LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN
FAKULTÄT FÜR MATHEMATIK, INFORMATIK UND STATISTIK
INSTITUT FÜR INFORMATIK
LEHRSTUHL FÜR DATENBANKSYSTEME UND DATA MINING

Programmierstil in Java

Coding-Standards und Dokumentation

SEP

Felix Zenz

23.10.2017

Wissenschaftliche Betreuer:
Prof. Dr. Peer Kröger, Janina Sontheim, Daniel Kaltenthaler, Johannes Lohrer
Verantwortlicher Professor:
Prof. Dr. Peer Kröger

DBS

Inhalt des Vortrags

- Coding-Standards
  - Objektorientierte Designprinzipien
  - Namenskonventionen
  - Formatierung
  - Weiteres
- Dokumentation (mit JavaDoc)
  - Tags
  - Erzeugen einer HTML Dokumentation

Coding Standards – Motivation

- Reicht es nicht, wenn der Code am Ende funktioniert?

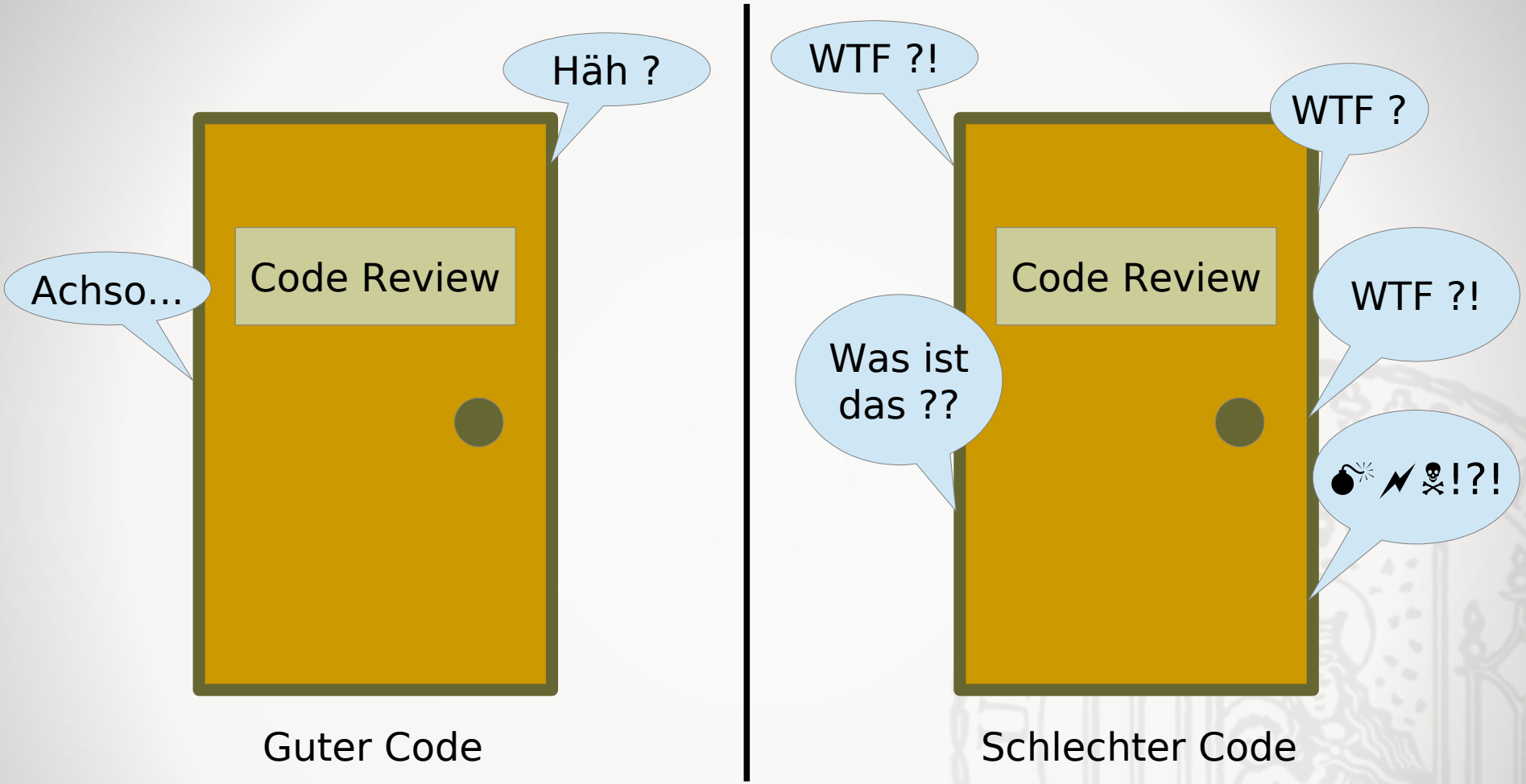
```
class asdf{public static void
main(String[]
jkl){System.out.println(((char)72)+""+
(char)101+(char)108+(char)108+(char)
111+(char
)32+(char)87+(char)111+(char)114+(char)108+(char)100
);}}
```

```
public class HelloWorld {

    private static final String MESSAGE = "Hello World";

    public static void main(String[] programArguments){
        System.out.println(message);
    }
}
```

