PROFESSIONAL SOFTWARE ENGINEERING

PSE SWE LE 4 und 5 - Domain Driven Design

SOLID Design Patterns

Dominik Neumann

SOLID

- The SOLID principles tell us how to arrange our functions and data structures into classes, and how those classes should be interconneted.

- The goal of SOLID is the creation of mid-level software structures that:

  - Tolerate change

  - Are easy to understand, and

  - Are the basis of components that can be used in many software systems.

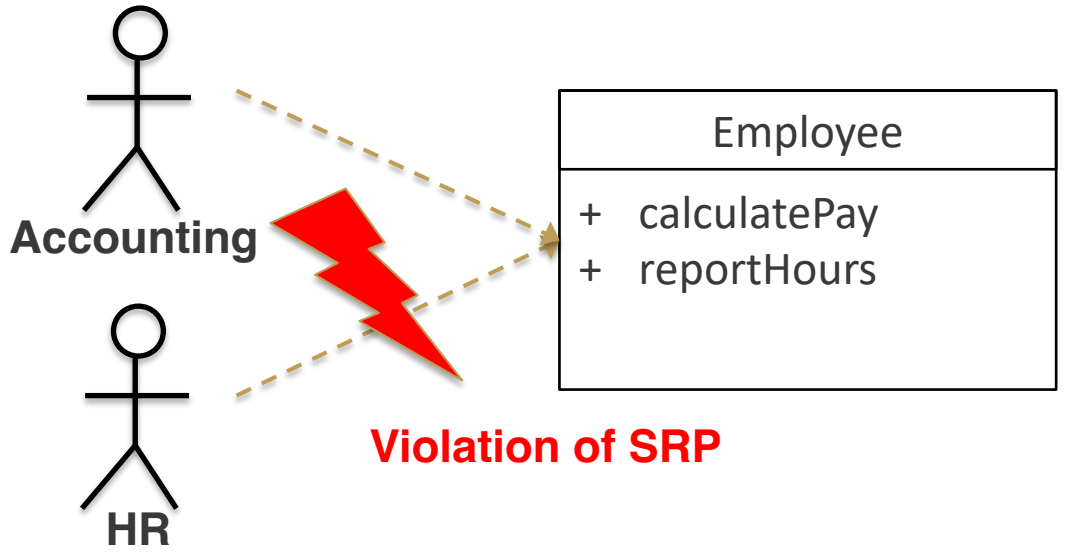
SOLID

- **S**ingle Responsibility Principle
- **O**pen-Closed Principle
- **L**yskov Substitution Principle
- **I**nterface Segregation Principle
- **D**ependency Inversion Principle

SINGLE RESPONSIBILITY PRINCIPLE (SRP)

- Each software module* has one , and only one, reason to change. (original)
- Each software module* should be responsible to one, and only one, actor.
- Cohesion is the force that binds together the code responsible to a single actor.

Symptoms:
- **Accidential Duplication**
- **Merge Conflicts**



* Module could be a source file, a cohesive set of functions and data structures.

