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

Objektorientierte Analyse und Design

Wintersemester 2025
7. Aufgabenblatt

Aufgabe 0.7 (1 Punkt)
Geben Sie das Lösungswort des Quiz aus der Lernnotiz an.

Aufgabe 20 (2 Punkte)

Gegeben Sei die folgende Klasse aus dem Projekt ooadAufgabeTiefeKopie von der Veranstaltungsseite.

```
public class Doppelliste implements Cloneable {
  private List<Punkt> liste1;
  private List<Punkt> liste2;
```

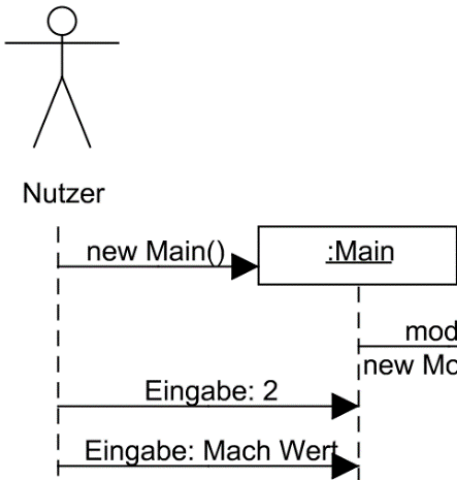
Ersetzen Sie die clone()-Methode, so dass alle Tests in entity.DoppellisteTest laufen. Bei ihrem ersten eigentlich korrekten Implementierungsversuch werden wahrscheinlich zwei Tests scheitern, schreiben Sie auf, warum. Auf dem Weg zur hier korrekten Lösung könnte https://stackoverflow.com/questions/39191522/deep-cloning-while-preserving-shared-references-to-mutable-objects helfen. (Die dort am Ende gemachte Aussage, dass Clonen eine schlechte Idee ist, ist generell falsch.)

Aufgabe 21 (3 Punkte)

Sequenzdiagramme können auch zur Analyse gegebener Programme genutzt werden, indem man typische Abläufe visualisiert. Auf der Internet-Seite der Veranstaltung befindet sich im Projekt ooadAufgabeSequenzdiagramm eine vollständige Implementierung.

a) Leiten Sie zunächst aus dem Programm ein vollständiges Klassendiagramm ab.

b) Zeichnen Sie *ein* genaues Sequenzdiagramm für Nutzereingaben, Ausgaben an den Nutzer und Methoden mit Sichtbarkeit public für folgenden informellen Ablauf: Nutzer startet das Programm, Nutzer erzeugt zwei Controller mit unterschiedlichen Texten, Nutzer erzeugt einen View mit Ausgabezeichen #, Nutzer ändert den Wert des Modells mit dem ersten Controller auf 7, Nutzer erzeugt einen zweiten View mit Ausgabezeichen *, Nutzer ändert Wert des Modells mit dem zweiten Controller auf 5, Nutzer terminiert das Programm.
Im Sequenzdiagramm sind nur alle genutzten Objekte der Klassen Main, Model, View und Controller einzuzeichnen, Collection-Objekte können also weggelassen werden.
Der Anfang des Diagramms könnte wie rechts gezeigt aussehen. Ausgaben können Sie als „Ergebnisausgabe beim Nutzer“ darstellen, direkt aufeinanderfolgende Nutzerinteraktionen könnten Sie auch zu einem Schritt zusammenfassen. Im Sequenzdiagramm müssen die Ausgaben der Views auf dem Bildschirm (Konsole) enthalten sein.



Hinweis: Es kann sinnvoll sein, das Programm zuerst einfach laufen zu lassen, um zu verstehen was das Programm machen soll.

c) Lesen Sie sich in der Dokumentation http://kleuker.iui.hs-osnabrueck.de/querschnittlich/Sequencediagrammer.pdf durch, wie im Sequencediagrammer die Klasse main.Messages ausgeblendet werden kann. Generieren und Speichern Sie mit dem Sequencediagrammer ein Sequenzdiagramm für den in b) beschriebenen informellen Ablauf ohne die Klasse main.Messages. Die Diagramme aus b) und c) werden nicht identisch sein, welche Unterschiede bestehen warum zu Ihrem Sequenzdiagramm zu b).