- Beides compiliert und gibt „Hello World“ aus



Es geht bei gutem Programmierstil darum, wie leicht der Code zu lesen, verstehen und warten ist. Ca. 80% der „Lebenszeit“ eines Codes findet in der Wartung/Weiterentwicklung statt.

OO-Designprinzipien

Auch das Einhalten von Designprinzipien ist notwendig um Code lesbar zu halten:

- **S**ingle-Responsibility-Prinzip
- **O**pen-Closed-Prinzip
- **L**iskovsches Substitutionsprinzip
- **I**nterface-Seggreation-Prinzip
- **D**ependency-Inversion-Prinzip
- u.a.

Namenskonventionen

- Packages
  - Reihenfolge: Umgedrehte Internetdomain
  - `package com.example.mypackage`

- Klassen, Methoden, Variablen, Konstanten

Packages	`lowercase`
Klassen	`PascalCase`
Interfaces	`PascalCase`
Methoden	`lowerCamelCase`
Felder(Variablen), Parameter	`lowercase`
Konstanten(final)	`UPPER_CASE`

Namenskonventionen

- Sprache: Englisch
- Abkürzungen möglichst vermeiden
  - `rectangleBottom` nicht `rBottm`
- Sprechende Namen
- boolscheVariablen positiv formulieren:
  - `active` nicht `notActive`
- Methoden tun etwas → Verben
  - `fillRectangle()` nicht `rectangleFiller()`
- get/set/is
  - `getRectangle()` liefert `rectangle`
  - `isActive()` liefert `active`

Formatierung

- Einheitliches Encoding (UTF-8)
- Einheitliche Tabulator-Länge / keine Tabs
- Zeilenlänge beschränkt halten (80 Zeichen) evtl. 100

  Sollte die Zeile zu lang werden:
  - Umbruch nach Operator oder Komma
  - Ausrichtung am Anfang des Ausdrucks

```
static final ImmutableMap<String, SomeMutableType> mutableValues =
    ImmutableMap.of("Ed", mutableInstance, "Ann", mutableInstance2);
```

Formatierung

```
@Override public void method() {
    if (condition()) {
        try {
            something();
        }  catch (ProblemException e) {
            recover();
        }
    } else if (otherCondition()) {
      somethingElse();
    } else {
      lastThing();
    }
}
```

Google Java StyleGuide

Weiteres

- Checkstyle
- Google Java StyleGuide: https://google.github.io/styleguide/javaguide.html

