OO Design Principles

Encapsulate what varies.

If you've got some aspect of your code that is changing, say with every new requirement, then you know you've got a behavior that needs to be pulled out and separated from all the stuff that doesn't change.

Favor composition over inheritance.

Creating systems using composition gives you a lot more flexibility. Not only does it let you encapsulate a family of algorithms into their own set of classes, but it also lets you change behavior at runtime as long as the object you're composing with implements the correct behavior interface.

Classes should be open for extension but closed for modification.

Our goal is to allow classes to be easily extended to incorporate new behavior without modifying existing code.

What do we get if we accomplish this? Designs that are resilient to change and fl exible enough to take on new functionality to meet changing requirements.

Depend on Abstraction. Do not depend on concrete classes.

Our high-level components should not depend on our low-level components; rather, they should both depend on abstractions

Only talk to your friends.

When you are designing a system, for any object, be careful of the number of classes it interacts with and also how it comes to interact with those classes.

A class should have only one reason to change.

Every responsibility of a class is an area of potential change. More than one responsibility means more than one area of change.