Software-Qualität

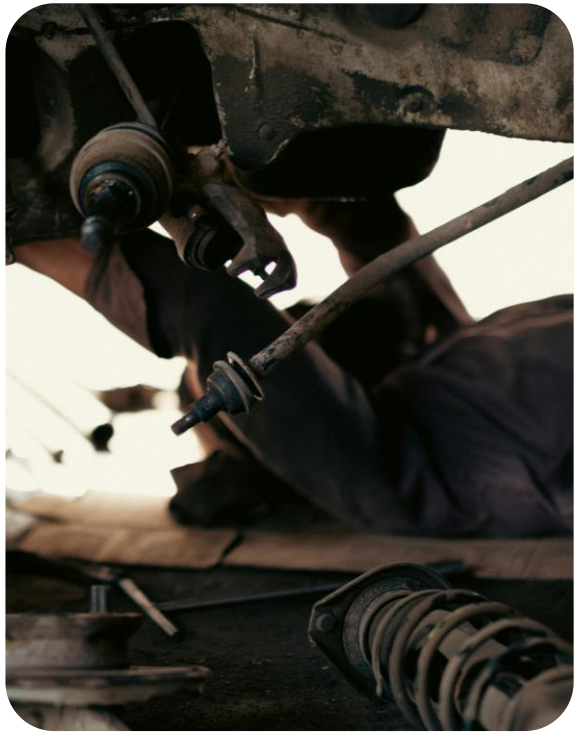
Teil 3 – Wartbarkeit

INNOVET
KI B³
Künstliche Intelligenz in die berufliche Bildung bringen

INNOVET
GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung
bibb Bundesinstitut für Berufsbildung

Gefördert als InnoVET-Projekt aus Mitteln des Bundesministeriums für Bildung und Forschung.

Wartung

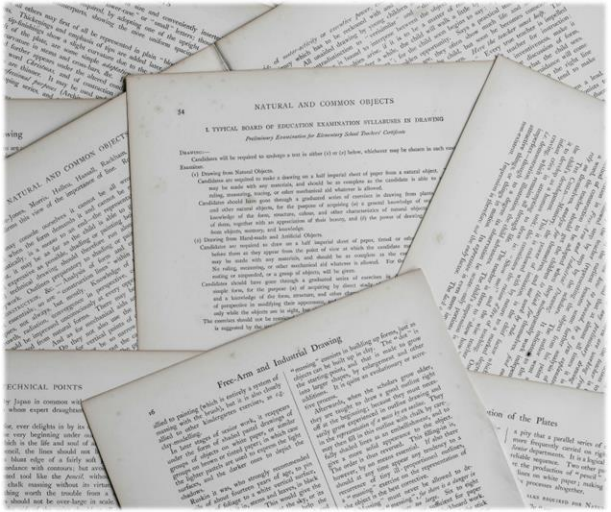


- Software ist immateriell – und hat daher keinen Verschleiß wie physische Produkte
- **Aber**: An Software wird ständig „herumgeschraubt“
  - Software wird kontinuierlich erweitert und verändert

    → Kontinuierliche Entwicklung (**Continuous Development**)
  - Regelmäßige Software-Updates
- Wir müssen sicherstellen, dass Software **leicht zu warten** ist
- Code, der heute geschrieben wird, sollte auch in 10 Jahren noch erweitert werden können

Dokumentation

- Ein wichtiger Aspekt, um Wartbarkeit zu ermöglichen, ist die **Dokumentation** der Software
- **Alle** Phasen des Softwareentwicklungs-Lebensszyklus sollten dokumentiert werden
  - Anforderungen
  - Architekturentscheidungen und -entwürfe
  - Implementierung
  - Testdurchläufe
- Dokumente in der Softwareentwicklung sind “**lebendig**”
  - Sie sollten kontinuierlich aktualisiert werden und immer den aktuellen Stand des Systems wiederspiegeln



Verständlichkeit (1)

```python
def fl(s): return s[0], s[-1]

print(fl("Hello World!"))
```

```python
def first_and_last(string):
    first_letter = string[0]
    last_letter = string[-1]
    return first_letter, last_letter

print(first_and_last("Hello World!"))
```

```
OUTPUT: ('H', '!‘)
```

- Guter Code ist nicht immer der kürzeste
- Oft ist es wichtiger, **verständlichen** Code zu schreiben

Verständlichkeit (2)

