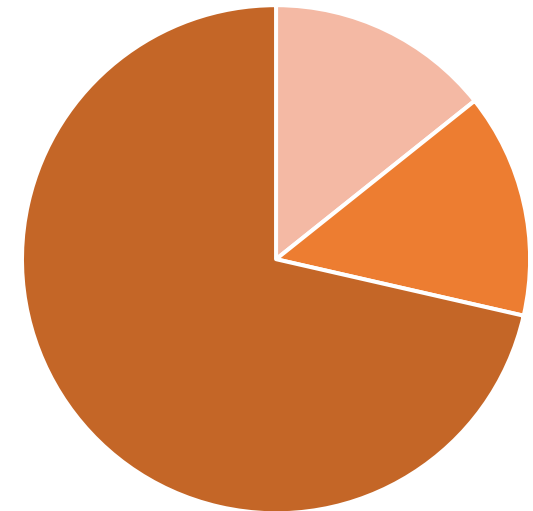


Code lesen
kann doch jeder!?

Vorwort

Alle Kinder lernen lesen...

- Alphabetische Phase
 - char[] → String
- Orthographische Phase
 - String → Satz
- Reflektiertes/Analytisches Lesen
 - **Verstehen**
 - > 10 Jahre



Alle Kinder lernen lesen...

Entwickler?

Alle Kinder lernen lesen...

Entwickler?

- Code schreiben
- Code Review – lesbaren Code schreiben

Measuring Program Comprehension: A Large-Scale Field Study with Professionals

2017

Xin Xia, Lingfeng Bao, David Lo, Zhenchang Xing, Ahmed E. Hassan, Shanping Li

We analyze this recorded data, and we find that on average, developers spend up to ~58% of their time on program comprehension, and they frequently use web browsers and

X. Xia et al.: "Measuring Program Comprehension: A Large-Scale Field Study with Professionals", IEEE TSE 2018



"Indeed, the ratio of time spent reading versus writing is well over 10 to 1. We are constantly reading old code as part of the effort to write new code. ...[Therefore,] making it easy to read makes it easier to write."

— Robert C. Martin, *Clean Code: A Handbook of Agile Software Craftsmanship*

<https://www.goodreads.com/quotes/835238-indeed-the-ratio-of-time-spent-reading-versus-writing-is>

I Know What You Did Last Summer An Investigation of How Developers Spend Their Time

Roberto Minelli, Andrea Mocci and Michele Lanza
REVEAL @ Faculty of Informatics — University of Lugano, Switzerland

2015

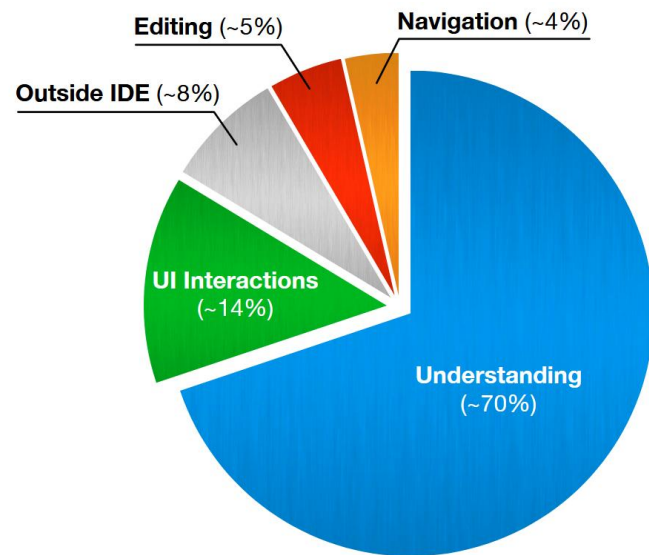


Fig. 3. How do developers spend their time?

R. Minelli, A. Mocci, M. Lanza: "I Know What You Did Last Summer", IEEE MSR 2015

Kapitel 1

Falschfahrer



...ich rede von einem Ort, wo einen das Lesen in den Wahnsinn treiben kann. Wo Bücher verletzen, vergiften, ja sogar töten können. Nur wer wirklich bereit ist, für die Lektüre dieses Buches derartige Risiken in Kauf zu nehmen, wer bereit ist, sein Leben auf Spiel zu setzen, um an meiner Geschichte teilzuhaben, der sollte mir zum nächsten Absatz folgen.

....

Und sagt nicht, ich hätte euch nicht gewarnt!

Walter Moers: Die Stadt der Träumenden Bücher (2004)

```
List<Integer> lst = Arrays.stream(a).boxed().collect(Collectors.toList());

s = lst.stream().map(n -> "[" + n + "]").collect(Collectors.joining(", "));
log.debug("input: {}", s);

for (int i = 0; i < a.length; i++) {
    tmp = a[i] * 2;
    if (tmp > 50) {
        x = x + tmp;
    }
}
```

```
List<Integer> lst = Arrays.stream(a).boxed().collect(Collectors.toList());

s = lst.stream().map(n -> "[" + n + "]").collect(Collectors.joining(", "));
log.debug("input: " + s);

for (int i = 0; i < a.length; i++) {
    tmp = a[i] * 2;
    if (tmp > 50) {
        x = x + tmp;
    }
}
```

...ich rede von einem Ort, wo einen das Lesen in den Wahnsinn treiben kann. Wo Bücher verletzen, vergiften, ja sogar töten können. Nur wer wirklich bereit ist, für die Lektüre dieses Buches derartige Risiken in Kauf zu nehmen, wer bereit ist, sein Leben auf Spiel zu setzen, um an meiner Geschichte teilzuhaben, der sollte mir zum nächsten Absatz folgen.

....

