1. OOP
   1. Abstraction: models real world objects
   2. Cohesion: elements in a model relate
   3. Coupling: elements use necessary connections and nothing else
   4. Encapsulation: Data is hidden from all except those who need it
2. Vocabulary
   1. Open/close principle: software entities should be open for extension and closed for modification
   2. Single responsibility principle: Every context should have one responsibility only
3. Singleton Pattern
   1. Ensures a class has only one instance
   2. Provides a single, global point of access
   3. Variants:
      1. Private Constructor
      2. Abstract class
      3. getInstance() method
4. Strategy Pattern
   1. Defines a family of algorithms, encapsulates each, and makes them interchangeable.
5. Composite Pattern
   1. Composes objects into tree structures to represent part-whole hierarchies. Let’s clients treat individual objects and compositions of objects uniformly
6. Chain of Responsibility Pattern
   1. Avoids coupling sender of request to receiver by giving more than one object a chance to handle the request. Passes request along the chain until the most appropriate object can handle it.
7. Visitor Pattern
   1. Represents operation to be performed on elements of object structure. Lets you define new operation without changing classes of elements on which it operates. Lets you modify functionality while visiting.
8. Mediator Pattern
   1. Defines object that encapsulates how set of objects interact. Promotes loose coupling by keeping objects from referring to each other explicitly, and lets their interaction vary independently
   2. Mediator object allows you to simplify objects by delegating functionality to mediator
9. Decorator Pattern
   1. Attaches additional responsibility to object dynamically. Provides flexible alternative to subclassing for additional functionality.