…

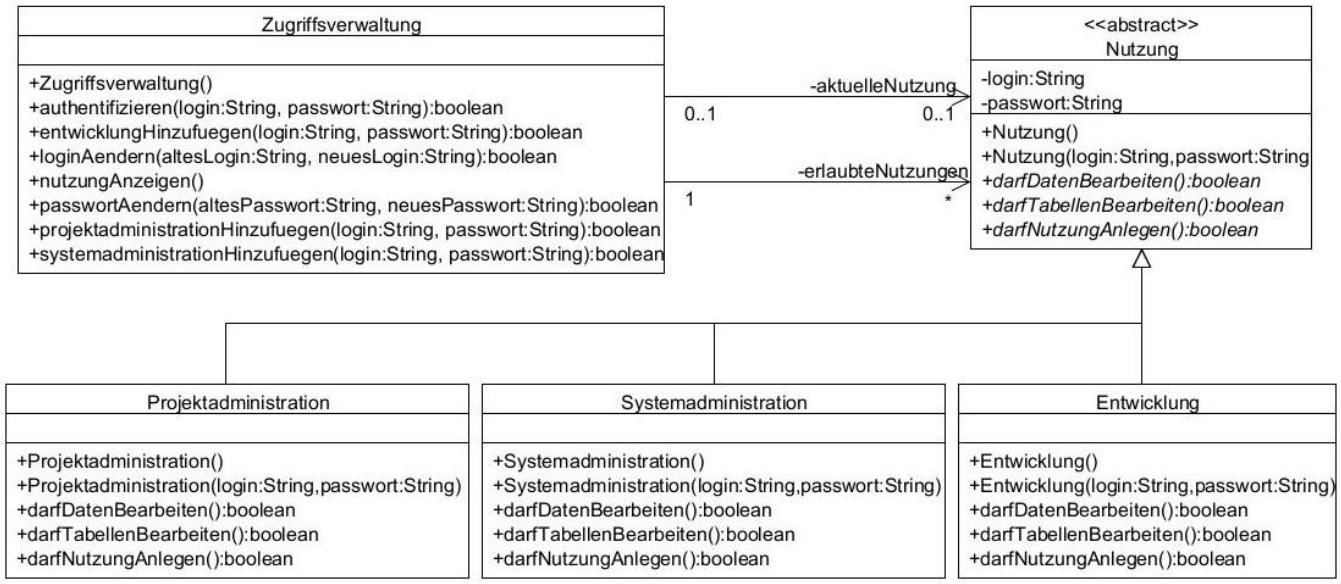

Prof. Dr. Stephan Kleuker
Hochschule Osnabrück
Fakultät Ing-Wiss. und Informatik
- Software-Entwicklung -

Objektorientierte Analyse und Design

Wintersemester 2025
7. Aufgabenblatt

Hinweis: Falls es Sie berechtigt irritieren sollte, warum ein Objekt der Klasse Main mehrfach im Diagramm erscheint, lesen Sie sich den Abschnitt zum Konstruktor in der Dokumentation des Sequencediagrammers durch.

Aufgabe 22 (3 Punkte)



Das vorherige Klassendiagramm zeigt die wesentlichen Klassen einer Zugriffsverwaltung für Nutzungen mit unterschiedlichen Rechten, dabei hat jede dem System bekannte Person (login) ein Nutzungsobjekt mit den zugehörigen Rechten. Die abstrakte Klasse Nutzung enthält die dort angegebenen Exemplarvariablen, Konstruktoren und drei abstrakte Methoden (sichtbar durch die Kursivschrift), die zur Prüfung der im Namen der Methode beschriebenen Eigenschaft dienen. Die abstrakte Klasse Nutzung wird durch die angegebenen drei Klassen realisiert, dabei soll eine Systemadministration alles machen können, eine Projektadministration nur Tabellen und Daten bearbeiten und eine Entwicklung nur Daten bearbeiten. Die get- und set-Methoden der Klassen sind nicht explizit angegeben, existieren aber.
Die Klasse Zugriffsverwaltung verwaltet alle Nutzungen des Systems, wobei immer nur maximal eine Person der zugehörigen Nutzung beim System gemeldet sein kann.

