
Prof. Dr. Stephan Kleuker
Hochschule Osnabrück
Fakultät Ing-Wiss. und Informatik
- Software-Entwicklung -

Objektorientierte Analyse und Design

Sommersemester 2026

5. Aufgabenblatt

Aufgabe 0.5 (1 Punkt)

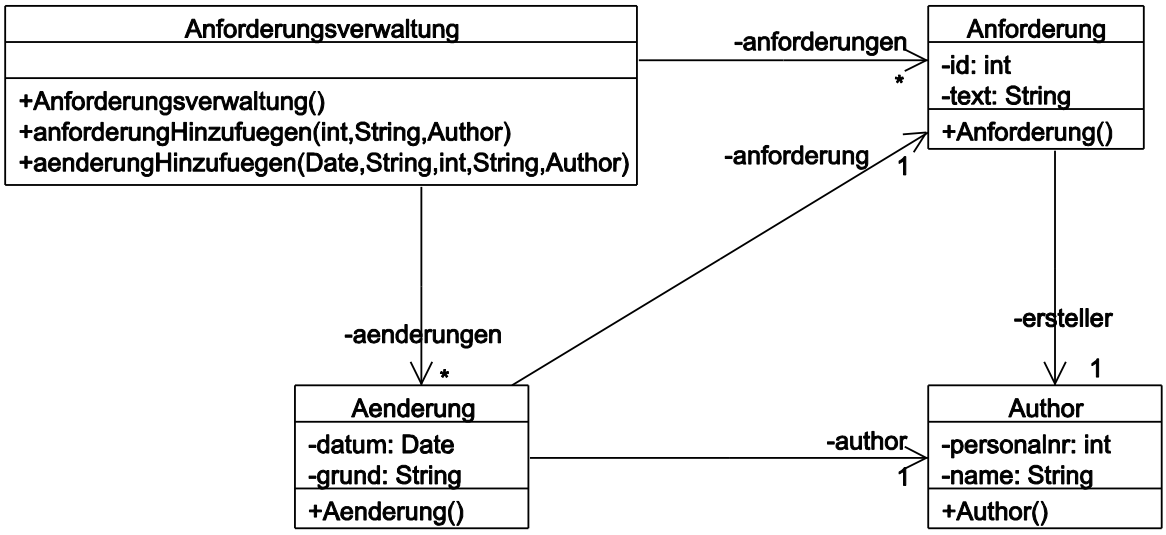
Geben Sie das Lösungswort des Quiz aus der Lernnotiz an.

Aufgabe 15 (1 Punkt)

Beantworten Sie folgende Fragen schriftlich in ganzen Sätzen oder mit Stichpunkten.

a) Welche 4 Pfeilarten haben Sie bis jetzt in UML-Klassendiagrammen kennengelernt?
b) Wieviele Objekte des gleichen Typs können in der obersten Zeile eines Sequenzdiagramms stehen?
c) Können in Java und der UML Interfaces voneinander erben? Wenn ja, wann könnte dies sinnvoll sein?
d) Kann in Java und der UML eine Klasse abstrakt sein und trotzdem alle Methoden ausimplementiert sein? Wenn ja, wann ist dies sinnvoll?

Aufgabe 16 (3 Punkte)



Sie stehen in der Projektphase Ihres Studiums vor dem Problem, sich in den Aufbau einer existierenden Software einzuarbeiten. Sie wissen nur, dass mit der Software Anforderungen mit eindeutigen Identifikatoren verwaltet werden können. Weiterhin sind die Texte der Anforderungen später veränderbar, die u. a. den genauen Zeitpunkt und den Änderungsgrund enthalten. Ältere Versionen der Anforderungen werden leider nicht verwaltet.
Gegeben sei dazu obiges Klassendiagramm, bei dem für alle Exemplarvariablen die natürlich vorhandenen get- und set-Methoden weggelassen wurden.

a) Geben Sie für jede Klasse an, wieviele Exemplarvariablen ihre Objekte haben (erinnern Sie sich daran, wofür -aenderungen steht.
b) Für die Methode anforderungHinzufuegen sind zwar die Typen der Parameter angegeben, ihre Bedeutung aber nicht. Geben Sie für jeden der Parameter an, welche Bedeutung Sie vermuten.
c) Für die Methode aenderungHinzufügen sind zwar die Typen der Parameter angegeben, ihre Bedeutung aber nicht. Geben Sie für jeden der Parameter an, welche Bedeutung Sie vermuten.
d) Welchen Typen würden Sie in Java für die Exemplarvariablen anforderungen und aenderungen nutzen?
e) Überlegen Sie sich, wie die Methode anforderungHinzufuegen nur mit den genannten Methoden, Exemplarvariablen und gezeigten Konstruktoren realisiert sein könnte. Dies kann z. B. als Java-Code-Ausschnitt nur für die Methode passieren.

f) Gehen Sie davon aus, dass es bereits passende Objekte der Typen Anforderungsverwaltung av und Author*in au gibt. Zeichnen Sie ein genaues Sequenzdiagramm (mit set- und Collection-Nutzung), bei dem ein externes Objekt die Methode av.anforderungHinzufuegen(42,“SD zeichnen“,au) aufruft.

g) Überlegen Sie sich, wie die Methode aenderungHinzufügen nur mit den genannten Methoden, Exemplarvariablen und gezeigten Konstruktoren realisiert sein könnte, mögliche Fehlerfälle können Sie ignorieren.

