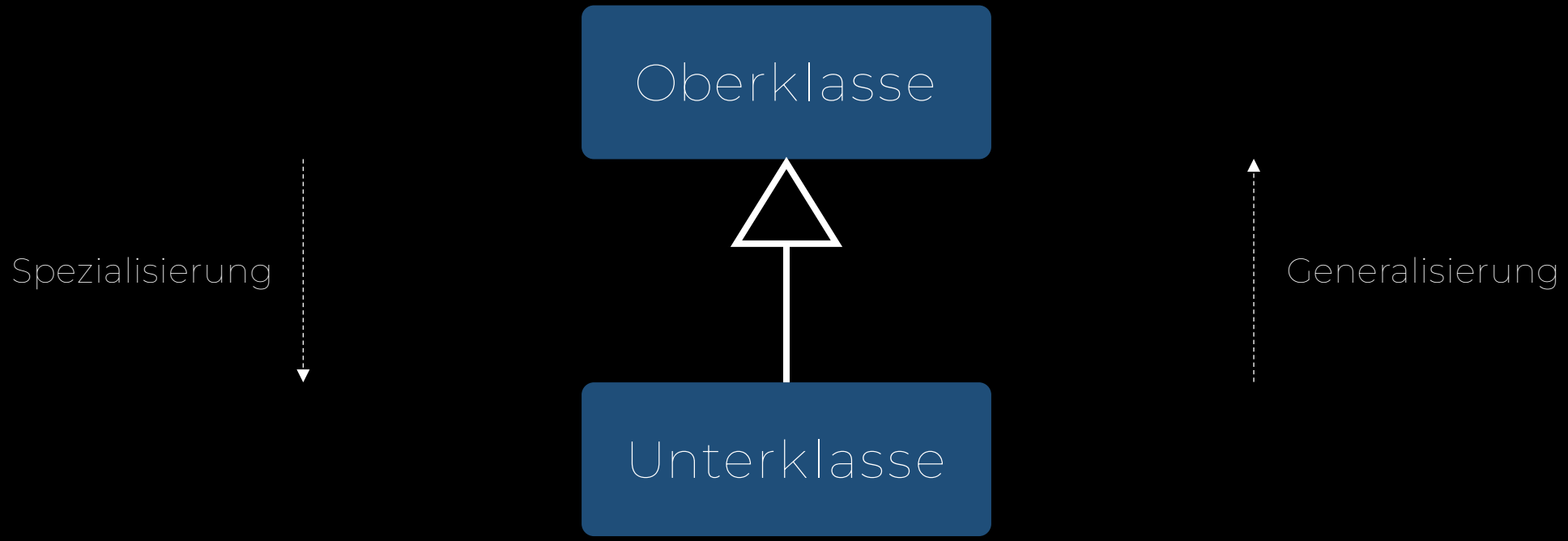
Informatik Q1 Abels

Vererbung

Vererbung


Oberklasse
Spezialisierung
Generalisierung
Unterklasse

Oberklasse

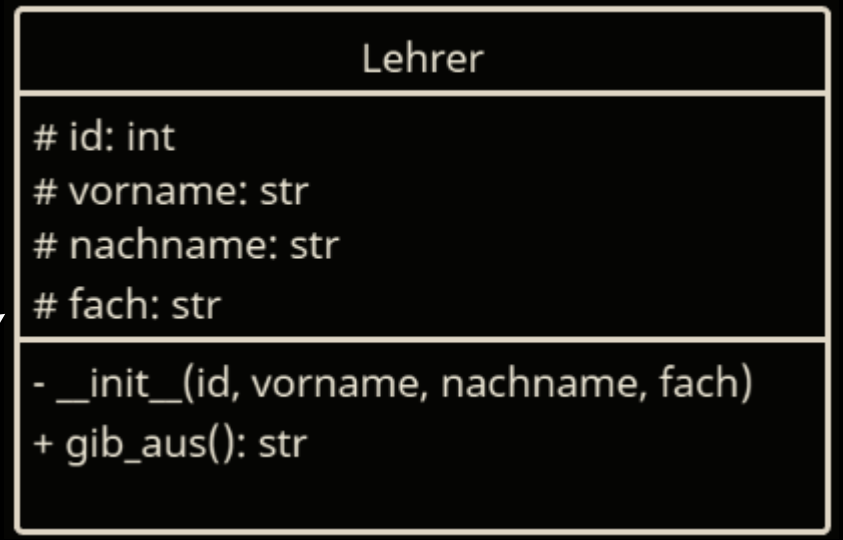
Schule.py

```python
class Lehrer:
    def __init__(self, id, vorname, nachname, fach):
        self._id: int = id
        self._vorname: str = vorname
        self._nachname: str = nachname
        self._fach: str = fach

    def gib_aus(self) -> str:
        print(f'''
            {20*"_"}
            Lehrer
            {20*"-"}
            ID:   {self._id}
            Name: {self._vorname} {self._nachname}
            Fach: {self._fach}
            {20*"_"}
        ''')

abe = Lehrer(42, "Patrick", "Abels", "Informatik")
abe.gib_aus()
```