Genauer kann die Klasse Zugriffsverwaltung wie folgt spezifiziert werden.

Field Summary	
private Nutzung	**aktuelleNutzung** aktuelles Nutzungsobjekt der Person, die gerade im System angemeldet ist, am Anfang ist niemand angemeldet.
private „List“ <Nutzung>	**erlaubteNutzungen** Sammlung aller im System vorhandenen Nutzungen. [Sie dürfen einen anderen Collection-Typ nutzen.]

Constructor Summary
Zugriffsverwaltung() Erzeugt Objekt, wobei bereits eine Nutzung, genauer eine Systemadministration, mit login und passwort "admin" als Nutzung am Anfang eingetragen, aber nicht angemeldet wird.


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

Objektorientierte Analyse und Design

Wintersemester 2025
7. Aufgabenblatt

Method Summary	
boolean	**authentifizieren**(java.lang.String login, java.lang.String passwort) Prüft, ob eine eingetragene Nutzung zum übergebenen Paar login, passwort gehört, ist eine solche vorhanden, wird diese zur aktuellen Nutzung, das Ergebnis informiert, ob die Anmeldung erfolgreich war.
boolean	**entwicklungHinzufuegen**(java.lang.String login, java.lang.String passwort) Insofern die aktuelle Nutzung neue Nutzungen anlegen darf, wird eine Entwicklung mit angegebenem login und passwort hinzugefügt, das Ergebnis gibt an, ob das Hinzufügen erfolgreich war.
boolean	**loginAendern**(java.lang.String altesLogin, java.lang.String neuesLogin) Insofern die aktuelle Nutzung neue Nutzungen anlegen darf und eine Nutzung unter dem alten Login existiert, wird das Login auf das neue Login abgeändert, das Ergebnis gibt an, ob die Änderung erfolgreich war.
void	**nutzungAnzeigen**() Zeigt zu jeder eingetragenen Nutzung das Login, das Passwort und die Rechte, ob eine Nutzung angelegt, ob Tabellen angelegt und ob Daten bearbeitet werden dürfen.
boolean	**passwortAendern**(java.lang.String altesPasswort, java.lang.String neuesPasswort) Insofern eine aktuelle Nutzung existiert und das richtige alte Passwort übergeben wird, wird das Passwort auf neuesPasswort geändert, das Ergebnis informiert, ob die Änderung erfolgreich war.
boolean	**projektadministrationHinzufuegen**(java.lang.String login, java.lang.String passwort) Insofern die aktuelle Nutzung neue Nutzungen anlegen darf, wird eine Projektadministration mit angegebenem login und passwort hinzugefügt, das Ergebnis gibt an, ob das Hinzufügen erfolgreich war.
boolean	**systemadministrationHinzufuegen**(java.lang.String login, java.lang.String passwort) Insofern die aktuelle Nutzung neue Nutzungen anlegen darf, wird eine Systemadministration mit angegebenem login und passwort hinzugefügt, das Ergebnis gibt an, ob das Hinzufügen erfolgreich war.

Ihre Aufgabe besteht darin, die im Klassendiagramm angegebenen Klassen zu implementieren und zuerst manuell zu testen. Zur Vereinfachung finden Sie auf der Veranstaltungsseite ein Projekt ooadAufgabeZugriffsverwaltung, das u. a. eine Klasse Zugriffsdialog enthält, die den Zugriff auf ein Zugriffsverwaltungsobjekt über die Konsole steuert. Das Gesamtprogramm wird mit der Klasse Main aufgerufen. Zum Testen rufen Sie dann die Klasse main.AllTest auf, das Ergebnis sollte ein grüner Balken sein. Da Eclipse JUnit 5 (Jupiter) nicht vollständig integriert hat, könnte die Angabe bei „Runs:“ irritieren.
Um das Beispiel klein zu halten, wurde wesentliche Funktionalität, wie eindeutige Logins, das explizite Ausloggen und das Löschen von Nutzungen weggelassen. Wenn es Sie stört, dürfen Sie diese Funktionalität gerne (ohne Punkte, aber mit Anerkennung) ergänzen.