Anti-patterns

Constant interface, God object, Yo-yo problem

Constant interface

Проблема

Использование интерфейса исключительно для объявления констант.

Нарушение одного из принципов ООП: Инкапсуляции.

```java
public interface UsefulConstants {
    public static final int FIRST_USEFUL_CONSTANT = 0;
    public static final int SECOND_USEFUL_CONSTANT = 1;
    ...
}

public class First implements UsefulConstants {
    public void firstMethod() {
        int value = FIRST_USEFUL_CONSTANT;
        ...
    }
    public static final int CONSTANT = 2 * SECOND_USEFUL_CONSTANT;
    ...
}

public class Second implements UsefulConstants {
    public void secondMethod() {
        int key = SECOND_USEFUL_CONSTANT;
        ...
    }
    ...
}
```

Что делать?

```
public final class UsefulConstants {

    private UsefulConstants()
    {
        // restrict instantiation
    }
    public static final int FIRST_USEFUL_CONSTANT = 0;
    public static final int SECOND_USEFUL_CONSTANT = 1;
    ...
}
```

```java
import static UsefulConstants.FIRST_USEFUL_CONSTANT;        // import static UsefulConstants.*
import static UsefulConstants.SECOND_USEFUL_CONSTANT;

public class First {
    public void firstMethod() {
        int value = FIRST_USEFUL_CONSTANT;
        ...
    }
    public static final int CONSTANT = 2 * SECOND_USEFUL_CONSTANT;
    ...
}

import static UsefulConstants.SECOND_USEFUL_CONSTANT;

public class Second {
    public void secondMethod() {
        int key = SECOND_USEFUL_CONSTANT;
        ...
    }
    ...

    }
}
```

God Object

Божественный объект

- Что это? Объекты, имеющие много зависимостей и ответственностей
- Нарушают принцип единственной ответственности
- Затрудняют сопровождение, отладку, работу с тестами

Как избежать?

Разбивать ответственность по мелким классам

Следить за зависимостями между классами, методами, не связанными с основной деятельностью

Избегать слишком больших классов с большим количеством зависимостей

Yo-Yo Problem

Проблема

Возникает, когда необходимо разобраться в программе, иерархия наследования и вложенность вызовов методов которой очень длинна и сложна

Как избежать?

- Стараться использовать композицию там, где можно обойтись без наследования
- Документировать слои иерархии наследования