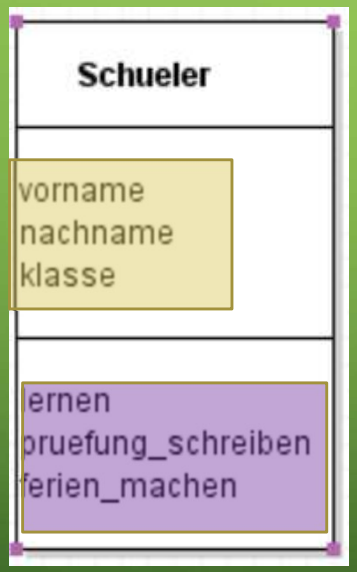

OBJEKTORIENTIERT
PROGRAMMIEREN
EINFÜHRUNG
GRUNDBEGRIFFE

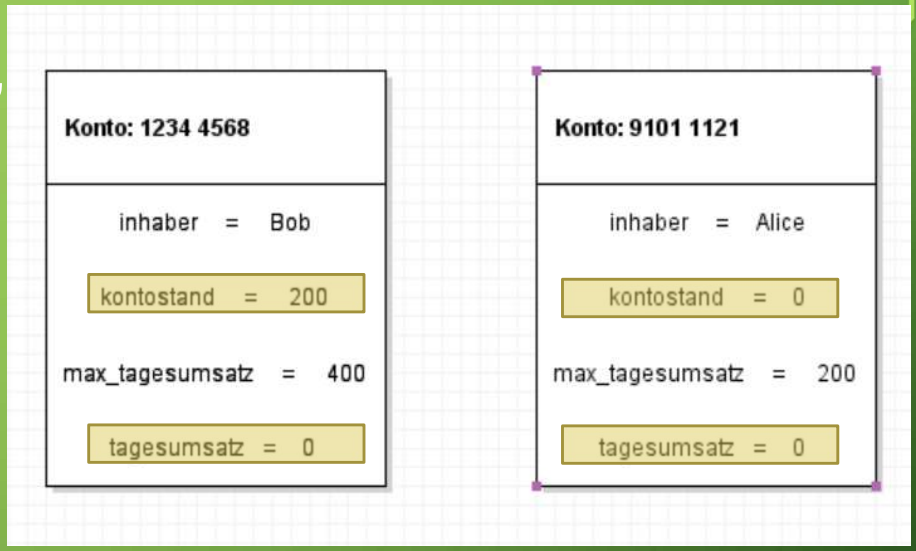
HINTERGEDANKE - ABSTRAKTION

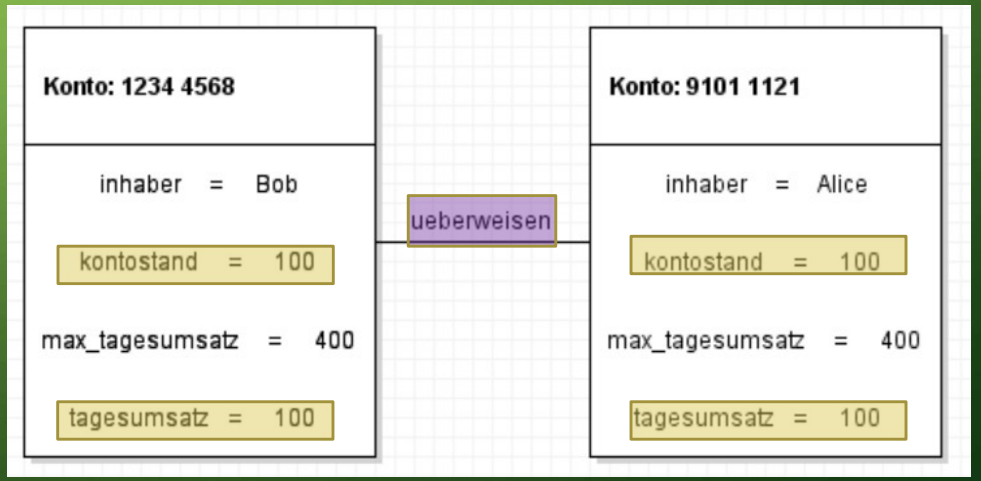
- (Vereinfachte) Abbildung realer Objekte (Begriffe) als Software-Konstrukt:
  - Eigenschaften
  - Fähigkeiten



HINTERGEDANKE - KAPSELUNG

- Verwendung auch ohne detaillierte Kenntnisse der dahinterstehenden Implementierung möglich:
  - Für die Anwendung reichen folgende Kenntnisse aus:
    - Eingabedaten
    - Ergebnis

