JAVA FAKTURA

Object Oriented Programming

Nothing is more dangerous than an idea, when you have only one idea.

Émile Chartier

About me

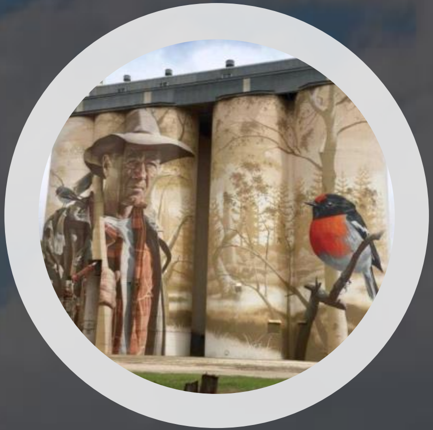
JAVA FAKTURA

- Scrum Master @ Harman
- Former Java/Mobile developer
- Co-organizer of Javafaktura

www.linkedin.com/in/piotr-kotynia

Grain Elevators

https://www.facebook.com/pg/Najpiękniejsze-elewatory-zbożowe-1507269532915111



Agenda

JAVA FAKTURA

- SOLID
- DRY
- KISS
- YAGNI
- TDA

Murphy's Law

- **Whatever can go wrong, will go wrong**. So a solution is better the less possibilities there are for something to go wrong.

Solid

JAVA FAKTURA

- Single responsibility principle
- Open–closed principle
- Liskov substitution principle
- Interface segregation principle
- Dependency inversion principle

Single responsibility principle

- every object should have a single responsibility and that all of its services should be aligned with that responsibility.
- “Responsibility” is defined as “a reason to change”

Single Responsibility Principle
Just because you *can* doesn’t mean you *should*.

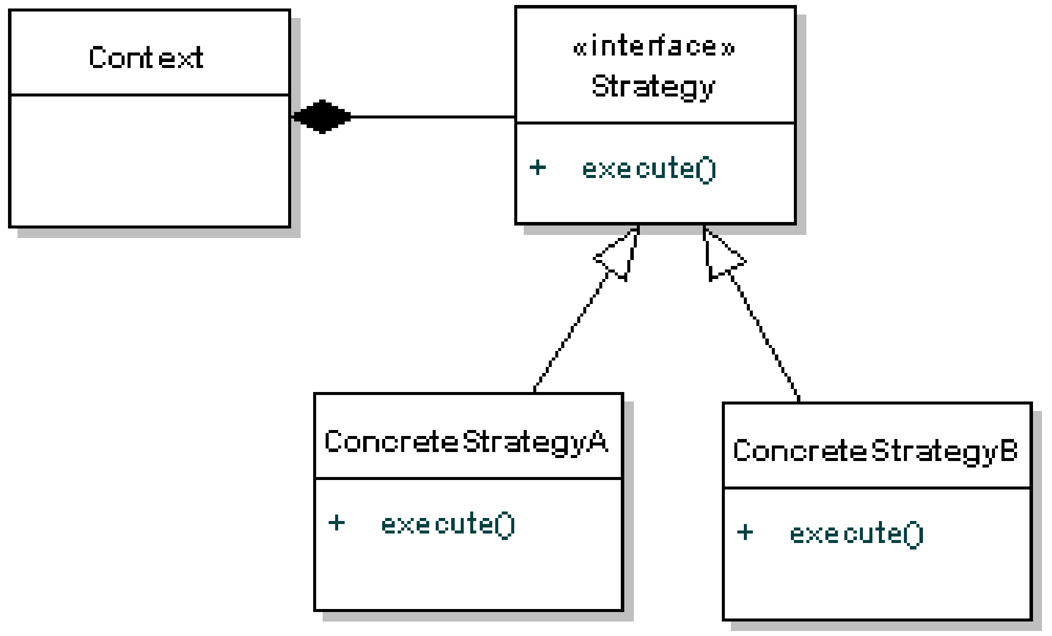
Open–closed principle

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

Strategy Pattern to the rescue

- the behaviors of a class should not be inherited. Instead they should be encapsulated using interfaces. This is compatible with the **open/closed principle** (OCP), which proposes that classes should be **open** for extension but **closed** for modification.

