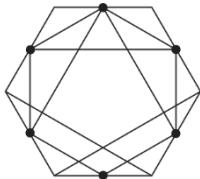
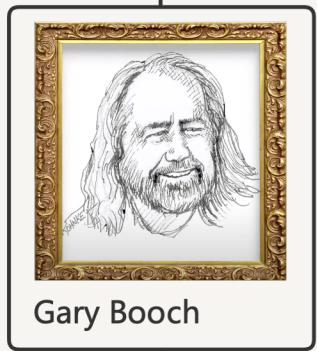


OPEN CLOSE PRINCIPLE



Clean Code

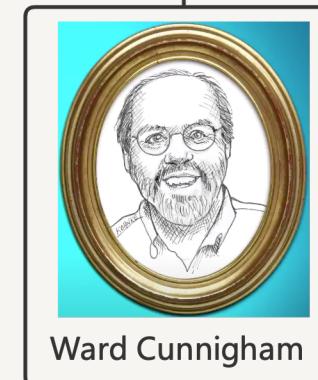


Simple & Direct

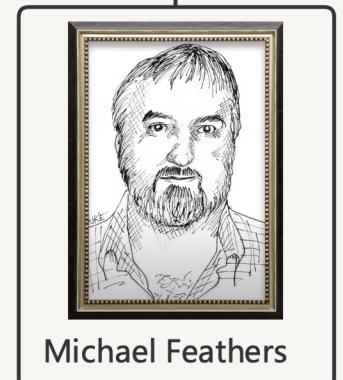
Reads as well-written prose



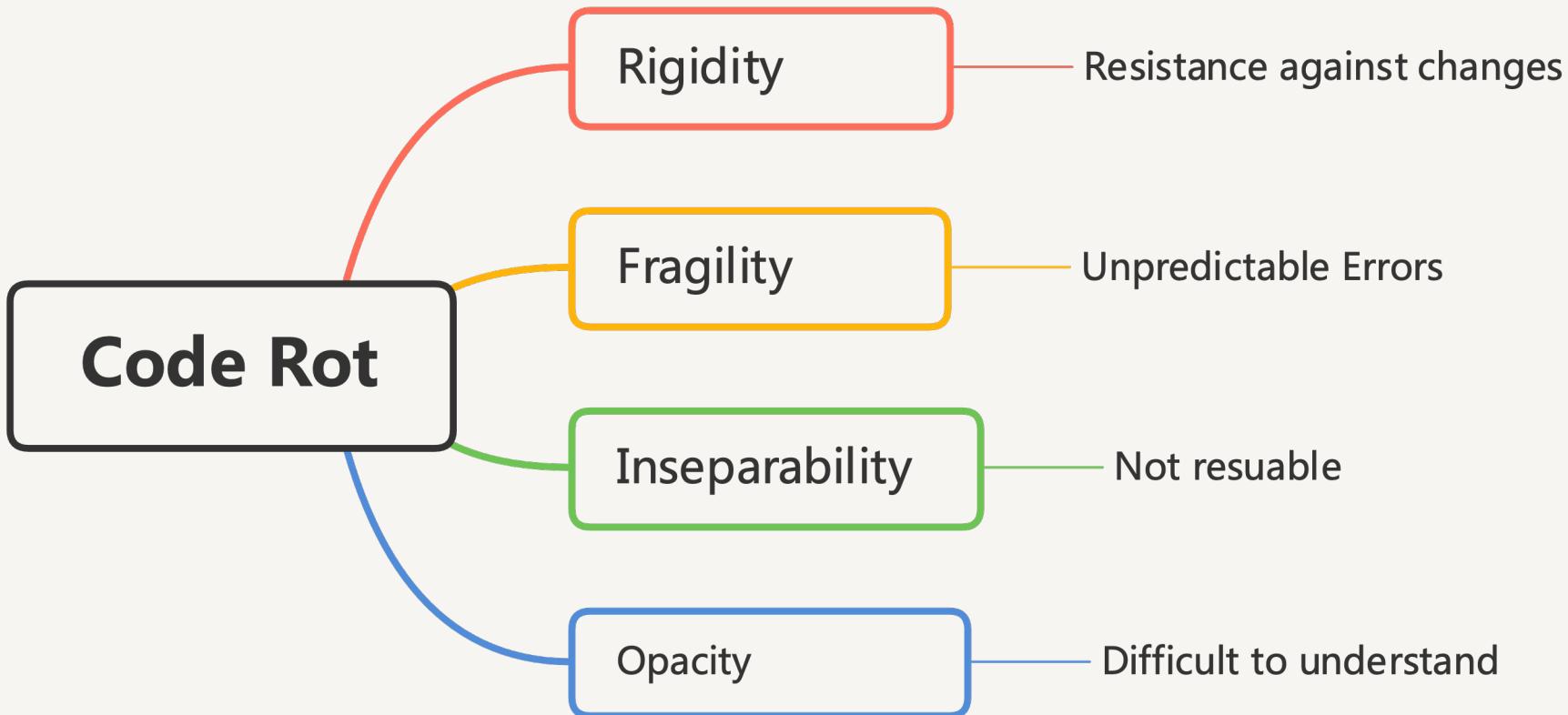
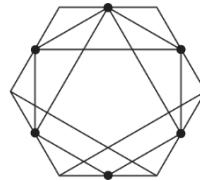
Elegant & Efficient. Should do one thing

