Matthias Heming 25. November 2009

Hinweise zur Phase »Klassen«

1 Kompetenzformulierung

Schülerinnen und Schüler ...

- unterscheiden Objekte und Klassen. Klassen werden dabei als Kategorie erkannt, der Objekte zugehörig sind. Klassenbeschreibungen werden als Baupläne für Objekte festgehalten.
- nutzen einen Konstruktor zur Erzeugung von Objekten und initialisieren diese passend.
- unterscheiden Anfragen und Aufträge und implementieren solche selbstständig.
- unterscheiden intuitiv private und öffentliche Attribute (optional).

2 Detaillierte Zielsetzung

Das Arbeitsblatt ist gliedert sich in zwei Abschnitte. Die erste Seite ist hauptsächlich theoretischer Natur. Ohne näher auf die Konstruktionstechniken von Klassen(karten) einzugehen, werden Schülerinnen und Schüler »ins kalte Wasser geworfen«. Durch die Erstellung einer Klassenkarte für die Klasse *Buch* kann der Erfolg der Maßnahme geprüft werden.
Im nächsten Abschnitt wird hauptsächlich getippt. Die Implementierung eigener Methoden kann teilweise in Analogie zum vorgegebenen Quelltext stattfinden, ohne Kenntnis über die informatischen Hintergründe. Einige Elemente – wie z. B. der Aufruf eines Konstruktors – sind jedoch nicht auf dem Arbeitsblatt angegeben. Damit soll deutlich gemacht werden, dass dieses Arbeitsblatt nicht für sich alleine steht, sondern nur im regen Kontakt mit der Lehrkraft bearbeitet werden kann. Es sollten während der Bearbeitung des Arbeitsblattes einzelne Phasen eingestreut werden, die Begrifflichkeiten/-Techniken (Konstruktoren, Anfragen/Aufträge, das Wort *self*, Konkatenation, ...) näher erläutern. Zur Ergebnissicherung bieten sich hier Einträge im Merkheft an, auf die später zurückgegriffen werden kann.
Da im Internet häufig Fehlinterpretationen formuliert sind, wird an dieser Stelle explizit darauf hingewiesen, dass die Methode `__init__(self,...)` **kein** Konstruktor ist. Wie der Name bereits andeutet, wird diese Methode mit der Intention genutzt, das bereits konstruierte/erzeugte Objekt zu initialisieren. Mit dem tatsächlichen Konstruktor `__new__` wird man als Anwender in nur sehr speziellen Fällen überhaupt konfrontiert, ein entsprechend künstliches Konstrukt, in welchem den Konstruktor überladen wird, um eine passende Ausgabe zu erzeugen, ist im Folgenden beschrieben[1].

```
class Beispiel(object):
    def __new__(cls):
        print("Konstruktor aufgerufen")
        instance = object.__new__(cls)
        cls.__init__(instance)
    def __init__(self):
        print("Objekt wird initialisiert")
        self.name = "blabla"
```

[1] Detaillierter unter `http://docs.python.org/reference/datamodel.html` – zuletzt geprüft am 15. Oktober 2009

Ebenfalls soll an dieser Stelle darauf hingewiesen werden, dass der Aufruf `del x` nicht direkt den Aufruf der Methode `x.__del__` (dem Destruktor) zur Folge hat. Die Anweisung `del x` löscht den Bezeichner `x` für das dahinterliegende Objekt. Falls aber noch andere Referenzen auf dieses Objekt bestehen, wird `__del__` noch nicht aufgerufen. Bei zyklischen Referenzen kann dieses problematisch werden.
Die letzte Aufgabe greift das bereits bekannte Beispiel des Diktiergerätes auf. Es kann dazu genutzt werden, das mit der vorherigen Aufgabe gelernte zu wiederholen und damit zu festigen. In der nächsten Phase soll anhand dieses Beispiel das Thema von Zuständen und Verzweigungen verdeutlicht werden. Dieses Beispiel wurde aufgrund des Wiedererkennungseffektes und der Einfachheit des Zugangs zu Zuständen/Verzweigungen gewählt.

3 Mögliche Weiterarbeit

Die letzte Frage wirft das Thema der Zugriffsspezifikationen auf. Ohne näher auf die Begrifflichkeiten einzugehen, können Attribute, die *für Außenstehende* wichtig sind, von denen, die nur *für den Programmierer der Klasse* wichtig sind, unterschieden werden. Es ist für den Verlauf des weiteren Unterrichts hier noch nicht notwendig, diese Unterscheidung zu treffen, die Bearbeitung dieser Aufgabe ist daher optional. Falls sie dennoch bearbeitet wird, ist zu überlegen, ob auch auf das *Private Name Mangling* eingegangen wird, welches einen Bezeichner `__foo` innerhalb der Klasse `Bar` in den Bezeichner `_Bar__foo` übersetzt.

