2.6 Objekte, die andere Objekte kennen

Ein Attribut muss keine Zahl sein. Es kann selbst ein Objekt sein — oder eine ganze Liste davon. Erst dadurch entstehen Modelle, die mehr als ein Ding beschreiben.

Beispiel: Eine Bank hält ihre Konten

```
class Bank:
    def __init__(self, name):
        self.__name = name
        self.__konten = []                      # Liste von Konto-Objekten

    def aufnehmen(self, konto):                 # Auftrag: nimmt ein fertiges
    Konto auf
        self.__konten.append(konto)
        return konto

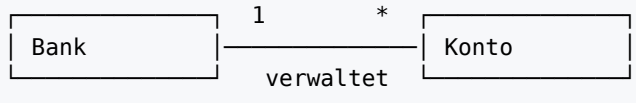
    def eroeffnen(self, inhaber):               # Anfrage: liefert das neue Konto
        return self.aufnehmen(Konto(inhaber))

    def gesamtvermoegen(self):                  # Anfrage
        summe = 0.0
        for konto in self.__konten:
            summe = summe + konto.stand()       # Nachricht an jedes Konto!
        return summe

    def anzahl(self):
        return len(self.__konten)
```

Beachten Sie die Zeile mit `konto.stand()`. Die Bank weiß nicht, *wie* ein Konto seinen Stand verwaltet — sie fragt danach. Das ist Message-Passing in zwei Zeilen: Ein Objekt erfüllt seine Aufgabe, indem es anderen Objekten Nachrichten schickt.

Im UML-Diagramm wird diese Beziehung als Linie zwischen den Klassen gezeichnet, mit der Zahl der beteiligten Objekte an den Enden:



Gelesen: Eine Bank verwaltet beliebig viele (*) Konten; jedes Konto gehört zu genau einer Bank.

Arbeitsauftrag 4

8. Erweitern Sie `Bank` um eine Anfrage `reichster()`, die das Konto mit dem höchsten Stand zurückgibt. Was soll bei einer Bank ohne Konten geschehen?
9. Modellieren Sie eine Schulbibliothek: Welche Klassen brauchen Sie, welche Attribute, welche Nachrichten gehen zwischen den Objekten hin und her? Zeichnen Sie das UML-Diagramm, bevor Sie eine Zeile Quelltext schreiben.