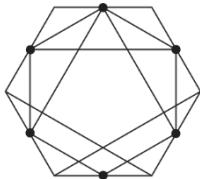


# IT DESIGNERS GRUPPE

IT DESIGNER GRUPPE  
[HTTPS://IT-DESIGNERS-GRUPPE.DE](https://it-designers-gruppe.de)



https://it-designers-gruppe.de

IT DESIGNERS GRUPPE

Unternehmensgruppe Individuelle Softwareentwicklung Embedded SW für Automotive Application Lifecycle Management Forschung und Lehre Schulungen Karriere

Karriere starten.  
Software entwickeln.

# DIE IT-DESIGNERS GRUPPE

IT DESIGNERS GRUPPE + STZ SOFTWARETECHNIK

## Kundenindividuelle Softwareentwicklung

Sie benötigen individuelle Softwarelösungen? Wir entwickeln individuelle Softwarelösungen oder unterstützen Sie bei Ihren Entwicklungsprojekten. Mobile Anwendungen, Websysteme, Informationssysteme oder Automatisierungstechnik, in Java, C#, oder C++.

## Ingenieur-Dienstleistungen für Automotive Software

Automobil-Entwicklung bedeutet nicht erst seit heute auch Software-Entwicklung. Ob modellbasiertes Entwickeln, Embedded Programmierung oder umfangreiches Testen – wir sind seit 20 Jahren verlässlicher Partner für Hersteller und Zulieferer in der Region Stuttgart.

## Application Lifecycle Management

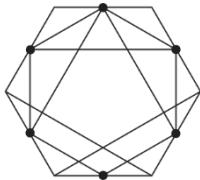
Gute Softwareentwicklung braucht gute Werkzeuge. Wir sind Ihr Partner rund um die Planung, Beratung, Anpassung und den Betrieb der führenden ALM Werkzeuge: Polarion, Doors, agosense.symphony, PTC Integrity, HP Quality Center.

IT-Designers Gruppe

Entennest 2, 73730 Esslingen  
E-Mail: office(at)it-designers-gruppe(dot)de

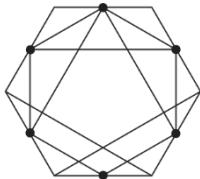
Fon +49 711 / 30 51 11 50  
Fax +49 711 / 30 51 11 12

