SOLID

En introduksjon

Agenda

- Presentasjon av oss.
- Single Responsibility Principle
- Open / Closed Principle
- Mat?
- Liskov Substitution Principle
- Interface Segregation Principle
- Dependency Inversion Principle
- Oppsummering


ARE YOU A
PROFESSIONAL?
Robert C. Martin


S
SRP
Single Responsibility Principle
O
OCP
Open/Closed Principle
L
LSP
Liskov Substitution Principle
I
ISP
Interface Segregation Principle
D
DIP
Dependency Inversion Principle

SRP

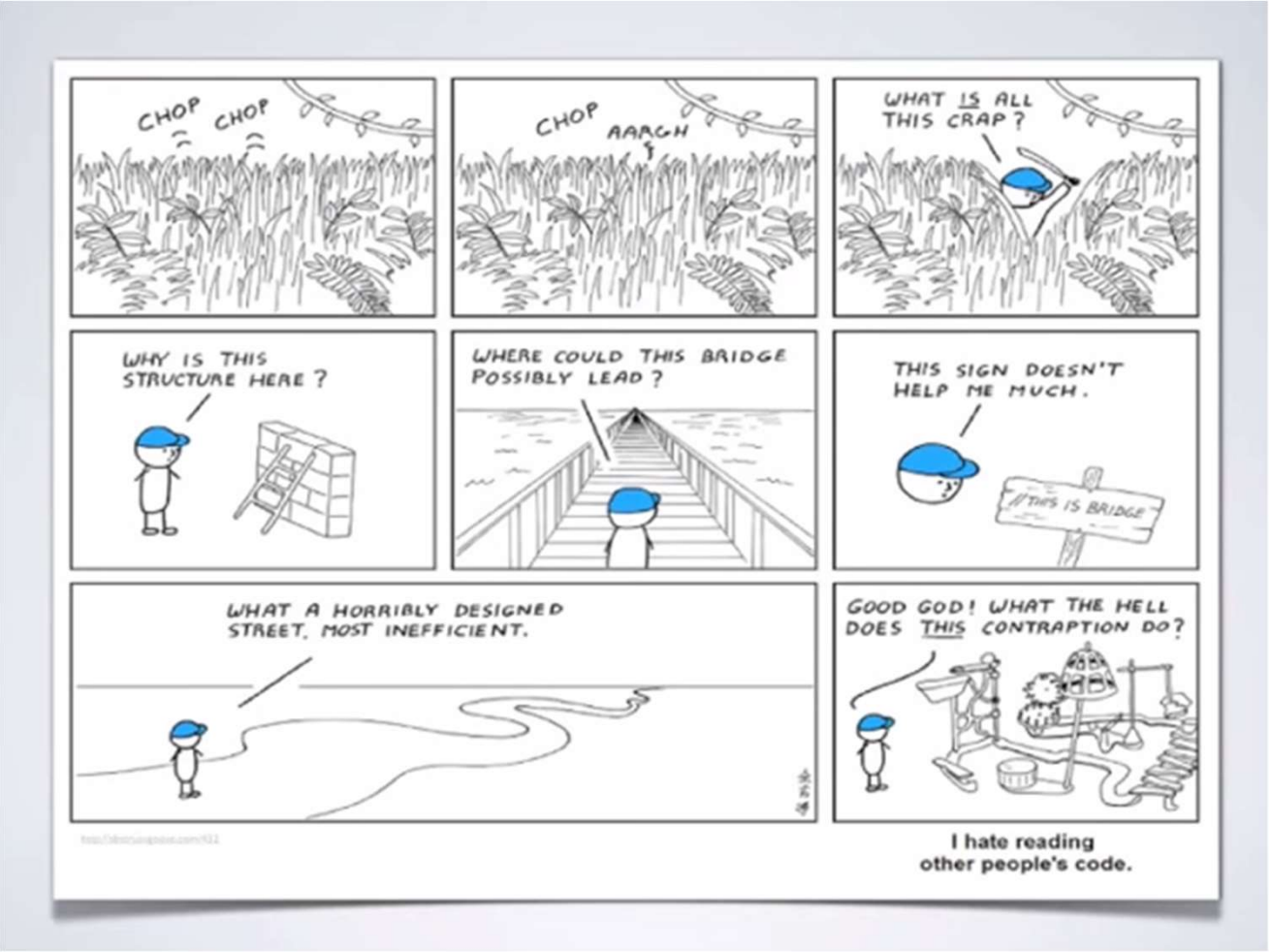
Single Responsibility Principle

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

