Übungsgruppe: Qianli Wang und Nazar Sopiha

Aufgabe 9-1:

a) **"Need to know":** only if sb needs to access the information, then make it publicly possible. **Information hiding** is the principle of **segregation** of the design decisions in a computer program that are most likely to change, thus protecting other parts of the program from extensive modification if the design decision is changed. In general, we will hide as much details as we can. However, there are, of course, some things that are required to be clear to all users or other components. That is exactly "**Need to know**".

b) **Information hiding** can prevent certain aspects of a class or software component from being accessible to its clients, using either programming language features (like private variables) or an explicit exporting policy.

Example: In the class **Bank,** there is an attribute which is called deposit. It's meaningful that this attribute should be declared in Java with "private". Otherwise, the unauthorized users can change the value of this attribute without hindrances.

c) The principle of "**Design by contract**" is a metaphor on how elements of a software system collaborate with each other on the basis of mutual obligations and benefits.

d) **Design by contract** prescribes that software designers should define formal, precise and verifiable interface specifications for software components, which extend the ordinary definition of abstract data types with **preconditions**, **postconditions** and **invariants** on base of **constraints** by using **OCL**.

Aufgabe 9-2:

a) Für alle: Prüfer vergeben nur ganzzahlige Punkten

c1: Jede Aufgabe hat die entsprechenden Punkte, deren Werte positiv sind. Die Punktanzahl kann jede Person sehen.

c2: In einer Klausur gibt es gleiche Anzahl von Aufgaben wie die Anzahl von der Lösung. Das heißt, alle Aufgaben sollen von dem Kursteilnehmer gelöst werden.

c3: Jeder Teilnehmer, deren Lösung als bestanden bezeichnet wird, soll zur jeder Aufgabe, die sich mit der vorgegebenen Klausur bezieht, mindestens einen Punkt erwerben. Falls zwei Klausuren die gleiche Aufgabe haben, wird die Lösung während der ersten Klausur in der zweiten ungültig.

b)

```
context Klausur inv c4:
    self.aufgaben -> exists(a | a.punktanzahl = 1)
```

```
context Klausur inv c5:
    self.nachklausur -> notEmpty() implies
        self.nachklausur.nachklausur -> isEmpty()
```

```
context Kursteilnehmer inv c6:
    self.zugelassen implies lösungen -> size() = klausur.aufgaben -> size()
```

Aufgabe 9-3:

a) Wir würden die korrigieren() Funktion anders nennen, und zwar aktualisieren(), weil die Funktion erst dann aufgerufen wird, wenn die Lösung einer Aufgabe schon von den Dozierenden korrigiert wird und Punkteanzahl vergeben wird. Das Ziel der Funktion ist die Punktzahl in der Klasse “Lösung” hinzuzufügen und die gegebene Punkten in Gesamtpunkten einzutragen, also die Punkteanzahl des Klausurteilnehmers zu aktualisieren.
Da die Punkten in der Lösung privat sind und nur innerhalb der Klasse **“Lösung”** zugreifbar, soll die Funktion mit der Signatur **+korrigieren(punkte : int)** in der Klasse “**Lösung**” liegen. Die anderen Klassen bleiben unverändert. Gesamtpunktzahl soll in der Klasse Kursteilnehmer liegen. (**gesamtpunktzahl : int**)

b) **<u>Sprachliche Beschreibung:</u>**
PRE: self.punkte = null
Lösung ist zu einer Aufgabe zugewiesen
gegebene Punkten sind >= 0
<= max. erreichbare Punkte für eine Aufgabe
self.teilnehmer.zugelassen

POST: not self.punkte = null
self.punkte = punkte
self.teilnehmer.gesamtpunktzahl += punkte

```
context Lösung::korrigieren(punkte : int)
    pre: self.punkte = null and
         punkte >= 0 and
         not self.aufgabe = null and
         punkte <= self.aufgabe.punktzahl and
         self.teilnehmer.zugelassen

context Lösung::korrieren(punkte: int)
    post: self.punkte = punkte and
         self.teilnehmer.gesamtpunktzahl = self.teilnehmer.gesamtpunktzahl@pre() + punkte
```

c) Keine null-Anforderung und alte Punkte abgezogen, neue addiert

```
context Lösung::korrigieren(punkte : int)
    pre: punkte >= 0 and
         not self.aufgabe = null and
         punkte <= self.aufgabe.punktzahl and
         self.teilnehmer.zugelassen

context Lösung::korrieren(punkte: int)
    post: self.punkte = punkte and
         self.teilnehmer.gesamtpunktzahl = self.teilnehmer.gesamtpunktzahl@pre() + punkte - self.punkte@pre()
```

d)

```
context Klausur::punkteanzahl():int
    post:
        result = self.aufgabe.punktzahl -> sum()
```