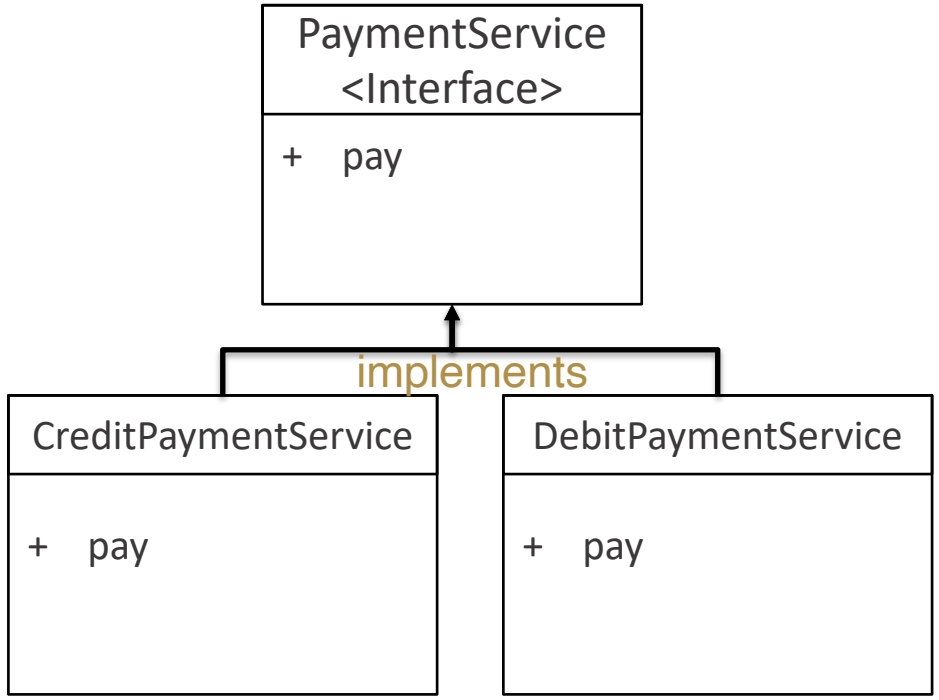
OPEN-CLOSED PRINCIPLE (OCP)

- Each software artifact*.should be open for extention but closed for modification. (original)
- The behavior of the software artifact ought to be extensible, without having to modify it.

```
Class PaymentService:

  def pay(self, order: Order, payment_type: str):

    if payment_type == "debit":
      //do someting in case of debit

    elif payment_type == "credit":
      //do someting in case of credit
```