SINGLE RESPONSIBILITY PRINCIPLE

Just Because You Can, Doesn't Mean You Should

© Derick Bailey


CHOP CHOP
CHOP AARGH
WHAT IS ALL THIS CRAP?
WHY IS THIS STRUCTURE HERE?
WHERE COULD THIS BRIDGE POSSIBLY LEAD?
THIS SIGN DOESN'T HELP ME MUCH.
THIS IS BRIDGE
WHAT A HORRIBLY DESIGNED STREET. MOST INEFFICIENT.
GOOD GOD! WHAT THE HELL DOES THIS CONTRAPTION DO?
I hate reading other people's code.

Praktisk Oppgave

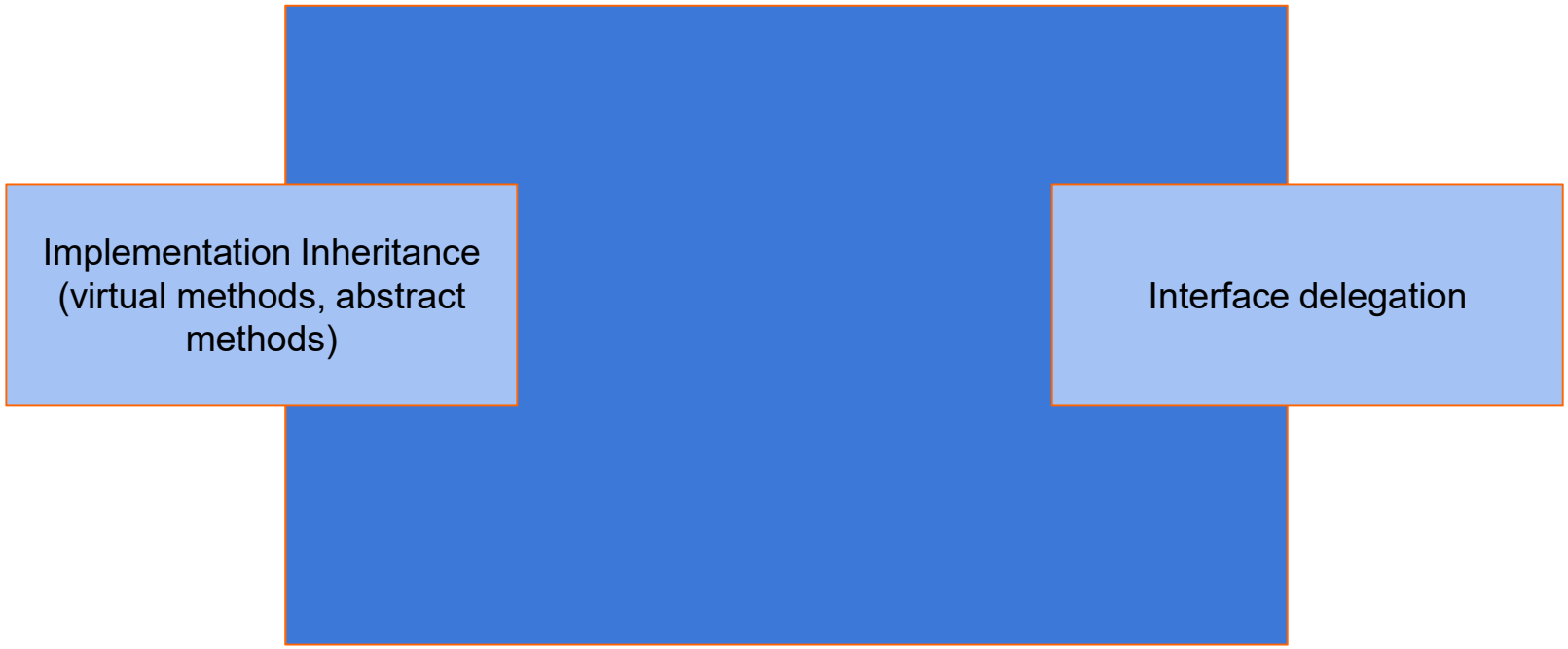
OCP

Open/Closed Principle

“software entities … should be open for extension, but closed for modification”