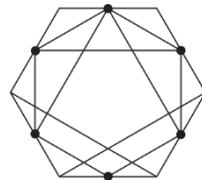


Every Routine is pretty
much what you expected



Written by someone who cares





SOLID

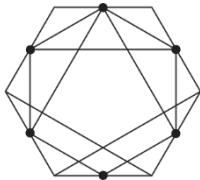
Single Responsibility Principle

Open Close Principle

Liskov Substitution Principle

Interface Segregation Principle

Dependency Inversion Principle



OPEN CLOSE PRINCIPLE

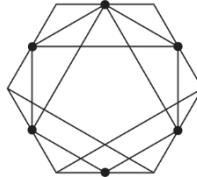


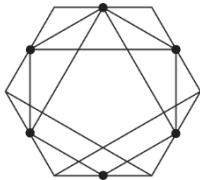
- Software artifacts (classes, modules, function,etc)
- Should be open for extnsions
- But closed for change



OPEN CLOSE PRINCIPLE

- More simply: It should be possible to
- modify the behavior of objects
- or introduce new kinds of objects
- Without touching the existing code

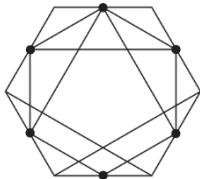




OPEN CLOSE PRINCIPLE



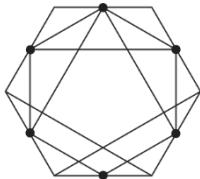
- More concretely: Implementation
- against interface
- Not concrete classes



OPEN CLOSE PRINCIPLE



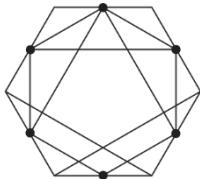
- More concretely:
- Use of method overriding (polymorphy)
- Instead of if or switch statements



OPEN CLOSE VIOLATION I

```
switch (each.getMovie ().getPriceCode ()) {  
    case Movie.REGULAR:  
        thisAmount += 2;  
        if (each.getDaysRented () > 2)  
            thisAmount += (each.getDaysRented () - 2) * 1.5;  
        break;  
    case Movie.NEW_RELEASE:  
        thisAmount += each.getDaysRented () * 3;  
        break;  
    case Movie.CHILDRENS:  
        thisAmount += 1.5;  
        if (each.getDaysRented () > 3)  
            thisAmount += (each.getDaysRented () - 3) * 1.5;  
        break;  
}
```





OPEN CLOSE VIOLATION II

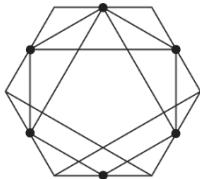
```
if (figure instanceof Triangle)
    drawTriangle();

else if (figure instanceof Square)
    drawSquare();

else
    Console.WriteLine("Error Message");

}
```





OPEN CLOSE VIOLATION III

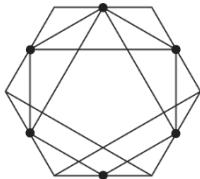
```
if (loadBalancer.algorithm == Algorithm.ROUND_ROBIN)
    executeRoundRobinAlgorithm();

else if (loadBalancer.algorithm == Algorithm.WEIGHTED.ROUND_ROBIN)
    executeWeightedRoundRobinAlgorithm();

else
    throwException();

}
```