Home Kontakt Anfahrt Impressum Datenschutz

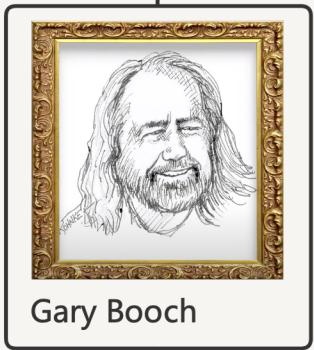


# SINGLE RESPONSIBILITY





## Clean Code

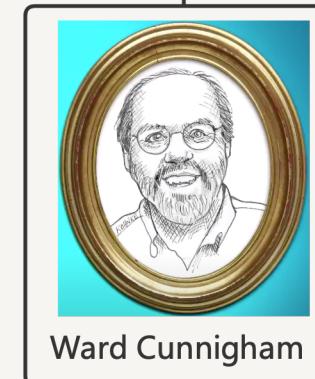


Simple & Direct

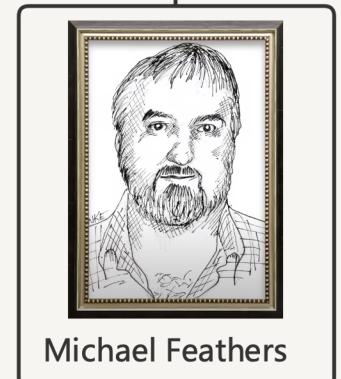
Reads as well-written prose



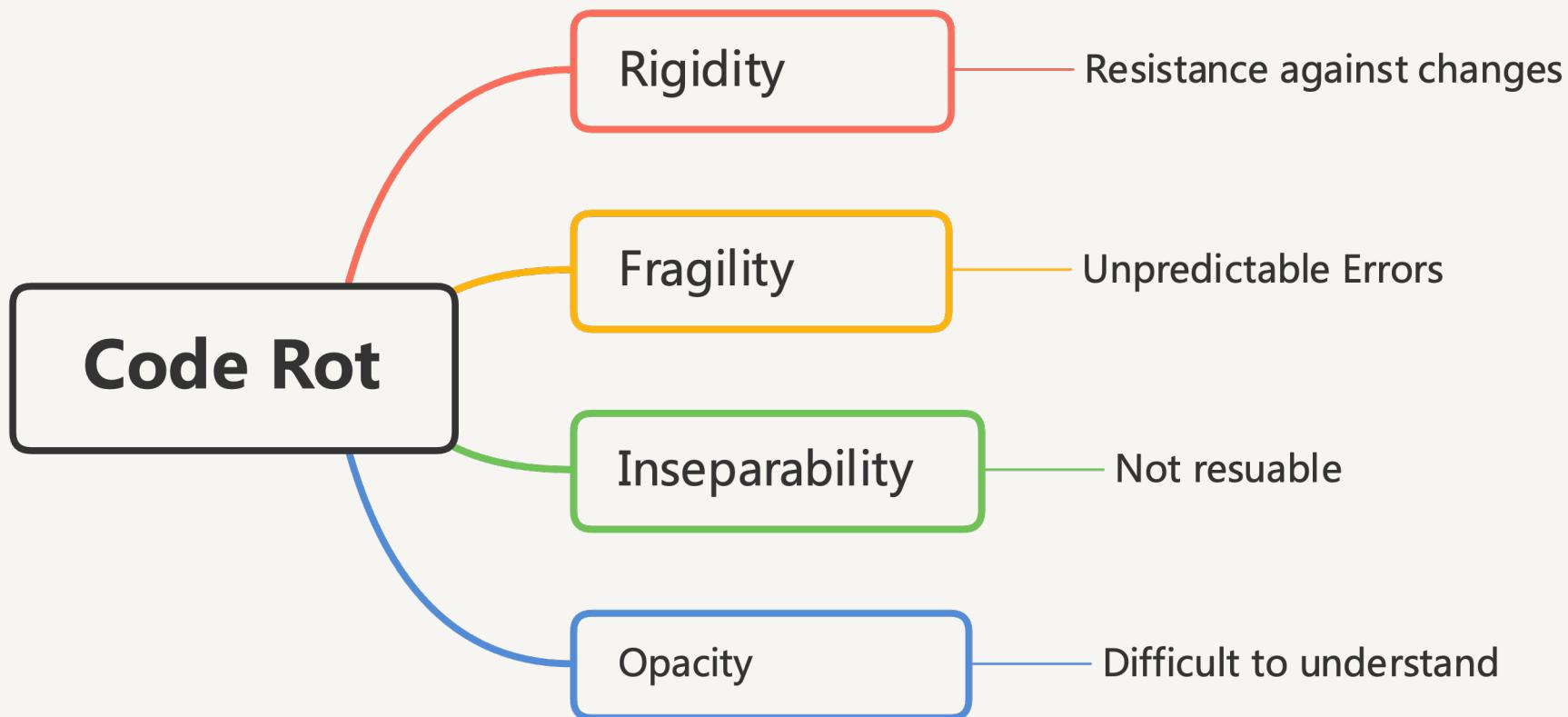
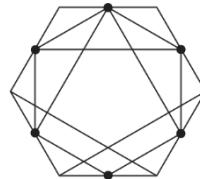
Elegant & Efficient. Should do one thing

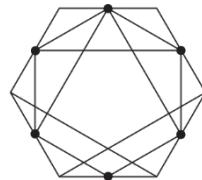


Every Routine is pretty  
much what you expected



Written by someone who cares





**SOLID**

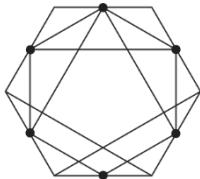
**Single Responsibility Principle**

**Open Close Principle**

**Liskov Substitution Principle**

**Interface Segregation Principle**

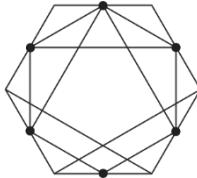
**Dependency Inversion Principle**



## SINGLE RESPONSIBILITY PRINCIPLE



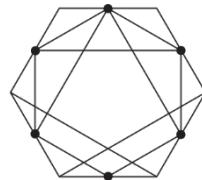
- For each class
- There should only be
- a single reason to change



## SINGLE RESPONSIBILITY PRINCIPLE

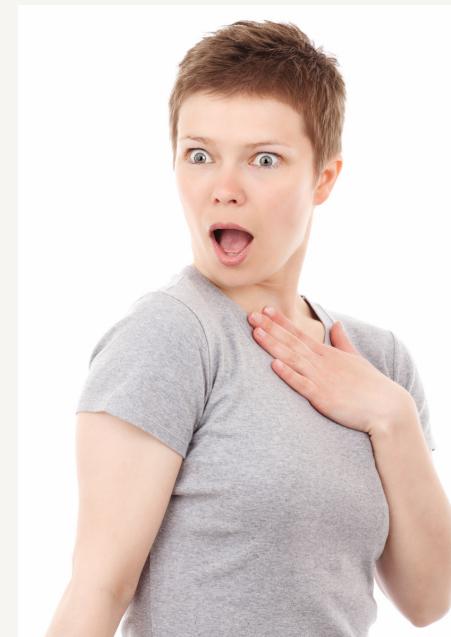


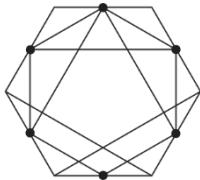
- The name is not quite correct!
- SRP does not state that each class
- may only have a single responsibility!



