

## Übungen zur Vorlesung Softwaretechnologie

- Wintersemester 2010/2011 -

Dr. Günter Kniesel

### Übungsblatt 11

Zu bearbeiten bis: 16.01.2010

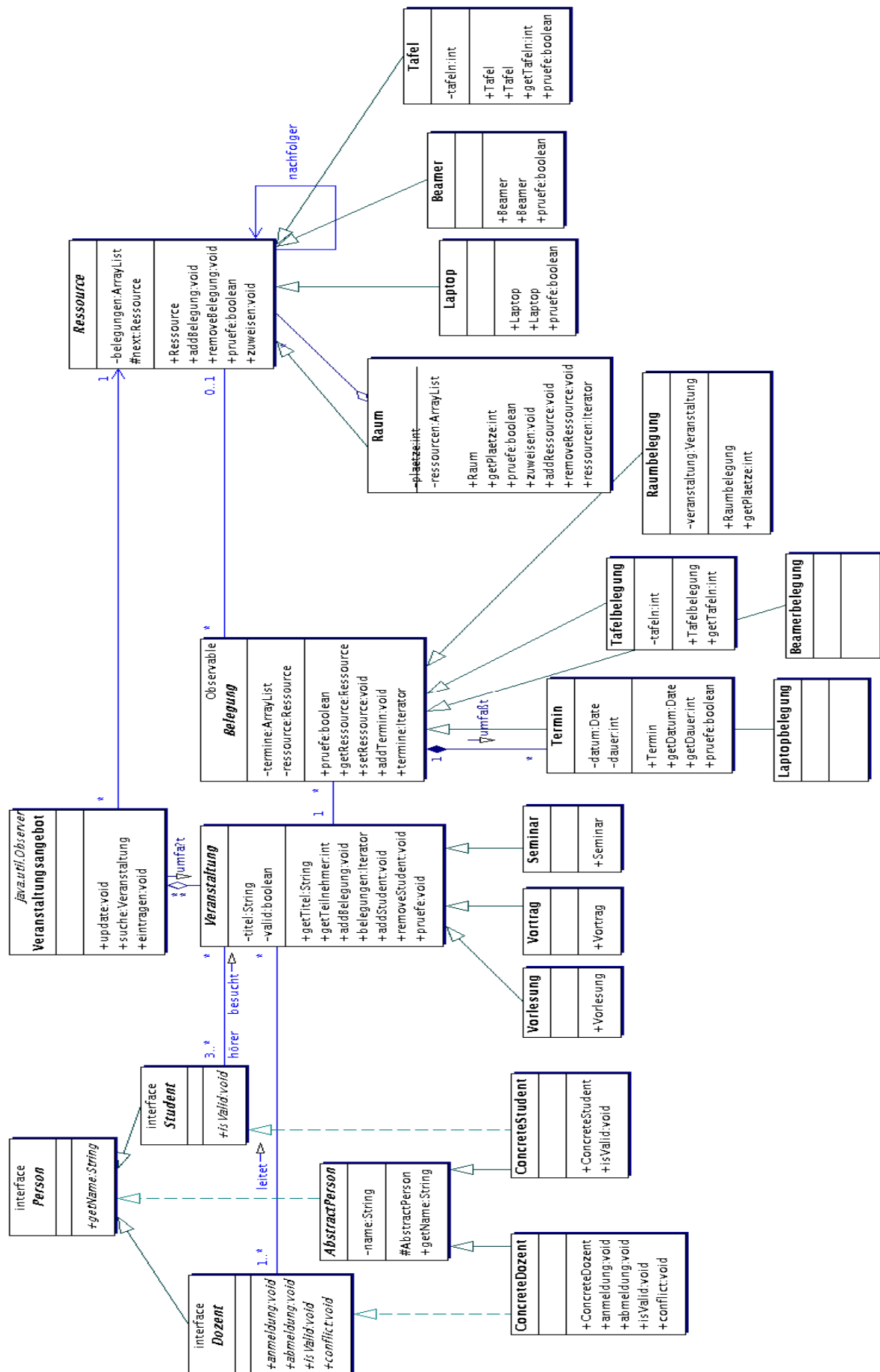
Bitte fangen Sie **frühzeitig** mit der Bearbeitung an, damit wir Ihnen bei Bedarf helfen können. Checken Sie die Lösungen zu den Aufgaben in Ihr SVN-Repository ein, Diagramme als VP-Dateien, Texte als Textdatei. Fragen zu Übungsaufgaben/Vorlesung können Sie auf der Mailingliste [swt-tutoren@lists.iai.uni-bonn.de](mailto:swt-tutoren@lists.iai.uni-bonn.de), bzw. [swt-vorlesung@lists.iai.uni-bonn.de](mailto:swt-vorlesung@lists.iai.uni-bonn.de) stellen.

