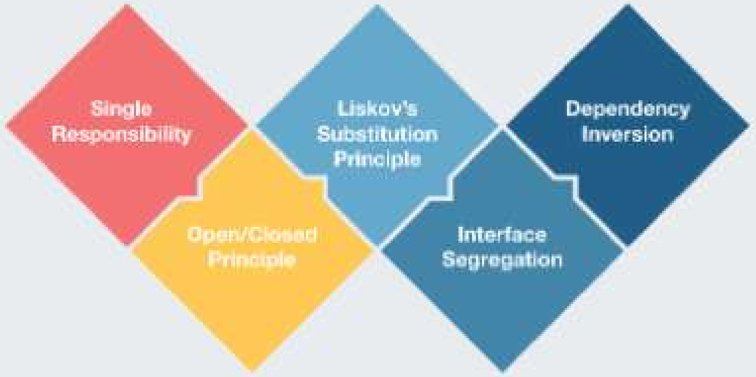
Liskov Substitution Principle

S.O.L.I.D.



Part 4

Visit GitHub @BCAPATHSHALA

1. LISKOV Substitution Principle (LSP)

1. Any `**derived**` class should be able to substitute its `**parent**` class without the consumer knowing it.
2. Every part of the code should get the expected result no matter what instance of a class you send to it, given it implements the same interface.
3. If a function takes a `Base` class as parameter then, this code should work for all the `Derived` classes.
*4. **LSP** insures that the good application i.e., built using **abstraction** does not break.*
*5. It states that the objects of a `subclass` should behave the same way as the objects of the `superclass`, such that they are **replaceable**.*
*6. **Key:** `**Child**` class should be able to do what a `**parent**` class can.*
*7. **Goal**: The goal of **LSP** is to ensure that a `**subclass**` can stand in for its `**superclass**`. This principle helps in maintaining the correctness of the program when objects of a superclass are replaced with objects of a subclass.*

In One Statement
The ***LISKOV Substitution Principle*** *states that objects of a superclass should be replaceable with objects of a subclass without affecting the correctness of the program.*

Key Idea
You should be able to use any `subclass` where you use its `parent` class.

Real-Time Examples
You have a remote control that works for all types of `TVs`, regardless of the `brand`. Where,
- ***Brand:*** *as a Parent class*
- ***Remote Control:*** *as a feature of Brand*
- ***TVs:*** *as a Child class*

How can LISKOV Substitution Principle be applied?