Institut für Theoretische Informatik
Logik und Formale Methoden
Prof. Dr. P. H. Schmitt,
D. Faragó, M. Ulbrich, B. Weiß

12.11.2010

Praktikum
Formale Entwicklung objektorientierter Software
Übungsblatt 3: JML

Aufgabe 5 — Methodenverträge

Geben Sie in natürlicher Sprache die Bedeutung der folgenden JML-Spezifikation an:

```
//@ assignable x;
//@ signals_only IllegalArgumentException, IllegalStateException;
//@ signals(IllegalArgumentException e) e.getMessage().equals("Oops");
//@ diverges false;
//@ also normal_behavior
//@   requires x >= 0 && i >= 0;
//@   ensures x == \old(x) + i;
Object m(int i);
```

Aufgabe 6 — Invarianten

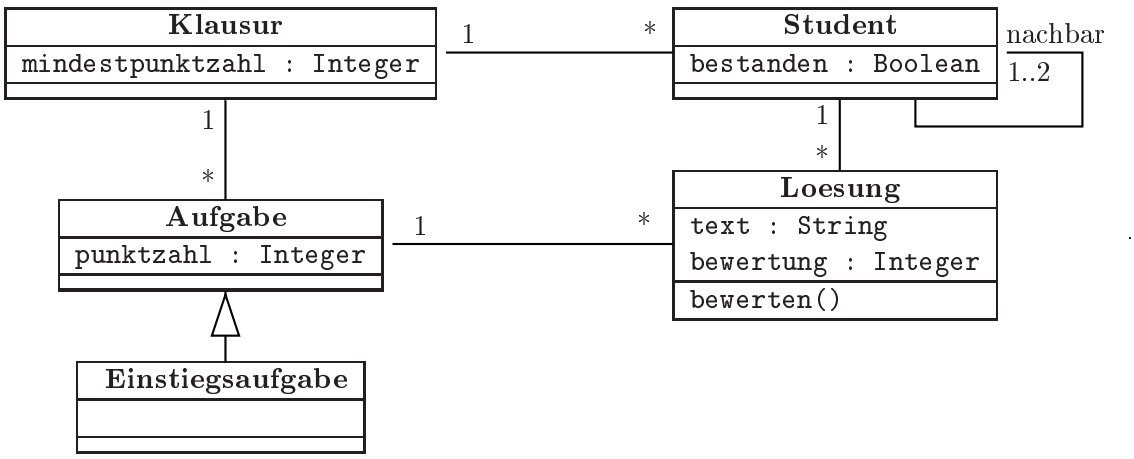
Betrachten Sie die auf der Praktikums-Webseite verfügbare Datei `Invarianten.java`. Welche der Klassen `C1` bis `C4` erfüllen nach der "Visible State Semantics" für Invarianten ihre Spezifikation, und welche nicht? Begründen Sie!

Zur Erinnerung: Die "Visible State Semantics" von JML verlangt, dass eine Instanz-Invariante für ein Objekt o mindestens in folgenden Programmzuständen erfüllt sein muss:

- Am Ende eines Konstruktoraufrufs, der o initialisiert,
- am Anfang und Ende eines Aufrufs einer nicht-statischen Methode auf o,
- am Anfang und Ende eines Aufrufs einer statischen Methode aus der Klasse von o oder einer ihrer Oberklassen, und
- immer wenn *kein* Aufruf einer der drei beschriebenen Arten im Gang ist.

Aufgabe 7 — Spezifizieren von existierendem Code mit JML

Das folgende UML-Klassendiagramm sei gegeben:



Betrachten Sie das auf der Praktikums-Webseite verfügbare Programm `Klausuren.java`. Es implementiert die Klassen des obigen Klassendiagramms.

(a) Übersetzen Sie folgende natürlichsprachliche Beschreibungen in JML-Spezifikationen (Invarianten, Methodenverträge) und fügen Sie diese in das obige Java-Programm ein:

i. Für jede Aufgabe gilt, dass ihre Punktzahl größer als 0 ist.

ii. Die Zahl der Lösungen eines Studenten ist gleich der Zahl der Aufgaben seiner Klausur. Drücken Sie diesen Sachverhalt als Vertrag für die Methode `Student.macheLoesungen(Loesung[] loesungen)` aus.

iii. Die folgende Bedingung soll sich ebenfalls in einem Vertrag für diese Methode wiederfinden: Ein Student hat nicht bestanden, wenn er eine Lösung hat, deren Text mit dem Text einer Lösung eines seiner Nachbarn übereinstimmt.

iv. Für jede Einstiegsaufgabe gilt: Ihre Punktzahl ist kleiner oder gleich der Punktzahl aller Aufgaben der Klausur, zu der sie gehört.

v. Nachdem eine Lösung bewertet worden ist (d.h., nach Ausführung von `bewerten()`), ist ihre Bewertung größer oder gleich 0 und kleiner oder gleich der Punktzahl der Aufgabe, zu der sie gehört.

vi. *Verwenden Sie zur Modellierung des folgenden Sachverhalts das* `\sum`*-Konstrukt:*

Die Summe der Punktzahlen aller Aufgaben einer Klausur ist größer als die Mindestpunktzahl der Klausur.

Stellen Sie mit Hilfe des JML-Syntaxprüfers (Befehl "`jml Klausuren.java`") sicher, dass Ihre Spezifikationen syntaktisch korrekt sind. Achten Sie auch darauf, dass nur Kommentare, deren *erstes* Zeichen ein "`@`" ist, als JML interpretiert werden.

(b) Erfüllt die Implementierung die in den vorherigen Teilaufgaben erstellten Spezifikationen?

Wenn ja, (wie) könnten Sie das nachweisen? Wenn nein, implementieren Sie die `main`-Methode so, dass nach ihrer Ausführung ein Systemzustand eintritt, der eine der spezifizierten Eigenschaften nicht erfüllt und geben Sie an, welche Eigenschaft verletzt wird.

Abgabe bis Mittwoch, 01.12.
Abgabe (als Java-, ASCII- oder PDF-Dateien) per E-Mail an Benjamin Weiß. Es braucht pro Gruppe und Aufgabe nur *eine* Lösung abgegeben werden. Bitte dokumentieren Sie Ihre Lösungen ausreichend und seien Sie darauf vorbereitet, sie auf Nachfrage zu erklären.

Praktikums-Webseite: `http://lfm.iti.kit.edu/keyprakt1011.php`

David Faragó: R. 308, Tel. 608-7322, E-Mail: `farago@ira.uka.de`
Mattias Ulbrich: R. 106, Tel. 608-4338, E-Mail: `mulbrich@ira.uka.de`
Benjamin Weiß: R. 309, Tel. 608-4324, E-Mail: `bweiss@ira.uka.de`