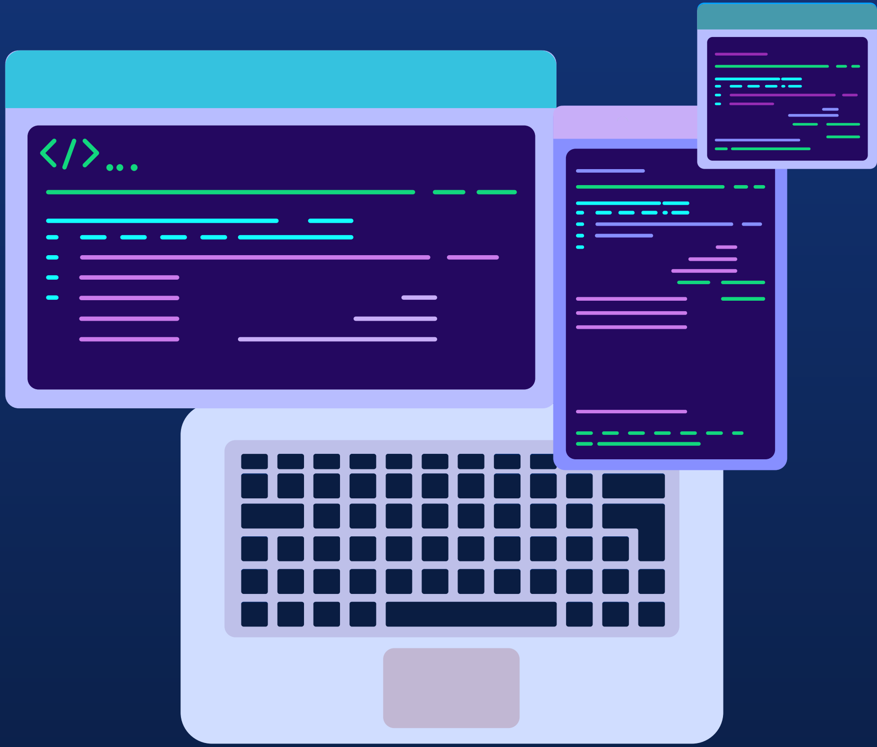
SOLIDe Software durch Linting



@DanielRosowski
@smartsquarehq

AGENDA

01
SOLID
Was heißt SOLID? Warum sollten wir darauf achten?

02
Linting
Was ist Linting, mit welchen Tools?

03
Demo
Demo Time!

04
Recap
Welche Verstöße lassen auf welche SOLID Verletzungen schließen?

01

SOLID

S ingle Responsibility

O pen-Closed

L iskov Substitution

I nrface Segregation

D ependency Inversion

Single Responsibility Principle

SINGLE RESPONSIBILITY PRINCIPLE
Just Because You Can, Doesn't Mean You Should

Quelle: http://www.vishalchovatiya.com

