Die Verwendung der ***Object Constraint Language (OCL)*** *in UML-Modellen*

Gliederung

- Einleitung
- Grundlegende Prinzipien
  - Was ist ein Constraint?
  - Kontext
  - Arten von Constraints
    - Invarianten
    - Vor- und Nachbedingungen
  - Typen und Operatoren
  - Collections
- Exceptions

Einleitung

- UML kann Randbedingungen und Einschränkungen gar nicht oder schlecht ausdrücken
- Ziele von OCL:
  - genaue Beschreibung des Modells
  - Ausschluss unerwünschter Systemrealisierungen
- OCL Syntax ähnlich zu Java
- formale Grammatik
- aber: OCL nur Spezifikationssprache, keine Programmiersprache
  - → keine Seiteneffekte, System bleibt gleich

Grundlegende Prinzipien

Was ist ein Constraint?

- *„A constraint is a restriction on one or more values of (part of) an object-oriented model or system.“*
- Konvention: Constraints sollen das Modell nicht verändern
- Constraints sind OCL Ausdrücke vom Typ Boolean: Bsp. Klasse Student: `immaNr >= 1000`
- stehen im UML-Modell oder in separatem Dokument
- jeder Constraint ist einem Kontext zugeordnet

Grundlegende Prinzipien

Kontext:

- stellt Bezug der Bedingung auf UML-Modell her
- mögliche Kontexte:
  - Klassen (meistens), Attribute, Operationen, Interfaces, Komponenten
- Syntax: `context Student`
- die Auswertung eines Constraints bezieht sich immer auf eine Instanz des Kontextes
- Schlüsselwort `self` bezieht sich auf diese Instanz

