




Принципы программирования

*ФИО: Жанар Бозоева
Группа: Python Evening Vol.6
Дата: 07-Июля-2020*



Содержание

- Основное понимание о принципах программирования
 - Ключевые принципы
 - Основные задачи
 - Вопросы & Ответы
- 

Признаки плохого кода

Жесткость

01



02



Неподвижность

Ненужная
сложность

03



04



Ненужная
повторяемость

Хрупкость

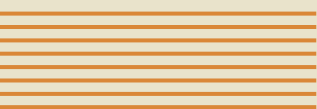
05



06



Плохая
читабельность



Основные принципы программирования

Duplication is evil

01

DRY or DIE

Don't repeat yourself

SOLID

03

02

KISS

Keep it simple, stupid!

YAGNI

You aren't going to need it

04

DRY / DIE

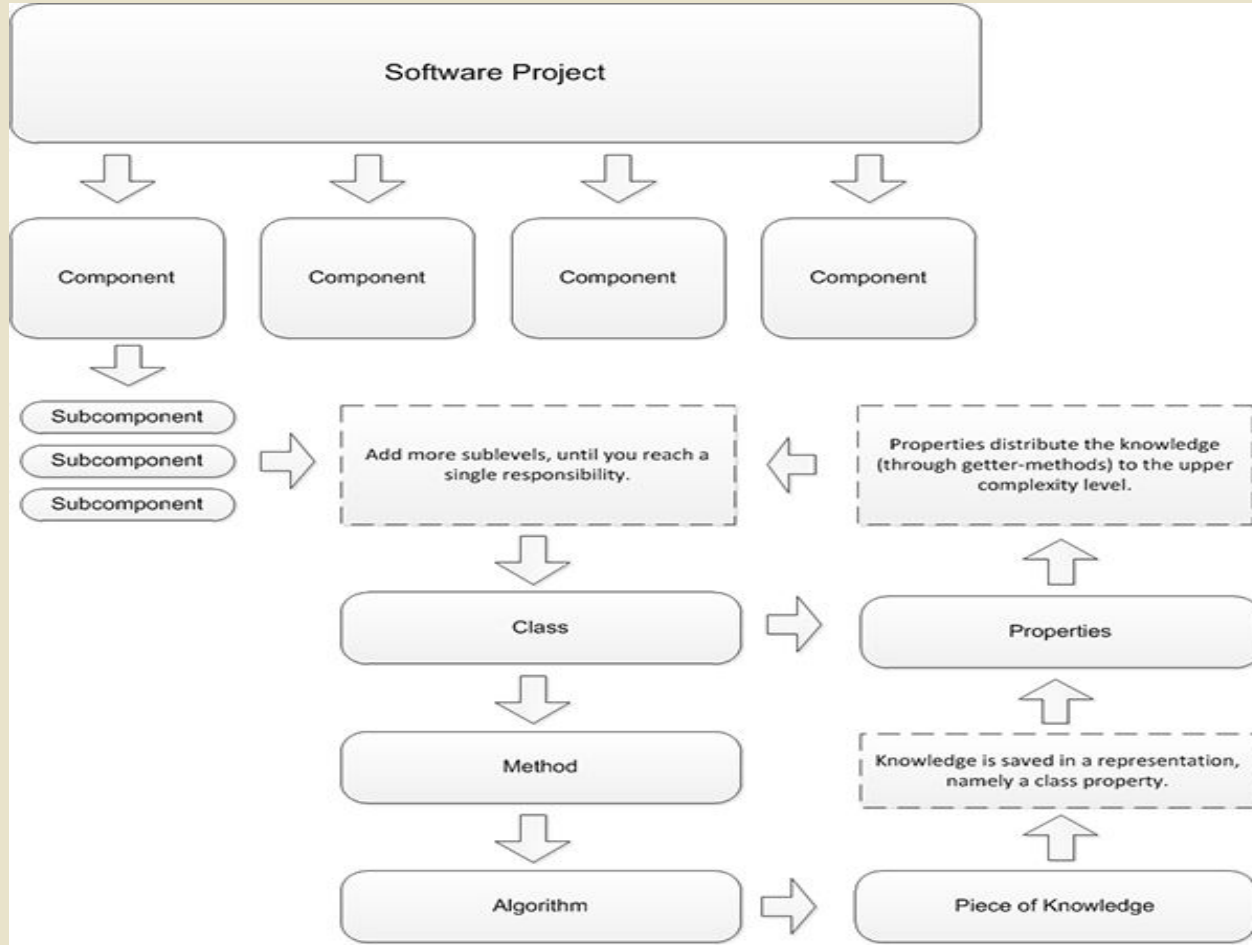
| Не повторяй сам себя

Основная задача — упростить чтение и понимание кода

- Приводит в модульному построению кода
- Разделяет логику на представления
- Требуется изначального четкого, надежного представления данных в системе



DRY / DIE



KISS

| Оставьте код простым и тупым

Основная задача - представлять максимально простую и понятную архитектуру кода

- Предотвращает необоснованную перегруженность ненужными функциями
- Позволяет разрабатывать решения, которые просты в использовании и в сопровождении.
- Декомпозиция чего-то сложного на простые составляющие — это архитектурно верный подход (тут **KISS** перекликается с DRY);



YAGNI

| Вам это не понадобится

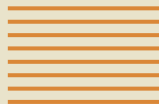
| Лучший код — ненаписанный код.