Open-Closed Principle

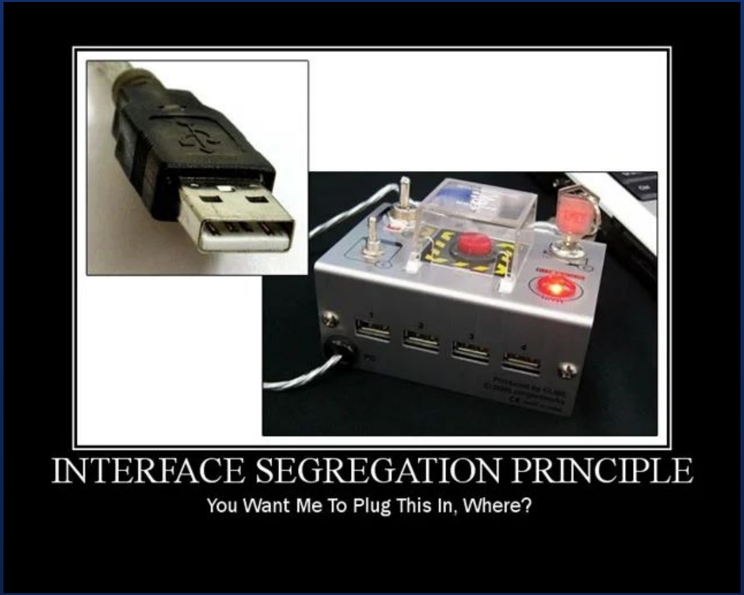
Quelle: http://www.vishalchovatiya.com

Liskov-Substitution Principle

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

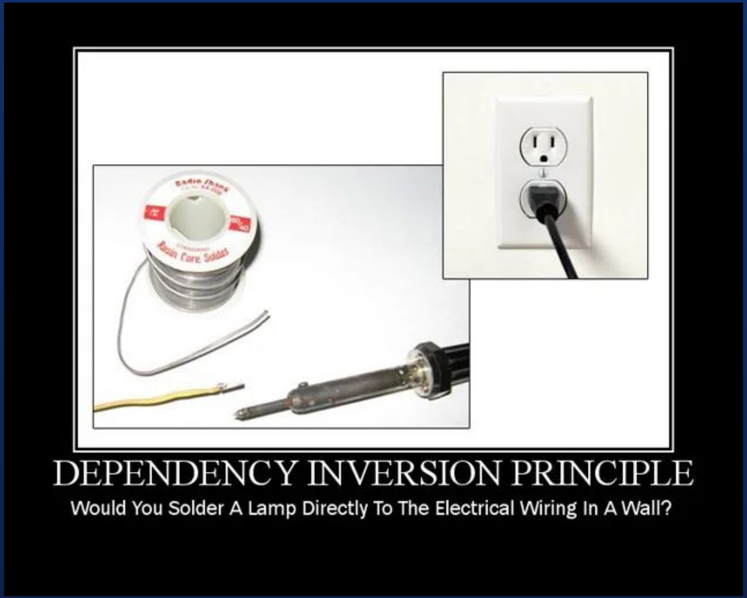
Quelle: http://www.vishalchovatiya.com

Interface-Segregation Principle


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

Quelle: http://www.vishalchovatiya.com

Dependency-Inversion Principle

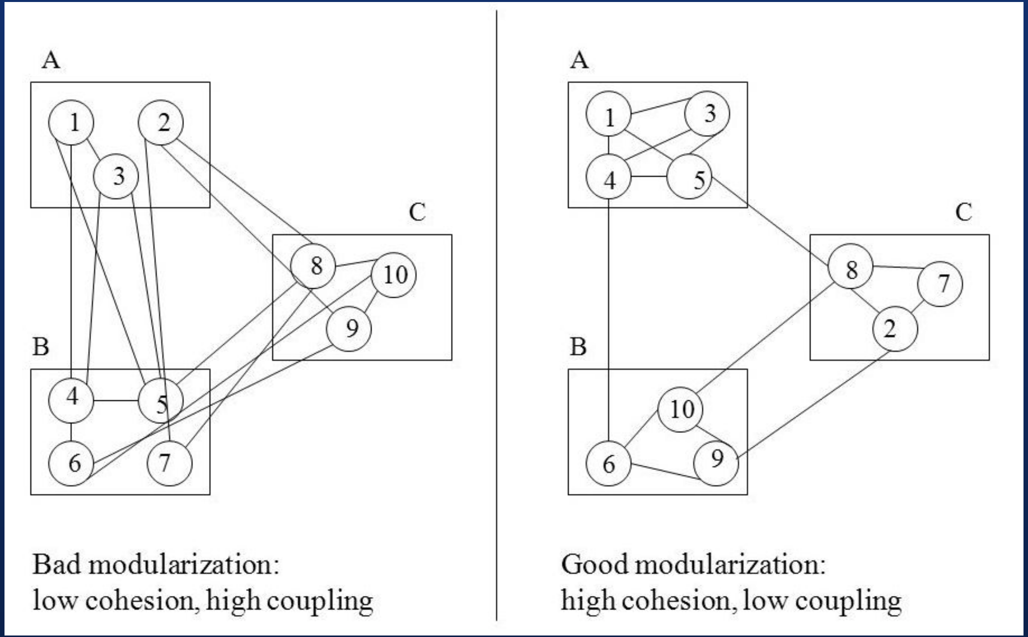

Radio Shack
Rosin Core Solder
DEPENDENCY INVERSION PRINCIPLE
Would You Solder A Lamp Directly To The Electrical Wiring In A Wall?

