3. Konstruktor

Konstruktoren dienen der Initialisierung von Objekten bei deren Erzeugung. Der Code des Konstruktors wird genau einmal ausgeführt.

Merkmale

- Ein Konstruktor liefert KEINEN Rückgabewert.
- Ein Konstruktor kann die Attribute mit einem Defaultwert setzen oder mittels Parameter konkrete Werte zuweisen.
- Ein Konstruktor führt i.d.R. keine Programmlogik aus.

Default Konstruktor

Er wird OHNE Parameter aufgerufen und initialisiert alle Attribute mit einem vorgegebenen Wert (Default-Wert).

Beispiel 2.3: Default Konstruktor

```
class Shoe:
  def __init__(self):
    self._shoe_size = 40
    self._color     = 'green'
    self._shoe_type = 'sneaker'
```

Parametrierter Konstruktor

Ein Konstruktor kann Initialwerte entgegennehmen und ein Objekt so in einem definierten Zustand erzeugen. Es empfiehlt sich, die Attribute mit einem Initalwert zu versehen, damit keine undefinierten Werte resultieren können. Dies ist vor allem bei Wertzusicherungen wichtig.

Beispiel 2.4: Parametrierter Konstruktor

```
class Shoe:
  def __init__(self, shoe_size = 40, color = 'green', shoe_type =
'sneaker'):
    self._shoe_size = shoe_size
    self._color     = color
    self._shoe_type = shoe_type
```

Wird im Code nach Beispiel 2.4 ein Objekt erzeugt, dem keine Werte mitgeliefert werden (Default-Konstruktor), übernimmt Python die Initialwerte. Der entsprechende Aufruf sieht dann wie folgt aus:

```
  shoe = Shoe()
```

Werden alle Werte übergeben, empfiehlt es sich, die Reihenfolge der Parameter einzuhalten. Im gezeigten Beispiel 2.4 also den Code wie folgt zu realisieren:

```
  shoe = Shoe(44, 'black', 'boots')
```

Es ist aber auch möglich, nur einzelne der definierten Parameter anzugeben! Dazu muss beim Aufruf dann aber das Attribut genannt werden. Wird im Beispiel nur für das Attribut`color` ein Wert geliefert, ist der Code wie folgt zu realisieren:

```
 shoe = Shoe(color='blue')
```

Darstellung in UML

Viele Programmiersprachen kennen die Möglichkeit der Benennung von Parametern - wie oben gezeigt - nicht. Daher wird bei der Nutzung unterschiedlicher Konstruktoren das Konzept des Überladens (**overloading**) angewendet. Dabei darf ein Methodenname mehrfach deklariert werden, solange die Paramterliste eine klare Unterscheidung (bezüglich der Datentypen) zulässt. Als Beispiel wird hier der Code der Klasse Shoe in Java wiedergegeben.

Beispiel 2.5: Parametrierte Konstruktoren
Java-Code zu Beispiel 2.4

```
public class Shoe{
  // Deklaration der Attribute und Zuweisung der Initialwerte
  private int shoe_size    = 40;
  private String color     = "green";
  private String shoe_type = "sneaker";

  public Shoe(){
    // do nothing --> Default Konstruktor
    // die Attribute sind hier mit den oben zugewiesenen Werten
initialisert.
  }

  public Shoe(int size){
    // überladener Konstruktor mit einem Parameter
    this();  // Aufruf des eigenen Default-Konstruktors
    shoe_size = size;
  }
```

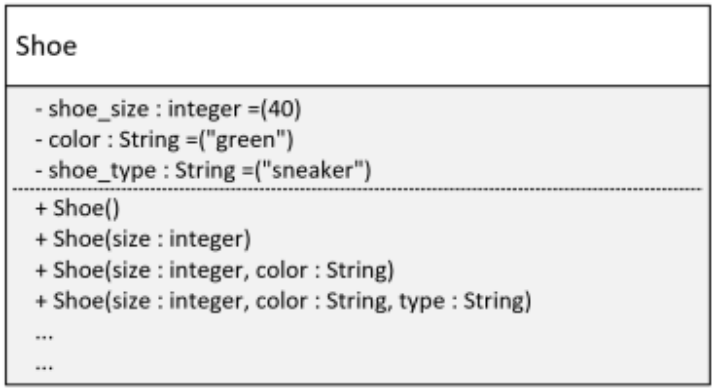
```
  public Shoe(int size, String color){
    // überladener Konstruktor mit zwei Parametern
    this(size); // Aufruf des Konstruktors mit dem Parameter size
    this.color = color; // um das Attribut vom Parameter zu unterscheiden,
wird das Schlüsselwort this benötigt.
  }

  public Shoe(int size, String color, String type){
    // überladener Konstruktor mit drei Parametern
    this(size, color); // Aufruf des Konstruktors mit den Parametern size
und color
    shoe_type = type;
  }
  :
  :
 }
```

Bei der Objekterzeugung wird je nach Parameterliste dann auf den entsprechende Konstruktor zugegriffen.

```
  ....
  Shoe shoe_1 = new Shoe();                          // Aufruf des Default-
Konstuktors
  Shoe shoe_2 = new Shoe(45);                        // Aufruf des
Konstruktors mit dem Parameter für size
  Shoe shoe_3 = new Shoe(39, 'yellow');              // Aufruf des
Konstruktors mit den Parametern für size und color
  Shoe shoe_4 = new Shoe(41, 'grey', 'slipper');     // Aufruf des
Konstruktors mit den Parametern für size, color und style
```

Diese Technik hat sich auch in die Darstellung der UML übertragen. So wird im Klassendiagramm für jede mögliche Parameterliste der Konstruktor vollständig angeschrieben.



Die Klasse weist 4 verschiedene Konstruktoren auf, die alle einzeln aufgelistet werden.

Abb 2.3: UML-Klassendiagramm mit überladenen Konstruktoren
