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

Objektorientierte Analyse und Design

Wintersemester 2020/21
7. Aufgabenblatt

Aufgabe 21 (2 Punkte)

Gegeben Sei die folgende Klasse aus dem Projekt ooadAufgabeTiefeKopie.

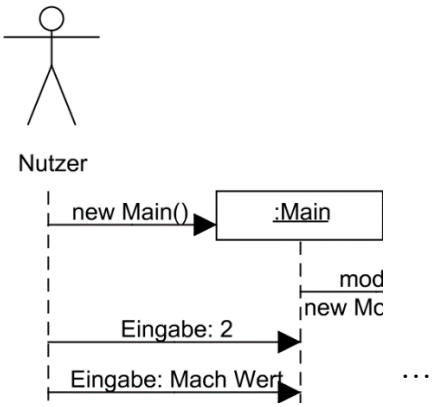
```
public class Doppelliste implements Cloneable{
  private List<Punkt> liste1;
  private List<Punkt> liste2;
```

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.

Aufgabe 22 (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 folgt 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.



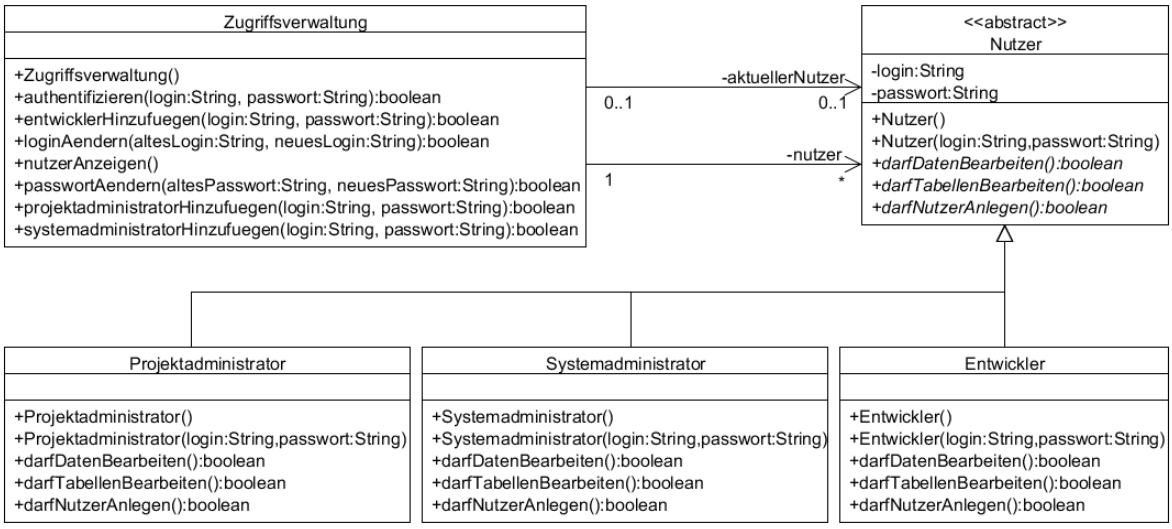
Hinweis: Es kann sinnvoll sein, das Programm einfach laufen zu lassen, um zu verstehen was das Programm machen soll.

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

Objektorientierte Analyse und Design

Wintersemester 2020/21
7. Aufgabenblatt

Aufgabe 23 (4 Punkte)



Das vorherige Klassendiagramm zeigt die wesentlichen Klassen einer Zugriffsverwaltung für Nutzer mit unterschiedlichen Rechten. Die abstrakte Klasse Nutzer 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 Nutzer wird durch die angegebenen drei Klassen realisiert, dabei soll ein Systemadministrator alles machen können, ein Projektadministrator nur Tabellen und Daten bearbeiten und ein Entwickler nur Daten bearbeiten. Die get- und set-Methoden der Klassen sind nicht explizit angegeben, existieren aber.
Die Klasse Zugriffsverwaltung verwaltet alle Nutzer des Systems, wobei immer nur maximal ein Nutzer sich beim System anmelden kann.

Genauer kann die Klasse Zugriffsverwaltung wie folgt spezifiziert werden.

Field Summary	
private Nutzer	**aktuellerNutzer** aktueller Nutzer, der gerade im System angemeldet ist, am Anfang ist niemand angemeldet.
private „List“ <Nutzer>	**nutzer** Sammlung aller im System vorhandenen Nutzer. [Sie dürfen einen anderen Collection-Typ nutzen]

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

Method Summary	
boolean	**authentifizieren**(java.lang.String login, java.lang.String passwort) Prüft, ob ein Nutzer zum eingegebenen Paar login, passwort gehört, ist ein solcher vorhanden, wird er zum aktuellen Nutzer, das Ergebnis informiert, ob die Anmeldung erfolgreich war.

boolean	**entwicklerHinzufuegen**(java.lang.String login, java.lang.String passwort) Insofern der aktuelle Nutzer neue Nutzer anlegen darf, wird ein Entwickler 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 der aktuelle Nutzer neue Nutzer anlegen darf und ein Nutzer unter dem alten Login existiert, wird das Login auf das neue Login abgeändert, das Ergebnis gibt an, ob die Änderung erfolgreich war.
void	**nutzerAnzeigen**() Zeigt zu jedem eingetragenen Nutzer das Login, das Passwort und die Rechte, ob Nutzer angelegt werden, ob Tabellen angelegt und ob Daten bearbeitet werden dürfen.
boolean	**passwortAendern**(java.lang.String altesPasswort, java.lang.String neuesPasswort) Insofern ein aktueller Nutzer existiert und das richtige alte Passwort übergeben wird, wird das Passwort auf neuesPasswort geändert, das Ergebnis informiert, ob die Änderung erfolgreich war.
boolean	**projektadministratorHinzufuegen**(java.lang.String login, java.lang.String passwort) Insofern der aktuelle Nutzer neue Nutzer anlegen darf, wird ein Projektadministrator mit angegebenem login und passwort hinzugefügt, das Ergebnis gibt an, ob das Hinzufügen erfolgreich war.
boolean	**systemadministratorHinzufuegen**(java.lang.String login, java.lang.String passwort) Insofern der aktuelle Nutzer neue Nutzer anlegen darf, wird ein Systemadministrator 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.
Um das Beispiel klein zu halten, wurde wesentliche Funktionalität, wie eindeutige Logins, das explizite Ausloggen und das Löschen von Nutzern weggelassen. Wenn es Sie stört, dürfen Sie diese Funktionalität gerne (ohne Punkte, aber mit Anerkennung) ergänzen.