Strategy Pattern

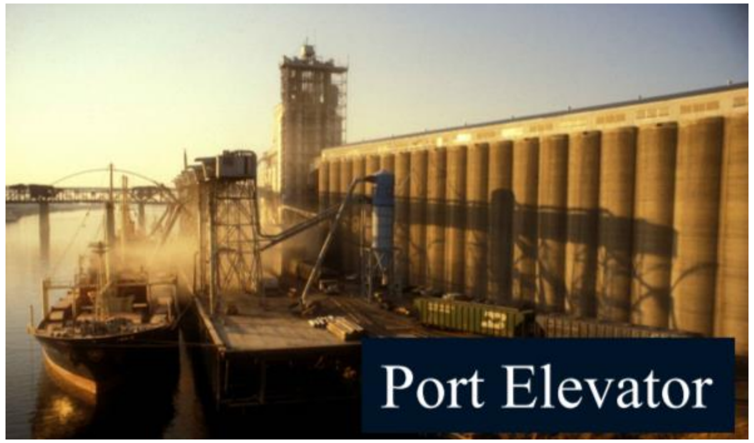


Liskov Substitution Principle

JAVA FAKTURA

Functions that use pointers to base classes must be able to use objects of derived classes without knowing it.

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

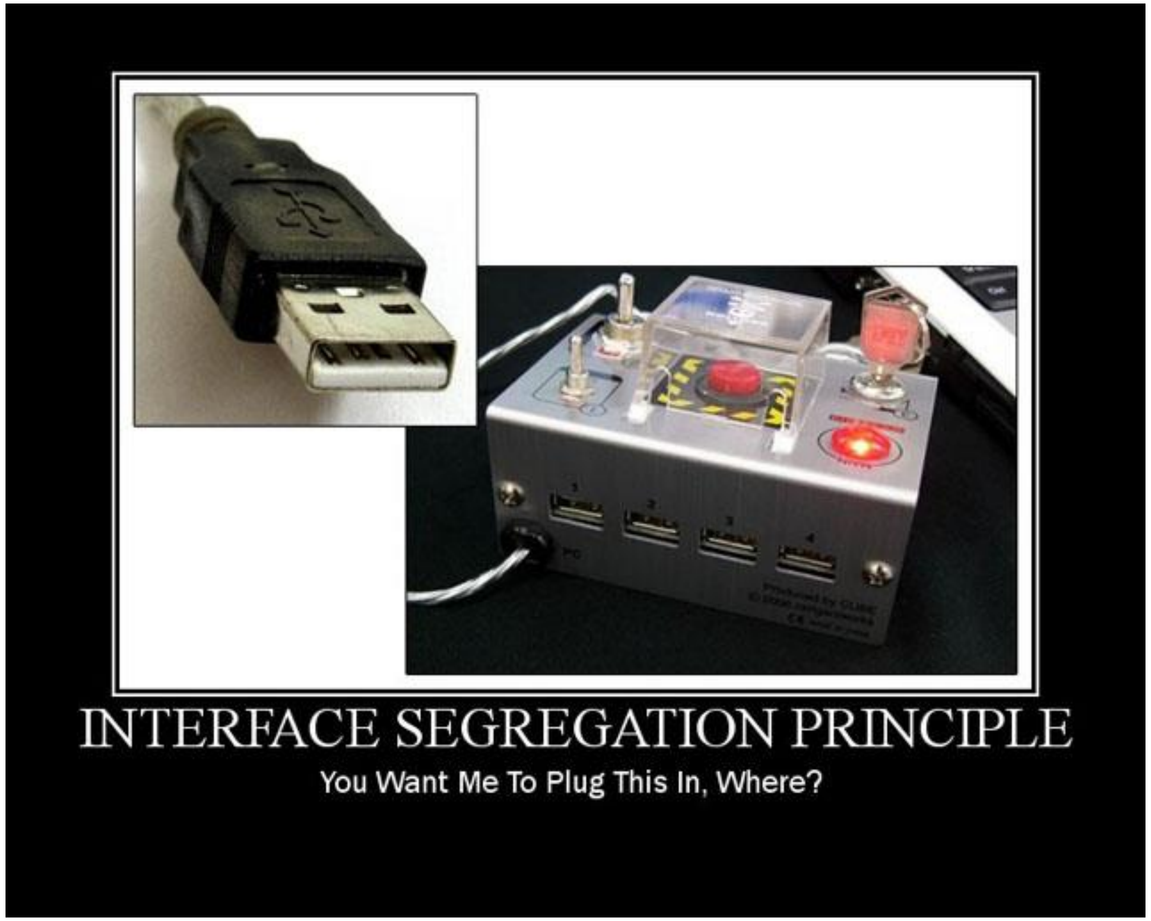


JAVA FAKTURA

Elevator’s transport types

Interface segregation principle

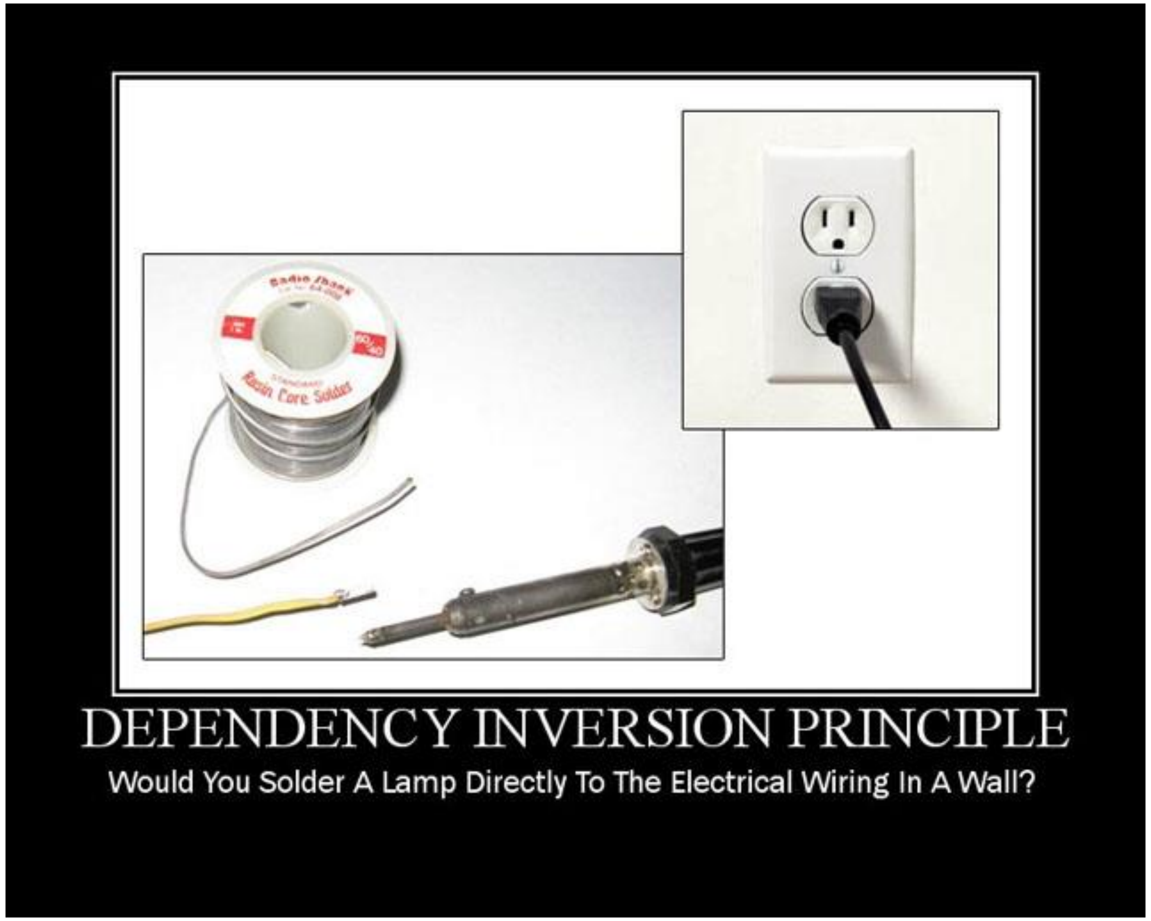
Many client-specific interfaces are better than one general-purpose interface.


