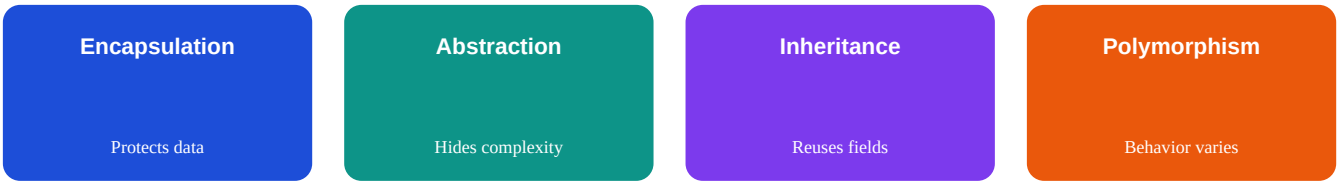
OOP Code Review Checklist

Spot encapsulation gaps, leaky abstractions, and anemic domain models during code review.

1. The Four Pillars



2. Encapsulation Checklist

- ■ Does every setter validate its input, or does it accept anything silently?
- ■ Is there more than one code path that can write to this object's fields (UI, batch import, API)?
- ■ If validation exists, is it enforced INSIDE the class, or scattered in a controller somewhere else?

3. Abstraction Checklist

- ■ Does error handling leak provider-specific codes or types through the interface?
- ■ Would adding a second implementation break any calling code today?
- ■ Can a caller tell which concrete implementation it's using from the code it has to write?

4. Anemic vs. Rich Domain Model

Model	Where Logic Lives	Risk
Anemic	Separate service classes	Any new path can bypass validation
Rich	Inside the domain objects	More upfront design discipline needed

Rule of thumb: a private field with an unrestricted setter isn't encapsulation — it's a public field with extra syntax. Real encapsulation means the object itself enforces its own valid states.