embarc
Software Consulting GmbH

Legacy Code meistern in x einfachen Schritten

FALK SIPPACH // EMBARC

Developer Week 2021

Mittwoch, 30. Juni 2021, 14:15 Uhr

DWX DEVELOPER WEEK '21

Legacy Code meistern in x einfachen Schritten

Zusammenfassung

In einer idealen Welt würden wir nur "neuen" Code schreiben, der natürlich perfekt und wunderschön ist. Wir müßten nie wieder unseren Code anschauen geschweige denn 10 Jahre alte Projekte warten. Ende des Tagtraums ...

Leider ist unsere Welt nicht so ideal, unser Code von gestern ist heute schon Legacy. Diesen im Nachhinein zu verstehen, zu erweitern oder darin Fehler zu beheben ist immer eine Herausforderung, insbesondere wenn Tests fehlen. Trotzdem gibt es einfache Möglichkeiten, wie man die Qualität von Legacy Code verbessern kann. Das Wichtigste ist das Einziehen von Fangnetzen, so daß man trotz fehlender Tests guten Gewissens Änderungen durchführen kann. Wer Golden Master, Subclass to Test und Extract Pure Functions an konkreten Beispielen kennenlernen möchte, ist in dieser Session genau richtig.

Falk Sippach

- Softwarearchitekt, Berater, Trainer bei embarc
- früher bei Orientation in Objects (OIO), Trivadis

Schwerpunkte:

- Architekturberatung und -bewertung
- Cloud- und Java-Technologien

fs@embarc.de

@sippsack

➔ xing.to/fsi

embarc
Software Consulting GmbH

Legacy Code meistern in

8 einfachen Schritten

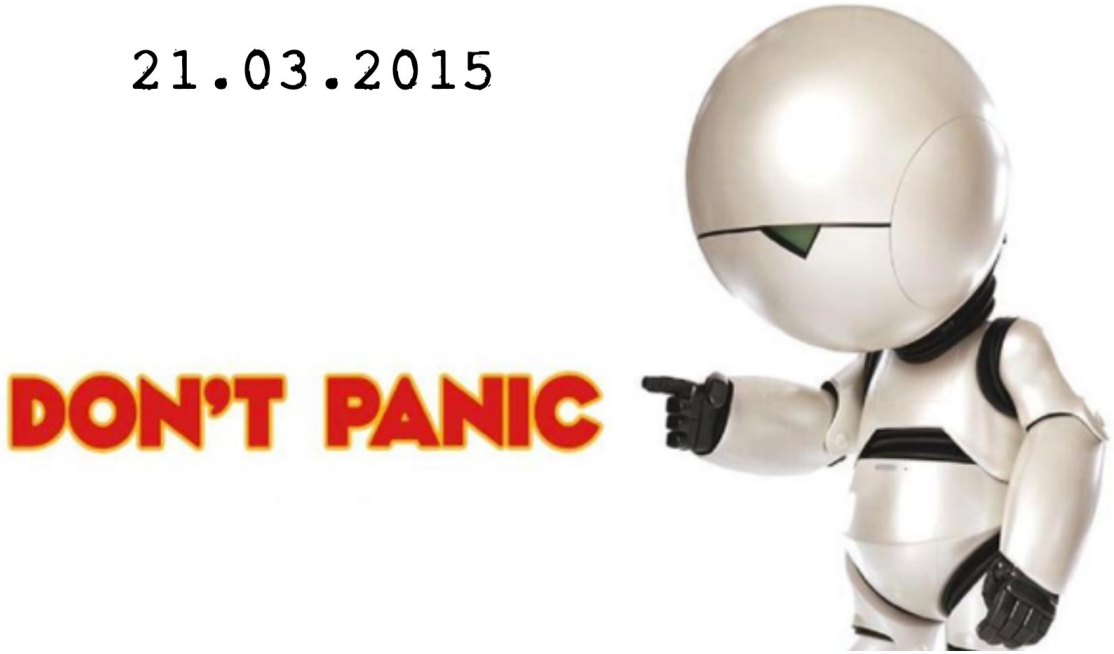
Nur heute,
nicht 1,
nicht 2,
...

Disclaimer: KEIN Projekterfahrungsbericht

1. Darmstädter

Legacy Code Retreat

21.03.2015



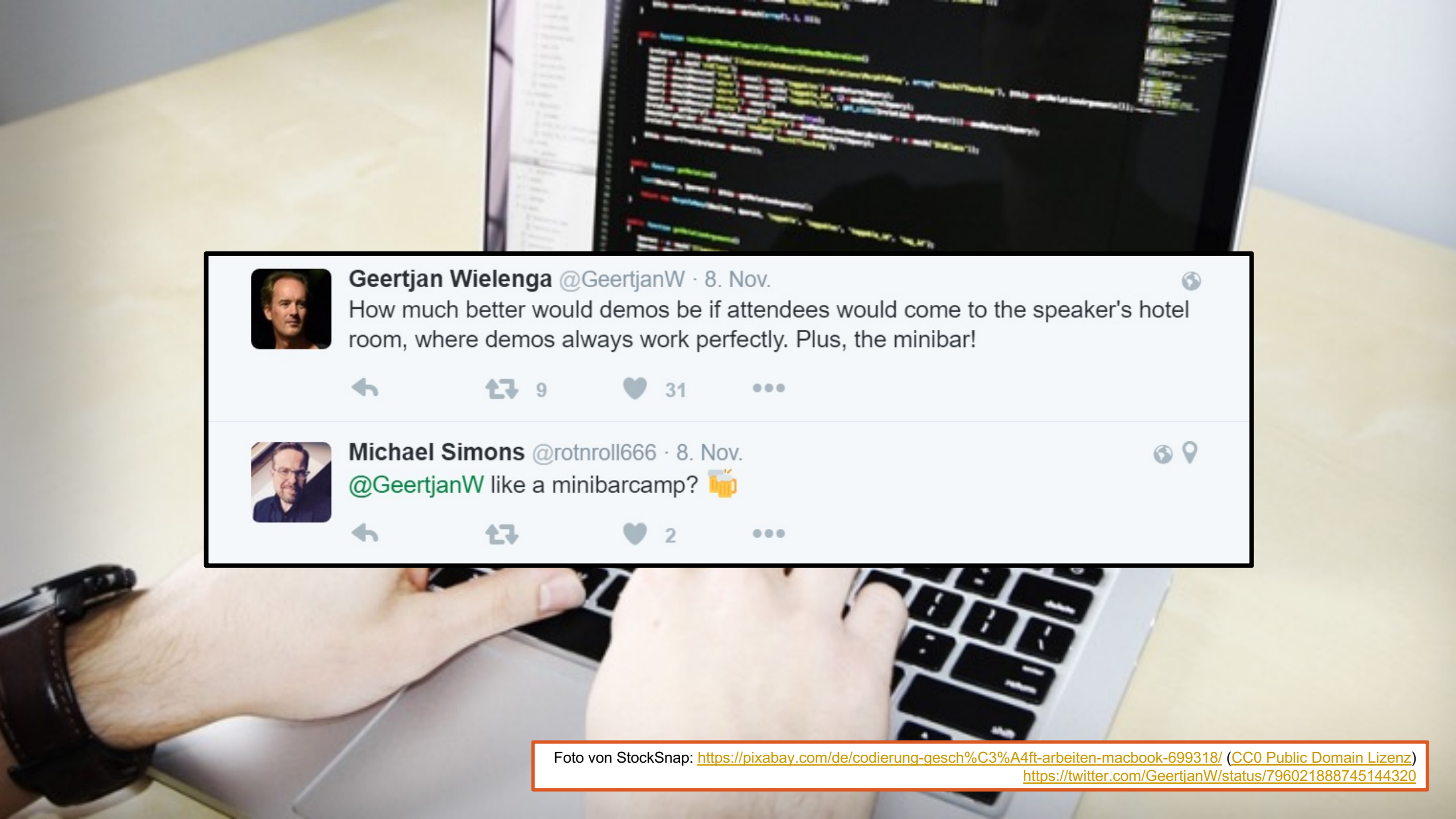


Foto von StockSnap: https://pixabay.com/de/codierung-gesch%C3%A4ft-arbeiten-macbook-699318/ (CC0 Public Domain Lizenz)
https://twitter.com/GeertjanW/status/796021888745144320

