Orientation in Objects

Legacy Code meistern in x einfachen Schritten

Orientation in Objects GmbH

Weinheimer Str. 68
68309 Mannheim

www.oio.de
info@oio.de

JUG Saxony Leipzig, 26.10.2016

Version: 1.0

Abstract

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, sodass 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.

Über mich

Orientation in Objects

Falk Sippach (@sippsack)

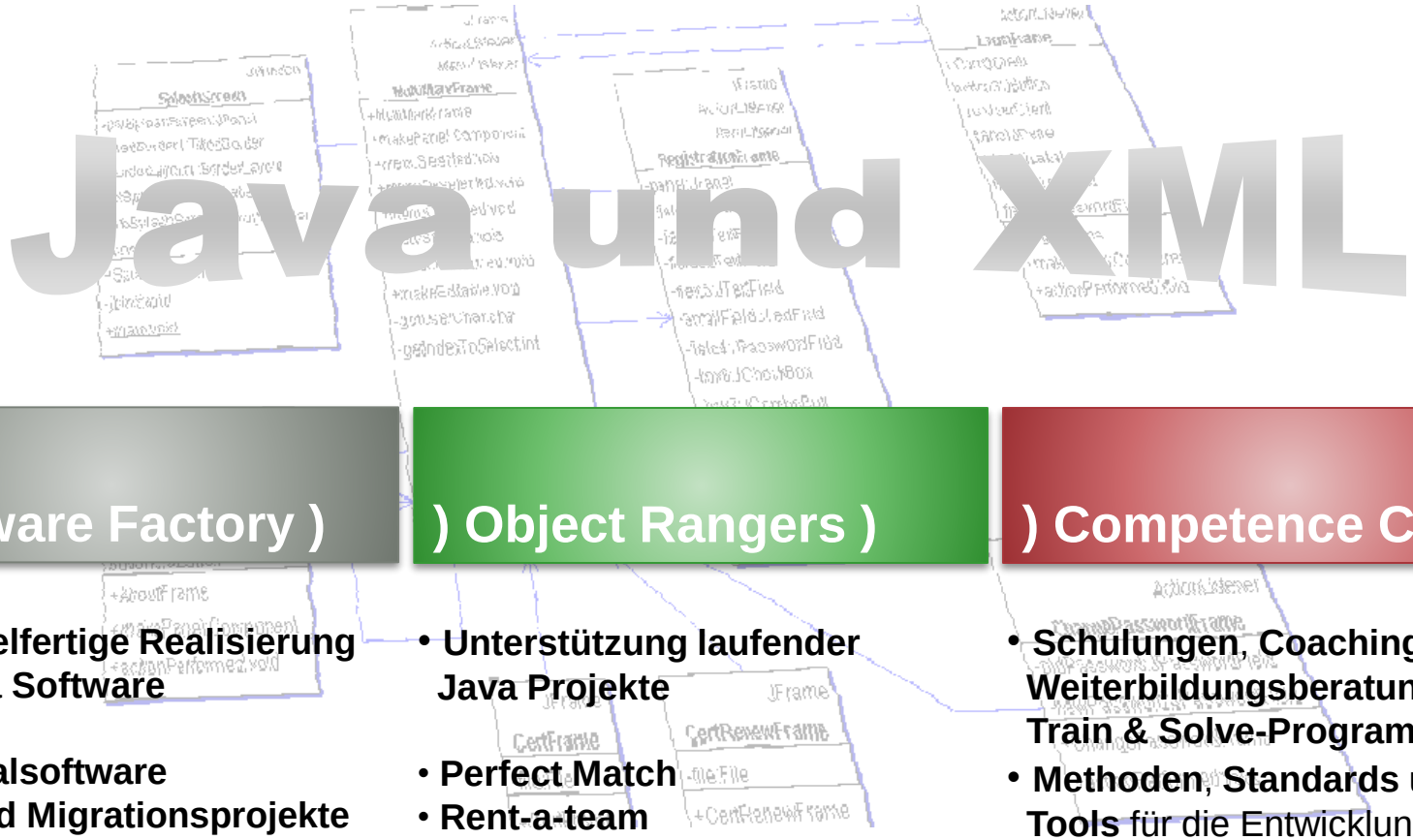
Trainer, Berater, Entwickler

Schwerpunkte
Architektur
Agile Softwareentwicklung
Codequalität

Co-Organisator

Java, XML und Open Source seit 1998

Orientation in Objects



) Software Factory)

- **Schlüsselfertige Realisierung von Java Software**
- **Individualsoftware**
- **Pilot- und Migrationsprojekte**
- **Sanierung von Software**
- **Software Wartung**

) Object Rangers)