INTERFACE SEGREGATION PRINCIPLE
You Want Me To Plug This In, Where?

Dependency inversion principle

- 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


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

Keep It Simple Stupid

JAVA FAKTURA

- Any fool can write code that a computer can understand. Good programmers write code that humans can understand.

  Martin Fowler

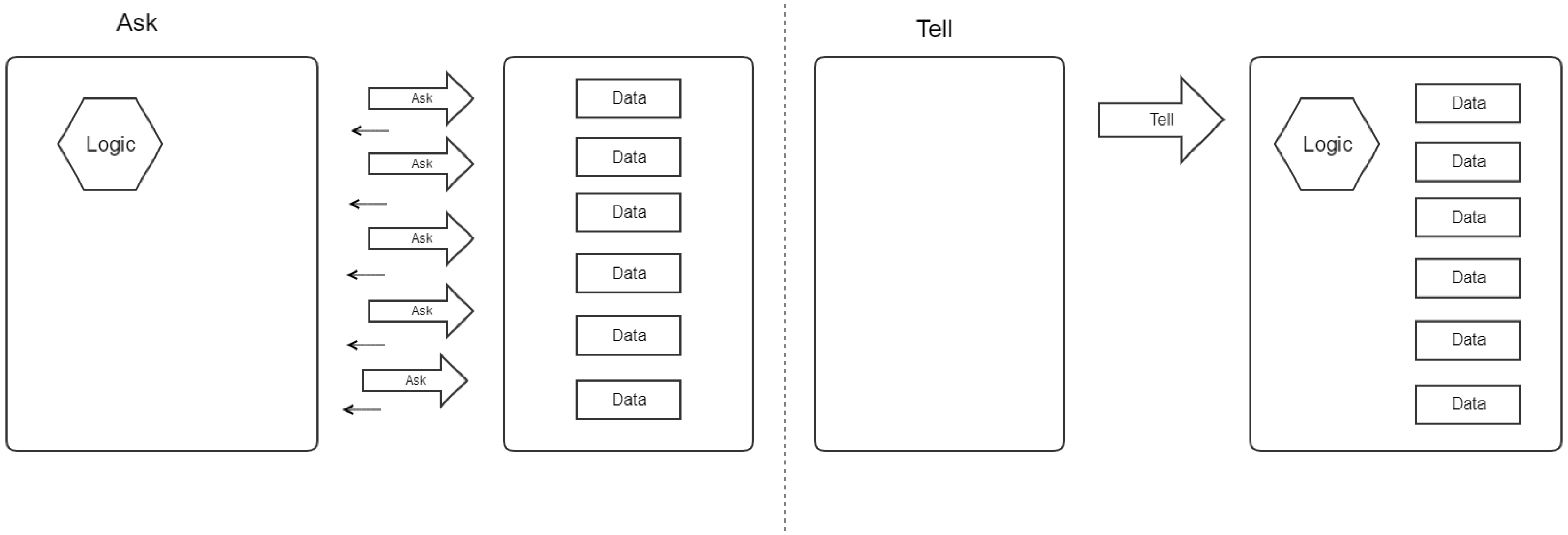
Don’t Repeat Yourself

JAVA **FAKTURA**

- Every piece of knowledge must have a single, unambiguous, authoritative representation within a system.

- Variants and Alternative Names:
  - Single Point of Truth (SPOT)
  - Single Source of Truth (SSOT)

Tell Don’t Ask



You Ain't Gonna Need It

"Always implement things when you actually need them, never when you just foresee that you need them”

Ron Jeffries

Thank YOU

And good luck fighting Murphy's Laws