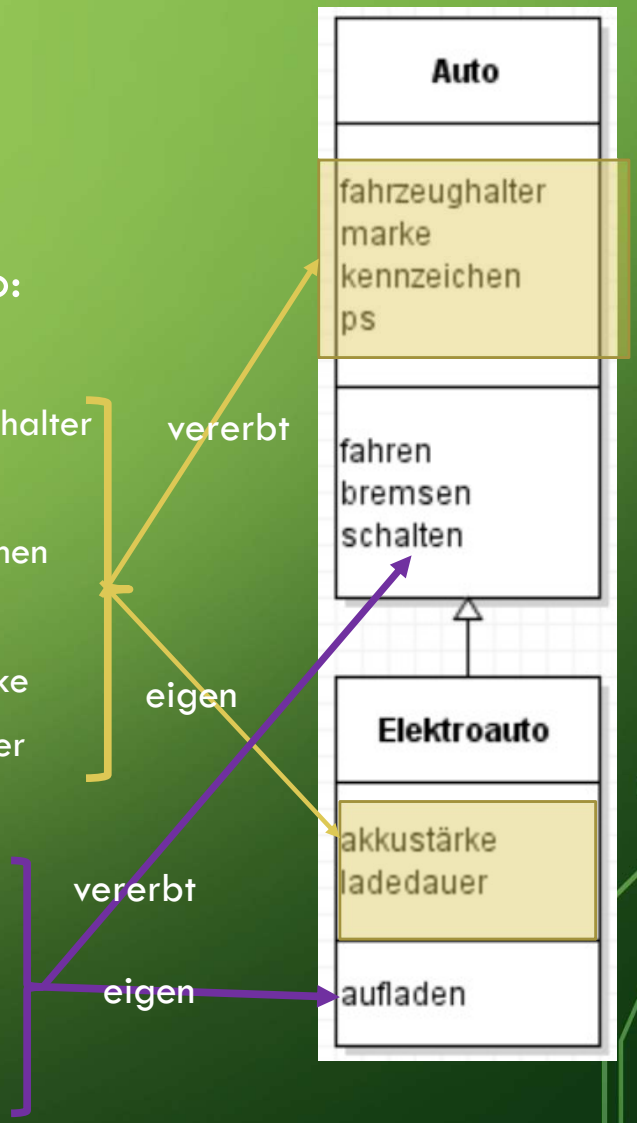




HINTERGEDANKE VERERBUNG

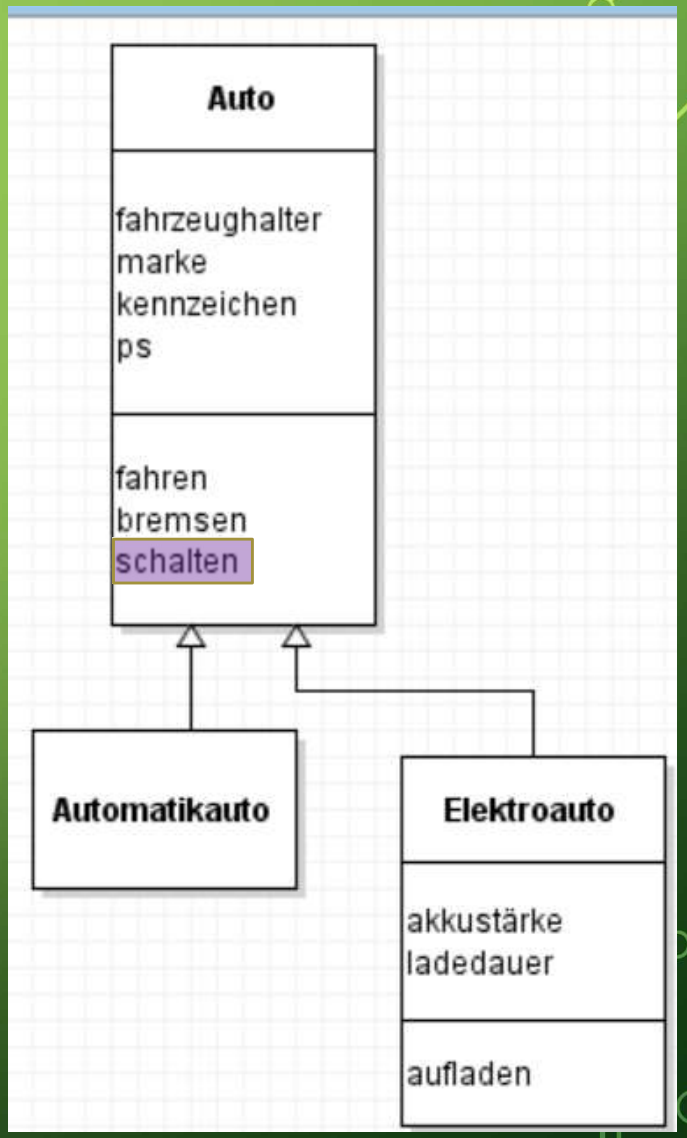
- Spezialformen können:
  - von allgemeinen Objekten „erben“:
    - Eigenschaften
    - Methoden
    - ➔ d.h. sie müssen nicht neu implementiert werden
  - Spezielle Erweiterung beinhalten:
    - Eigenschaften
    - Fähigkeiten

