Lösung 4 - Klassendiagramm umsetzen

```
class BankAccount:
    """
    Die Klasse definiert ein Bankkonto mit der Möglichkeit eines negativen
Saldos (Überzug).
    """

    def __init__(self, max_overdrfaft, customer):
        """
        Initialisiert das Objekt als leeres Konto, dem ein Kunde zugewiesen
und
        der Betrag für einen möglichen maximalen Überzug festgelegt wird.
        :param max_overdrfaft: maximaler Überzug für dieses Konto
        :param customer: Referenz zum Kunden
        """
        self._balance    = 0.0; # zwingend 0.0 damit ein float erzeugt wird
        self._overdraft  = max_overdrfaft
        self._customer   = customer

    def booking(self, amount):
        """
        Bucht einen Betrag ins Konto ein.
        :param amount: Betrag der eingebucht wird
        """
        self._balance += amount

    def get_money(self, amount):
        """
        Legt einen Betrag fest, der vom Konto abgezogen werden soll. Dabei
wird
        überprüft, ob der Betrag auf dem Konto verfügbar ist.
        Sollte der angeforderte Betrag zu gross sein, wird kein Geld
geliefert.
        :param amount: Betrag der bezogen werden soll
        :return: Betrag der bezogen werden kann
        """
        if (self._balance + self._overdraft) > amount:
            self._balance -= amount
            return amount
        else:
            return 0.0

    @property
    def balance(self):
        """
        Liefert den aktuellen Saldo des Kontos.
        :return: Saldo des Kontos
        """
        return self._balance
```

```
    @property
    def overdraft(self):
        """
        Liefert den möglich Überzug auf diesem Konto.
        :return: Überzugslimite des Kontos
        """
        return self._overdraft

    @property
    def customer(self):
        """
        Liefert die Referenz des Kunden.
        :return: Referenz zum Kunden
        """
        return self._customer
```

```
class Bottle:
    """
    Die Klasse beschreibt eine Trinkflasche mit den wichtigsten
Eigenschaften und Fähigkeiten.
    """

    def _init__(self, color, capacity):
        """
        Erstellt eine leere Flasche mit der durch color angegebenen Farbe
und
        der vorgegebene Grösse (capacity).
        :param color:
        :param capacity:
        """
        self._quantity_avaiable  = 0.0 # 0.0 da float
        self._color              = color
        self._capacity           = capacity

    @property
    def color(self):
        """
        Liefert die Farbe der Flasche
        :return: Farbe der Flasche
        """
        return self._color

    @property
    def quantity_avaiable(self):
        """
        Liefert die noch verfügbare Mengde an Flüssigkeit.
        :return: verfügbare Menge
        """
```

```
        return self._quantity_avaiable

    @property
    def capacity(self):
        """
        Liefert die Grösse (Kapazität) der Flasche
        :return: Grösse der Flasche
        """
        return self._capacity

    def open_bottle(self):
        """
        ohne Implementation
        """

    def close_the_bottle(self):
        """
        ohne Implementation
        """

    def fill_bottle(self):
        """
        Füllt die Flasche bis sie voll ist.
        """
        self._quantity_avaiable = self._capacity

    def get_liquid(self, amount):
        """
        Liefert die durch amount angegeben Menge, sofern diese verfügbar
ist.
        Ansonsten wird der Rest des Inhalts geliefert.
        :param amount: gewünschte Menge
        :return: verfügbare Menge
        """
        if amount > self._quantity_avaiable:
            amount = self._quantity_avaiable
            self._quantity_avaiable = 0.0
        else:
            self._quantity_avaiable -= amount
        return amount
```

© René Probst

From:
https://wiki.bzz.ch/ - **BZZ - Modulwiki**

Permanent link:
https://wiki.bzz.ch/modul/archiv/m320/learningunits/lu01/loesungen/lu1-aufgabe_3

Last update: **2026/08/30 14:06**