Rheinische Friedrich-Wilhelms-Universität Bonn
Institut für Informatik III

Übungen zur Vorlesung Softwaretechnologie

- Wintersemester 2019/20 -
Dr. Günter Kniesel

Übungsblatt 7

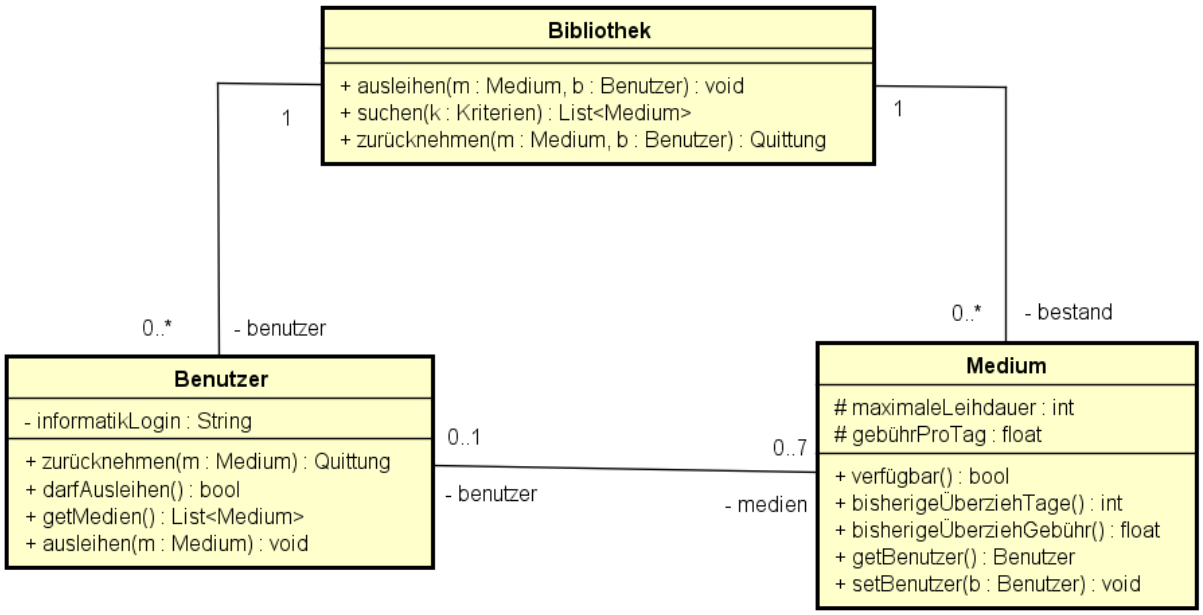
Zu bearbeiten bis: 22.11.2019, 16 Uhr

Bitte fangen Sie **frühzeitig** mit der Bearbeitung an, damit wir Ihnen bei Bedarf helfen können. Checken Sie die Lösungen zu den Aufgaben bitte in Ihr Repository ein, „Erklärungen" bitte als Textdatei. Fragen zu Übungsaufgaben respektive zur Vorlesung können Sie auf der Mailingliste swt-tutoren@lists.iai.uni-bonn.de, bzw. swt-vorlesung@lists.iai.uni-bonn.de stellen.

Praktische Aufgaben

Aufgabe 1. *Design by Contract* — 9,5 Punkte

Betrachten Sie folgendes Klassendiagramm zur Ausleihe bei einer Bibliothek (der äußere Diagrammrahmen wurden aus Platzgründen weggelassen):



Bewertung: Je 0,5 Punkte für jedes gestrichelt unterstrichene Element.
Je 1 Punkt für jedes einfach unterstrichene Element.

a) **(6,5 Punkte)** Erstellen Sie für die nachfolgend beschriebenen Vorgänge zur jeweils passenden Methode je eine Vor- und eine Nachbedingung in OCL:

1) **(4 Punkte)** Sie möchten als Bibliothekar für ein Medium die bisherige Überziehungsgebühr abfragen: Die Abfrage soll immer möglich sein und das Produkt der Überziehungsgebühren pro Tag und der Anzahl der überzogenen Tage liefern.

```
context   Medium::bisherigeÜberziehGebühr()
          pre:     true
          post:    result   = self.gebührProTag * self.bisherigeÜberziehtage()
```

2) **(2,5 Punkte + 2 Bonus)** Ein Benutzer *b* möchte ein Medium *m* ausleihen: Die Ausleihe ist möglich, wenn *m* verfügbar (= nicht ausgeliehen) ist und *b* Medien ausleihen darf. Sobald die Ausleihe erfolgreich beendet ist, gilt, dass *m* an *b* ausgeliehen wurde und somit nicht mehr anderweitig ausleihbar ist.

```
context Bibliothek::ausleihen(Medium m, Benutzer b)
          pre:     m.verfügbar() && b.darfAusleihen()
          post:    !(m.verfügbar())
                   && m.getBenutzer() = b
                   && b.getMedien().contains(m)
```

Da das ursprüngliche Vorgabediagramm die Methoden getBenutzer() und getMedien() nicht beinhaltete, werden die beiden letzten Teilbedingungen der Nachbedingung als Bonuspunkte angerechnet.

b) **(3 Punkte)** Setzen Sie die nichttrivialen Pre- und Postconditions aus Teil a) als Java-Assertions mit sinnvollen Fehlermeldungen um.

1) (1 Punkt) Methode bisherigeÜberziehGebühr():

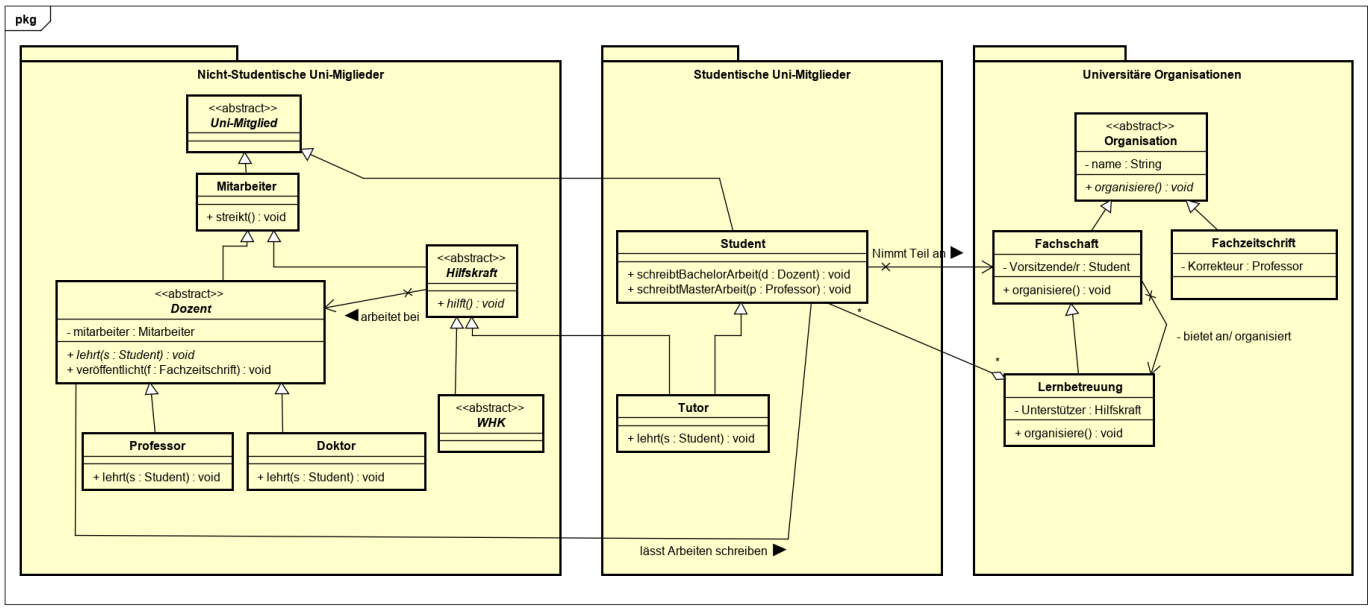
- Vorbedingung: Trivial.
- Nachbedingung:
  assert result == this.gebührProTag * this.bisherigeÜberziehTage():
  „Interner Fehler bei der Berechnung der Überziehungsgebühr.“;

