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

# Coding-Standards und Dokumentation

SEP 2018

Tobias Lingelmann

2018-04-17

DBS

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

# Übersicht

- Coding-Standards
  - Objektorientierte Designprinzipien
  - Namenskonventionen
  - Formatierung
  - Kommentare
- Dokumentation (mit Javadoc)
  - Tags
  - Erzeugen der HTML-Dokumentation

# Motivation

```
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)1
00);}}
```

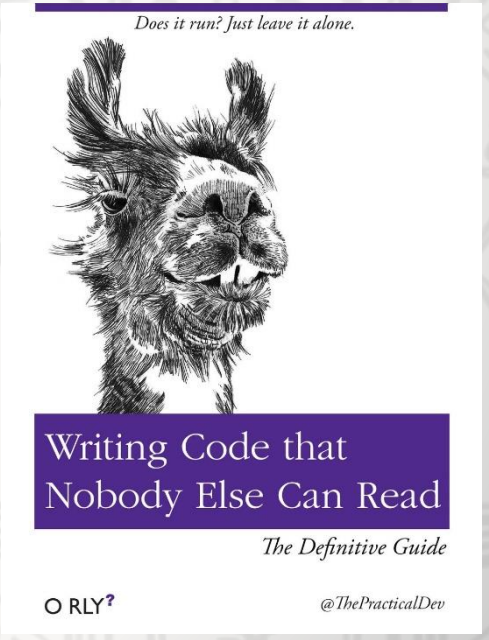
```
public class HelloWorld {

    private static final String MESSAGE = "Hello World";

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

# Motivation

- Bei gutem Programmierstil geht es darum, wie gut der Code zu lesen, verstehen und warten ist.
- 80% der Zeit befindet sich Code in der Wartung/Weiterentwicklung.
- Ihr arbeitet nicht alleine! ☺



# Designprinzipien

Einhalten gewisser Designprinzipien sorgt für besser lesbaren und einfacher zu wartenden Code:

- Single-Responsibility-Prinzip
- Open-Closed-Prinzip
- Liskovsches Substitutionsprinzip
- Interface-Segregations-Prinzip
- Dependency-Inversion-Prinzip

- Don‘t repeat yourself!

# Namenskonventionen

- Packages
  - Reihenfolge: Umgedrehte Internetdomain
  - Beispiel: com.company.project.mypackage
- Groß-/Kleinschreibung:
  - Packages: `lowercase`
  - Klassen: `PascalCase`
  - Interfaces: `PascalCase`
  - Methoden: `lowerCamelCase`
  - Variablen/Parameter: `lowercase`
  - Konstanten: `UPPER_CASE`

# Namenskonventionen

- Sprache: Englisch
- Abkürzungen möglichst vermeiden
  - `rocketFuelSensor` vs. `rFlSns`
- Selbsterklärende Namen
- Boolsche Variablen positiv formulieren
  - `active` vs. `notActive`
- Methoden tun etwas → Verben
  - `startRockets()` vs. `rocketStarter()`
- get/set/is
  - `getRocket()` liefert `Rocket`-Object
  - `isActive()` liefert `active` (bool)

# Formatierung

- Einheitliches Encoding (UTF-8)
- Tabs oder Leerzeichen (feste Anzahl!)
- Zeilenlänge beschränkt halten
- Zeilenumbrüche so, dass Code klar und einfacher zu verstehen ist. Das Ziel ist NICHT so wenig Zeilen wie möglich zu benutzen!

# Formatierung

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

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

# Javadoc

- Dokumentation von Packages, Klassen, Interfaces, Methoden, Variablen, Konstanten
- Struktur: `/** Javadoc */`
- Umwandlung in schöne HTML-Dokumentation möglich.
- Ist für `public` Klassen, Methoden, etc. immer nötig.
- Kann (und soll hier im Praktikum) auch für `private`, `protected`, etc. genutzt werden. (Es hilft euren Team-Mitgliedern!)
- Erleichtert fremde Methoden und Klassen zu nutzen ohne in den Code zu schauen.
- Eclipse bietet die Möglichkeit Warnungen für fehlende und falsche Javadoc einzublenden. ☺

# Javadoc

- Der erste Satz jedes Javadoc-Kommentar wird in der Übersicht angezeigt → sollte informativ sein.
- Javadoc für Methoden sollten mit einem Verb beginnen.
- Vermeidet (wenn möglich) lediglich den Methodennamen umzuformulieren.
- Nutzt @link um auf andere Klassen/Methoden zu verlinken.
- Man kann HTML-Formatierungen wie beispielsweise <b>, <ul>, <li> und <pre> nutzen.
- Klassen, Methoden und Parameter ändern sich → Javadoc sollte gleich mit geändert werden.

# Tags (Ausschnitt)

- @author name
  - Autor der Klasse/des Interfaces
- @version version
  - Version der Klasse/des Interfaces
- @param name beschreibung
  - Parameter einer Methode oder eines Konstruktors
- @return beschreibung
  - Rückgabewert einer Methode
- @throws klasse beschreibung
  - Exception
- @see package.Class#field
  - Verweise auf andere Klassen/Methoden

Viele mehr: http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html

# Javadoc-Export als HTML