EAPLI

Princípios de Design OO

Paulo Gandra de Sousa
pag@isep.ipp.pt

Principios

GRASP

- Information Expert
- Low Coupling
- High Cohesion
- Creator
- Controller
- Polymorphism
- Indirection
- Pure Fabrication
- Protected Variations

SOLID

- Single Responsibility Principle
- Open/Close
- Liskov Substitution Principle
- Interface Segregation
- Dependency Inversion

What's wrong with this code?

```
class Employee {
   private int number;
   private double salary;
   private List skil = new ArrayList<String>;

   ...
   public int getNumber() { … }
   public void setNumber(int n) { … }
   public double getSalary() { … }
   public setSalary(double s) { … }
   public List getSkills() { … }
   public void setSkills(List s) { … }
}
```

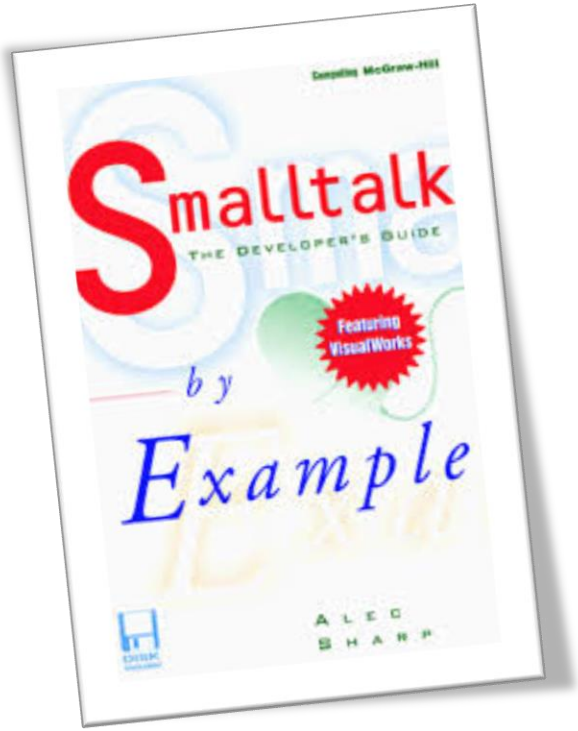
GETTERS AND SETTERS ARE EVIL

Information Hiding

Segregation of the design decisions that are most likely to change, thus protecting other parts from extensive modification if the design decision is changed.

Parnas, D.L. (December 1972). **"On the Criteria To Be Used in Decomposing Systems into Modules"**. Communications of the ACM. 15 (12): 1053–58. doi:10.1145/361598.361623.

