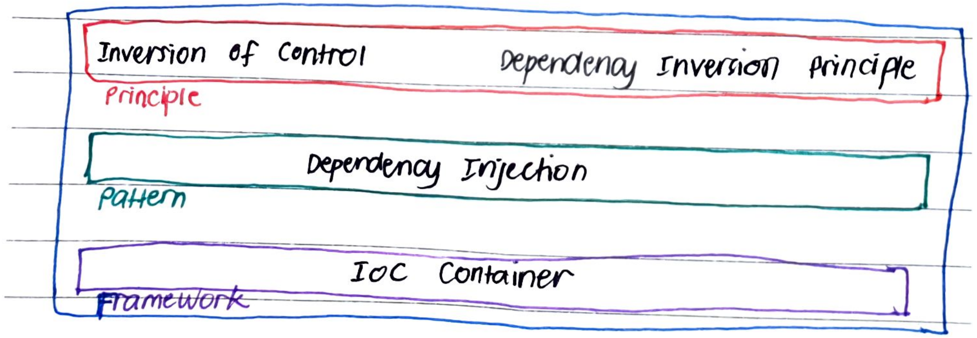
CLEAN CODE : SYSTEMS

! separate constructing a system from using it

- * software systems should separate the startup process from the runtime logic.
- * one way to separate construction from use is to move all aspects of construction to main or modules called by ~~moo~~ main.
- * Application has no knowledge of main or construction process.

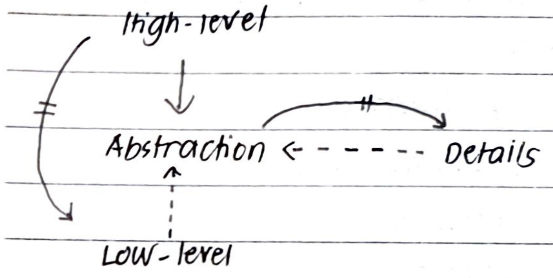
Inversion of Control (IoC)

- invert the control , separate controlling stuff to another class ↳ additional responsibilities
- invert the dependency creation and management of object lifetime
- Factory , Dependency Injection

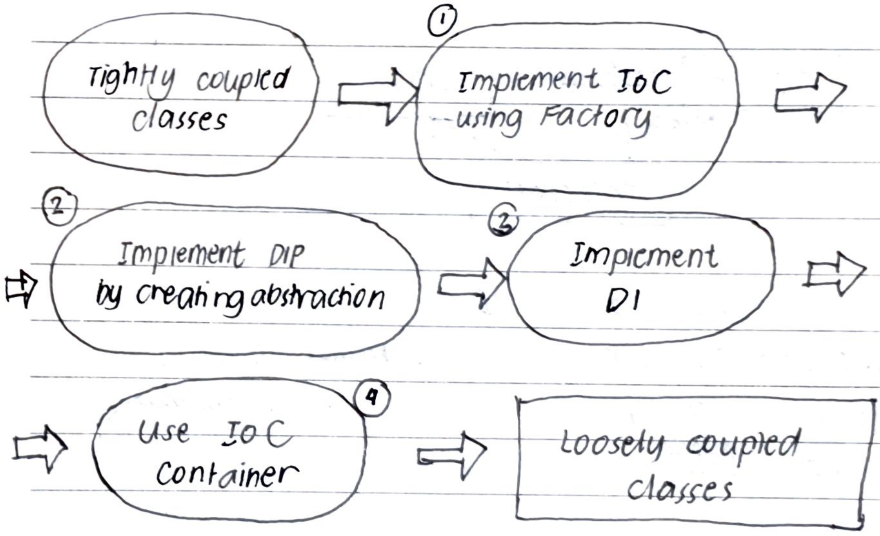


Dependency Inversion Principle (DIP)

- High-level modules should not depend on low-level modules , both should depend on abstraction



- Abstractions should not depend on details. details should depend on abstractions

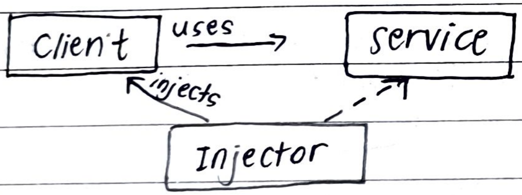


Dependency Injection (DI)

- a design pattern used to implement IOC where it allows creation of dependent objects outside of a class and provide those objects to a class through different ways.

3 Types of Classes

1.) Client class - dependent class
2.) Service class - dependency
3.) Injector class - injector



Types of DI

1. Constructor Injection
2. Property Injection
   ↳ set a public property
3. Method Injection
   ↳ set dependency

IOC container

- framework for implementing automatic dependency injection
- manages object creation and lifetime and injects dependencies to classes

* FOUR SIMPLE RULES OF DESIGN *

P asses the tests
R eveals intent
D RY implementation
F ewest elements

srp - single responsibility principle

→ A class should only have one reason to change

Responsibilities to consider that may need to be separated :

- Persistence
- Validation
- Notification
- Formatting
- Mapping
- Error-handling
- Logging
- Class selection / Instantiation
- Parsing

/* Gather the things together that change for the same reason. separate those things that change for different reasons. */

How would I like to use this class? VS What does this class need to do?