*Урок 8. Антипаттерны*

# едение

**Антипаттерны** – это паттерны наоборот или «вредные советы».

**Антипаттерны разработки (development antipatterns).** Самые простые и говорят о том, как мы пишем код.

**Архитектурные антипаттерны (architectural antipatterns).** Их труднее обнаружить. И это скорее сделает опытный разработчик-архитектор. Они в архитектуре – в программе в целом и ее частях.

**Организационные антипаттерны (managerial antipatterns).** Это немного про другое – про организацию командной разработки.

Нас интересуют прежде всего антипаттерны блоков 1 и 2.

# Антипаттерны разработки

### Магические числа (Magic numbers)

(Визуально непонятно что означает каждый из параметров.)

**В нашем коде нет.**

### Спагетти-код (Spaghetti Code)

(Код похож на миску лапши. )

**Изначально код был простым, но в процессе обучения мы его усложняли.**

**Нужно его немного доработать, чтобы он стал более простым и понятным.**

### Лазанья-код (Lasagna Code)

(В коде слишком много слоев. )

**Наш код приемлем.**

### Слепая вера (Blind Faith)

(Недостаточная проверка входных данных.)

**В код нужно добавить валидатор на проверку правильности вводимых данных.**

**Общий для всех разделов блок валидации .**

### Шифрованный код (Cryptic code)

(Когда переменные называются неправильно.)

**Мы не поленились придумать названия .**

### Жёсткое кодирование (Hard code)

( Конкретные данные, которые мы «забили» в программу.)

**В разделе с полями БД есть жесткая типизация.**

### Мягкое кодирование (Soft code)

(Встречается у любителей все сделать супергибко.)

**У нас учебный код.**

### Поток лавы (Lava flow)

(Это когда ошибки в одних частях кода несут в себе ошибки в других частях. )

**Из за того, что наш код учебный и писался по мере добавления материала. Соответственно требуется оптимизация.**

## В ООП

### Боязнь размещать логику в объектах предметной области

**Так как логику мы получали от преподавателя — она у нас есть.**

Должно быть – «толстые модели, тонкие контроллеры, тупые шаблоны».

### Божественный объект (God Object)

(Отвечающий буквально за все! )

**У нас есть Engine , но он не перегружен и т. к. содержит логику — оправдан.**

### Полтергейст (Poltergeist)

(Объект, который, то есть, то нет. )

**В процессе обучения остались для дальнейшей доработки.**

### Сплошное одиночество (Singletonitis)

**class Logger(metaclass=SingletonByName)**

### Приватизация (Privatization)

(Это когда мы используем двойное подчеркивание.)

**Отсутствует.**

## Методологические антипаттерны

Переходим к методологическим антипаттернам. Т.е. что мы делаем не так в процессе разработки.

### Программирование методом копирования – вставки (Copy – Paste programming)

**Т.к. проект учебный — немного использовал.**

### Золотой молоток (Golden Hammer)

(Штука, которая решает все задачи. )

**Под определённую задачу мы писали классы.**

### Фактор невероятности (Improbability factor)

-.

### Преждевременная оптимизация (Premature optimization)

**Не оптимизировали.**

### Изобретение велосипеда (Reinventing the wheel)

**Я не волшебник — я только учусь.**

### Изобретение квадратного колеса (Reinventing the square wheel)

**В процессе обучения написали проект с использованием проверенных колес.**

# Архитектурные антипаттерны

Переходим к архитектурным антипаттернам

## Инверсия абстракции (Abstract Inversion)

**Функциональность не скрыли.**

## Большой комок грязи (Big ball of mud)

**Код структурирован. Применяли известные шаблоны.**

## Затычка на ввод (Input kludge)

**Не реализован.**

## Волшебная кнопка (Magic pushbutton)

**В проекте придерживались - «толстая модель, тонкий контроллер».**

## Членовредительство (Mutilation)

**Объекты можно переделать под любые задачи.**

## Дымоход предприятия (Stovepipe Enterprise)

**У нас всё индивидуально.**

## Дымоход системы (Stovepipe System)

-

## Путаница (Jumble)

**Нет путаницы в проекте.**