SOLID Principles

Friday Presentation

Shrijan Subedi, 16th October 2020

SOLID Principles

Introduction

- Originally compiled by Robert C. Martin in the 1990s, SOLID principle provide a clear pathway for moving from tightly coupled code with poor cohesion and little encapsulation to the desired results of loosely coupled code, operating very cohesively and encapsulating the real needs of the business appropriately.
- SOLID is a mnemonic abbreviation for a set of design principles created for software development in object-oriented languages.
- Each letter in SOLID corresponds to a principle for development.

SOLID Principle

Letters in SOLID

- S: Single Responsibility
- O: Open / Closed Principle
- L: Liskov Substitution Principle
- I: Interface Segregation Principle
- D: Dependency Inversion Principle

The Object-Oriented Principles

Coupling

- Coupling in software development is defined as the degree to which a module, class, or other construct, is tied directly to others.
- The degree of coupling between two classes can be seen as how dependent one class is on the other.
- If a class is very tightly coupled to one or more classes, then you must use all of the classes that are coupled when you want to use only one of them. If a class has no coupling inherently it becomes useless.
- We can reduce coupling by defining standard connections or interfaces between two pieces of a system.
- Goal: Attain Low levels of coupling while creating a system that is functional, understandable , and maintainable.

The Object-Oriented Principles

Cohesion

- Cohesion is the extent to which two or more parts of a system are related and how they work together to create something more valuable than the individual parts.
- A software system cannot be cohesive if it is made up of excessively large classes that fulfills multiple concerns. Such a class cannot create cohesion outside of it.
- To create a cohesive system we can break out concerns into separate classes. Eg: UI, data loading, processing can be broken into separate classes.
- Goal: Attain high cohesion through seperation of needs into smaller units.

The Object-Oriented Principles

Encapsulation

- Encapsulation means to hide the underlying implementation details of a class while exposing functionality through use of interfaces.
- A class need not know the implementation specifics of another class but can interact with it through the interface that the class provides.
- Encapsulation allows us to freely modify parts of code and it’s implementation as long as the public interface doesn’t change.
- Goal: Achieve strong encapsulation through use of well defined interfaces

The Object-Oriented Principles

Interfaces

- An interface is a contract between a class and the outside world. When a class implements an interface, it promises to provide the behavior published by that interface.
- Interfaces don’t allow for implementation or to define fields.
- We use properties and methods to provide a template for other classes to implement.
- Implementing an interface allows a class to become more formal about the behavior it promises to provide.

SOLID Principles

Code Demonstration

References

- https://www.researchgate.net/publication/323935872_SOLID_Python_SOLID_principles_applied_to_a_dynamic_programming_language
- https://dev.to/ezzy1337/a-pythonic-guide-to-solid-design-principles-4c8i
- https://medium.com/@vubon.roy/solid-principles-with-python-examples-10e1f3d91259
- https://github.com/heykarimoff/solid.python/blob/master/1.srp.py