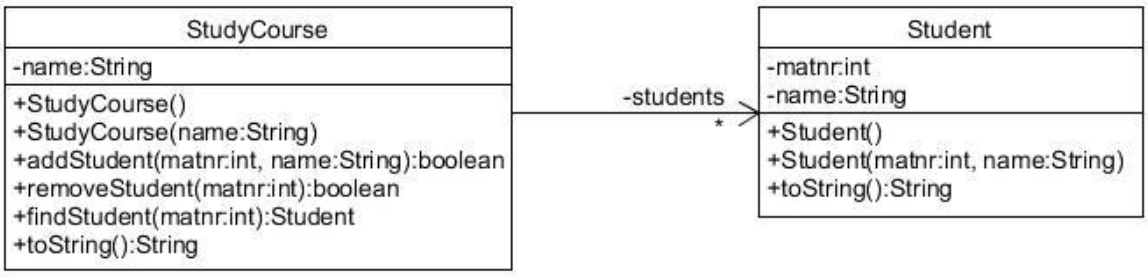
Aufgabe 17 (2 Punkte)

Erstellen Sie ein Klassendiagramm, mit dem man folgende Zusammenhänge verdeutlichen kann.

- Es werden Studierende mit Namen und Matrikelnummer verwaltet. Jede studierende Person kann beliebig viele Prüfungen gemacht haben.
- Es werden Module mit Modulnummer, Namen und Inhalt verwaltet.
- Es werden lehrende Personen mit Namen und Personalnummer verwaltet.
- Es werden Lehrveranstaltungen verwaltet, die eine eindeutige Veranstaltungsnummer, das Semester, in dem sie stattfindet, und mindestens eine lehrende Person haben, sowie zu genau einem Modul gehören.
- Es werden Prüfungen verwaltet, mit denen individuelle Noten von Studierenden für genau ein Modul von einer oder zwei lehrenden Personen in der Rolle „Prüfend“ festgehalten werden. Jede Prüfung hat weiter ein Datum, zu dem sie stattfindet und optional eine beisitzende Person deren Namen festgehalten wird.

Für jede der Klassen soll es einen Konstruktor geben, der Werte für die jeweiligen Exemplarvariablen liefert, für die es keinen sinnvollen Default-Wert gibt.
Die Verwaltungsklassen selbst können weggelassen werden.

Aufgabe 18 (2 Punkte)



Gegeben sei obiges minimales Klassendiagramm. Dabei steht das * für eine Sammlung (Collection) von Objekten. Mit addStudent wird nur eine studierende Person hinzugefügt, wenn es noch keine mit der Matrikelnummer gibt. Ob hinzugefügt wurde steht im Ergebnis der Methode. Bei removeStudent wird ebenfalls angegeben, ob eine studierende Person gelöscht wurde.
Im Diagramm ist nicht angegeben, wie diese Sammlung umgesetzt werden soll. Implementieren Sie die Klassen aus dem Klassendiagramm in *drei verschiedenen* Lösungen unter der Nutzung einer Implementierung von List, Set und Map in Java. Drei bis auf den Namen identische Teilprojekte ooadAufgabeStudyCourseList, ooadAufgabeStudyCourseMap und ooadAufgabeStudyCourseSet sollen von der Veranstaltungsseite geladen werden. Weiterhin gibt es eine Testklasse main.SystemTest, deren Tests alle durchlaufen müssen. Im nachfolgenden Beispieldialog sind Eingaben umrandet.
Beachten Sie, dass vorhandene Funktionalität nicht doppelt implementiert wird, z. B. erkennen Set-Implementierungen automatisch doppelte Elemente, wenn Sie sauber implementieren. Dazu dürfen Sie natürlich die Klasse Student ergänzen.

Welche Ihrer drei Implementierungen würden Sie in der Praxis nutzen?

Hinweis: Halten Sie sich an die üblichen Java-Coding Guidelines, z. B. http://kleuker.iui.hs-osnabrueck.de/querschnittlich/CodingGuidelinesUndGlossar.pdf

```
(0) terminate
(1) add new student
(2) remove student
(3) search student
(4) show course of Studies
1
matriculation number: 42
name: Lisa
student added

(0) terminate
(1) add new student
(2) remove student
(3) search student
(4) show course of Studies
1
matriculation number: 42
name: Arno
student not added

(0) terminate
(1) add new student
(2) remove student
(3) search student
(4) show course of Studies
3
matriculation number: 42
Student [matnr=42, name=Lisa]

(0) terminate
(1) add new student
(2) remove student
(3) search student
(4) show course of Studies
3
matriculation number: 43
student not found

(0) terminate
(1) add new student
(2) remove student
(3) search student
(4) show course of Studies
4
StudyCourse [name=Informatik,
students=[Student [matnr=42,
name=Lisa]]]

(0) terminate
(1) add new student
(2) remove student
(3) search student
(4) show course of Studies
2
matriculation number: 43
student not removed

(0) terminate
(1) add new student
(2) remove student
(3) search student
(4) show course of Studies
2
matriculation number: 42
student removed

(0) terminate
(1) add new student
(2) remove student
(3) search student
(4) show course of Studies
4
StudyCourse [name=Informatik,
students=[]]

(0) terminate
(1) add new student
(2) remove student
(3) search student
(4) show course of Studies
0
```