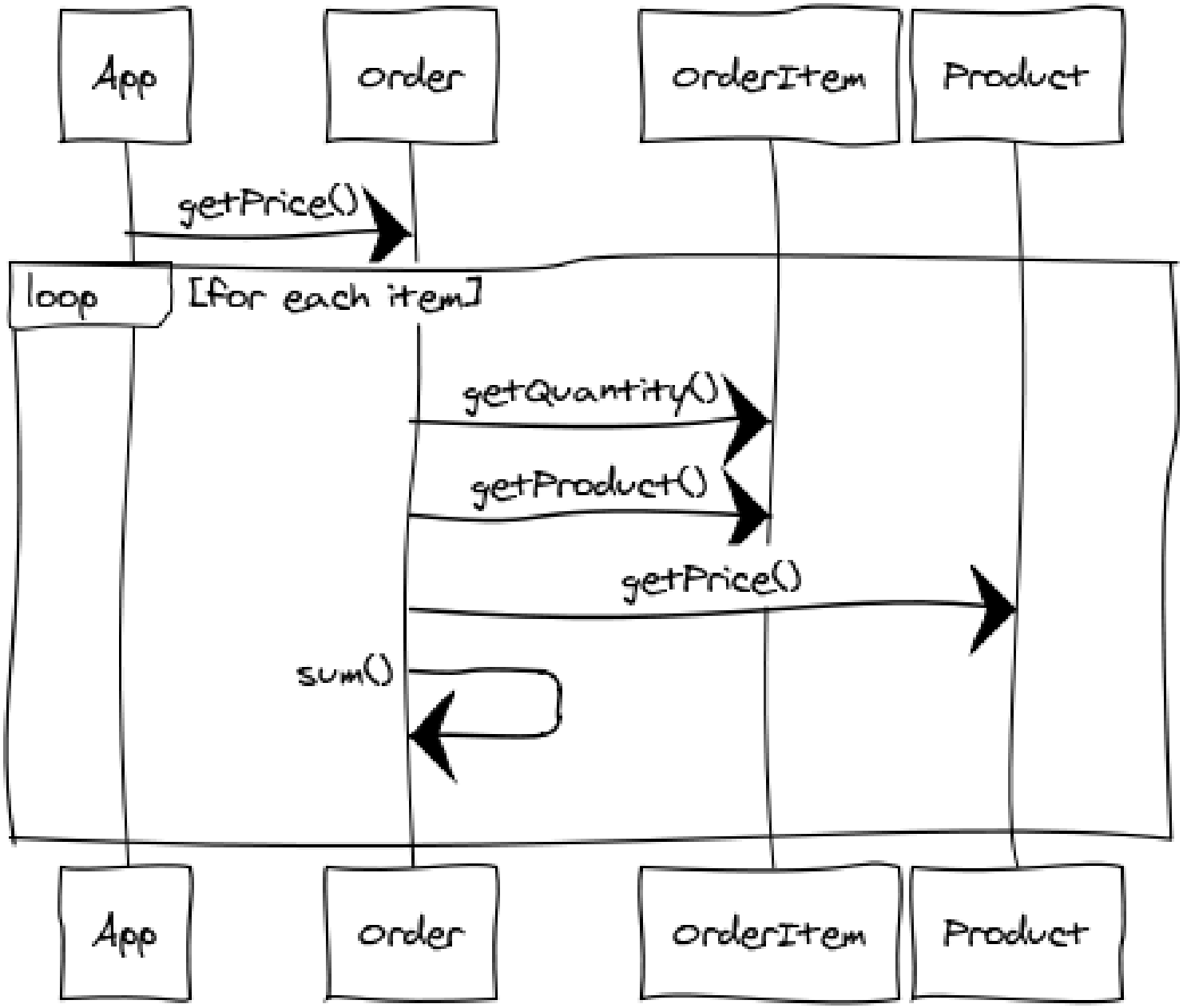
Tell, don't ask



Procedural code gets information then makes decisions.
Object-oriented code tells objects to do things.

