6.2.5 Setter und Getter Methoden

Mittwoch, 1. März 2023 10:58

6.2.5. Setter und Getter Methoden

Nun kann man sich fragen, wozu solche **private** Instanzvariablen gut sind, wenn sie von außen sowieso nur umständlich verändert werden können. Sehen wir uns dazu die folgende Klasse an:

```
class Motorrad():
    def __init__(self, marke, hubraum):
        self.marke = marke
        self.__hubraum = hubraum
```

Wir wollen nicht, dass ein Benutzer ein Motorrad mit negativem Hubraum modelliert. Aus diesem Grund haben wir hier die Instanzvariable `_hubraum` **private** gesetzt. Der Wert kann nun unter Beobachtung mit einer sogenannten setter() Methode geändert werden:

```
class Motorrad():
    def __init__(self, marke, hubraum):
        self.marke = marke
        self.__hubraum = hubraum

    def set_hubraum(self, kubik):
        if (kubik <= 0):
            print("Error: Negativer Wert für den Hubraum! \
Der Wert wurde nicht geändert")
        else:
            self.__hubraum = kubik
            print("Hubraum wurde geändert.")
```

Auf diese Weise haben wir in Zeile 6 die Kontrolle, dass `_hubraum` keine negativen Werte annehmen kann. Dies bedeutet, dass die Veränderung der Instanzvariable `_hubraum` in eine Klassenmethode ausgelagert wurde, in welcher die Manipulation am Objekt und somit der Instanzvariable kontrolliert und überwacht werden kann.

Um nun auch noch den Wert abfragen zu können, erstellen wir zusätzlich noch eine `getter()` Methode. Dies kann dann so aussehen:

```
class Motorrad():
    def __init__(self, marke, hubraum):
        self.marke = marke
        self.__hubraum = hubraum

    def set_hubraum(self, kubik):
        if (kubik <= 0):
            print("Error: Negativer Wert für den Hubraum! \
Der Wert wurde nicht geändert")
        else:
            self.__hubraum = kubik
            print("Hubraum wurde geändert.")

    def get_hubraum(self):
        return self.__hubraum
```