Alejandro de la cruz
7490 OOP

SOLID PRINCIPLES

- opened closed

the software type classes, must be open to extensions and will be closed by the modification

A class is closed if it is available for use by another class

Hotfix: Inheritance makes it possible for a specialization (a derived class) to reuse the generalization (a base class):

Abstractions do not have to depend on details.

Both low-order and high-order modules have to rely on abstractions

- engineering goals

Program Engineers are committed to producing quality products on time and within budget.

LISKOV SUBSTITUTION PRINCIPLE

Each class that inherits from another can be used as its parent without the need to know the differences between them.

INTERFACE SEGREGATION PRINCIPLE

An interface is a “window” into the functionality of a component

Clients should not be forced to depend upon interfaces that they do not use

DEPENDENCY INVERSION PRINCIPLE

Abstractions should not depend on details

High-level modules should not depend on low-level modules

Both low-level and high-level modules should depend on abstractions

-Background

On several occasions, there are contradictory definitions within the same paradigm • In addition, there are several other proposed principles that overlap and break the ideas in different ways.

- observations related to modularity

Good modularity should reduce domino effects once program changes happen as expected (and some unexpected).

- practical idioms and patterns

Best practices are methods or techniques that help developers adhere to principles, without having to consider the details of a situation at a theoretical level.