“You should be able to extend the behavior of a system *without having to modify that system*”

EXTENSION POINTS



OPEN CLOSED PRINCIPLE

Open Chest Surgery Is Not Needed When Putting On A Coat

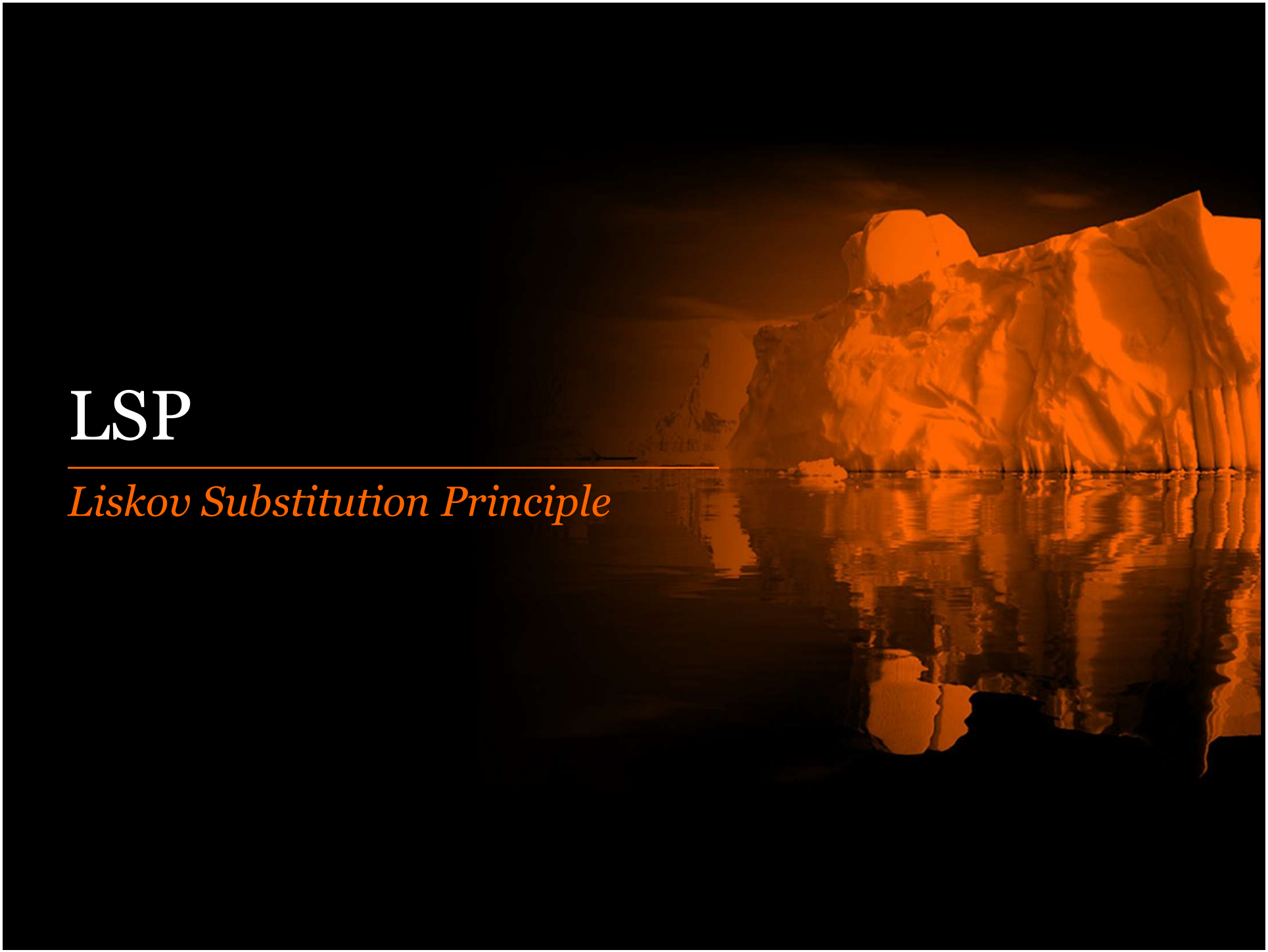
© Derick Bailey


STOP

Praktisk Oppgave

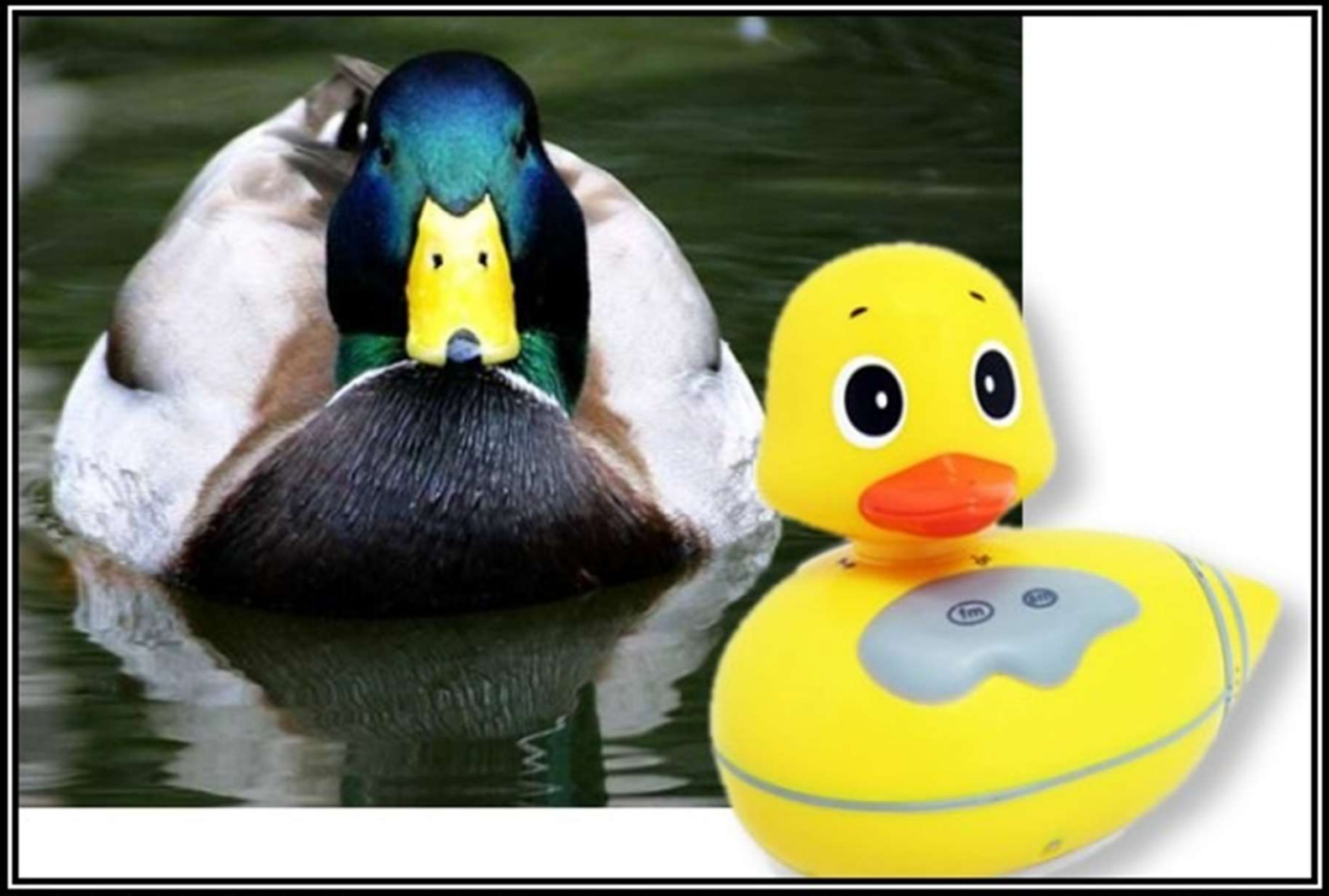
LSP

Liskov Substitution Principle



“If S is a subtype of T, then objects of type T may be replaced with objects of type S, without breaking the program”