- D.h. Elektroauto:
  - Attribute:
    - fahrzeughalter
    - marke
    - kennzeichen
    - ps
    - akkustärke
    - ladedauer
  - Methoden:
    - fahren
    - bremsen
    - schalten
    - aufladen

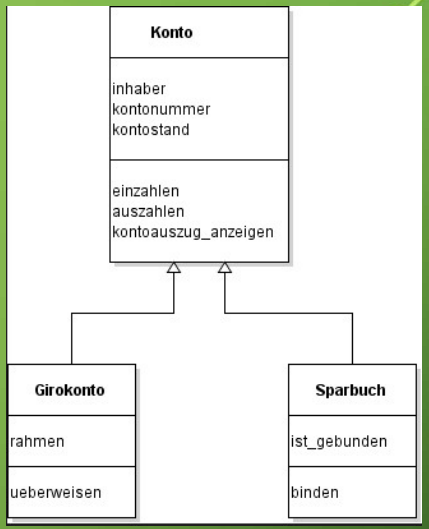


HINTERGEDANKE POLYMORPHISMUS

- Vielgestaltigkeit:
  - Dieselbe Fähigkeit kann abhängig vom Typ des Objekt:
    - andere Eingabedaten benötigen
    - anders ablaufen
  - z.B. Auto.schalten(gang):
    - kuppeln
    - Schalter umlegen
  - z.B. Automatikauto.schalten():
    -



HINTERGEDANKE WIEDERVERWERTBARKEIT



- Wiederverwertbarkeit:
  - Klassen können:
    - in Modulen/Libraries gekapselt
    - in anderen Programmen importiert werden
    - um dort neue Instanzen/Objekte zu erzeugen und zu verwenden

```
import Konto as kontoservice

#Primitives Anwendungsbeispiel der Basisklasse Konto
#Instanz/Objekt erzeugen
konto_jane = kontoservice.Konto("Jane", 1111, 1000)
konto_john = kontoservice.Konto("John", 2222, 0)

konto_test = kontoservice.GiroKonto("Test", 333, 100, -3000)

#Kontostand der beiden Konten ausgeben
print("TEST KONTO.KONTOAUSZUG_ANZEIGEN:")
#Methodenaufruf der Methode kontoauszug_anzeigen()
#in der Abgeleiteten Klasse
konto_jane.kontoauszug_anzeigen()
konto_john.kontoauszug_anzeigen()
```

GRUNDBEGRIFFE

ALLGEMEINER BEGRIFF

- Vorlage → Objekt
- Eigenschaften
- Fähigkeiten

Konto.py *

```python
inhaber = "Alice"
kontonummer = 12345678
tagsumsatz = 100
max_tagesumsatz = 200

def neues_konto():
    pass

def ueberweisen(betrag, nummer):
    pass

def einzahlen(betrag):
    pass

def auszahlen(betrag):
    pass

def zeigen():
    pass
```

