


Praktikum 7

Folgendes Szenario über Personen ist vorgegeben:

- Jede Person hat einen Namen (Typ string), ein Geburtsdatum (Typ DATS) und kann sich vorstellen.
- Um das Alter einer Person in Jahren zu bestimmen existiert die funktionale Methode `get_alter()`, die das aktuelle Alter der Person in Abhängigkeit von seinem Geburtsdatum berechnet und zurückgibt. Zur Erinnerung: Funktionsbaustein `COMPUTE_YEARS_BETWEEN_DATES` kann verwendet werden.
- `vorstellen()` ist eine funktionale Methode und liefert einen String der Art "Hallo, ich heiße <name> und bin <alter> Jahre alt." zurück. Achtung: Keine WRITE-Anweisung in der Methode verwenden, sondern der Text wird von der Methode **zurückgegeben**.
- Manche Personen sind Studenten. Diese erwähnen immer ihre Semesterzahl, wenn sie sich vorstellen: "Hallo, ich heiße <name> und bin <alter> Jahre alt. Ich studiere im <x>. Semester!"
- Andere Personen sind Arbeiter. Jeder Arbeiter hat eine bestimmte Anzahl Wochenstunden zu arbeiten und nennt diese auch, wenn er sich vorstellt: "Hallo, Hallo, ich heiße <name> und bin <alter> Jahre alt. Ich arbeite <y> Stunden pro Woche!". Achtung: Arbeiter sagen zweimal „Hallo“!

1. Klassen mit Vererbung

- a) Erstellen Sie die Klasse `ZXXX_XX_CL_PERSON`, mit geeigneten Attributen und Methoden. Die Attribute sind jeweils `private`. Für die Methoden `vorstellen()` und `get_alter()` verwenden Sie geeignete Zugriffsmodifizierer. Die Methode `get_alter()` soll als `final` gekennzeichnet werden. Hinweis: dafür müssen Sie bei der Methode zu den Eigenschaften wechseln → 
- b) Erstellen Sie die Klassen `ZXXX_XX_CL_STUDENT` und `ZXXX_XX_CL_ARBEITER`.
 - Für `STUDENT` und `ARBEITER` wird jeweils die Oberklasse `PERSON` hinterlegt. Ergänzen Sie in jeder Klasse das zusätzlich erforderliche Attribut und legen jeweils einen neuen Konstruktor an.
 - Redefinieren Sie in beiden Klassen die Methode `vorstellen()`.

2. Polymorphie:

- a) Zum Testen ihrer Klassen erstellen Sie einen Report `ZXXX_XX_PERSONEN_PROG`. Legen Sie darin jeweils ein Objekt der Klasse `PERSON`, `STUDENT` und `ARBEITER` an (siehe Screenshot unten).
- b) Legen Sie eine interne Tabelle passenden Typs an, die (Referenzen auf) Personen speichern kann und legen Sie die erzeugten Personen in dieser ab.
- c) Programmieren Sie eine Schleife über die interne Tabelle und lassen sie die enthaltenen Personen sich vorstellen.

Personen Vererbung

```
Hallo, ich heiße Laura und bin 14 Jahre alt.  
Hallo, ich heiße Max und bin 21 Jahre alt. Ich studiere im 3. Semester!  
Hallo, Hallo, ich heiße Tom und bin 27 Jahre alt. Ich arbeite 40 Stunden pro Woche!
```

3. Abstrakte Methoden/Klassen:

Erweitern Sie die ABAP-Klassen um folgende neue Anforderungen (erst komplett lesen!):

