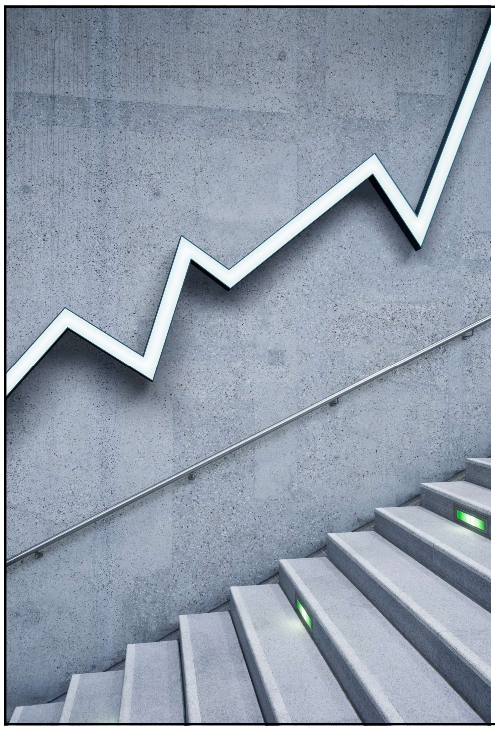

SOLID Principles

AMEYA JOSHI

SOLID Principles

- ➢**SOLID principles** are an object-oriented approach that are applied to software structure design.
- ➢Conceptualized by **Robert C. Martin.**
- ➢There are 5 SOLID Principles.
  - ✓Single Responsibility Principle (SRP)
  - ✓Open-Closed Principle (OCP)
  - ✓Liskov Substitution Principle (LSP)
  - ✓Interface Segregation Principle (ISP)
  - ✓Dependency Inversion Principle (DIP)

Single Responsibility Principle

- The single responsibility principle states that **every Java class must perform a single functionality**.
- Implementation of multiple functionalities in a single class mashup the code and if any modification is required may affect the whole class.

Open-Closed Principle

- The open-closed principle states that according to new requirements **the module should be open for extension but closed for modification.**
- The extension allows us to implement new functionality to the module.

Liskov Substitution Principle

- ➢The Liskov Substitution Principle (LSP) was introduced by **Barbara Liskov**.
- ➢It applies to inheritance in such a way that the **derived classes must be completely substitutable for their base classes**.
- ➢It extends the open-close principle and also focuses on the behavior of a superclass and its subtypes.

Interface Segregation Principle

- ➢The principle states that the larger interfaces split into smaller ones.
- ➢The implementation classes may use only the methods that are required. We should not force the client to use the methods that they do not want to use.
- ➢The goal of the interface segregation principle is similar to the single responsibility principle.

Dependency Inversion Principle

- The principle states that we must use abstraction (abstract classes and interfaces) instead of concrete implementations.
- High-level modules should not depend on the low-level module but both should depend on the abstraction.
- The abstraction does not depend on detail but the detail depends on abstraction, resulting in decoupling the software.

DRY – Don't Repeat Yourself

- A basic principle of software development aimed at reducing repetition of information.
- Every piece of knowledge or logic must have a single, unambiguous representation within a system.
- Less code is good: It saves time and effort, is easy to maintain, and also reduces the chances of bugs.

KISS - Keep It Simple, Stupid

Keep the code simple and clear, making it easy to understand.

Keep your methods small.

Each method should only solve one small problem