7. Aufgabenblatt

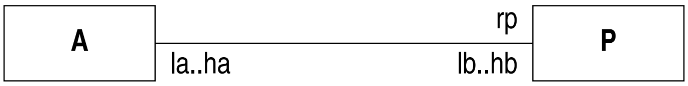
Kurs „Objekt-orientierte Software-Entwicklung“ (SomSem 2002)
Abgabe bis 14.6.2002 / 16^{00} Uhr im Postfach 22 MZH 6. Ebene
Besprechung am 19.6.2002 in der Übung (15-17 Uhr)

Aufgabe 7.1:

Geben Sie stichwortartig die Unterschiede der Aggregation und Komposition gegenüber der allgemeinen Form der Assoziation an.

Aufgabe 7.2:

Die folgenden drei Bedingungen charakterisieren einige mögliche Eigenschaften der Aggregation. Sie beziehen sich jeweils auf zwei Klassen `A` (Aggregate) und `P` (Part), siehe Abbildung. Welche Werte für `la..ha` und `lb..hb` werden durch die Bedingungen jeweils ausgedrückt?



a) *Existential Dependency for the Part:* The part is existentially dependent from the aggregate. In technical terms, this means that a `P` object can only exist if it is connected to an `A` object.

```
P->forAll( p |
  A->exists( a | a.rp->includes(p) ) )
```

b) *Existential Dependency for the Aggregate:* The aggregate is existentially dependent from the part. In technical terms, this means that an `A` object can only exist if it is related to a `P` object.

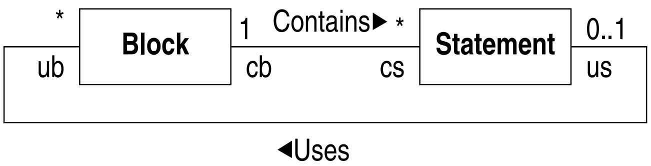
```
A->forAll( a |
  P->exists( p | a.rp->includes(p) ) )
```

c) *Weak Form of Forbidding Sharing:* A `P` object cannot be shared by two different `A` objects. In technical terms, if two aggregates comprise one common part, then the aggregates coincide.

```
P->forAll( p |
  A->exists( a, a' |
    ( a.rp->includes(p) and a'.rp->includes(p) )
    implies
    a=a' ) )
```

Aufgabe 7.3:

Das Diagramm zeigt einen Ausschnitt aus der Modellierung einer Programmiersprache. Ein Block kann Anweisungen enthalten, die wiederum Blöcke von weiteren Anweisungen verwenden (z.B. *if*-Anweisungen).
Bearbeiten Sie die folgenden Aufgabenteile bitte mit USE. Modellieren Sie also zunächst das Klassendiagramm in USE und erzeugen Sie durch die Angabe von Kommandodateien geeignete Objektzustände.



a) Geben Sie ein Objektdiagramm für folgendes Pascal-Programm an:

```
begin
  x := 0;
  if condition then begin
    x := x + 1
  end else begin
    x := x - 1
  end
end
```

b) Gültige Objektdiagramme sollen eine Baumstruktur aufweisen. Warum ist dies durch das Klassendiagramm nicht sichergestellt? Formulieren Sie einen OCL–Ausdruck, der gewährleistet, daß Objektdiagramme immer eine Baumstruktur darstellen.

c) Welche Objektdiagramme wären erlaubt, wenn statt der Multiplizität `0..1` der Wert `1` im Klassendiagramm verwendet würde? (Kommandodatei und Erläuterung)

Aufgabe 7.4:

a) Begründen Sie anhand der formalen Interpretation von Klassendiagrammen, warum die Objektdiagramme (A) - (D) in Abbildung 1 keine gültigen Objektdiagramme zum gegebenen Klassendiagramm sind.

b) Konstruieren Sie aus den folgenden Angaben ein gültiges Objektdiagramm.

