# Abstrakte Klassen

## Abstrakte Klassen implementieren

Häufig ergibt es sich aufgrund einer Vererbungshierarchie, dass eine Oberklasse zwar sinnvoll ist, aber dort selbst nie Objekte erzeugt werden sollen, sondern nur in den Unterklassen. Diese Klasse, von der keine Objekte instanziiert werden sollen, stellt eine so genannte abstrakte Klasse dar. Mit abstrakten Klassen werden Vorgaben (Attribute und Methoden) festgelegt, die alle Unterklassen erben.

Ein Beispiel für eine abstrakte Klasse stellt die Oberklasse *Konto* dar. Ein Bankkunde kann ein *Girokonto* und ein *Sparkonto* eröffnen. Aus der Sachlogik ergibt sich, dass aber niemals ein Objekt der Oberklasse *Konto* erzeugt werden soll, weil dieses nur gemeinsame Attribute und Methoden hat.

|  |  |  |  |
| --- | --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | from abc import ABC, abstractmethod  class Konto(ABC):  def \_\_init\_\_(self, p\_nr, p\_name):  self.kontonummer = int(p\_nr)  self.kontoinhaber = p\_name  self.kontostand = 0.0  @abstractmethod  def auszahlen(self, p\_betrag):  pass  […] |  | *Konto* |
| + kontonummer: int  + kontoinhaber: str  + kontostand: float |
| + Konto(*p\_nr: int, p\_name: str*)  *+ auszahlen(p\_betrag: float): void* |
| Abstrakte Klassen[[1]](#footnote-1) und Methoden[[2]](#footnote-2) werden im UML-Klassendiagramm kursiv gedruckt. |

Eine abstrakte Klasse muss in Python von der Klasse ABC (=“abstract base classes“) erben, die importiert werden muss (Zeile 1). Eine Klasse ist immer dann abstrakt, wenn sie abstrakte Methoden enthält. Eine abstrakte Methode wird mit dem Schlüsselwort @abstractmethod eingeleitet. Danach wird die abstrakte Methode ohne Körper implementiert. Sie erhält statt des Körpers das Schlüsselwort pass als Platzhalter, das so viel bedeutet wie „nichts tun“.

Beim Versuch im Hauptprogramm ein Objekt der Klasse *Konto* zu erzeugen erscheint die Fehlermeldung, dass kein Objekt der abstrakten Klasse *Konto* mit der abstrakten Methode *auszahlen*(…) erzeugt werden kann.

|  |  |  |
| --- | --- | --- |
| 1  2  3 | # Hauptprogramm  from konto import Konto  k = Konto(12, "Simon Amsel") | |
|  |  | |
| Fehlermeldung: | | Traceback (most recent call last):  File "C:\Users\#######\Hauptprogramm.py", line 3, in <module>  k = Konto(12, "Simon Amsel")  TypeError: Can't instantiate abstract class Konto with abstract methods auszahlen |

## Abstrakte Klassen und Vererbung

Erbt eine Klasse von einer abstrakten Klasse müssen alle abstrakten Methoden „ausprogrammiert“ werden.

|  |  |
| --- | --- |
| konto.py | |
| 1  2  3  4  5  6  7  8  9  10 | from abc import ABC, abstractmethod  class Konto(ABC):  def \_\_init\_\_(self, p\_nr, p\_name):  […]    @abstractmethod  def auszahlen(self, p\_betrag):  pass  […] |

|  |  |
| --- | --- |
| sparkonto.py | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | 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 auszahlen(self, p\_betrag):  if p\_betrag > self.kontostand:  print("Keine Auszahlung möglich")  else:  self.kontostand = self.kontostand - p\_betrag  […] |

Die Klasse *Sparkonto* stellt eine Unterklasse der abstrakten Oberklasse *Konto* dar, weswegen die abstrakte Methode *auszahlen*(…) programmiert bzw. überschrieben werden muss. Würde die Methode *auszahlen(…)* nicht implementiert werden, könnte auch von der Klasse *Sparkonto* kein Objekt erzeugt werden.

Mit einer abstrakten Methode kann folglich erzwungen werden, dass eine spezielle Codierung für die Unterklasse erfolgen muss. Der Programmierer muss sich also für jede Kontoart (*Girokonto*, *Sparkonto* usw.) überlegen, wie die Methode *auszahlen*(…) codiert werden muss.

1. Eine abstrakte Klasse kann alternativ auch mit dem Attribut {abstract} dargestellt werden (im Violet UML Editor nötig). Bspw. Konto {abstract} [↑](#footnote-ref-1)
2. Eine abstrakte Methode kann alternativ auch mit dem Attribut {abstract} dargestellt werden (im Violet UML Editor nötig). Bspw. + auszahlen(p\_betrag: float):void {abstract} [↑](#footnote-ref-2)