Softwarekonstruktion – Übung 1

Organisatorisches

Präsenzübungen

- Kurzvorstellung wesentlicher Fallstricke aus den Hausübungen (ca. 5-10 min)
- Bearbeitung der Aufgaben in Gruppen (ca. 60 min)
- Gemeinsames Besprechen der Aufgaben (ca. 20-25 min)
- Die Tutoren stehen für Fragen bei der Bearbeitung zur Verfügung, es ist aber **keine** Vorrechenübung

Hausübungen

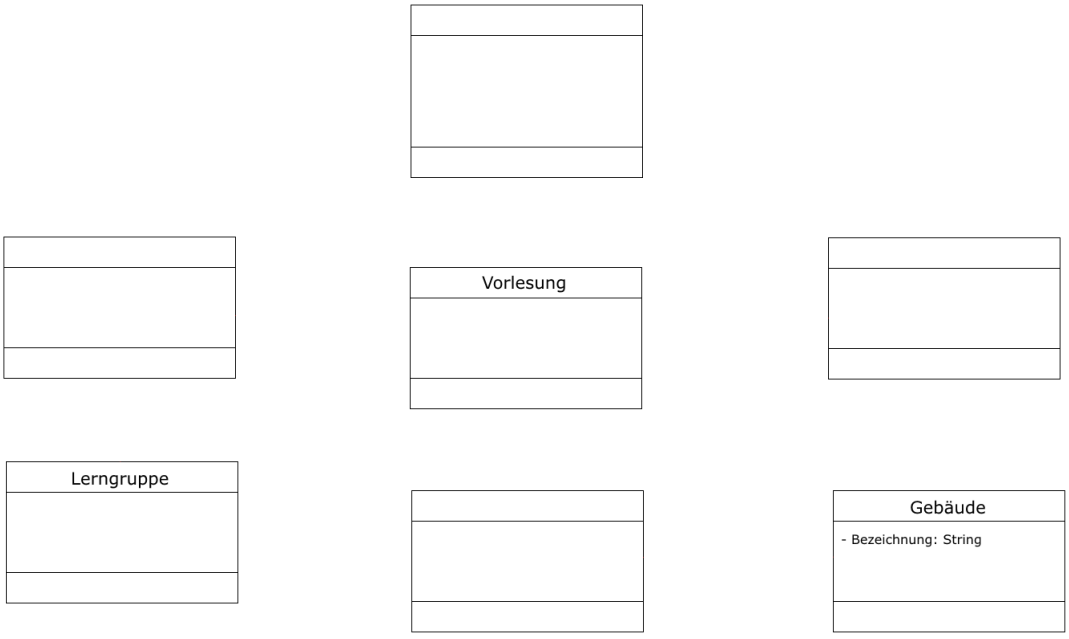
- Übungezettel:
  - Es wird insgesamt 6 Hausübungen mit insgesamt 100 Punkten geben.
  - Bei den letzten beiden Hausübungen handelt es sich voraussichtlich um Online-Übungen.
  - Die Abgabe erfolgt in Gruppen von 2-4 Teilnehmern.
  - Deadlines der Abgaben sowie Informationen zur Online-Übung sind der Webseite der Veranstaltung zu entnehmen.
- Mögliche Abgabe:
  - in den Übungen
  - Einwurf in die Briefkästen Nr. 49-51 (je nach Gruppe) in der Otto-Hahn-Str. 12
- Keine Abgabe:
  - per Mail oder Hauspost
  - persönlich am Lehrstuhl oder Sekretariat
- Rückgabe:
  - in der entsprechenden Übung
  - nicht abgeholte Übungszettel liegen im Sekretariat zur Abholung bereit
- Zum Erbringen der Übungsleistung:
  - Insgesamt sind mind. **50% der Gesamtpunkte** zu erreichen, dabei aber mind. **30% in der 1. Hälfte der Übungszettel** und **30% in der 2. Hälfte der Übungszettel** (d.h. mind. 50 Punkte in Summe, davon mind. 15 Punkte aus den Zetteln 1-3 und mind. 15 Punkte aus den Zetteln 4-6)
- **Abgabe von Duplikaten führt zu 0 Punkten für alle beteiligten Gruppen**