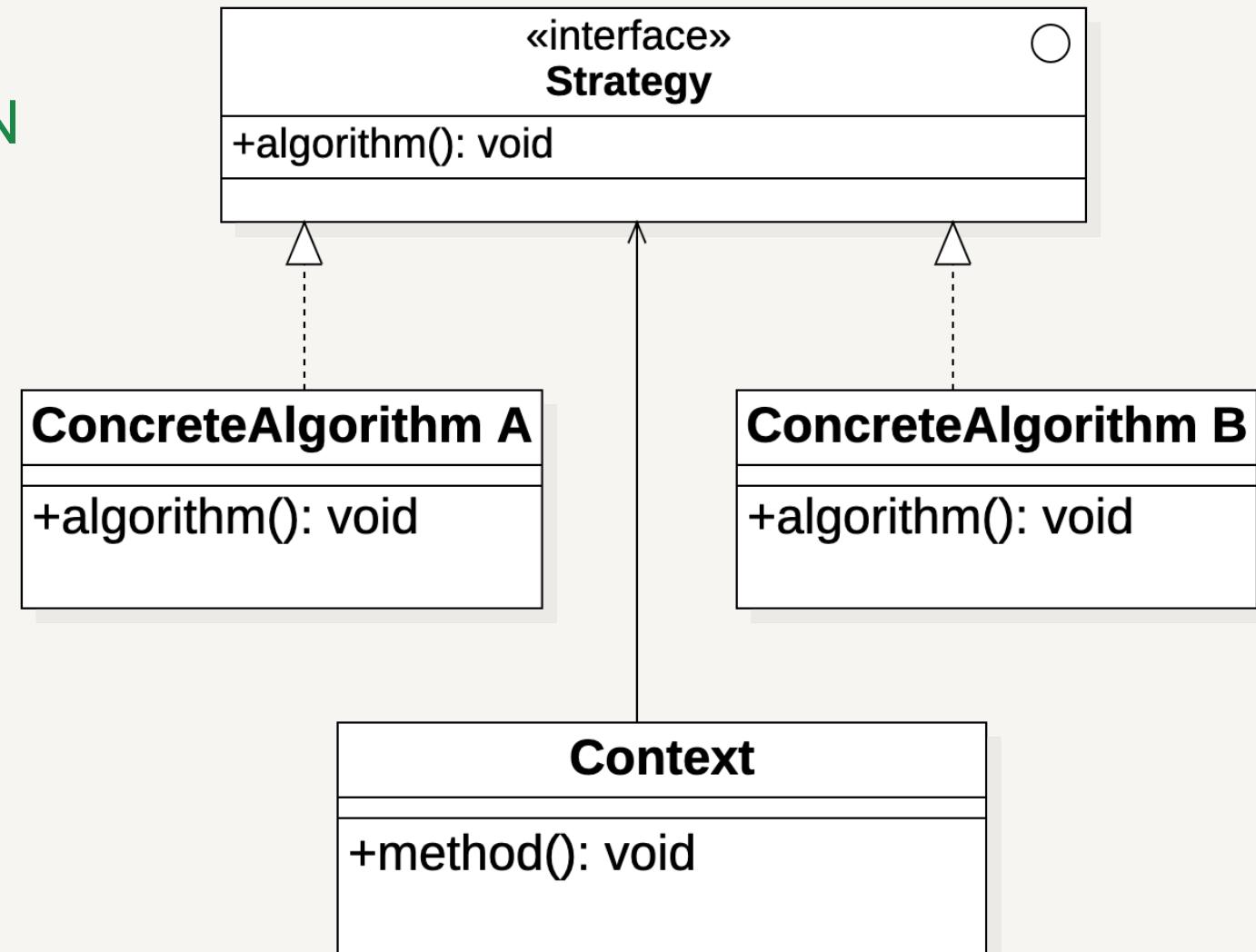
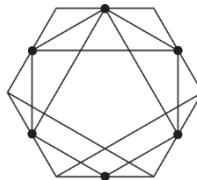


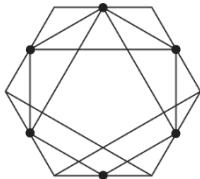
COOLER KUNDE!