"Derived classes must be substitutable for their base classes"



LISKOV SUBSTITUTION PRINCIPLE

If It Looks Like A Duck, Quacks Like A Duck, But Needs Batteries - You Probably Have The Wrong Abstraction

© Derick Bailey

Litt teori – «Regler»

- Contravariance of method arguments in sub class
- Covariance of return types in the sub class
- No new exception types are allowed to be thrown, unless they are sub-types of previously used ones
- Preconditions cannot be strengthened in a subtype
- Postconditions cannot be weakened in a subtype
- Data invariants

Praktisk Oppgave

ISP

Interface Segregation Principle

“many client-specific interfaces are better than one general-purpose interface”

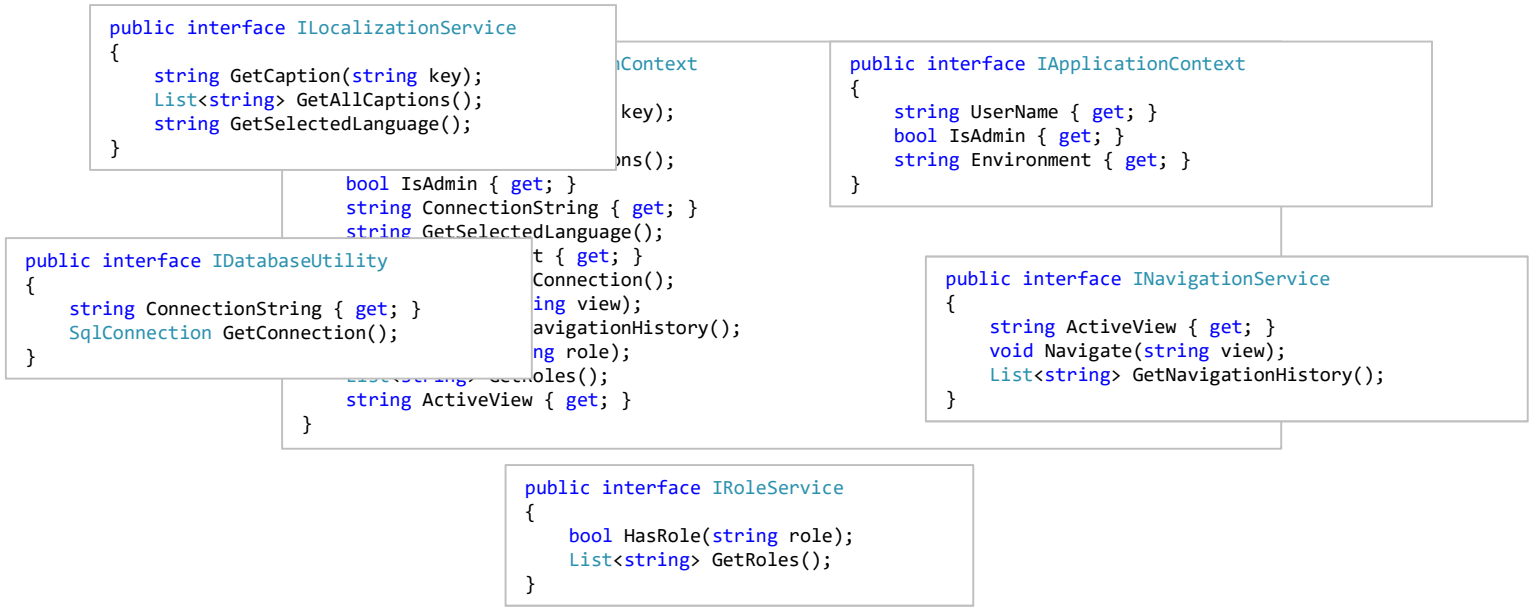
INTERFACE SEGREGATION PRINCIPLE

You Want Me To Plug This In, Where?

