The SOLID Principles

Joshua Lewis

School of Electrical and Information Engineering
University of the Witwatersrand

2013

Context

5 Principles of OO design

'Collected' by Robert Martin (Uncle Bob) early 2000s

Code will change

- Rigid - Cascade of changes
- Fragile - Unexpected breakages
- Immobile - Entangled, can't reuse
- Viscous - Hard to do right thing

Common Smells

Code will change

- Rigid - Cascade of changes
- Fragile - Unexpected breakages
- Immobile - Entangled, can't reuse
- Viscous - Hard to do right thing

Related to dependency management

Well designed code

What we'd like

- Flexible
- Robust
- Reusable
- 'Pit of success'

Well designed code

What we'd like

- Flexible
- Robust
- Reusable
- 'Pit of success'

Principles to follow to get good code

Some gotchas

- Various (competing) definitions
- Lots of misunderstandings & misconceptions
- Lots of confusion (eg DIP and DI)

Some gotchas

- Various (competing) definitions
- Lots of misunderstandings & misconceptions
- Lots of confusion (eg DIP and DI)

Well-known 'motivational' posters

Single Responsibility Principle

Single Responsibility Principle

A class should have one, and only one, reason to change

Open/Closed Principle

Open/Closed Principle

You should be able to extend a class's behavior, without modifying it

Liskov Substitution Principle

Liskov Substitution Principle

Derived classes must be substitutable for their base classes

Interface Segregation Principle

Interface Segregation Principle

Make fine grained interfaces that are client specific

Dependency Inversion Principle

Dependency Inversion Principle

Depend on abstractions, not on concretions

Dependency Inversion Principle

- High-level modules should not depend on low-level modules. Both should depend on abstractions
- Abstractions should not depend upon details. Details should depend upon abstractions

Great destination
Terrible roadmap

Code Examples and Exercises

Let's code