Agenda

1. Was ist Legacy Code?
2. Refactoring
3. Die ‘x‘ Schritte
4. Fazit und Ausblick

Agenda

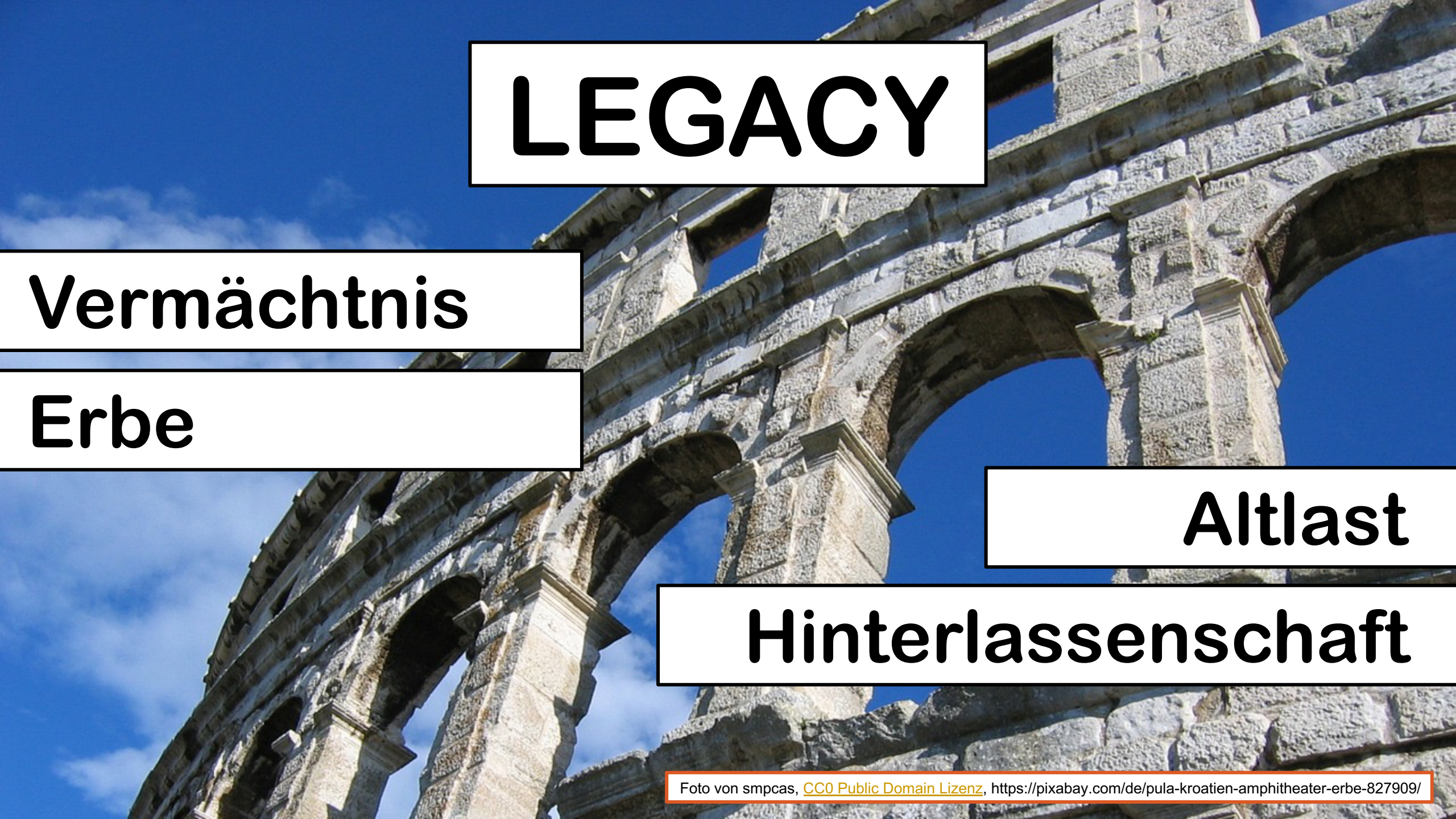
1 **Was ist Legacy Code?**

2 Refactoring

3 Die ‘x‘ Schritte

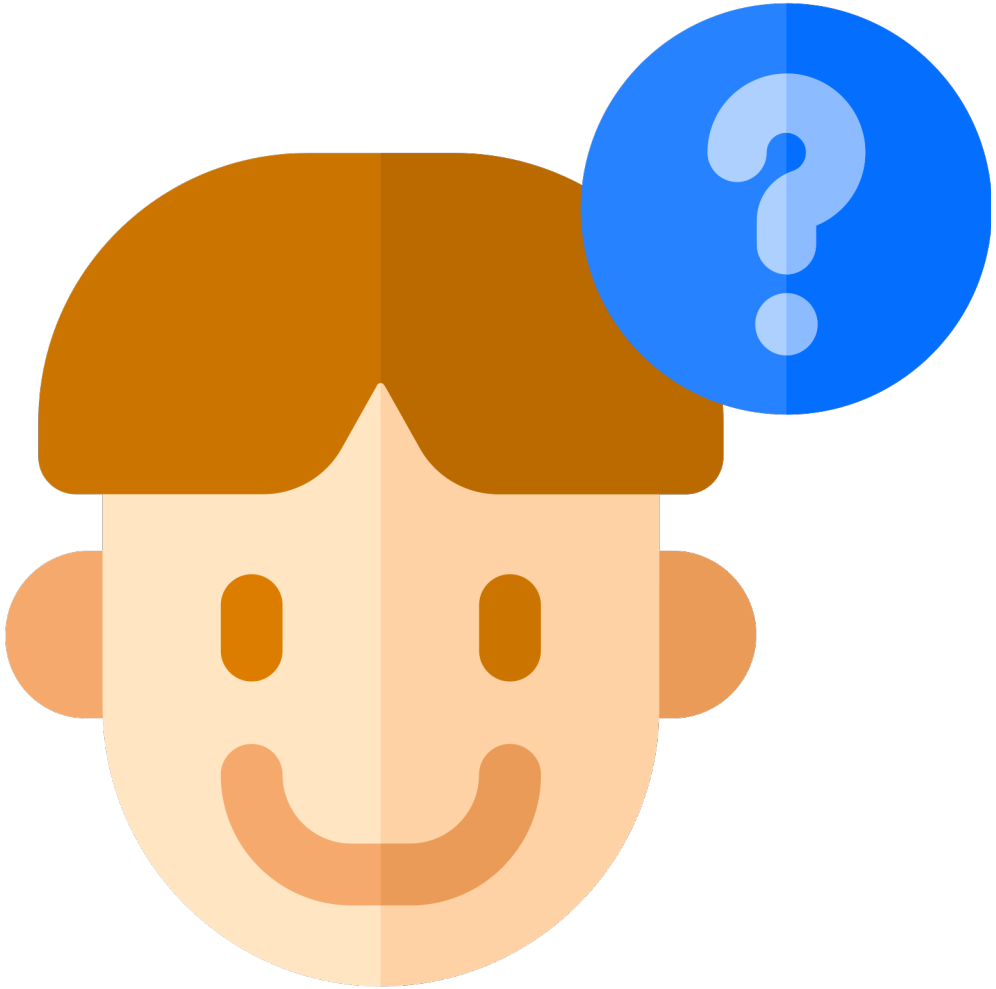
4 Fazit und Ausblick

1

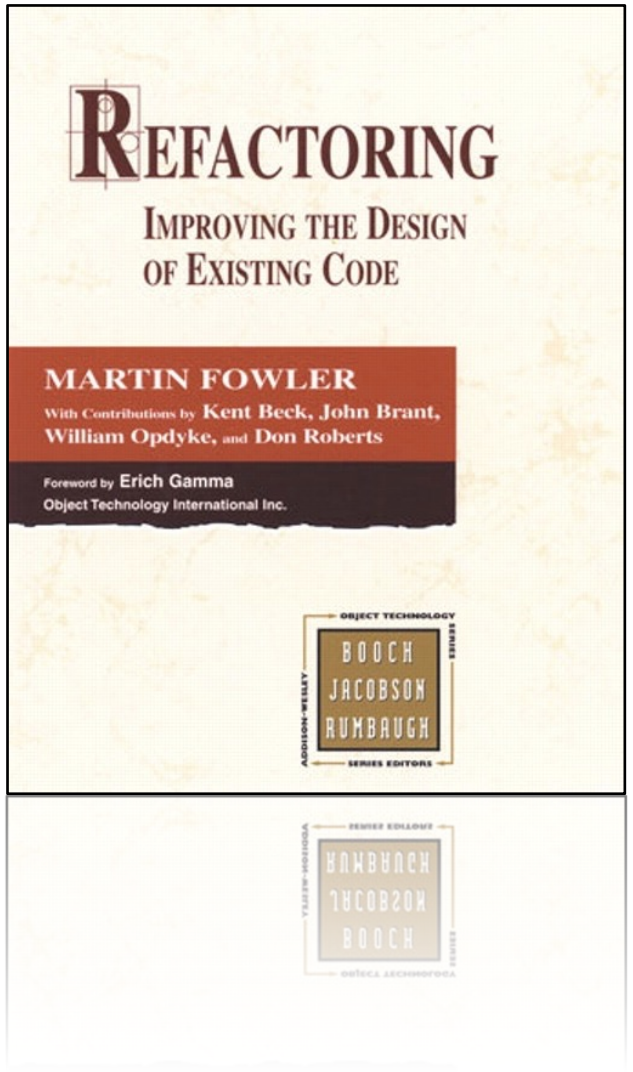


Somebody else's code

Was ist mit unserem eigenen Code?



Verstehen
Fehler beheben
Neues Feature
Optimierung



Annahmen

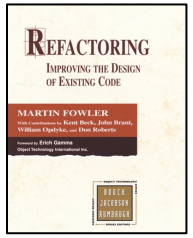
Es gibt automatisierte Tests …

Quellcode ist schon testbar …

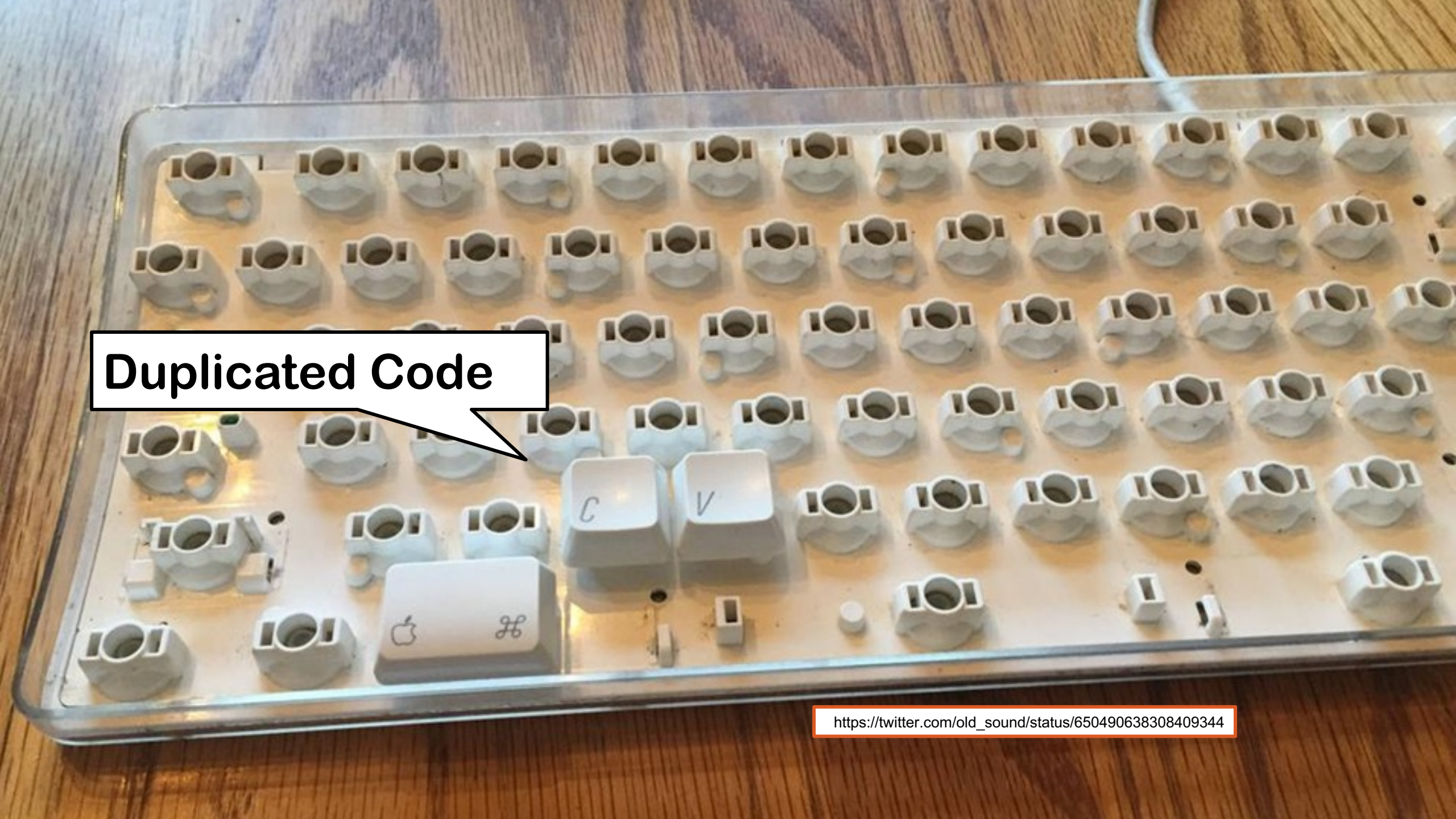
Code Smells

- Temporary Field
- Long Method
- Feature Envy
- ...

