

Fragen, Antworten und Kommentare zur aktuellen Vorlesung

Frage: Muss bei Nutzungseingaben immer eine Anforderung formuliert werden, was bei einer fehlerhaften Eingabe passiert?

Antwort: Wenn es nur um die Prüfung eines passenden Datenformats geht, z. B. wurden statt Ziffern Buchstaben eingegeben, sollte dies in einer generellen Anforderungsliste stehen, die für alle Eingabemöglichkeiten zu nutzen ist, z. B.

Generell1: Nachdem eine Eingabe von Daten erfolgt, muss das System prüfen, ob die eingegebenen Daten das korrekte Format haben.

Generell2: Nachdem bei der Prüfung eingegebener Daten Fehler gefunden wurden, muss das System den Eingebenden auf den Fehler hinweisen, das erwartete Format anzeigen und die Wiederholung der Eingabe ermöglichen.

Sollte die Eingabe für die Steuerung des weiteren Ablaufs relevant sein, muss sie in den individuellen Anforderungen berücksichtigt werden. Ein konkretes Beispiel sind Suchanfragen, für die Fälle „gefunden“ und „nicht gefunden“ in den Folgeanforderungen bearbeitet werden.

Frage: Ich habe eine Frage bezüglich der Zielformulierung nach der Muss/Soll/Kann/ Nicht-Methode für ein Projekt, an dem ich derzeit arbeite. Im Rahmen meiner Praxisphase soll ich ein Programm entwickeln, das Daten sammelt, diese in einer Datenbank speichert und anschließend für Anfragen aufbereitet und darstellt. Ich möchte dieses Projekt gleichzeitig als Übung nutzen, um die Zieldefinition zu verfeinern und die Funktionalität des Programms in Absprache mit meinen Vorgesetzten grob zu skizzieren. Obwohl ich bereits einige Unterlagen zum Thema Zielformulierungen habe, tue ich mich etwas schwer, da dies keine alltägliche Aufgabe ist. Mir ist bekannt, dass es verschiedene Methoden zur Zielformulierung gibt, wie die *SMART-Methode* und die *RUBB-Schablone*. Könnte ich in diesem Fall beide Methoden verwenden, oder gibt es eine bevorzugte Herangehensweise, die für diesen Kontext besser geeignet ist?

Antwort: SMART (spezifisch, messbar, ausführbar, realistisch, terminiert) beschreibt ja nur generell Kriterien an Ziele oder funktionale und nicht-funktionale Anforderungen, die sollte man also immer im Hinterkopf haben.

Rupp-Schablone (nach Chris. Rupp) ist sehr gut geeignet funktionale Anforderungen zu formulieren (s. meine OOAD-Videos). Wenn Sie mit dem Projekt starten, ist eher ein grober Überblick sinnvoll, bei dem Use Cases oder genauso gut Epics und User Stories eingesetzt werden. Aus beiden können dann Verfeinerungen stattfinden, z. B. durch Prozessanalysen (z. B. mit Aktivitätsdiagrammen), die dann mit der Rupp-Schablone in Text gegossen werden können. Nicht-funktionale Anforderungen werden nebenbei eingesammelt.

muss: ist garantiert gefordert

soll: Variante 1: umsetzen, wenn jetzt möglich

Variante 2: ist irgendwann geplant, soll aber drüber nachgedacht werden, dass es umsetzbar ist

kann: wäre schön, wenn (also Ausblick, ist damit ähnlich, wie obige Variante 2)

Variante in Bachelorarbeiten: Immer nur muss nutzen, aber gruppieren nach Muss- und Kann-Kriterien/Anforderungen.