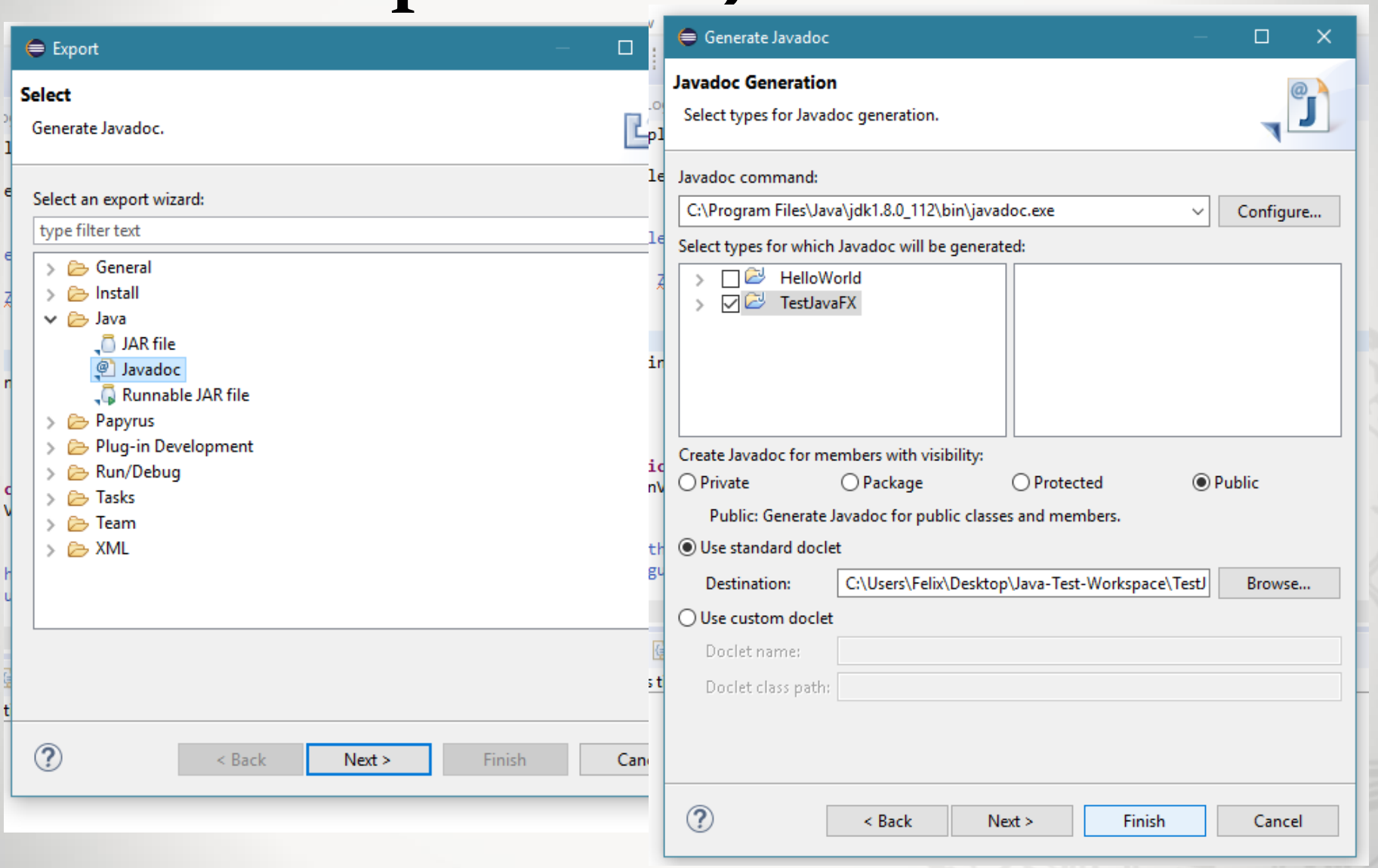
JavaDoc

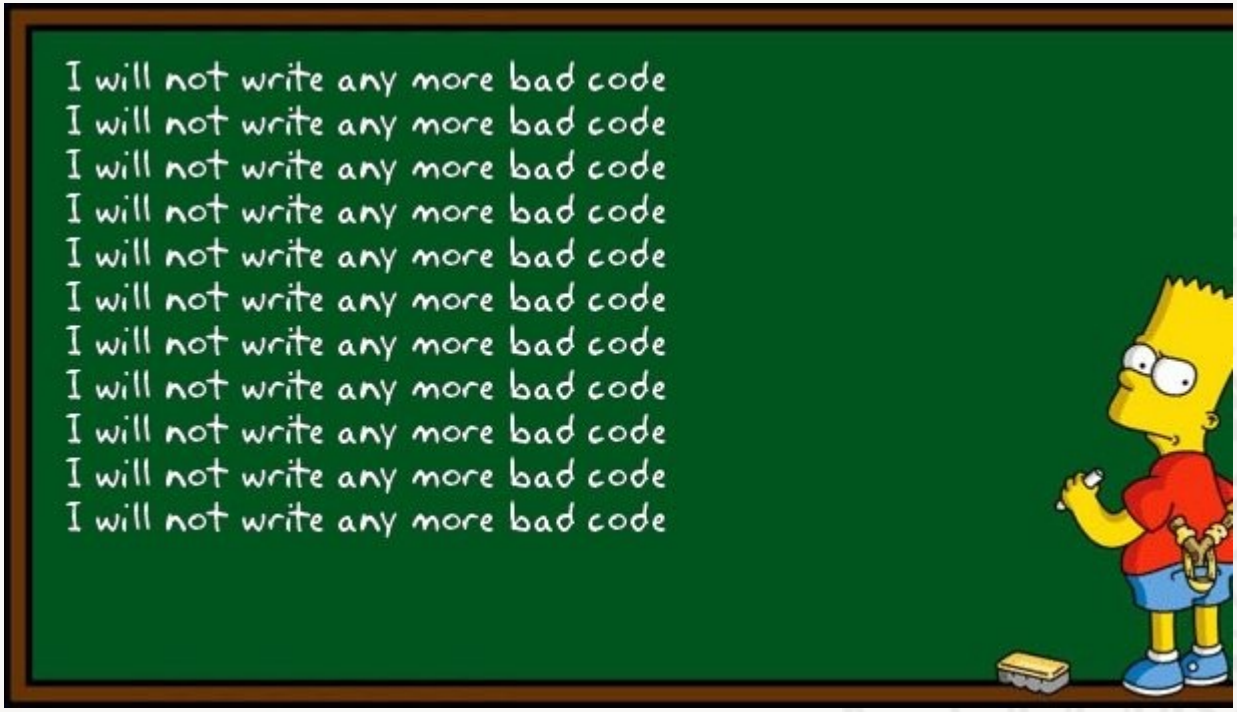
- Dokumentation von Packages, Klassen, Interfaces, Methoden, Variablen, Konstanten
- Hat die Struktur /** JavaDoc Kommentar */
- Lässt sich als HTML-File ausgeben
- Ermöglicht div. Tags (z.B. `@author`)
- Ist in `public` Klassen, Methoden, etc. immer nötig
- Kann aber natürlich auch für `private`, `protected`, etc. genutzt werden
- Erleichtert, fremde Methoden und Klassen zu nutzen, ohne in den Code zu schauen

Tags (Auswahl)

`@author` name	Dokumentiert den Autor
`@version` version	Dokum. die Version
`@since` jdk-version	Seit wann verfügbar
`@param` name beschreibung	Dokum. Parameter einer Methode
`@return` beschreibung	Dokum. Rückgabewert einer Methode
`@exception` klassenname beschreibung `@throws` klassenname beschreibung	Dokum. den „Wurf“ einer Exception
`@code` beschreibung	Beschreibung in `Consolas`
`@link` javadoc-referenz beschreibung	Verlinkt zu anderer Dokumentation

HTML Export der JavaDoc




I will not write any more bad code
I will not write any more bad code
I will not write any more bad code
I will not write any more bad code
I will not write any more bad code
I will not write any more bad code
I will not write any more bad code
I will not write any more bad code
I will not write any more bad code
I will not write any more bad code
I will not write any more bad code

Weiterführendes

- http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html#styleguid
- https://google.github.io/styleguide/javaguide.html
- http://www.oracle.com/technetwork/articles/java/index-137868.html