**Design Patterns**

Gang of four

J2ee patterns

Design Pattern – Solution to a recurring problem.

OOD concepts

1. Cohesion – degree to which methods and attributes of class focus on only one purpose in the system.
2. Encapsulation – Information hiding. Don’t expose data members directly. Expose them through public methods.
3. Coupling – measure of how one component depends on other component.
   1. A change in one component needs major change in other component – tight coupling.
   2. A change in one component need not require change in other component – loose coupling.
4. Polymorphism – invocation of operation of specific objects through generic references

Associations

1. Aggregation – weaker has a (car can exist without carradio)
2. Composition – stronger has a (car cannot exist without engine)

Design Principles

1. Favoring composition (encourage has-a over is-a : advantage is reducing coupling)
2. Programming to an interface(write code to top level interface)
3. Designing for change

SOLID Principles

1. Single responsibility principle(A Class should have only one reason to change)
2. Open close principle(open for extension closed for logic modification)
3. Liskov substitution principle (Derived types should be always substitutable to their base types) For subclasses to be substitutable, sub classes should satisfy all properties of super class.
4. Interface segregation principle(Client Classes should not be forced to implement interfaces they don’t use)
5. Dependency inversion principle(abstractions need not know about implementations but implementations should know about abstractions)

GOF Patterns:

1. Behavioral Patterns: How Objects distribute functionality. (focus more on behavior – overriding the methods)
   1. Strategy Pattern: multiple algorithms
   2. Command Pattern : Encapsulate a request as an object.
   3. Template method: Defines the skeleton of algorithm in an operation, deferring some steps to subclasses.
2. Creational Patterns: Robust mechanism to create objects. By inserting an interface between a client class and its target constructor, a creational pattern separates the interface for making an object from the details of construction.
   1. Factory pattern: define an interface for creating an object, but let subclasses decide which class to instantiate.
   2. Flyweight : object with same content is available, return the same object
   3. Abstract Factory: Provide an interface for creating families or related dependent objects without specifying their concrete classes.

Factory produce at most one product

Abstract factory may product more than one product.

1. Structural Patterns: how (unrelated) objects are linked together.