# Vererbung

## Vererbung – Implementierung

In der Klassendefinition einer Unterklasse muss darauf hingewiesen werden, dass die Attribute und Methoden der Oberklasse (Superklasse) zur Verfügung stehen sollen.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | class Konto:  def \_\_init\_\_(self, p\_nr, p\_name):  self.kontonr = int(p\_kontonummer)  self.inhaber = p\_name  self.kontostand = 0.0  class Sparkonto(Konto):  […] |

Das *Sparkonto* erbt im oberen Beispiel von der Oberklasse *Konto*. Dies wird implementiert, indem der Klassenname der Oberklasse in Klammern nach dem Klassennamen der Unterklasse geschrieben wird. Danach erhält die Unterklasse alle Attribute und Methoden der Oberklasse und kann diese innerhalb der Klasse auch aufrufen, selbst wenn die Attribute privat wären.

**3.1.2.1 Vererbung und Konstruktoren**

Innerhalb eines Konstruktors in einer Unterklasse kann der Konstruktor der Oberklasse verwendet bzw. aufgerufen werden.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | class Konto:  def \_\_init\_\_(self, p\_nr, p\_name):  self.kontonr = int(p\_nr)  self.inhaber = p\_name  self.kontostand = 0.0  […]  class Sparkonto(Konto):  def \_\_init\_\_(self, p\_nr, p\_name, p\_zinssatz):  super().\_\_init\_\_(p\_nr, p\_name)  self.zinssatz = p\_zinssatz  # Hauptprogramm  s = Sparkonto(42, "Simon Amsel", 0.5)  print("Kontostand:", s.kontostand) |

Der Konstruktor der Unterklasse *Sparkonto* verwendet in Zeile 11 den Konstruktor der Oberklasse *Konto*. Der Konstruktor wird mit dem Schlüsselwort *super()* angesteuert, darauf folgt ein Punkt. Der Konstruktor der Oberklasse wird mit der *\_\_init\_\_-*Methode inkl. den implementierten Parametern aufgerufen. Verfügt die Unterklasse über weitere Übergabeparameter, können diese wie gewohnt verarbeitet werden.

**3.1.2.2 Überschreiben von Methoden**

Eine Methode einer Oberklasse kann in der Unterklasse überschrieben werden. Dies ist immer dann sinnvoll, wenn in einer Unterklasse eine Methode anders ausgeführt werden soll als in der Oberklasse. Möchte man in der Unterklasse eine Methode aus der Oberklasse überschreiben, müssen sowohl der Methodenname als auch die Übergabeparameter gleich zur Methode der Oberklasse sein.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | class Konto:  def \_\_init\_\_(self, p\_nr, p\_name):  self.kontonr = int(p\_nr)  self.inhaber = p\_name  self.kontostand = 0.0  def auszahlen(self, p\_betrag):  self.kontostand = self.kontostand - p\_betrag  class Sparkonto(Konto):  def \_\_init\_\_(self, p\_nr, p\_name, p\_zinssatz):  super().\_\_init\_\_(p\_nr, p\_name)  self.zinssatz = p\_zinssatz  def auszahlen(self, p\_betrag):  if p\_betrag > self.kontostand:  print("Auszahlung nicht möglich")  else:  self.kontostand = self.kontostand - p\_betrag  def kontostand\_ausgeben(self):  print("Kontostand:", self.kontostand, "EUR")  # Hauptprogramm  s = Sparkonto(42, "Simon Amsel", 0.5)  s.auszahlen(500) |

Ausgabe: Auszahlung nicht möglich

Durch den Methodenaufruf im Hauptprogramm (Zeile 26) wird in der Klasse *Sparkonto* die Methode *auszahlen(p\_betrag)* „gesucht“. Wird diese in der Unterklasse nicht gefunden, würde die Methode *auszahlen(p\_betrag)* in der Oberklasse ausgeführt werden. Da in der Unterklasse *Sparkonto* eine Methode *auszahlen(p\_betrag)* vorhanden ist, wird diese aber ausgeführt und eine Überziehung des Kontostands ist nicht mehr möglich.

**3.1.2.3 „Auslagern“ von Klassen**

Umfasst ein Programmierprojekt etliche Klassen, kann der Code schnell unübersichtlich werden (siehe Überschreiben von Methoden). Es bietet sich dann an, die einzelnen Klassen in unterschiedlichen Dateien abzuspeichern. Um die Funktionsfähigkeit des Programmes zu gewährleisten, müssen verwendete Klassen (bspw. Oberklassen) dann importiert werden.

|  |  |  |
| --- | --- | --- |
| konto.py | Zeile | sparkonto.py |
| class Konto:  def \_\_init\_\_(self, p\_nr, p\_name):  self.kontonummer = p\_nr  self.kontoinhaber = p\_name  self.kontostand = 0.0  def einzahlen(self, p\_betrag):  self.kontostand =  self.kontostand + p\_betrag | 1  2  3  4  5  6  7  8  9 | from konto import Konto  class Sparkonto(Konto):  def \_\_init\_\_(self, p\_nr, p\_name, p\_zinssatz):  super().\_\_init\_\_(p\_nr, p\_name)  self.zinssatz = p\_zinssatz    def kontostand\_ausgeben(self):  print(self.kontostand, "EUR") |

In Zeile 1 der Datei *sparkonto.py* muss aus der Datei *konto.py* die Klasse *Konto* importiert werden, damit die Vererbung umgesetzt werden kann. Der Code gliedert sich dabei in vier Teile:

|  |  |  |  |
| --- | --- | --- | --- |
| from | konto | import | Konto |
| Schlüsselwort | Dateiname ohne .py | Schlüsselwort | Klassenname |

Hinweis: Bei der Implementierung ist darauf zu achten,

* dass die **Dateien** in **einem Ordner** liegen und
* sich die **Dateinamen** von den **Klassennamen** **unterschieden** (ggf. durch Klein- und Großschreibung)!

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | # Hauptprogramm  from sparkonto import Sparkonto  s = Sparkonto(42, "Simon Amsel", 0.5)  s.einzahlen(500)  s.kontostand\_ausgeben() |

Im Hauptprogramm wird die Klasse *Sparkonto* aus der Datei sparkonto.py verwendet und muss deswegen importiert werden (vgl. Zeile 1). Die Klasse *Konto* wird nicht verwendet und muss deswegen auch **nicht** importiert werden, obwohl die Klasse *Konto* die Oberklasse der Klasse *Sparkonto* ist.