#### Aufgabe 1. *Design by Contract* (11 Punkte)

- a) Erläutern Sie die drei Kernkonzepte von „Design by Contract“.
- b) Wie können diese Konzepte in Java durch die Verwendung von `assert()`-Anweisungen umgesetzt werden (siehe <http://java.sun.com/developer/technicalArticles/JaVaLP/assertions/>)? Beschreiben Sie zu jedem Konzept kurz die Umsetzung.
- c) Wie würde man mit einem `AssertionError` umgehen, der im Fehlerfall ausgelöst wird? Würde man ihn abfangen und wenn dann wo (direkt nach der Assertion oder an der Stelle die die Methode aufruft, in der sich die Assertion befindet)? Begründen Sie ihre Meinung.
- d) Implementieren Sie für den Entwurf auf der nächsten Seite des Übungsblatts die Methoden
  - a. `Veranstaltung.addStudent(Student s)`,
  - b. `Veranstaltung.removeStudent(Student s)` und
  - c. `Veranstaltungsangebot.eintragen(Veranstaltung v)`.

Garantieren Sie dabei durch die Verwendung geeigneter `assert`-Ausdrücke, dass:

- a. ein Student nicht mehrfach als Teilnehmer einer Veranstaltung eingetragen sein kann,
  - b. ein Student sich nur von einer Veranstaltung abmelden kann, für die er eingetragen ist,
  - c. der Titel einer Veranstaltung im System zur Raumreservierung eindeutig ist (d.h. es darf keine zwei Veranstaltungen mit demselben Titel geben).
- e) Wie würden Sie Vorbedingungen in Java ohne Assertions realisieren? Nennen Sie zwei unterschiedliche Herangehensweisen. Beschreiben Sie die Unterschiede mit Beispiel-Code unter Abwandlung eines der Codestücke aus der vorherigen Teilaufgabe.
- f) In Teilaufgabe e) haben Sie festgestellt, dass es sehr einfach ist DBC in Java zu realisieren, selbst ohne spezielle Sprachunterstützung. Warum glauben Sie, haben sich die Entwickler von Java trotzdem dazu entschlossen, die Sprache um Assertions zu erweitern?



## Aufgabe 2. *Split Objects* (8 Punkte)

Zur weiteren Flexibilisierung des Studiums hat die Kultusministerkonferenz eine beitragsorientierte Reform vorgeschlagen. Als ersten Schritt zur Verwaltung der Studierenden existiert ein Eclipse-Projekt *StudienPlan* (im share-Ordner des SVN-Repositories), welches die Basis-Studiengänge modelliert. Die Klassen enthalten Operationen zur Abfrage der Kostenbeiträge und der Beschreibung eines Studiengangs.

a) Es soll nun möglich sein, zu jedem der obigen Basis-Studiengängen folgende Studien-Optionen zusätzlich zu belegen:

- |                                     |                          |             |
|-------------------------------------|--------------------------|-------------|
| a. <i>Advertising Medium Option</i> | mit Beitragszuschuss von | 75.00 €     |
| b. <i>Tutorial Option</i>           | mit Beitragszuschlag von | 700.00 €    |
| c. <i>Crisis Hotline Option</i>     | mit Beitragszuschlag von | 200.00 €    |
| d. <i>Gratuity Option</i>           | mit Beitragszuschlag von | 16.000.00 € |

Die *Gratuity Option* soll diskret behandelt werden und daher nicht in der Beschreibung auftauchen.