Und sagt nicht, ich hätte euch nicht gewarnt!

```
List<Integer> lst = Arrays.stream(a)
```

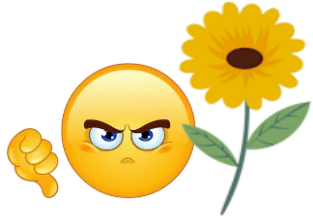
```
s = lst.stream().map(n -> "[" + n +  
log.debug("input: " + n));
```

```
for (int i = 0; i < a.length; i++) {  
    tmp = a[i] * 2;
```

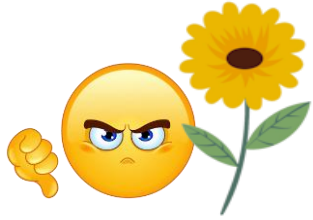
```
    if (tmp > 50) {  
        x = x + tmp;
```

```
    }
```

```
}
```



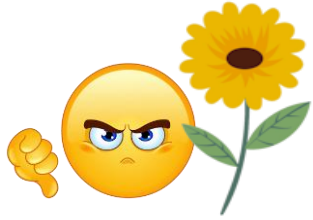
- Ablenkung / Selbstblockade
- mentale Ressourcenverschwendung



- Ablenkung / Selbstblockade
- mentale Ressourcenverschwendung

der zu lesende Code ist nicht änderbar !!

public static final



- Ablenkung / Selbstblockade
- mentale Ressourcenverschwendung

Entwicklertypen

der zu lesende Code ist nicht änderbar !!

public **static final**

Entwicklertypen

Der Feigling

- Keine Entscheidungen
- auskommentierter Code
- ‚old‘, ‚backup‘, ...
- if, if, if ganz unten
- // don't touch this



Entwicklertypen

Der Trapper

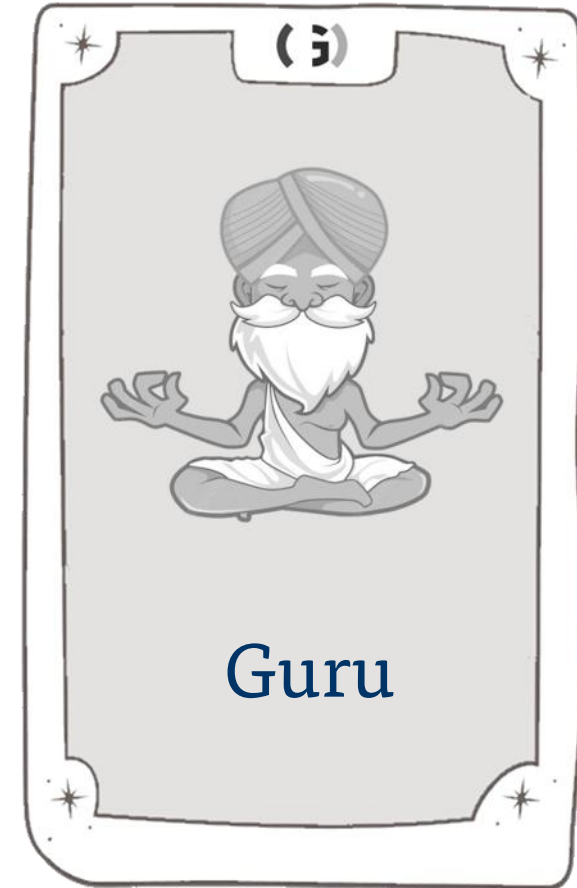
- Stellt Fallen (und wartet hinter einem Baum)
- Zustand, der von Reihenfolgen abhängt
- ähnliche Überladungen
- Seiteneffekte vs. Methodennamen



Entwicklertypen

Der Guru

- Code als Selbstbeweihräucherung
- Kann/will nicht erklären
- Hauptsache ‚anders‘
- eigene Frameworks
- Gerne auch andere Programmiersprache als das Team



Entwicklertypen

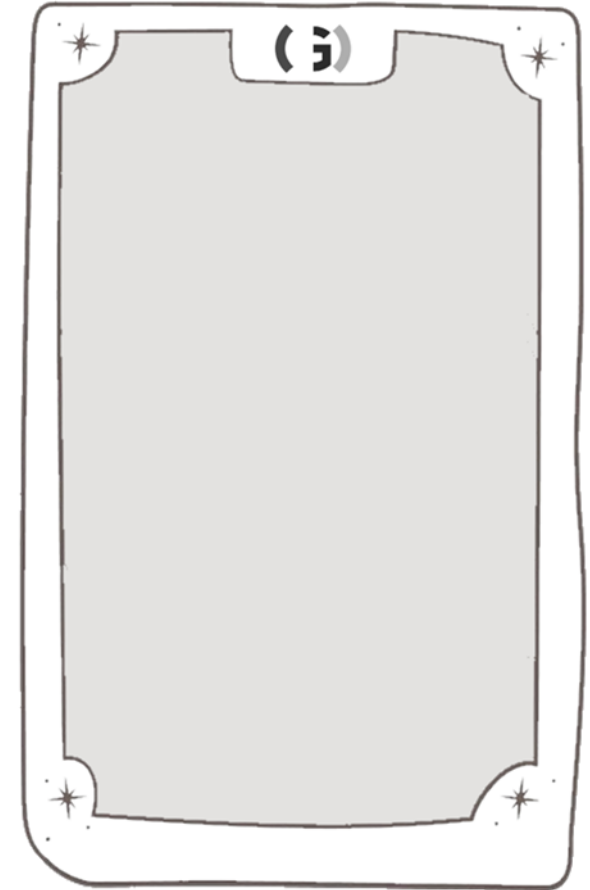
Der Mönch

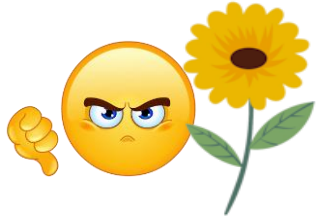
- Regeln, Regeln, Regeln
- Uncle Bob hat gesagt...
- private Einzeiler mit sehr sprechenden Namen - Methodenkaskaden
- Diskutiert über Namen und Formatierung