- New Requirements:
- Least Connections Algorithm
- Weighted Least Connections

STRATEGY PATTERN



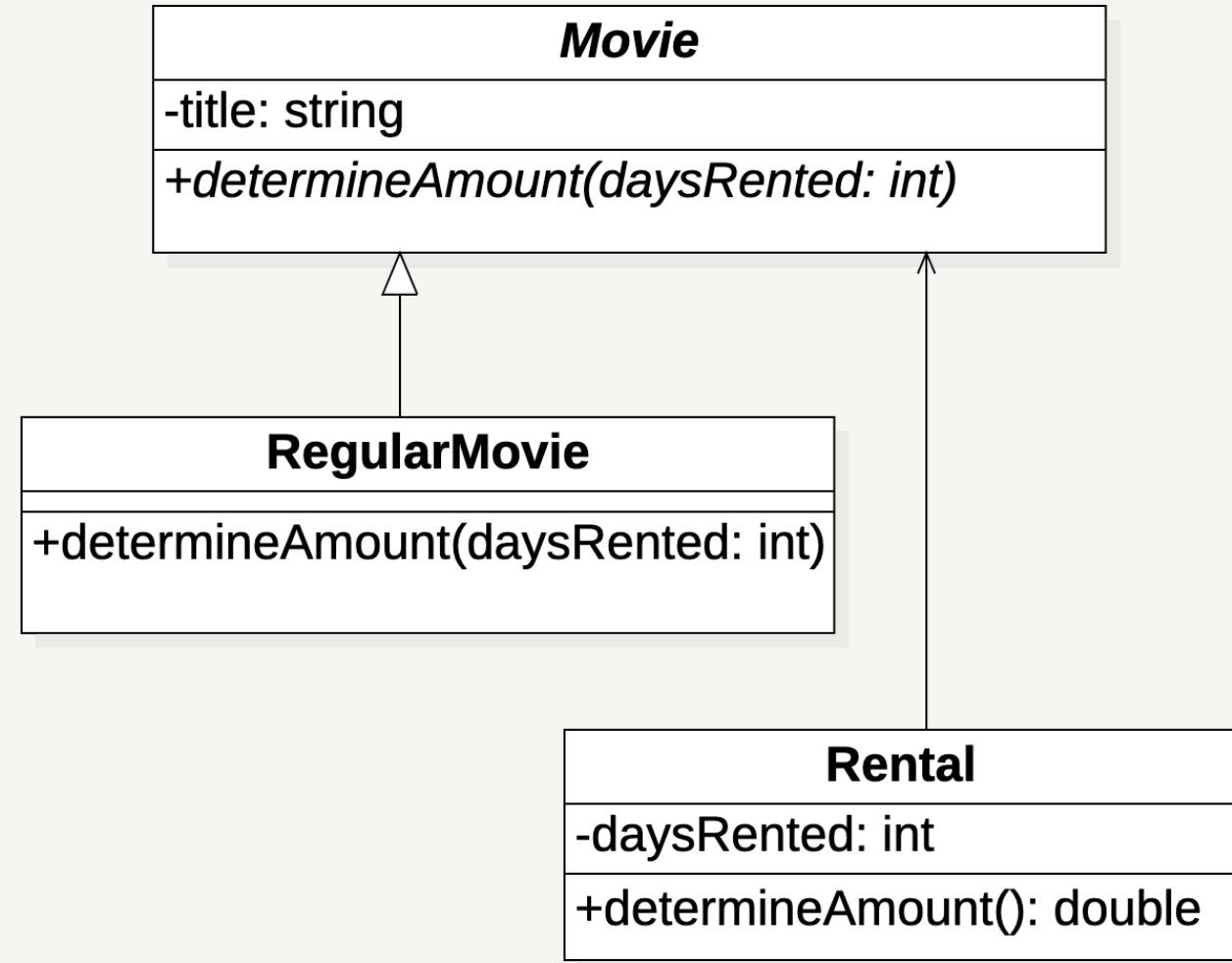
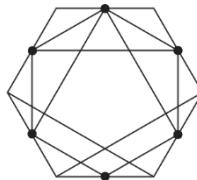


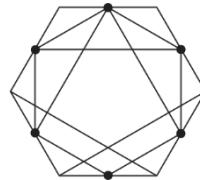
OPEN CLOSE VIOLATION I

```
switch (each.getMovie ().getPriceCode ()) {  
    case Movie.REGULAR:  
        thisAmount += 2;  
        if (each.getDaysRented () > 2)  
            thisAmount += (each.getDaysRented () - 2) * 1.5;  
        break;  
    case Movie.NEW_RELEASE:  
        thisAmount += each.getDaysRented () * 3;  
        break;  
    case Movie.CHILDRENS:  
        thisAmount += 1.5;  
        if (each.getDaysRented () > 3)  
            thisAmount += (each.getDaysRented () - 3) * 1.5;  
        break;  
}
```



STRATEGY PATTERN

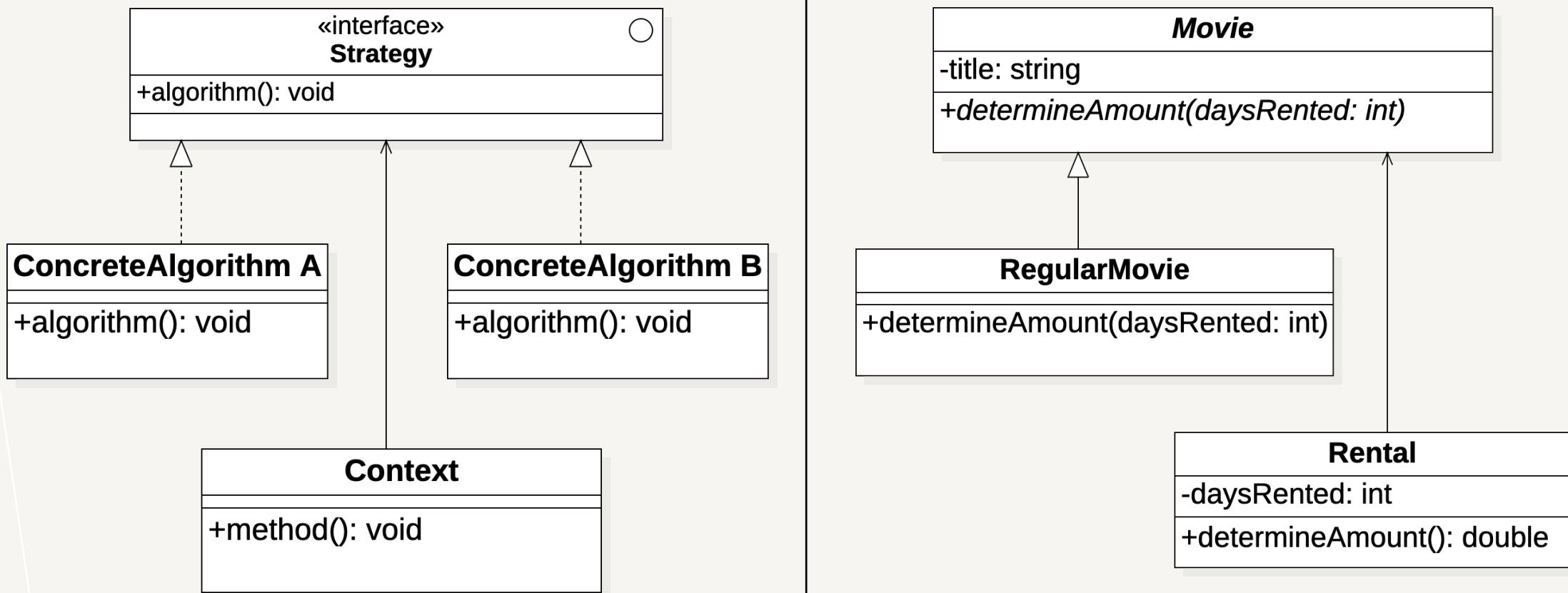
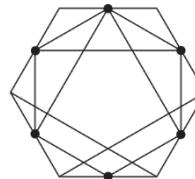


