

## Aufgabe 4 - Erweitern von Fähigkeiten und mehr

### Ziel

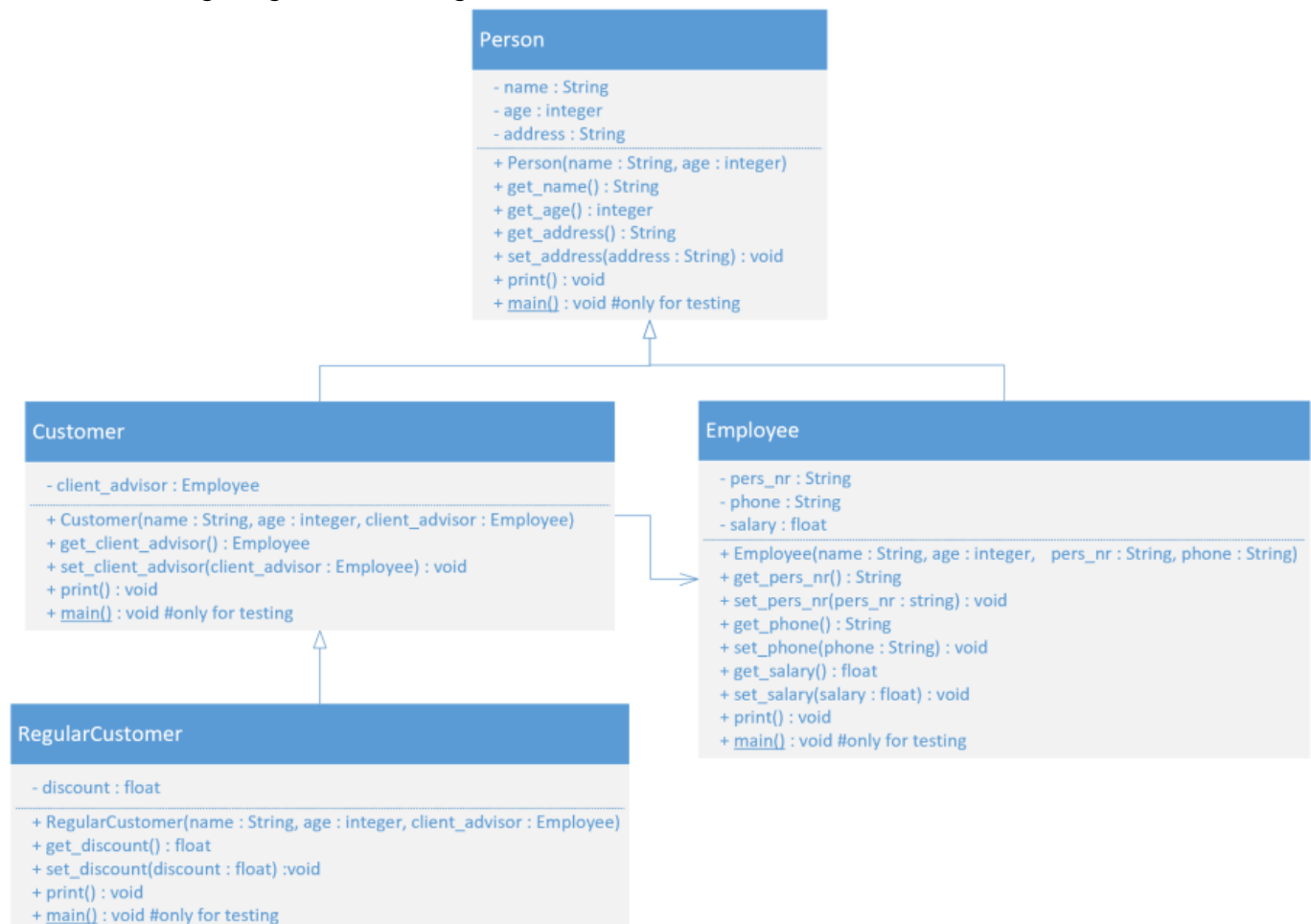
- Sie können aufzeigen, wie Vererbung für die Erweiterung von Fähigkeiten eingesetzt wird.
- Sie können aufzeigen, wie Konstruktoren in Vererbungshierarchien deklariert sein müssen.

### Vorbereitung

Laden Sie das Repo von github-classroom.

### Auftrag

Setzen Sie das gezeigte Klassendiagramm schrittweise um.



### Teilaufgabe 1: Klasse Person

Implementieren und testen Sie die Klasse Person.

Die Methode print liefert (vergleichbar) folgende Ausgabe:

```

Person: Max
      Alter: 2000
      Adresse: Musterdorf
  
```

### Hinweise:

- Name, Alter und Adresse können variieren.
- Der Zeilenumbruch wird durch das Zeichen `'\n'` im Ausgabe-String erwirkt, der Abstand durch das Tabulatorzeichen `'\t'`.
- Fügen Sie keine Leerzeichen ein, da sonst die Ausgabe nicht mit dem erwarteten Resultat übereinstimmt und in der Folge der Unit-Test einen Fehler meldet.

## Teilaufgabe 2: Klasse Employee

Implementieren und testen Sie die Klasse Employee. Sie erbt von Person.

Hier müssen Sie sowohl den Konstruktor (`__init__(...)`) als auch die Methode `print()` der Oberklasse Person aufrufen. Dazu schreiben Sie `super().__init__(...)` bzw. `super().print()` an. Die Parameter richten sich nach dem Konstruktor bzw. der Methode der Oberklasse!

