



## Formale Methoden

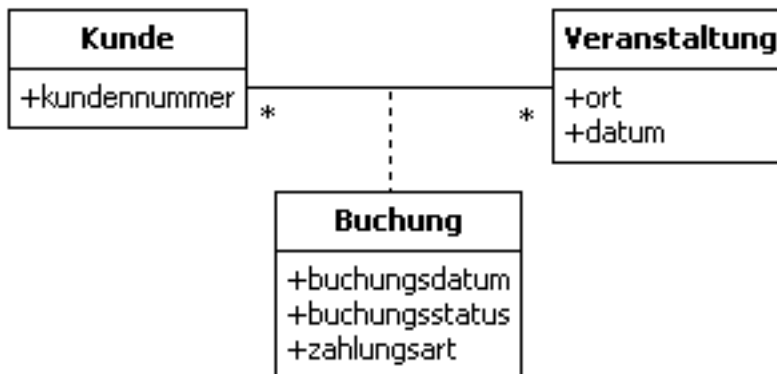
SS 2014 – Übungsblatt 12

Ausgabe: 9. Juli 2014

keine Abgabe (optional, klausurähnliche Aufgaben)

### Aufgabe 1. Assoziationsklassen

Was ist eine Assoziationsklasse wie zum Beispiel:



Welche Vorteile hat sie?

Schreiben Sie die OCL-Constraints:

- Das Buchungsdatum darf nicht länger als zwei Jahre in der Vergangenheit liegen.
- Veranstaltungsdaten dürfen nicht in der Vergangenheit liegen.
- Für Veranstaltungen in den nächsten drei Tagen muss die Zahlungsart `bar` gewählt werden.
- Kunden mit Kundennummern größer als 10000 dürfen die Zahlungsart `Kreditkarte` wählen.

Schreiben Sie ein analoges UML-Diagramm für den Fall, dass Assoziationsklassen von Ihrem UML-Tool nicht unterstützt werden oder beschreiben Sie den in der Vorlesung benutzten Workaround für die aktuelle Papyrus-Version.

## Aufgabe 2. *OclHelper*

Was sind OclHelper? Definieren Sie (in OCL-Syntax) ein Helper-Attribut und eine Helper-Methode für das Modell „Kunde/Buchung/Veranstaltung“ von Aufgabe 1.

Wo überall dürfen sie benutzt werden?

## Aufgabe 3. *OCL-Constraints*

Entwerfen Sie in UML-Form ein Softwaresystem zu:

- Rathaus
- Etage
- Pförtnerloge
- Büro
- Bürger
- Waschraum
- Cafeteria

Zeichnen Sie das zugehörige UML-Klassendiagramm!

Schreiben Sie fünf sinnvolle OCL-Constraints der folgenden Form:

- Jedes Rathaus enthält genau eine Pförtnerloge.
- Jede Etage enthält mindestens einen Waschraum.
- ...

