SOLID

Agenda

- What it is?
- How it works
- Why use it?

What it is…

- SOLID is basically 5 principles, which will help to create a good software architecture. You can see that all design patterns are based on these principles. SOLID is basically an acronym of the following:
  - **S** *is single responsibility principle*
  - **O** *stands for open closed principle*
  - **L** *Liskov substitution principle*
  - **I** *interface segregation principle*
  - **D** *Dependency inversion principle (DIP)*

Single Responsibility

- A class should take one responsibility and there should be one reason to change that class.

Now what does that mean?

Open-Closed principle

- Open for extensibility but closed for modification.

If we have a class which is being used by other clients, that class should be open for extensibility. The only reason to change the class should be to fix a bug in the class.

Liskov substitution principle

- Derived types must be substituted for base types.

This principle is simple but very important to understand. Child class should not break parent class’s type definition and behavior.

Interface segregation principle

- ISP is SOLID design principle which is used to help us achieve the LSP by having more granularity.

This principle states that any client should not be forced to use an interface which is irrelevant to it.

Dependency inversion principle

- This principle tells you not to write any tightly coupled code because that is a nightmare to maintain when the application is growing bigger and bigger. If a class depends on another class, then we need to change one class if something changes in that dependent class. We should always try to write loosely coupled class.

Q&A