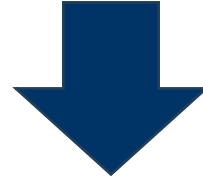
Entwicklertypen

- [6 Programmierer-Typen im Kurzprofil – welcher bist du?](https://t3n.de/news/programmierer-typen-700582/)
(<https://t3n.de/news/programmierer-typen-700582/>)
- [19 Programmierer-Archetypen: Der Psychotest für Entwickler](https://entwickler.de/programmierung/19-programmierer-archetypen-der-psychotest-fur-entwickler)
(<https://entwickler.de/programmierung/19-programmierer-archetypen-der-psychotest-fur-entwickler>)
- [Die verschiedenen Charaktere in Entwickler-Teams: Humorvoll erklärt](https://pep-digital.de/blog/die-verschiedenen-charaktere-in-entwickler-teams-humorvoll-erklaert)
(<https://pep-digital.de/blog/die-verschiedenen-charaktere-in-entwickler-teams-humorvoll-erklaert>)
- [4 types of programmers](https://www.linkedin.com/pulse/4-types-programmers-william-ghelfi/)
(<https://www.linkedin.com/pulse/4-types-programmers-william-ghelfi/>)





- Ablenkung / Selbstblockade
- mentale Ressourcenverschwendung



Entwicklertypen

(überliste dich selbst)

Was für ein Entwickler hat wohl diesen Code geschrieben?

Kapitel 2

Surrounded by Idiots



98,765%

des Codes den wir lesen,
tut genau das, was er soll

98,765%

des Codes den wir lesen,
tut genau das, was er soll

- von deinen Teamkollegen geschrieben
- in deinem Team gereviewed
- **ist produktiv**
- in deinem Unternehmen entwickelt
- vom Unternehmen gekaufte Lösung (nach Auswahlprozess)
- weltweit im Einsatz
- von Dir geschrieben

98,765%

des Codes den wir lesen,
tut genau das, was er soll

76,543 %

des Codes den wir lesen,
halten wir für **fehlerhaft**

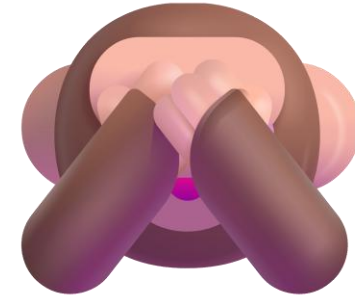
Vertrauen

der Code den wir lesen,
tut genau das, was er soll

(fast) niemand schreibt
code um dir das
Le(b|s)en schwer zu
machen

Vertrauen

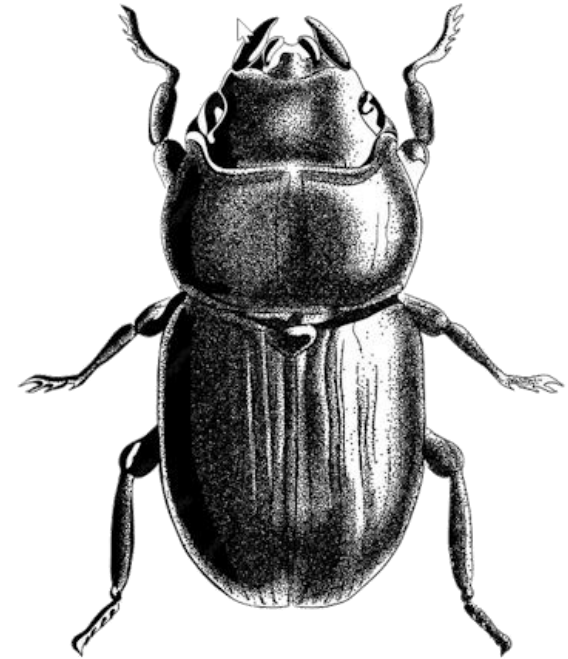
der Code den wir lesen,
tut genau das, was er soll



Blind für die Details

Kapitel 3

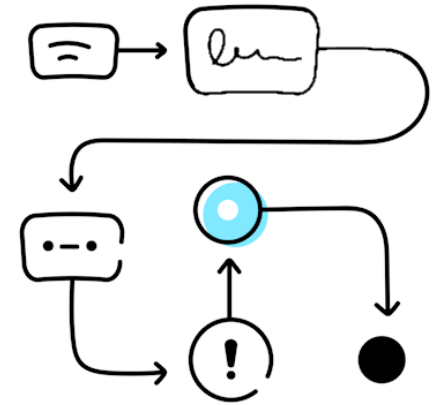
Verwandlung



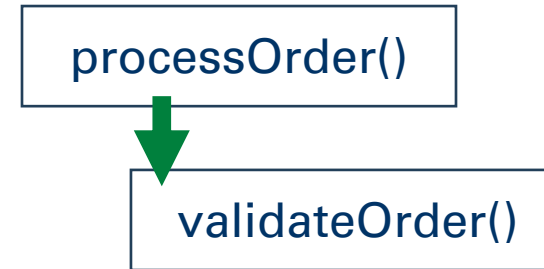
processOrder()



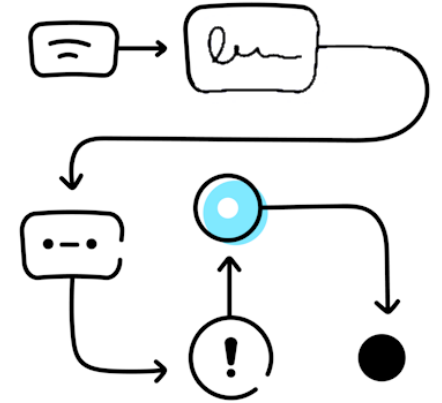
```
public void processOrder(Order order) {  
    validateOrder(order);  
    calculateTotalPrice(order);  
}
```



