Schrankwand modellieren

Vom Schrank zur Schrankwand

durch Modellierung

Schrankwand modellieren

- Die mit copy-and-paste aus der Schrank-Klasse erzeugte Schrankwand-Klasse funktioniert.
- Das allein reicht aber nicht.

Schrankwand modellieren

Ausgangspunkt für eine kritische Auseinandersetzung kann wieder gerade der Aspekt des copy-and-paste sein:

- Wir finden dadurch in zwei Klassen fast identischen Code.
- Es ist ja auch jeweils (fast) dieselbe Aufgabe, nämlich
  - einmal, die Figur eines Schranks zu definieren,
  - im anderen Fall, die Figur eines Schranks in einer Schrankwand zu definieren.

Schrankwand modellieren

Definition eines Ziels:

- Modelliere so die Klassen Schrank und Schrankwand, dass der doppelte Programmcode nur in einer der Klassen auftaucht.

Schrankwand modellieren

Ein Ansatz:

- Man benutzt die Klasse Schrankwand dazu, einen einzelnen Schrank zu erzeugen.
- Ein Schrank ist dann also eine Schrankwand, die aus genau einem Schrank besteht.

Nachteil:

- Die Klasse Schrankwand ist nun nicht mehr für genau eine Aufgabe zuständig,
- das ist also ein Verstoß gegen Kohäsion!

Schrankwand modellieren

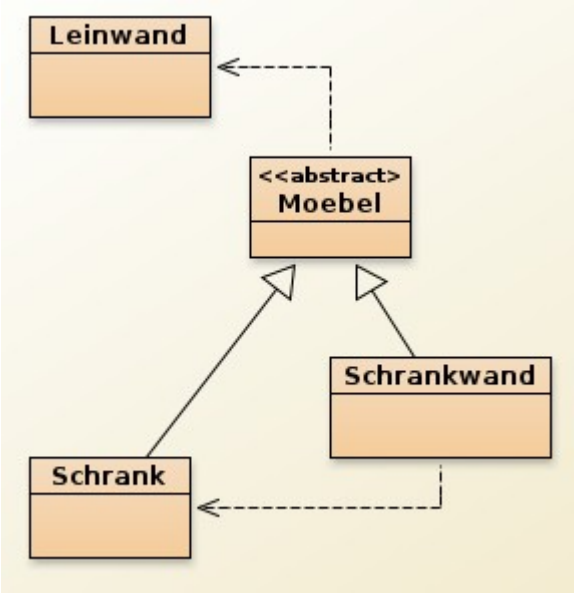
Ein umgekehrter Ansatz:

- Die Klasse Schrank kann Schrankobjekte darstellen.
- Weshalb also soll das auch die Klasse Schrankwand können,
- sie muss dazu doch nur Schrankobjekte verwenden.

Schrankwand modellieren

Schrankwand verwendet Schrank

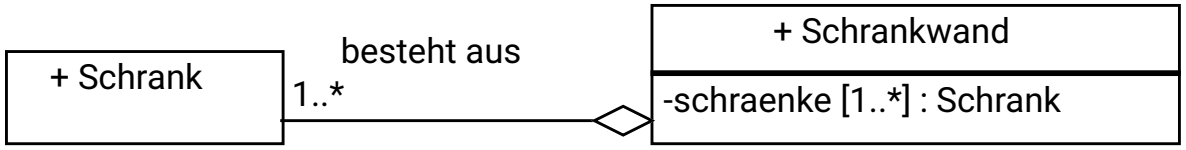
- Die BlueJ-Darstellung (Java) dieser Beziehung:



Schrankwand modellieren

Etwas genauer:

- Ein Schrankwandobjekt ***besteht aus*** Schrankobjekten.
- Die UML-Darstellung dieser Beziehung mit ArgoUML:

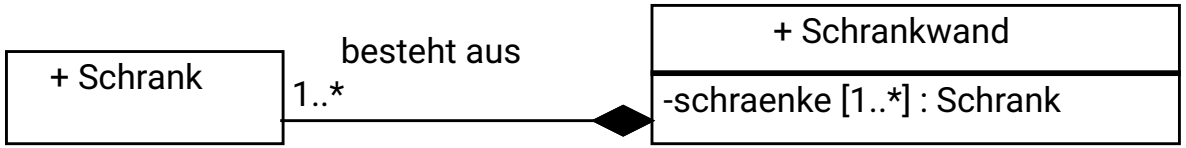


Aggregation

Schrankwand modellieren

Alternativ:

- Ein Schrankwandobjekt ***besteht aus*** Schrankobjekten.
- Die UML-Darstellung dieser Beziehung mit ArgoUML:

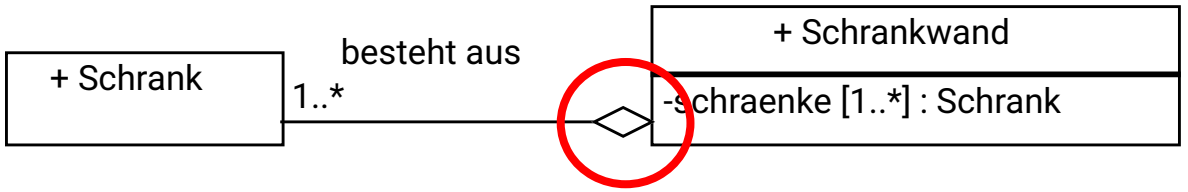


Komposition

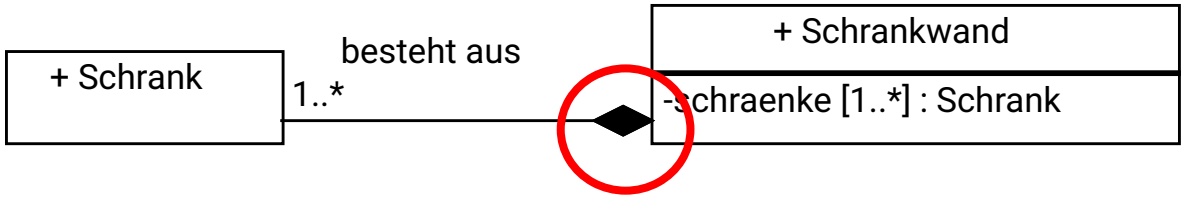
Schrankwand modellieren

Noch zu klären:

- Was unterscheidet die beiden Fälle?



Aggregation



Komposition