Übungen zur Vorlesung

Softwaretechnologie

- Wintersemester 2011/2012 -

Dr. Günter Kniesel

Übungsblatt 11

Zu bearbeiten bis: 15.01.2012

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, „Erklärungen“ bitte als Textdatei. Fragen zu Übungsaufgaben/Vorlesung können Sie auf der Mailingliste swt-tutoren@lists.iai.uni-bonn.de, bzw. 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

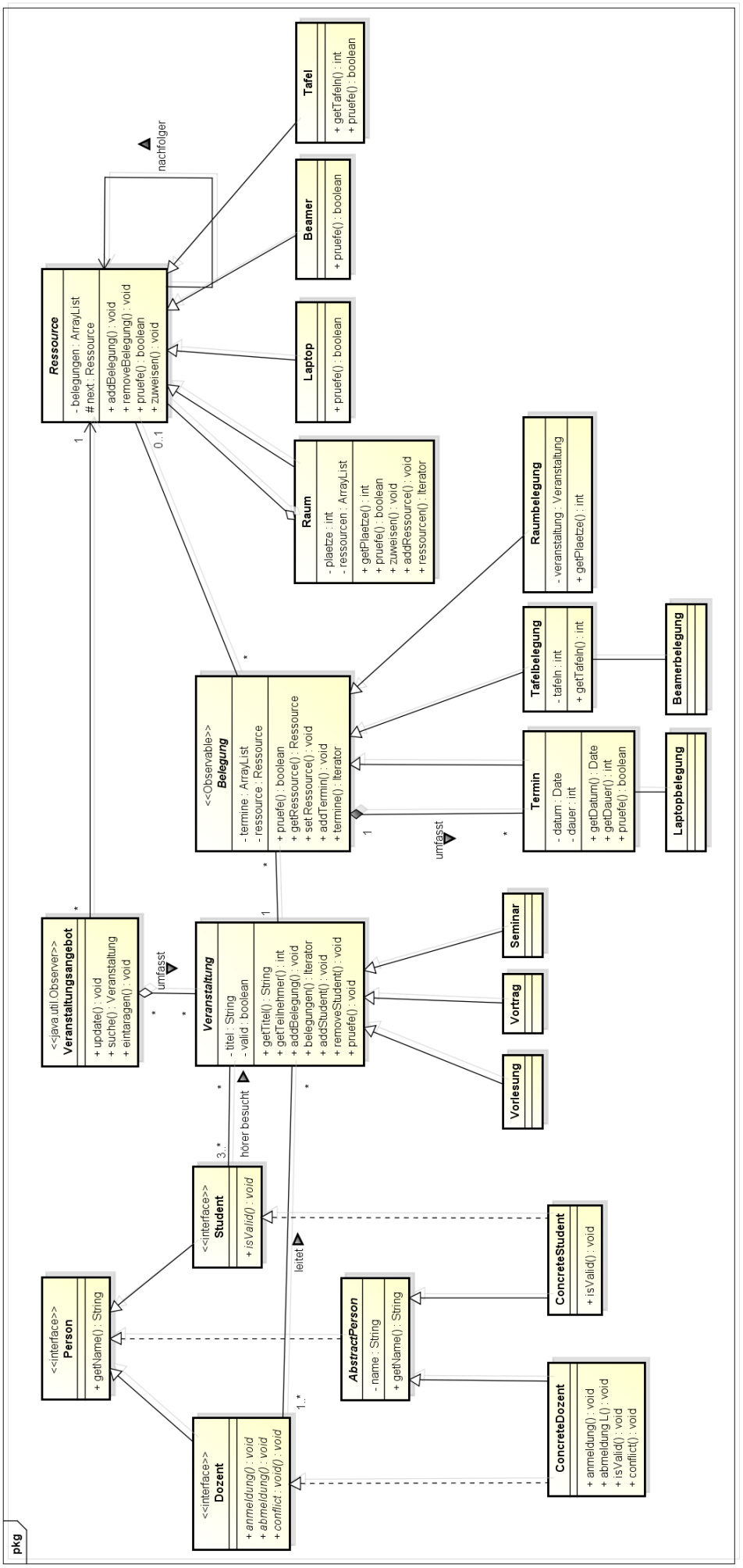
   a. `Veranstaltung.addStudent(Student s)`,
   b. `Veranstaltung.removeStudent(Student s)` und
   c. `Veranstaltungsangebot.eintragen(Veranstaltung v)`.

   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?


pkg
Ressource
- belegungen : ArrayList
next : Ressource
+ addBelegung() : void
+ removeBelegung() : void
+ pruefe() : boolean
+ zuweisen() : void
nachfolger
Tafel
+ getTafeln() : int
+ pruefe() : boolean
Beamer
+ pruefe() : boolean
Laptop
+ pruefe() : boolean
Raum
- plaetze : int
- ressourcen : ArrayList
+ getPlaetze() : int
+ pruefe() : boolean
+ zuweisen() : void
+ addRessource() : void
+ ressourcen() : Iterator
Raumbelegung
- veranstaltung : Veranstaltung
+ getPlaetze() : int
Tafelbelegung
- tafeln : int
+ getTafeln() : int
Beamerbelegung
<<Observable>>
Belegung
- termine : ArrayList
- ressource : Ressource
+ pruefe() : boolean
+ getRessource() : Ressource
+ set Ressource() : void
+ addTermin() : void
+ termine() : Iterator
Termin
- datum : Date
- dauer : int
+ getDatum() : Date
+ getDauer() : int
+ pruefe() : boolean
Laptopbelegung
umfasst
<<java.util.Observer>>
Veranstaltungsangebot
+ update() : void
+ suche() : Veranstaltung
+ eintragen() : void
umfasst
Veranstaltung
- titel : String
- valid : boolean
+ getTitel() : String
+ getTeilnehmer() : int
+ addBelegung() : void
+ belegungen() : Iterator
+ addStudent() : void
+ removeStudent() : void
+ pruefe() : void
Seminar
Vortrag
Vorlesung
hört er besucht
3..*
<<interface>>
Student
+ isValid() : void
leitet
<<interface>>
Person
+ getName() : String
AbstractPerson
- name : String
+ getName() : String
ConcreteStudent
+ isValid() : void
<<interface>>
Dozent
+ anmeldung() : void
+ abmeldung() : void
+ conflict : void() : void
1..*
ConcreteDozent
+ anmeldung() : void
+ abmeldung L() : void
+ isValid() : void
+ conflict() : void

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 readonly-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. *Advertising Medium Option* mit Beitragszuschuss von 75.00 €
   b. *Tutorial Option* mit Beitragszuschlag von 700.00 €
   c. *Crisis Hotline Option* mit Beitragszuschlag von 200.00 €
   d. *Gratuity Option* 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:

   a. Jeder Studiengang soll mit beliebig vielen der obigen Optionen kombinierbar sein.
   b. Optionen können auch mehrfach (beitragspflichtig) belegt werden.
   c. Ein Basis-Studiengang und eine Option bilden selbst wieder einen Studiengang.
   d. Die Berechnung der Kostenbeiträge wird abhängig von dem gewählten Basisstudiengang und den Optionen durchgeführt. Gleiches gilt für die Beschreibung.
   e. Das spätere Hinzufügen von Basis-Studiengängen und Studien-Optionen soll möglich sein, ohne bereits existierenden Code anpassen zu müssen.
   f. Es ist <u>nicht</u> 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€`

b) 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.

c) Gehen Sie nun davon aus, dass jeder Basis-Studiengang mit nur einer <u>einzigen</u> 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.