1 Object Constraint Language (OCL)

1.1 UML-Klassendiagramm

Ergänzen Sie das unten stehende UML-Klassendiagramm gemäß nachfolgender Beschreibung. Vervollständigen Sie dabei die Klassen *Person*, *Student*, *Professor*, *Vorlesung*, *Lerngruppe*, *Hörsaal* und *Gebäude* und verbinden diese entsprechend.

Verwenden Sie dabei Aggregationen, Kompositionen, Multiplizitäten und Leserichtungen. Nutzen Sie auch das Konzept der Vererbung. Geben Sie zudem zu allen Attributen passende Datentypen an.



1. Personen besitzen einen Vornamen, einen Nachnamen und ein Alter.
2. Studenten sind Personen, die außerdem eine Matrikelnummer besitzen.
3. Professoren sind ebenfalls Personen, allerdings besitzen sie aufgrund ihrer Anstellung eine Personalnummer und beziehen Gehalt, eine Matrikelnummer besitzen sie natürlich nicht.
4. Studenten besuchen Vorlesungen, Professoren halten Vorlesungen. Professoren sind dabei Lehrende, Studenten die Lernenden. Studenten können einsehen, welche Vorlesungen sie besucht haben, Vorlesungen hingegen haben keinen Zugriff auf die Listen ihrer Besucher.
5. Vorlesungen finden in Hörsälen statt. Jeder Hörsaal besitzt eine Nummer.
6. Ein Hörsaal gehört zu einem Gebäude.
7. Studenten können Lerngruppen bilden.

1.2 OCL Begriffe

Erläutern Sie stichpunktartig nachfolgende Begriffe im Zusammenhang mit der OCL. Beschreiben Sie dabei die Bedeutung der Begriffe und geben Sie entsprechende OCL-Schlüsselwörter an.

1. Klasseninvariante
2. Vorbedingung bzw. Nachbedingung

In der Vorlesung sowie der Übung wird OCL in der Version 2.4 behandelt. Die Spezifikation ist bei Bedarf unter `http://www.omg.org/spec/OCL/2.4/PDF/` zu finden.

1.3 Vordefinierte Operationen

Die OCL ist eine sehr mächtige Sprache, daher wird in der Veranstaltung eine dedizierte Teilmenge der in OCL vordefinierten Operationen betrachtet. Machen Sie sich mit den nachfolgenden Operationen vertraut, indem Sie stichpunktartig beschreiben, was die jew. Operationen berechnen.

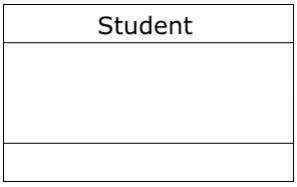
Wenn ein Klassendiagramm gegeben ist, so wenden Sie die entsprechende Operation als gültige Invariante auf alle Klassen an!

- **logische Operatoren**
  - and, or, xor — Konjunktion (und), Disjunktion (oder) sowie Kontravalenz
  - not, implies
  - drücken Sie implies nur mit Hilfe der anderen logischen Operatoren aus:

- **Aufzählungstypen (enumeration types)**
  - *variable* = *enumeration name* :: *enumeration value* Enumerations sind Datentypen der UML, welche die Definition einer Reihe von Literalen als mögliche Werte erlauben. Für den Zugriff auf die Werte ist die :: Syntax zu verwenden.

