**Principles:**

Single responsibility

Open/closed

Liskov substitution principle

Interface segregation

Dependency inversion.

Tell don't ask - Tell an object what you want to know rather than asking for properties and determining that yourself

Hollywood principle – Don’t call us, we’ll call you

Reused abstraction principle

Fail Fast – Stop processing on failure.

Return early – Return as soon as you can.

Dry (once and only once) – Don’t repeat yourself

Loose coupling - Code to an interface, not an implementation

Abstract what changes

Encapsulation - information hiding

Separation of concerns – UI, BL, DAL, SOA

Programming style as documentation – Comments and outside docs are a last resort

Turn implicit concepts into explicit (DDD)

Guard clause – Check up front before starting business logic

Law of Simplicity - The ease of maintenance of any piece of software is proportional to the simplicity of its individual pieces.

Mayfly variables - Minimize variable lifetime

**Design Patterns:**

Domain event

Composite

Facade

Flyweight

Ravioli Code

Strategy - Interchangeable algorithms

Decorator

Singleton – Assure only a single instance is created

Null Object pattern -

Factory – Centralized point for object instantiation

Repository -

State (closely related to strategy) - objects for states

Virtual proxy

Parameter objects (fowler)

Object pooling

Circuit breaker

Dependency injection

Humble object – Lean object that delegates to another class

N-teir

Active record – Object per DB row

Query by example – Instantiate an object with desired properties for filtering

Data Transfer Object (DTO) – Dumb data structure for passing around – no behavior.

Arrange act assert (unit testing)

Design by contract