Karlsruher Institut für Technologie
Institut für Theoretische Informatik
Prof. Dr. B. Beckert
Thorsten Bormer

Formale Systeme, WS 2010/2011

Übungsblatt 11

Dieses Übungsblatt wird in der Übung am 11.02.2011 besprochen.

Dieses Übungsblatt beschäftigt sich mit dem folgenden Klassendiagramm:

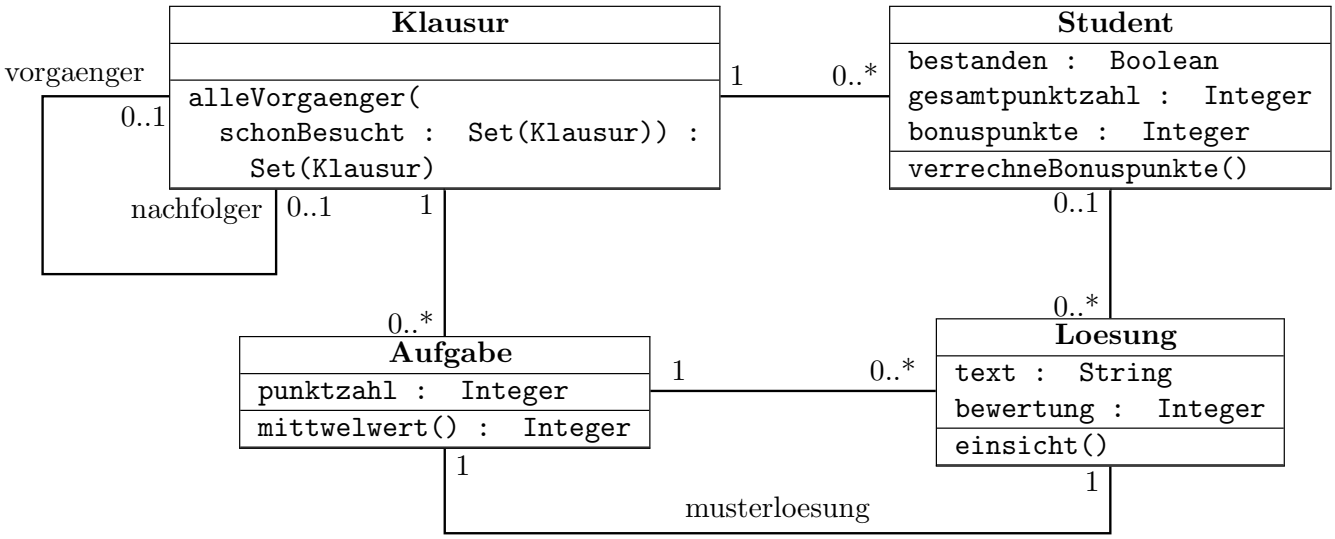

Abbildung 1: UML-Klassendiagramm (`nachfolger` und `vorgaenger` bezeichnen die einer Klausur direkt vorausgehende bzw. nachfolgende Klausur zur selben Vorlesung).

Aufgabe 1

Geben Sie OCL-Bedingungen (Invarianten) an, die die folgenden Sachverhalte modellieren:

(a) Die Lösung jedes Studenten muss mindestens so lang sein wie die der Musterlösung, wenn der Student für diese Aufgabe die volle Punktzahl bekommen hat.

(b) In jeder Klausur gibt es (mindestens) einen Studenten, der bestanden hat, aber nicht zu jeder Aufgabe eine Lösung abgibt.

Folgende Invariante können Sie als gegeben annehmen:

```
context Student
inv: loesung->forAll(l1, l2 | l1.aufgabe = l2.aufgabe implies l1 = l2)
```

(c) Unter allen Klausuren gibt es eine Klausur, bei der weniger als die Hälfte der teilnehmenden Studenten bestanden hat.

Aufgabe 2

Spezifizieren Sie die folgenden Operationen mittels Vor-/Nachbedingungspaaren.

(a) `mittelwert()` berechnet die Durchschnittsbewertung für die Lösungen einer Aufgabe.

(b) Die Methode `verrechneBonuspunkte()` addiert die Bonuspunkte (≥ 0) eines Studenten zur Gesamtpunktzahl, aber nur, falls der Student die Klausur bestanden hat. Zusätzlich wird die Zahl der Bonuspunkte durch die Methode auf 0 gesetzt.

Aufgabe 3

Spezifizieren Sie, dass die `vorgaenger`/`nachfolger`-Relation azyklisch ist. Spezifizieren Sie dafür zuerst die Operation `alleVorgaenger(schonBesucht : Set(Klausur))`. Beachten Sie dabei die folgende Aussage des OCL-Standards:

> The right-hand-side of this [postcondition] definition may refer to the operation being defined (i.e., the definition may be recursive) as long as the recursion is not infinite.

Wie kann man diese Einschränkung auf terminierende Rekursion mittels Parameters `schonBesucht` umsetzen? Was bedeutet sie für die Syntaxdefinition von OCL?

Aufgabe 4

Laut OCL Standard liefert die Operation **reject** mit der Signatur

```
source->reject(iterator | body)
```

die Gesamtheit, die genau die Elemente in `source` enthält, für die `body` zu falsch auswertet. Der Typ des Ergebnisses entspricht dabei dem Typ von `source`.

Geben Sie eine Definition des **reject**-Konstrukts mittels des **iterate**-Konstrukts an. Nehmen Sie dabei an, dass `source` den Typ `Set(T)` hat.