- **Unterstützung laufender Java Projekte**
- **Perfect Match**
- **Rent-a-team**
- **Coaching on the project**
- **Inhouse Outsourcing**

) Competence Center)

- **Schulungen, Coaching, Weiterbildungsberatung, Train & Solve-Programme**
- **Methoden, Standards** und **Tools** für die Entwicklung von **offenen, unternehmens-weiten Systemen**

Orientation in Objects

Legacy Code meistern in

8 einfachen Schritten

**Nur heute,
nicht 1,
nicht 2,
…**

Orientation in Objects

Unser Thema heute:

Refactoring Legacy Code

Disclaimer: KEIN Projekterfahrungsbericht

Foto von smpcas, CC0 Public Domain Lizenz, https://pixabay.com/de/pula-kroatien-amphitheater-erbe-827909/

Legacy Code

Orientation in Objects

Somebody else's code

Was ist mit unserem eigenen Code?

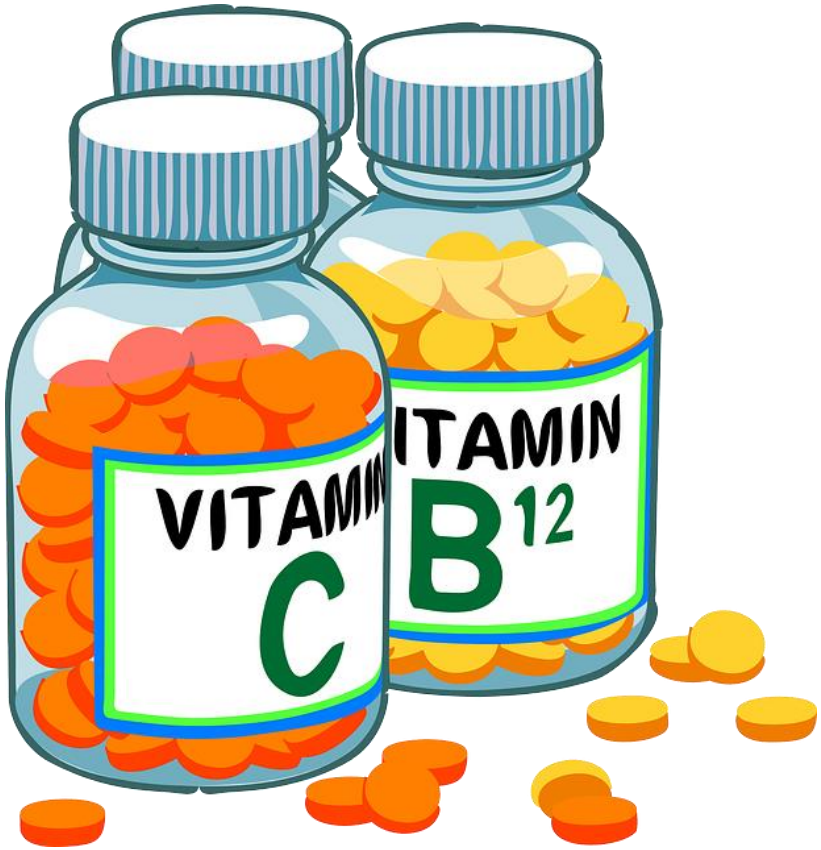
- Jeder kennt ihn …
- Keiner mag ihn …

Refactoring Legacy Code

Orientation in Objects

Warum?

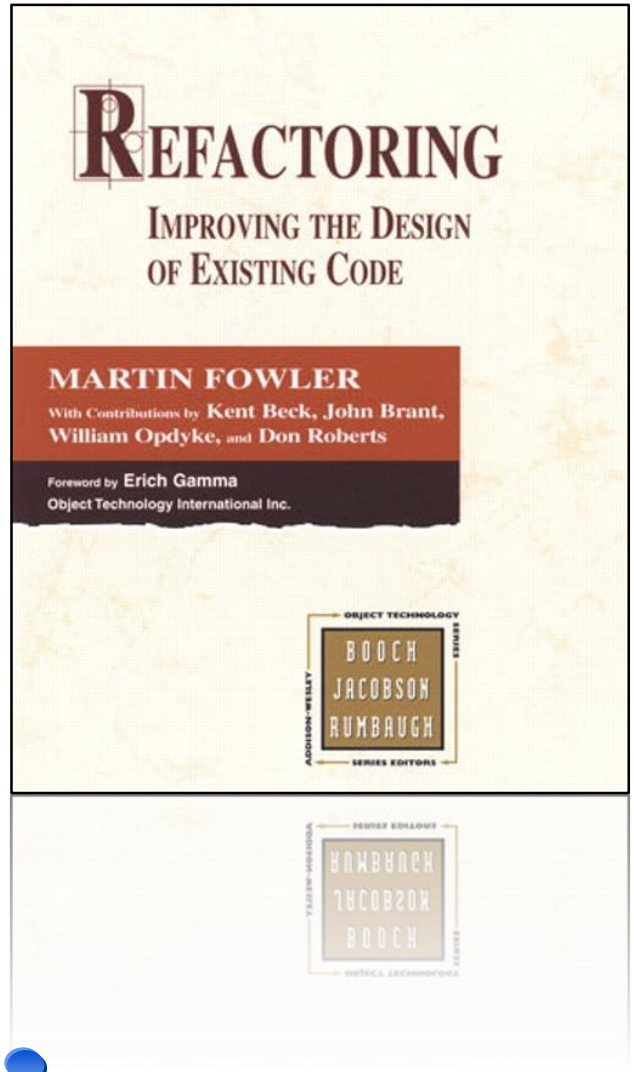
- Verstehen
- Erweitern
- Bugfixing
- Optimierung