© Derick Bailey

Hva prøver vi å løse?

Unngå «God-interfaces» som prøver å gjøre alt, og som lar alle gjøre alt



Hva er fordelene?

- Gir bedre kontroll over hvem som kan gjøre hva
- Det er lettere å implementere små og spesifikke interfacer
- Lettere å bytte ut deler av systemet – trenger ikke å implementere et enormt interface for å erstatte en enkelt feature
- Lettere å teste (enklere å mocke)

Praktisk Oppgave

DIP

Dependency Inversion Principle

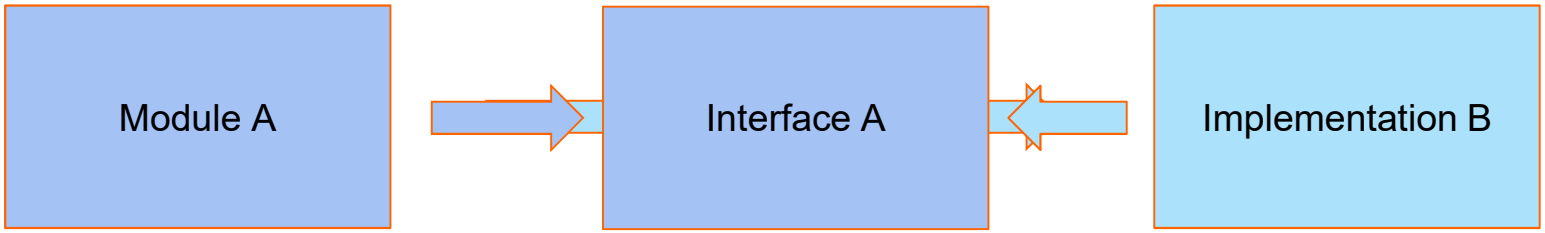
DEPENDENCY INVERSION PRINCIPLE

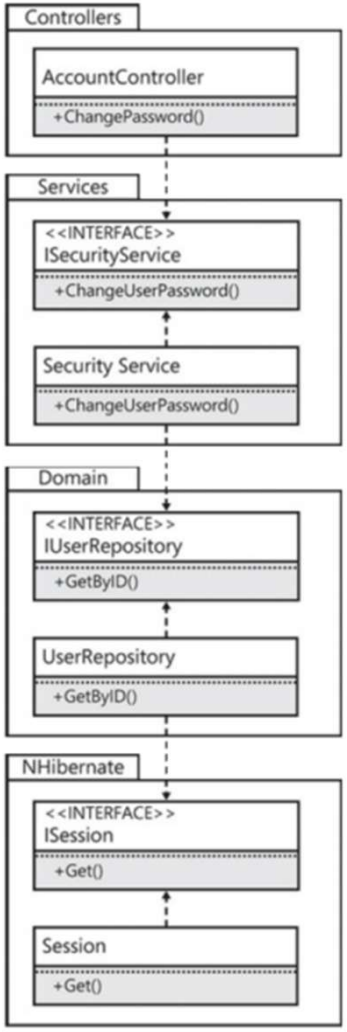
Would You Solder A Lamp Directly To The Electrical Wiring In A Wall?