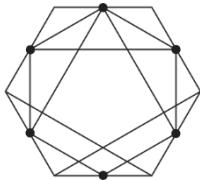


OPEN CLOSE PRINCIPLE WORKSHOP

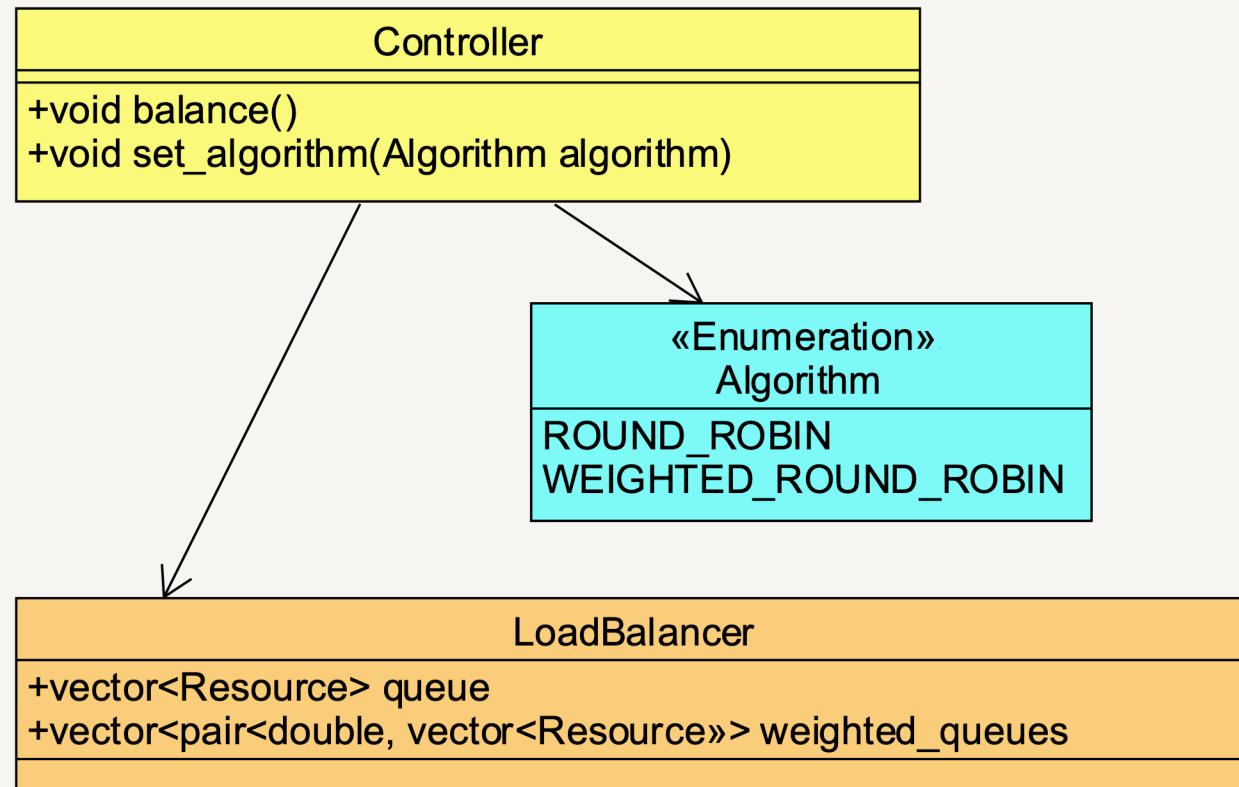


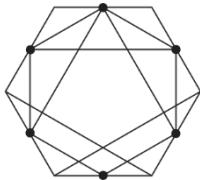
