Use Solid principles

What is Solid principles?

The SOLID principles are a set of five design principles that provide guidelines for writing maintainable, flexible, and understandable software. These principles can be applied throughout the software development lifecycle, from the initial design and architecture phase to the ongoing maintenance and evolution of the software.

when and how to use each SOLID principle:

- **Single Responsibility Principle (SRP):**

Use when designing classes or modules to ensure that they have a clear and well-defined responsibility.

Aim to have each class or module focused on a single task or responsibility.

- **Open/Closed Principle (OCP):**

Apply during the design phase to create extensible and adaptable software components.

Design classes and modules in a way that new functionality can be added through inheritance, interfaces, or extension without modifying existing code.

- **Liskov Substitution Principle (LSP):**

Use when designing class hierarchies or inheritance relationships.

Ensure that derived classes can be substituted for their base classes without affecting the correctness of the program.

- **Interface Segregation Principle (ISP):**

Apply when designing interfaces to ensure that they are focused and tailored to specific client needs.

- **Dependency Inversion Principle (DIP):**

Apply during architectural design and component interaction.

Use abstractions (interfaces or abstract classes) to define high-level modules and their dependencies on lower-level modules.