Zusätzliche Unterlagen zu Übung 2: Objektorientierte Programmierung in Python

---

Klassen und Vererbung

Im nachfolgenden Beispielcode seht ihr, wie zwei Klassen definiert werden und eine Vererbung zwischen ihnen implementiert ist.

Klassenbeschreibung

In der `City`-Klasse nimmt der Konstruktor `__init__` zwei Parameter, `name` und `kfz`, auf und initialisiert die Attribute der Klasse. Die `__str__` Methode überschreibt die Standarddarstellung eines Objekts, sodass bei Aufrufen der `print`-Funktion ein lesbarer String zurückgegeben wird. Das Format ist `"Name, "KFZ"`.
Die `Stadtteil`-Klasse erbt von der Klasse `City`. Das ist zum einen erkenntlich dadurch, dass während der Klassendefinition `City` in Klammern angegeben wird: `class Stadtteil(City)`. Darüber hinaus fügt der Konstruktor der Klasse `Stadtteil` (`__init__`) ein zusätzliches Attribut, `plz`, hinzu und ruft den Konstruktor der `City`-Klasse mit `super().__init__(name, kfz)` auf. Das `super()` signalisiert, dass der Konstruktor der Oberklasse (oder *superclass*) `City` aufgerufen wird.

Objektinstanzen

- `Hamburg` ist eine Instanz von `City` mit `name = "Hamburg"` und `kfz = "HH"`
- `Altona` ist eine Instanz von `Stadtteil` mit `name = "Altona"`, `plz = 20000` und `kfz = "HH"`

Wenn der Befehl `print(Altona)` aufgerufen wird, wird durch die Vererbung die `__str__`-Methode aufgerufen, die in der Klasse `City` definiert ist. Da `Stadtteil` von `City` erbt, hat `Stadtteil` Zugriff auf die `__str__`-Methode. Das bedeutet, dass `Altona, HH` ausgegeben wird.

```python
class City:
    def __init__(self, name, kfz):
        self.name = name
        self.kfz = kfz

    def __str__(self):
        return f"{self.name}, {self.kfz}"

class Stadtteil(City):
    def __init__(self, name, plz, kfz):
        super().__init__(name, kfz)
        self.plz = plz

Hamburg = City("Hamburg", "HH")
Altona = Stadtteil("Altona", 20000, "HH")

print(Altona)
```