Grafik von ClkerFreeVectorImages: https://pixabay.com/de/vitamine-tabletten-pillen-medizin-26622/ (CC0 Public Domain Lizenz)

Refactoring ~~Legacy~~ Code

Orientation in Objects



Annahmen

Es gibt automatisierte Tests …

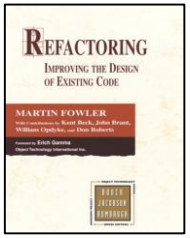
Quellcode ist schon testbar …

Refactoring ~~Legacy~~ Code

Orientation in Objects

Code Smells 

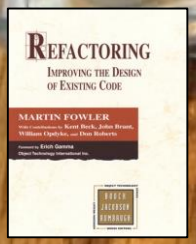
- Temporary Field
- Long Method
- Feature Envy
- ...
- Code Comments



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

"Gimme Google, Stack Overflow, and this keyboard, and I'll program you anything. "

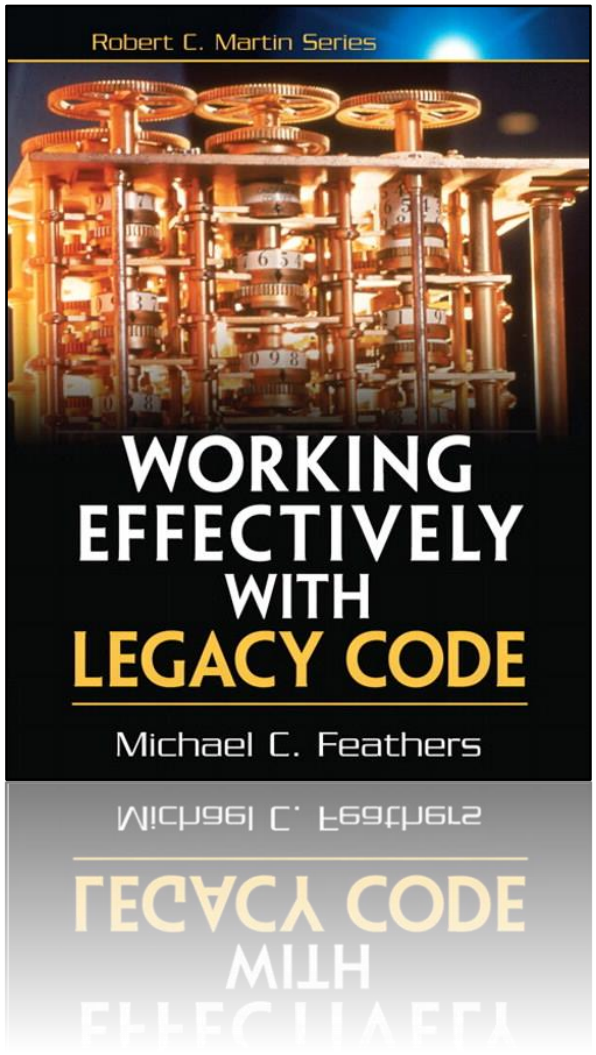
Duplicated Code



https://twitter.com/old_sound/status/650490638308409344

Refactoring Legacy Code

Orientation in Objects



“Code without tests

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

Michael Feathers

Orientation in Objects

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 …

Orientation in Objects

… 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 …

Keine Wissenschaft

Gesundes Menschenverstand

Foto von Engel62: https://pixabay.com/de/daniel-d%C3%BCsentrieb-helferlein-123206/ (CC0 Public Domain Lizenz)

Was macht es dann schwierig?

Orientation in Objects

- Hello World vs. 50.000++ LOC
- Disziplin (kleine Schritte, …)
- Aussagekräftige Testabdeckung

Legacy zu Clean Code?