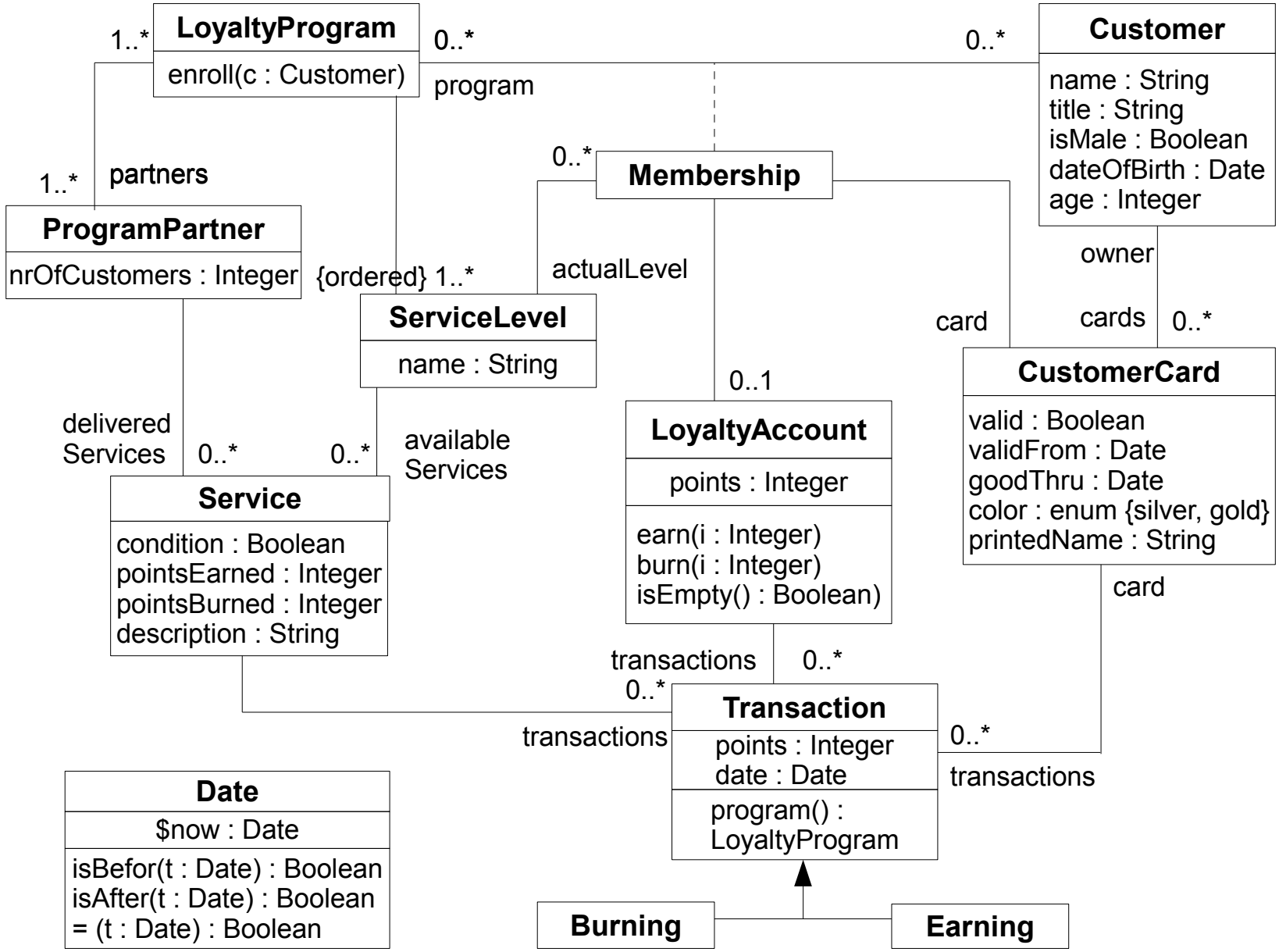
Grundlegende Prinzipien

Welche Constraints gibt es?

- 2 Arten: Invarianten und Vor-/Nachbedingungen
- Invarianten sind zu jedem Zeitpunkt wahr
- Notation: `inv: Bedingung`
- Vor- und Nachbedingungen sind einer Funktion zugeordnet
- nur zu dem Zeitpunkt des Benutzens der Funktion wahr → Design by Contract
- Notation: `pre : Bedingung`

  `-post: Bedingung`

„Royal and Loyal“



Beispiele Invarianten

Invarianten auf Attributen:

```
context Customer inv:
age >= 18
```
(Standard Typ Integer)

Customer
name : String title : String isMale : Boolean dateOfBirth : Date age : Integer

```
context CustomerCard inv:
validFrom.isBefore(goodThru)
```
(benutzt eigene Date Klasse als Typ)

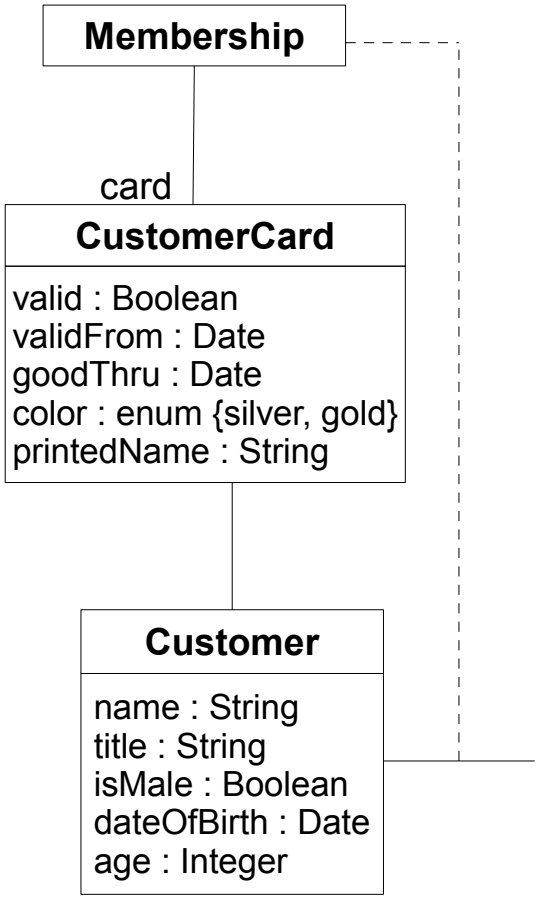
CustomerCard
valid : Boolean validFrom : Date goodThru : Date color : enum {silver, gold} printedName : String

Date
$now : Date
isBefor(t : Date) : Boolean isAfter(t : Date) : Boolean = (t : Date) : Boolean

Beispiele Invarianten

Invarianten auf Assoziationen:

- navigieren durch Modell mittels „.“ +
- Rollenname oder kleiner Klassenname

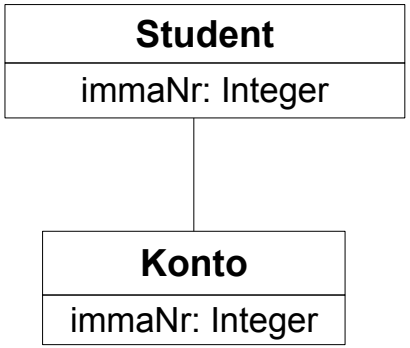



```
context Membership inv:

card.customer = customer


context CustomerCard inv:

printedName =

customer.title.concat(customer.name)
```