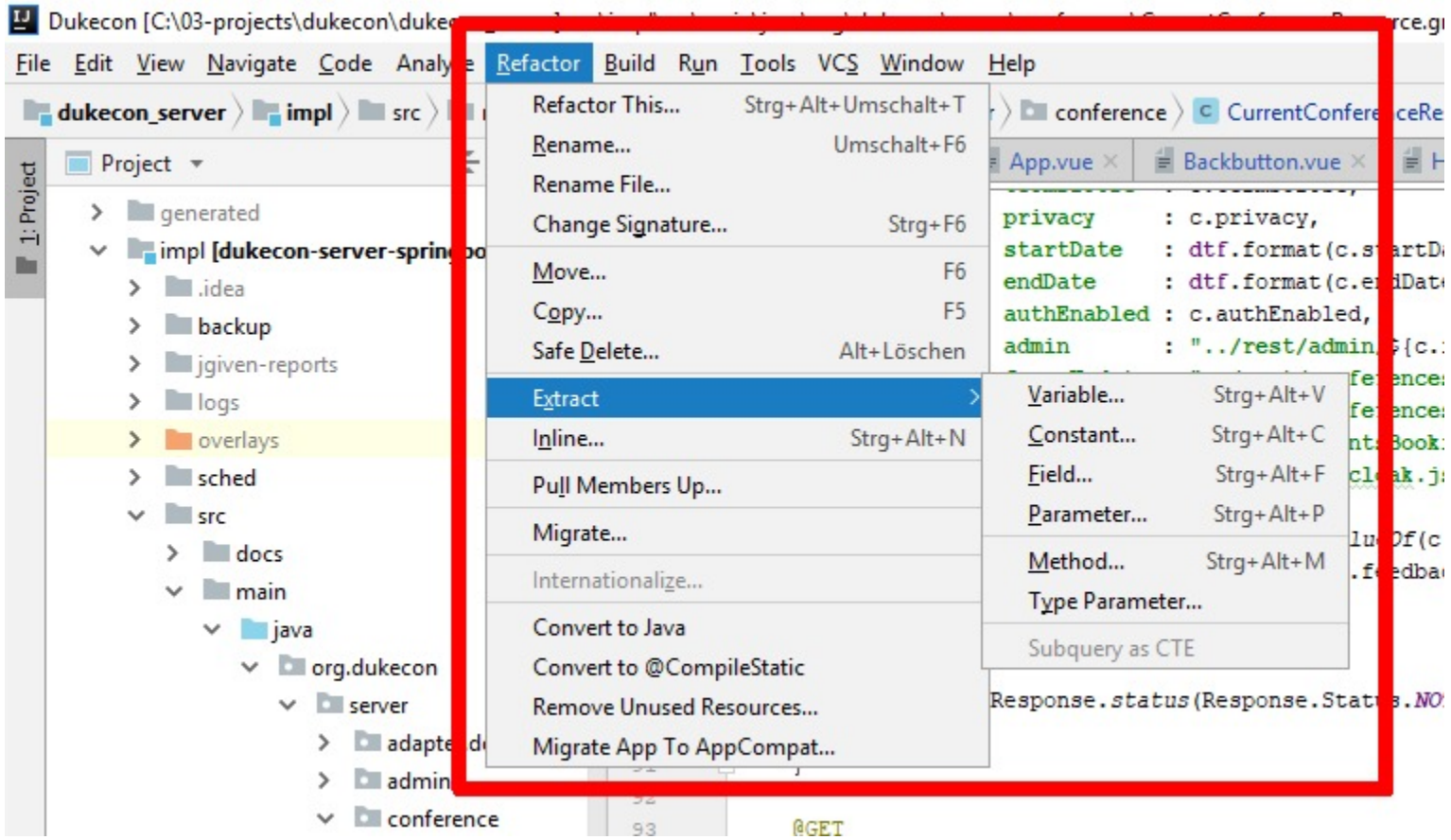
Code Comments

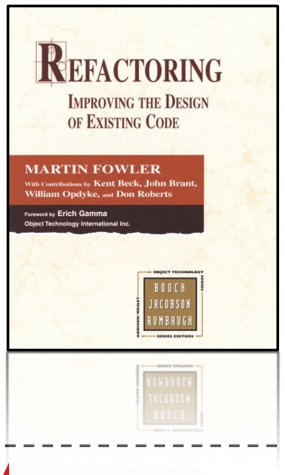


https://twitter.com/petecheslock/status/646507209413775360/photo/1


Duplicated Code
C
V
https://twitter.com/old_sound/status/650490638308409344

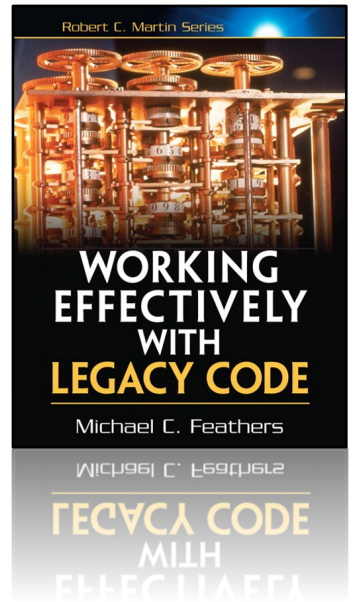
Toolunterstützung bei einfachen Refactorings





> “Code **without tests** is **bad code**.

Michael Feathers



J. B. Rainsberger

> “Legacy code is **valuable code** that we feel **afraid to change**.

Foto von PublicDomainPictures, CC0 Public Domain Lizenz, https://pixabay.com/de/menschen-abdeckung-schrei-314481/

Es ist egal, wie …

… gut geschrieben der Code ist

… schön der Code ist

… objektorientiert der Code ist

… entkoppelt der Code ist

Tests

lassen unser Verhalten schnell und verifizierbar ändern

Ohne Tests

wissen wir nicht, ob der Code besser oder schlechter wird

Die gute Nachricht …

Agenda

1 Was ist Legacy Code?

2 **Refactoring**

3 Die ‘x‘ Schritte

4 Fazit und Ausblick

2

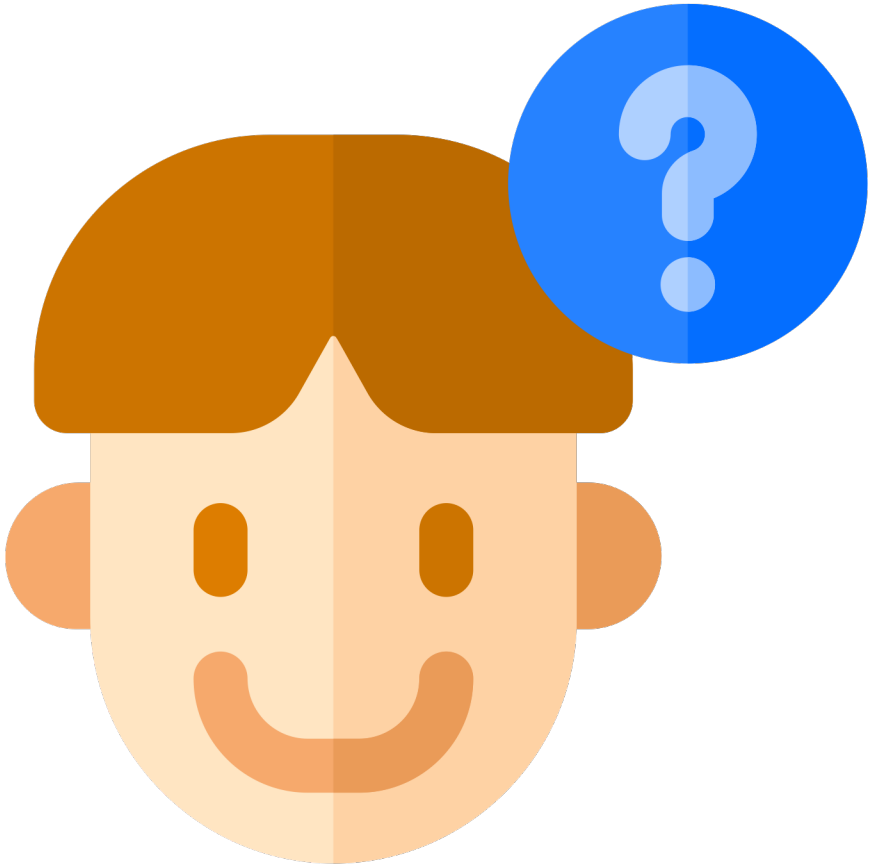
Sichern bestehender Investitionen in Software

Verhindert Desginverfall

Erhöhen der Lesbarkeit und Änderbarkeit

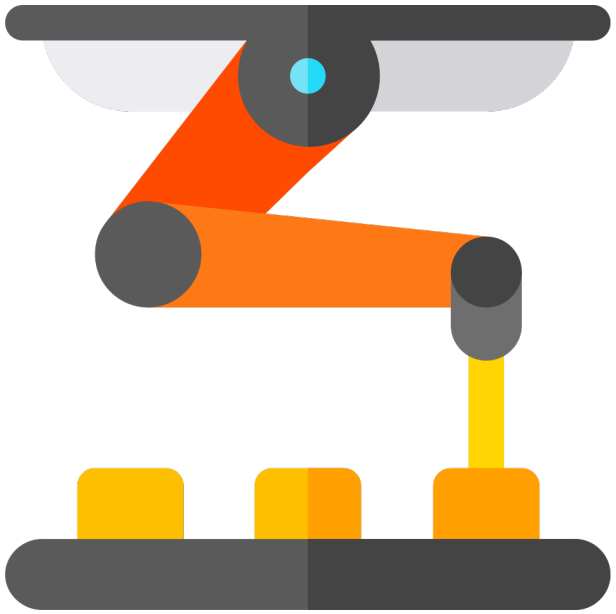
Finden von Fehlern

Wissenstransfer durch aktives Lernen am Code



Auch ein anderer Entwickler ist ein (oft vergessener) Nutzer

Don‘t break the program



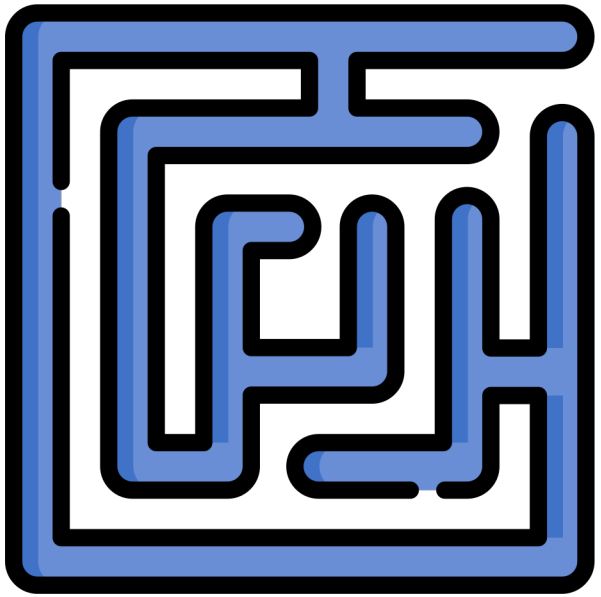
Monological thinking

Do a tiny little bit Compile and test Repeat

Hello World vs. 50.000++ LOC

Disziplin (kleine Schritte, …)