Orientation in Objects

Clean Code ist NICHT das Ziel

Hauptfokus: testbarer Code

Dann schreiben wir halt Tests …

Orientation in Objects

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

Die Katze beißt sich in den Schwanz!

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 ve n könnte man Code refactoren, um C Tests, Tests würden helfen, Code zu ve n könnte man Code refactoren, um C sts, Tests würden helfen, Code zu ve nte 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

Vorgehen

Orientation in Objects

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

No Silver Bullet

Jedes Projekt individuell

Vorsicht beim Beheben von offensichtlichen Fehlern

Foto von stevepb: https://pixabay.com/de/kugel-patrone-munition-kriminalit%C3%A4t-408636/ (CC0 Public Domain Lizenz)

Orientation in Objects

Dann mal her mit den

X einfachen Schritten!

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)

1 Golden Master

gegenwärtiges Verhalten dokumentieren und erhalten

Foto von istara: https://pixabay.com/de/gold-bar-goldbarren-reich-geld-296115/ (CC0 Public Domain Lizenz)

```
# 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)

Live-Coding

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

Golden Master

Orientation in Objects

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

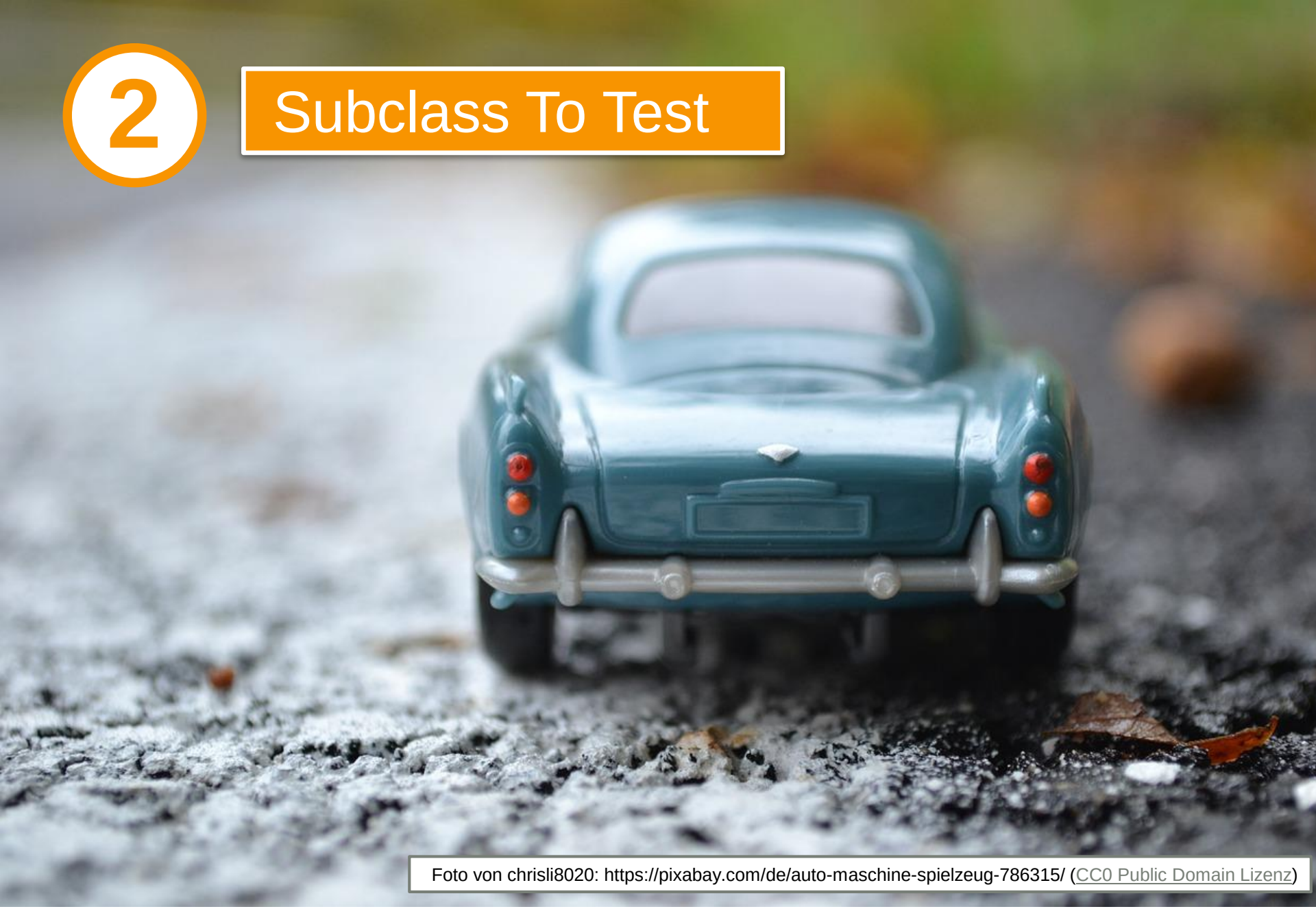


Foto von chrisli8020: https://pixabay.com/de/auto-maschine-spielzeug-786315/ (CC0 Public Domain Lizenz)

2 Subclass To Test

Orientation in Objects

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

Live-Coding

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

3 Extract Pure Functions

Foto von PublicDomainPictures: https://pixabay.com/de/wasser-tropfen-tr%C3%B6pfchen-grau-72879/ (CC0 Public Domain Lizenz)

3 Extract Pure Functions

Orientation in Objects

- seiteneffektfrei
- keine Statusänderung

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

3 Extract Pure Functions

Orientation in Objects