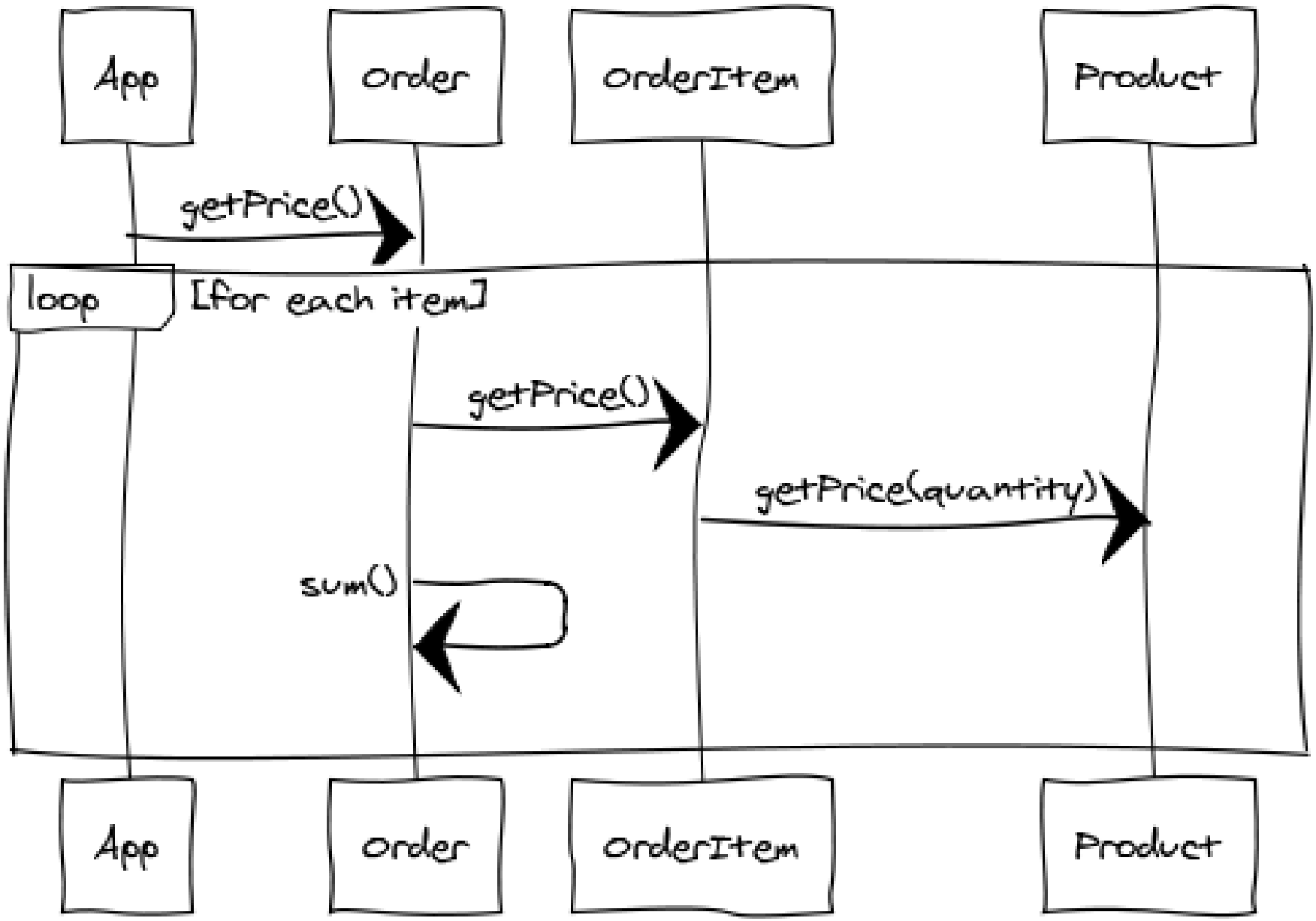
— Alec Sharp

calculating the total for an order (procedural)



www.websequencediagrams.com

calculating the total for an order (object orientation)



www.websequencediagrams.com

Replace "set" sintax with strong names

- setFirstName(string fn)
- setLastName(string ln)

vs.

- changeName(
  string first,
  string last)

- setStatus(STATUS st)
- getStatus()

vs.

- activate()
- deactivate()
- isActive()

DDD

Intention Revealing Interface

Name classes and operations to describe their effect and purpose, without reference to the means by which they do what they promise.

Intention Revealing Interface

```
interface ISapService {
    double getHourlyRate(int sapID);
}
```

vs.

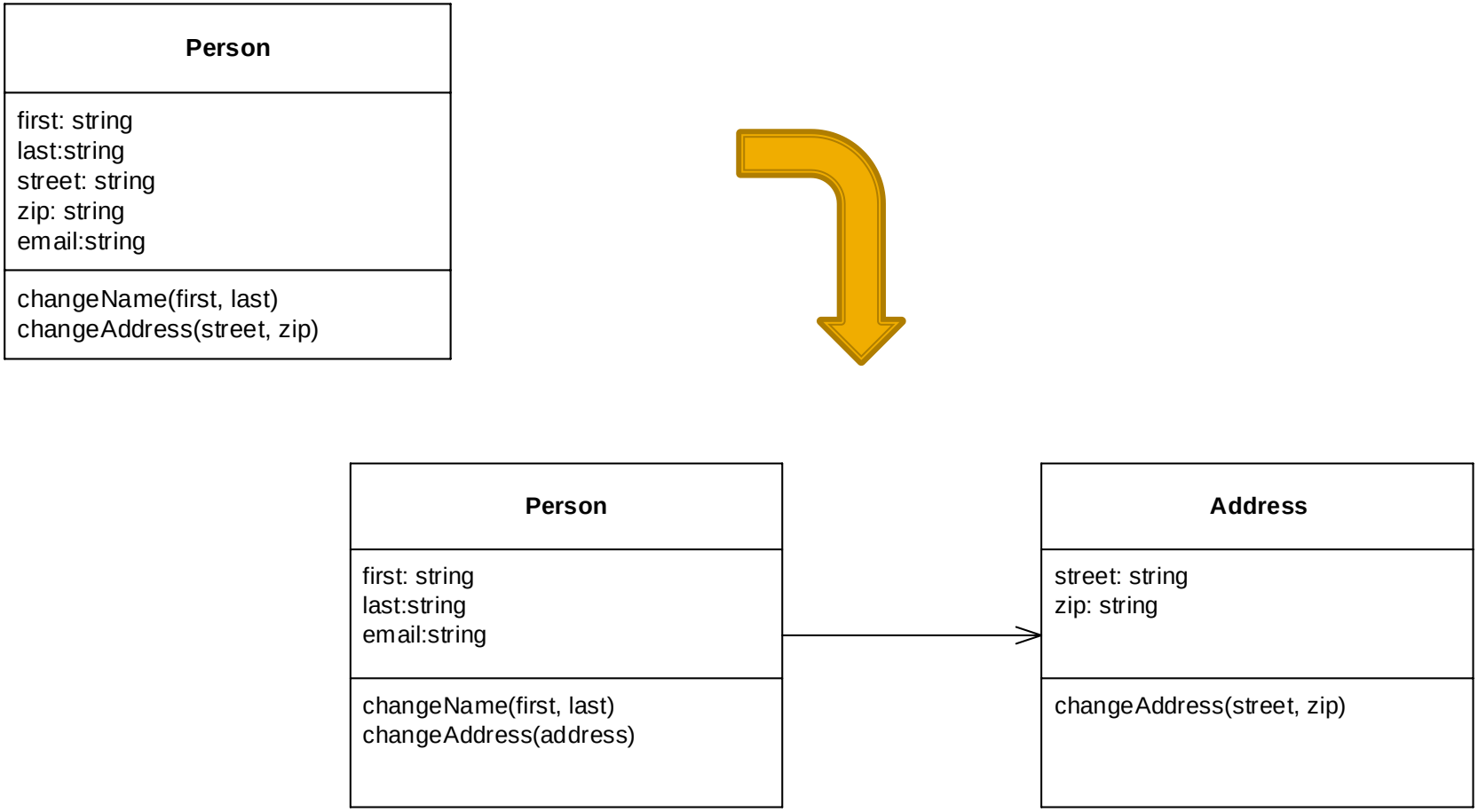
```
interface IPayrollService {
    double getHourlyRate(Employee e);
}
```

