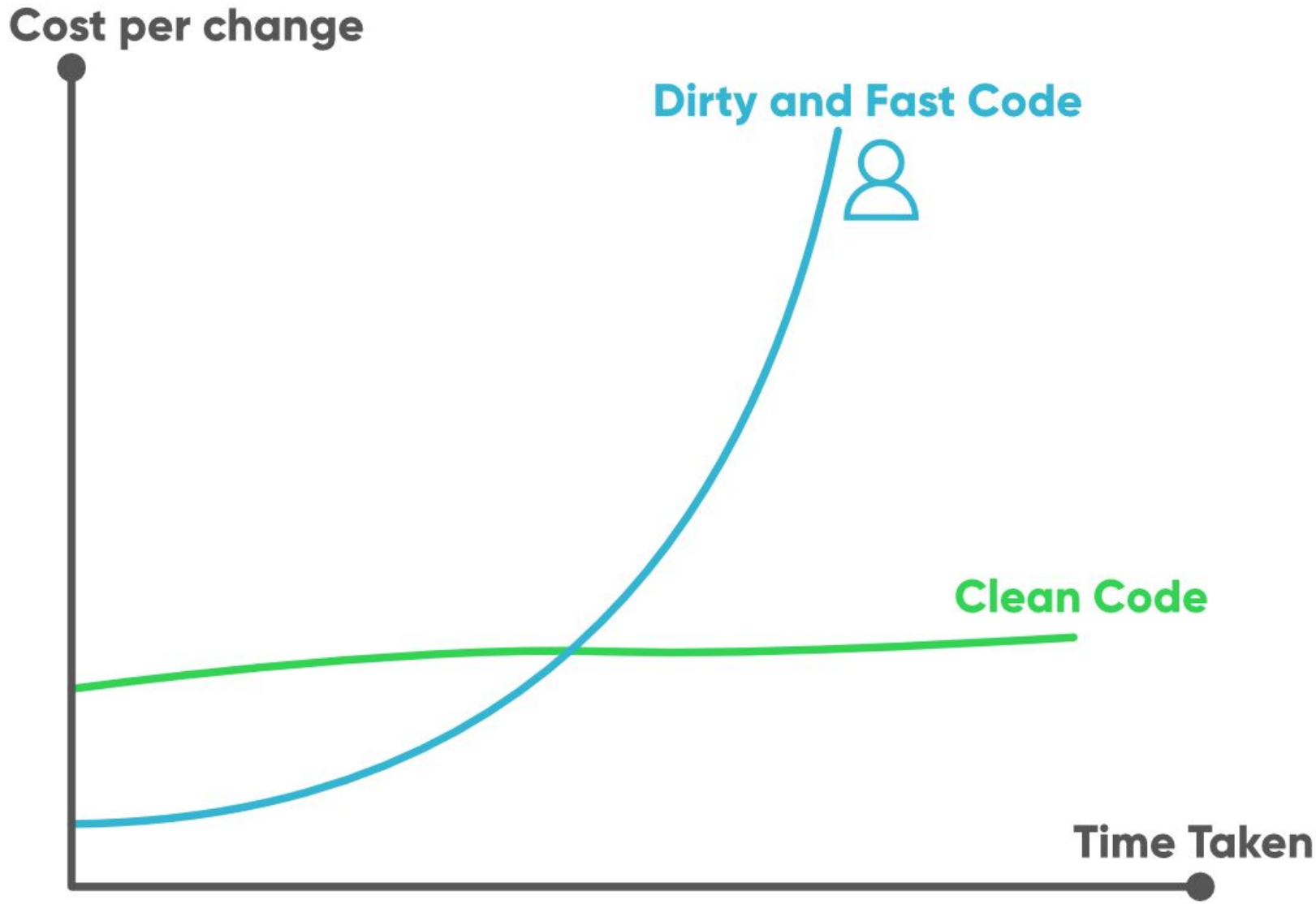
What and Why

"Indeed, the ratio of time spent reading versus writing is well over 10 to 1. We are constantly reading old code as part of the effort to write new code. ...[Therefore,] making it easy to read makes it easier to write."