- **Operationen auf Objekten**
  1. **boolesche Operatoren** (wenn *self* mit Objekt o vergleichbar ist)
     - =(o:T): Boolean — gibt true zurück, wenn *self* gleich dem Objekt o ist
     - <>(o:T): Boolean — gibt true zurück, wenn *self* ungleich dem Objekt o ist
     - <(o:T): Boolean — gibt true zurück, wenn *self* kleiner als o ist
     - <=(o:T): Boolean — gibt true zurück, wenn *self* kleiner gleich o ist
     - >(o:T): Booelan — gibt true zurück, wenn *self* größer als o ist
     - >=(o:T): Boolean — gibt true zurück, wenn *self* größer gleich o ist

- **Operationen auf Collections**
  1. **einfache Operationen**
     - → size(): Integer
     - → sum(): Real
     - → isEmpty(): Boolean
     - → notEmpty(): Boolean gibt true zurück, wenn *Collection* mindestens ein Element enthält
  2. **Elementbezogene Operationen**
     - → includes(o:T): Boolean
     - → excludes(o:T): Boolean gibt true zurück, wenn o nicht in der *Collection* enthalten ist
     - → count(o:T): Integer
     - → isUnique(expr:OclExpression): Boolean
  3. **Iterator-Ausdrücke**
     - → select(expr:OclExpression): Collection(T)
     - → collect(expr:OclExpression): Collection(T)
     - → reject(expr:OclExpression): Collection(T) liefert eine Collection, die alle Elemente von *self* enthält, außer diejenigen, welche die *expr* erfüllen
     - → forAll(expr:OclExpression):Boolean
- **Operationen auf Klassen**
  - classifier.allInstances(): Set(T)



Es soll nicht mehr als 10.000 Studenten geben:

context Student

1.4 Erste Anwendung von OCL

Betrachten Sie das UML-Klassendiagramm über Veranstaltungshallen, wie beispielsweise die Dortmunder Westfalenhalle, in nachfolgender Abbildung 1.

