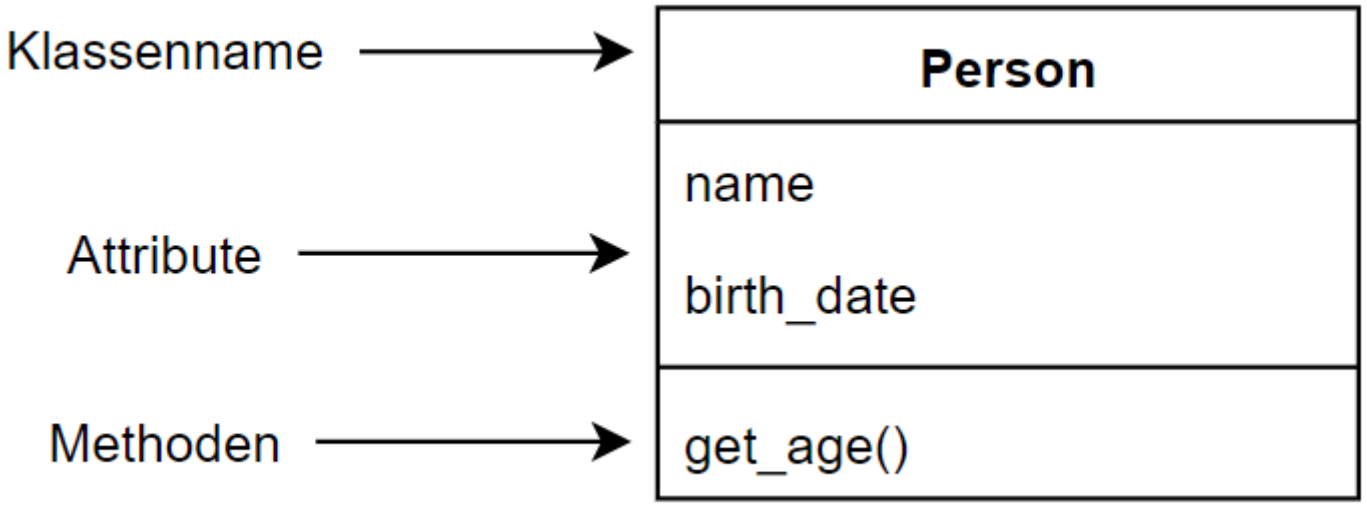
UML-Klassendiagramme

Unified Modeling Language (UML)

- Grafische Modellierungs*sprache*
- Erlaubt die Konstruktion und Visualisierung von (Software-)Elementen
- Hat eine eigene Syntax
- 14 verschiedene Diagrammtypen:
  - 7 für Modellierung von Strukturen (Strukturdiagramme)
  - 7 für Modellierung von Prozessen (Verhaltensdiagramme)
- Häufigster Anwendungsfall für die Software-Entwicklung:
  - Modellierung eines Programms (Anforderungsanalyse, Grob- und Feinkonzeption)
- Empfohlenes Tool zum Modellieren: https://app.diagrams.net/
  - In der IHK-Prüfung handschriftlich, daher empfohlen mind. 1 mal auf Papier zu üben

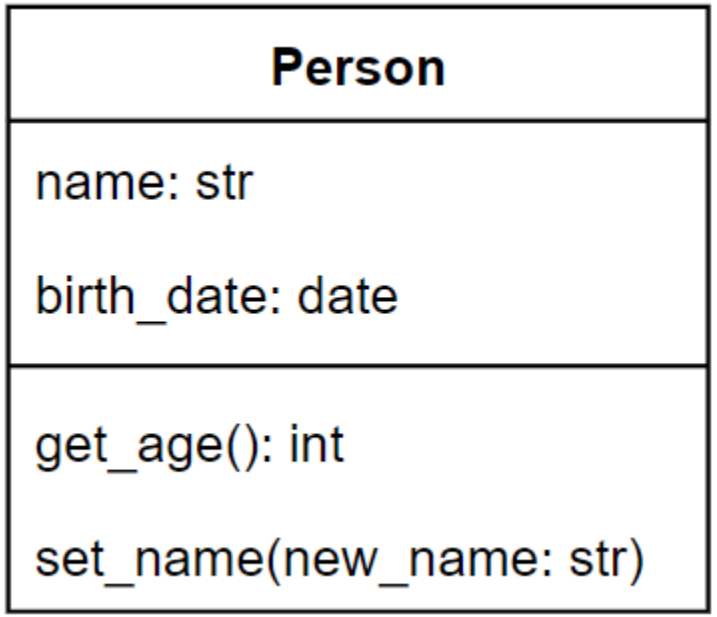
UML Klassendiagramm: Klasse

- Für die Modellierung von Klassen
- Darstellung: Jede Klasse ist ein Rechteck mit 3 Teilen
  - Darüber hinaus: Beziehungen, Kardinalitäten etc.



