Solid Principles

SOLID stands for:

- **S** - Single-responsiblity Principle
- **O** - Open-closed Principle
- **L** - Liskov Substitution Principle
- **I** - Interface Segregation Principle
- **D** - Dependency Inversion Principle

Single-Responsibility Principle

A class should have one and only one reason to change, meaning that a class should have only one job.

Why? SRP helps in increasing the reusability of code.

Open-Closed Principle

Objects or entities should be open for extension but closed for modification.

This means that a class should be extendable without modifying the class itself.

Liskov Substitution Principle

Objects should be replaceable by their subtypes without altering how the program works. In other words, derived classes must be substitutable for their base classes without causing errors.

Interface Segregation Principle

This states that many client-specific interfaces are better than one general-purpose interface. In other words, classes should not be forced to implement interfaces they do not use.

Dependency Inversion Principle

Classes should depend upon abstractions, not concretions. Essentially, don't depend on concrete classes, depend upon interfaces.