```
"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

Orientation in Objects

Codestellen isolieren

Ziele

Separat testbar

Duplikation reduzieren

Live-Coding

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

4 Remove Duplication

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

Beachte: Rule of Three

Foto von AdinaVoicu: https://pixabay.com/de/zwilling-schwestern-liebe-m%C3%A4dchen-948713/ (CC0 Public Domain Lizenz)

Live-Coding

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

5 Extract Class

Large class

SRP verletzt

Ziele

Unabhängiges Testen einzelner Teile

Sauberes OO-Design

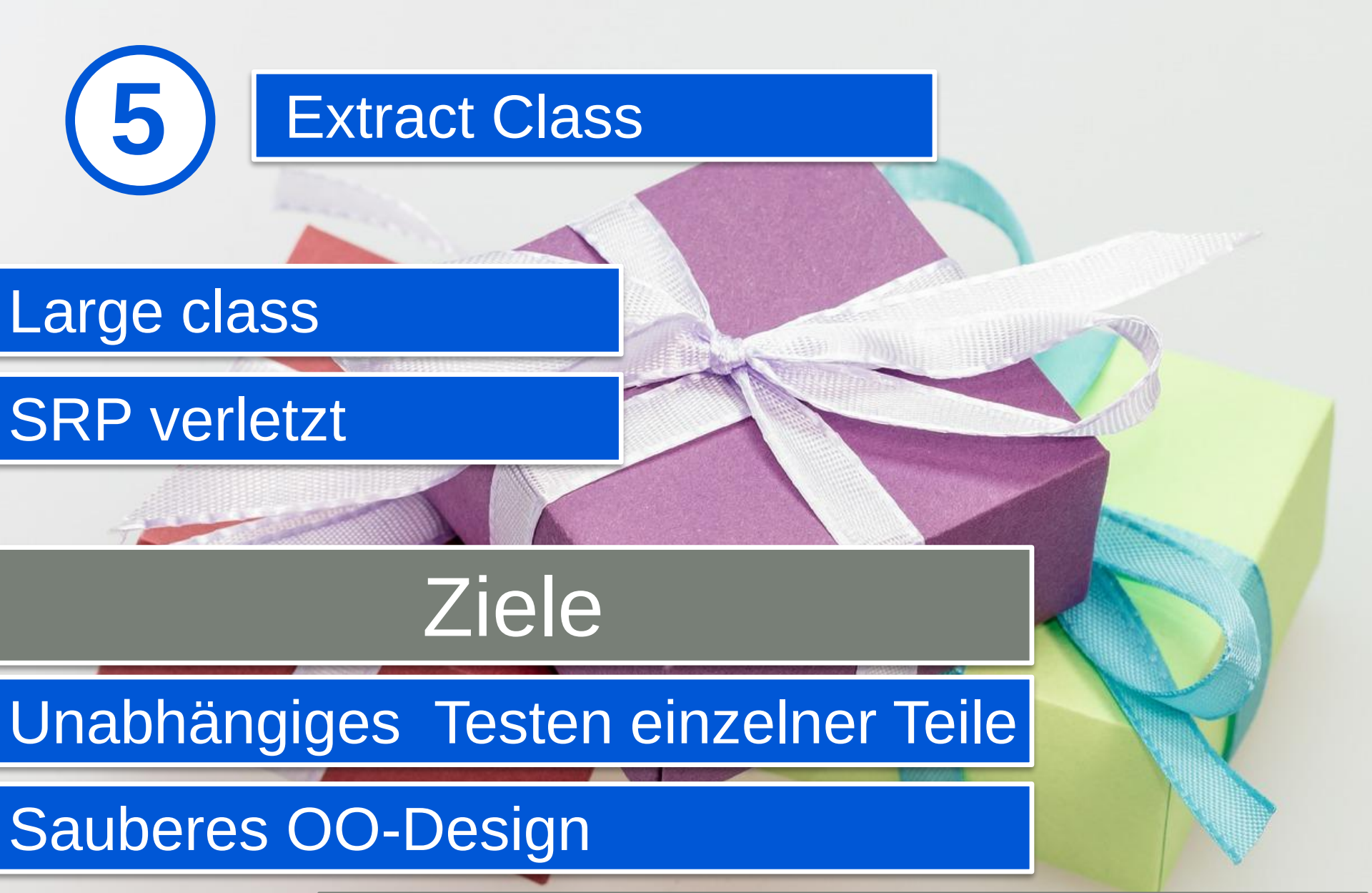


Foto von blickpixel: https://pixabay.com/de/weihnachtsdekoration-pakete-geschenk-570797/ (CC0 Public Domain Lizenz)

Live-Coding

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

~~Hätten wir mehr Zeit ...~~

6 Dependency Inversion

7 Test non-public member

8 Mocking Framework

Zusatzmaterial

Orientation in Objects

6 Dependency Inversion

7 Test non-public member

8 Mocking Framework

6 Dependency Inversion

Entkoppeln durch **explizites** Setzen der **Abhängigkeiten**

Foto von tatlin: https://pixabay.com/de/zigarette-rauch-tabak-106610/ (CC0 Public Domain Lizenz)

Live-Coding

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

7 Test non-public member

Wirklich?

Gute Alternative bei Legacy Code!

Foto von Guian Bolisay: https://www.flickr.com/photos/instantvantage/5151841152/ (CC BY-SA 2.0)