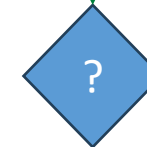
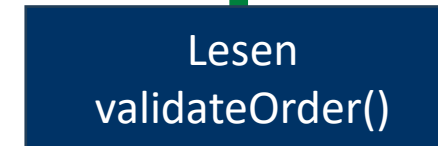
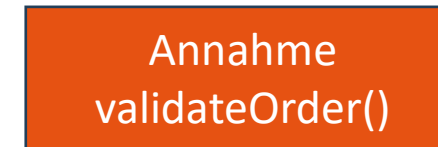
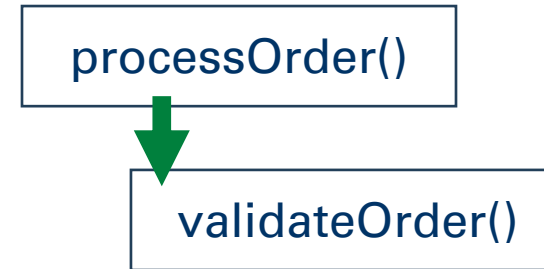
```
public void processOrder(Order order) {  
    validateOrder(order);  
    calculateTotalPrice(order);  
}
```



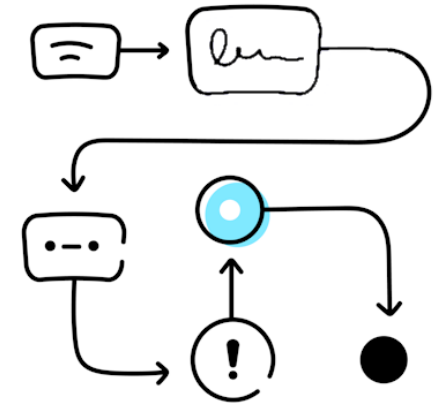
Annahme
validateOrder()



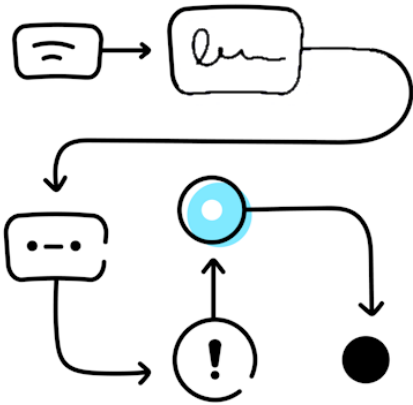
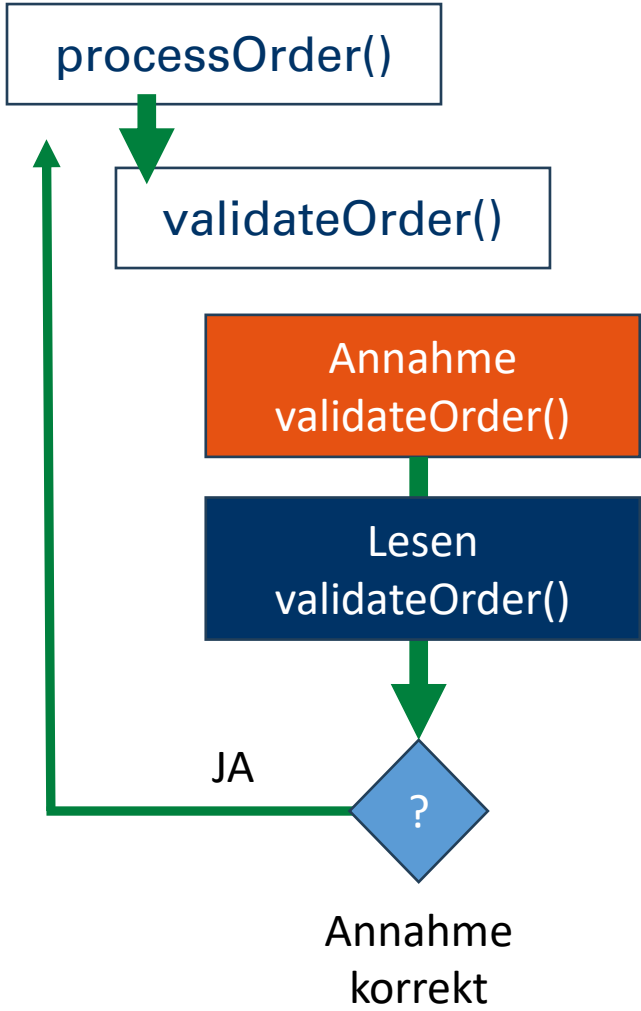
```
public void processOrder(Order order) {
    validateOrder(order);
    calculateTotalPrice(order);
}
```