# SINGLE RESPONSIBILITY VIOLATION

<b>Book</b>
-title: string
-author: string
-page: int
+print(): void
+save(): void





Marketing  
Department

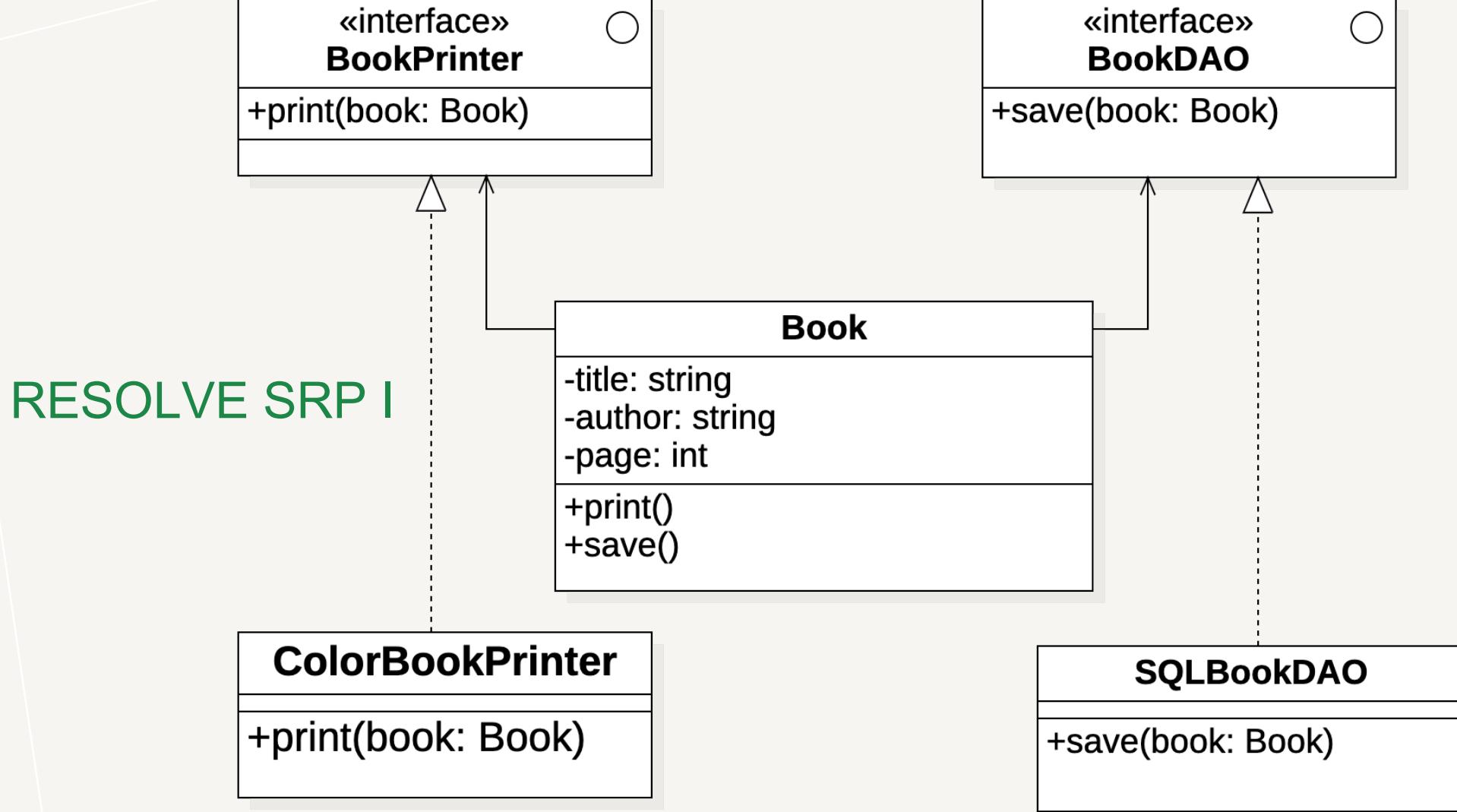
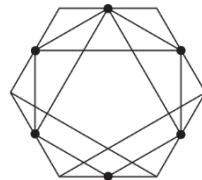
Database  
Administrators