- a) Es soll nun außer Studenten und Arbeitern auch Angestellte geben.
- b) Legen Sie dafür die Klasse ZXXX_XX_CL_ANGESTELLTER an. Angestellte haben eine Besoldungsgruppe zwischen 1 und 16. Falls beim Anlegen eines Objektes eine Besoldungsgruppe größer als 16 übergeben wird, dann wird diese auf 16 korrigiert. Für Werte kleiner als 1 werden diese auf 1 korrigiert. Im Screenshot unten können Sie entnehmen, wie sich Angestellte vorstellen.
- c) Jeder Person ist entweder Student, Arbeiter oder Angestellter. Es können also von nun an keine Objekte der Klasse PERSON mehr erstellt werden. Die Klasse PERSON ist ab sofort abstrakt.
- d) Jede Person kann Auskunft darüber geben, wieviel Gehalt sie monatlich erhält. Dafür wird innerhalb der Klasse PERSON die abstrakte funktionale Methode `get_gehalt()` angelegt, die das Gehalt einer Person zurückgibt (mit zwei Nachkommastellen → Datenelement DEC8_2 verwenden).
- e) Das Gehalt der einzelnen Personengruppen errechnet sich wie folgt:
 - i. Manche Studenten erhalten Bafög, andere nicht. Wenn sie Bafög erhalten, sind dies genau 500 Euro im Monat. Für die Klasse STUDENT ist dafür ein weiteres Attribut erforderlich (vom Typ C). Erinnerung: „X“ ist True und „ “ (Leerzeichen) ist False.
 - ii. Arbeiter erhalten 15 Euro je Stunde, die sie arbeiten. Jeder Arbeitsmonat hat genau vier Wochen.
 - iii. Angestellte erhalten ein Monatsgehalt, dass von der Berufserfahrung (=Alter) abhängig ist. Es errechnet sich in unserem Szenario nach folgender Formel:
Besoldungsgruppe * Alter * 10 Euro.

4. Testdaten:

Verwenden Sie geeignete Testdaten, die alle verschiedenen Konstellationen testen (Student mit/ohne Bafög), verschiedene Besoldungsgruppen (gültig/ungültig), usw.

Speichern Sie Ihre Testdaten in einer internen Tabelle ab und geben die einzelnen Tabelleneinträge mit Hilfe einer Schleife aus. Für alle Personen werden auch deren Gehälter ausgegeben. Die Gehälter werden zusätzlich aufsummiert und am Ende ausgegeben.

Ausgabe:

Personen Vererbung	
Hallo, ich heiße Max und bin 21 Jahre alt. Ich studiere im 3. Semester!	Gehalt: 500,00
Hallo, ich heiße Eva und bin 22 Jahre alt. Ich studiere im 2. Semester!	Gehalt: 0,00
Hallo, Hallo, ich heiße Tom und bin 27 Jahre alt. Ich arbeite 40 Stunden pro Woche!	Gehalt: 2.400,00
Hallo, Hallo, ich heiße Nicole und bin 31 Jahre alt. Ich arbeite 20 Stunden pro Woche!	Gehalt: 1.200,00
Hallo, ich heiße Franziska und bin 28 Jahre alt. Ich habe die Besoldungsgruppe 5.	Gehalt: 1.400,00
Hallo, ich heiße Johannes und bin 30 Jahre alt. Ich habe die Besoldungsgruppe 16.	Gehalt: 4.800,00
Gesamtgehälter: 10.300,00	

5. Erweiterungen:

- a) Arbeiter stellen sich ab sofort folgendermaßen vor: „Servus, ich bin <name>. Mein Alter ist <alter> und arbeite <y> Stunden pro **Monat!**“. Um innerhalb der Klasse Arbeiter zugriff auf das Attribut name zu haben, erstellen Sie eine öffentliche funktionale Methode `get_name()`, die den Namen der Person zurückgibt.
- b) Erweitern Sie die Klasse PERSON um ein privates statisches Attribut GN_ANZ_PERSONEN. Dieses zählt die Anzahl aller bisher erzeugten Personen automatisch mit (egal ob Studenten, Arbeiter oder Angestellte). D.h. wird ein neues Objekt erzeugt, dann wird der Wert von GN_ANZ_PERSONEN um 1 erhöht. Erstellen Sie auch eine funktionale Methode `get_anz_personen()`, die diese Anzahl zurückliefert und testen diese Methode durch einen Aufruf im Hauptprogramm.