Annahme
korrekt



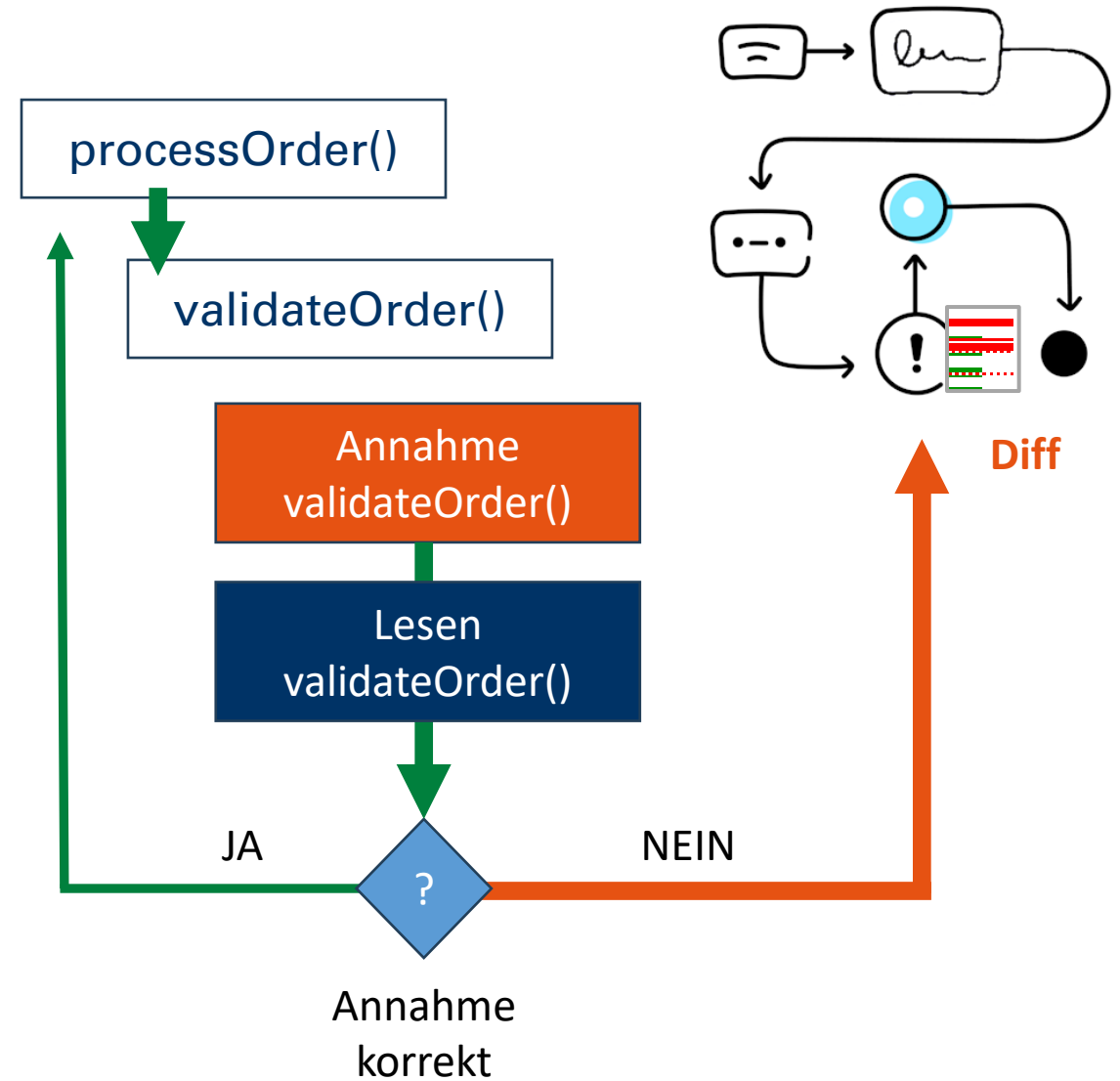
```
public void processOrder(Order order) {
    validateOrder(order);
    calculateTotalPrice(order);
}
```



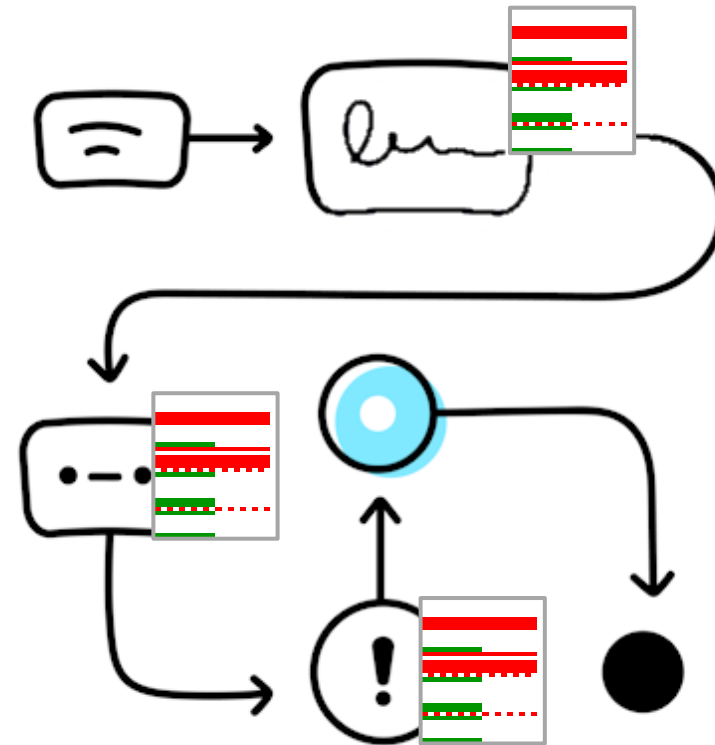
```

public void processOrder(Order order) {
    validateOrder(order);
    calculateTotalPrice(order);
}