Aussagekräftige Testabdeckung



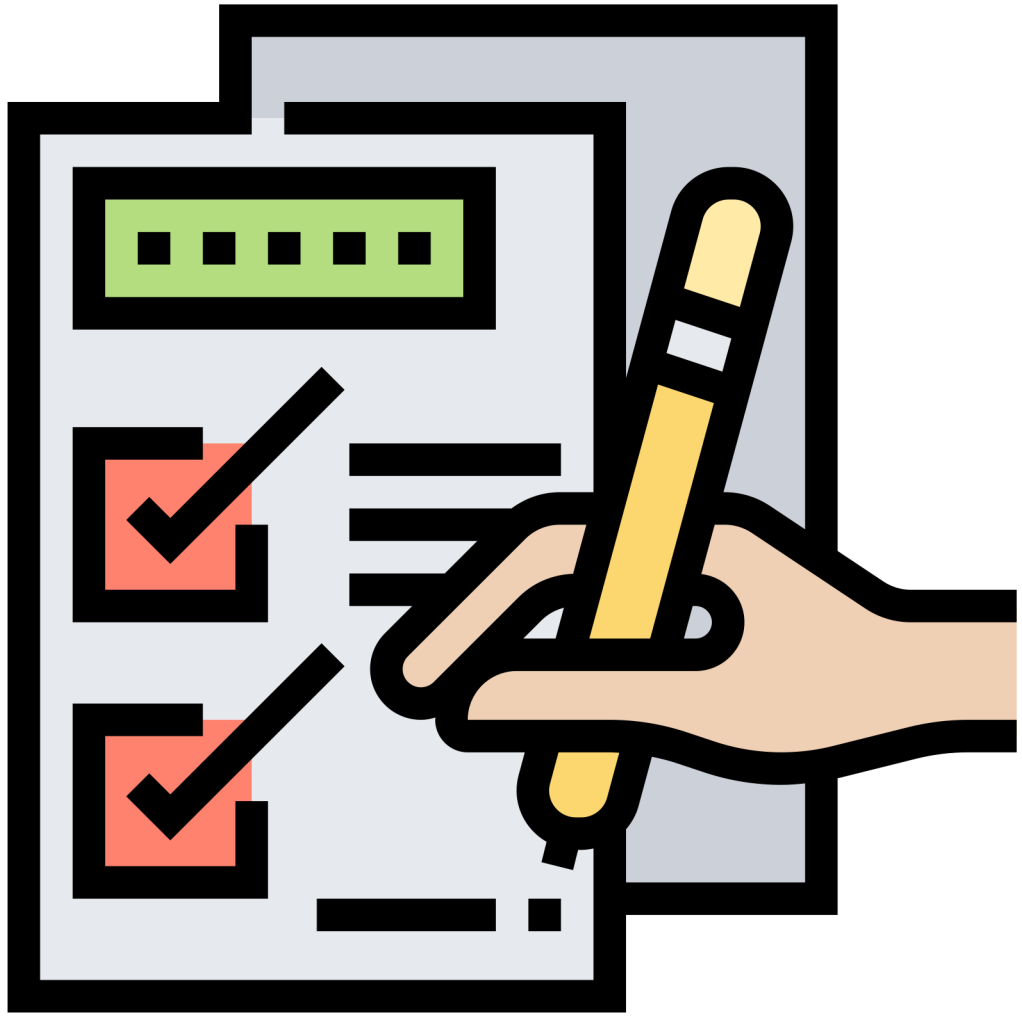
Clean Code ist NICHT das Ziel

Hauptfokus: testbarer Code

viel zu teuer

Code meist
kaum/nicht testbar

starke Kopplung,
geringe Kohäsion



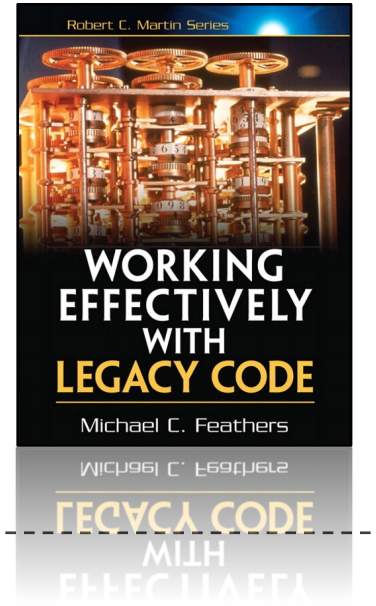
um Tests zu schreiben muß der Code testbar sein, um testbaren Code zu erhalten müßte man refactoren, um zu refactoren könnte man Tests schreiben, um Tests zu schreiben muß der Code testbar sein, um testbaren Code zu erhalten müßte man refactoren, um zu refactoren könnte man Tests schreiben, **um Tests zu schreiben muß der Code testbar sein, um testbaren Code zu erhalten müßte man refactoren, um zu refactoren könnte man Tests schreiben**, um Tests zu schreiben muß der Code testbar sein, um testbaren Code zu erhalten müßte man refactoren könnte man Tests schreiben, um Tests Code testbar sein, um testbaren Code zu erhalten, um zu refactoren könnte man Tests schreiben, um Tests zu schreiben muß der Code testbar sein, um testbaren Code zu erhalten müßte man refactoren, um zu refactoren könnte man Tests schreiben, um Tests zu schreiben muß der Code test

Die Katze beißt sich in den Schwanz!

um Code zu verstehen könnte man Code refactoren, um Code zu refactoren bräuchte man Tests, Tests würden helfen, Code zu verstehen, um Code zu verstehen könnte man Code refactoren, um Code zu refactoren bräuchte man Tests, Tests würden helfen, Code zu verstehen, um Code zu verstehen könnte man Code refactoren, um Code zu refactoren bräuchte man Tests, Tests würden helfen, Code zu verstehen, um Code zu verstehen könnte man Code refactoren, um Code zu refactoren bräuchte man Tests, Tests würden helfen, Code zu verstehen, um Code zu verstehen könnte man Code refactoren, um Code zu refactoren bräuchte man Tests, **Tests würden helfen, Code zu verstehen, um Code zu verstehen könnte man Code refactoren, um Code zu refactoren bräuchte man Tests,** Tests würden helfen, Code zu verstehen, um Code zu verstehen könnte man Code refactoren, um Code zu refactoren bräuchte man Tests, Tests würden helfen, Code zu verstehen, um Code zu verstehen könnte man Code refactoren, um Code

Vorgehen

1. Identify what to change
2. Identify what to test
3. Break dependencies
4. Write the tests
5. Modify and refactoring



Dann mal her mit den x einfachen Schritten!

Agenda

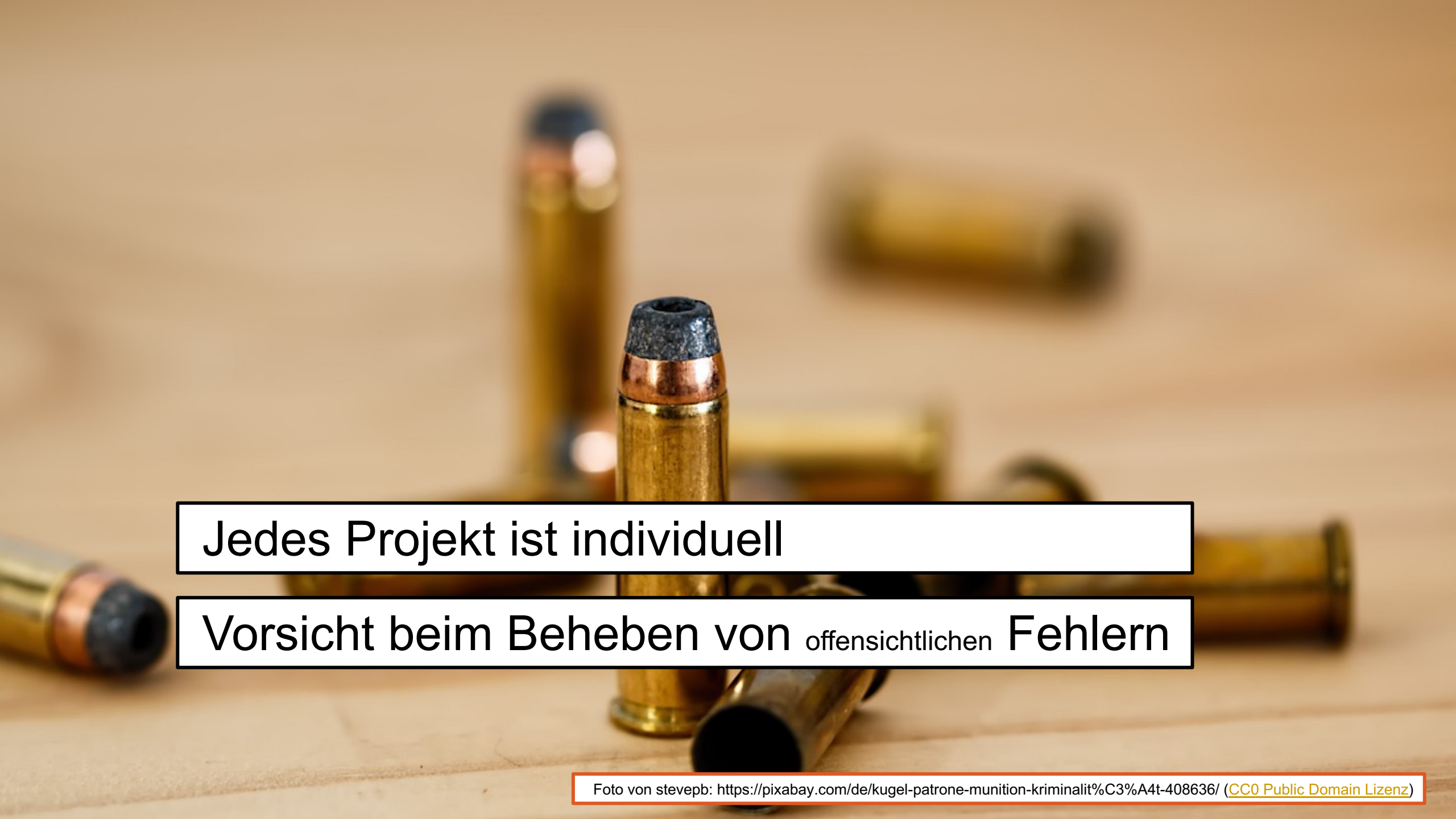
1 Was ist Legacy Code?