• WORKSHOP



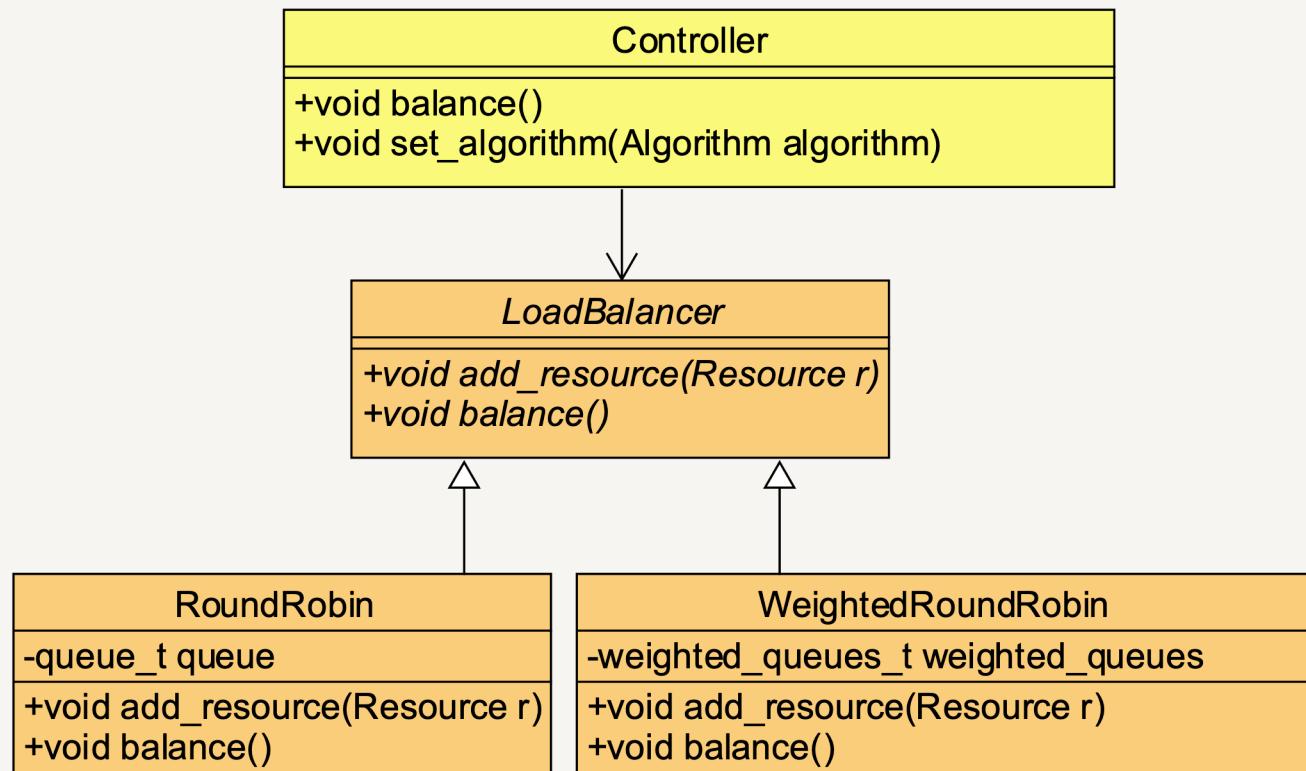


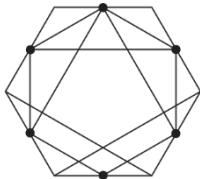
LOAD BALANCER I



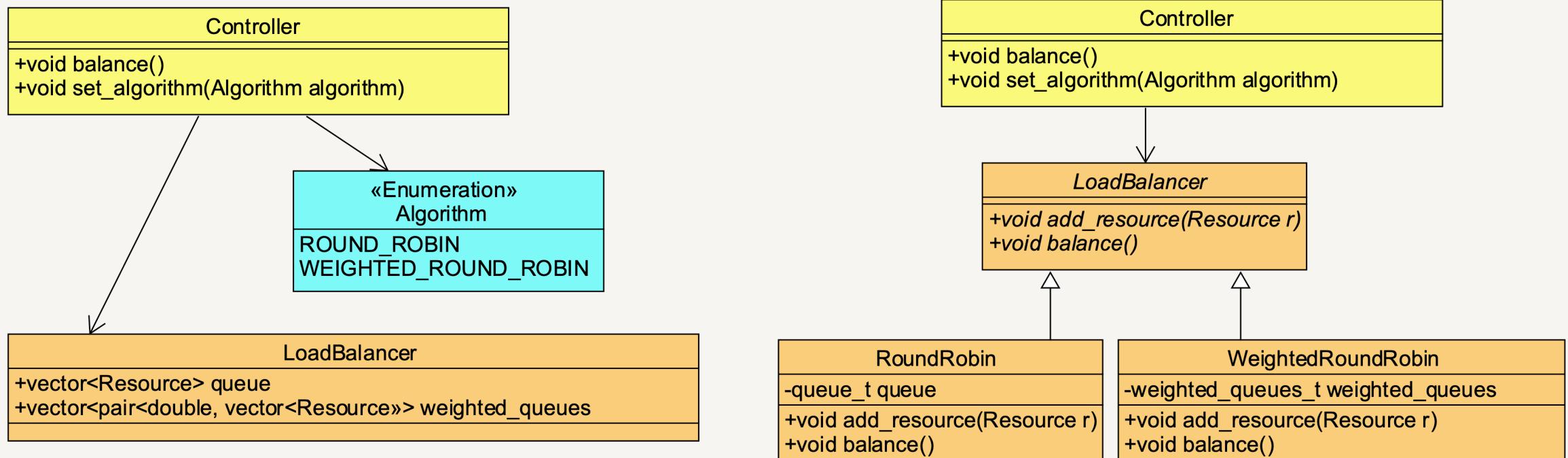


