Object Oriented Design Notes

Feature - 12345

**Revisions**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **User** | **Modifications** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# OOD General

OOD design notes based on collections of researches and work experiences, the main reference includes:

* <https://www.educative.io/courses/grokking-the-object-oriented-design-interview>
* Java books including in Effective Java and Head First design pattern book.

## Open Questions

* Aggregation and Composition?
* Requirement clarification:

## Resources

## S.O.L.I.D.: The 5 Principles of OOD

**Reference**:

* <https://dzone.com/articles/a-gentle-and-easy-to-grasp-introduction-to-solid-p>
* <https://javarevisited.blogspot.com/2018/07/10-object-oriented-design-principles.html>

S.O.L.I.D is an acronym for the first five object-oriented design(OOD)\*\* principles\*\* by Robert C. Martin, popularly known as Uncle Bob.

These principles, when combined together, make it easy for a programmer to develop software that are easy to **maintain** and **extend**. They also make it easy for developers to avoid code smells, easily **refactor code**, and are also a part of the agile or adaptive software development.

**1. Single-responsibility Principle:**

* A class should have one and only one reason to change, meaning that a class should have only one job.
* This applies separation of concerns, which means that you should try and separate the concerns into different classes.
* A class should be focusing on a single problem, piece of logic, or a single domain. When the domain, specification, or logic changes, it should only affect one class.

**2. Open-closed Principle (OCP):**

* Objects or entities should be open for extension, but closed for modification.
* The term “Open for extension” means that we can expand and include extra cases/functionalities in our code without altering or affecting our existing implementation.
* The term “Closed for modification” means that after we add the extra functionality, we should not modify the existing implementation.
* The key benefit of this design principle is that already tried and tested code is not touched which means they won’t break.

**3. Liskov substitution principle**

* Let q(x) be a property provable about objects of x of type T. Then q(y) should be provable for objects y of type S where S is a subtype of T.
* All this is stating is that every subclass/derived class should be substitutable for their base/parent class.
* LSP states that the software should not alter the desirable results when we replace a parent type with any of the subtypes.
* LSP is more of a problem definition than being a design pattern and what we can do to prevent undesirable effects are to apply the Open-Closed Principle and a design through the Contract pattern.

**4. Interface segregation principle**

* A client should never be forced to implement an interface that it doesn't use or clients shouldn't be forced to depend on methods they do not use.
* ISP states that we should split our interfaces into smaller and more specific ones.

**5. Dependency Inversion principle**

* The DIP states that we should depend on abstractions (interfaces and abstract classes) instead of concrete implementations (classes).
* This principle emphasis for decoupling, code to interface.
* Also called Dependency Injection, this has been very well implemented in Spring framework.

**6. Others:**

* Favor Composition over Inheritance
* Programming for Interface not implementation

## OOD Design Patterns (now)

Resources:

<https://www.oodesign.com/>

--> use 80-20 rule, 20% of hot URLs generate 80% traffic, **per day**

## System Interface

## System Interface

# Designing NEXT

## Array

123

## Requirement and Goal of the System

Functional and none functional requirement

## Design Consideration

## Capacity Estimation and Constrains

## High level System Design

## Database Schema

## Data Size Estimation

## Component Design

## Reliability and Redundancy

## Data Sharding

## Ranking and News Feed Generation

## News Feed with Sharded Data

## Cache and Load Balancing