Software Design Techniques and Mechanisms

Topic: SOLID Principles

Presenter: Drumea Vasile

Chișinău 2021

Overview

- SOLID is an acronym for 5 OOP design principles. [1]
- The intent is to make software:
  - more understandable
  - easier to maintain/test
  - extendable

Single Responsibility Principle

- Every piece of software (i.e. module, class or function) should be responsible over a single part of functionality provided by the software.
- Being responsible only for one thing, it will have only one reason to change.

Open Closed Principle

- Each piece of software should be open for extension and closed for modification.
- The behavior should be extended without needing to modify the internals.

Liskov Substitution Principle

- **Objects of a class should be substitutable with instances of the existing subclasses, without altering the functionalities of the software.**

Interface Segregation Principle

- A client shouldn’t be forced to implement an interface, or methods from an interface, that it doesn’t use.
- It is recommended to split larger interfaces into multiple smaller ones.

Dependency Inversion Principle

- Software entities must depend on abstractions, not on concrete things.
- Separate modules, that are located on different levels must not depend directly on each other, but should rely on abstractions.

References

1. https://itnext.io/solid-principles-explanation-and-examples-715b975dcad4
2. https://scotch.io/bar-talk/s-o-l-i-d-the-first-five-principles-of-object-oriented-design
3. Robert C. Martin, 2000, *Design Principles and Design Patterns*
4. Robert C. Martin, 2008, *Clean Code*

Thanks for your attention!
Questions?