Beispiele Invarianten

Invarianten auf Assoziationen mit self:

```
context Student inv:
konto.immaNr = self.immaNr

context Konto inv:
student.immaNr = self.immaNr
```

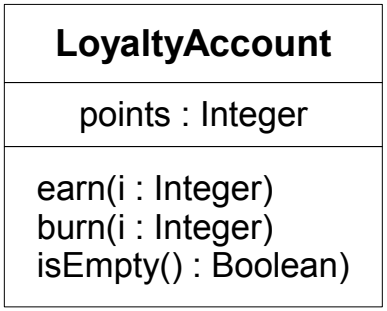


Beispiele Vor-/Nachbedingung

Syntax:

```
context Typname::operationName(parameter1 : Typ1 …) : returnTyp
```

```
context LoyaltyAccount::isEmpty()
pre:    --none
post:   result = (points = 0)
```



- @pre und result in Nachbedingungen:
  - @pre: Wert der Vorbedingung
  - result: Ergebnis der Operation
- Kommentare mitels - - oder /* */

Grundlegende Prinzipien

OCL Typen und einige Operatoren:

- `Boolean:`
  - or, and, xor, not, = (Java: ==), <> (Java: !=), implies, if then else
- `Integer und Real:` keine Wertegrenzen
  - =, +, -, *, /,<, >, abs, max, mod, round, floor
- `String:` 'apple'
  - string.size, string.toLower, toUpper, =, <>
- eigene Typen können ebenfalls implementiert werden → Modell Typen

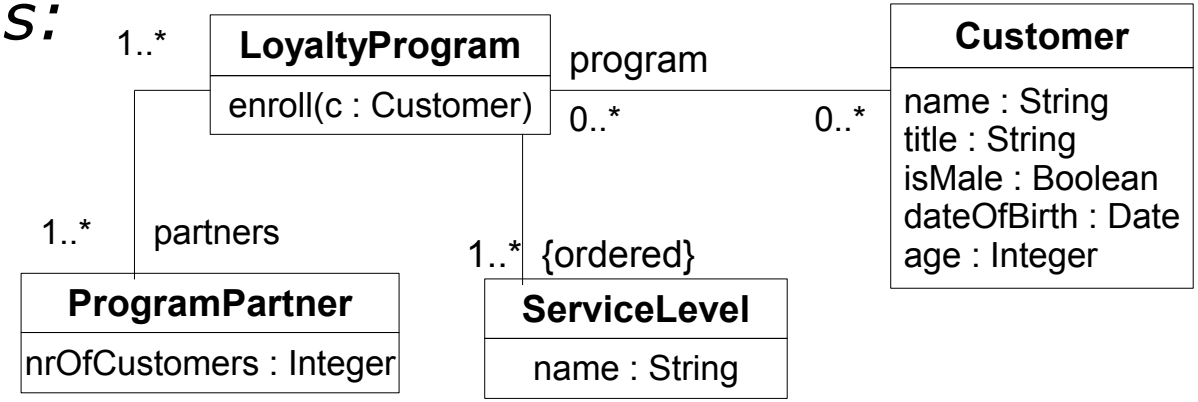
Grundlegende Prinzipien

Collections:

- bei Multiplizitäten > 1 → zu einem Objekt können mehrere andere gehören
- OCL besitzt verschiedene Möglichkeiten der collections: Set, Bag, Sequence
- Set: jedes Element kommt nur einmal vor
- Bag: Elemente auch mehrfach
- Sequence: sortierter Bag
- Operationen: select(), exists(), forAll(), collect()

Beispiele Collections

Collections:




```
context ProgramPartner inv:

nrOfCustomers = loyaltyProgram.customer->size()

                          vs.

context ProgramPartner inv:

nrOfCustomers = loyaltyProgram.customer->asSet()

->size()


context LoyaltyProgram

serviceLevel->first().name = 'Silver'
```

Exceptions

- *Was passiert, wenn Constraints gebrochen werden?*
- bisher in OCL nicht berücksichtigt
- ein gebrochener Constraint invalidiert das gesamte System
- manuell müssen Exceptions ausgewertet werden
- z.B. in Nachbedingungen mittels if zu verzweigen um einen kontrollierten Abbruch hervrzurufen

Quellen

- Kleppe, A/ Warmer, J.: The Object Constraint Language. Precise Modeling with UML. Addison-Wesley 1999.
- http://kaul.inf.h-bonn-rhein-sieg.de/home/script/se1/folien/03_OCL.pdf, Stand 06.05.2010
- http://softech.informatik.uni-kl.de/twiki/pub/Homepage/SeminarSS03/Becker.ps.gz, Stand 06.05.2010