2 Refactoring

3 **Die ‘x‘ Schritte**

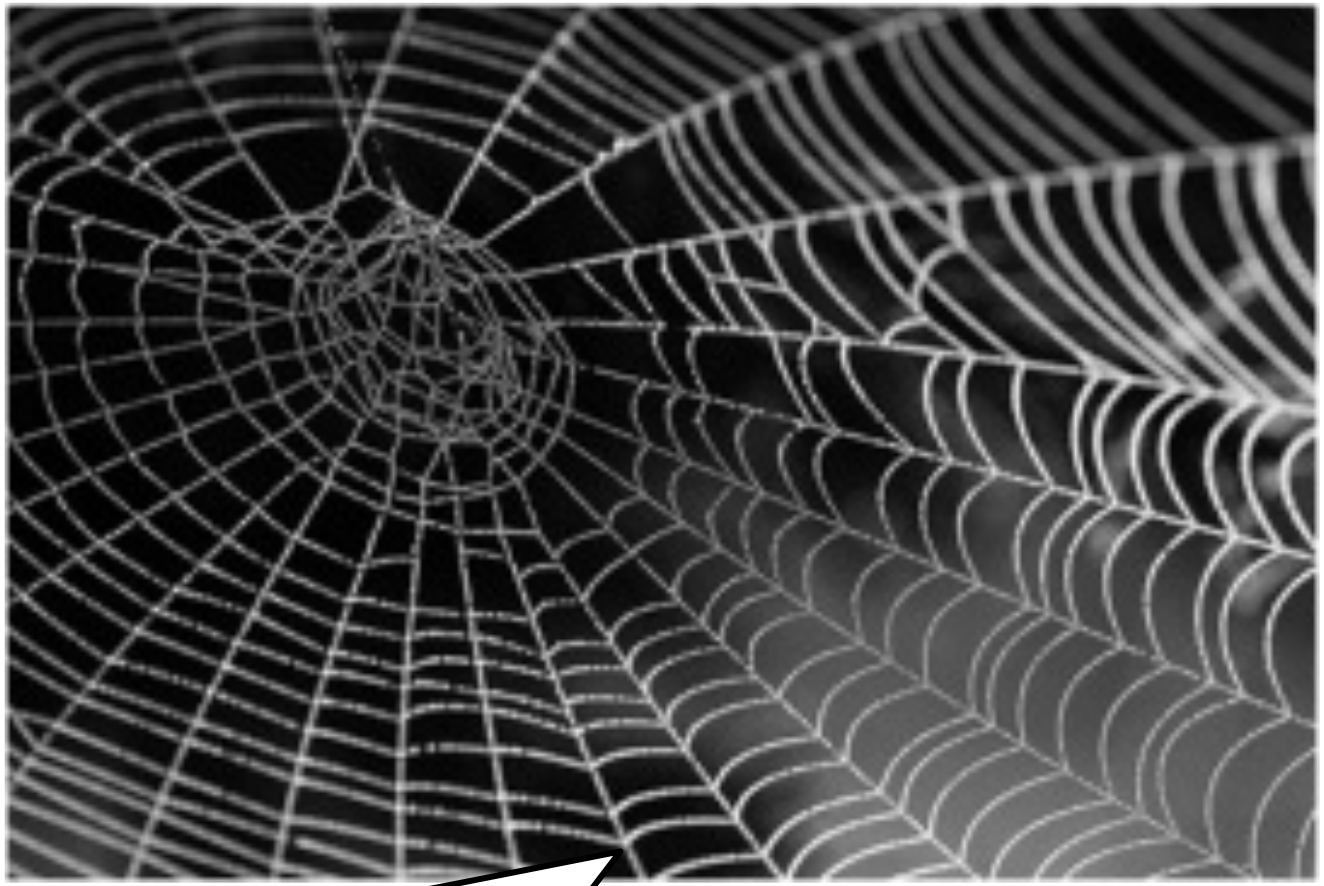
4 Fazit und Ausblick

3



Jedes Projekt ist individuell

Vorsicht beim Beheben von offensichtlichen Fehlern



Sicherheitsnetz + Tests

Foto von bella67: https://pixabay.com/de/spinnennetz-mit-wasserperlen-netz-921039/ (CC0 Public Domain Lizenz)

Sanierung

Foto von KlausHausmann: https://pixabay.com/de/bauarbeiter-bau-bauen-bohrhammer-921224/ (CC0 Public Domain Lizenz)

Einfach loslegen?

*Es **fehlen** automatisierte **Tests!***

*Machen wir auch nichts **kaputt?***

1 Golden Master

gegenwärtiges Verhalten dokumentieren und erhalten

```
# suppose that our legacy code is this program called 'game'
$ game > GOLDEN_MASTER

# after some changes we can check to see if behaviour has changed
$ game > OUT-01
$ diff GOLDEN_MASTER OUT-01
# GOLDEN_MASTER and OUT-01 are the same

# after some other changes we check again and...
$ game > OUT-02
$ diff GOLDEN_MASTER OUT-02
# GOLDEN_MASTER and OUT-02 are different -> behaviour changed
```

Golden Master

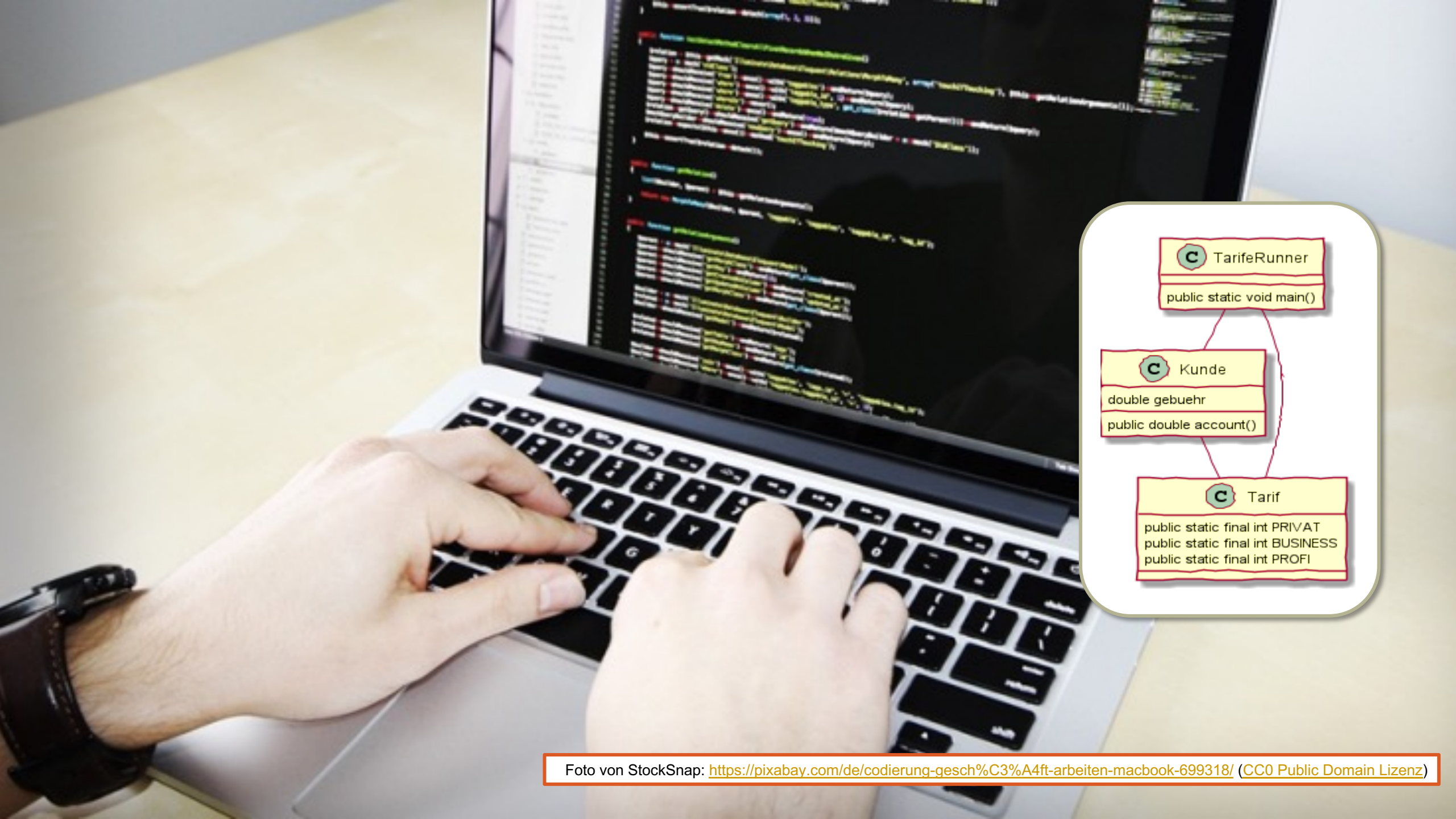
(aka characterization tests)

Konsolenausgaben abprüfen

```
@Before
public void init() {
    originalSysOut = System.out;
    consoleStream = new ByteArrayOutputStream();
    PrintStream printStream = new PrintStream(consoleStream);
    System.setOut(printStream);
}

@Test
public void testSimpleOutput() {
    System.out.println("Hallo Publikum!");
    System.out.print("Hallo Falk!");
    assertEquals("Hallo Publikum!\r\nHallo Falk!", consoleStream.toString());
}

@After
public void teardown() {
    System.setOut(originalSysOut);
}
```



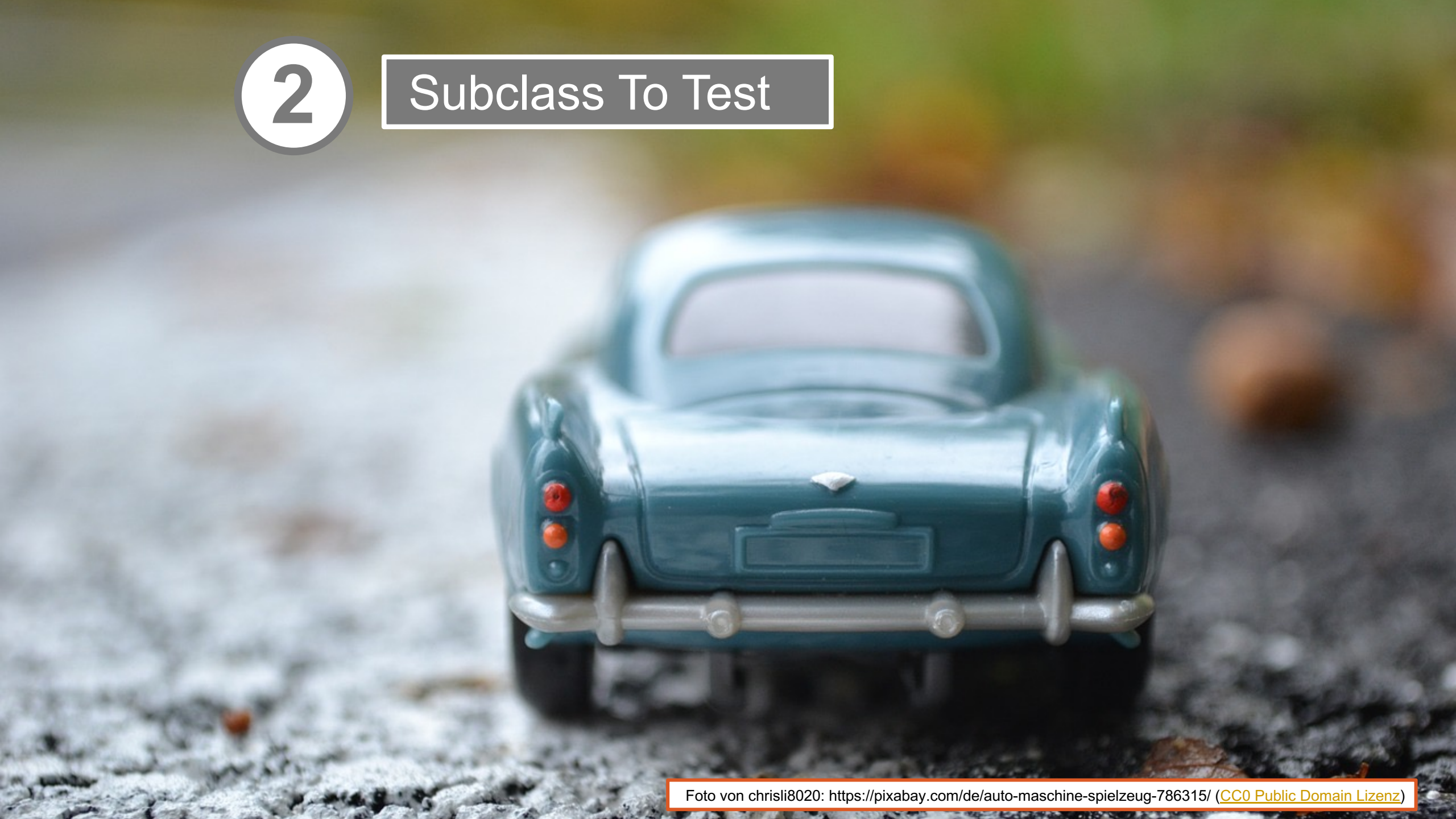
Golden Master

Vorsicht bei Zufallsgeneratoren

```
1.9.3 > g = Random.new
1.9.3 > (1..10).map{g.rand(1000)}
 => [691, 362, 997, 692, 236, 532, 687, 616, 218, 702]
1.9.3 > g = Random.new
1.9.3 > (1..10).map{g.rand(1000)}
 => [865, 186, 89, 382, 894, 708, 769, 850, 452, 85]
1.9.3 > g = Random.new(1)
1.9.3 > (1..10).map{g.rand(1000)}
 => [37, 235, 908, 72, 767, 905, 715, 645, 847, 960]
1.9.3 > g = Random.new(1)
1.9.3 > (1..10).map{g.rand(1000)}
 => [37, 235, 908, 72, 767, 905, 715, 645, 847, 960]
```

