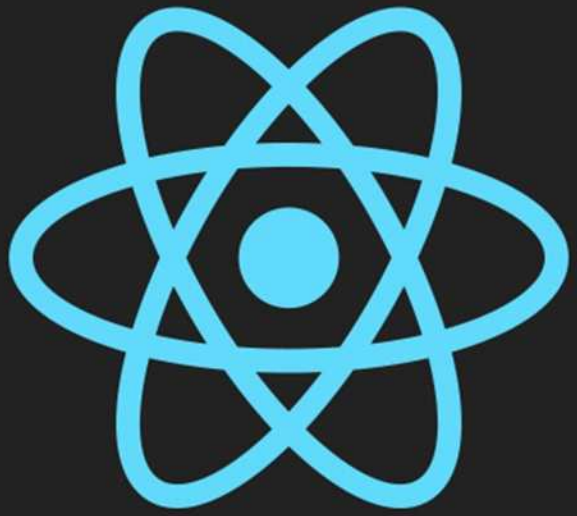


S.O.L.I.D

Luzern, 29. April 2021

Marco Birrer

ALCOR Academy Training

CSS Versicherung

S.O.L.I.D – Single Responsibility

“A class should have only one reason to change”

S.O.L.I.D – Single Responsibility

The classes you write, should not be a swiss army knife.
They should do one thing, and to that one thing well.

S.O.L.I.D – Single Responsibility

- ✓ **Testing** – more easy to test
- ✓ **Lower Coupling** – less dependencies
- ✓ **Organization** – smaller is more readable / understandable

S.O.L.I.D – Single Responsibility

Large Class
Divergend Change

CancelManager

markPolicyCanceled
createConsequeces
save

Split

CancelManager

markPolicy

ConsequencesManager

create

ProcessManager

save

S.O.L.I.D – Open / Closed Principle

“Software components should be open for extension, but closed for modification”

S.O.L.I.D – Open / Closed Principle

Open Chest Surgery Is Not Needed
When Putting On A Coat!

S.O.L.I.D – Open / Closed Principle

Switch
Primitive Obsession
Shotgun Surgery

Rover

```
switch(movement) {
    case "MoveForward": …
    case "RotateRight": …
    …
}
```

Pattern

Rover

+ move(command)

command.execute()

<Interface>
Command

+ execute()

Which Pattern we used?

MoveForward

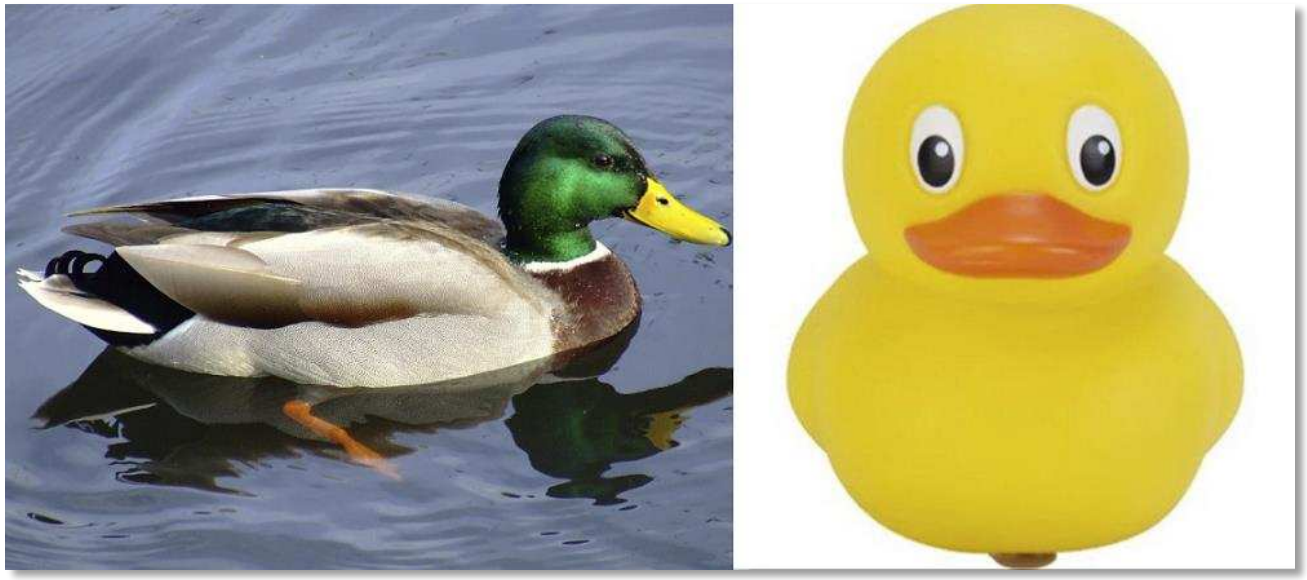
RotateRight

…

S.O.L.I.D – Liskov substitution principle

"*Derived classes should be able to substitute their base classes without the behavior of your code changing.*"

S.O.L.I.D – Liskov substitution principle



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

S.O.L.I.D – Liskov substitution principle

Refused Bequest

MobSession

employee. startMob()

<Interface>
Employee
+ startMob()

Better
Abstraction

<Interface>
Mobable
+ startMob()

Developer

```
void startMob() {
  ….
}
```

Manager

```
void startMob() {
   throw new Error()
}
```

S.O.L.I.D – Interface segregation principle

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

S.O.L.I.D – Interface segregation principle

Where to Plug in here?

S.O.L.I.D – Interface segregation principle

Large Class
Refused Bequest
Alternative Classes differnt Interfaces
Duplicated Code

Vehicle

setPrice
setColor
forward
backward
fly

Split

Vehicle

setPrice
setColor

Movable

forward
backward

Flyable

fly

<Vehicle> Car

setPrice
setColor
forward
backward
fly

<Vehicle, Moveable> Car

setPrice
setColor
forward
backward

S.O.L.I.**D** – Dependency Inversion Principle

"*High level modules should not depend on low level modules they should both depend on abstractions.*"

"*Abstractions should not depend on details. Details should depend on abstractions.*"

S.O.L.I.**D** – Dependency Inversion Principle

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

S.O.L.I.D – Dependency Inversion Principle

no smell directly ...

Rover

```
switch
   cas
   case L: …
   …
}
```

Depending on Interface

Rover

+ move(command)

command.execute()

<Interface>
Command

+ execute()

MoveForward

RotateRight

...

References

- https://dzone.com
- https://springframework.guru/principles-of-object-oriented-design
- https://www.baeldung.com/solid-principles
- https://alcor.acedemy
- https://miro.medium.com

THANK YOU
FOR
YOUR ATTENTION