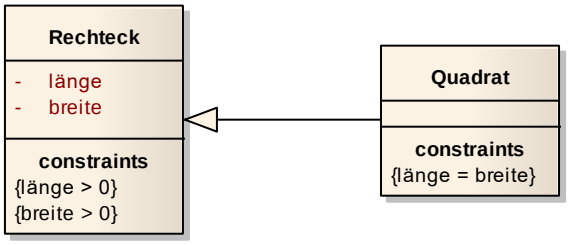
Quadrat (Vererbung)

Lösungsvorschlag

Variante 1 - Quadrat als Unterklasse von Rechteck:

- Das Quadrat ist eine spezielle Form des Rechtecks und wird daher bei dieser Variante als Unterklasse des Rechtecks definiert:

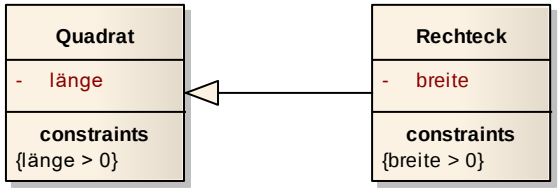


- Vorteile
  - Die Attribute vom Rechteck können wiederverwendet werden.
  - Man benötigt einzig eine zusätzliche Zusicherung, die sicherstellt, dass Länge und Breite gleich gross sind.

- Nachteile:
  - Für die Angabe der Abmessungen benötigt das Quadrat nur die Seitenlänge.
  - Die breite ist redundant. Dies wird aber in Kauf genommen, da nach unserer normalen Vorstellung ein Quadrat eine Spezialisierung eines Rechtecks ist.
  - In der Klasse Quadrat wird eine Zusicherung auf Eigenschaften der Oberklasse gemacht.

Variante 2 - Rechteck als Unterklasse von Quadrat:

- Um die Nachteile der ersten Variante zu beheben, kehren wir die Abhängigkeit um.
- Neu soll das Rechteck als Spezialisierung eines Quadrats modelliert werden:



-

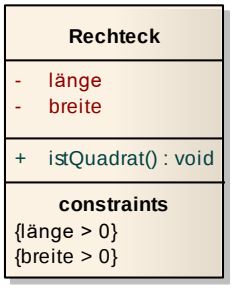
- Vorteile:
  - Das redundante Attribut der ersten Variante gibt es nicht mehr.
  - Die Zusicherungen sind auf die Eigenschaften der jeweiligen Objekte definiert.

- Nachteile:
  - Es kann kein sinnvolles Unterscheidungsmerkmal (Diskriminator) für die Erstellung der Hierarchie angegeben werden.

- Gemäss Definition kann eine Spezialisierung immer anstelle der Oberklasse eingesetzt werden. Wenn wir nun zum Beispiel ein Objekt q1 von Typ Quadrat haben und ein Objekt r1 vom Typ Rechteck, so wäre q1 = r1 eine gültige Zuweisung. Diese Möglichkeit ist aber sicherlich nicht beabsichtigt.

Variante 3 – Quadrat als Besonderheit des Rechtecks

- Als mögliche Lösung für unser Problem kann zum Beispiel folgender Ansatz gewählt werden: Das Quadrat wird gar nicht als eigene Klasse modelliert.



- Stattdessen wird das Quadrat als eine Besonderheit eines Rechtecks betrachtet. Dazu müssen einzig Länge und Breite übereinstimmen.

- Damit dieser Fall unterschieden werden kann, wird die Klasse Rechteck mit der Methode istQuadrat() ergänzt.

Fazit

- Das Beispiel zeigt, dass die Vererbung zwar einfach einzusetzen ist, aber Ihre Tücken hat.

- Generell wird davon abgeraten, in Unterklassen, Zusicherungen auf Attribute und Operationen von Oberklassen zu definieren, da die Unterklasse nicht sicherstellen kann, dass sich alle Methoden der Oberklasse daran halten.

- Anstelle der Vererbung existieren alternative Lösungen (Aggregation, Delegation, Schnittstellen sowie generische und generative Ansätze), mit denen die verschiedenen Probleme der Vererbung umgangen werden können.

- Es gilt daher die Regel / Warnung:

Vermeide Vererbung, wenn es alternative Lösungen dazu gibt.