Quelle: http://www.vishalchovatiya.com

“One bad programmer can easily create two new jobs a year.”

—David Parnas

High cohesion, low Coupling

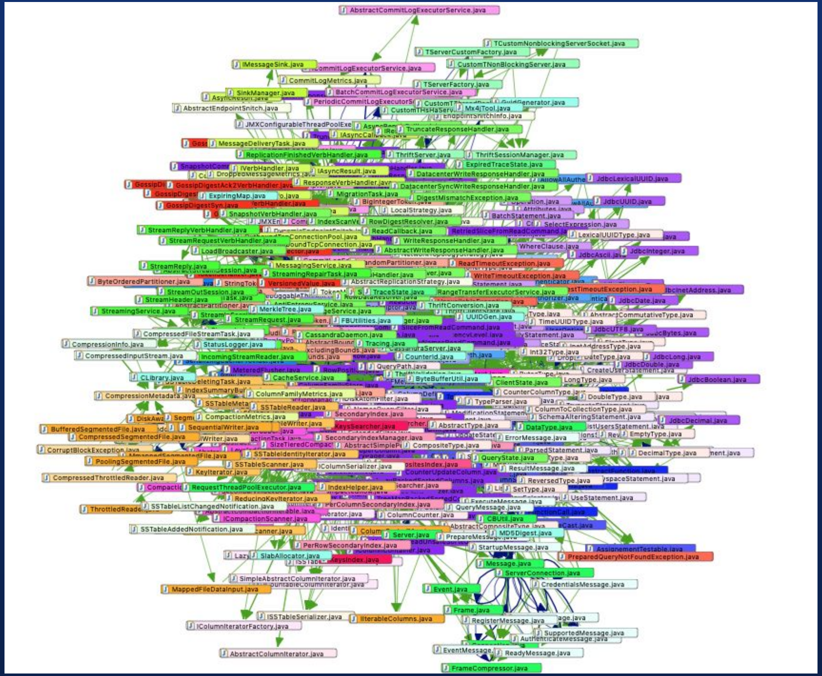


Bad modularization:
low cohesion, high coupling

Good modularization:
high cohesion, low coupling

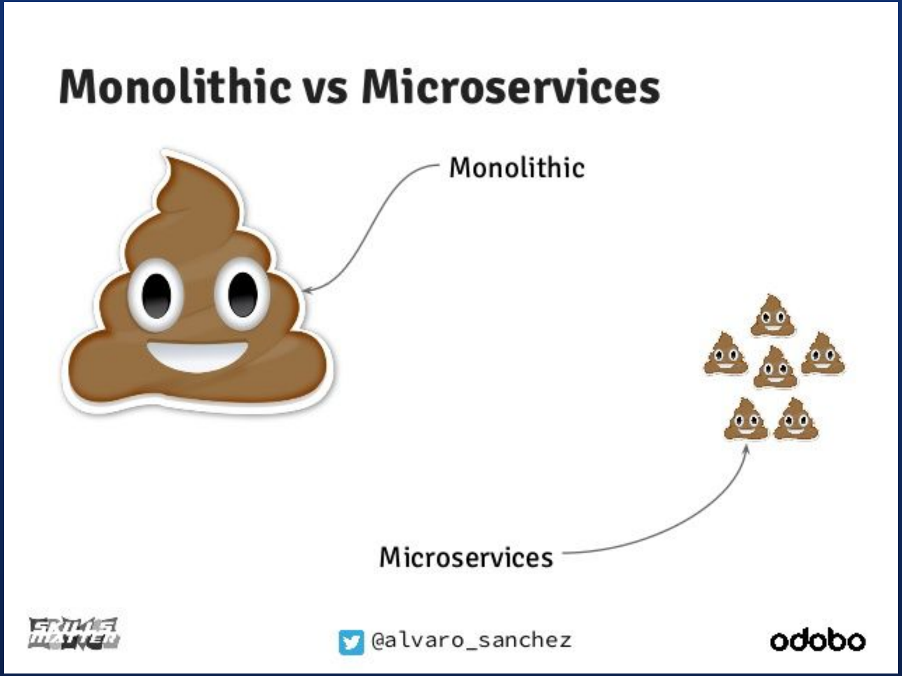
Quelle: https://devopedia.org/cohesion-vs-coupling