— **Robert C. Martin, Clean Code: A Handbook of Agile Software Craftsmanship**

tags: agile, clean-code, computer-programming, software-engineering

Read more quotes from **Robert C. Martin**


Cost per change
Dirty and Fast Code
Clean Code
Time Taken

What and Why

Care

Clean code always looks like it was written by someone who cares

Michael Feathers

Bjarne Stroustrup, inventor of C++ and author of *The C++ Programming Language*

I like my code to be elegant and efficient. The logic should be straightforward to make it hard for bugs to hide, the dependencies minimal to ease maintenance, error handling complete according to an articulated strategy, and performance close to optimal so as not to tempt people to make the code messy with unprincipled optimizations. Clean code does one thing well.

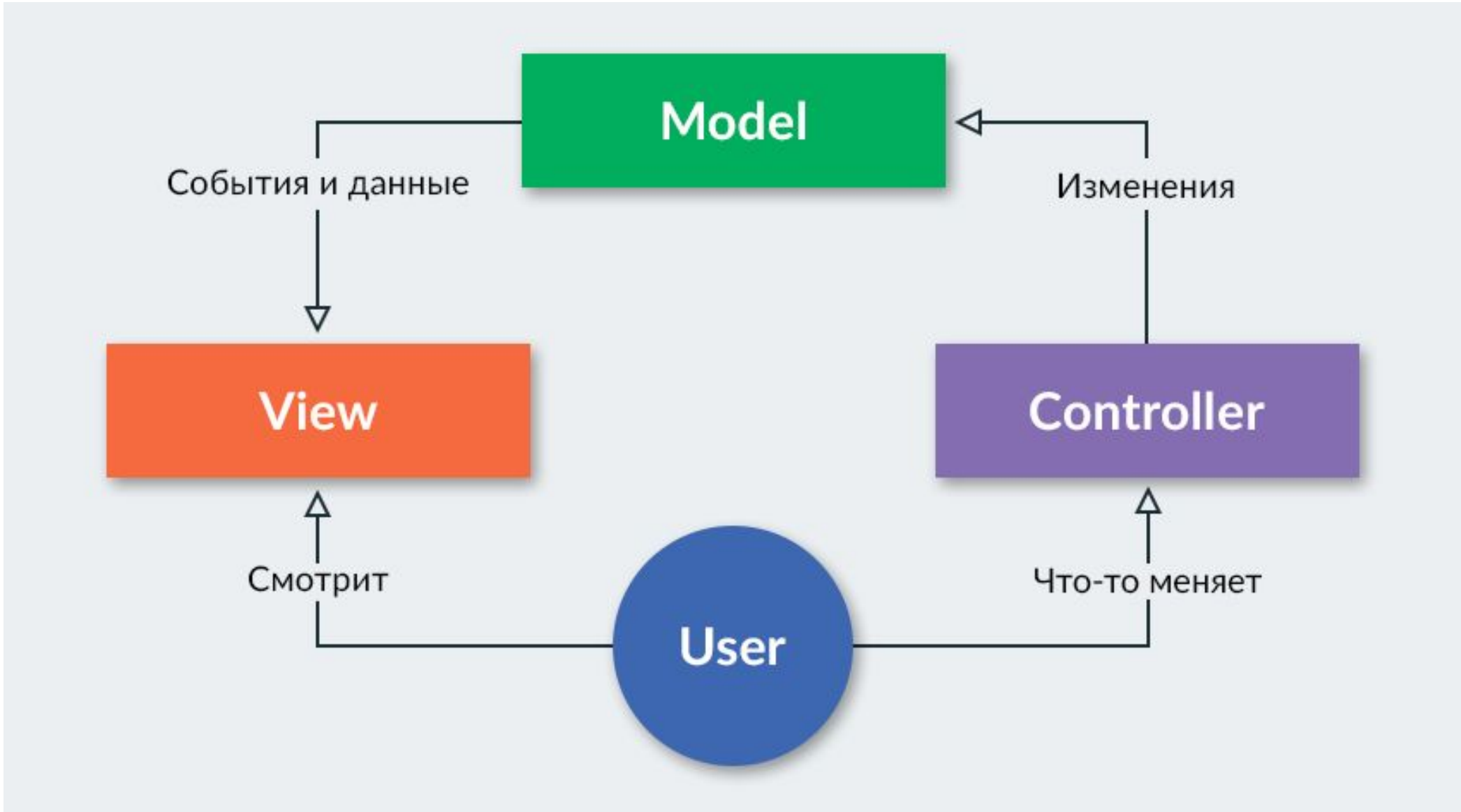
Содержательные имена

Имена должны отражать намерения

70% клин кода – это нормально названные имена

MVC (Model View Controller)

- Принцип разделения ответственности.
- Model – обработка данных (состояние, бизнесс логика, каркас управления данными бизнеса).
- View – визуальный интерфейс, его состояние и поведение.
- Controller – прослойка логики управления между View, Model и старонними сервисами.
- View и Model могут быть синхранизированны, но не могут обращаться с друг другом напрямую, это нужно делать через контроллер.