Hinweis: Teilweise motiviert durch eine andere Entwicklungsumgebung sehe ich bei if-Anweisungen häufiger die folgende Formatierungsart:

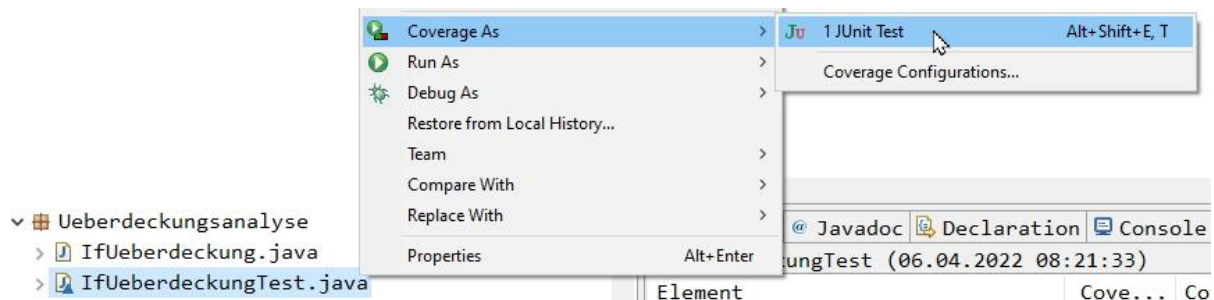
```
public class IfUeberdeckung {
    public String ueber42(int wert) {
        if(wert <= 42) return "nicht ok";
        return "ok";
    }
}
```

Diese ist aus mehreren Gründen schlecht.

Zu jeder Software werden Tests geschrieben, wie der folgende Code zeigt.

```
@Test
void test() {
    Assertions.assertEquals("ok", new IfUeberdeckung().ueber42(43));
}
```

Um zu erkennen, ob es genügend Tests gibt, hilft die Messung der Testüberdeckung, die auf verschiedene Wege passieren kann (siehe [Kle19], auch Vorlesung zum Software-Engineering-Projekt). Um die von Eclipse automatisch unterstützte Messung zu nutzen, wird eine Testklasse über „Coverage As“ ausgeführt.



Die Überdeckung wird dann u. a. farblich angezeigt (grün ist genutzt, rot wurde nicht ausgeführt, gelb „so halb“). Aus der folgenden Überdeckung ist zwar ersichtlich, dass irgendwas beim if nicht ganz geklappt hat, aber u. a. die Auswirkungen sind unklar.

```
7 public String ueber42(int wert) {
8     String erg = "ok";
9     if(wert <= 42) erg = "nicht ok";
10    return erg;
11 }
```

Wird der Code besser formatiert, ist das Problem des nicht ausgeführten Codes unmittelbar sichtbar.

```

7 public String ueber42(int wert) {
8     String erg = "ok";
9     if(wert <= 42)
10        erg = "nicht ok";
11    return erg;
12 }

```

Leider ist die Formatierung immer noch schlecht, wenn auch für Testüberdeckungsmessungen in Ordnung. Das Problem zeigt der folgende Code, bei dem eine nachfolgende entwickelnde Person eine Warnung ergänzen wollte.

```

public String ueber42(int wert) {
    String erg = "ok";
    if(wert <= 42)
        erg = "nicht ok";
        System.err.println("zu niedriger Wert");
    return erg;
}

```

Es sollte klar sein, dass die Warnung genau beim falschen Fall ausgegeben wird. Mit geschweiften Klammern kann das Problem nicht auftreten.

```

public String ueber42(int wert) {
    String erg = "ok";
    if(wert <= 42) {
        erg = "nicht ok";
        System.err.println("zu niedriger Wert");
    }
    return erg;
}

```

Das Beispiel ist nebenbei nicht „akademisch“, es hat schon zu einem gravierenden Sicherheitsproblem geführt, wie es in <https://www.spiegel.de/netzwelt/web/goto-fail-apples-furchtbarer-fehler-a-955154.html> beschrieben ist. Sie schreiben Code immer so, dass er für andere entwickelnde Personen leicht les- und erweiterbar ist. Obiger Fehler passiert z. B. sehr schnell, wenn bei der Entwicklung zwischen Java und Python umgeschaltet werden muss, da Leerzeichen zur Einrückung in Python eine zentrale Bedeutung haben um die Blockzugehörigkeit definieren.