```

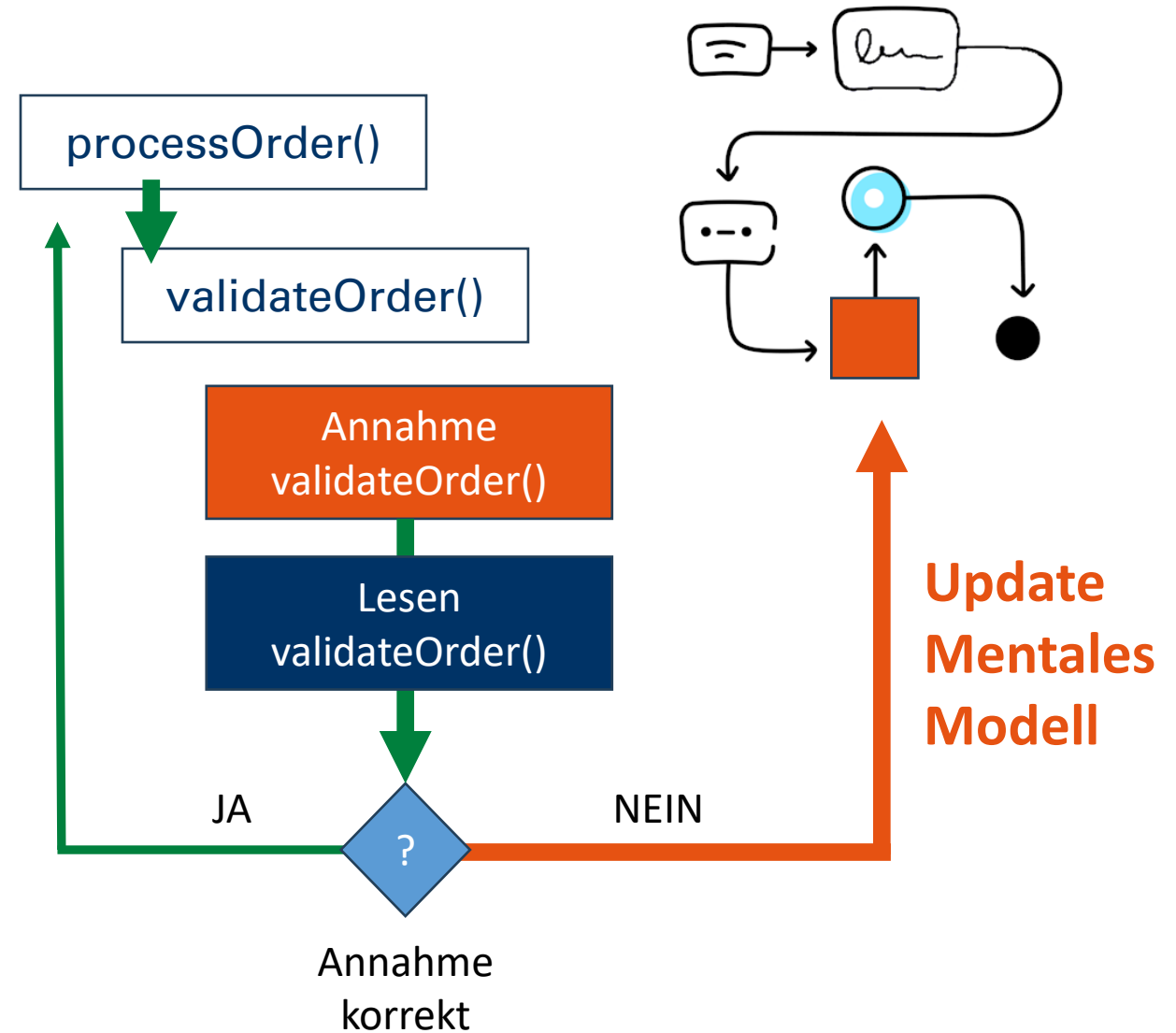


```
public void processOrder(Order order) {  
    validateOrder(order);  
    calculateTotalPrice(order);  
}
```



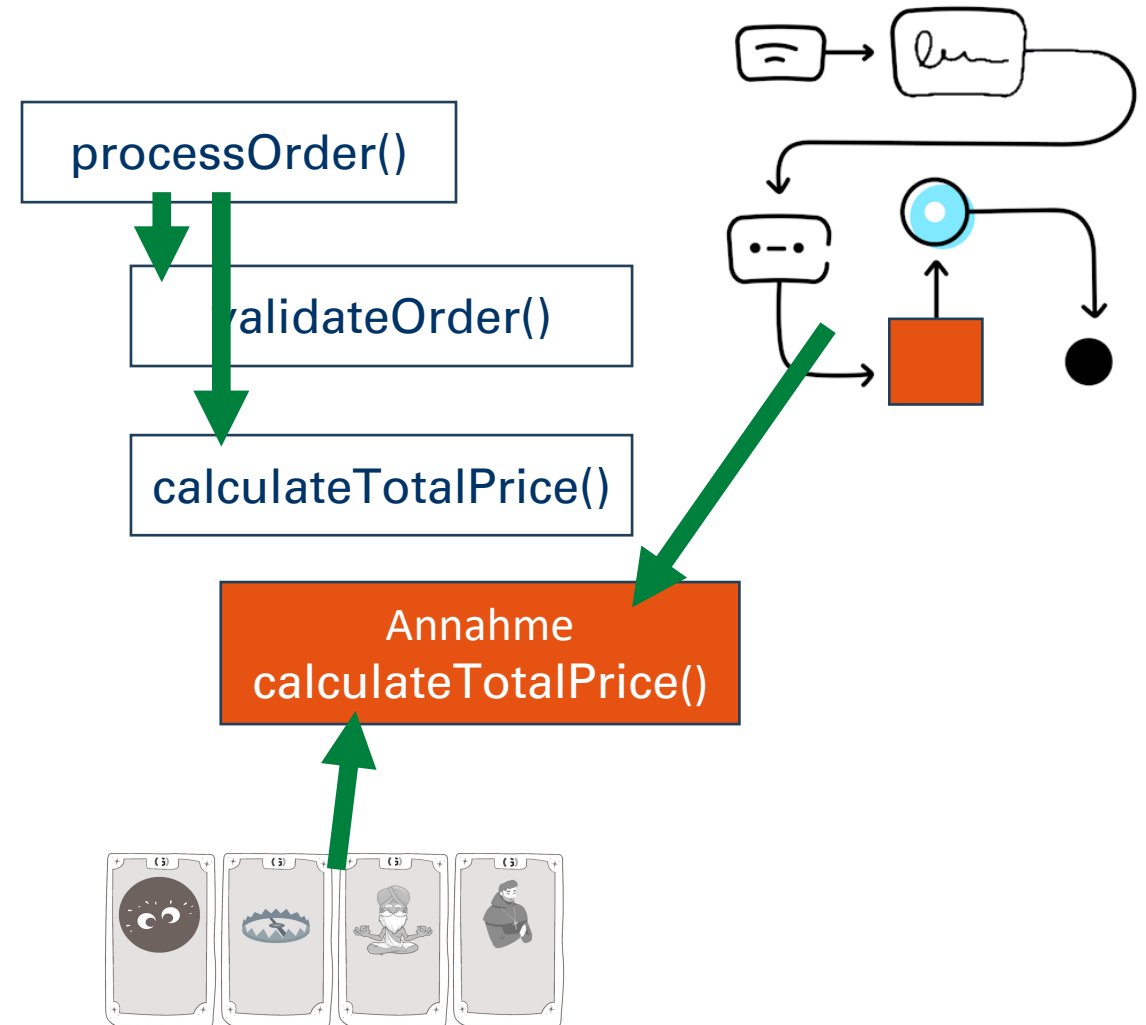
Verwandlung

```
public void processOrder(Order order) {  
    validateOrder(order);  
    calculateTotalPrice(order);  
}
```