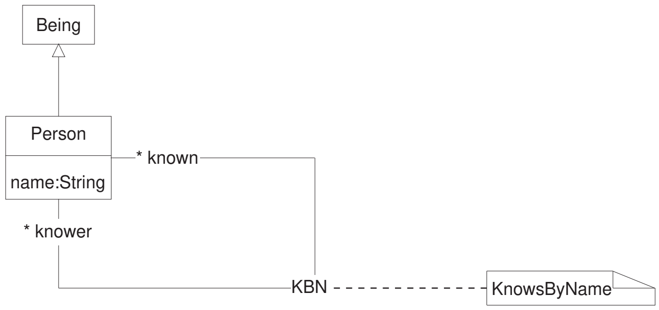
$roles(KBN)$ = ⟨ knower, known ⟩

$\sigma(Being)$ = { b1, p1, p2 }

$\sigma(Person) = \ldots$

$\sigma(KBN)$ = { (p1,p2), (p2,p1) }

$\sigma(name)$ = { (p1 $\mapsto$ ”Harry Potter”), (p2 $\mapsto$ ”Hermine Granger”) }



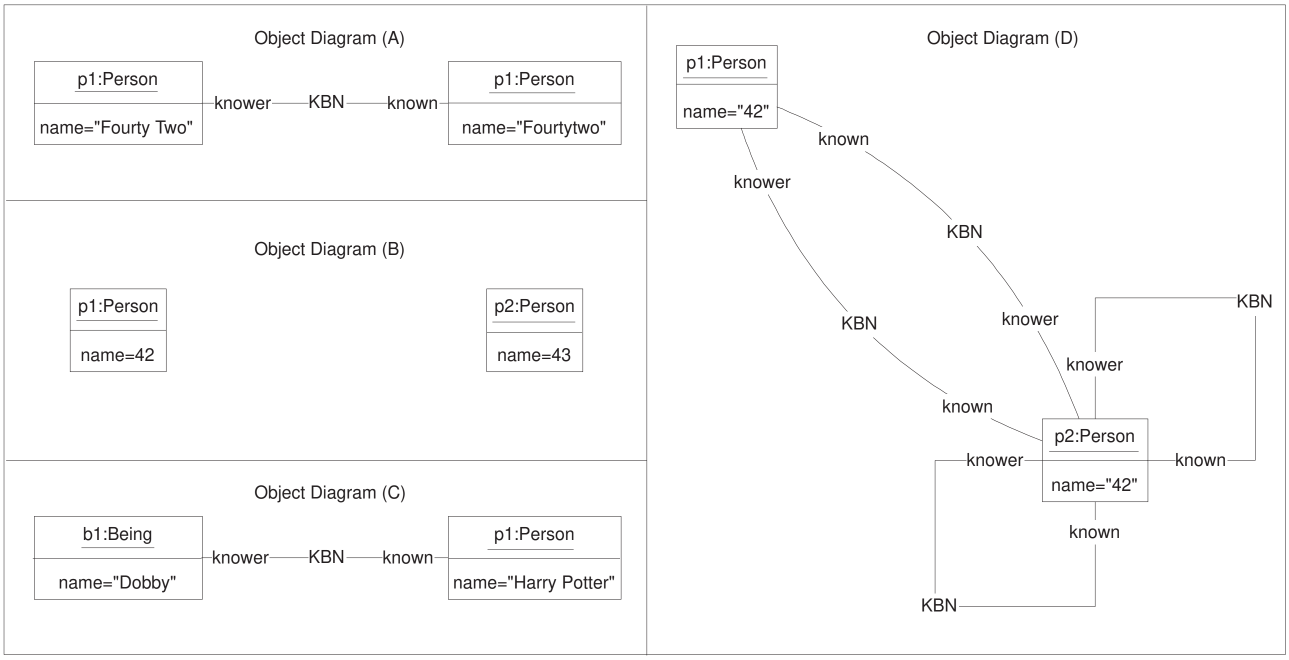


Abbildung 1: Objektdiagramm