Coding Principles

are guidelines for developers to craft high-quality, efficient, and maintainable code.

1 SOLID

is a set of five object-oriented design principles—**S**ingle Responsibility, **O**pen/Closed, **L**iskov Substitution, **I**nterface Segregation, and **D**ependency Inversion—aimed at promoting modularity, flexibility, and maintainability in software development, providing a foundation for robust and scalable code architectures.

2 KISS

Keep It Simple, Stupid. Encourages simplicity in design and implementation, advocating for straightforward, uncomplicated solutions. Prioritizing simplicity can enhance readability and reduce the likelihood of errors.

3 YAGNI

You Ain’t Gonna Need It. Advises developers to avoid adding functionality until it is actually necessary. This principle helps prevent over-engineering and promotes a focus on current requirements rather than speculative future needs.

Nabi Karampoor
@thisisnabi

4 DRY

Don’t Repeat Yourself. Emphasizes code reusability by avoiding duplication, reducing redundancy, and promoting the use of functions, modules, or classes to encapsulate and share common functionality.

5 Modularity

This principle emphasizes the importance of breaking down a software system into smaller, self-contained modules or components.

6 Law of Demeter

Advocates for loose coupling between objects by limiting interactions to only immediate neighbors. This reduces the potential impact of changes in one part of the system on other parts, enhancing maintainability and flexibility.

7 Fail-Fast

Promotes the idea that a system should detect and report errors as soon as they occur, allowing for faster diagnosis and resolution. This minimizes the impact of errors and aids in creating more robust software.

8 Separation of Concerns

Encourages dividing a program into distinct sections, or concerns, each responsible for a specific aspect of functionality. This principle facilitates code organization, maintenance, and comprehension.

Nabi Karampoor

@thisisnabi