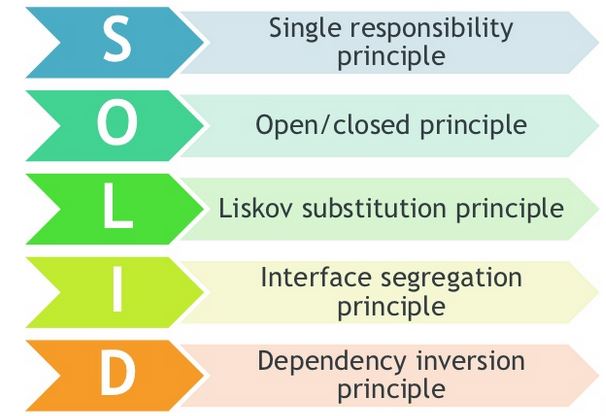
**SOLID Design in C#**



1. **Single Responsibility Principle**

The Single Responsibility principle states that every module or class should have responsibility over a single part of the functionality provided by the software, and responsibility should be entirely encapsulated by the class.

* There should not be more than one reason for a class to change.
* Class should always handle single functionality.

public class TicketReservation

{

public void OnlineReservations(ReservationDetails reservation)

{

// business logic, validation, etc...

// save booking to database

new OnlineReservationDB().Save(TicketReservation);

}

}

It only has one responsibility, it also has only one reason to change.

**Advantages of SRP**

* The primary value of software is ease of change.
* We must have a design that is easy to change.

1. **Open Closed Principle**

In object oriented programming the open/close principle states “Software Entities (classes, modules, functions, etc.) should be open for extension, but closed for modification”; that is, such an entity can allow its behavior to be extended without modifying its source code.

Implementation (of class or functions), once created, should be closed for further modification, in other words one should not modify an implementation (of a class or functions) of logic and functionality.

An electronic adapter in the wall is always close for modification.

An electronic adapter always provides a method of extension, so we can plug in an extension board of an adapter.

1. Liskov Substitution Principle

You should be able to use any derived class instead a parent class and have it behave in the same manner without modification.

Liskov Substitution Principle states that the derived classes should be perfectly substitutable for their base classes. If class **D** derived from **A** then **D** should be substitutable for **A**.

Orange class cannot substitute an Apple, which result in printing the color of Apple as Orange.

A father is a doctor whereas his son wants to become a cricketer. So here the son’s cannot replace his father even though they both belong to the same family hierarchy.

1. Interface Segregation Principle

The Interface Segregation Principle (ISP) states that clients should not be forced to depend upon interfaces that they do not use. When we have non-cohesive interfaces, the ISP guides us to create multiple, smaller, cohesive Interfaces.

1. Dependency Inversion Principle