* SOLID
  + **SRP** (Single responsibility Principle)
    - A class should have single reason to change.
    - A class should have single responsibility.
    - If an object takes lots of responsibilities, it’s called as a **god object** which is an anti pattern
    - String.join (System.lineSeparator(), list)
  + **OCP** (Open Close Principle)
    - Open for extension, close for principle
    - Specification Pattern
  + **Liskov** substitute Principle:
    - You should be able to substitute a subclass for a base class (object slicing)
  + **ISP** (Interface segregation principle)
    - Segregate the concerns and put them in difference interfaces rather than putting in one; then extend multiple interfaces to change the contract of reliazation
    - Not Implemented Exception
    - YAGNI
      * You aint going to need it
  + **Dependency Inversion** Principle
    - High level modules should not depend on low level modules
    - Abstractions should not depend on details.
    - Don’t misunderstand dependency inversion with dependency injection.
* Decorator:
  + Why/When to use?
    - Adding behavior without altering the class itself or directly inheriting from it.
    - Want to augment the object with additional functionalities later some point in future.
    - To maintain SRP and OCP and interact with existing structure
    - How to do now? We have two options to achieve it:
      * Inherit the required class which you want to change and supply the changed class to the place where you want to use it. **BUT** sometimes classes are marked as final.
      * Build a Decorator, which simply references the decorated objects