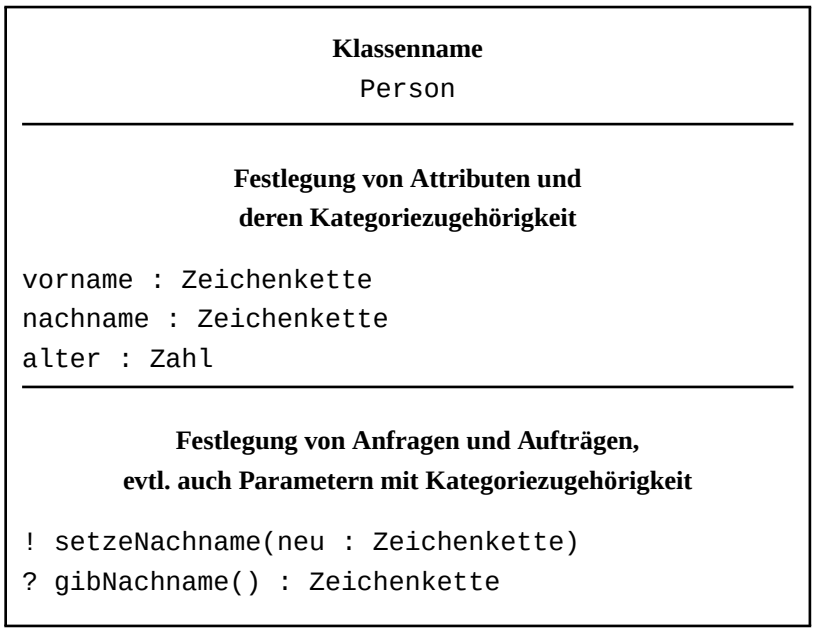
4 Genderaspekt

Bisher keine Besonderheiten gefunden, die zu berücksichtigen wären.

5 Für das Merkheft

Bei der Entwicklung von Klassen treffen wir folgende Abmachungen:

- Klassennamen beginnen immer mit einem großen Buchstaben
- Attribute und Methoden beginnen immer mit einem kleinen Buchstaben
- Bei Klassenkarten werden Klassenname, Attribute und Methoden wie folgt angeordnet:



Klasse: Einen Bauplan oder Schablone, nachdem sich stark ähnelnde Objekte erstellt werden können.

Konstruktor: Eine Methode, die aufgerufen wird, um nach dem vorgegebenen Bauplan tatsächlich ein Objekt zu erstellen. In Python sorgt der Konstruktor nur für die technischen Hintergründe, die für die interne Objektverwaltung notwendig sind. Vor der tatsächlichen Nutzung müssen eventuell noch Attribute initialisiert werden.

Hinweis: In anderen Programmiersprachen wird der Konstruktor häufig mit der Initialisierung zusammengefasst.

Initialisierung: Nachdem der Konstruktor die Hintergrundarbeit zur Objekterzeugung geleistet hat, können – falls notwendig – Startwerte für die Attribute des neuen Objektes eingestellt werden. Dies geschieht durch die automatisch aufgerufene Methode `__init__`. Falls keine Initialisierung benötigt wird, keine die Methode bei eigenen Klassen auch weggelassen werden.

Objekterzeugung: Durch den Aufruf von `x = Klassenname()` wird automatisch zuerst der Konstruktor für die Klasse `Klassenname` und dann die Initialisierungsmethode (falls eine definiert wurde) aufgerufen. Danach ist das fertige Objekt über den Bezeichner `x` zugreifbar.

Destruktor: Eine Methode zur kontrollierten Löschung eines Objektes. Python hat hierfür eine automatische Müllabfuhr (*garbage collector*), der nicht mehr benötigte Objekte automatisch entsorgt.

Anfrage: Eine Methode, die vorgegebene Befehle abarbeitet und am Ende ein Objekt als Ergebnis der Arbeit zurückgibt (engl. **`return`**). Eine solche Methode wird im Klassendiagramm mit einem Fragezeichen vor dem Bezeichner gekennzeichnet.

Auftrag: Eine Methode, die auch vorgegebene Befehle abarbeitet, jedoch am Ende nichts zurückgibt. Sie wird im Klassendiagramm mit einem Ausrufezeichen vor dem Bezeichner gekennzeichnet.
Achtung: Die Programmiersprache Python sieht vor, dass jede Funktion/Methode ein Objekt an denjenigen zurückgibt, der sie aufgerufen hat. Möchte man kennzeichnen, dass eigentlich »nichts« zurückgegeben werden soll, so wird das Spezialobjekt `None` dazu genutzt. Bei dem Implementieren eigener Methoden darf **`return`** `None` aber der Einfachheit halber weggelassen werden.

self: Steht in der Klassenbeschreibung stellvertretend für das Objekt, von welchem eine bestimmte Methode im späteren Programmverlauf tatsächlich aufgerufen wird.

Konkatenation: Bezeichnet das Eineinanderhängen von Wörtern bzw. Buchstaben zu einer längeren Zeichenkette. In Python kann dies durch ein `+` befohlen werden.
Beispiel: Ist mit `y = "Welt"` bereits eine Zeichenkette mit dem Bezeichner `y` bekannt, so wird mit

```
x = "Hallo"+"␣"+y
```

ein neues Zeichenketten-Objekt erstellt (mit dem Bezeichner `x` versehen), welches der Zeichenkette `"Hallo␣Welt"` entspricht.