Big Ball of Mud



Quelle:
https://blog.hello2morrow.com/2021/05/analyzing-software-with-advanced-visualizations/

Microservices?



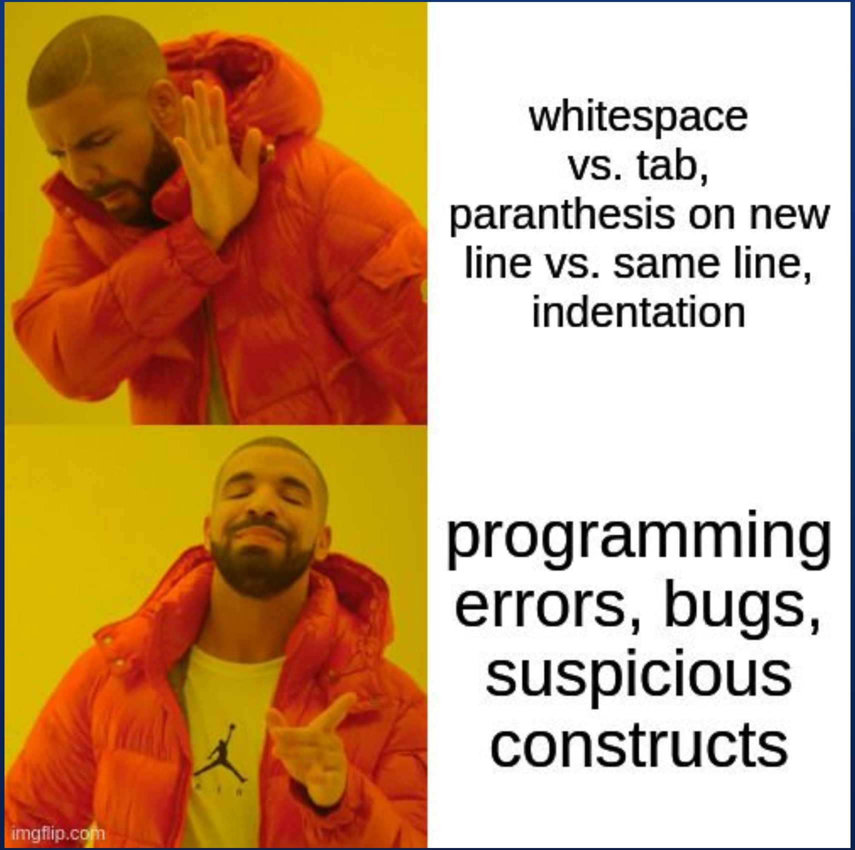
Quelle: https://blog.tekaris.com/blog/tag/metrics/

02

LINTING

"Lint, or a linter, is a static code analysis tool used to flag programming errors, bugs, stylistic errors and suspicious constructs."

—Wikipedia


whitespace
vs. tab,
paranthesis on new
line vs. same line,
indentation
programming
errors, bugs,
suspicious
constructs
imgflip.com

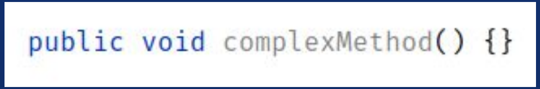
Hör auf deinen Linter!

