# АНТИПАТТЕРНЫ

## BLOATERS (РАЗДУВАЛЬЩИКИ)

### Длинный метод

Метод содержит слишком большое число строк кода. Длина метода более 10 строк должна начинать вас беспокоить.

Если хочется что-то закомментить, лучше вынести этот кусок в отдельный метод

### Большой класс

Класс содержит множество полей/методов/строк кода. Часто как следствие того, что класс расфокусирован (low cohesion)

### Одержимость элементарными типами

Использование элементарных типов вместо маленьких объектов для небольших задач (например, валюта, диапазоны, специальные строки для телефонных номеров и т.п.)

Использование констант для кодирования какой-то информации (например, константа USER\_ADMIN\_ROLE = 1

Использование строковых констант в качестве названий полей в массивах.

### Длинный список параметров

Количество параметров метода больше 4

### Группы данных

Иногда в разных частях кода встречаются одинаковые группы переменных (например, параметры подключения к базе данных).

Решение: группы следует превращать в самостоятельные классы.

## OBJECT-ORIENTATION ABUSERS (Нарушители ООП)

### Операторы switch

У вас есть сложный оператор switch или последовательность if-ов.

Решение: каждой ветке можно создать свой метод\класс и вынести туда логику.

### Временное поле

Временные поля – это поля, которые нужны объекту только при определённых обстоятельствах. Только тогда они заполняются какими-то значениями, оставаясь пустыми в остальное время.

Решение — сделать локальные переменные полями нового класса

### Отказ от наследства

Если подкласc использует лишь малую часть унаследованных методов и свойств суперкласа, это является признаком неправильной иерархии. При этом ненужные методы могут просто не использоваться либо быть переопределёнными и выбрасывать исключения.

Решение: создав второй класс, заменить наследование делегированием, в котором второй класс будет пользоваться методами первого

### Альтернативные классы с разными интерфейсами

Два класса выполняют одинаковые функции, но имеют разные названия методов — программист мог не заметить, что похожий класс уже существует.

Решение: попробовать расформировать один из классов, если не получается, вынести повторяющиеся методы в суперкласс

## CHANGE PREVENTERS (утяжелители изменений)