Leider gibt es an dieser Stelle oft die Frage, muss ich das auch bei der Hausarbeit berücksichtigen. Die Standardantwort ist, dass Sie die Hausarbeit so gestalten können wie Sie wollen; nur muss ich Ihnen auch keine gute Note geben.

Abschließend, ja die obige Methode könnte noch etwas kürzer programmiert werden. Das wäre auch ok. Da es aber praktisch keinen Performance-Unterschied gibt, würde das für eine Bewertung irrelevant sein.

[Kle19] S. Kleuker, Qualitätssicherung durch Softwaretests, 2. aktualisierte und erweiterte Auflage, Springer Vieweg, Wiesbaden, 2019

Frage: Ich habe mir die schlechte Formatierung angewöhnt, ist das ein Problem für die Hausarbeit?

Antwort: Ja. (bzw. „nein“, Sie dürfen formatieren wie Sie wollen und ich darf bewerten fast wie ich will)

Frage: Dürfen wir für die Hausarbeit KI nutzen?

Antwort: Generell ja. Es sind aber einige Punkte zu beachten. Da die KI Ergebnisse erzeugen kann, die direkt in die Arbeit übernommen werden, würde sie in diesem Fall als weitere Person auftreten, was so nicht erwünscht ist. Wenn Sie generierte Texte nutzen wollen, sind diese wie ein Zitat zu kennzeichnen, dabei sind folgende Daten als Quelle anzugeben: welche KI wurde mit welcher Frage/ welchen Fragen (Prompts) befragt, ein dazugehöriger Dialog ist als Kopie mit im Zip-Verzeichnis zu ergänzen. Nutzen Sie für so eine Dokumentation einfach Copy & Paste in ein leeres Word-Dokument oder schöner ein Tool wie SingleFile als Erweiterung von Firefox, Chrome und Edge, mit dem lesbare Kopien von Webseiten erstellt werden können.



Generell sollten solche wörtlichen Zitate aber möglichst vermieden werden, da sie nur eingeschränkt eine eigene Leistung darstellen. Nicht ordentlich gekennzeichnete KI-generierte Texte führen als Täuschungsversuch automatisch zur Note „mangelhaft“.

KI-Tools können wie andere Quellen zur Recherche genutzt werden. Generell gilt, dass Bücher und Fachartikel bessere (im Sinne der Fachlichkeit und der Bewertung der Arbeit) Quellen als Webseiten und damit auch KI-Texte sind. Natürlich sind Webseiten trotzdem eine wichtige Quelle, da oft aktuellste Details sich nicht in den anderen Quellen befinden und sollen deshalb ebenfalls genutzt werden. Studierende neigen oft zum Minimalismus „es reicht eine Quelle aus“, was akademisch falsch ist, da es sinnvoll ist, mehrere Quellen anzusehen und dann auch zu benennen. Sollten die Quellen nicht genau fachlich übereinstimmen, ist auf diese Unterschiede einzugehen. Wenn Sie eine KI-generierte Information als Quelle für ihren eigenen Text nutzen, gehört sie, wie oben beschrieben, in das Literaturverzeichnis, das ein Verweis auf den abgespeicherten Dialog mit der KI in einer abgegebenen Datei enthält. Vereinfachtes Beispiel:

in der Arbeit:

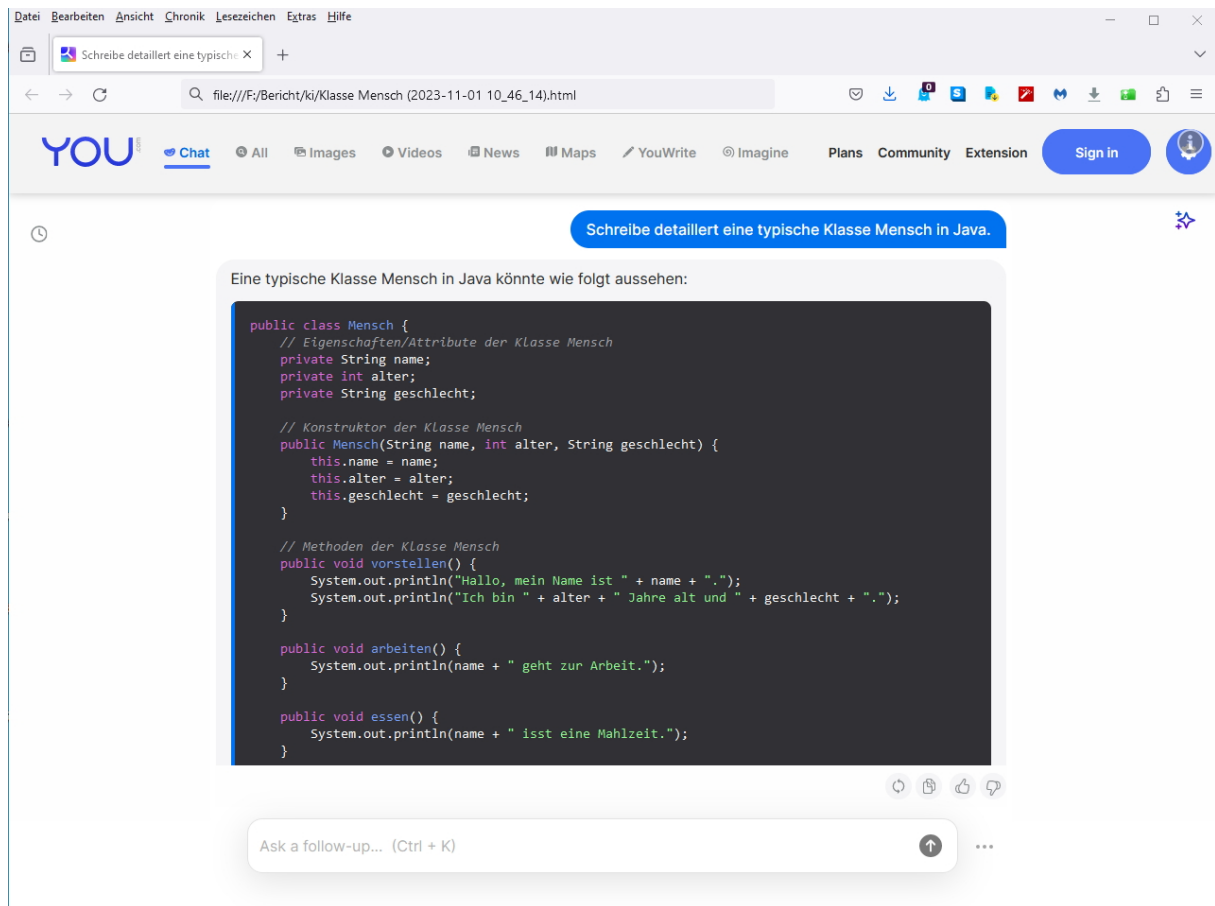
Nach [KI01] sieht eine typische Realisierung für eine Klasse Mensch in Java wie folgt aus:

```
public class Mensch {  
    // Eigenschaften/Attribute der Klasse Mensch  
    private String name;  
    private int alter;  
    private String geschlecht;  
    ...  
}
```

im Literaturverzeichnis:

[KI01] You, <https://you.com>, abgefragt am 1.11.2023, siehe ki/Klasse Mensch (2023-11-01 10_46_14).html

der Anfang der erwähnten Datei: (Typfehler bewusst beibehalten, würden zu keinen Abwertungen führen, Datei darf auch passender umbenannt werden)



Frage: Dürfen wir KI zur Korrektur von Rechtschreibfehlern nutzen?

Antwort: Ja, immer gerne, ohne Notwendigkeit zur Kennzeichnung.