SpotBugs

checkstyle

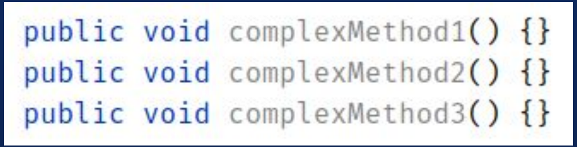
ktlint

Zyklomatische Komplexität



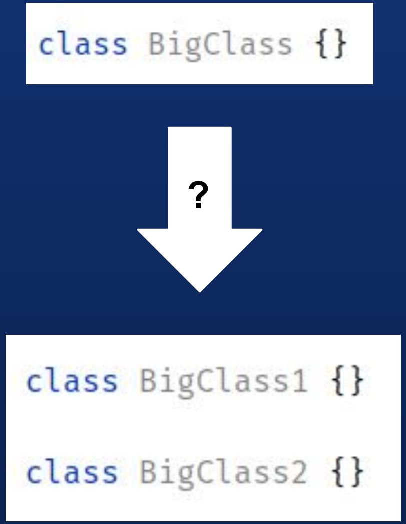
```
public void complexMethod() {}
```

?



```
public void complexMethod1() {}
public void complexMethod2() {}
public void complexMethod3() {}
```

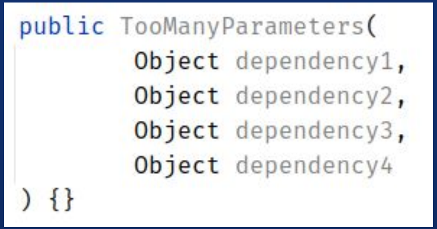
Zu viele Methoden



```
class BigClass {}
```

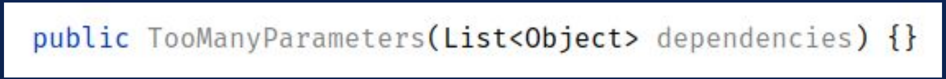
```
class BigClass1 {}

class BigClass2 {}
```

Zu viele Parameter



```java
public TooManyParameters(
        Object dependency1,
        Object dependency2,
        Object dependency3,
        Object dependency4
) {}
```

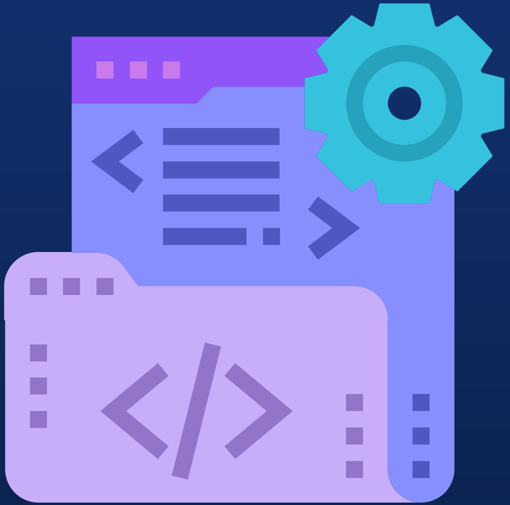
?



```java
public TooManyParameters(List<Object> dependencies) {}
```

03

DEMO!



04

RECAP

Verletzungen und mögliche Ursachen

Verletzung	**detekt**	**checkstyle**
Single Responsibility	LargeClass, TooManyFunctions, ComplexMethod, LongMethod	MethodCount, MethodLength

Verletzungen und mögliche Ursachen

Verletzung	**detekt**	**checkstyle**
Open/Closed	TooManyFunctions, ComplexMethod	MethodCount, ParameterNumber, JavaNCSS

Verletzungen und mögliche Ursachen

Verletzung	**detekt**	**checkstyle**
Interface Segregation	TooManyFunctions	MethodCount

Verletzungen und mögliche Ursachen

Verletzung	**detekt**	**checkstyle**
Dependency Inversion	LongParameterList	ParameterNumber

DANKE!

Fragen? Antworten!

@DanielRosowski
www.smartsquare.de