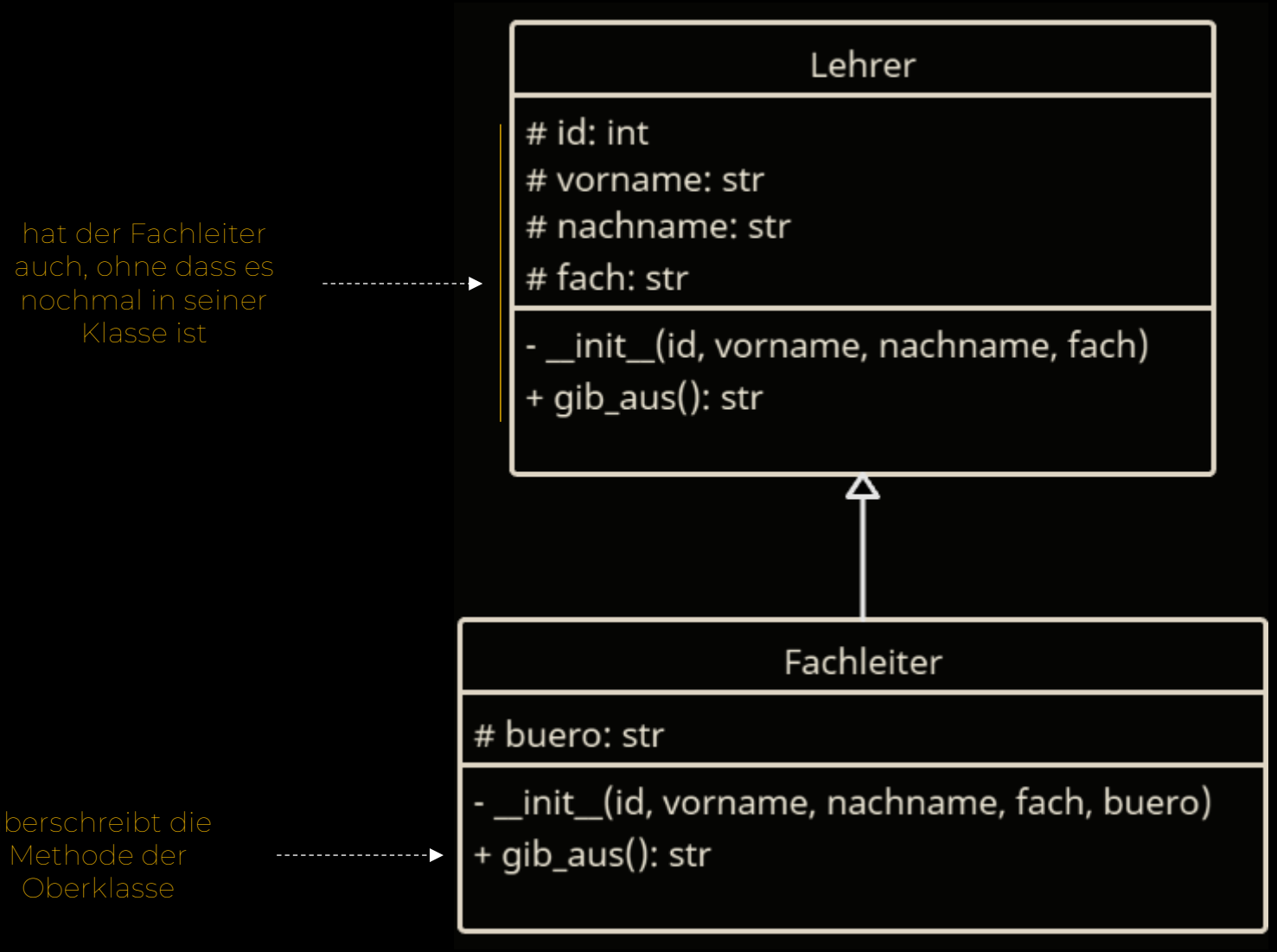
Sichtbarkeit



Terminal

```
____________________
Lehrer
____________________
ID:   42
Name: Patrick Abels
Fach: Informatik
____________________
```

Oberklasse + Unterklasse



Unterklasse

Schule.py

```python
class Lehrer:
    def __init__(self
        self._id: int
        self._vorname
        self._nachnam
        self._fach: s

    def gib_aus(self)
        print(f'''
            {20*"_"}
            Lehrer
            {20*"-"}
            ID:   {se
            Name: {se
            Fach: {se
            {20*"_"}
        ''')
```

ruft Konstruktor von Oberklasse auf

"erbt von"

```python
class Fachleiter(Lehrer):
    def __init__(self, id, vorname, nachname, fach, buero):
        super().__init__(id, vorname, nachname, fach)
        self._buero: str = buero

    def gib_aus(self) -> str:
        print(f'''
            {20*"_"}
            Fachleiter
            {20*"-"}
            ID:   {self._id}
            Name: {self._vorname} {self._nachname}
            Fach: {self._fach}
            Büro: {self._buero}
            {20*"_"}
        ''')

abe = Lehrer(42, "Patrick", "Abels", "Informatik")
abe.gib_aus()

tav = Fachleiter(12, "Rene", "Tavernier", "Physik", "A 106")
tav.gib_aus()
```