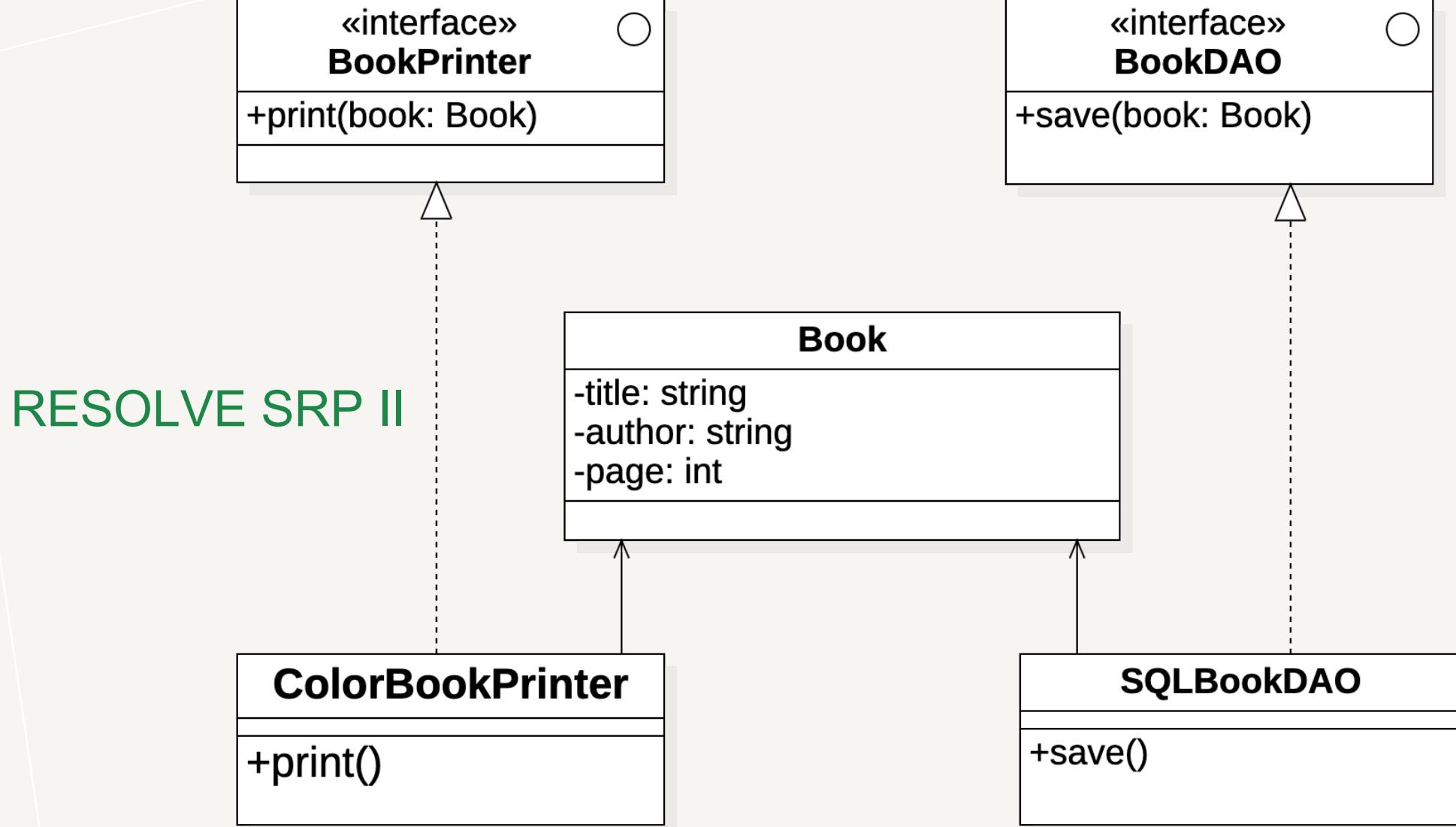
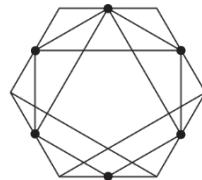