Verwandlung

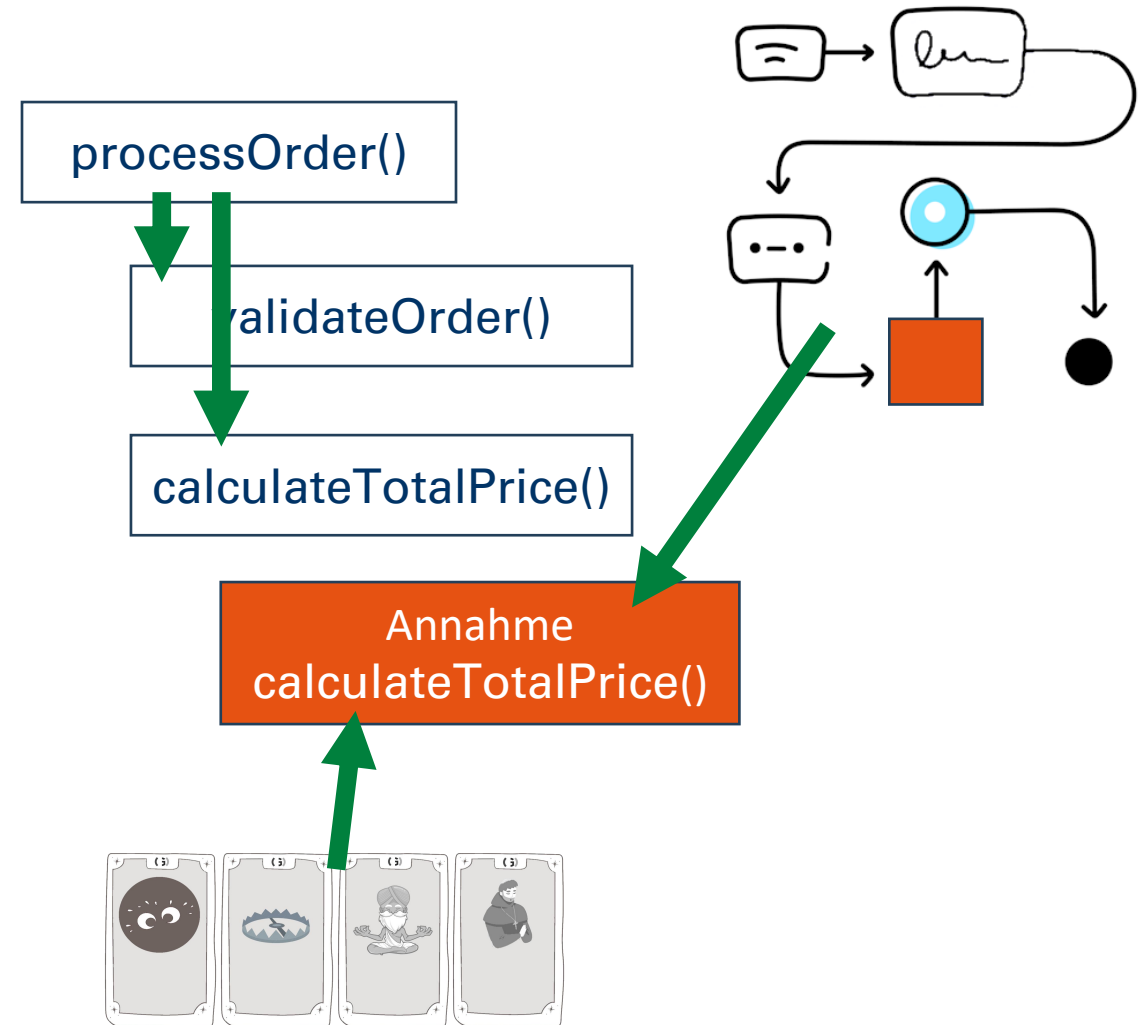
```
public void processOrder(Order order) {  
    validateOrder(order);  
    calculateTotalPrice(order);  
}
```



Verwandlung



Denkt wie der
Ersteller des Codes



Finale

Lesbarer Code



Denkt wie der
Ersteller des Codes

Code ist gut lesbar, wenn ein geübter Leser schnell im Flow ist, unabhängig von der Stelle, an der er mit dem Lesen beginnt.

Ein Quellcode C gilt genau dann als lesbar, wenn die kognitive Latenz bis zum Erreichen des konzeptuellen Kontextes (Flow-Zustand) für einen domänenkundigen Entwickler über alle beliebigen Einstiegspunkte $p \in C$ hinweg vernachlässigbar klein und im Wesentlichen konstant ist.

Software **lesen** und **verstehen** ist sehr anstrengend.

- erfordert Verständnis der Fachlichkeit
- Code hat wenig Redundanzen — Code ist verzweigt
- Bücher haben einen Anfang
- Bücher haben einen Autor — Code hat viele
- Bücher erlauben Mehrdeutigkeit — Code nicht
- Geschichten können unvollständig sein und trotzdem funktionieren — Code nicht
-

Software **lesen** und **verstehen** ist sehr anstrengend.