Single Responsibility Principle

A class should have only one reason to change.

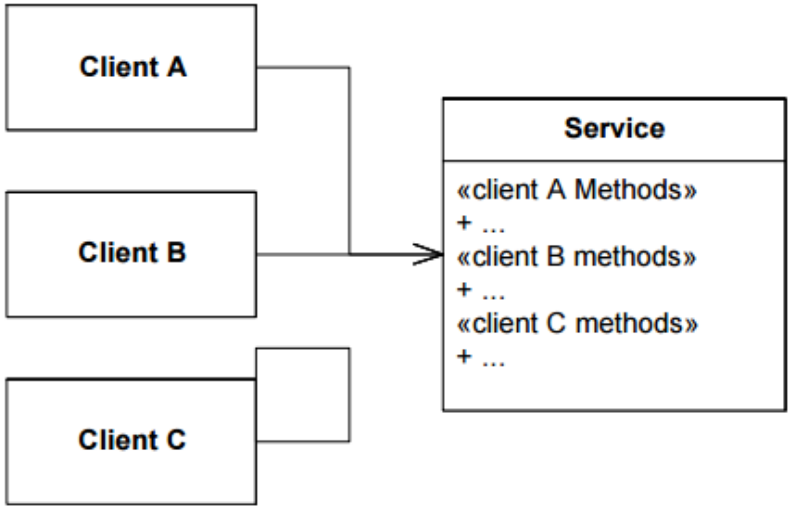
SOLID

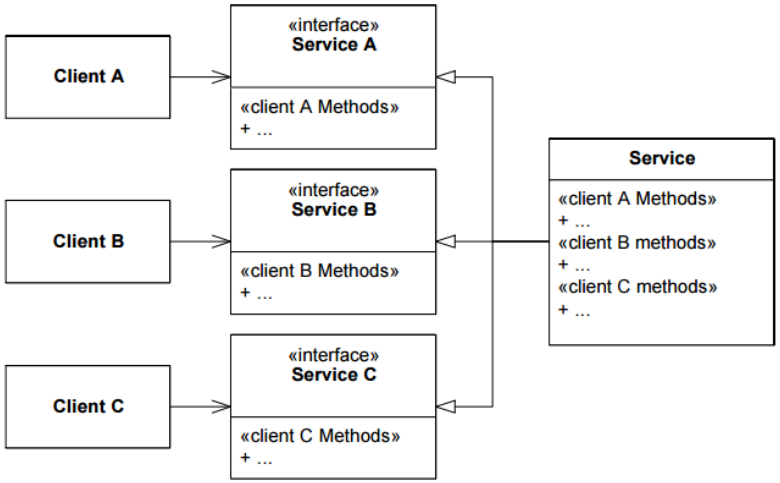
Single Responsibility Principle



Interface Seggregation Principle

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





SOLID

Bibliografia

- Why getters and setters are Evil. Allan Holub. http://www.javaworld.com/article/2073723/core-java/why-getter-and-setter-methods-are-evil.html
- Tell, don't ask. The Pragmatic Programmers. https://pragprog.com/articles/tell-dont-ask
- Design Principles and Design Patterns. Robert Martin. http://www.cvc.uab.es/shared/teach/a21291/temes/object_oriented_design/materials_adicionals/principles_and_patterns.pdf