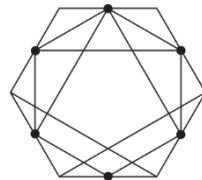
class Book

+ print

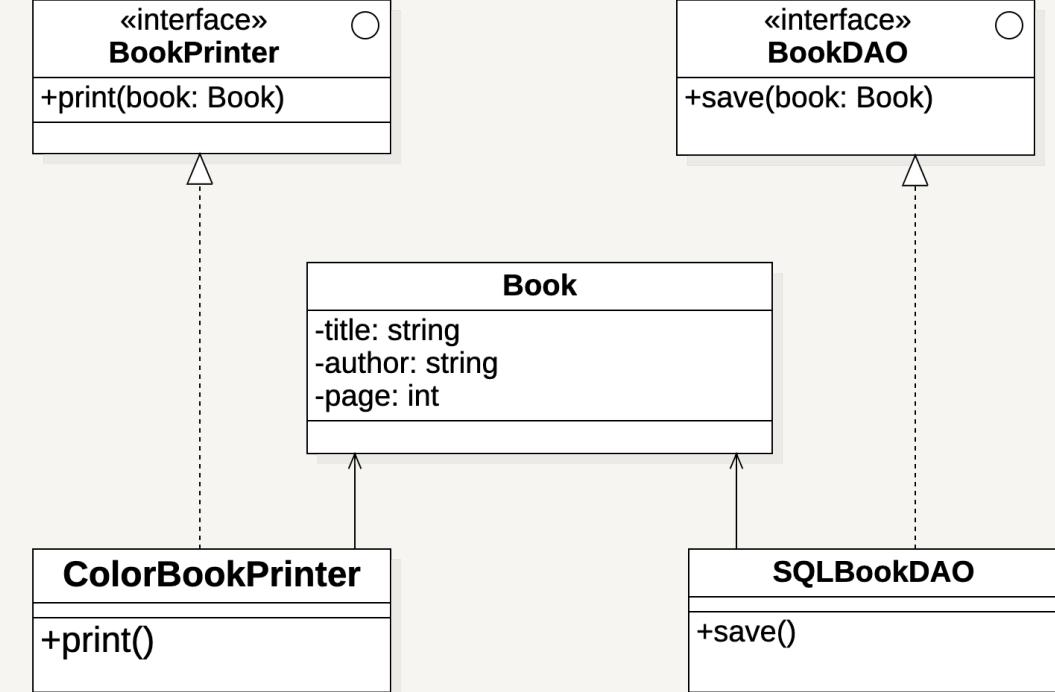
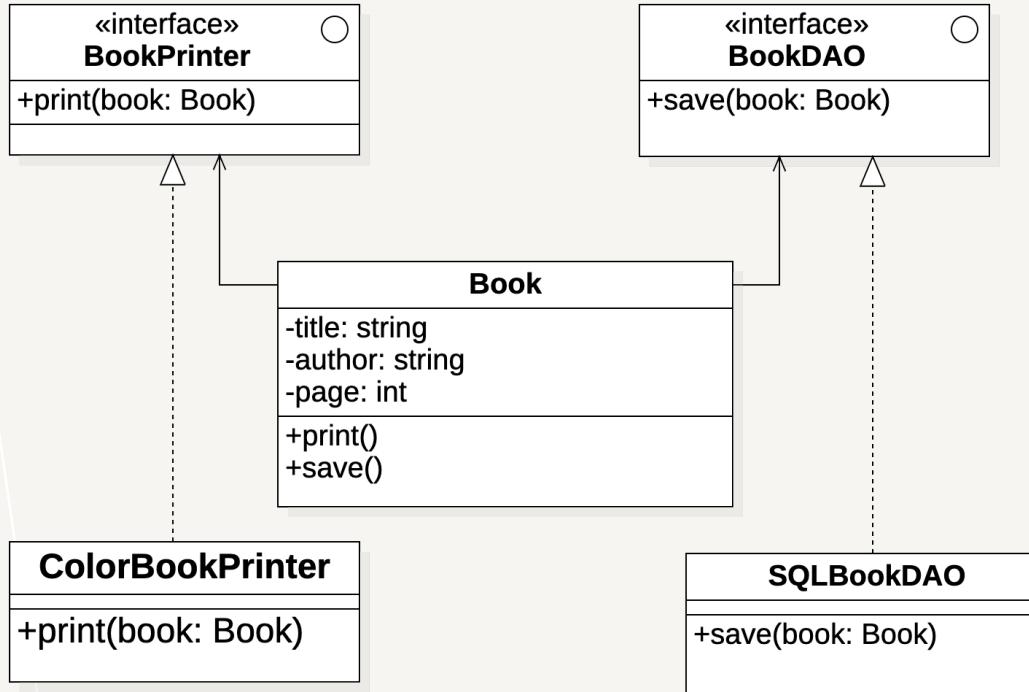
+ save

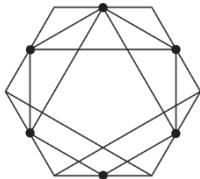






## RESOLVE THE SRP VIOLATION I VS. II

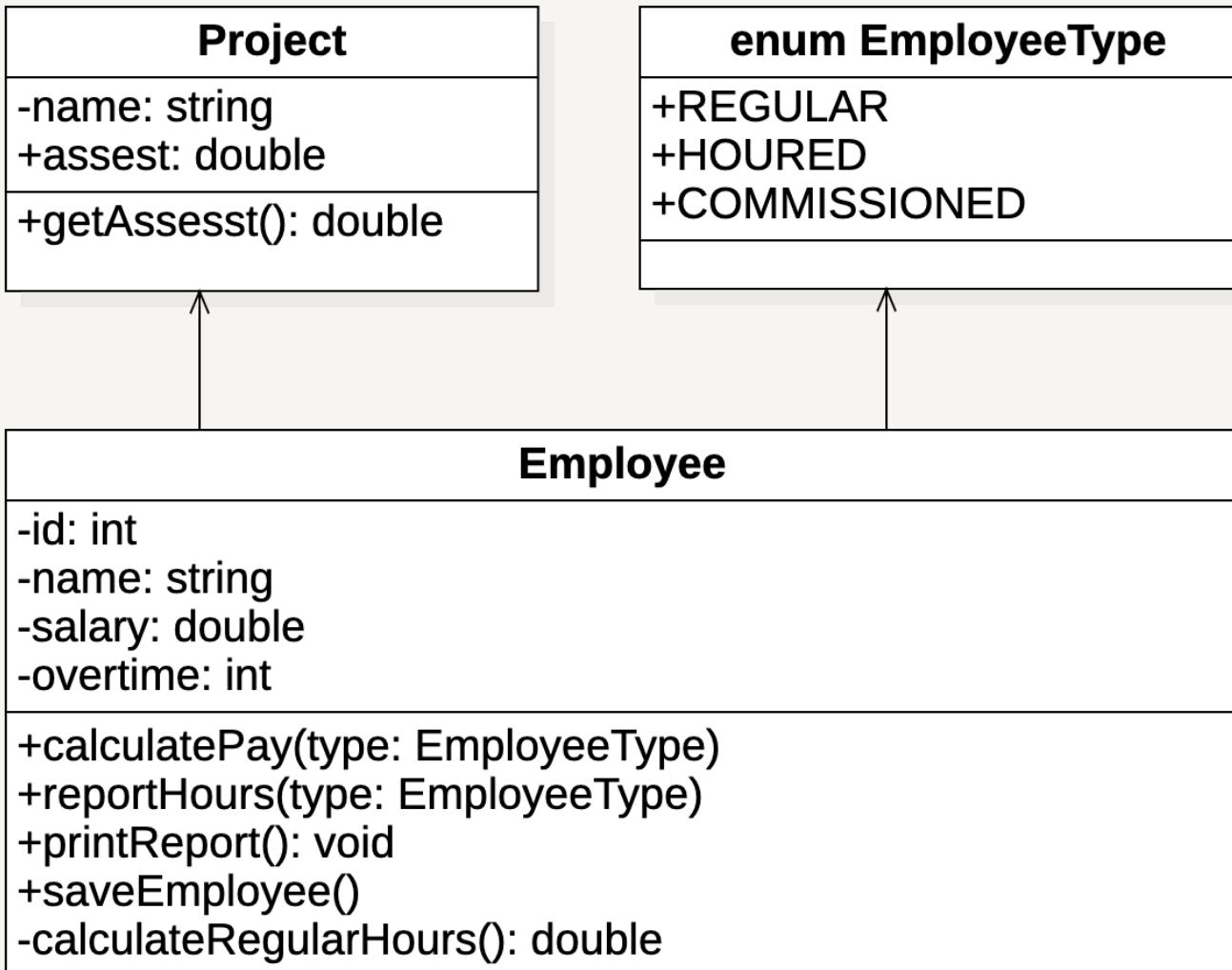
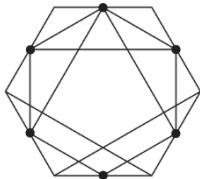