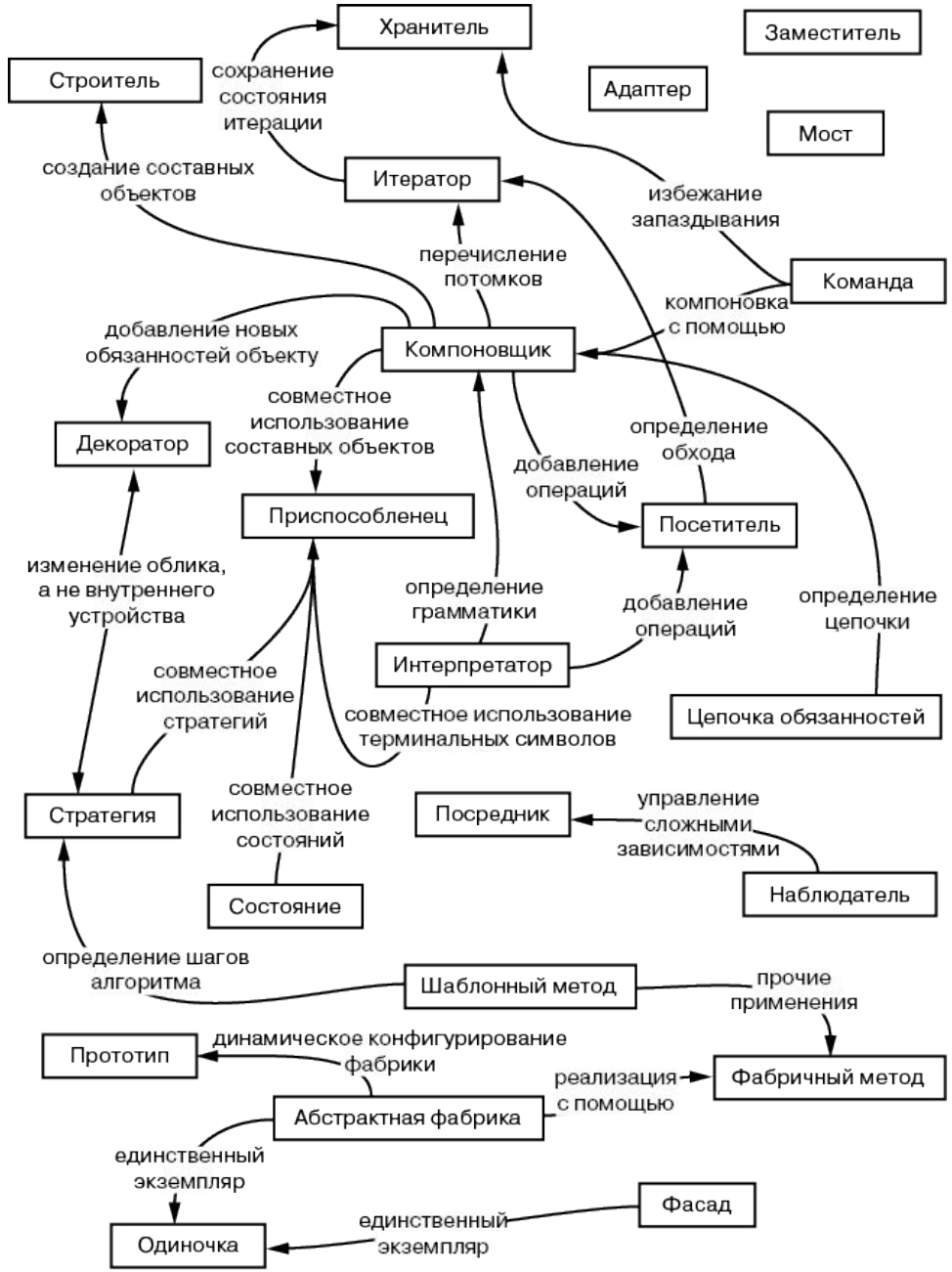

Model
События и данные
Изменения
View
Controller
Смотрит
Что-то меняет
User

SOLID

- Single Responsibility (Для каждого класса должно быть определено единственное назначение. Все ресурсы, необходимые для его осуществления, должны быть инкапсулированы в этот класс и подчинены только этой задаче.)
- Open/Closed (программные сущности … должны быть открыты для расширения, но закрыты для модификации)
- Liskov Substitution (объекты в программе должны быть заменяемыми на экземпляры их подтипов без изменения правильности выполнения программы)
- Interface segregation (много интерфейсов, специально предназначенных для клиентов, лучше, чем один интерфейс общего назначения)
- Dependency inversion principle (Зависимость на Абстракциях. Нет зависимости на что-то конкретное)

GOF паттерны

- Паттерны – хорошее решение популярной проблемы
- Паттерны не являются конечными решениями. Любой паттерн необходимо адаптировать под свою программу.
- Паттерны делятся на структурные, пораждающие и поведенческие)
- Отличный ресурс для изучения паттернов -https://refactoring.guru/


Хранитель
Заместитель
Строитель
сохранение состояния итерации
Адаптер
Мост
создание составных объектов
Итератор
избежание запаздывания
перечисление потомков
Команда
компоновка с помощью
добавление новых обязанностей объекту
Компоновщик
совместное использование составных объектов
определение обхода
Декоратор
добавление операций
Посетитель
Приспособленец
изменение облика, а не внутреннего устройства
определение грамматики
добавление операций
определение цепочки
Интерпретатор
совместное использование стратегий
совместное использование терминальных символов
Цепочка обязанностей
Стратегия
совместное использование состояний
Посредник
управление сложными зависимостями
Состояние
Наблюдатель
определение шагов алгоритма
Шаблонный метод
прочие применения
Прототип
динамическое конфигурирование фабрики
Фабричный метод
реализация с помощью
Абстрактная фабрика
единственный экземпляр
Фасад
Одиночка
единственный экземпляр

Homework

- Следовать практикам чистого кода при написании логики календаря. (Если на ваш взгляд код не является чистым, его необходимо отрефакторить [сделать читабельным и чистым не меняя логику]).
- При написании каледаря следовать принципу разделения ответственности (SRP), желательно использовать MVC паттерн для разделения ответственности на высоком уровне абстракции.