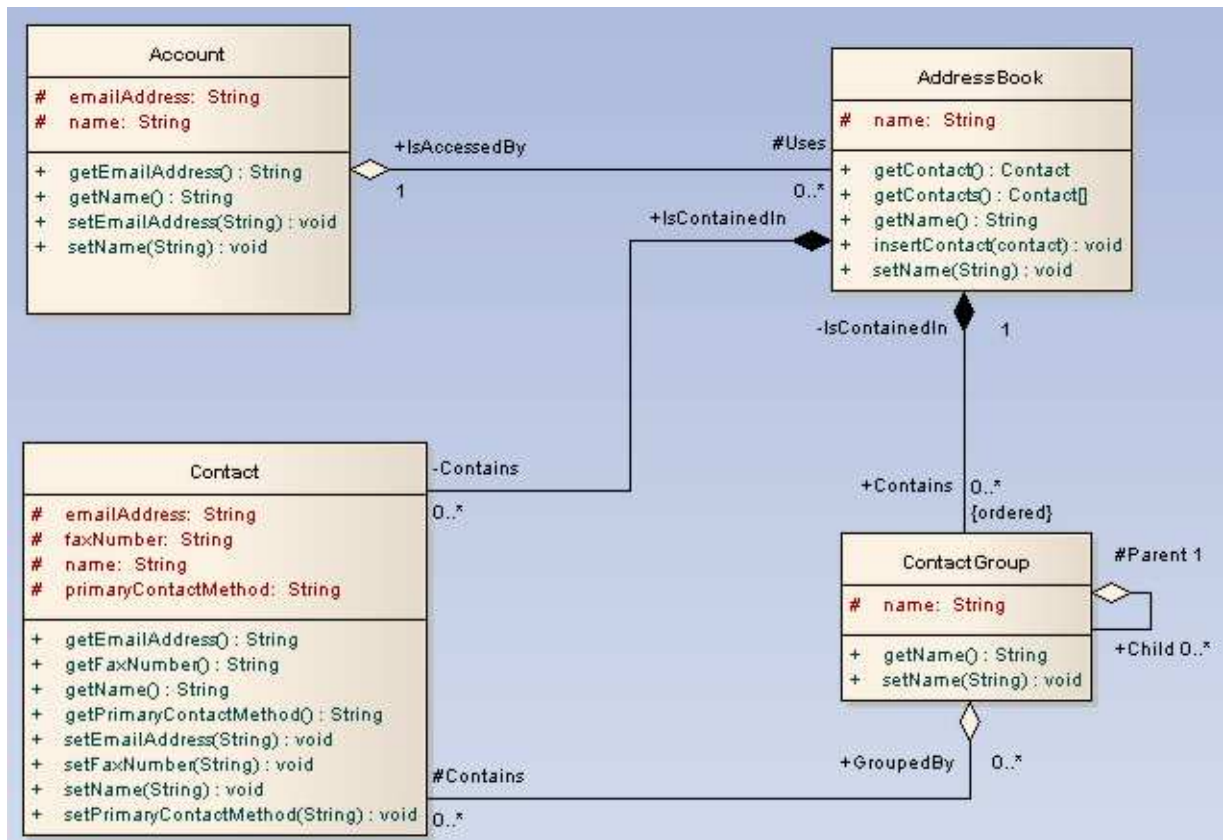
## Aufgabe 4. *(umgangssprachliche) Bedeutung von OCL-Constraints*

Erläutern Sie in eigenen Worten (umgangssprachlich) die Bedeutung der folgenden OCL-Constraints/Ausdrücke:

- **context** Model::Pruefungsergebnisvermerk  
**inv**: student.kursnote[pruefungsnummer] = self
- **context** Model::Pruefungsergebnisvermerk  
**inv** : Model::Pruefungsergebnisvermerk.allInstances()  
    ->isUnique(pruefungsnummer)
- **context** Model::Universitaet  
**inv** : student->size() > 0 **implies**  
    student.kursnote->size() > 0 **implies**  
    student.kursnote->asSet()->isUnique(pruefungsnummer)
- **context** Model::Universitaet  
**inv** : name <> ''  
**inv** : student->size() >= 0  
**inv** : immatrikulation->isUnique(matrikelnummer)  
**inv** : student.universitaet->includes(self)  
**inv** : student[03123456].familienname = 'Bauer'
- **context** Model::Pruefungsergebnisvermerk  
**inv** : kurs <> ''  
**inv** : 1 <= note **and** note <= 5  
**inv** : matrikelNr > 0  
**inv** : student.immatrikulation.matrikelnummer->includes(matrikelNr)  
**inv** : pruefungsnummer > 0

## Aufgabe 5. Aggregationen und Kompositionen

Was ist im Modell



(aus: [http://www.sparxsystems.com.au/resources/uml2\\_tutorial/uml2\\_classdiagram.html](http://www.sparxsystems.com.au/resources/uml2_tutorial/uml2_classdiagram.html))

der Unterschied zwischen den Rollen `Account::Uses` und `AddressBook::Contains`? Was sind die grundlegenden Observatoren der vier Klassen? Schreiben Sie Invarianten und Codeverträge für die Klasse `Account`. Wie sollte der Codevertrag des Destruktors `~AddressBook` aussehen?