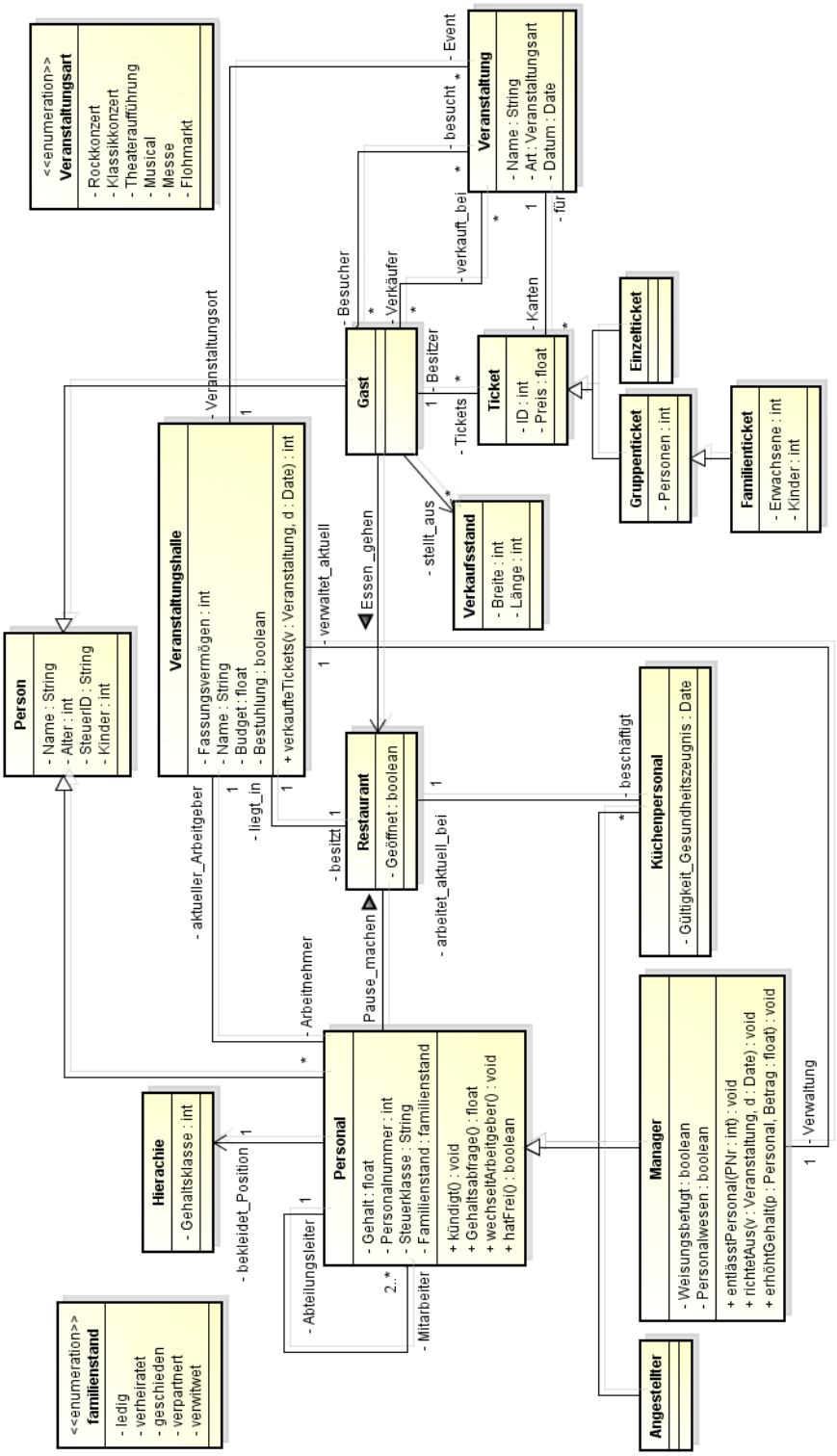


Abbildung 1: UML-Klassendiagramm der Veranstaltungshallen

Das UML-Klassendiagramm in Abbildung 1 erlaubt leider noch die Konstruktion ungewollter Konstellationen. Daher muss die OCL verwendet werden, um notwendige Randbedingungen durch die Spezifikation von Einschränkungen (constraints) zu formulieren. Drücken Sie folgende Constraints in Form von gültigen OCL-Ausdrücken aus

Formulieren Sie in dem Kontext, den die Aufgabenstellung impliziert. Verwenden Sie dabei ausschließlich die Operationen aus Aufgabe 1.3.

1. Das Alter eines Menschen ist immer positiv.
2. Eine Veranstaltung können maximal so viele Gäste besuchen, wie der entsprechende Veranstaltungsort fassen kann.
3. Die Veranstaltungshalle *Westfalenhalle* muss für jede Veranstaltung mindestens 60 Karten für den freien Verkauf bereitstellen.
4. Aus wirtschaftlichen Gründen muss die *Westfalenhalle* sparen. Daher dürfen die Gehälter aller Arbeitnehmer der Westfalenhalle 60% des Budgets der Halle nicht übersteigen.
5. Bei einem Flohmarkt darf es in der Veranstaltungshalle keine Bestuhlung geben.

Hausübung

1.5 Mengen in OCL (5,5 Punkte)

1. Erläutern Sie stichpunktartig die Unterschiede zwischen **Bag**, **Set**, **OrderedSet** und **Sequence**!
2. Geben sie jeweils eine Lösung für die nachfolgenden OCL Ausdrücke an. Schlagen sie bei Bedarf die entsprechende Stelle in der OCL Spezifikation nach. Sollte ein Ausdruck keine gültige Anfrage sein, so notieren Sie dies.