Festlegen von Seeds
(Pseudo-Random)

2 Subclass To Test



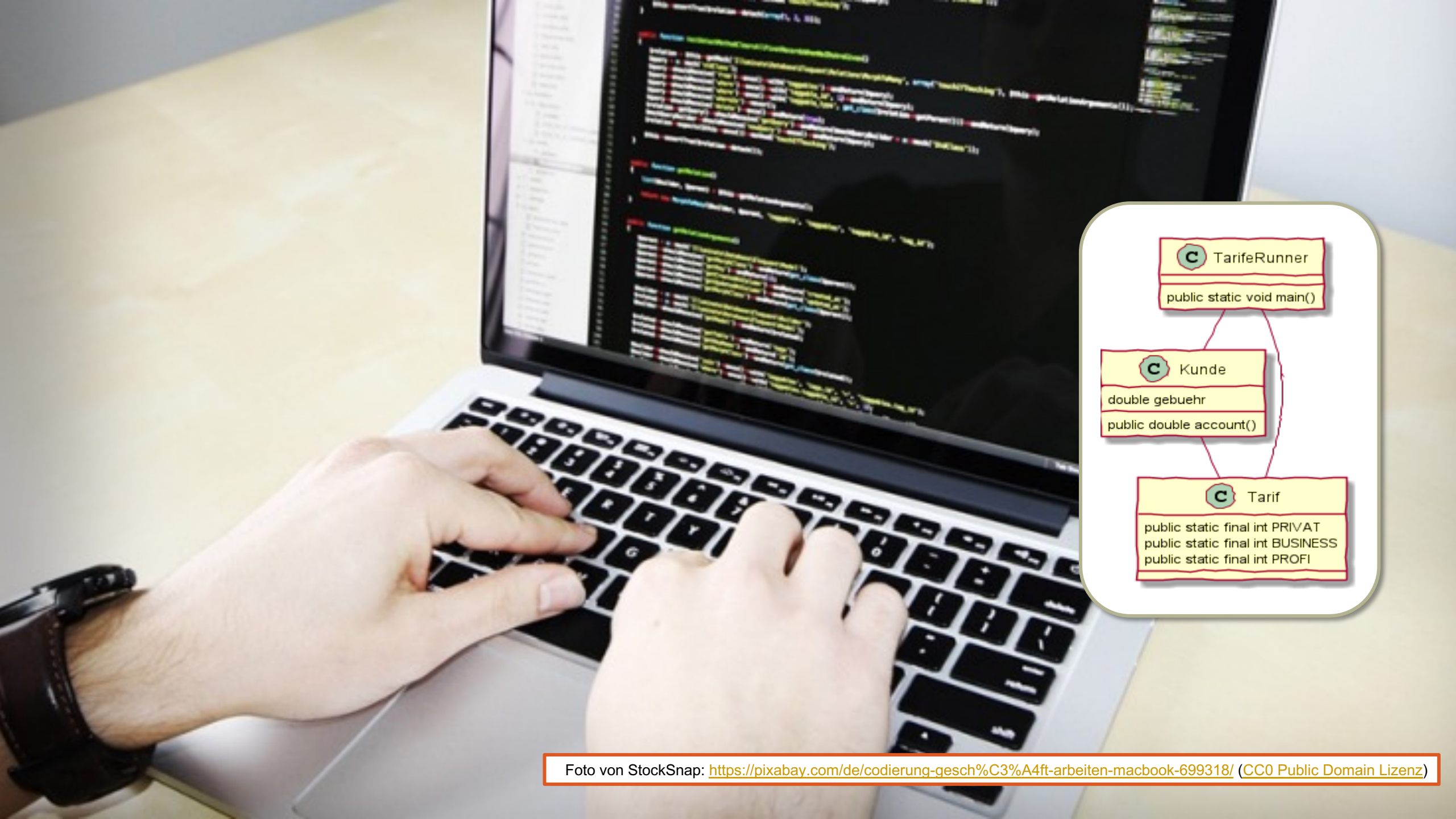
Subclass To Test

Seam (Nahtstelle)

Ein Seam ist eine Stelle, an der man das Verhalten editieren kann, ohne direkt an dieser Stelle zu ändern.

Aufbrechen stark gekoppelter Abhängigkeiten

aka Extract and Override



3 Extract Pure Functions

3 Extract Pure Functions

seiteneffektfrei

keine Statusänderung

"It's a classic,
we call it a Klassiker"

3 Extract Pure Functions

```
"pure function".substring(5);
```

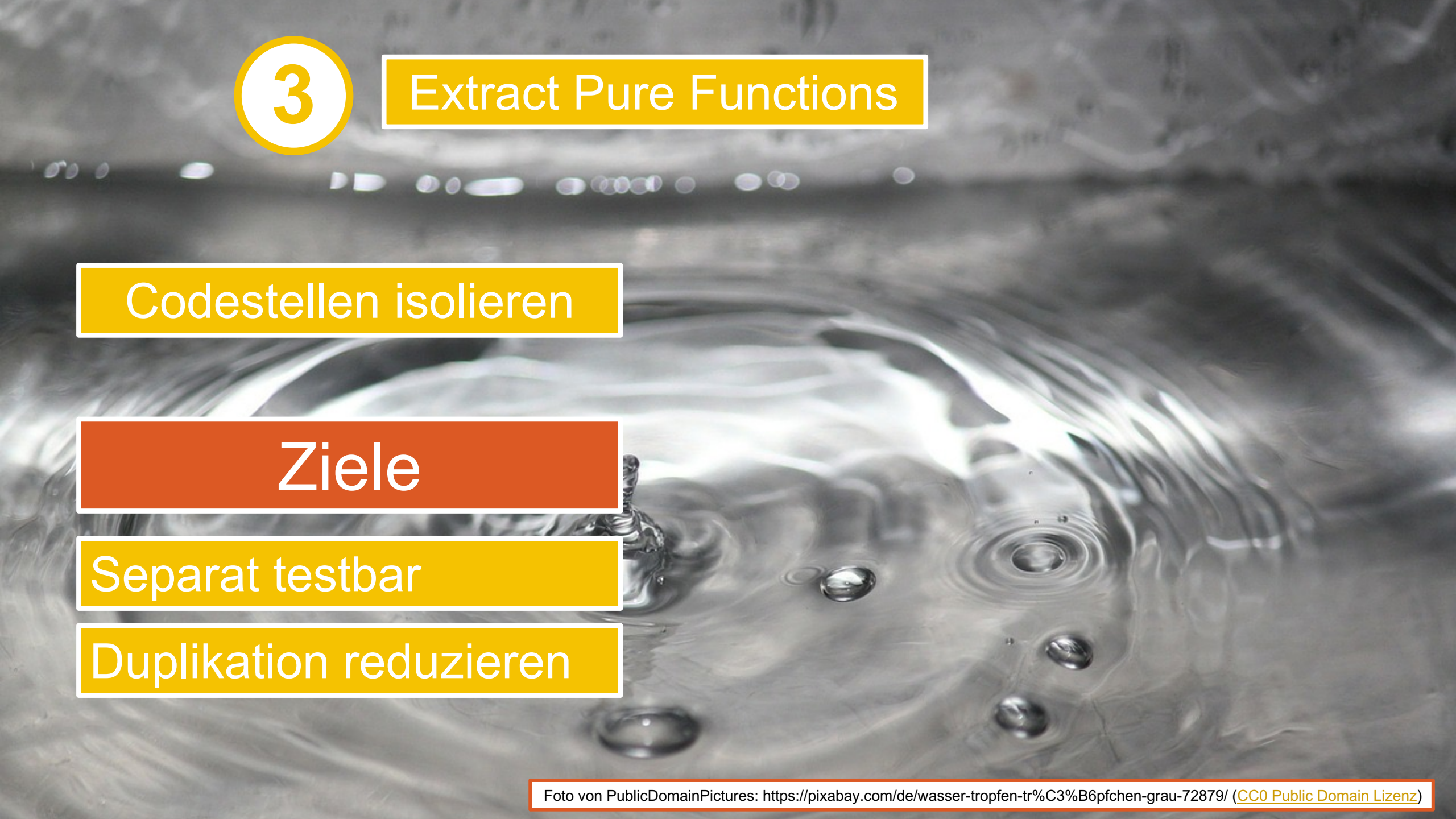
```
UrlEncoder.encode("pure function");
```

```
Math.max(x, y);
```

```
System.out.println("unpure");
```

```
list.add(3);
```

```
Collections.sort(list);
```



