**Institut für Informatik**
**Arbeitsgruppe Software Engineering**
Stephan Salinger, Julia Schenk, Timo Hanisch, Christopher Pockrandt, Marco Träger

**Softwaretechnik SoSe 2013·Übungsblatt 7: OCL**
Bearbeitung zum Tutorium in KW 22

Bereiten Sie Ihre **Lösungen** grundsätzlich so vor, dass Sie diese in der Übung Ihren Kommilitonen in geeigneter Form **zeigen** und **diskutieren** können. Geben Sie bitte stets Ihre verwendeten **Quellen** an.

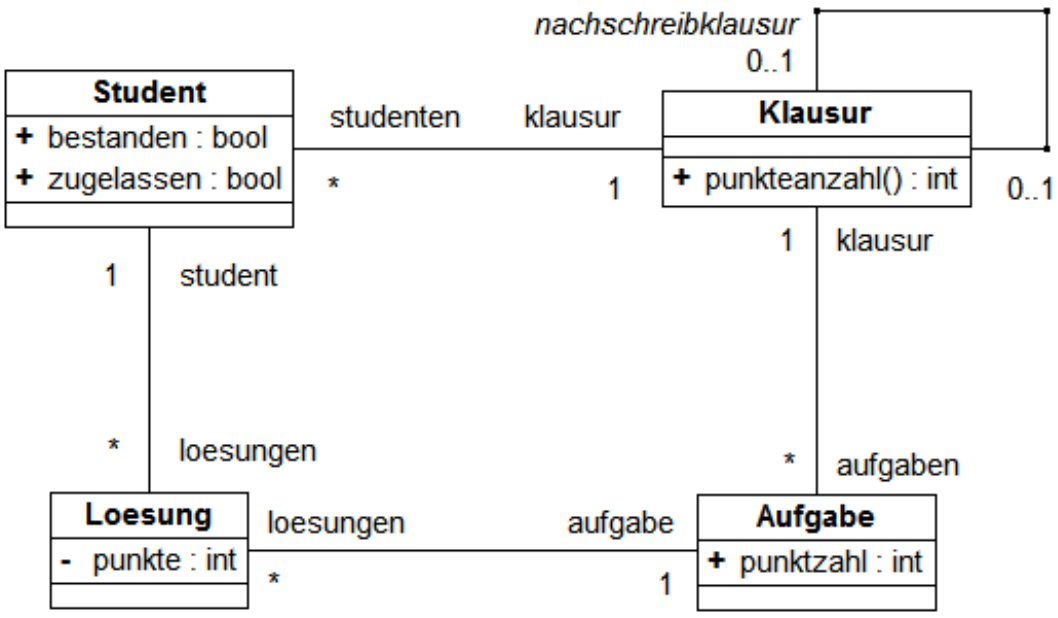
## Aufgabe 7-1 – Begriffe klären

1. Was versteht man im Kontext der Softwareentwicklung unter dem Begriff ***Information Hiding****?*
2. Erläutern Sie Prinzipien und Heuristiken des Information Hiding.
3. In welchem Zusammenhang stehen die Begriffe *Information Hiding* und das „***Need to Know***“ Prinzip?
4. Warum ist das Prinzip des Information Hiding sinnvoll? Erläutern sie dieses an einem Beispiel.
5. Was versteht man unter dem ***Design by Contract*** Prinzip?
    a. In welchem Kontext wird es eingesetzt und warum?
6. Erläutern Sie in welchem Zusammenhang die Begriffe ***OCL, Invariante, Design by Contract, precondition, constraints, postcondition*** stehen.

## Aufgabe 7-2 – OCL

In einem Prüfungsverwaltungssystem sollen die Klausurergebnisse und die Punktzahlen der Studierenden bei den Lösungen der einzelnen Aufgaben verwaltet werden.

Folgendes UML-Klassendiagramm modelliert einen Teil der benötigten Daten.



1. Drücken Sie auf Basis des Diagramms folgende OCL-Constraints natürlichsprachlich aus. `size` berechnet die Elementanzahl einer Menge oder Liste. `self` bezeichnet das Kontextexemplar.

   **a.** `context Aufgabe inv: punktzahl > 0`

   **b.** `context Student inv: loesungen->size() = klausur.aufgaben->size()`

   **c.**
   ```
   context Klausur inv:
         studenten->forAll (s |
               s.bestanden = true implies
               s.lösungen->exists (l |
                     l.punkte > 0 and l.aufgabe.klausur = self))
   ```

2. Geben Sie OCL-Constraints an, die folgende Sachverhalte formalisieren. Schauen Sie auch in der OCL 2.2 Spezifikation (http://www.omg.org/spec/OCL/2.2/PDF) nach, falls Ihnen Ausdrucksmittel fehlen.

   **a.** In jeder Klausur gibt es mindestens eine Aufgabe mit genau einem Punkt.

   **b.** Eine Nachschreibklausur kann keine Nachschreibklausur haben.

   **c.** Pro zugelassenen Student gibt es für jede Aufgabe auch eine Lösung des Studenten.

3. Nun soll zusätzlich auch die Korrektur von Lösungen modelliert werden.

   **a.** Erweitern Sie das Klassendiagramm um ein passendes Attribut und um eine Methode *korrigieren(…)* mit passender Signatur. *korrigieren(…)* wird aufgerufen, sobald der Dozent eine Lösung korrigiert hat. Es soll insbesondere damit möglich sein, dass

   - das Attribut *punkte* in *Lösung* gefüllt wird. (Die Sichtbarkeit des Attributs darf aber nicht verändert werden!)
   - die erreichte Punktzahl des Studenten aktualisiert wird.

   Beschreiben Sie zunächst verbal, was *korrigieren()* genau leisten soll.

   **b.** Geben Sie möglichst strenge Vorbedingungen für Ihre Methode *korrigieren()* in OCL an.

   **c.** Spezifizieren Sie den Effekt (post condition) der Methode *korrigieren()* komplett in OCL.