TECHNISCHE UMSETZUNG BISHER

- Anforderung → Programm
- Variablen
- Funktionen

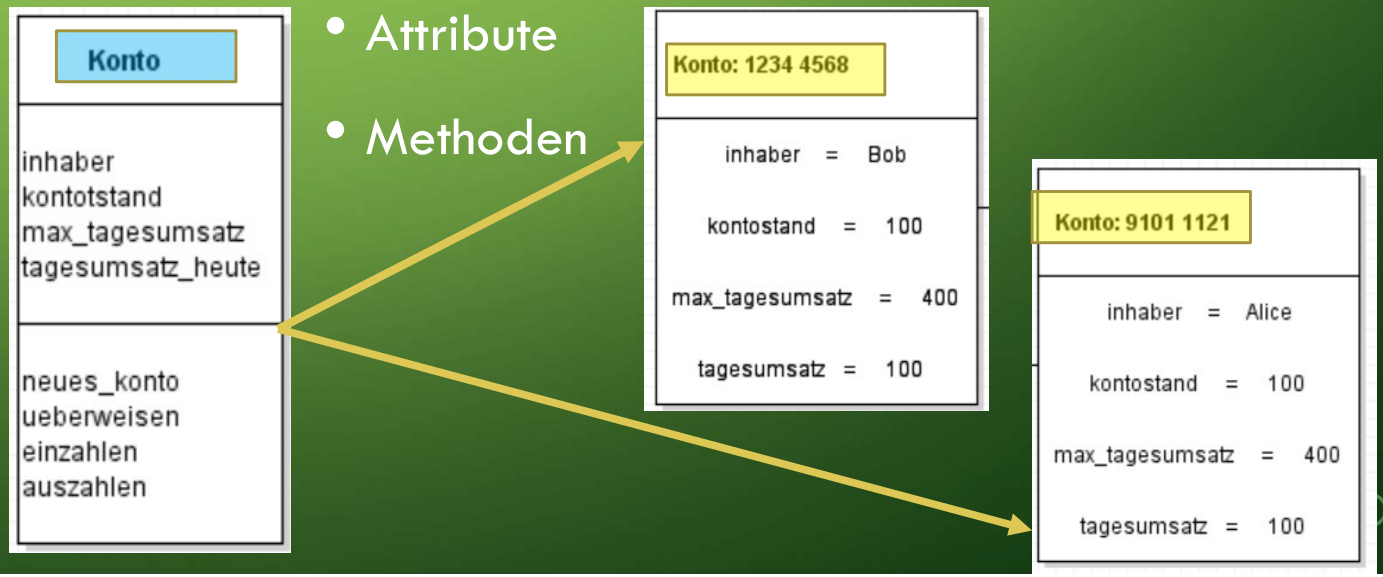
GRUNDBEGRIFFE

ALLGEMEINER BEGRIFF

- Vorlage/Schema → Objekt
- Eigenschaften
- Fähigkeiten

TECHNISCHE UMSETZUNG OOP

- Klasse (Abstraktion) → Objekt/ Instanz
- Attribute
- Methoden



```python
#Klasse(ndefinition)
class Konto:
    #Konstruktor: erzeugt ein neues Objekt und ordnet den Attributen konkrete Werte zu
    def __init__(self, the_inhaber, the_kontonummer, the_kontostand):
        #Attribute der Klasse Konto
        self.inhaber = the_inhaber
        self.nummer = the_kontonummer
        self.kontostand = the_kontostand

    #Methode
    def einzahlen(self, betrag):
        """Modellierung einer Einzahlung auf das bestehende Konto
        """

        #nur positive Beträge
        if (betrag < 0):
            print("Ungültige Eingabe: Bitte Betrag > 0 eingeben.")
            return False
        #alles ok: Einzahlung wird vorgenommen
        else:
            self.kontostand = self.kontostand + betrag
            return True
```

GRUNDBEGRIFFE IM CODE

- Klasse
- Konstruktor
- Methode
- Attribute

```
#Aufruf der Methode einzahlen
#aus der Basisklasse, da sie in der abgeleiteten Klasse
#nicht überschrieben wird.
konto_jane.einzahlen(100)
konto_jane.kontoauszug_anzeigen()
#Attributzugriff über Instanz/Objekt
#ändert den Wert des Attributes kontostand
#am Objekt/Instanze konto_john
konto_john.kontostand = konto_john.kontostand + 1
```

```
import Konto as kontoservice

#Primitives Anwendungsbeispiel der Basisklasse Konto
#Instanz/Objekt erzeugen
konto_jane = kontoservice.Konto("Jane", 1111, 1000)
konto_john = kontoservice.Konto("John", 2222, 0)

#Kontostand der beiden Konten ausgeben
print("TEST KONTO.KONTOAUSZUG_ANZEIGEN:")
#Methodenaufruf der Methode kontoauszug_anzeigen()
#in der Abgeleiteten Klasse
konto_jane.kontoauszug_anzeigen()
konto_john.kontoauszug_anzeigen()
```

GRUNDBEGRIFFE IM CODE

- Methodenaufruf
  - aus der Basisklasse
  - aus der abgeleiteten Klasse
- Instanzen/Objekt erzeugen
- Attribut-Zugriff bzw. ändern