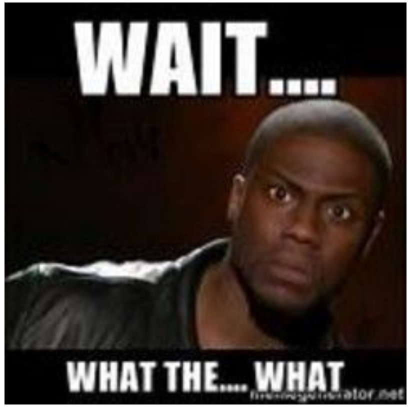
Ashot Nalbandyan

OOP Design Principles

Be solid, dry and kiss your code, slap your functions



Be **SOLID**, **DRY** and **KISS** your code, **SLAP** your functions

Basic OOP Design Principles

- SOLID
- KISS
- DRY
- SLAP
- YAGNI
- Others

SOLID

Single **R**esponsibility **P**rinciple (**SRP**)

S**O**LID

<u>**O**</u>pen-**C**losed **P**rinciple (**OCP**)

Software entities (classes, modules, functions, etc.) should be open for extension, but closed for modification

SOLID

Liskov **S**ubstitution **P**rinciple (**LSP**)

Objects of a superclass shall be replaceable with objects of its subclasses without breaking the application.

SOL**I**D

Interface **S**egregation **P**rinciple (**ISP**)

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

SOLI**D**

Dependency **I**nversion **P**rinciple (**DIP**)

- High-level modules should not depend on low-level modules. Both should depend on abstractions (e.g., interfaces).
- Abstractions should not depend on details. Details (concrete implementations) should depend on abstractions.

KISS

Keep **I**t **S**tupid **S**imple

- **K**eep **i**t **s**imple, **s**illy
- **K**eep **i**t **s**hort and **s**imple
- **K**eep **i**t **s**imple and **s**traightforward
- **K**eep **i**t **s**mall and **s**imple
- **K**eep **i**t **s**imple, **s**oldier
- **K**eep **i**t **s**imple, **s**ailor

DRY

Don’t **R**epeat **Y**ourself

Violations of DRY are typically referred to as WET solutions

- **W**rite **e**very **t**ime
- **W**rite **e**verything **t**wice
- **W**e **e**njoy **t**yping
- **W**aste **e**veryone's **t**ime

SLAP

Single **L**evel of **A**bstraction **P**rinciple

Functions should do just one thing, and they should do it well. Robert Martin

YAGNI

You **A**in't **G**onna **N**eed **I**t

A programmer should not add functionality until deemed necessary

Others

- Encapsulate what changes
- Favor composition over inheritance
- Program to interfaces, not implementation
- Delegation principle

Q&A

Thank you!