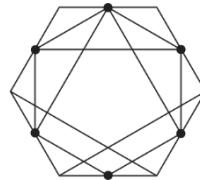
LOAD BALANCER II





LOAD BALANCER I VS. II

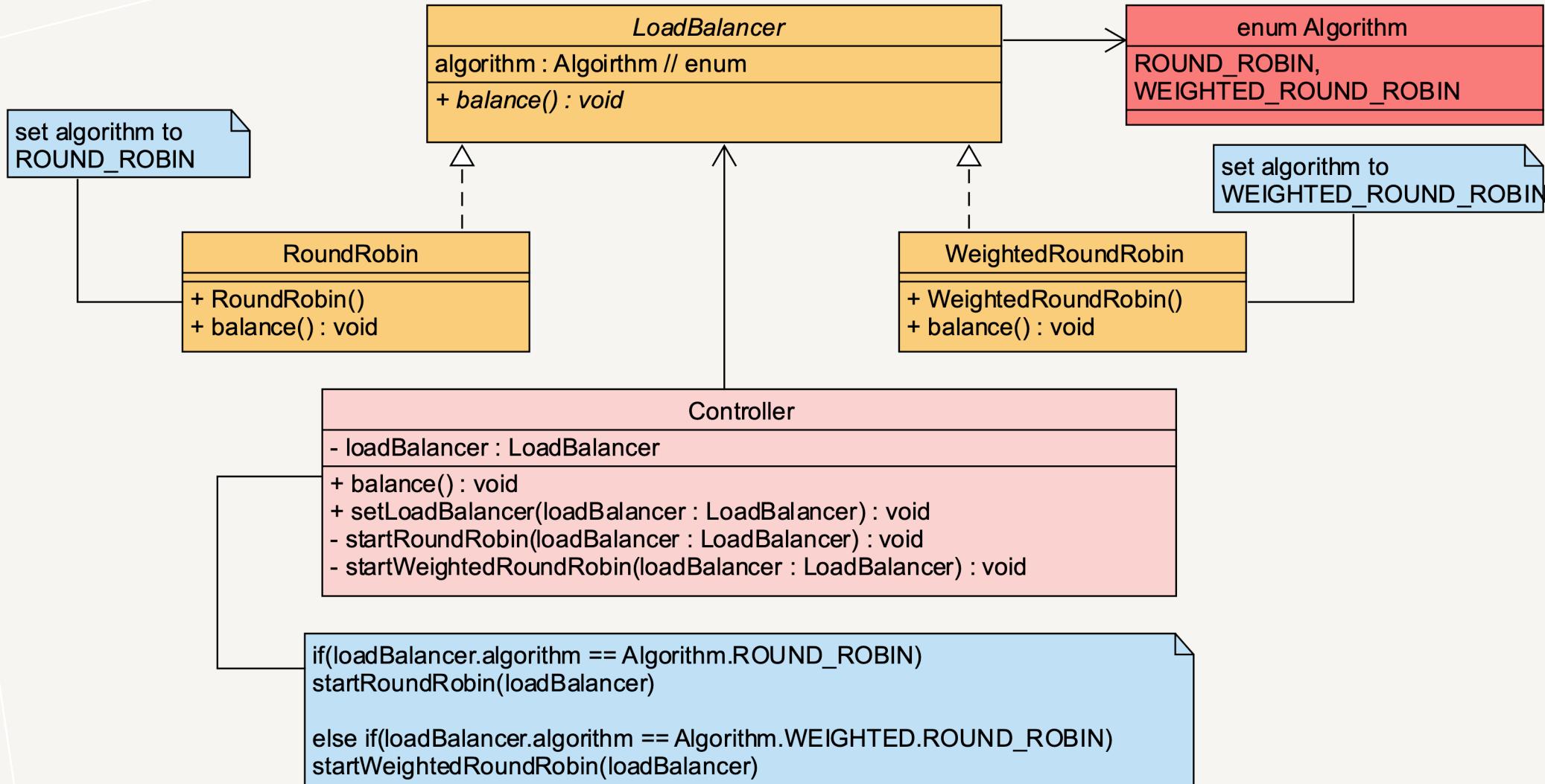
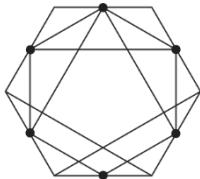


