2.7 Vererbung

LK

LK

Vererbung und Polymorphie stehen im Themenfeld *Objektorientierung* nur im Lehrplan des Leistungskurses. Der Grundkurs endet bei Klassen, Objekten, Attributen, Konstruktoren und Methoden samt UML-Darstellung einzelner Klassen.

Zwei Klassen ähneln einander oft stark. Ein Girokonto und ein Sparkonto haben beide Inhaber und Stand; nur das Abheben läuft verschieden. Alles doppelt zu schreiben wäre Arbeit und würde beim nächsten Fehler zweimal korrigiert werden müssen.

Definition: Vererbung

Eine Klasse kann von einer anderen **erben**. Die **Unterklasse** übernimmt alle Attribute und Methoden der **Oberklasse** und darf eigene hinzufügen oder geerbte überschreiben.

Die Probe aufs Exempel ist immer derselbe Satz: „*Ein Sparkonto **ist ein** Konto.*" Stimmt der Satz, passt die Vererbung. Stimmt er nicht, nehmen Sie ein Attribut. (Ein Konto *ist* kein Inhaber — es *hat* einen.)

Python

```python
class Sparkonto(Konto):
    def __init__(self, inhaber, zinssatz):
        super().__init__(inhaber)               # Konstruktor der Oberklasse
        self.__zinssatz = zinssatz

    def verzinsen(self):                        # neu
        self.einzahlen(self.stand() * self.__zinssatz / 100)

    def abheben(self, betrag):                  # überschrieben
        if betrag > 500:
            return False                        # Sparkonto: höchstens 500 €
        return super().abheben(betrag)
```

Object Pascal

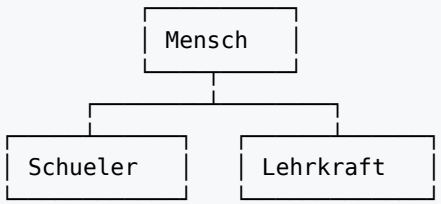
```
type
  TSparkonto = class(TKonto)
  private
    FZinssatz: Real;
  public
    constructor Create(inhaber: String; zinssatz: Real);
    procedure Verzinsen;
    function Abheben(betrag: Real): Boolean; override;
  end;

constructor TSparkonto.Create(inhaber: String; zinssatz: Real);
begin
  inherited Create(inhaber);
  FZinssatz := zinssatz;
end;

function TSparkonto.Abheben(betrag: Real): Boolean;
begin
  if betrag > 500 then
    Result := False
  else
    Result := inherited Abheben(betrag);
end;
```

`inherited` beziehungsweise `super()` ruft die Fassung der Oberklasse auf. So muss die Prüfung auf ausreichende Deckung kein zweites Mal geschrieben werden — das Sparkonto fügt nur seine eigene Zusatzregel hinzu und reicht dann weiter.

Beispiel: Eine Vererbungshierarchie



`Mensch` trägt Vorname, Nachname, Geburtsdatum. `Schueler` ergänzt Klassenstufe, `Lehrkraft` ergänzt Fächer. Was beide gemeinsam haben, steht genau einmal da.

2.7.1 Polymorphie

Weil ein Sparkonto ein Konto *ist*, darf es überall stehen, wo ein Konto erwartet wird. Die Bank aus Abschnitt 2.6 funktioniert unverändert auch mit Sparkonten in ihrer Liste.

```
bank = Bank("Sparkasse")
giro = bank.eroeffnen("A. Lovelace")        # ein gewöhnliches Konto
spar = Sparkonto("C. Babbage", 1.5)
bank.aufnehmen(spar)                        # ein Sparkonto in derselben Liste

giro.einzahlen(1000)
spar.einzahlen(1000)
print(giro.abheben(600))                    # True  – Konto.abheben
print(spar.abheben(600))                    # False – Sparkonto.abheben
print(bank.gesamtvermoegen())               # 1400.0
```

Zweimal dieselbe Nachricht `abheben(600)`, zweimal ein anderes Ergebnis — und die Bank hat dafür keine einzige Zeile geändert. Welche der beiden Fassungen läuft, entscheidet sich erst zur Laufzeit, und zwar am Empfänger der Nachricht. Das nennt man **Polymorphie** („Vielgestaltigkeit").

Stolperstein

In Python geschieht das von allein. In Object Pascal **nicht**: Eine Methode wird nur dann zur Laufzeit ausgewählt, wenn sie in der Oberklasse als `virtual` und in der Unterklasse als `override` gekennzeichnet ist. Deshalb steht oben in `TKonto` `function Abheben(...): Boolean; virtual;` und in `TSparkonto ... override;`. Fehlt das `virtual`, lehnt der Übersetzer das `override` ab; schreibt man stattdessen bloß eine gleichnamige Methode ohne beides, übersetzt alles anstandslos — und es läuft trotzdem immer die Fassung der Oberklasse, sobald das Objekt in einer Variablen vom Typ `TKonto` steckt. Das ist einer der am schwersten zu findenden Fehler in objektorientiertem Pascal.

Merke

Polymorphie ist der eigentliche Gewinn der Vererbung. Der aufrufende Quelltext muss nicht wissen, welche Art von Konto vor ihm steht. Er schickt die Nachricht, das Objekt weiß selbst, was zu tun ist.

Arbeitsauftrag 5

10. Ergänzen Sie `Girokonto` mit einem Dispo-Rahmen von 1000 €, bis zu dem der Stand negativ werden darf.
11. Fügen Sie der Bank ein Girokonto und ein Sparkonto hinzu und rufen Sie auf beiden `abheben(600)` auf. Erklären Sie die unterschiedlichen Ergebnisse.
12. Prüfen Sie mit dem *ist-ein*-Satz, ob Vererbung passt:

    a) Quadrat — Rechteck, b) Auto — Motor, c) Lehrkraft — Mensch, d) Liste — Baum.