2) (2 Punkte) Methode Ausleihen():

- Vorbedingung:
  assert m.verfügbar() && b.darfAusleihen():
  „Medium ist ausgeliehen oder Sie haben keine Ausleihberechtigung“;
- Nachbedingung:
  assert !(m.verfügbar()) && m.getBenutzer() == b &&
  b.getMedien().contains(m):
  „Interner Fehler beim Ausleihen eines Mediums.“;

Aufgabe 2. *Wiederverwendbarkeit* **13 Punkte**

In der Vorlesung haben Sie die Definition der Abstraktheit und (In)Stabilität für Typen und Pakete kennengelernt. Beantworten Sie dazu die Fragen a) – d) anhand des folgenden Beispiels:



a) **(1 Punkt)** Wie würden Sie bei der Berechnung der Abstraktheit und Instabilität mit leeren Klassen umgehen?

(0,5 Punkte): Da wir laut Formel der Abstraktheit bei einer leeren Klasse durch Null teilen müssten, ist dieser Fall undefiniert. Wenn man versuchen würde, die Definition sinnvoll zu ergänzen wäre eine z.B. eine sinnvolle Variante sich danach zu richten, ob die Klasse im Programmcode selbst als abstract definiert ist (→ 1) oder nicht (→ 0).

(0,5 Punkte): Laut Formel der Instabilität ergibt sich für leere Klassen der Wert Null (Pathologischer Sonderfall: Leere Klasse erbt *explizit* von einer anderen, dann ergibt sich ein Wert > 0. Implizites Erben, wie das erben von Object in Java, wird nicht betrachtet, denn es ist nicht Teil des Programmtextes und muss daher nicht aktualisiert werden, wenn sich zB der Name der Oberklasse ändert).

b) **(1 Punkt)** Wie würden Sie Variablendeklarationen in die Berechnung der Abstraktheit und Stabilität einbeziehen?

(0,5 Punkte): Bei der Berechnung der Abstraktheit würde man nicht-private Variablen wie konkrete Elemente (analog zu nicht-privaten Methoden) behandeln. Man betrachtet nur nicht-private Elemente, denn Abstraktheit soll ein Maß für die Wiederverwendbarkeit sein, und nur die nicht-privaten Teile können von anderen Klassen explizit wiederverwendet werden.

(0,5 Punkte): Bei der Berechnung der Stabilität stellen die Typen von Variablen (aller Variablen, egal ob privat oder nicht) genauso Abhängigkeiten dar wie alle anderen Typdeklarationen im Programm und werden daher genauso gezählt.

c) **(4 Punkte)** Berechnen Sie für die Typen *Student*, *Hilfskraft*, *Organisation, und Lernbetreuung* aus dem obigen Klassendiagramm die Abstraktheit und Instabilität.

