Solid Principles

Geir Ingi Sigurðsson

SOLID is an acronym coined by Robert C. Martin (also known as Uncle Bob). It stands for:

- S – Single responsibility Principle
- O - Open-closed Principle
- L - Liskov Substitution Principle
- I - Interface Segregation Principle
- D - Dependency Inversion Principle

Single responsibility principle

- In object-oriented programming, the Single Responsibility Principle (SRP) states that every object should have a single responsibility, and that responsibility should be entirely encapsulated by the class. All its services should be narrowly aligned with that responsibility.
- *A class should have one, and only one, reason to change*.
- A great heuristic for the Single Responsibility Principle is that we can describe what a class does without using “and” or “or.”

"Everything should be made as simple as possible, but no simpler." Albert Einstein

Open/Closed Principle

- Objects or entities should be open for extension but closed for modification.
- Software objects should be extendable without changing the object itself.
- One of the great advantages of a correct implementation of the Open/Closed Principle is that future developments of new functionality becomes much faster than before. This is usually achieved by identifying and modeling the common behavior into a higher layer of abstraction. For introducing new behavior, it will then be just a matter of implementing a new concrete component to be plugged in.

'I'll tell you what the problem is, mate,' said Majikthise, 'demarcation, that's the problem!' [...] 'That's right' shouted Vroomfondel, 'we demand rigidly defined areas of doubt and uncertainty!'" Douglas Adams, The Hitchhiker's Guide to the Galaxy

Liskov Substitution Principle

- Derived classes should keep promises made by base classes. This also applies to interfaces, and it means that classes that implement some interface should keep the promises made by that interface.
- Everything regarding the LSP is about the behavior of a class. So, the "is-a" relationship in OOP must be seen from the perspective of exposed behavior, not the internal structure.

"If it looks like a duck and quacks like a duck, but needs batteries, you probably have the wrong abstraction" Spring Framework Guru

Interface Segregation Principle

Clients should not be forced to depend on methods they do not use.— Robert Martin

- The dependency should be on the interface, the whole interface, and nothing but the interface.
- The goal of the ISP is to reduce the side effects and the number of changes needed in a system by splitting the software into multiple, smaller and independent parts grouped by functionality.

Dependency Inversion Principle

- Decouple conventional dependency relationships established from high-level, policy-setting modules to low-level, dependency modules by inversion for the purpose of rendering high-level modules independent of the low-level module implementation details.
- This is achieved by having high- and low-level modules depend on an abstraction.

"Depend on abstractions, not on concretions". Robert C. Martin

Questions?

- The answer to the Ultimate Question of Life, The Universe, and Everything?
- Little or big endian?
- Red or blue pill?

"He who asks is a fool for five minutes, but he who does not ask remains a fool forever" Mark Twain

Thank you!

Contact: geir.sigurdsson@bouvet.no