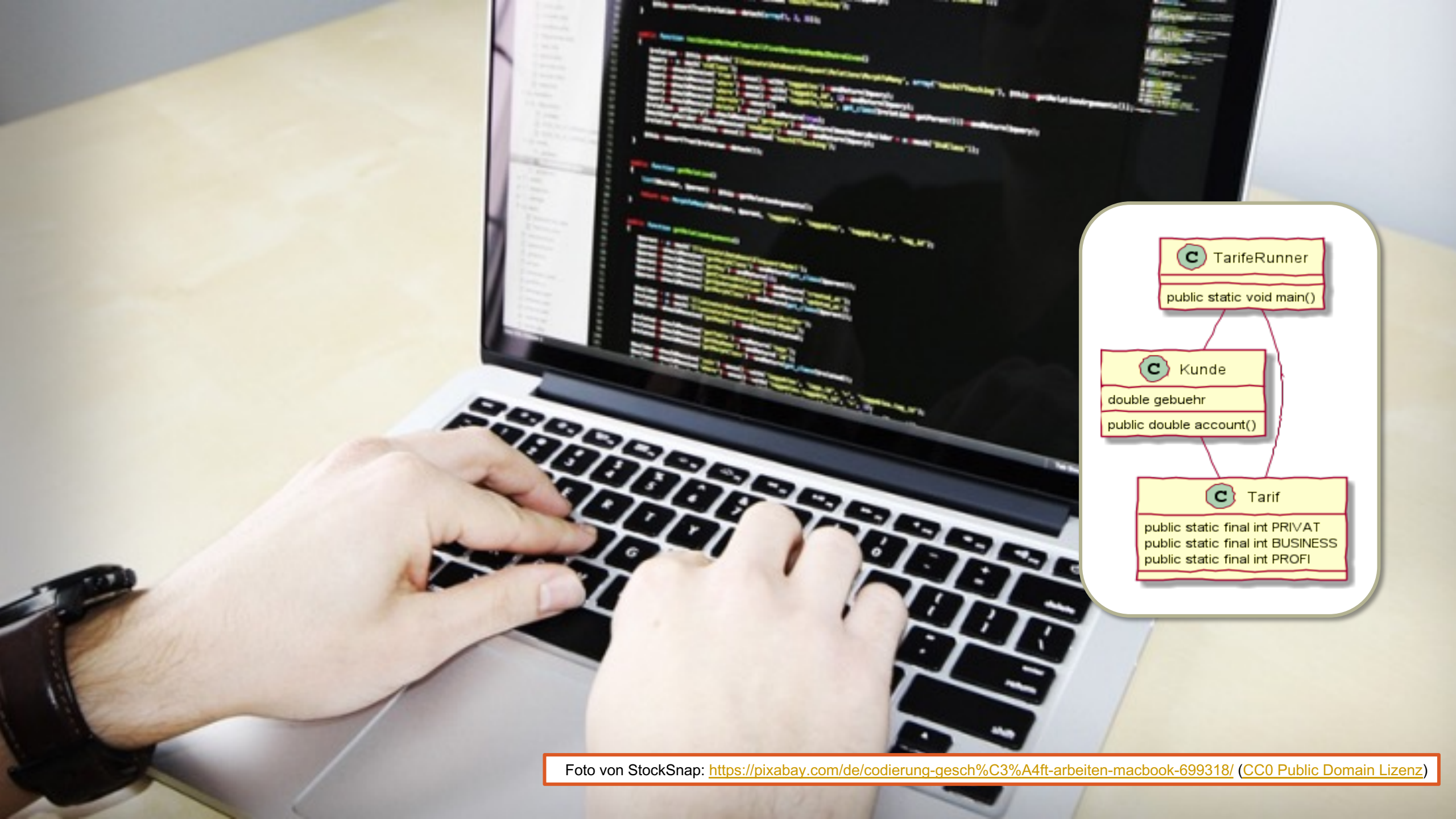
3 Extract Pure Functions

Codestellen isolieren

Ziele

Separat testbar

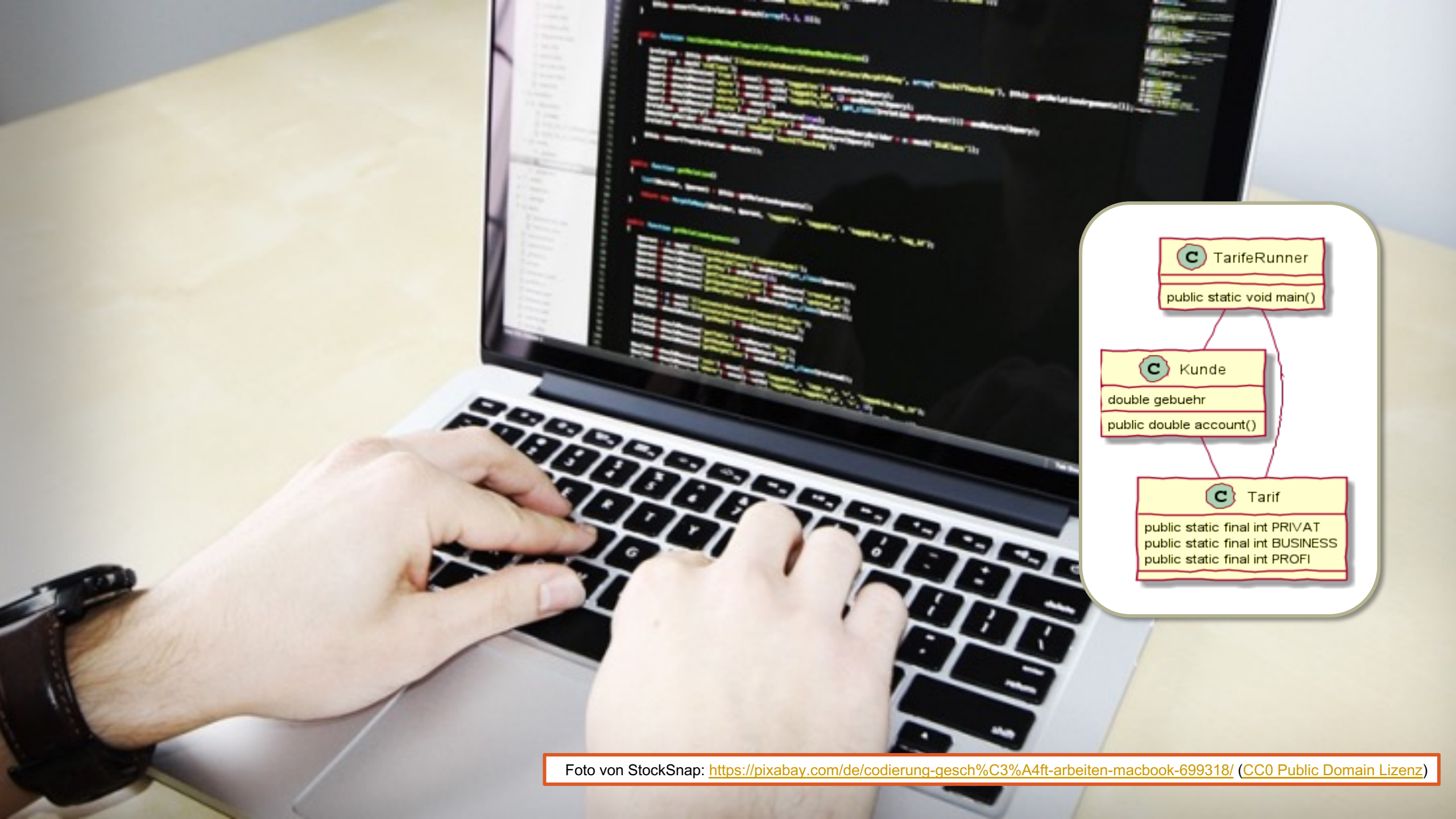
Duplikation reduzieren



4 Remove Duplication

Don't **R**epeat **Y**ourself

Beachte: Rule of Three



5 Extract Class

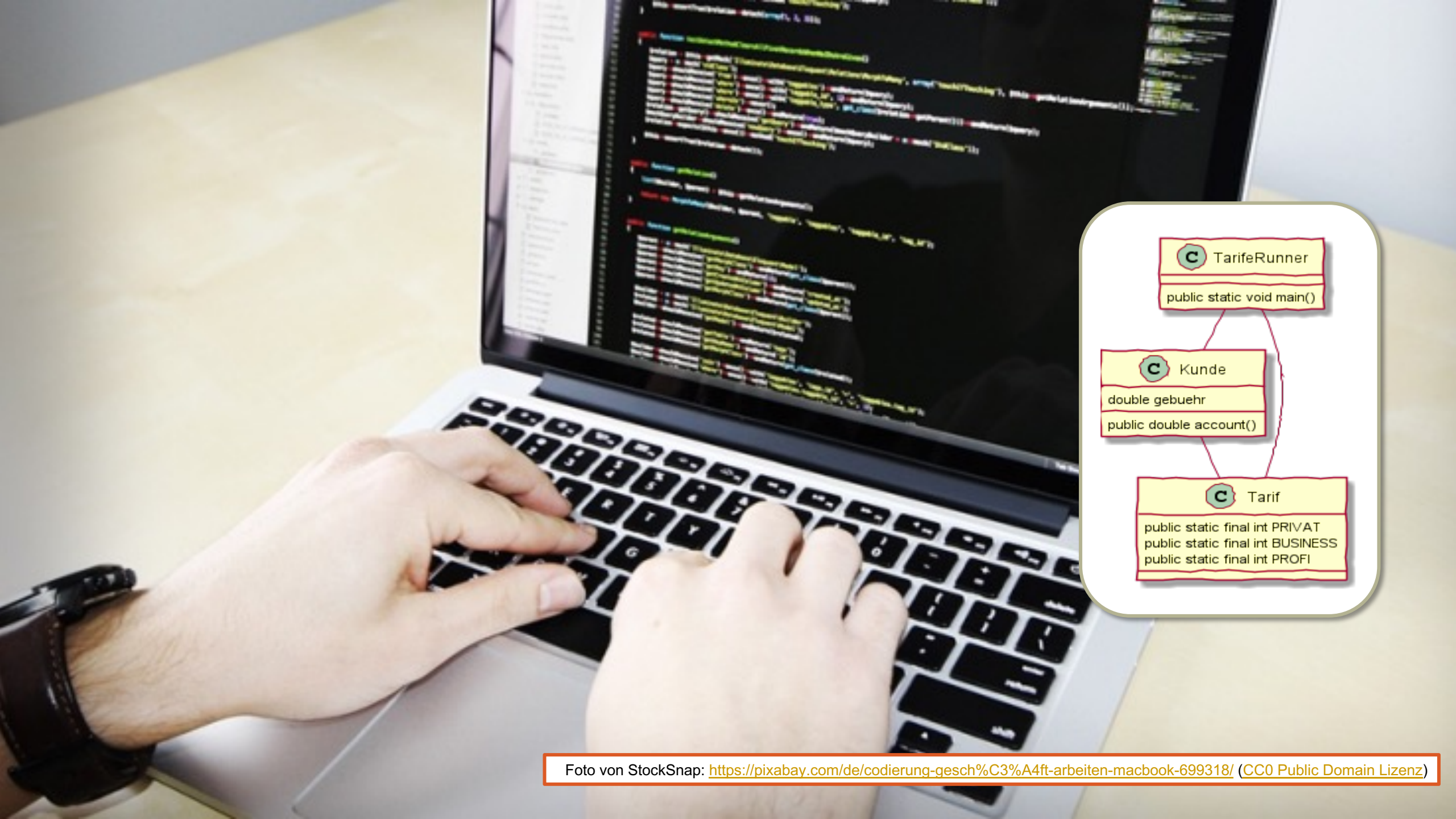
Large class

SRP verletzt

Ziele

Unabhängiges Testen einzelner Teile

Sauberes OO-Design



~~Hätten wir mehr Zeit ...~~

6 Dependency Inversion

7 Test non-public member

8 Mocking Framework

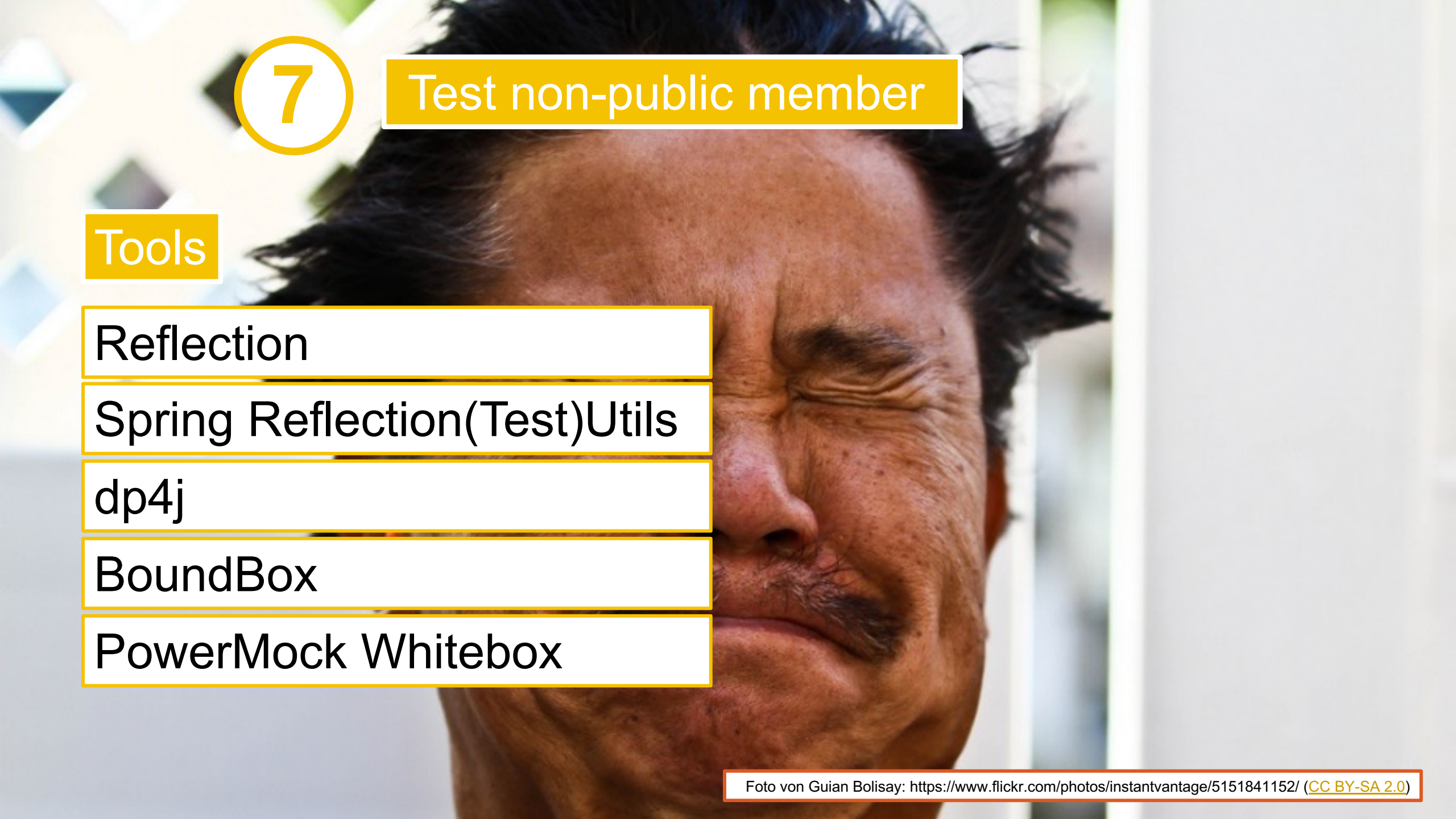
6 Dependency Inversion

Entkoppeln
durch **explizites**
Setzen der
Abhängigkeiten

7 Test non-public member

Tools

- Reflection
- Spring Reflection(Test)Utils
- dp4j
- BoundBox
- PowerMock Whitebox



8 Mocking

Subclass To Test on Steorids!

Interaktion mit Umgebung testen

Erwartete Parameter und Aufrufreihenfolge

Agenda

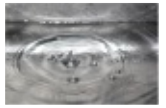
1. Was ist Legacy Code?
2. Refactoring
3. Die ‘x‘ Schritte
4. **Fazit und Ausblick**

4

1. Golden Master

2. Subclass To Test

3. Extract Pure Functions



4. Remove Duplication

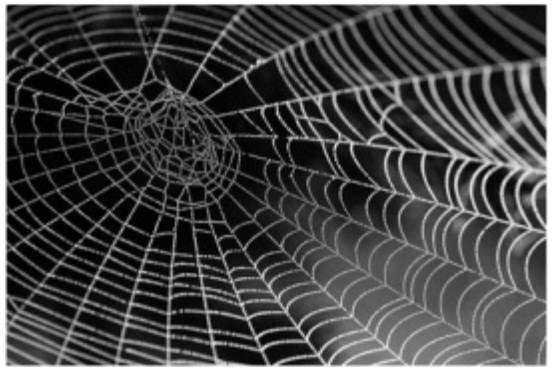
5. Extract Class

6. Dependency Inversion

7. Test non-public member

8. Mocking Framework

Sicherheitsnetz



1. Golden Master

1

2

3

```
public void account(int minuten, int stunde, int minute) {
    System.out.println(String.format("Berechne Gespräch mit
    boolean mondschein = false;
    double preis = 0;
    this.
    // Mon
    if (st
        mo
    // Ges
```

gebuehr : double - Kunde
tarif : Tarif - Kunde
account(int minuten, int stunde, int minute) : void - Kunde
berechnePreis(int minuten, double d) : double - Kunde

retest

recheck

```
