LU10.A04 - Vererbung mit vielen Facetten



Setzen Sie verschiedene Facetten der Vererbung in einer Applikation ein.

Auftrag

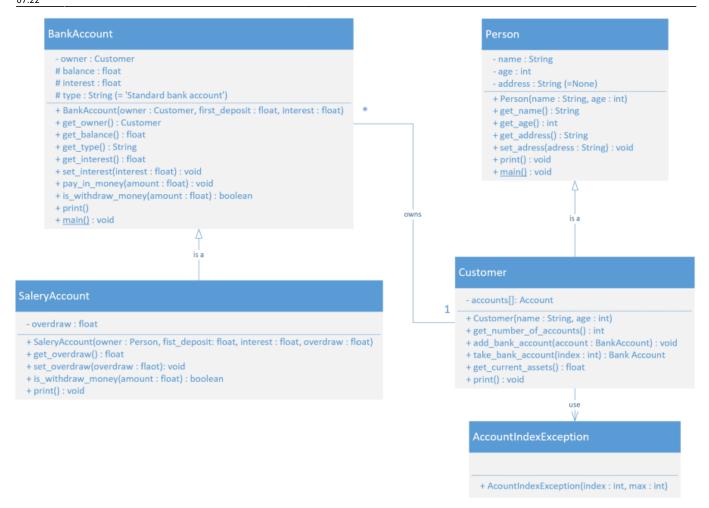
Hinweis

Diese Aufgabe umfasst viel Wissen und Können, das Sie bisher angewendet haben. Schauen Sie sich die entsprechenden Ausführungen der Theorie und Übungen an!

Ausgangslage

In LU10d - Anpassen und Erweitern wird gezeigt, was Überschreiben (overwriting) bewirken soll. Die Aufgabenstellung basiert auf dem Beispiel "BankAccount".

Gegeben ist dazu das folgende Klassendiagramm.



Die Methoden der Klasse BankAccount sollen folgende Funktionen ausführen:

- is_withdraw_money liefert einen angeforderten Geldbetrag, sofern der Saldo (balance) nicht negativ wird. Ist das der Fall, liefert die Methoden False (Geldbezug nicht möglich), sonst True
- pay in money erhöht den Saldo (balance) um den angegebenen Betrag.
- get owner liefert die Referenz auf ein Customer-Objekt.
- get_balance liefert den Saldo des Kontos.
- get interest liefert den Zinssatz des Kontos.
- set_interest legt den Zinssatz fest.
- get_type liefert den Typ des Kontos. Dieser Wert wird fix im Konstruktor der jweiligen Klasse auch der abgeleiteten - festgelegt.
- print gibt den Namen des Kunden sowie den aktuellen Saldo und den Zinsatz aus.

Die Methoden der Klasse SalaryAccount solllen zusätzlich folgende Funktione ausführen:

- is_withdraw_money liefert einen angeforderten Geldbetrag, sofern der Saldo (balance) plus der Überzug (overdraw) nicht überschritten werden. Ist das der Fall, liefert die Methoden False (Geldbezug nicht möglich), sonst True. Hier benötigen Sie den Zugriff auf das Attribut balance in der Oberklasse.
- set overdraw legt den möglichen Überzug (Negativsaldo) fest.
- get overdraw liefert den möglichen Überzug.
- print gibt den Namen des Kunden sowie den aktuellen Saldo, den Zinsatz und den Überzug aus.

https://wiki.bzz.ch/ Printed on 2025/11/06 16:50

Bei der Klasse Customer setzen wir nun voll auf die OO-Technik. Wir verwenden nämlich die Klasse Person aus Aufgabe 4 der LU06 als Basis und leiten Sie ab. Daher muss die Klasse Customer nur noch die Methoden zur Verwaltung der Referenzen zu den Konto-Objekten realisieren. Hinweis: Die Klasse Person ist im Repo bereits vorhanden

- get_current_assets liefert das summierte Vermögen aller Konten.
- get number of accounts liefert die Anzahl der Konten, die der Kunde aufweist.
- take_bank_account liefert die Referenz zu einem Konto, das durch den index angegeben wird. Dabei kann dies eine Referenz auf BancAccount oder aber auch SalaryAccount sein. Bei falschm Index soll eine AccountIndexException geworfen werden.
- add bank account fügt eine Referenz für ein Konto zu.
- print gibt Namen und Jahrgang der Person aus sowie die Angaben zu allen Konti. Dazu wird die print-Methode des Kontos aufgerufen

Schritt 1

- 1. Akzeptieren Sie das Assignment im GitHub Classroom.
- 2. Testen Sie die Klasse Person mit der Testklasse TestPerson in test_person.py. Dieser Test muss fehlerfrei ablaufen, da die Klasse Person ja bereits besteht.
- 3. Erstellen Sie die Exception-Klasse AccountIndexException. Die Fehlermeldung soll wie folgt aussehen:
 - Ungültiger Indexwert: {max} Einträge vorhanden, Nummer {index} gefordert
- 4. Erstellen Sie nun die Klasse Customer. Implementieren Sie die Methoden gemäss der obigen Beschreibung.
 - Hinweis: Sie können diese Klasse noch nicht testen, da die Klasse BankAccount fehlt. Das werden Sie später aber nachholen.
- 5. Erstellen Sie nun die Klasse BankAccount gemäss der Beschreibung oben, aber ohne main-Methode.
- 6. Fügen Sie der Klasse BankAccount nun noch die main-Methode zu. Die Ausgabe muss wie folgt aussehen

Kunde : Pia

Kontotyp: Standard bank account

Saldo: 3500.0 Zins : 1.75

- 7. Testen Sie die Klasse BankAccount, bevor Sie diese für die weitere Arbeit nutzen. Dazu verwenden Sie die Klasse TestBankAccount in test bank account.py.
- 8. Testen Sie nun die Klasse Customer. Dazu verwenden Sie die Testklasse TestCustomer in der Datei test customer.py.
- 9. Erstellen Sie die Klasse SalaryAccount gemäss der Beschreibung oben.
- 10. Testen Sie die Klasse. Verwenden Sie dazu die Klasse TestSalaryaAccount in test_salary_account.py.