Beispiel: `Set{1,4,7,10} - Set{4,7} = Set{1,10}`

- `Set{1,2,3,4,5}->sum() =`
- `Set{1,2,3,4,5}->size() =`
- `Sequence{'h','a','l','l','o'}->last() =`
- `Bag{'h','a','l','l','o'}->last() =`
- `Set{'h','a','l','l','o',' ','w','e','l','t'}->one(e|e='a') =`
- `Set{'h','a','l','l','o',' ','w','e','l','t'}->select(e|e='l')->size() =`
- `Set{1,2,-1,-2}->isUnique(e|e*e) =`

1.6 OCL Anwendung II (8,5 Punkte)

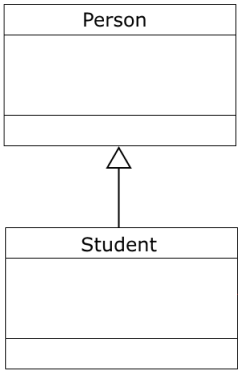
Betrachten Sie das UML-Klassendiagramm zu Veranstaltungshallen aus Aufgabe 1.4. Drücken Sie folgende Constraints in Form von korrekten OCL-Ausdrücken aus!
Verwenden Sie dabei ausschließlich die Operationen aus Aufgabe 1.3!

1. Manager verdienen mehr als 5.000 Euro, Angestellte und Küchenpersonal jedoch maximal 2500 Euro.
2. Jedes Ticket besitzt eine eindeutige ID.
3. Es darf nur bei Flohmärkten Verkäufer geben, bei allen anderen Veranstaltungen sind alle Gäste Besucher! Zudem muss es bei Flohmärkten mindestens einen Verkäufer geben.
4. Bei Flohmärkten dürfen pro Verkäufer in der *Westfalenhalle* nicht mehr als 5 Verkaufsstände aufgestellt werden.
5. Eine Theateraufführung muss mehr als 10 Besucher haben (um nicht abgesagt zu werden)! Zudem darf dann in der Veranstaltungshalle die Bestuhlung nicht fehlen.
6. Kultur soll bereits in der Jugend gefördert werden. Aus diesem Grund müssen Kinder unter 6 Jahren keinen Eintritt zu Klassikkonzerten oder Musicals zahlen, d.h. Tickets dieser Veranstaltungskategorien sind für sie kostenlos.

1.7 Vordefinierte Operationen - Teil 2 (6 Punkte)

Machen Sie sich mit den nachfolgenden Operationen vertraut, indem Sie stichpunktartig beschreiben, was die Operationen berechnen! Wenn ein Klassendiagramm gegeben ist, so wenden Sie die entsprechende Operation als gültige Invariante auf alle Klassen an, d.h. formulieren Sie mittels der Operation alle gültigen Invarianten zu dem Klassendiagramm!

- **weitere Operationen auf Collections**
  1. **boolesche Operatoren**
     - = (c: Collection(T)): Boolean
     - <> (c:Collection(T)): Boolean
  2. **Mengen-Operationen**
     - set → union(c:Collection(T)):Collection(T)
     - bag → union(c:Collection(T)):Bag(T) Die Vereinigung einer ungeordneten Menge mit Duplikaten (bag) mit einer Collection liefert wieder eine ungeordnete Menge, die Duplikate enthalten kann!
     - set → intersection(c:Collection(T)):Set(T)
     - bag → intersection(c:Collection(T)):Collection(T) Der Schnitt einer ungeordneten Menge mit Duplikaten (bag) mit einer Collection liefert eine Menge, welche den Typen der Collection hat!
     - → includesAll(c:Collection(T)): Boolean
- **weiter Operationen auf Objekten**
  1. **Typ vs. Art**
     - self.oclIsTypeOf(typespec): Boolean
     - self.oclIsKindOf(typespec): Boolean



context Person
$inv : self.oclIsTypeOf(Person)$

context Student
$inv : self.oclIsTypeOf(Student)$

2. **undefiniert vs. ungültig**
   - self.oclIsUndefined(): Boolean _____
   - self.oclIsInvalid(): Boolean _____