SOLID Principles: Do You Really Understand Them?

Video

https://youtu.be/kF7rQmSRlq0?si=Jdm5zZx2AazLaPBM

Notes

Single Responsibility

- Your class should have a single responsibility
- Making change to a class with many responsibilities means you are more likely to introduce bugs
- A class should only have one reason to change
- If many developers needed to make changes to various methods (which are all held inside of one class) then it would mean they are all working on the same file which can lead to merge conflicts and bugs

Open Closed

- Your code should be open to extension but closed to modification
- If additional functionality needs to be added, existing classes or methods should not need editing to do this
- This can be done with decorator patterns and extension methods

Liskov Substitution

- A child class should be able to do everything that a parent class can

- This means that, given that class B is a subclass of class A, we should be able to pass an object of class B to any method that expects an object of class A and the method should not give any weird output in that case
- For example, a `Child` class cannot do everything that a `Parent` class can (such as `go_to_work()` or `make_dinner()`). This would break the Liskov substitution principle. Instead `Child` and `Parent` should be child classes of the parent class `Human`. This way `Adult` can have 'adult methods' which are unque to it but `Human` can store methods that apply to both `Adult` and `Child`, such as `walk()`.

Interface Segregation

- Interfaces provide a contract that your classes need to implement
- If you have bulky interfaces then your classes are forced to implement method that they might not use
- Split methods into different interfaces (one for reading, writing, deleting, etc)

Dependency Inversion

- High-Level modules shouldn't depend on Low-Level models. They should both depend on an abstraction.

Resources

- https://www.freecodecamp.org/news/solid-principles-explained-in-plain-english/