



WRITING GOOD CODE WITH ACRONYMS

Jan Kalbermatter



CUPID 🏹💖 - for Joyful Coding

Created by Daniel Terhorst North

“I believe that there are properties or characteristics of software that make it a joy to work with. The more your code has these qualities, the more joyful it is to work with; but everything is a trade-off so you should always consider your context.”

Presentation in 2017 - Why Every Element of SOLID is Wrong



Remember SOLID?

Single Responsibility Principle

Code should have one, and only one, reason to change



Single Responsibility Principle

Just because you *can* doesn't mean you *should*.

Open/Closed Principle

Open for extension, closed for modification



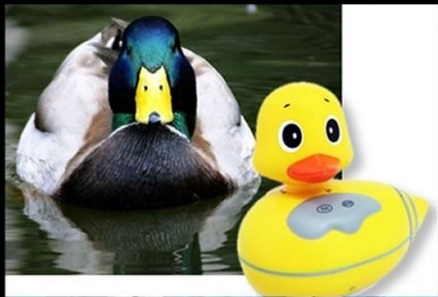
Open-Closed Principle

Open-chest surgery isn't needed when putting on a coat.

Remember SOLID?

Liskov Substitution Principle

Being able to replace the subclass everywhere the parent class is used without breaking the meaning of the code



Liskov Substitution Principle

If it looks like a duck and quacks like a duck but needs batteries, you probably have the wrong abstraction.

Interface Segregation Principle

Clients should not be forced to depend upon interfaces that they do not use



Interface Segregation Principle

You want me to plug this in *where*?

Remember SOLID?

Dependency Inversion Principle

High-level modules should not depend on low-level modules -> Both should depend on abstractions



Dependency Inversion Principle

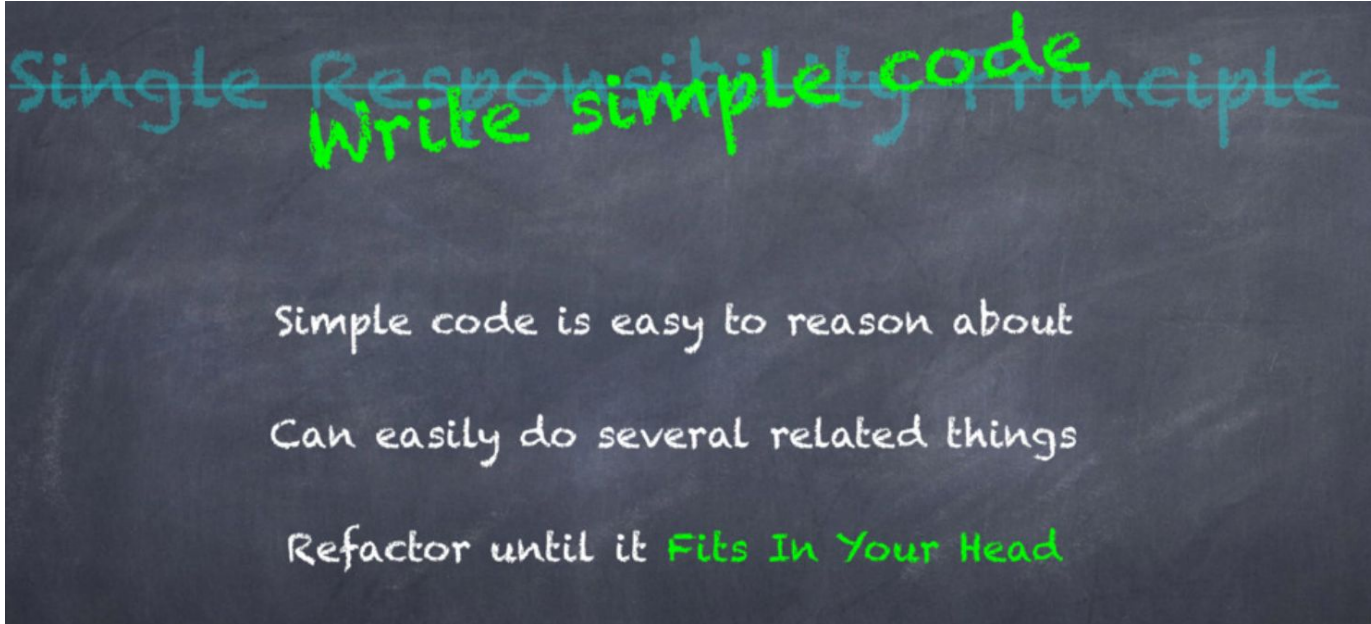
Would you solder a lamp directly to the electrical wiring in a wall?

SOLID++

- Balanced Abstraction 🏛️
 - Group same level of abstraction
- Least Astonishment (WTF) 🤖
 - Minimize wtf/min



Pointlessly Vague Principle





Cruft Accretion Principle

~~Open-Closed principle~~
Write simple code!