Основная задача - реализовать только поставленные задачи и отказаться от избыточного функционала

- «Бесплатных» функций в программных продуктах не бывает
- Разработчики не должны тратить своё оплачиваемое время на реализацию того, что не требуется.
- «Бонусные» возможности увеличивают вероятность ошибок и усложняют взаимодействие с продуктом

Do We Really Need That?

SOLID



Включает в себя 5 базовых принципов, предложенные **Робертом Мартином**:

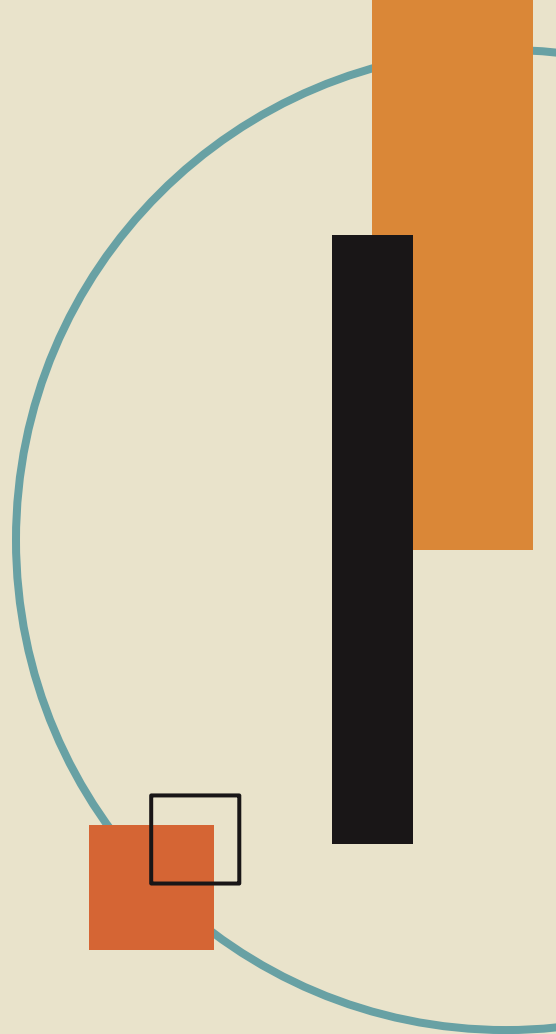
- **(S)** Single Responsibility Principle (принцип единственности ответственности)
- **(O)** Open/Closed Principle (принцип открытости/закрытости)
- **(L)** Liskov Substitution Principle (принцип подстановки Барбары Лисков)
- **(I)** Interface Segregation Principle (принцип разделения интерфейса)
- **(D)** Dependency Inversion Principle (принцип инверсии зависимостей)



Принцип единства ответственности

| «Одно поручение. Всего одно.»

- Каждый метод структуры кода должна выполнять какую-то одну единственную задачу
- Направлен на то, чтобы сделать код гибким и при изменении одних подсистем не затрагивались другие



2) Принцип открытости/закрытости

| Система должна быть открыта для расширения и закрыта для модификации.

- Когда требуется новая функциональность, мы должны не изменять существующий код, а наоборот написать новый, который будет использовать уже существующий.

3) Принцип подстановки Барбары Лисков

| Наследующий класс должен дополнять, а не замещать поведение базового класса.

- Подтипы должны дополнять базовые типы

4) Принцип разделения интерфейса

- Не все методы и свойства которые используются востребованы. Таким образом, интерфейс не должен получиться избыточным или "жирным".



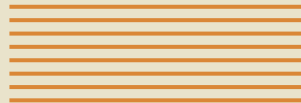
4) Принцип инверсии зависимостей

- Модули верхних уровней не должны зависеть от модулей нижних уровней, а оба типа модулей должны зависеть от абстракций;
- Абстракции не должны зависеть от деталей. Детали должны зависеть от абстракций.
- Позволяет реализовывать высокоуровневые компоненты без встраивания зависимостей от конкретных низкоуровневых классов





Спасибо за внимание!



Вопросы?