7 Test non-public member

Tools

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

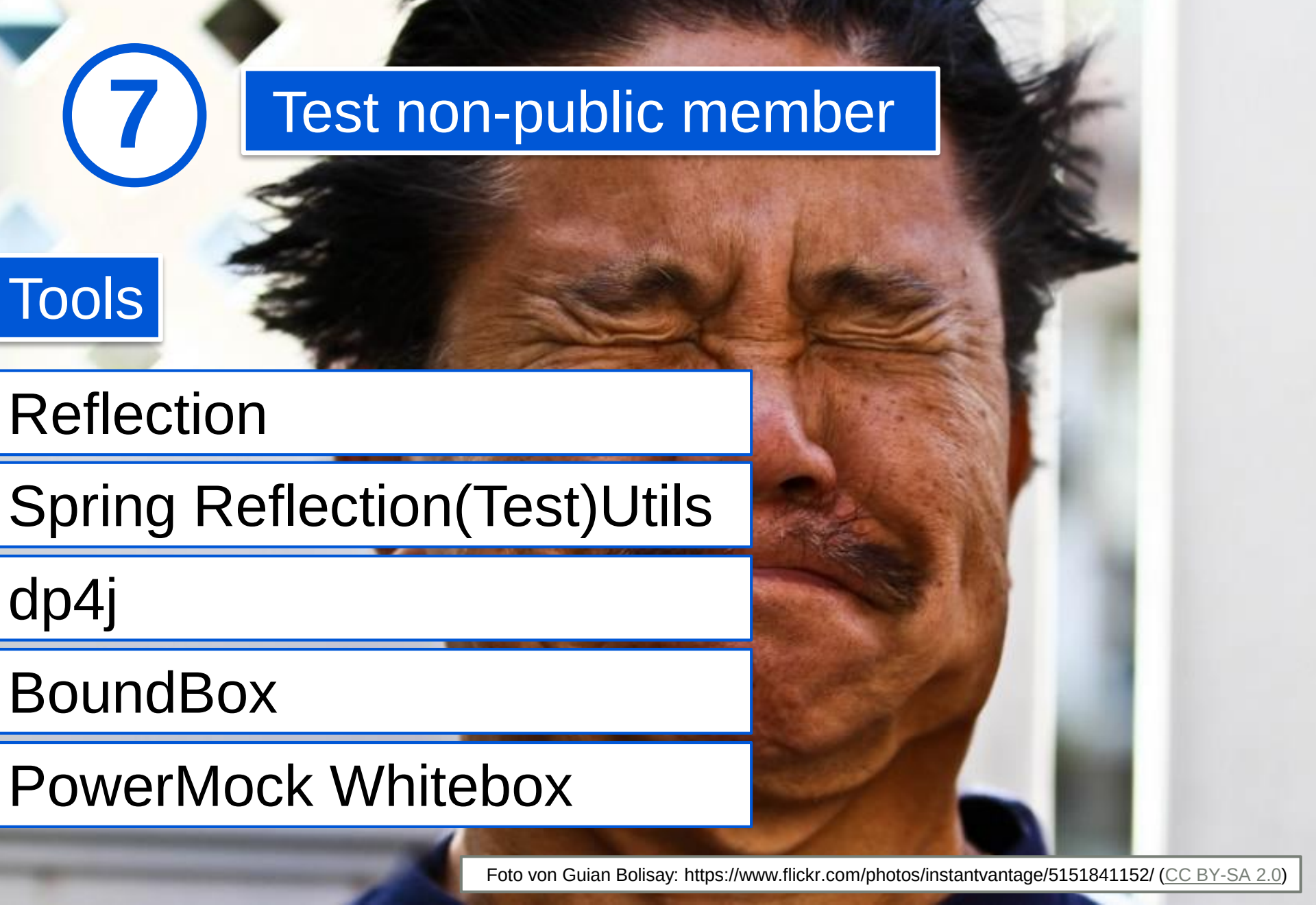
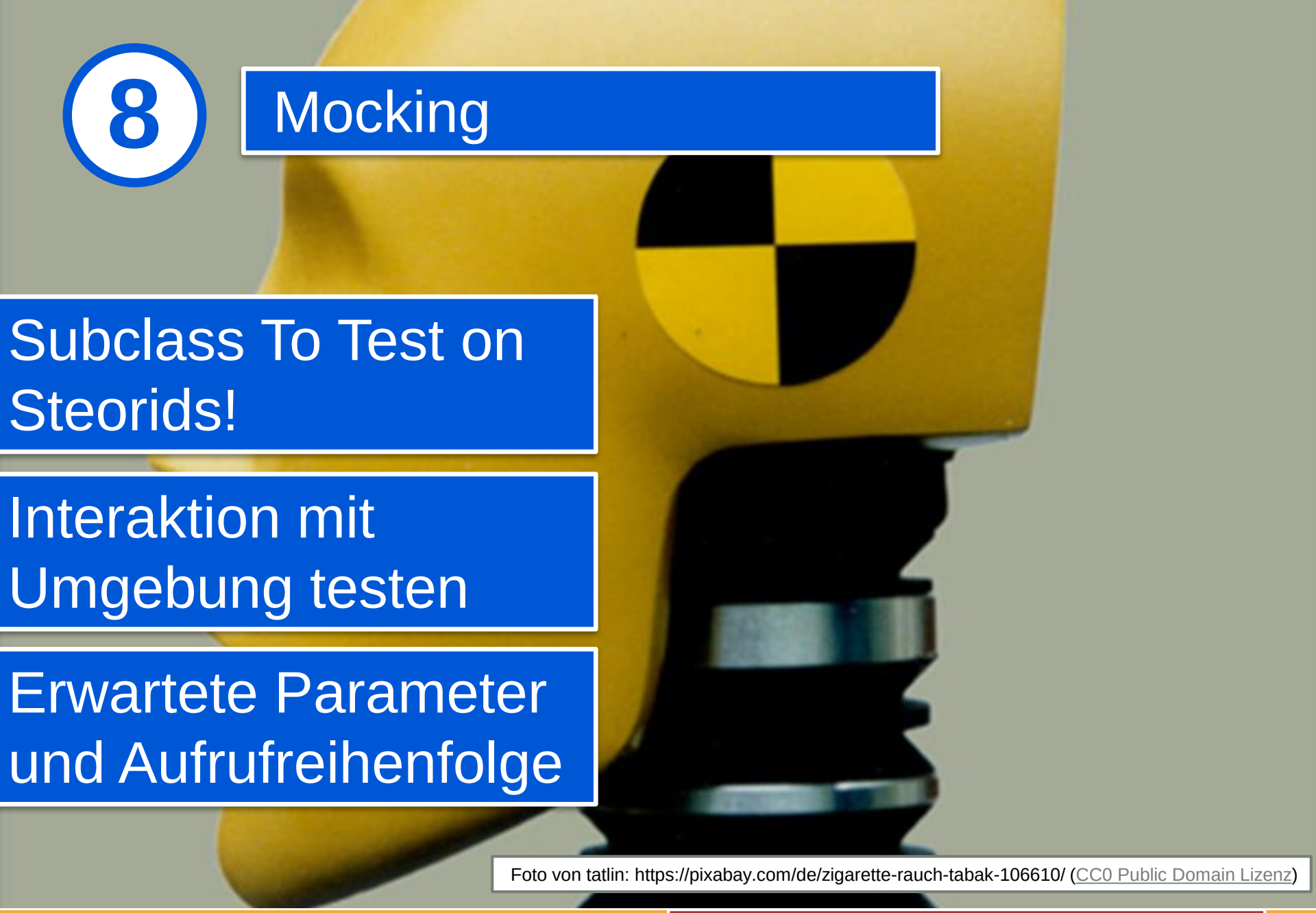