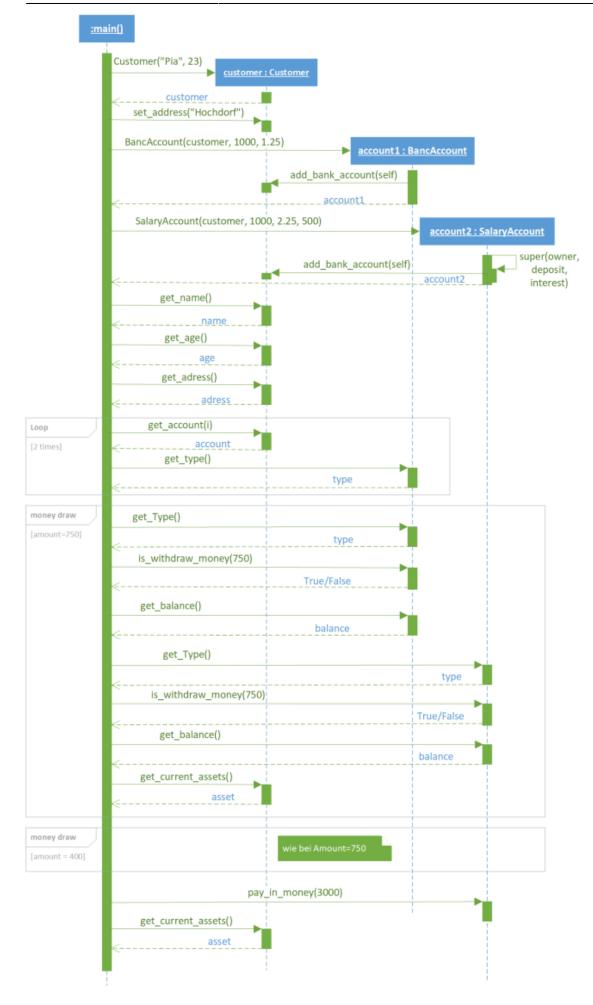
Wenn alle Tests fehlerfrei laufen, darf die Annahme getroffen werden, dass der Code die geforderten Spezifikationen erfüllt und die Klassen somit für eine Anwendung genutzt werden können.

Vorgehen Teil 2

Implementieren Sie die main-Methode gemäss dem Sequenzdiagramm.

- 2. Ergänzen Sie wenn nötig den Konstruktoren von BancAccount für die Übergabe der Referenz an die Klasse Customer.
- 3. Den Teil "money draw" im Diagramm führen Sie zwei Mal aus. Einmal mit einem Bezug von 700 und einmal mit einem Bezug von 400. Hier liefert Ihnen die Methode withdraw_money() den Wert True, wenn ein Bezug von Geld möglich ist. Wenn nicht weil Saldo zu klein liefert sie False. Dem entsprechend soll die Ausgabe am Bildschirm sein. Vergleichen Sie dazu unten den Printscreen.

https://wiki.bzz.ch/ Printed on 2025/11/06 16:50



Erwartetes Ergebnis

Wenn Sie die Klassen gemäss Klassendiagramm und den Ablauf gemäss Sequenzdiagramm implementiert haben, sollten Sie - unter Berücksichtigung einiger erklärender print()-Befehle in main - ein vergleichbares Ergebis erhalten.

Alle gelb markierten stellen werden durch Attribute geliefert bzw. sind im print-Befehl der jeweiligen Klasse zu finden.

```
Angaben zu Kunde
     Name: Pia
     Alter: 23
     Adresse: Hochdorf
Angaben zum den Konti
      Standard bank account
      Salary bank account
von jedem Konto 750.0 beziehen
     Bezug von Standard bank account
     Saldo = 250.0
     Bezug von Salary bank account
     Saldo = 250.0
Aktuelles Vermögen: 500.0
von jedem Konto noch einmal 400.0 beziehen
     Bezug von Standard bank account
     Fehler: Bezug ist zu hoch für Saldo von 250.0
     Bezug von Salary bank account
     Saldo = -150.0
Aktuelles Vermögen: 100.0
Ende Monat: 3000.- Lohn wird eingezahlt
Aktuelles Vermögen: 3100.0
```

Dauer

3-5 Stunden (als Hausaufgabe)

Abgabe

auf github-classroom

M320-LU10



BY NO SA René Probst, bearbeitet durch Marcel Suter

https://wiki.bzz.ch/ Printed on 2025/11/06 16:50

From:

https://wiki.bzz.ch/ - BZZ - Modulwiki

Permanent link:

 $https://wiki.bzz.ch/modul/m320_2024/learningunits/lu10/aufgaben/vererbung?rev=1730096570$

Last update: **2024/10/28 07:22**

