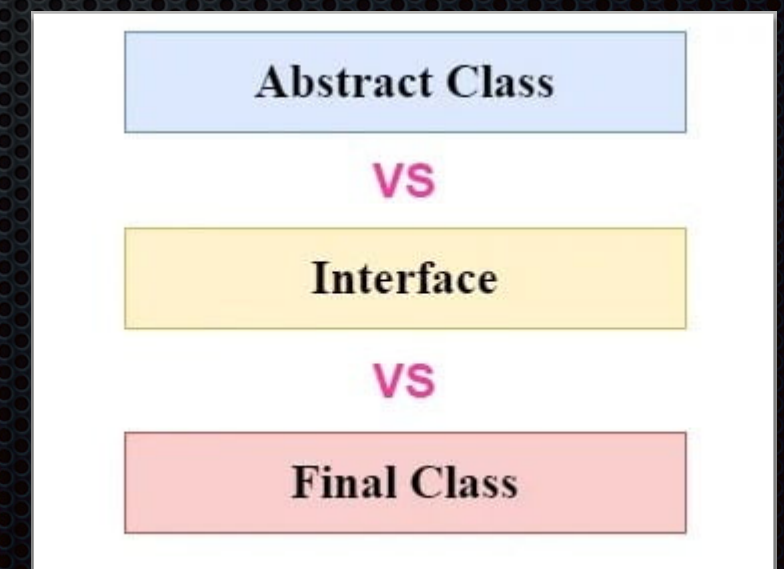
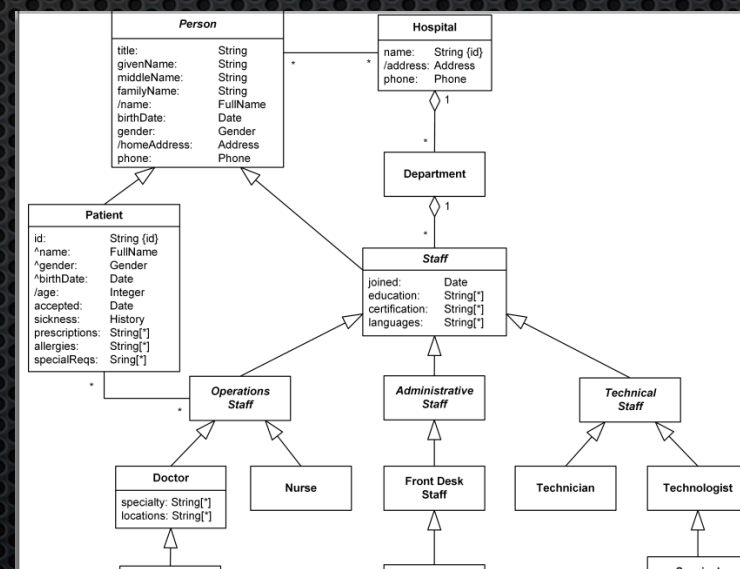
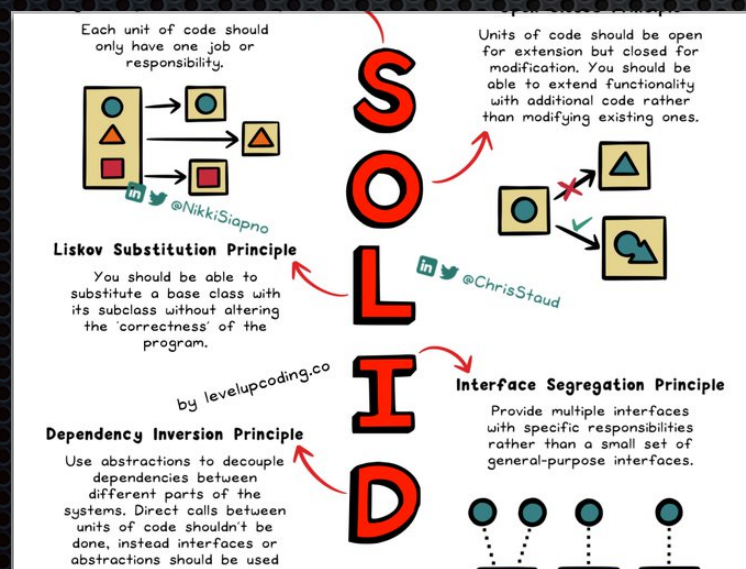


SOLID Design Principles

David Hackbarth



General

- part of clean code
- principles for better
 - readability
 - changeability
 - extensibility
 - maintainability
- are abstract not concrete

S.O.L.I.D.

- ✦ Single-responsibility principle (SRP)
- ✦ Open-closed principle (OCP)
- ✦ Liskov substitution principle (LSP)
- ✦ Interface segregation principle (ISP)
- ✦ Dependency inversion principle (DIP)

Single Responsibility Principle

“a class should have one, and only one, reason to change.”

- ✦ only one function
- ✦ solving one problem
- ✦ can be applied to classes, modules or services
- ✦ easy to implement

Single Responsibility Principle

- ✦ maintainability
- ✦ testability
- ✦ flexibility
- ✦ complex GameController vs. Single Function Classes

Open-Closed Principle

“You should be able to extend a class’s behavior without modifying it.”

- ✦ open for extension
- ✦ closed for modification
- ✦ abstraction
- ✦ inheritance or interfaces

Open-Closed Principle

- ✦ extensibility
- ✦ stability
- ✦ flexibility

Liskov substitution principle

“Functions that use pointers or references to base classes must be able to use objects of derived classes without knowing it.”

- ✦ originally by Barbara Liskov in 1987
- ✦ extension to open-closed principle
- ✦ derived classes should not change behavior

Liskov substitution principle

- ✦ polymorphism
- ✦ reliability
- ✦ predictability
- ✦ e.g. bird class vs. penguin

Interface Segregation Principle

“Make fine grained interfaces that are client-specific. Clients should not be forced to implement interfaces they do not use.”

- ✦ smaller interfaces
- ✦ no big ones
- ✦ don't extend interfaces
- ✦ build new ones

Interface Segregation Principle

- ✦ decoupling
- ✦ flexibility
- ✦ avoid unnecessary dependencies

Dependency Inversion Principle

“high level modules should not depend upon low level modules. Both should depend on abstractions.”

- ✦ don't use implementation
- ✦ use abstraction instead

Dependency Inversion Principle

- ✦ loose coupling
- ✦ flexibility
- ✦ maintainability
- ✦ e.g. render system