UML Klassendiagramm: Datentypen

- Datentypen angeben: optional aber nützlich
  - Die IHK sieht es gerne
  - Fast wie in Python außer beim Rückgabetyp
- Es muss nicht jede Methode rein (z.b. Konstruktor)



Lässt sich einfach in Python überführen:

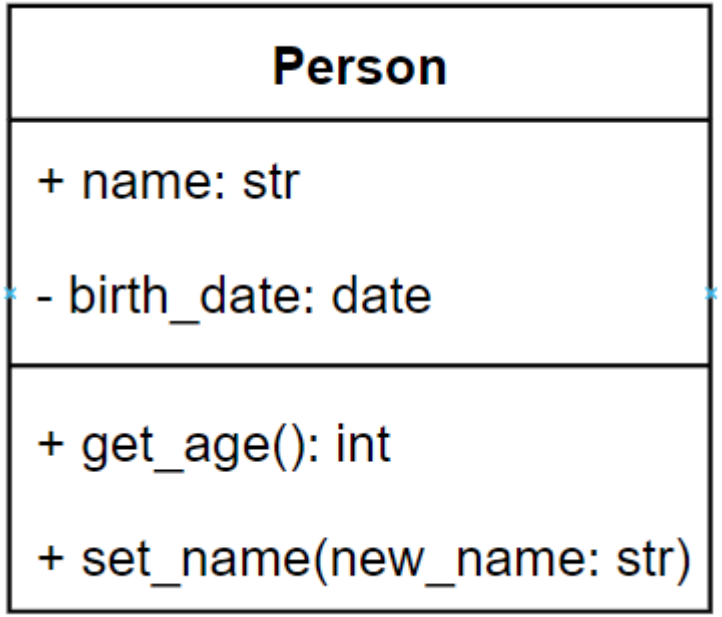
```python
class Person:
    def __init__(self, name: str, birth_date: date):
        self.name = name
        self.birth_date = birth_date

    def get_age(self) -> int:
        return date.today().year - self.birth_date.year

    def set_name(new_name: str):
        self.name = new_name
```

UML Klassendiagramm: Sichtbarkeiten

- Zugriffsmodifikatoren zum Einschränken der Sichtbarkeit:
  - Public (+) -> Zugriff von überall aus möglich (z.b andere Klassen)
  - Protected (#) -> Zugriff nur innerhalb der Klasse und Unterklassen
  - Private (-) -> Zugriff nur innerhalb der Klasse
  - Gibt noch Package (~) und Derived (/) -> beide nicht relevant
- Wenn nicht näher spezifiziert:
  - Attribut mind. protected setzen
  - Methode .get_ … implementieren
  - Implementierte Methode public setzen