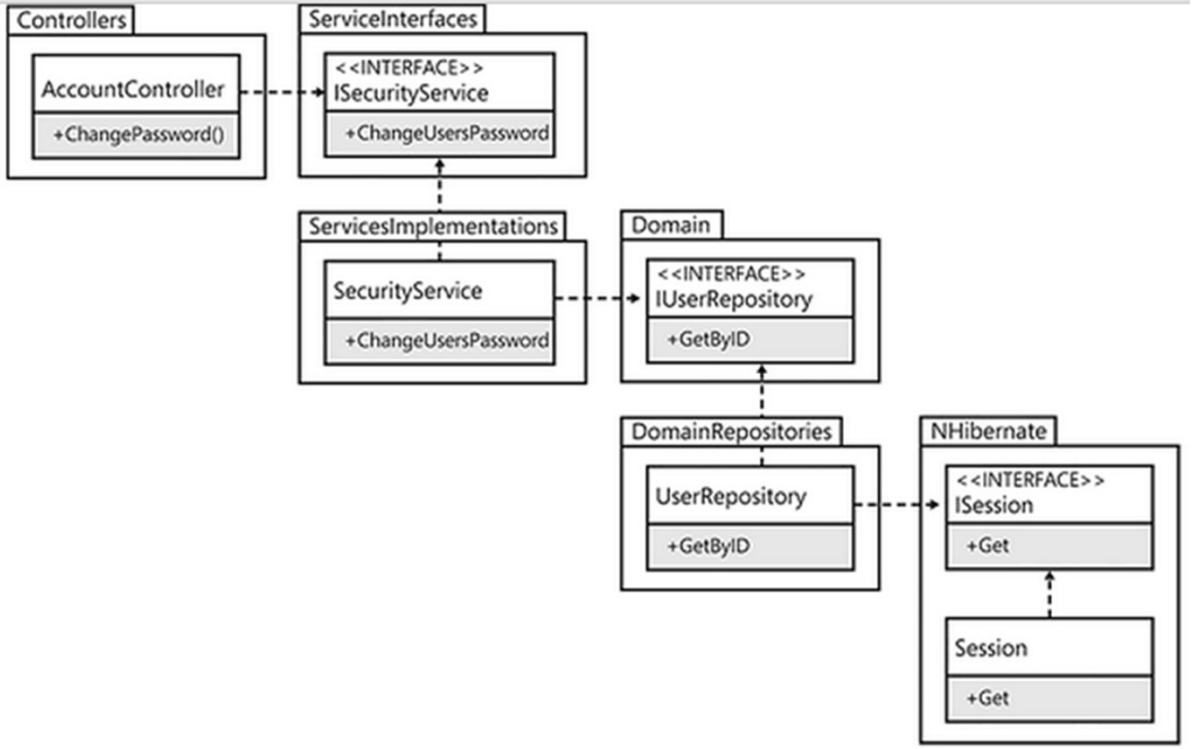
© Derick Bailey

“Depend on abstractions, not on concretions”

“Abstractions should not depend on details. Details should depend on abstractions.”


Module A
Interface A
Implementation B


Controllers
AccountController
+ChangePassword()
Services
<<INTERFACE>>
ISecurityService
+ChangeUserPassword()
Security Service
+ChangeUserPassword()
Domain
<<INTERFACE>>
IUserRepository
+GetByID()
UserRepository
+GetByID()
NHibernate
<<INTERFACE>>
ISession
+Get()
Session
+Get()


Controllers
AccountController
+ChangePassword()
ServiceInterfaces
<<INTERFACE>>
ISecurityService
+ChangeUsersPassword
ServicesImplementations
SecurityService
+ChangeUsersPassword
Domain
<<INTERFACE>>
IUserRepository
+GetByID
DomainRepositories
UserRepository
+GetByID
NHibernate
<<INTERFACE>>
ISession
+Get
Session
+Get

Praktisk Oppgave

Til ettertanke…

- Finn en balanse mellom de ulike akronymene:
  - SOLID, KISS, YAGNI, The rule of three, DRY osv
- Tenk lesbar kode som er lett å vedlikeholde
- Det blir mer kode
- Finn et kompromiss, bestem «hvilken del» du vil gjøre koden SOLID. Statisk (intrinsic) vs dynamisk del.

Ha SOLID i bakhodet.

Det vil hjelpe deg å se «code smells», og vil gi deg et verktøy for hva du kan gjøre for å fikse opp i problemene.


That's all Folks!