Violation of OCP

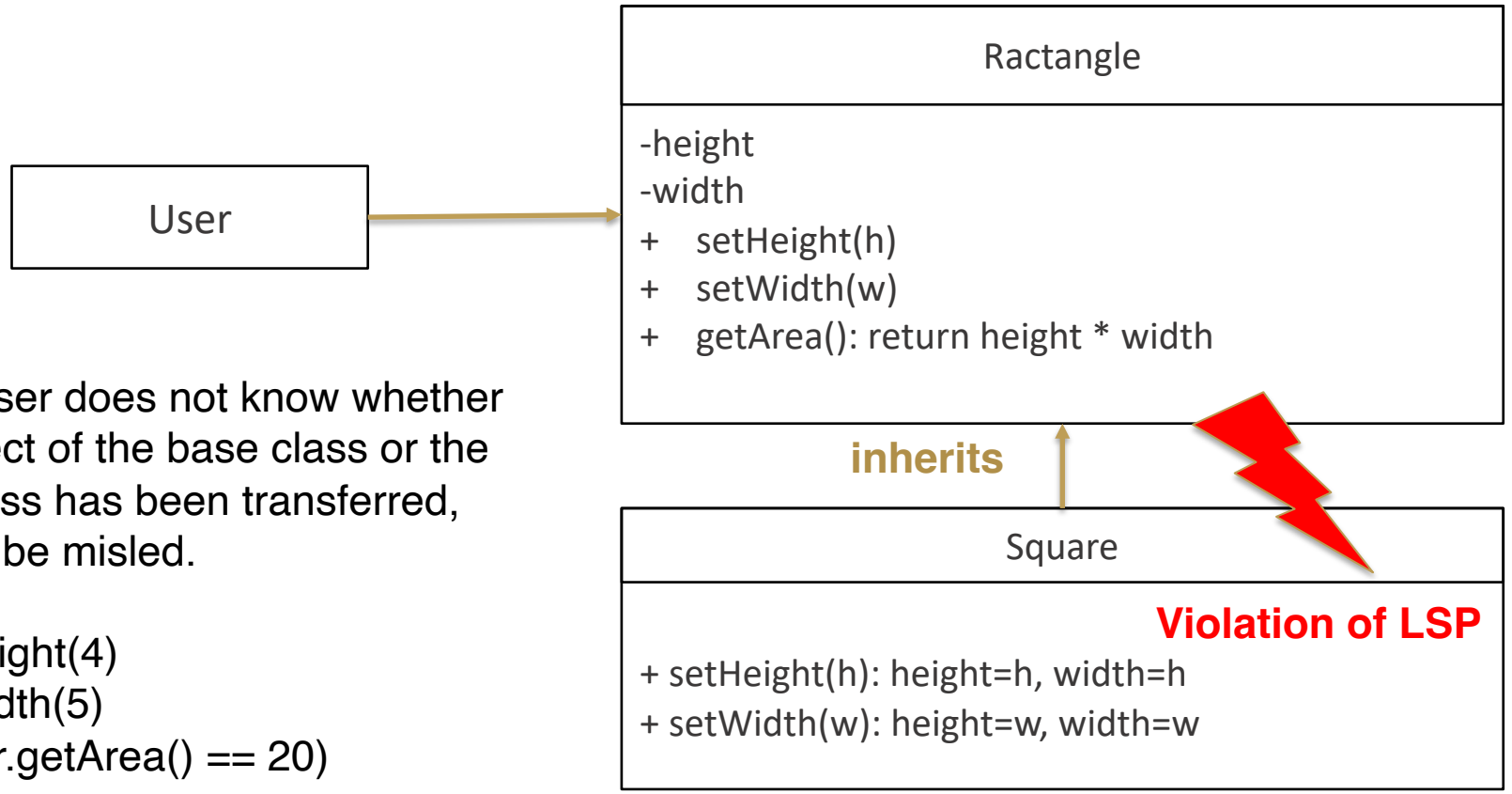
solution



* Artifact could be a source file, a cohesive set of functions and data structures.

LISKOV SUBSTITUTION PRINCIPLE (LSP)

- Subtypes must be substitutable for their base types (Barbara Liskov 1988)



If the user does not know whether an object of