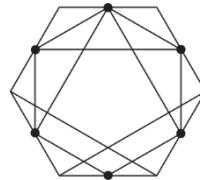


OPEN CLOSE PRINCIPLE WORKSHOP



• WORKSHOP

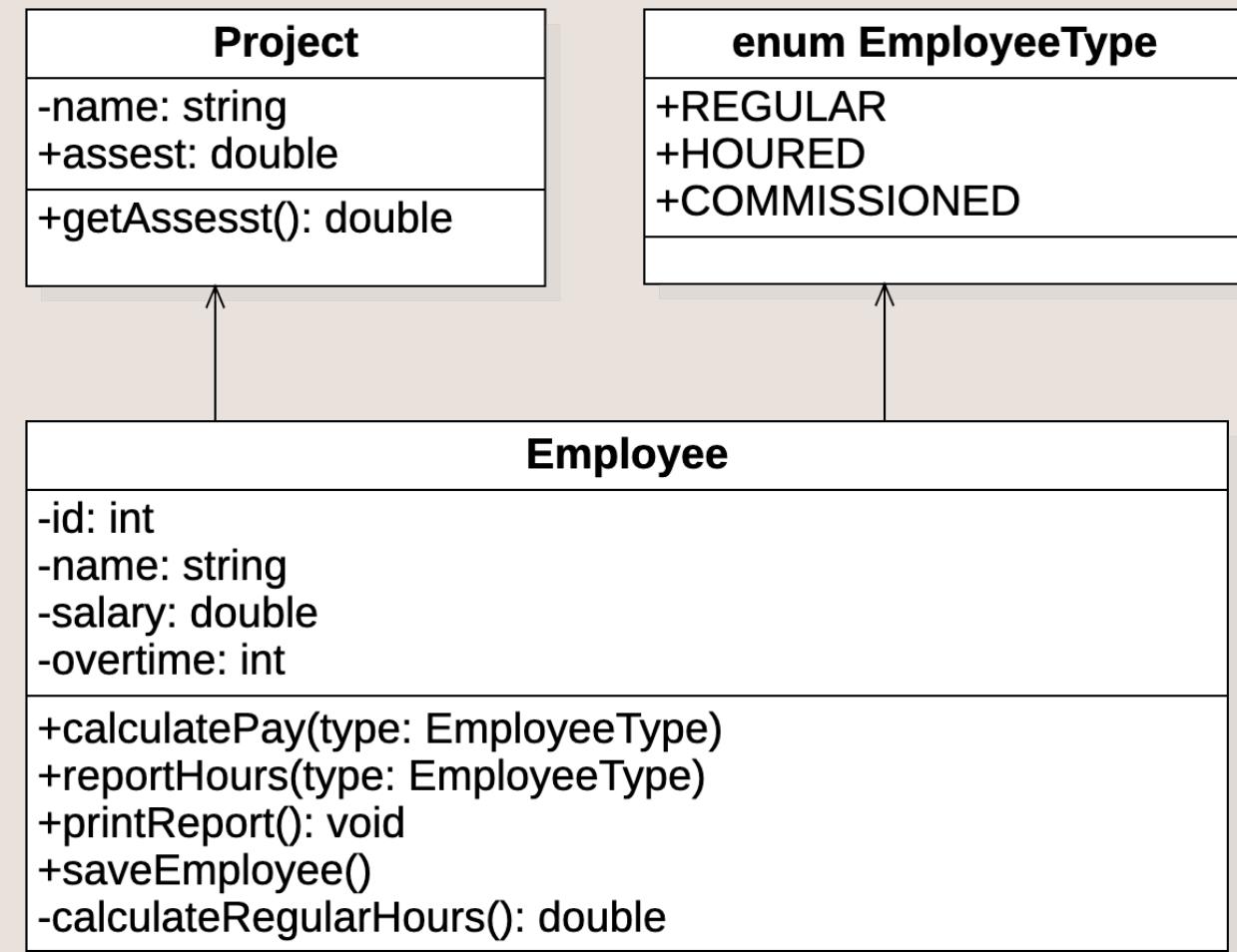
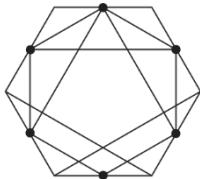




OPEN CLOSE PRINCIPLE WORKSHOP

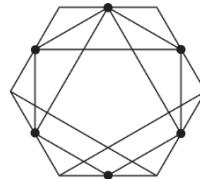


• WORKSHOP



OCPBESTEHENDES DESIGN

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



ÖSUNGEN UND DISKUSSIONEN





VIELEN DANK FÜR IHRE AUFMERKSAMKEIT.