**When to use Solid principles**

The SOLID principles are a set of five design principles in object-oriented programming that aim to make software systems more maintainable, flexible, and easy to understand. Here's a guideline on when to apply the SOLID principles:

1. Single Responsibility Principle (SRP): A class should have only one reason to change. Apply this principle when a class has multiple responsibilities or when changes in one area may cause unrelated changes in other areas. Splitting the class into smaller, more focused classes improves maintainability and reduces the impact of changes.
2. Open/Closed Principle (OCP): Software entities (classes, modules, functions) should be open for extension but closed for modification. Use this principle when you want to add new functionality without modifying existing code. By relying on abstraction, interfaces, and inheritance, you can introduce new behavior through extension rather than modification.
3. Liskov Substitution Principle (LSP): Subtypes must be substitutable for their base types without affecting the correctness of the program. Apply this principle when working with inheritance hierarchies. Subclasses should adhere to the contract defined by their base classes and not introduce new behaviors or violate existing ones.
4. Interface Segregation Principle (ISP): Clients should not be forced to depend on interfaces they do not use. Use this principle when designing interfaces. Instead of having large and monolithic interfaces, split them into smaller and more focused ones, tailored to the specific needs of clients. This promotes loose coupling and avoids unnecessary dependencies.
5. Dependency Inversion Principle (DIP): High-level modules should not depend on low-level modules; both should depend on abstractions. Apply this principle to achieve loose coupling and promote flexibility. Depend on abstractions (interfaces or abstract classes) rather than concrete implementations. This allows for easier swapping of implementations and facilitates testing and maintainability.