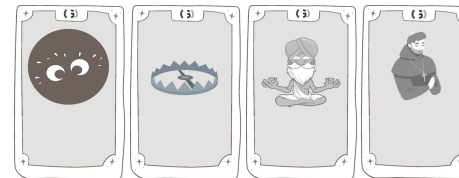
Bleibt auf
dem Weg



Denkt wie
der Autor



habt Vertrauen
seit neugierig



Software **lesen** und **verstehen**
ist sehr anstrengend.

Üben!



Ende

Über mich

Ronald Brill
Principal



G E B I T
Solutions



<https://github.com/rbri>

<HtmlUnit/>

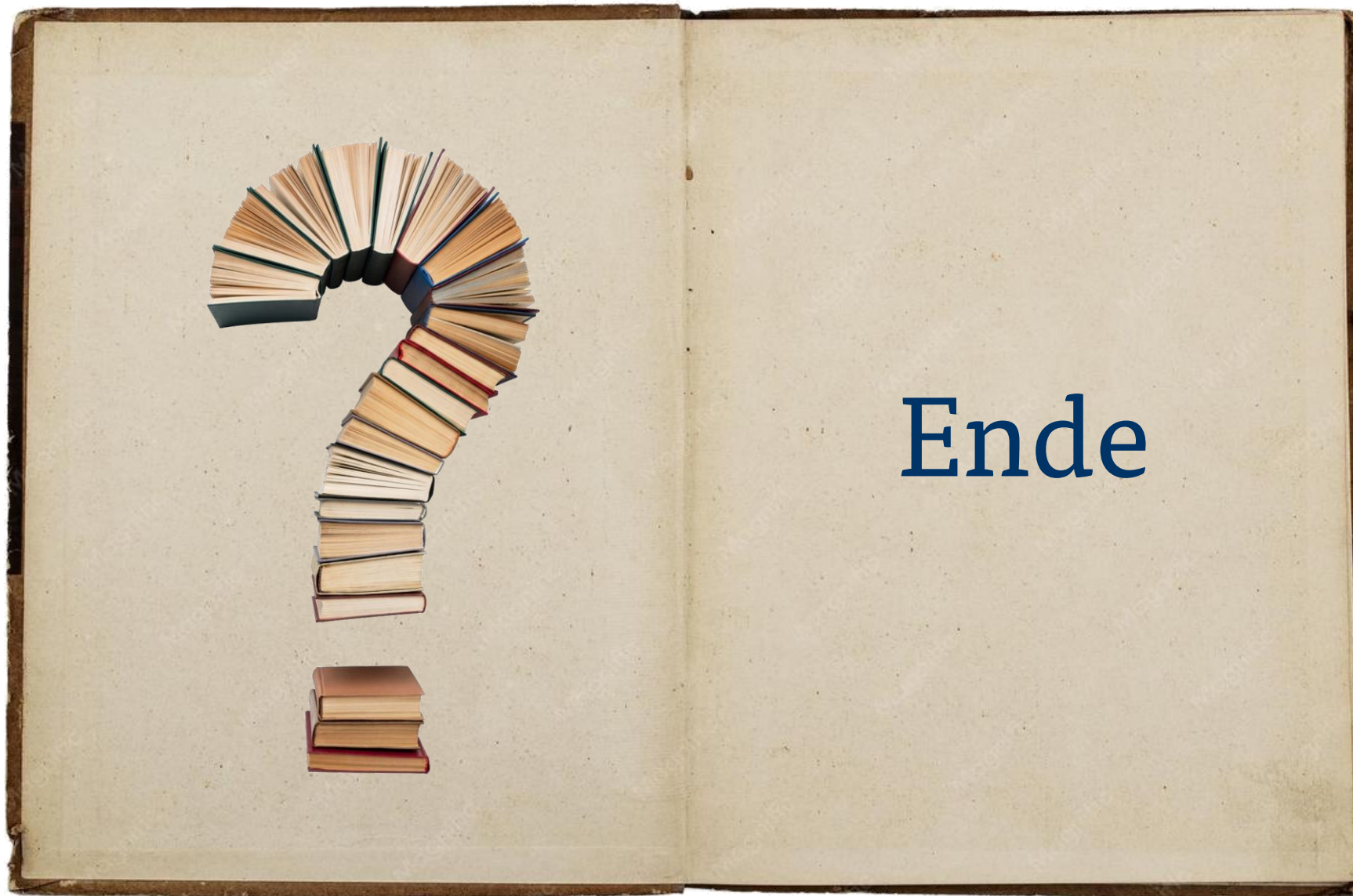
- [@HtmlUnit](#)
- <https://github.com/HtmlUnit>
- <https://github.com/SeleniumHQ/htmlunit-driver>
- [Mozilla Rhino,](#)

BingoBongoTesting

- www.bingobongotesting.org/



- www.wetator.org/



Referenzen

Studien

- R. Minelli, A. Mocci, M. Lanza (2015): "I Know What You Did Last Summer: An Investigation of How Developers Spend Their Time." IEEE International Conference on Mining Software Repositories (MSR)
<https://de.slideshare.net/slideshow/i-know-what-you-did-last-summer-an-investigation-of-how-developers-spend-their-time/48381643>
- Xia, X., Bao, L., Lo, D., Xing, Z., Hassan, A. E., & Li, S. (2018). "Measuring Program Comprehension: A Large-Scale Field Study with Professionals." IEEE Transactions on Software Engineering, 44(10), 951–976.
<https://baolingfeng.github.io/papers/tsecomprehension.pdf>

Bücher & Literatur

- Robert C. Martin (2008): Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall.
- Walter Moers (2004): Die Stadt der Träumenden Bücher. Piper Verlag.

Bilder & Grafiken

- <https://www.magnific.com>
- <https://openclipart.org>