Foto von Guian Bolisay: https://www.flickr.com/photos/instantvantage/5151841152/ (CC BY-SA 2.0)

Live-Coding

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

8 Mocking



- Subclass To Test on Steorids!
- Interaktion mit Umgebung testen
- Erwartete Parameter und Aufrufreihenfolge

Foto von tatlin: https://pixabay.com/de/zigarette-rauch-tabak-106610/ (CC0 Public Domain Lizenz)

Live-Coding

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

Orientation in Objects

Zusammenfassung

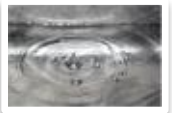
Sicherheitsnetz

1. Golden Master

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

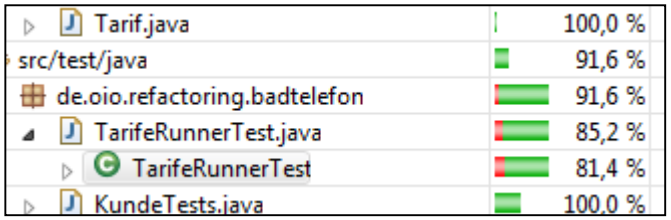
4. Remove Duplication

5. Extract Class

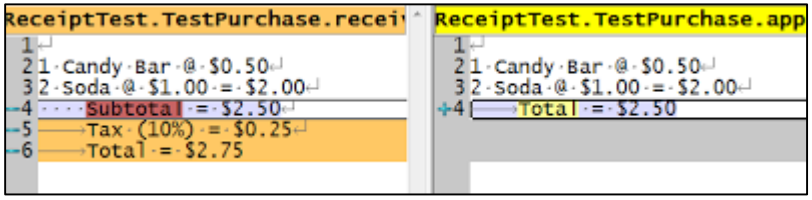
Tools

Orientation in Objects

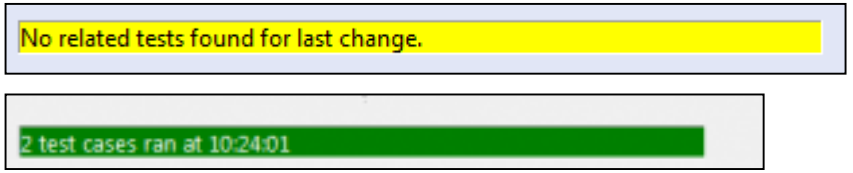
Code Coverage



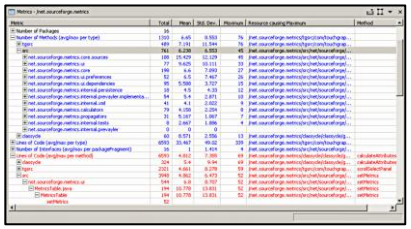
Approval Tests



Infinitest



Eclipse Metrics



Legacy Code Retreat

https://github.com/jbrains/trivia

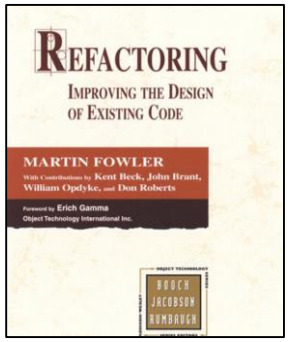
Links

Orientation in Objects

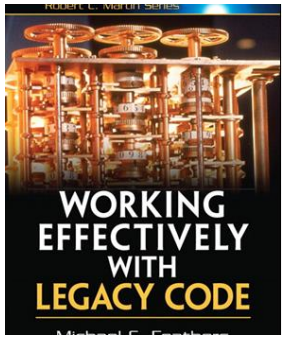
- Code-Beispiel der Live-Demo
  - https://github.com/sippsack/BadTelefon-Refactoring-Legacy-Code
- anderes Code-Beispiel für Legacy Code
  - https://github.com/jbrains/trivia
- Blog: Techniken zu Legacy Code-Retreat
  - http://blog.adrianbolboaca.ro/2014/04/legacy-coderetreat/

Literaturhinweise

Orientation in Objects



- Refactoring
  - **Sprache: Englisch**
    Gebunden - 464 Seiten - Addison Wesely
    Erscheinungsdatum: 1. Juni 1999
    ISBN: 0201485672



- Working Effectively with Legacy Code
  - **Sprache: Englisch**
  - **Gebunden**

Orientation in Objects

Fragen ?

Orientation in Objects GmbH

Weinheimer Str. 68
68309 Mannheim

www.oio.de
info@oio.de

Orientation in Objects

Vielen Dank für ihre Aufmerksamkeit !

Orientation in Objects GmbH

Weinheimer Str. 68
68309 Mannheim

www.oio.de
info@oio.de