Fügen Sie Klassen für die Studien-Optionen dem existierenden Modell hinzu. Die Klassen sollen Operationen erhalten, die die Kostenbeiträge und eine sinnvolle Beschreibung liefern. Erweitern Sie das Modell (mit Hilfe eines Entwurfsmusters) so, das folgende Anforderungen erfüllt sind:

- Jeder Studiengang soll mit beliebig vielen der obigen Optionen kombinierbar sein.
- Optionen können auch mehrfach (beitragspflichtig) belegt werden.
- Ein Basis-Studiengang und eine Option bilden selbst wieder einen Studiengang.
- Die Berechnung der Kostenbeiträge wird abhängig von dem gewählten Basisstudiengang und den Optionen durchgeführt. Gleiches gilt für die Beschreibung.
- Das spätere Hinzufügen von Basis-Studiengängen und Studien-Optionen soll möglich sein, ohne bereits existierenden Code anpassen zu müssen.
- Es ist nicht notwendig auf bestimmte Komponenten des Studiengangs zugreifen zu können, das Endergebnis (Gesamtbeschreibung des kombinierten Studiengangs und Gesamtbeitrag) ist ausreichend. z. B.:

`Bachelor-Studium, Tutorial Option, Advertising Medium Option – 1125.00€`

- Schreiben Sie ein Programm, das einen Basis-Studiengang mit drei Optionen kombiniert und geben Sie dann die Studienbeiträge und die Beschreibung des resultierenden Studiengangs auf der Konsole aus.
- Gehen Sie nun davon aus, dass jeder Basis-Studiengang mit nur einer einzigen Studien-Option kombiniert werden darf (ansonsten gelten die Anforderungen aus Teilaufgabe a). Würde sich dadurch ein anderes Entwurfsmuster anbieten, um die Aufgabenstellung zu lösen? Begründen Sie stichpunktartig Ihre Antwort.

### Aufgabe 3. Implementierung von Assoziationen (13 Punkte)

Ihr Teamleiter hat Ihnen folgenden Entwurf in die Hand gedrückt mit dem Auftrag ihn zu implementieren.

*Das UML Diagramm wird nachgereicht. Sie können sich in der Zwischenzeit schon mal mit den Funktionen von Paradigm für Forward und Reverse Engineering sowie mit den Vorlesungsfolien zum Thema „Umsetzung von Assoziationen“ vertraut machen, die Sie für nachfolgende Aufgaben benötigen werden.*

a) (2 Punkte) Da Sie wissen, dass Paradigm Codegenerierung (= Forward Engineering) unterstützt, nutzen Sie Paradigm, um das Modell umzusetzen (d.h. um aus dem UML-Modell Code zu generieren).

b) (3 Punkte) Schauen Sie sich den von Paradigm generierten Code genau an und diskutieren Sie die Qualität der Umsetzung der Assoziationen:

- a. Was hätten Sie genauso gemacht?
- b. Was ist unvollständig?
- c. Was hätten Sie grundlegend anders gemacht?

Begründen Sie jeweils Ihre Meinung.

c) (3 Punkte) Wandeln Sie die Implementierung so ab wie Sie es vorgeschlagen haben. Wenn das Ganze funktioniert checken Sie es ein.

d) (2 Punkte) Versuchen Sie, aus Ihrem manuell bearbeiteten Code mit Paradigm wieder ein Modell zu erzeugen (Reverse Engineering). Was geschieht?

e) (3 Punkte) Recherchieren Sie im Internet (max. ¼ Stunde pro Person): Wie steht Paradigm hinsichtlich des Reverse Engineering von automatisch generiertem und anschließend manuell veränderten Code besser im Vergleich zu anderen Werkzeugen da? Identifizieren Sie auf Basis Ihrer Recherche Unterscheidungsmerkmale und bewerten Sie Paradigm und die von Ihnen verglichenen Werkzeuge anhand dieser Merkmale.