public static void main(String... args) throws Except
  WebDriver driver = new FirefoxDriver();
  driver.get("http://www.retest.de");
  while (true) {
    List<WebElement> links = driver.findElements(By.tagName("a"));
    links.get(random.nextInt(links.size())).click();
    Thread.sleep(500);
    List<WebElement> fields =
        driver.findElements(By.xpath("//input[@type='text']"));
    WebElement field = fields.get(random.nextInt(fields.size()));
    field.sendKeys(randomString());
    Thread.sleep(500);
  }
}
```

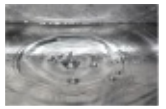
4

https://entwicklertag.de/frankfurt/2016/sites/entwicklertag.de.frankfurt.2016/files/slides/Bei%20uns%20testen%20lauter%20Affen_0.pdf

Leichtere Testbarkeit

2. Subclass To Test

3. Extract Pure Functions



5. Extract Class

6. Dependency Inversion

7. Test non-public member

8. Mocking Framework

Besseres Verständnis

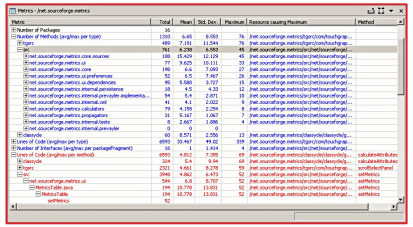
3. Extract Pure Functions

4. Remove Duplication

5. Extract Class

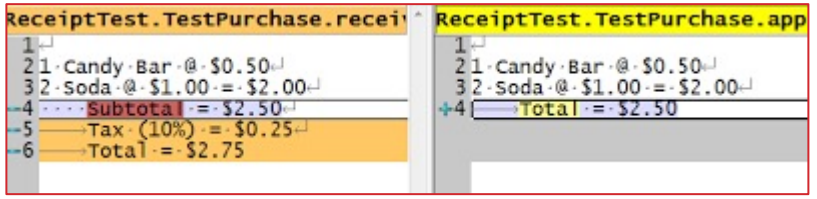
Tools

Eclipse Metrics

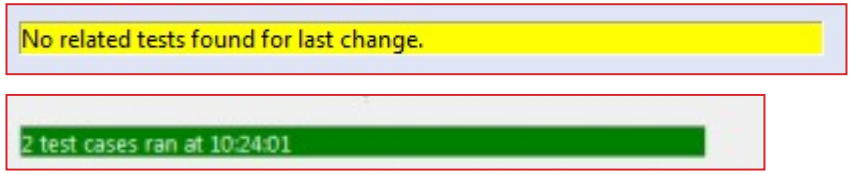


MoreUnit

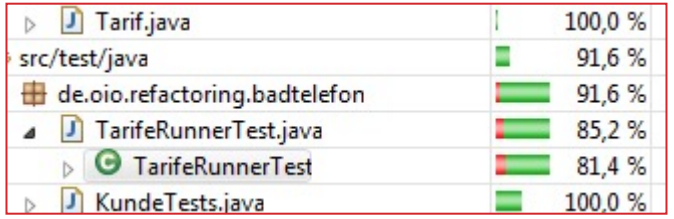
Approval Tests



Infinitest



Code Coverage



Legacy Code Retreat

Links

Code-Beispiel der Live-Demo

- https://github.com/sippsack/BadTelefon-Refactoring-Legacy-Code

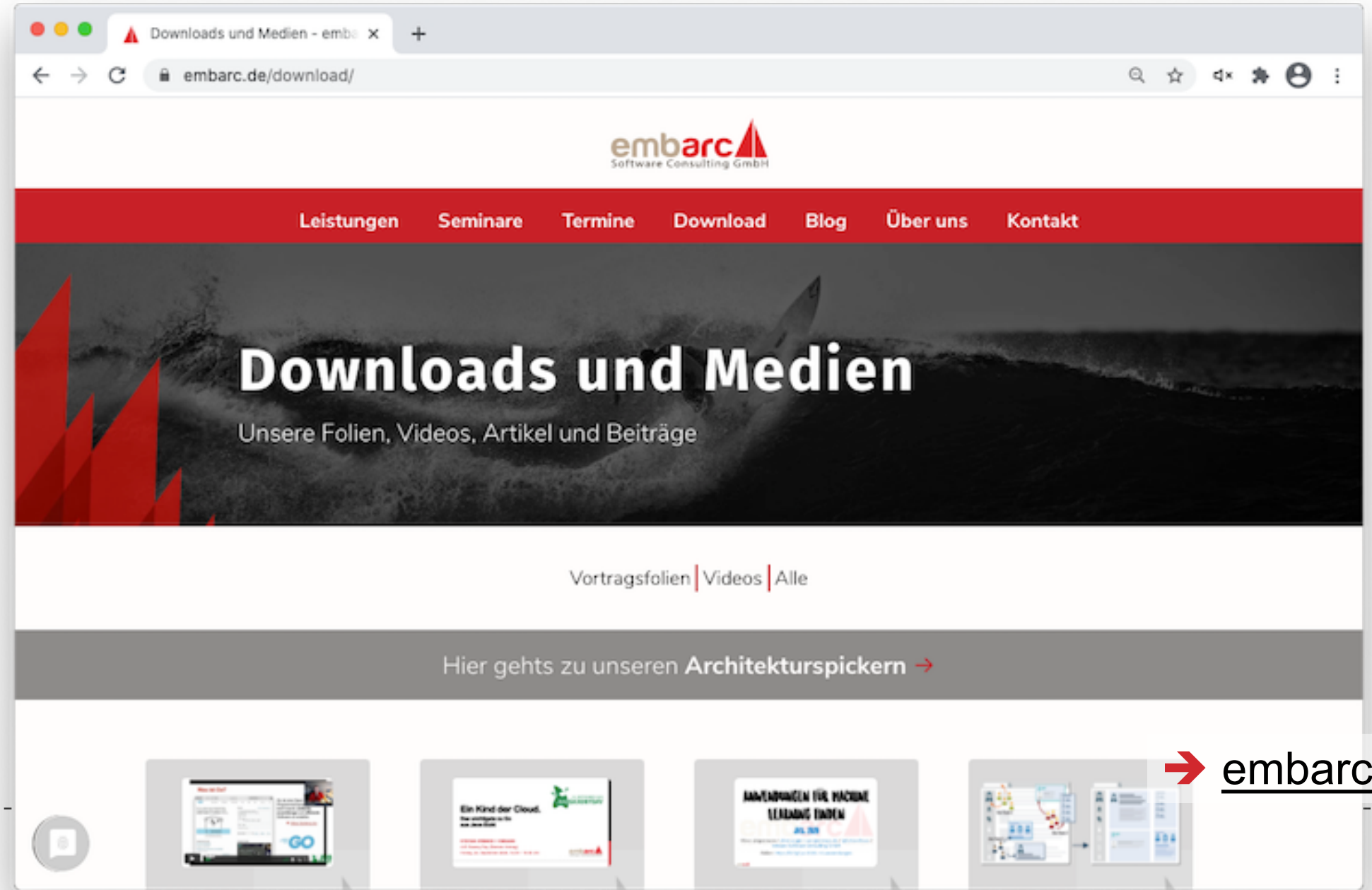
anderes Code-Beispiel für Legacy Code

- https://github.com/jbrains/trivia

Blog: Techniken zu Legacy Code-Retreat

- http://blog.adrianbolboaca.ro/2014/04/legacy-coderetreat/

Folien und Quellcode zum Download



→ embarc.de/download/

Vielen Dank.

Ich freue mich auf Eure Fragen!

Falk Sippach

fs@embarc.de

@sippsack

➔ xing.to/fsi