1. Design Pattern

* Are the best practices
* Are the solutions to general problems that software developers face during development

1. Gang of Four (GOF)?

* In 1994, four authors ***Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides*** published a book titled Design patterns – Elements of Reusable Object-Oriented Software
* These authors are known as Gang Of Four (GOF)
* Design patterns are primarily based on following principles
  1. Program to an interface not an implementation
  2. Favor object composition over inheritance

1. Classification of Design Patterns

* .
  1. Creational pattern
     + Create objects hiding the creation logic rather than instantiating objects directly using new operator
  2. Structural Pattern
     + These patterns concern class and object composition.
     + Inheritance is used to compose interfaces and define ways to compose objects to obtain new functionalities
  3. Behavioral Patterns
     + Specifically concerned with communication between objects

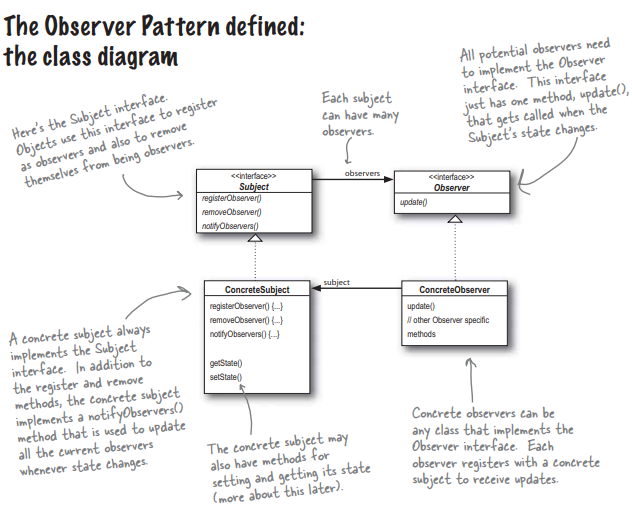
1. Design Principles

* .
  1. Indentify the aspects of your application that vary and separate them from what stays the same
     + Take what varies and “encapsulate” it so it won’t affect the rest of the code
  2. Program to an interface, not an implementation
     + You can actually program to an interface without actually using an interface
  3. Favor Composition over Inheritance
     + It lets you encapsulate a family of algorithms
     + Change behavior at runtime
  4. Strive for loosely coupled designs between objects that interact

1. Strategy Pattern

* It comes under behavior pattern
* In this pattern, a class behavior or its algorithm can be changed at runtime
* … defines a family of algorithms, encapsulates each one and make them interchangeable
* In this pattern, instead of adding functions to the interfaces, we create a new interface for behaviors that may vary and create other implementations of that interface so that main context can choose the strategies according to the varying behavior
* This pattern is useful when we have multiple algorithms for specific task and we want our application to be flexible to chose any of the algorithm at runtime for specific task.
* Realtime Example: Payment method in E-commerce may vary
* Examples: Collections.sort() using Comparable, Comparators, Arrays.sort()

1. Observer Pattern

* Publishers + Subscribers
* Newspaper subscriptions is most common example of this pattern
* We call the publisher as the SUBJECT and the subscribers as OBSERVERS
* The observer pattern defines a one-to-many dependency between objects so that when one object changes state, all of its dependents are notified and updated automatically
* 
* Subjects and observers are loosely coupled. They can interact but have very little knowledge of each other
  1. The only things the subject knows about an observer is that it implements a certain interface (Observer)
  2. We can add new observers at any time
  3. We never need to modify the subject to add new types of observers
  4. We can reuse subjects or observers independently of each other
  5. Changes to either the subject or an observer will not affect the other