Klasse	Abstraktheit	Instabilität	Wiederverwendbarkeit
Student	0/(0+2) = 0	5/(5+6) = 5/11	Tendenziell unwartbar
Hilfskraft	1/(1+0) = 1	2/(2+3) = 2/5	Tendenziell gut wiederverwendbar
Organisation	1/(1+0) = 1	1/(1+2) = 1/3	Tendenziell gut wiederverwendbar
Lernbetreuung	0/(0+1) = 0	3/(3+2) = 3/5	Tendenziell schwer wiederverwendbar

Bewertung: Je 0,5 Punkte pro korrekte Tabelleninhaltszelle (nicht Zeilen- oder Spalten-Beschriftungen

Konkrete Abhängigkeiten zur Berechnung der Instabilität:

Klasse	Efferente Abhängigkeiten	Afferente Abhängigkeiten
Student	Uni-Mitglied, Fachschaft, Lernbetreuung, Dozent, Professor	Dozent, Tutor, Professor, Fachschaft, Lernbetreuung, Doktor
Hilfskraft	Mitarbeiter, Dozent	WHK, Tutor, Lernbetreuung
Organisation	String	Fachschaft, Fachzeitschrift
Lernbetreuung	Fachschaft, Hilfskraft, Student	Student, Fachschaft

d) **(2 Punkte)** Klassifizieren Sie basierend auf Ihren Ergebnissen aus (b) diese 4 Typen bezüglich ihrer Wiederverwendbarkeit in die in der Vorlesung eingeführten Kategorien *gut wiederverwendbar*, *schwer wiederverwendbar*, *nutzlos* und *unwartbar*. Wenn eine Kategorie *k* nicht hundertprozentig passt, sagen Sie gegebenenfalls „tendenziell *k*“.

Siehe letzte Spalte in (b).

e) **(5 Punkte)** Betrachten Sie nun das Gesamtdiagramm. Welche Entwurfsprinzipien (von Foliensatz 5, Folie 66) sehen Sie für die vier in c) und d) betrachteten Klassen als erfüllt oder verletzt an? Begründen Sie jeweils ihre Antwort.

- (1 Punkt) *Dependency Inversion Principle* ist verletzt: Hilfskraft ist vollständig abstrakt, aber von vollständig konkretem Mitarbeiter abhängig.

  ➔ Jede bidirektionale Abhängigkeit zwischen zwei Klassen mit unterschiedlicher Abstraktheit verstößt gegen dieses Prinzip.

- (1 Punkt) *Stable Dependencies Principle* ist verletzt: 55% stabile Klasse Student von instabileren Klasse Lernbetreuung abhängig.

  ➔ Jede bidirektionale Abhängigkeit zwischen zwei Klassen mit unterschiedlicher Stabilität verstößt gegen dieses Prinzip.

- (2 Punkte) *Stable Abstractions Principle* besagt, dass das Verhältnis von Instabilität zu Abstraktheit umgekehrt proportional ist bzw. das Verhältnis von Stabilität zu Abstraktheit proportional (Stabilität = 1- Instabilitätswert). Wir rechnen im Folgenden Abstraktheit / Stabilität, woraus sich ergibt:

  - *Student*: 0 / 0,55 -- grenzwertig, am Rande der „main sequence“
  - *Hilfskraft*: 1 / 0,6 -- relativ OK, weiter in „main sequence“
  - *Organisation*: 1 / 0,7 -- OK, noch tiefer in „main sequence“
  - *Lernbetreuung*: 0 / 0,4 -- grenzwertig, am Rande der „main sequence“

- (1 Punkt) *Acyclic Dependencies Principle* ist verletzt: Zwischen Fachschaft und Lernbetreuung / Student und Lernbetreuung besteht eine zyklische Abhängigkeit.

Aufgabe 3. *Analysephase: Klassen- und Sequenzdiagramm* **8 Punkte**

Eine Firma hat den Auftrag eine Kasse zu entwerfen. Die Kasse soll folgende Elemente enthalten:

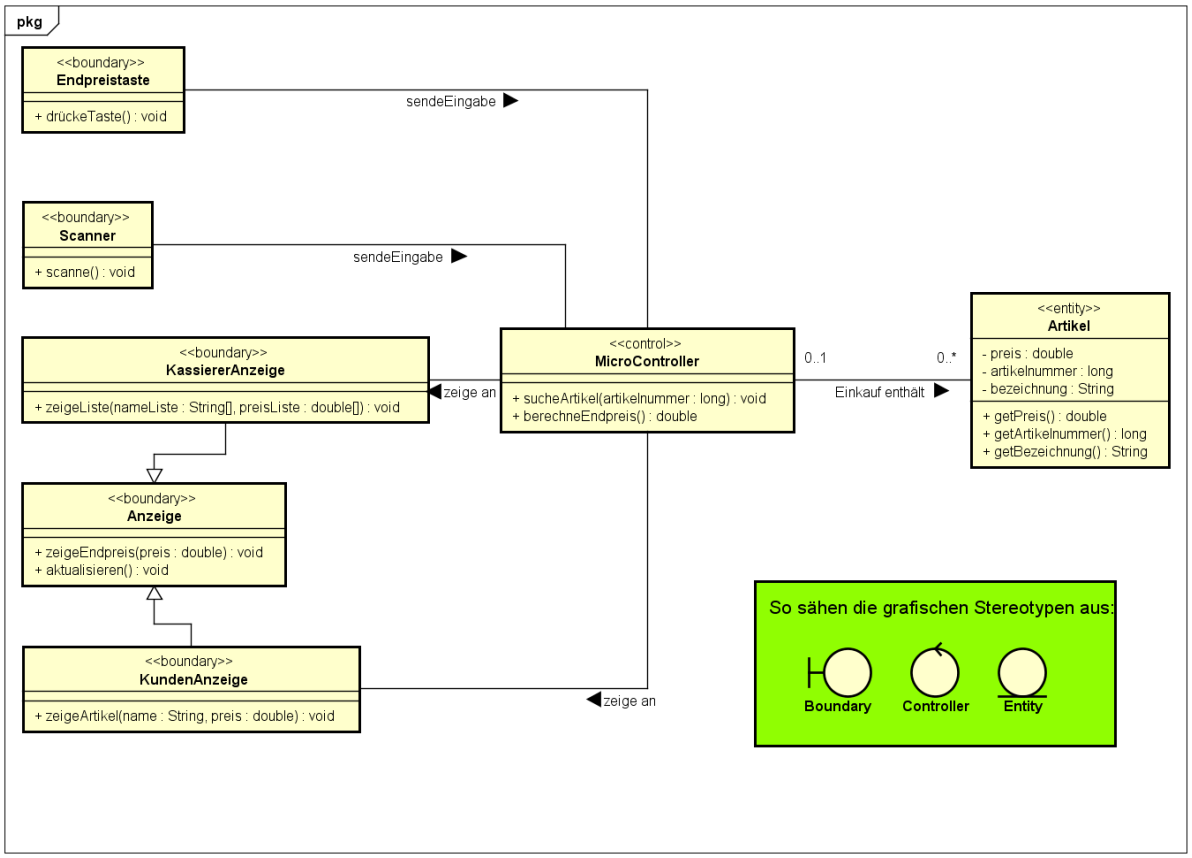
- Ein Scanner für die Erkennung der Artikelnummer
- Eine Taste zum Berechnen des Endpreises
- Eine Listen-Anzeige für den/die Kassierer/in mit einer Auflistung der Artikel des aktuellen Einkaufs (jeweils Bezeichnung und Preis)
- Eine Einzel-Anzeige auf Seite des Kunden, die den Preis des letzten eingegebenen Artikels bzw. den Endpreis anzeigt

Ihre Aufgabe ist nun folgendes:

Erstellen Sie ein Klassendiagramm für die Elemente der Kasse. Überlegen Sie sich Klassen, Operationen und Attribute, sowie Assoziationen, wo es Ihnen sinnvoll erscheint. Unterteilen Sie die Klassen in Boundaries, Entities und Controller.

