Design Principle

SOLID Principle:

S-Single Responsibility Principle

O-Open-Closed Principle

L-Liskov's Substitution Principle

I-Interface Segregation Principle

D-Dependency Inversion Principle

Single Responsibility Principle:

If we have 2 reasons to change for a class, we have to split the functionality in two classes. Each class will handle only one responsibility and on future if we need to make one change we are going to make it in the class which handle it

Open Closed Principle:

Usually, many changes are involved when a new functionality is added to an existing unit tested application. Design and writing of the code should be done in a way that new functionality should be added with minimum changes in the existing code

Software entities like classes, modules and functions should be open for extension but closed for modifications.

Liskov's Substitution Principle:

We must make sure that the new derived classes just extend without replacing the functionality of old classes. Otherwise the new classes can produce undesired effects when they are used in existing program modules.

▪ If a program module is using a Base class, then the reference to the Base class can be replaced with a Derived class without affecting the functionality of the program module

Derived types must be completely substitutable for their base types.

▪ "S is subtype of T, then objects of type T may be replaced with objects of type S”

Derived types must be completely substitutable for their base types

▪ LSP is a particular definition of subtyping relation, called strong subtyping

▪ Introduced by Barbara Liskov

▪ Extension of Open Closed Principle