LS 1 – Nutzungsdaten eines IT-Sytems zusammenfügen und speichern

Ausgangssituation

In einem mittelgroßen Unternehmen werden Daten über Nutzungsverhalten und Probleme mit der IT dezentral in unterschiedlichen Rhythmen und Formaten bereitgestellt. Erste Versuche diese in einer zentralen Datenbank zusammenzufassen sind in einem objekt-orientierten System vorhanden. Sie sollen diese weiter ausbauen und eine solide Basis für die spätere Datenauswertung schaffen.

Teil A: Grundbegriffe der Objekt-Orientierten-Programmierung

Aufgabe 1 – Was ist Objekt-Orientierung?

Beantworten Sie mit Hilfe der Informationen aus den zwei Videos die folgenden Fragen:

a) Beschreiben Sie mit eigenen Worten den Unterschied zwischen einem Objekt und einer Klasse.
b) Bei der Objekt-Orientierung besitzen Klassen Attribute und Methoden. Was steckt programmiertechnisch dahinter?
c) Was versteht man unter Vererbung?
d) Was ist ein Konstruktor?
e) Wofür steht die Abkürzung UML? Wozu dient UML?
f) Definieren Sie Objekt-Orientierte-Programmierung.

Beantworten Sie die folgenden Fragen mit Hilfe eigener Recherchen:

g) Nennen Sie mindestens drei Diagrammarten, welche UML bietet?
h) Nennen Sie mindestens drei klassische Objekt-Orientierte-Programmiersprachen.
i) Was wird in einem UML Klassendiagramm dargestellt? Erstellen Sie ein kleines Beispiel.
j) Was wird in einem UML Objektdiagramm dargestellt? Erstellen Sie ein kleines Beispiel.

UML-Diagramme zeichnen:

k) Beschreiben Sie das Beispiel aus Video 1 mit einem UML-Diagramm.

Aufgabe 2 - Python und Objekt-Orientierung (I)

a) Programmieren Sie das Beispiel aus Video 1 mit Python. Benutzen Sie dabei selbst erstellte Klassen und Objekte. Speichern Sie das Programm unter dem Namen

`aufgabe_ls1_02.py` ab.

b) Programmieren Sie das Beispiel aus Video 2 (Tier-Klasse und davon instanziierte Objekte; keine Vererbung) mit Python. Benutzen Sie dabei selbst erstellte Klassen und Objekte. Speichern Sie das Programm unter dem Namen **`aufgabe_ls1_03.py`** ab.

Aufgabe 3 - Aggregation und Komposition

Ein Kunde kann maximal zwei Konten besitzen. Jeder Kunde besitzt einen Vornamen und einen Nachnamen. Das Konto besitzt eine Kontonummer, ein Überziehungslimit und einen Kontostand.

a) Erstellen Sie ein UML-Klassendiagramm, welches die obige Situation als Aggregation beschreibt.
b) Erstellen Sie ein UML-Klassendiagramm, welches die obige Situation als Komposition beschreibt.
c) Worin unterscheiden sich die beiden Lösungen?

Aufgabe 4 - Modellierung

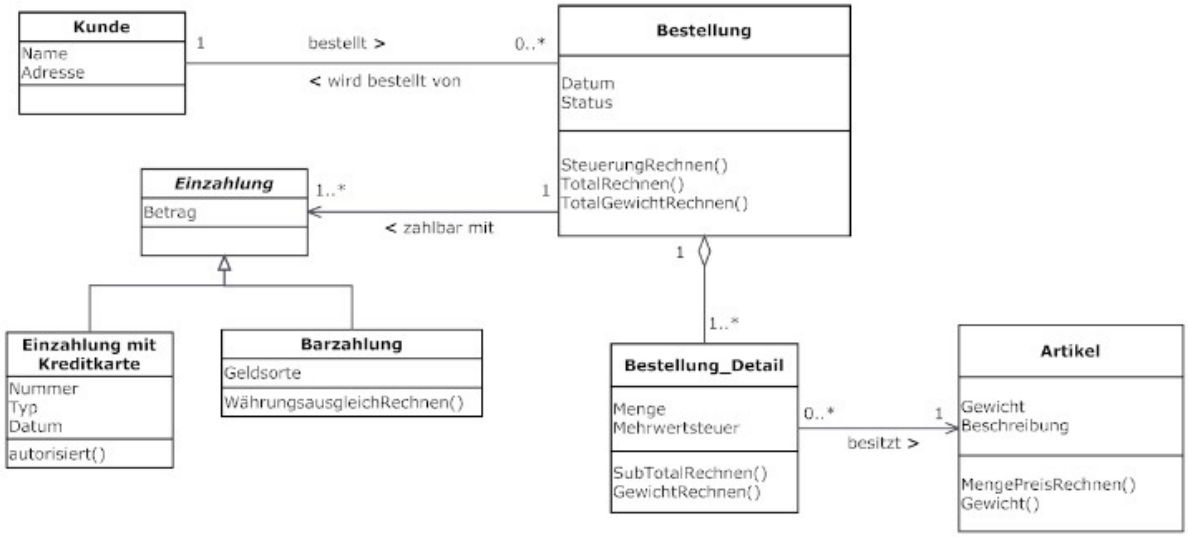
Situationsbeschreibung:

Eine Person kann Geschäftspartner oder Vertreter sein. Es kann Personen geben, die weder Geschäftspartner noch Vertreter sind. Ein Vertreter kann Abteilungsleiter sein, muss es aber nicht. Eine Person kann eine oder mehrere Telefonnummern haben.

Aufgabe:

- Erstelle ein UML Klassendiagramm, welches die obige Situation modelliert.

Aufgabe 5 - Diagramm-Interpretation



Bestimmen Sie, ob die folgenden Aussagen zum Klassendiagramm richtig oder falsch sind.

Es kann im System Kunden geben die nie eine Bestellung durchgeführt haben.
Die Klasse Einzahlung ist die Oberklasse der Klasse Bestellung.
Jedes Objekt der Klasse Bestellung_Detail besitzt genau einen Artikel.
Alle Einzahlungen mit Kreditkarte haben einen Betrag.
Es ist möglich, dass ein Artikel keine Assoziation mit einem Bestellung_Detail besitzt.
Jedes Bestellung_Detail, das Teil einer Bestellung ist, hat seinen eigenen Status und sein eigenes Datum.

Aufgabe 6 - Python und Objekt-Orientierung (II)

a) Das objekt-orientierte Konzept der **Vererbung** lässt sich auch mit Python umsetzen. Recherchieren Sie, wie dies geht und erstellen Sie ein kleines Beispiel hierzu mit Python. Dokumentieren Sie das Beispiel mit UML. Speichern Sie das Programm unter dem Namen **`aufgabe_ls1_04a.py`** ab.
b) Die objekt-orientierten Konzept der **Klassenattribute** und **Klassenmethoden** lassen sich auch mit Python umsetzen. Recherchieren Sie, wie dies geht und erstellen Sie ein kleines Beispiel hierzu mit Python. Dokumentieren Sie das Beispiel mit UML. Speichern Sie das Programm unter dem Namen **`aufgabe_ls1_04b.py`** ab.
c) Die objekt-orientierten Konzepte von **Konstruktoren** und **Destruktoren** lassen sich auch mit Python umsetzen. Recherchieren Sie, wie dies geht und erstellen Sie ein kleines Beispiel hierzu mit Python. Dokumentieren Sie das Beispiel mit UML. Speichern Sie das Programm unter dem Namen **`aufgabe_ls1_04c.py`** ab.

Teil B: Nutzungsdatensystem erweitern (optional)

Aufgabe 7 - Einarbeitung in das bestehende Nutzungsdatensystem

Machen Sie sich mit dem vorhandenen Nutzungsdaten-System V1.0 vertraut.

a) Erstellen Sie ein Kontextdiagramm und ein UML-Anwendungsfalldiagramm für das Nutzungsdatensystem.
b) Beschreiben Sie die zugrunde liegende Datenbankstruktur mit einem Relationen-Schemata und einem Entity-Relationship-Diagramm.
c) Beschreiben Sie die Struktur des Python-Programms mit einem UML-Klassendiagramm.