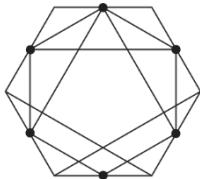


## SRP => MONSTER KLASSEN VERMEIDEN!

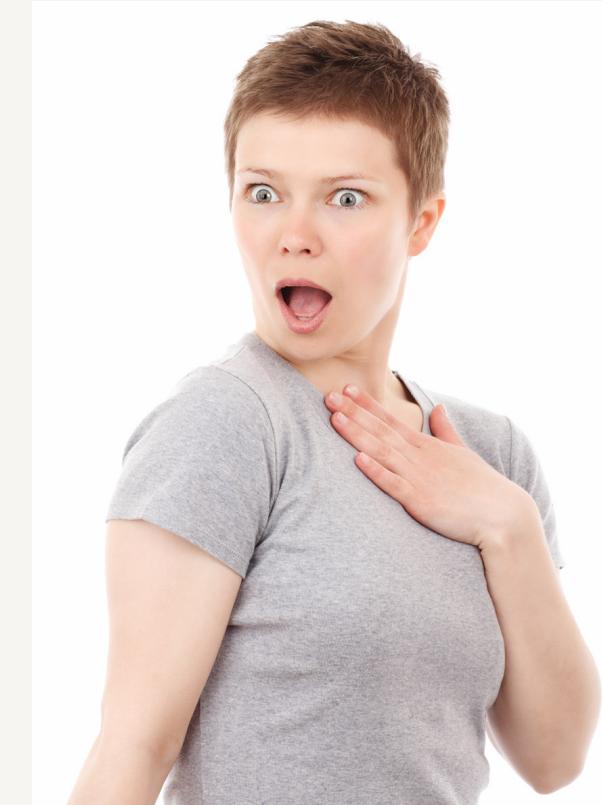
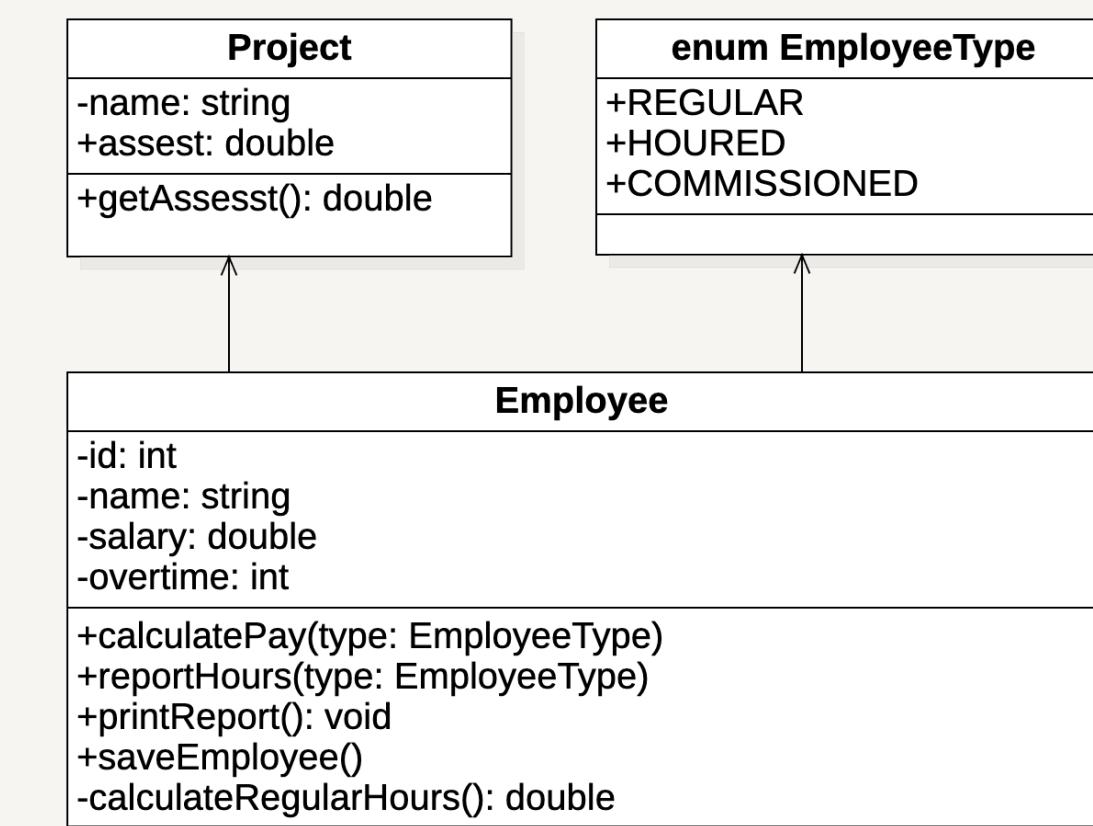