- **Verständlichkeit** eine der wichtigsten Eigenschaften von wartbarem code
- Meist der erste Schritt bei der Modifizierung oder Erweiterung von Code ist das
  - **Programmverstehen**
- Unverständlicher Code kann den Wartungsaufwand vergrößern
- Manchmal kann es wichtiger sein, Code verständlicher als effizienter zu machen

Lesbarkeit

❌

```
function calculateSum(a, b) { return a + b; } console.log("The sum is:", calculateSum(5, 7));
```

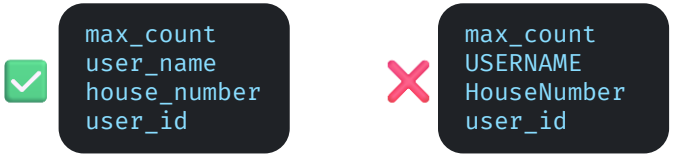
✅

```
function calculateSum(a, b) {
    let sum = a + b;
    return sum;
}

let result = calculateSum(5, 7);
console.log("The sum is:", result);
```

- Um Quellcode zu verstehen, muss er **gelesen** werden
- Struktur, Komplexität, Einrückungen etc. beeinflussen die Lesbarkeit von Quellcode
- Code-Editoren haben oft Funktionen, um Formatierung des Codes zu automatisieren

Benennung



- Variablen und Methoden sollten möglichst **konsistent** benannt werden

- Sprechende Namen

- Unkonventionelle Abkürzungen vermeiden

- Vermeiden von Variablen mit Einzelbuchstaben

Kommentare

```
/**
 * Calculates the sum of two numbers.
 *
 * @param {number} a The first number to add.
 * @param {number} b The second number to add.
 * @return {number} The sum of `a` and `b`.
 */
function calculateSum(a, b) {
    return a + b;
}
```

```
// Output the result
console.log("Ergebnis:", result);
```

- Kleine „Notizen“ im Code, die helfen, den Code zu verstehen
  - Methodenkommentare: Beschreiben die bereitgestellte Funktionalität einer Methode
  - Zeilenkommentare: Kleine zusätzliche Infos oder „Überschriften“ von Code-Abschnitten
- Code-Kommentare werden beim Ausführen von Programmen vom Computer ignoriert
- **Aber**: Kommentare sind keine Freikarte, um unlesbaren Code zu schreiben
  - Code der lesbar **und** gut kommentiert ist, ist **verständlich**

Code Smells

- **Code Smells** (Gerüche) sind wiederkehrende Merkmale von Code, die zeigen, dass hier etwas „stinkt“
  - Unsorgfältige Entwicklung hat zu niedriger Code-Qualität geführt
  - Code Smells sollte möglichst **vermieden** werden
  - Wir möchten **Clean** (Sauberen) **Code** schreiben
- Beispiele:
  - Kommentare wie „Ich habe keine Ahnung, was hier passiert“ oder „Das sollte mal jemand fixen“
  - Duplizierter Code
  - Lange Klassen/Methoden

Grundlegende Richtlinien für Clean Code

- ✅ Code ist **konsistent** formatiert
- ✅ Code ist hinreichend **kommentiert**
- ✅ Kein auskommentierter Code
- ✅ **Sinnvoll** benannte Variablen und Methoden
- ✅ Nicht zu komplexe / lange Methoden

Refactoring
Das Verändern von Quellcode ohne die Funktionalität zu beeinflussen

Code-Wartung

- Eine wichtige Aufgabe bei der Wartung von Software ist „**Refactoring**“
  - Wir versuchen, die Qualität des zu verbessern, ohne dabei die Funktionalität zu beeinträchtigen

- Ein hilfreiches Mantra:
  - **Hinterlasse Code immer besser, als Du ihn aufgefunden hast!**

- Es gibt weitere Methoden, um Code-Qualität zu fördern
  - **Coding Richtlinien**
  - **Code-Reviews**

Coding Richtlinien (Guidelines)

- Projekt- oder Unternehmensweiter **Standard**, wie Code geschrieben werden soll
  - Format & Stil
  - Kommentare

