SOLID Object Oriented Design

S: SRP, the Single Responsibility Principle

A class should have only one purpose (which is not “do everything”). In practice, this means careful design and separation of concerns, and refactoring when a class has taken on multiple roles.

O: OCP, the Open Closed Principle

A class should be open for extension but closed for modification. In practice this involves careful design of the public API and careful choice of private/protected tradeoffs. Avoiding implementation inheritance and sticking with interface inheritance will help, too!

L: LSP, the Liskov Substitution Principle

An object of a subclass must be able to be used anywhere an object of its parent class can be used. This is surprisingly easy to violate, see for example the circle/ellipse dilemma. In practice, ensure that your subclass does not violate any capability of the parent class.

I: ISP, the Interface Segregation Principle

Many small, specific interfaces are better than one general interface. In practice, this assumes you will create interfaces (and you should!), and that you will create many small specific interfaces, not a few big ones.

D: DIP, the Dependency Inversion Principle

Abstractions should not depend on details, details should depend on abstractions. In practice, this means: create an interface at the same level of abstraction as the client code that will use it, and then implement the interface in an adapter that wraps the low level detail and hides it.