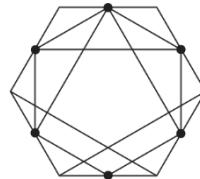
- No classes anymore
- Which know too much!
- Like the Swiss Knife!





## WHAT IS WRONG?

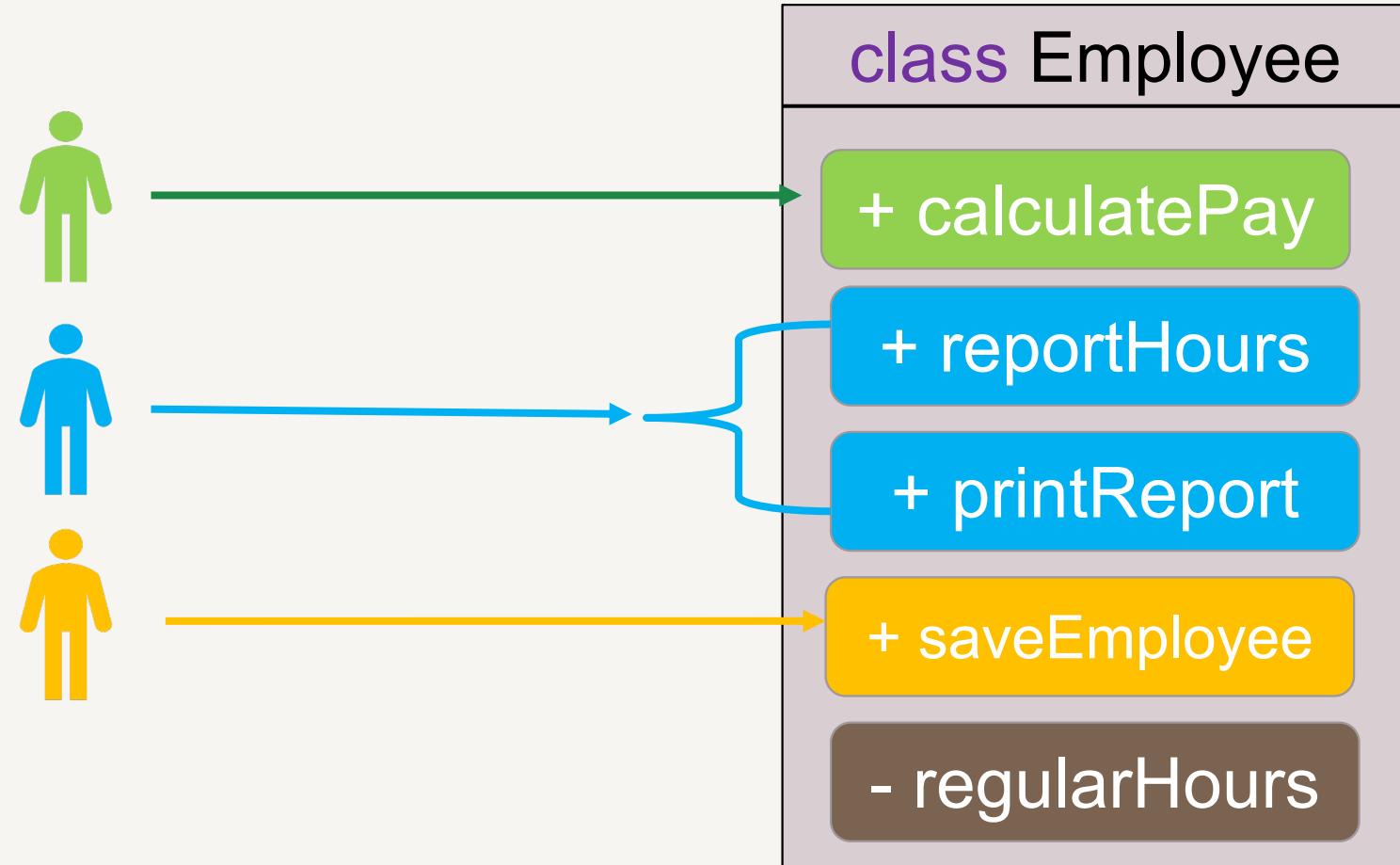


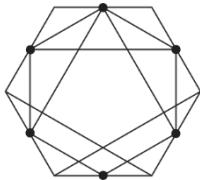


Accounting  
Department

Human  
Resources  
Department

Database  
Administrators





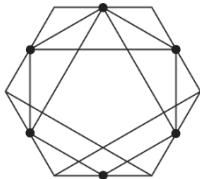
## class Employee

+ calculatePay

+ reportHours

- regularHours