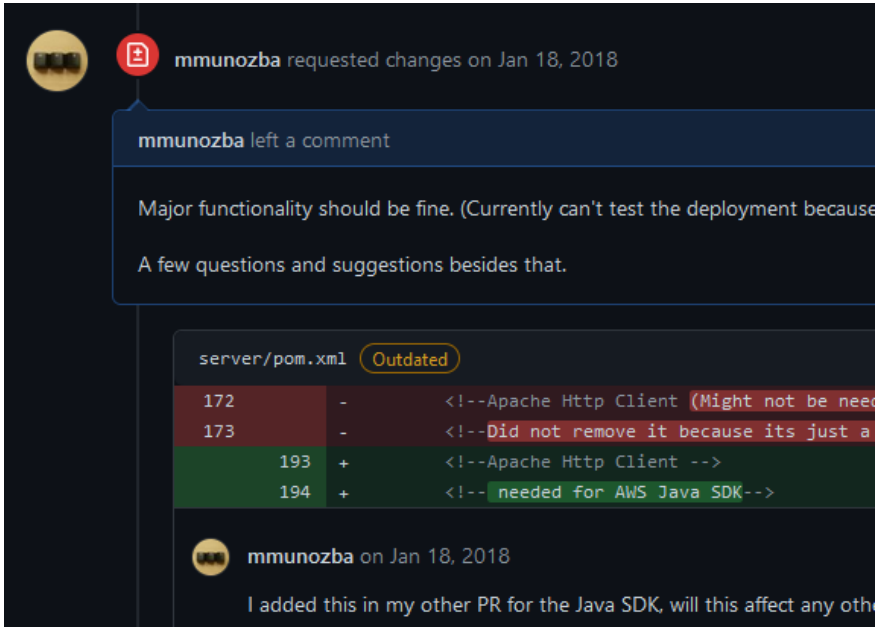
4.4 Column limit: 100

Java code has a column limit of 100 characters. A "character" means any Unicode code point. Except as noted below, any line that would exceed this limit must be line-wrapped, as explained in Section 4.5, Line-wrapping.

> Each Unicode code point counts as one character, even if its display width is greater or less. For example, if using fullwidth characters, you may choose to wrap the line earlier than where this rule strictly requires.

Google. "Google Java Style Guide“ unter CC-BY-3.0. Online am 06.01.2023: https://google.github.io/styleguide/javaguide.html

Code-Reviews



Screenshot eines GitHub Pull-Request Reviews. Online am 06.01.2023.

- „Vier Augen sehen besser als Zwei“
- Bevor Code in der Produktion ausgeführt wird, sollte er **überprüft** werden
- **Code-Review**
  - Eine unabhängige Person überprüft die geplante Änderung am Quellcode
  - Kommentare, Verbesserungsvorschläge, entdeckte Fehler

INNOVET
KI B³
Künstliche Intelligenz in die berufliche Bildung bringen

Universität Stuttgart
Institut für Maschinelle Sprachverarbeitung
Institut für Software Engineering

Erstellt durch:

Marvin Muñoz Barón

Umm-e-Habiba

Tobias Eisenreich

Universität Stuttgart
Institut für Software Engineering
Empirisches Software Engineering

IHK Industrie- und Handelskammer Reutlingen
Reutlingen | Tübingen | Zollernalb

IHK Region Stuttgart

IHK Industrie- und Handelskammer Karlsruhe

LMU Ludwig-Maximilians-Universität München

GEFÖRDERT VOM
INNOVET
Bundesministerium für Bildung und Forschung
bibb Bundesinstitut für Berufsbildung

Gefördert als InnoVET-Projekt aus Mitteln des Bundesministeriums für Bildung und Forschung.

Lizenzbestimmungen

Quellenverzeichnis

- Foto von Dmitriy Demidov: https://unsplash.com/de/fotos/eine-gruppe-von-schraubenschlusseln-die-im-kreis-angeordnet-sind-iuuJC_pjLU0 unter Unsplash Lizenz (https://unsplash.com/de/lizenz). Bildausschnitt verändert.
- Foto von Tahamie Farooqui: https://unsplash.com/de/fotos/ein-mann-der-an-einem-auto-unter-einem-fahrzeug-arbeitet-K5624F8cipE unter Unsplash Lizenz (https://unsplash.com/de/lizenz). Bildausschnitt verändert.
- Foto von Annie Spratt: https://unsplash.com/de/fotos/weisses-druckerpapier-lot-5cFwQ-WMcJU unter Unsplash Lizenz (https://unsplash.com/de/lizenz). Bildausschnitt verändert.
- Google. "Google Java Style Guide“ unter CC-BY-3.0. Online am 06.01.2023: https://google.github.io/styleguide/javaguide.html