Design Principles

Arnav Gupta

December 5, 2024

Contents

1 Design Principles

UML: general purpose modeling language for object-oriented software systems

UML used informally in practice.

Design principles ease communication, leverage existing design knowledge, enhance flexibility for future change, and increase reusability of developed code.

Gang of Four Design Patterns:

- Class, Creational → Factory Method
- Class, Structural → Adapter (class)
- Class, Behavioural → Interpreter, Template Method

- Object, Creational → Abstract Factory, Builder, Prototype, Singleton
- Object, Structural → Adapter (object), Bridge, Composite, Decorator, Facade, Flyweight, Proxy
- Object, Behavioural → Chain of Responsibility, Command, Iterator, Mediator, Memento, Observer, State, Strategy, Visitor

2 Single Responsibility Principle and Class Cohesion

Single Responsibility Principle: a class should have one and only one reason to change

Gives high cohesion between class members and better segregation of concerns.

Class cohesion: public methods of a class must be closely aligned to each other in behaviour

- methods should access common fields
- methods should call each other for desired functionality

To follow SRP, a class cannot expose unnecessary public methods to clients (pertains to segregation of concerns).

If different clients use different disjoint subsets of methods, separate these into different classes.

3 Open Closed Principle

Software entities (class, methods) should be open for extension (inheritance) but closed for modification (once tested and in production).

Gives low coupling between program, clients, and other dependent programs and promotes flexible, extensible, and reusable development (better speed, lower cost).

Possible through abstractions.

4 Liskov Substitution Principle

References to the base class must be completely substitutable by references of any derived classes.

Rules:

1. Invariants of supertype must not be violated or broken by subtype.
2. Preconditions on overridden methods of subtype must be subset of preconditions on the same method for the supertype.
3. Postconditions on overridden methods of subtype must be superset of postconditions on the same method for the supertype.

Violated if:

- a overridden method does nothing or just throws an exception
- subtype method modifies a field of the superclass, which is not exposed through a setter

5 Interface Segregation Principle

Client must not be forced to use interfaces that it does not need.

Gives readable, refactorable, and manageable code and promotes high cohesion for classes implementing the interface.

Advocates splitting large interfaces into smaller interfaces (role interfaces vs fat/polluted interface).

Violated if:

- a class implements an interface but some interface methods are not required and are implemented with empty body or return null/throw exception

Each method implemented must define a concrete behaviour for the class.

6 Dependency Inversion Principle

High level modules should not depend on low level modules. Instead, both should depend upon abstractions. Abstractions should not depend on details, instead details should depend upon abstractions.

Violated if:

- layered architecture: one layer sensitive to changes in another layer
  - instead use an interface that both layers use