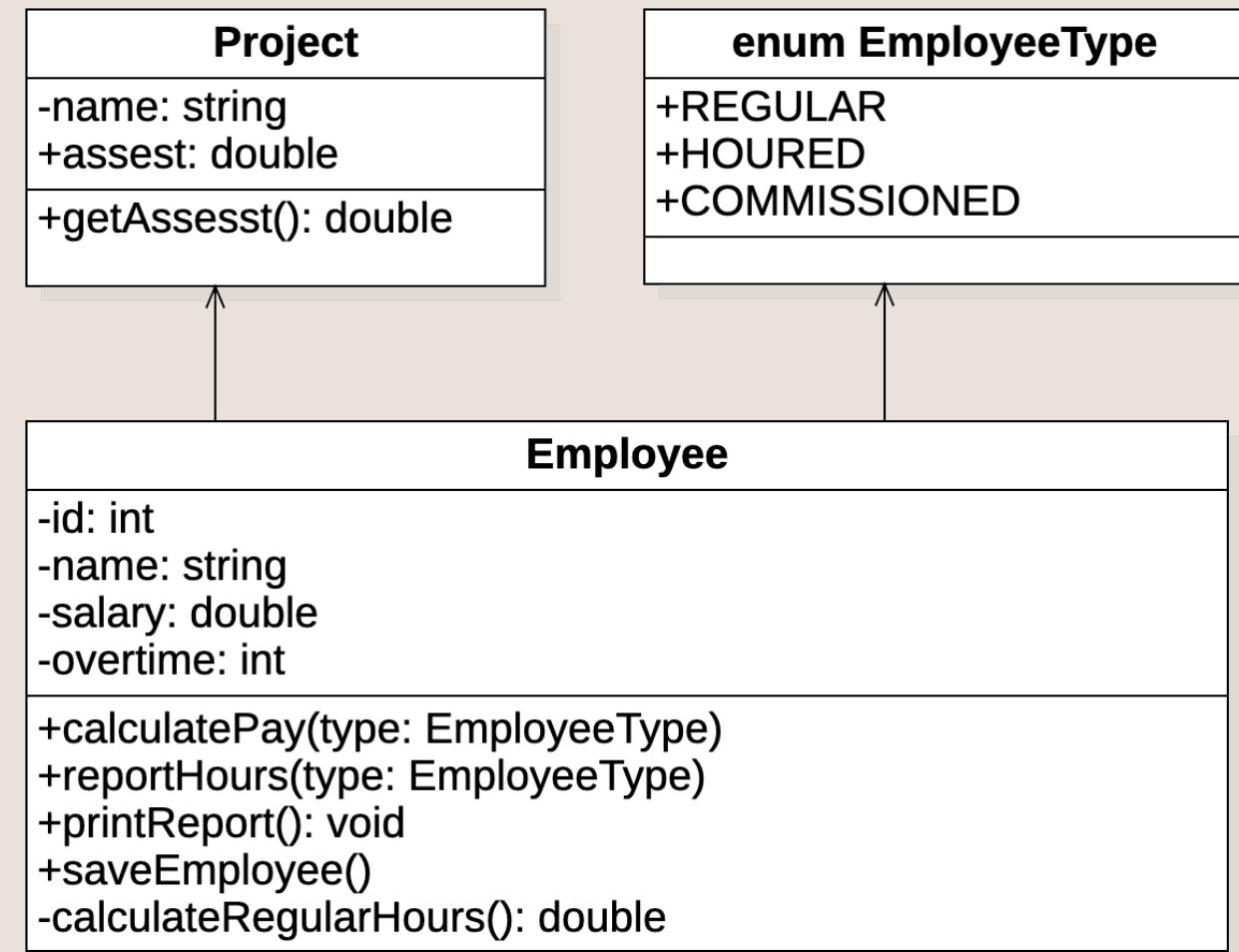
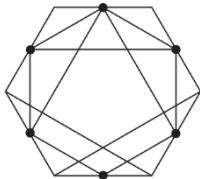
+ saveEmployee



# SINGLE RESPONSIBILITY PRINCIPLE WORKSHOP

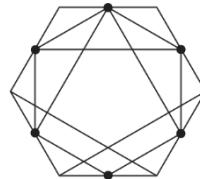


- Master the Workshop!



## SRPBESTEHENDES DESIGN

- Design breaks SRP !
- Re-Design it to pass SRP
- Implements the new Design ?



# DISCOUSIONÖSUNGEN UND DISKUSSIONEN





VIELEN DANK FÜR IHRE AUFMERKSAMKEIT.