Hinweis: Astah zeigt bei der grafischen Stereotypen im Klassendiagramm keine Attribute und Operationen an. Nutzen Sie daher „normale“ Klassensymbole mit textuellen Stereotypen und machen Sie als Kommentar eine Anmerkung, wie der entsprechende grafische Stereotyp aussähe (einmal pro Kategorie reicht 😉).

Achten Sie darauf, welche dieser Elemente von den jeweils anderen abhängig sein darf!



6x1 Klassen-Stereotypen mit Methoden + 5x0,5 Assoziationen + 0,5 Bezeichner = 8 Punkte

Theoretische Aufgaben

Aufgabe 4. *Objektorientierte Modellierung* 6,5 Punkte

a) **(1 Punkt)** Was sagen Methoden-Signaturen aus und warum reichen diese nicht zur Modellierung?
Signaturen sagen nur aus, wie man eine Methode aufruft. Was diese macht, wird dadurch noch nicht klar.

b) **(1 Punkt)** Definieren Sie die Begriffe *Assertion* und *Contract* im Kontext von DBC.
Assertion: Eine logische Bedingung, die wahr sein muss.
Contract: Menge aller Assertions, welche die Art, wie zwei Partner miteinander interagieren, festlegt.

c) **(2 Punkte)** Welche Teile des Programms sind jeweils für die Definition bzw. Einhaltung von Vor- und Nachbedingung einer Methode zuständig?
Der aufgerufene Code formuliert die Vor- und Nachbedingung und garantiert die Einhaltung der Nachbedingung, wenn die Vorbedingung wahr ist.
Der aufrufende Code garantiert die Einhaltung der Vorbedingung.

d) **(1,5 Punkte)** Wozu dienen Invarianten und wie werden diese technisch realisiert?
Invarianten garantieren legale Zustände (0,5). Gemeinsame Teilbedingung in allen Nachbedingungen der Methoden einer Klasse, auch Konstruktoren (0,5). Bei Threads zusätzlich am Ende von atomaren Blöcken („synchronized blocs“ in Java) (0,5).

e) **(1 Punkt)** Was bedeuten die Ausdrücke value@pre und value@post?
@pre greift auf den Wert der Variablen vor der Operation zu, @post auf den Wert nach der Operation.

Aufgabe 5. *Anforderungsanalyse* 5 Punkte

a) **(0,5 Punkte)** Welchem Typen im Analyse-Modell entspricht ein Typ des Domain Object Models im Regelfall?

Entity.

b) **(1 Punkt)** Halten Sie *BuchungsTabelle* als Benennung für einen Boundary, der eine Tabelle mit allen Buchungen auf dem Bildschirm ausgibt für sinnvoll? Erklären Sie warum!

BuchungsTabelle ist kein guter Name für einen Boundary, da BuchungsTabelle eher ein Entity darstellt. Des Weiteren sollte man Implementierungsdetails (Tabelle) nicht im Namen verwenden. Ein besserer Name wäre BuchungsBoundary.

c) **(0,5 Punkte)** Was kapselt ein Controller?

Ein Controller kapselt den Ereignisfluss eines Anwendungsfalls.

d) **(1 Punkt)** Welchen Sinn hat die Trennung aller Typen in die drei Analyse-Stereotypen?

Stabilität: Weniger Änderungsanfällig. Boundaries und Controller ändern sich häufig, Entities (und damit die Anwendungsdomäne) sollte möglichst stabil bleiben.

Flexibilität: Leichtere Kombinierbarkeit verschiedener Geschäftsabläufe und Oberflächen.

e) **(1 Punkt)** Welche Analyse-Typen dürfen voneinander abhängig sein?

Boundary <----> Controller ---> Entity

f) **(1 Punkt)** Entspricht das physische Gerät, welches vom Kunden verwendet wird, immer einem Boundary? Begründen Sie!

Nein, das Gerät an sich ist selbst ein Akteur, welcher mit einem Boundary in der Software interagiert.

∑ 42 Punkte