the base class or the sub-class has been transferred, he can be misled.

```
r.setHeight(4)
r.setWidth(5)
assert(r.getArea() == 20)
```

A model, viewed in isolation, Cannot be meaningful validated!

The validity of model can only be expressed in terms of its clients.

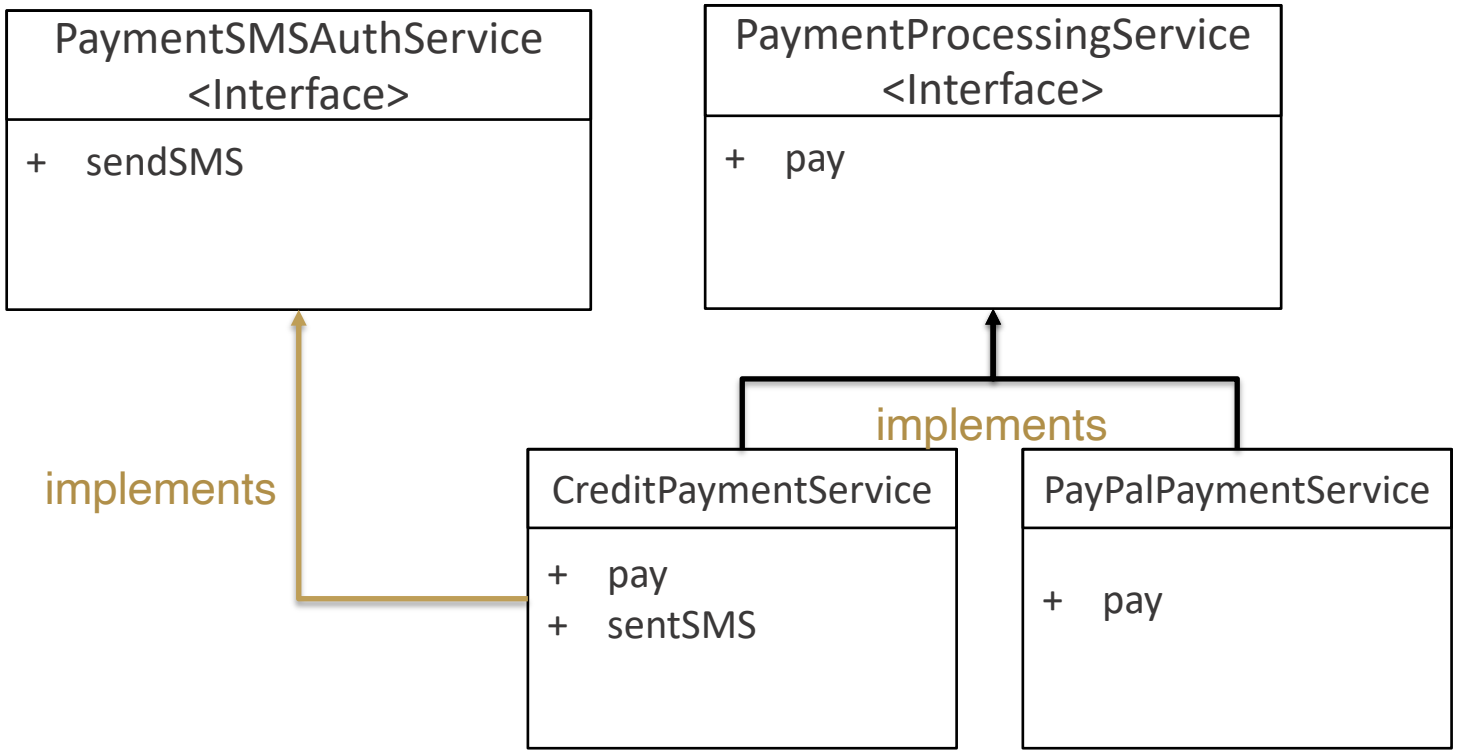
→ Use TDD!