Sichtbarkeiten in Python

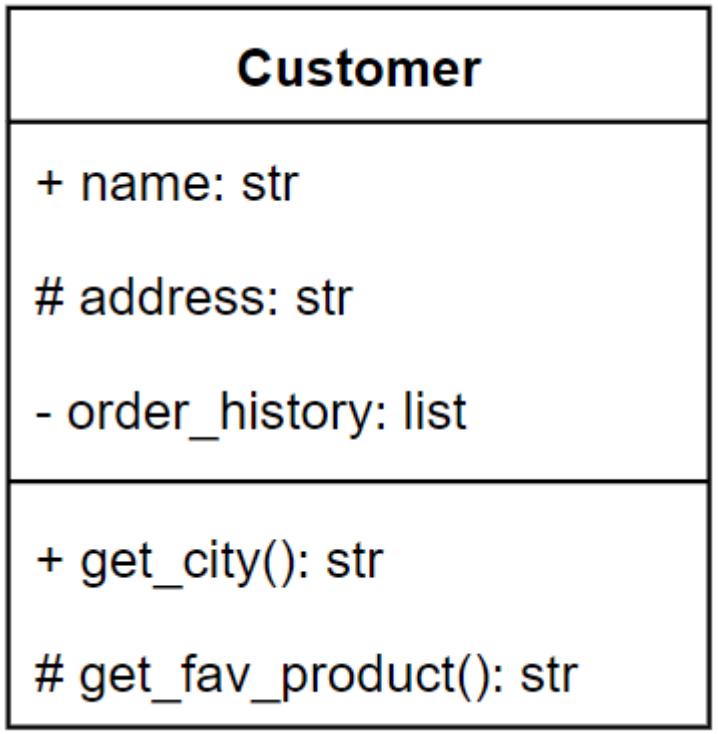
- Python machts ganz anders!
  - Einschränkungen der Sichtbarkeiten sind eher Einschränkungen des Komforts
- Attribute/Methoden sind per default public
  - Lassen sich somit von Außerhalb der Klasse lesen & verändern
  - Man kann in Python die Sichtbarkeit nicht verändern!

```python
class BankAccount:
    def __init__(self, balance: int):
        self.balance = balance

my_bank_acc = BankAccount(1000)
my_bank_acc.balance = 0
print(my_bank_acc.balance) # 0
```

- Warum macht Python sowas?
  - Die Frage sollte eher sein: Warum nicht? Welchen Schutz bietet die Visibility **wirklich**?

- Es gibt trotzdem Hinweise auf die Sichtbarkeit bzw. Einschränkungen beim Zugriff:




```python
class Customer:
    def __init__(self, name: str, address: str):
        self.name = name
        self._address = address
        self.__order_history = []

    def get_city(self) -> str:
        pass

    def _get_fav_product(self) -> str:
        pass

c = Customer("Riyad", "Turmstraße 21, 10559 Berlin")
print(c.name) # läuft
print(c.address) # AttributeError
print(c._address) # läuft
print(c.order_history) # AttributeError
print(c.__order_history) # AttributeError :)
# aber:
print(c._Customer__order_history) # läuft :(
# noch "schlimmer":
print(c.__dict__)
# {'name': 'Riyad',
#  '_address': 'Turmstraße 21, 10559 Berlin',
#  '_Customer__order_history': []}
```

Python macht *name mangling*:
Aus `Customer.__order_history`
wird
`Customer._Customer__order_history`
-> Idee: Versehentlichen Zugriff verhindern

Übung

1. Erstelle aus folgenden Anforderungen ein UML-Klassendiagramm:

a. *Es soll eine Bibliothek modelliert werden. Innerhalb dieser gibt es verschiedene Bücher, welche einen Titel, mind. einen Autor und die Anzahl der Ausleihen insgesamt haben. Diese Anzahl soll jedoch nicht von außen einsehbar sein. Dafür soll eine Methode existieren, welche prüft, ob ein neues Exemplar des Buchs gekauft werden soll. Dies ist dann der Fall, wenn ein Buch insgesamt 40 Mal ausgeliehen worden ist.*

b. *Die Bibliothek hat Mitarbeiter. Diesen werden durch einen Vornamen, eine Berufsbezeichnung und dem Arbeitsvertrag (als string) modelliert. Den Arbeitsvertrag soll man von außen nicht einsehen können außer ein Mitarbeiter mit Berufsbezeichnung = "Chef" schaut drauf.*

2. Überführe das Diagramm in Python

Hausi

1. Erstelle aus folgenden Anforderungen ein UML-Klassendiagramm:

Es die Aufgabenverteilung in einem Unternehmen modelliert werden. Dabei gibt es zu jeder Aufgabe eine Bezeichnung, Kurzbeschreibung und geschätzte Dauer. Die geschätzte Dauer soll nur für Mitarbeiter der Abteilung="Koordination" sichtbar sein. Jeder Mitarbeiter hat einen Vornamen, eine Abteilung und eine E-Mail Adresse.

2. Überführe das Diagramm in Python

BONUS) Betrachte den property Decorator:
https://docs.python.org/3/library/functions.html#property

https://stackoverflow.com/questions/17330160/how-does-the-property-decorator-work-in-python

Wie kann er genutzt werden, um mehr Kontrolle über den Zugriff auf Attribute zu gewinnen? Baue Ihn in deinen Coe