ONE PAGE SUMMARY

IT & Software › Design Patterns › SOLID Principles

Introduction to SOLID Principles

Master the SOLID Principles for Better Software Design!

DEFINITION

> The SOLID principles are five design guidelines that help developers create maintainable and scalable software. They promote good object-oriented design practices.

KEY CONCEPTS

Single Responsibility Principle

A class should have only one reason to change, meaning it should only have one job.

Open/Closed Principle

Software entities should be open for extension but closed for modification, allowing new functionality without altering existing code.

Liskov Substitution Principle

Subtypes must be substitutable for their base types without altering the correctness of the program.

Interface Segregation Principle

Clients should not be forced to depend on interfaces they do not use, promoting smaller, more specific interfaces.

Dependency Inversion Principle

High-level modules should not depend on low-level modules; both should depend on abstractions.

EXAMPLES

→ A class managing user data should not also handle file storage.

→ Using interfaces to define behaviors for different payment methods.

→ Creating a new feature by adding a new class instead of modifying existing ones.

MEMORY TIPS

★ Remember 'SOLID' as a solid foundation for good design.

★ Think of 'Single Responsibility' as 'One Job, One Class'.

★ Use 'OCP' to recall 'Open for extension, Closed for modification'.

COMMON MISTAKES

✗ Confusing interface segregation with class inheritance.

✗ Neglecting to apply the Liskov principle when creating subclasses.

✗ Overcomplicating designs instead of keeping them simple and focused.

QUICK RECAP

The SOLID principles guide developers in creating robust, maintainable software. Focus on single responsibilities, extensibility, and proper use of interfaces to enhance your design skills.