Terminal

```
____________________
Lehrer
____________________
ID:   42
Name: Patrick Abels
Fach: Informatik
____________________


____________________
Fachleiter
____________________
ID:   12
Name: Rene Tavernier
Fach: Physik
Büro: A 106
____________________
```

Übung 1

Schreibe ein Programm **Schule**.

a) Implementiere die Klasse **Lehrer** mit den Attributen `id`, `vorname`, `nachname`, `fach1`, `fach2` und der Methode `gib_aus()`.

b) Implementiere die Klasse **Fachleiter**, die von der Klasse **Lehrer** erbt. Sie hat außerdem die Attribute `leitendes_fach` und `buero` und überschreibt die Methode `gib_aus()`.

c) Implementiere die Klasse **Schulleitungsmitglied**, die von der Klasse **Lehrer** erbt. Sie hat außerdem die Attribute `aufgabe: str` und `team: list`, was eine Liste von Lehrern ist, die dessen Team darstellen. Ein Schulleitungsmitglied besitzt zusätzlich die Methoden `beurteilt(lehrer)` und `stellt_team_vor()`.

d) Teste alle Klassen und Methoden mit beliebigen Werten.

e) Stelle das Programm in einem UML dar.

Übung 1

Schule.py

```python
class Lehrer:
    def __init__(self, id, vorname, nachname, fach1, fach2):
        self._id: int = id
        self._vorname: str = vorname
        self._nachname: str = nachname
        self._fach1: str = fach1
        self._fach2: str = fach2

    def gib_aus(self):
        print(f'''
            {20*"_"}
            Lehrer
            {20*"-"}
            ID:     {self._id}
            Name:   {self._vorname} {self._nachname}
            Fächer: {self._fach1}, {self._fach2}
            {20*"_"}
        ''')

class Fachleiter(Lehrer):
    def __init__(self, id, vorname, nachname, fach1, fach2, leitendes_fach, buero):
        super().__init__(id, vorname, nachname, fach1, fach2)
        self._leitendes_fach: str = leitendes_fach
        self._buero: str = buero

    def gib_aus(self):
        print(f'''
            {20*"_"}
            Fachleiter
            {20*"-"}
            ID:             {self._id}
            Name:           {self._vorname} {self._nachname}
            Fächer:         {self._fach1}, {self._fach2}
            leitendes Fach: {self._leitendes_fach}
            Büro:           {self._buero}
            {20*"_"}
        ''')

class Schulleitungsmitglied(Lehrer):
    def __init__(self, id, vorname, nachname, fach1, fach2, aufgabe, team):
        super().__init__(id, vorname, nachname, fach1, fach2)
        self._aufgabe: str = aufgabe
        self._team: list = team

    def beurteilt(self, lehrer):
        print(f"Ich beurteile {lehrer._vorname} {lehrer._nachname} mit der Note 1.")

    def stellt_team_vor(self):
        print(f'''
            {20*"_"}
            {self._vorname} {self._nachname}
            {20*"-"}
            Aufgabe: {self._aufgabe}
            Team:    {[n._nachname for n in self._team]}
            {20*"_"}
        ''')

abe = Lehrer(42, "Patrick", "Abels", "Informatik", "Mathematik")
abe.gib_aus()

tav = Fachleiter(12, "Rene", "Tavernier", "Physik", "Mathematik", "Physik", "A 106")
tav.gib_aus()

das = Schulleitungsmitglied(98, "Sven", "Daschmann", "Chemie", "Erdkunde", "Leitung FB III", [abe, tav])
das.beurteilt(abe)
das.stellt_team_vor()
```

Terminal

```
____________________
Lehrer
____________________
ID:     42
Name:   Patrick Abels
Fächer: Informatik, Mathematik
____________________


____________________
Fachleiter
____________________
ID:             12
Name:           Rene Tavernier
Fächer:         Physik, Mathematik
leitendes Fach: Physik
Büro:           A 106
____________________

Ich beurteile Patrick Abels mit der Note 1.

____________________
Sven Daschmann
____________________
Aufgabe: Leitung FB III
Team:    ['Abels', 'Tavernier']
____________________
```

Tagebucheintrag

Vererbung

Wochenübung

Eine Firma besitzt Geschäftskunden, von denen sie die Telefonnummer, eine E-Mail-Adresse und eine Adresse verwaltet. Ein Geschäftskunde kann ein Lieferant oder ein Kunde sein. Von den Lieferanten sollte der Firmenname und ein Ansprechpartner gespeichert werden. Die Kunden werden durch eine Kundennummer, einen Namen und die Anzahl ihrer Bestellungen erfasst.

a) Modelliere geeignete Klassen **`Geschäftskunde`**, **`Lieferant`** und **`Kunde`** als UML.

b) Implementiere die Klassen in einem Programm **`Firma`**.