Рефакторинг (принципы + паттерны)

**Рефакторинг** – это перепроектирование кода, которое ставит целью не изменение внешнего поведения программы, а перепроектирование ее внутренних алгоритмов и структуры с целью оптимизации и упрощения восприятия (понимания ее работы).

Принципы построения / написания программ:

**KISS** – **keep it short and simple** – нужно упрощать а не усложнять – это касается и архитектуры/дизайна и самого кода внутри. Что касается кода внутри – чистый код лучше, чем умный код. Упрощение не всегда с точки зрения укорочения строчек записи – нужно помнить, что этот код будут читать другие и в т.ч. сам через какое-то время.

**YAGNI** – **you ain’t gonna need it** - Все что не предусмотрено техническим заданием проекта, не должно быть в нем.

**DRY** – **don’t repeat yourself** - Все что Вы пишете в проекте, должно быть определено только один раз.

**SOLID**

**Single responsibility principle** –

*- Принцип единственной ответственности:*

Один класс должен решать только какую-то одну задачу. Он может иметь несколько методов, но они должны использоваться лишь для решения общей задачи. Все методы и свойства должны служить одной цели. Если класс имеет несколько назначений, его нужно разделить на отдельные классы.

Или - 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 principle** –

*- Принцип открытости/закрытости:*

Программные сущности (классы, модули, функции и прочее) должны быть расширяемыми без изменения своего содержимого, т.е. открыты для расширения, но закрыты для модификации. Если строго соблюдать этот принцип, то можно регулировать поведение кода без изменения самого исходника.

**Liskov substitution principle** –

*- Принцип подстановки Барбары Лисков****:***

Объекты в программе должны быть заменяемыми на экземпляры их подтипов без изменения правильности выполнения программы или подкласс/производный класс должен быть взаимозаменяем с базовым/родительским классом. В оригинале: Функции, использующие указатели ссылок на базовые классы, должны уметь использовать объекты производных классов, даже не зная об этом.??))

**Interface segregation principle** –

*- Принцип разделения интерфейсов:*

много интерфейсов, специально предназначенных для клиентов, лучше, чем один интерфейс общего назначения. Нельзя заставлять клиента реализовать интерфейс, которым он не пользуется

**Dependency inversion principle** –

- Принцип инверсии зависимостей:

Зависимость на Абстракциях. Нет зависимости на что-то конкретное

* Модули верхних уровней не должны зависеть от модулей нижних уровней. Оба типа модулей должны зависеть от абстракций.
* Абстракции не должны зависеть от деталей. Детали должны зависеть от абстракций.

Паттерны проектирования

**Шаблон**или **паттерн** **проектирования** (design **pattern**) — повторяемая архитектурная конструкция, представляющая собой решение проблемы проектирования в рамках некоторого часто возникающего контекста.