When to use the SOLID principles:

1. **Single Responsibility Principle (SRP):**

   Imagine each class like a person with a job. Just like a person should have one main job, a class should have one main purpose. This makes the class easier to understand and change when needed.

2. **Open/Closed Principle (OCP):**
   Think of your code as a LEGO set. When you want to add something new, you shouldn't have to break apart what you already built. Your code should be open to adding new features without changing what's already working.

3. **Liskov Substitution Principle (LSP):**
   This is like having different types of cars that you can use interchangeably. If you have a base type (like a car) and a specific type (like a sports car), the specific type should be usable wherever you'd use the base type, without causing trouble.

4. **Interface Segregation Principle (ISP):**
   Imagine interfaces like menus in a restaurant. Each menu should only have a few choices related to a

certain type of food. Your classes should have small, specific interfaces that only include what they really need.

5. **Dependency Inversion Principle (DIP):**
   Think of it as teamwork. High-level stuff (like managers) shouldn't depend directly on low-level stuff (like workers). Instead, they both should talk to each other through clear plans (interfaces). This way, if something changes in the low-level stuff, it doesn't mess up the high-level stuff.

Use these principles when you want your code to be clear, easy to change, and ready for new things. It's like following good rules to build strong and flexible structures. Just remember, while these rules are helpful, sometimes bending them a bit can make sense in special cases.