→ In OOD is-a relationship pertains to behavior that that can be assumed and clients depend on!

→ Design by Contract

INTERFACE SEGREGATION PRINCIPLE (ISP)

- Clients should not be forced to depend on methods that they do not use.
- Classes that have fat interfaces are classes that are not cohesive
- If the fat interface has cohesive groups of methods provided to different clients, then the interface should be broken down in a group of interfaces

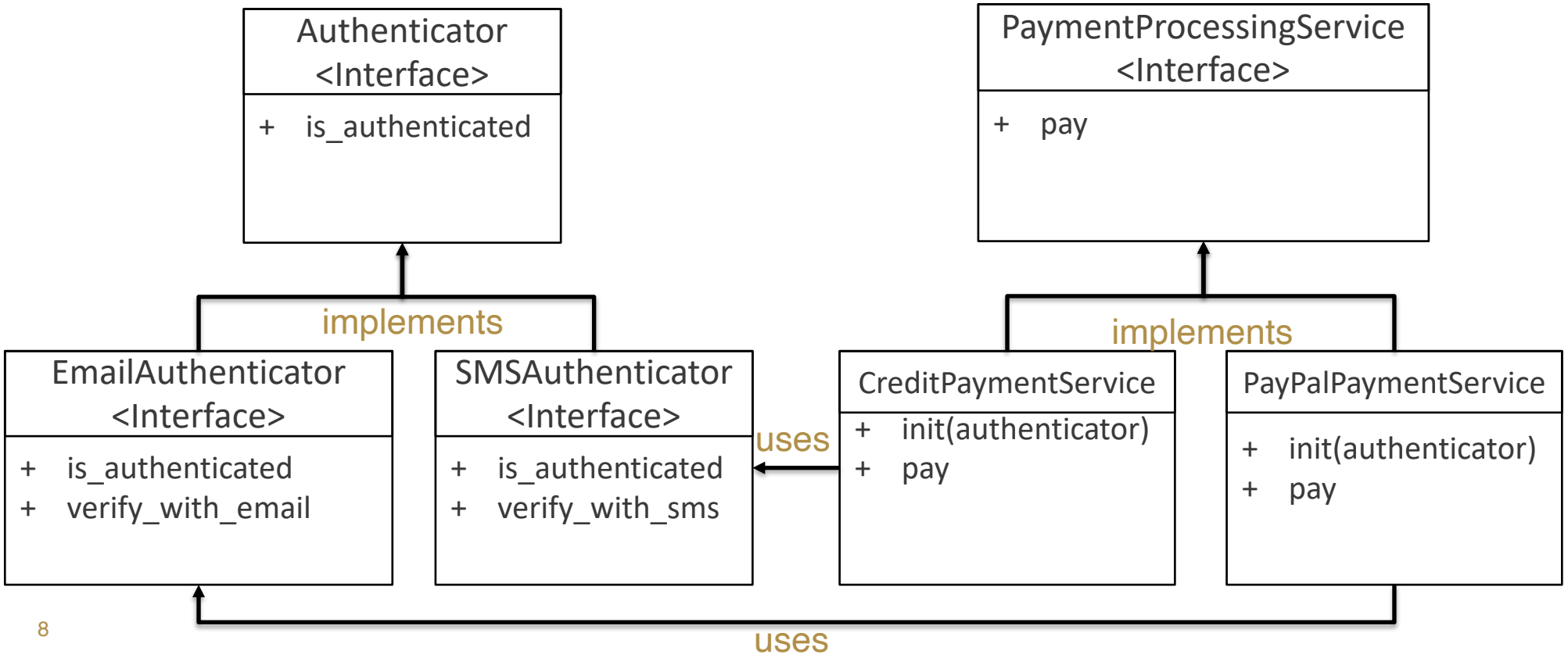


* Artifact could be a source file, a cohesive set of functions and data structures.

DEPENDENCY INVERSION PRINCIPLE (DIP)

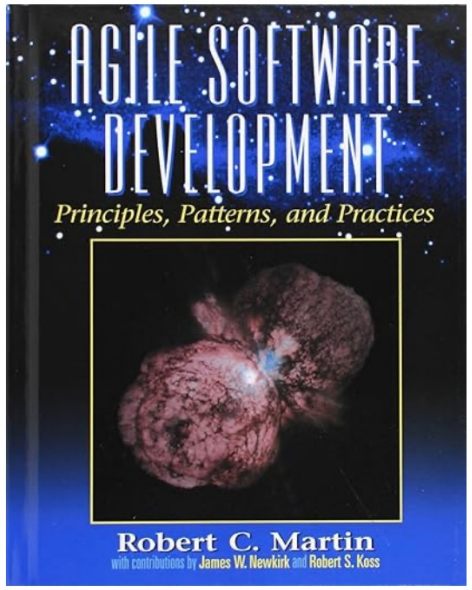
- The Dependency Inversion Principle states that high-level modules should not depend on low-level modules, but both should depend on abstractions.

- This means that you should not have to change other sections of your code when you change the implementation of a class.

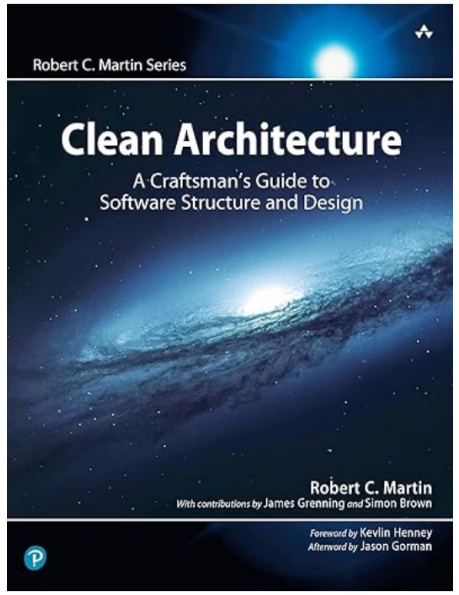


FURTHER READING

SOLID



2002 – Robert Martin
Agile Software Development, Principles, Patterns, and Practices.
Pearson



2018 – Robert Martin
Clean Architecture
A Craftman's Guide to Software Structure and Design
Pearson