S.O.L.I.D

A Brief Introduction

What does S.O.L.I.D. stand for?

- Single Responsibility
- Open/closed Principle
- Liskov Substitution Principle (LSP)
- Interface Segregation
- Dependency Inversion

Single Responsibility Principle

“a class should have only a single responsibility (i.e. changes to only one part of the software's specification should be able to affect the specification of the class).”

Single Responsibility Principle

- Cleaner and simpler classes
- Improves readability
- Improves testability
- Allows for easier extension
- Modular code is easier to reuse

Fabric Example: Multi-responsibility vs Single responsibility

Open/closed Principle

“software entities … should be open for extension, but closed for modification.”

Open/closed Principle

- De-coupled classes
- Not tied to a specific use case
- Reduced testing footprint
- Easier to streamline code

Code Example: Version 1, Version 2, Version 3

Liskov Substitution Principle (LSP)

“objects in a program should be replaceable with instances of their subtypes without altering the correctness of that program.”

Liskov Substitution Principle (LSP)

- Is Object B a subtype of Object A *or* is a Square a type of Rectangle
- Inherited methods allow for code reuse…
- … but not everything is related

“if it looks like a duck and quacks like a duck but uses a battery, you’ve got your abstractions wrong”

Code Example: Classic LSP Violation

Interface Segregation

“many client-specific interfaces are better than one general-purpose interface.”

Interface Segregation

- Large interfaces should be split up into multiple smaller interfaces
- How many jobs does your interface have
- If your implementation doesn’t need all the methods on the interface, your interface is probably too generic

Code Example: Version 1, Version 2

Dependency Inversion

“depend upon abstractions, [not] concretions.”

Dependency Inversion

- Not the same as Dependency Injection!
- Reduces scope of changes
- Allows for mocking
- Allows us to swap between or mix implementers with much more controlled effects

Code Example: Classic DI Example