Simple code is easy to change

Simple code is easy to test

Simple code is both open and closed



Drucker's Warning Principle

~~Liskov Substitution Principle~~
Write simple code!

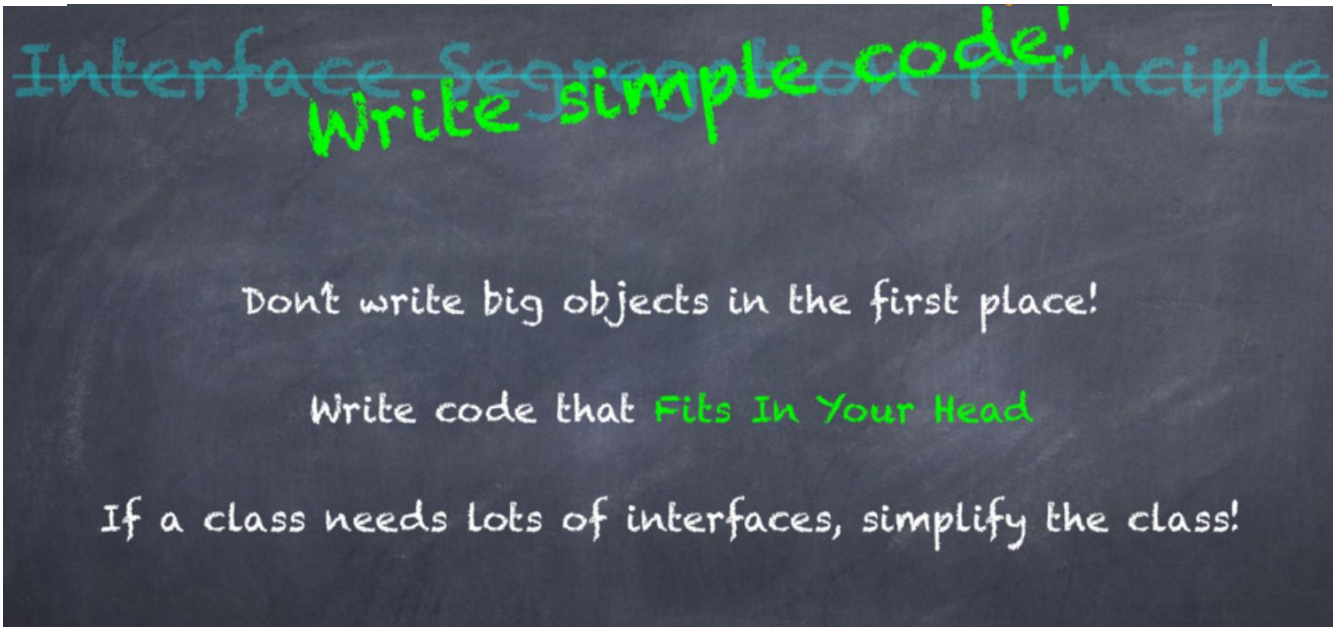
What about **acts-like-a, can-be-used-as-a?**

Composition is simpler than inheritance

Try to avoid object hierarchies altogether



Stable Door Principle





Wrong Goal Principle





Write simple Code?

Single Responsibility

Open/Closed

Liskov Substitution

Write simple code!

Interface Segregation

Dependency Inversion



What CUPID is all about

- Writing Software that is a Joy to work with
 - Caring about out fellow programmers ❤️
- Properties instead of Principles
 - Three Properties the CUPID Properties all have
 - Practical
 - Human
 - Layered



Bringing joy by making your software...

Composable

aka. plays well with others

- Easy to reuse
- Small surface area
- Intention Revealing
- Minimal Dependencies

Unix Philosophy

aka. does one thing well

- Sounds just like SRP?
- Outside-in perspective



Bringing joy by making your software...

Predictable 🧠

aka. does what you expect

- Behaves as expected
- Deterministic
- Observable

Idiomatic 🌳

aka. feels natural

- Conform to language idioms
- Conform with local idioms (made up by team)



Bringing joy by making your software...

Domain-based 

aka. the solution domain models the problem domain in
language and structure

- for language
- for structure



Is CUPID stupid or is SOLID... not so solid?

CUPID more straight forward at first glance

SOLID = Set of principle to use during development

CUPID = Properties that the finished code should have

CUPID is less restrictive -> does not care how you achieve the five properties

They follow the same goal!



Thanks for listening

More on Dan North & CUPID:

<https://dannorth.net/2022/02/10/cupid-for-joyful-coding/>

Dan North - Why Every Element of SOLID is Wrong:

<https://speakerdeck.com/tastapod/why-every-element-of-solid-is-wrong>

Jan Kalbermatter

✉ jan.kalbermatter@gmail.com