SOLID DESIGN PRINCIPLES

AGENDA

- Single Responsibility Principle
- Open Closed Principle
- Liskov Substitution Principle
- Interface Segregation Principle
- Dependency Inversion Principle

SINGLE RESPONSIBILITY PRINCIPLE

There should never be more than one reason for a class to change.

E.g.: JSON, XML(Message Format) might change.
(Consider it as input), Authentication

OPEN CLOSED PRINCIPLE

Open-Closed Principle: Software Entities(Classes, Modules, Methods, etc.) should be open for extension but closed for modification.

Open for Extension: Extend existing behavior-Derived Class

Closed for Modification: Existing code remains unchanged-Base Class should not be m

E.g.: Mobile Services

Base Class: Similar Features
Derived Class: Different Features

LISKOV SUBSTITUTION PRINCIPLE

We should be to substitute base class objects with child class objects & this should not alter behavior/characteristics of program

E.g.: Mathematically we can say Rectangle can be a Square but in coding concept we can't be able to derived a Square from Rectangle. Instead, we should have a Common Interface as Shape to have common behavior for both classes.

INTERFACE SEGREGATION PRINCIPLE

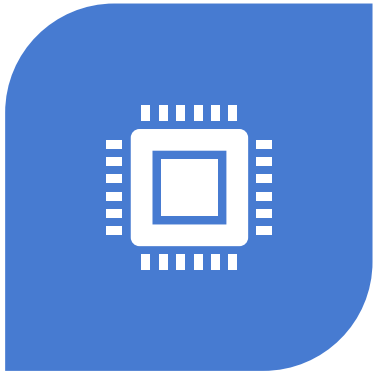
Clients should not be forced to depend upon interface that they don’t use.

Interface Pollution:

- Large Interfaces
- Unrelated Methods(Unsupported operative methods)

E.g.: Interface should relate methods – getByName valid for User , it can’t be valid other classes which doesn’t have name entity.

DEPENDENCY INVERSION PRINCIPLE



HIGH LEVEL MODULES SHOULD NOT DEPEND ON LOW LEVEL MODULES. BOTH SHOULD DEPEND ON ABSTRACTIONS

ABSTRACTION SHOULD NOT DEPEND UPON DETAILS. DETAILS SHOULD DEPEND UPON ABSTRACTIONS.

USE INTERFACES INSTEAD CREATING OBJECT INSTANCES. (REALLY POWERFUL - AUTOWIRED)