Die Methode `print` liefert (vergleichbar) folgende Ausgabe:

```
Person: Max
  Alter: 2000
  Adresse: Weltdorf
    Personalnummer: 723-123
    Telefon: 01 03 04 05
    Lohn: 6000.0
```

### Hinweise:

- Die führenden Abstände bei den 3 letzten Zeilen werden bei der Ausgabe durch 2 Tabulatorzeichen bewerkstelligt, also `'\t\t'`.

## Teilaufgabe 3: Klasse Customer

Implementieren und testen Sie die Klasse Customer. Sie erbt von Person.

Sie müssen auch hier den in Konstruktor und `print`-Methode jeweils explizit die Funktion der Oberklasse aufrufen.

Die Methode `print` liefert (vergleichbar) folgende Ausgabe:

```
Person: Pia
  Alter: 30
  Adresse: leer
    Kundenberater: Max 099-345-332-123
```

### Hinweise:

- Nach dem Namen des Kundenberaters verwenden Sie bitte ein Tabulatorzeichen, um die Telefonnummer abzusetzen.

## Teilaufgabe 4: Klasse RegularCustomer

Implementieren und testen Sie die Klasse RegularCustomer. Sie erbt von Customer.

Die Methode `print` liefert (vergleichbar) folgende Ausgabe:

```
Person: Pia
  Alter: 1980
  Adresse: Oberdorf
    Kundenberater: Max 099-345-332-123
    Stammkunde mit 2.4% Rabatt
```

## Teilaufgabe 5: Hauptprogramm (main)

Erstellen Sie in der Datei `main.py` das Hauptprogramm.

Es erzeugt je ein Objekt von Person, Employee, Customer und RegularCustomer und setzt noch

fehlende Attribute wie z.B. das Salär.

Zu jedem Objekt wird dann die `print()`-Methode ausgeführt. Das Ergebnis soll dem Screenshot vergleichbar sein.

```

Person: Max
  Alter: 25
  Adresse: leer
-----
Person: Joel
  Alter: 35
  Adresse: leer
  Personalnummer: 723-123
  Telefon: 01 03 04 05
  Lohn: 6000.0
-----
Person: Moritz
  Alter: 45
  Adresse: leer
  Kundenberater: Joel 01 03 04 05
-----
Person: Pia
  Alter: 25
  Adresse: leer
  Kundenberater: Joel 01 03 04 05
  Stammkunde mit 2.4% Rabatt
-----

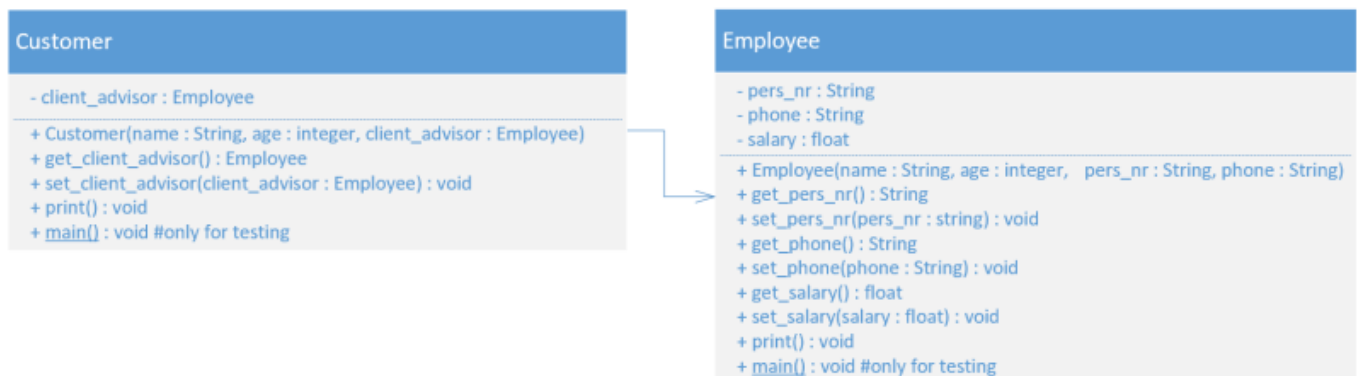
```

### Teilaufgabe 6: Klassen erweitern und Methoden überschreiben

Kopieren Sie das Klassendiagramm in eine Anwendung Ihrer Wahl, um dort mit der Farbe A die Methoden zu markieren, welche eine Klasse erweitern. Gibt es auch Methoden, die das Überschreiben realisieren? Wenn ja, markieren Sie diese mit einer Farbe B.

### Teilaufgabe 7: OO-Technik nutzen

Die gezeigte Lösung ist nicht optimal. Wie aus dem Klassendiagramm (siehe Ausschnitt) ersichtlich ist, hat der Kunde (Customer) unbeschränkt Zugriff auf die Methoden des Mitarbeiters (Employee). Das ist so sicher nicht im Sinne des Mitarbeiters, denn es wäre z.B. sein Lohn durch jeden Kunden einsehbar.



Suchen Sie nach einer Lösung dieses Problems, das sich durch die statische Struktur der Anwendung realisieren lässt!

Es geht also nicht darum, hier mit einem Passwort oder ähnlichen Sicherheitsmechanismen den Zugriff zu überwachen, sondern darum, eine Lösung unter Anwendung der OO-Technik zu realisieren.

## Dauer

2-4 Stunden

## Abgabe

- Teilaufgaben 1 bis 5 über github
- Teilaufgabe 6 und 7 als PDF in Moodle.

---

 © René Probst

From:  
<https://wiki.bzz.ch/> - **BZZ - Modulwiki**

Permanent link:  
[https://wiki.bzz.ch/modul/archiv/m320/learningunits/lu06/aufgaben/lu07-aufgabe\\_4](https://wiki.bzz.ch/modul/archiv/m320/learningunits/lu06/aufgaben/lu07-aufgabe_4)

Last update: **2026/08/30 14:10**

