Objekte und Klassen implementieren in Python

Nachdem wir uns nun mit der Objektorientierung als Modellierungsweise beschäftigt haben, wollen wir nun die Implementierung von Klassen und Objekten näher betrachten.

Deklaration von Klassen

Kopfzeile Zum Deklarieren einer Klasse verwendet man das Schlüsselwort *class*. Anschließend folgt der Klassenname und in einer Klammer die Klasse, von der die deklarierte Klasse abgeleitet wird (mehr dazu später). Wird sie von keiner bestimmten Klasse abgeleitet, ist es üblich hier die Klasse *object* (dies ist bei uns zunächst der Fall) einzutragen.

Attribute Bei den Attributen unterscheidet man zwischen Klassenattributen und Instanzattributen. Dabei ist die Frage entscheidend, ob die Attribute beim Instanziieren eines Objekts einen variablen Wert erhalten (also von Objekt zu Objekt unterschiedlich, wie das Kennzeichen eines Autos) oder stets denselben Wert erhalten (wie die Kilomenteranzeige eines Autos, die zu Beginn stets bei 0 ist). Klassenattribute werden mit der Wertzuweisung direkt eingerückt unterhalb der Kopfzeile geschrieben. Instanzattribute werden hingegen erst im Konstruktor festgelegt.

Hinweis: Es ist auch möglich, von dieser Konvention abzuweichen und beispielsweise zunächst allen Attributen den Wert *None* zuzuordnen und erst im Konstruktor den eigentlichen Wert. Dies ist aber nicht unbedingt sinnvoll.

Die Konstruktor-Methode Die Konstruktor-Methode dient dazu, neue Objekte dieser Klasse zu instanziieren. Mit dem Aufruf dieser Methode wird also ein neues Objekt der Klasse erzeugt. Der Name der Methode lautet *__init__* (Achtung: 2 Unterstriche vor und hinter dem init) und die Parameter dieser Methode sind *self*, sowie alle Instanzattribute. Innerhalb der Methode erfolgt dann die Zuweisung der übergebenen Parameter zu den Attributen.

Get- und Set-Methoden Um in der Objektorientierten Programmierung auf einzelne Attribute zugreifen zu können und die Attributwerte entweder auslesen oder verändern zu können, ist es üblich für jedes Attribut eine get- und set-Methode zu erstellen. Diese Methode kennen Sie bereits beispielsweise von den Datenstruktur dynamische Reihung. Allerdings muss man nur sinnvolle get- und set-Methoden erstellen. Handelt es sich z. B. um ein Attribut, welches nach der Instanziierung nie verändert werden würde (z. B. Tankvolumen eines Autos), benötigt man hierfür auch keine set-Methode. Es kann dann auch vom Kontext abhängen, welche get- und set-Methoden alle benötigt werden. Neben den Parametern, die für die jeweilige Methode benötigt wird, muss bei jeder Methode immer auch der Parameter self aufgelistet werden.

Übrige Methoden Alle weiteren Methoden, die man sich bei der Modellierung überlegt hat und in der Klassenkarte notiert hat werden unterhalb der get- und set-Methoden definiert.

Instanziieren von Objekten

Möchte man nun ein Objekt aus der zuvor deklarierten Klasse instanziieren, ruft man die Konstruktor-Methode auf und muss dabei allen Instanzattributen einen Attributwert zuordnen. Die Konstruktormethode ruft man über den Klassennamen auf. Diesen Aufruf muss man dann wie bei einer Variablen in einem Objektnamen speichern.
Möchte man sein Programm übersichtlicher gestalten, kann es auch sinnvoll sein, die Deklaration der Klassen von der Erstellung der Objekte zu trennen und in einer separaten Datei vorzunehmen, welche man dann in die andere Datei mit *import* einbindet.

Deklaration der Klasse Auto

```
class Auto(object):
    tankinhalt = 0
    kilometerstand = 0

    def __init__(self,kennzeichen,tankvolumen,verbrauch):
        self.tankvolumen = tankvolumen
        self.verbrauch = verbrauch
        self.kennzeichen = kennzeichen

    def getTankinhalt(self):
        return self.tankinhalt

    def getKilometerstand(self):
        return self.kilometerstand

    def setKilometerstand(self,kilometerstand):
        self.kilometerstand = kilometerstand

    def getTankvolumen(self):
        return self.tankvolumen

    def getVerbrauch(self):
        return self.verbrauch

    def getKennzeichen(self):
        return self.kennzeichen

    def setKennzeichen(self,kennzeichen):
        self.kennzeichen = kennzeichen

    def tanken(self,menge):
        self.tankinhalt += menge

    def fahren(self,strecke):
        self.tankinhalt -= verbrauch*strecke/100
        self.kilometerstand += strecke
```

Beispiel: Instanziierung und Verwendung eines Objektes der Klasse Auto

```
auto1 = Auto("HH-AA-1234",60,8)
auto1.tanken(30)
auto1.fahren(50)
print(auto1.getTankinhalt())
print(auto1.getKilometerstand())
```

Aufgabe 1

a) Erläutern Sie den Programmcode im Beispiel

b) Erweitern Sie das Beispiel, sodass das Auto nur soweit fahren kann, wie Benzin im Tank ist und der Tank auch nur bis zum Tankvolumen hin getankt werden kann.

Aufgabe 2

Implementieren Sie Ihr eigenes Beispiel.