Aufgabe 8 - Nutzungsdatensystem V2.0

```
Nutzungsdatensystem Datenmanagement
-----------------------------------
1. Stammdatenmagement
   (10) Benutzerdaten importieren
   (11) Benutzerdaten löschen
   (12) Rechnerdaten importieren
   (13) Rechnerdaten löschen
   (14) Anwendungsdaten importieren
   (15) Anwendungsdaten löschen
   (16) Anwendungskategorien importieren
   (17) Anwendungskategorien löschen
2. Bewegungsdatenmagement
   (20) Bewegungsdaten eines Tages importieren
   (21) Bewegungsdaten eines Tages löschen
   (22) Alle Bewegungsdaten löschen
3. Datenbestandsinformationen
   (30) Statistik Stammdaten
   (31) Statistik Bewegungsdaten
   (32) Datengüte Bewegungsdaten
(0) Beenden
Auswahl:
```

Führen Sie die folgenden Erweiterungsarbeiten am Nutzungsdatensystem aus:

a) Bei den Stammdaten gibt es neben der Rechnerdatei auch noch eine Datei für **Rechnerkategorien**. Diese wurde bisher nicht genutzt. Ergänzen Sie die Funktionalitäten für das Importieren (18) und das Löschen (19) von Rechnerkategorien. Passen Sie den Import von Rechnerdaten an, so dass auch die Kategorie mit abgespeichert wird. Ergänzen Sie die Stammdatenstatistik. Sie können sich bei diesem Erweiterungspunkt an der vorhanden Lösung mit Anwendungen und Anwendungs-kategorien orientieren.
b) Ergänzen Sie die Funktion **Datengüte für Bewegungsdaten (32)**. Diese soll tabellarisch folgende Spalten anzeigen:
   - Tag der Daten
   - Tag und Zeit des Imports
   - Rechnerdatensätze OK
   - Rechnerdatensätze verworfen
   - Anwendungsdatensätze OK
   - Anwendungsdatensätze Verworfen

   Alle Daten können aus der Tabelle `ndsimporte` entnommen werden.
c) Ergänzen Sie die Funktion **Statistik für Bewegungsdaten (31)**. Diese soll zuerst tabellarisch folgende Spalten anzeigen:
   - Tag der Daten
   - Rechnerdatensätze OK
   - Anwendungsdatensätze OK

   Anschließend sollen für die Anzahl der Rechnerdatensätze pro Tag und die Anzahl der Anwendungsdatensätze pro Tag jeweils das Minimum, das Maximum, der arithmetische Mittelwert und die Standardabweichung dargestellt werden.

Speichern Sie das Programm unter dem Namen **`aufgabe_ls1_01`** ab.

Teil C: Daten auswerten

Aufgabe 9 - Statistische Auswertung mit dem Modul `statistics`

Ein Produktionsprozess liefert täglich eine Datei, welche die Apha-Werte der produzierten Bauteile enthält. Dieser soll innerhalb den Grenzwerte von 95 bis 105 jeweils einschließlich liegen. Die Apha-Werte müssen Sie auswerten. Erstellen Sie hierzu ein Python Programm mit folgender Funktionalität:

- Der Benutzer muss zuerst den Namen der Datei mit den Alpha-Werten und das Produktionsdatum eingeben.
- Das Programm liest anschließend die Alpha-Werte ein und bestimmt folgende statistischen Kennwerte. Die Kennwerte werden auf dem Bildschirm ausgegeben.
  - Produktionstag
  - Anzahl
  - Mittelwert Alpha-Werte
  - Standardabweichung Alpha-Werte
  - Anzahl im Toleranzbereich (absolut und relativ)
  - Anzahl über oberem Grenzwert (absolut und relativ)
  - Anzahl unter unterem Grenzwert (absolut und relativ)
- Auf Wunsch werden die statistischen Kennwerte in eine Datei geschrieben. Der Name der Ausgabedatei ist der Name der Eingabedatei erweitert um „_statistik“.

Benutzen Sie das Modul `statistics`. Speichern Sie das Programm unter dem Namen **`aufgabe_ls1_08`** ab. Zum Testen steht Ihnen Dateien mit Alpa-Werten zur Verfügung (`alpa_20220704.txt, alpa_20220705.txt`).

Optionale Erweiterungen:

- Speichern Sie die Kennwerte in einer Datenbank. Diese müssen Sie hierzu selbst entwerfen.
- Visualisieren Sie Ihre Lösung mit geeigneten Diagrammen.

Aufgabe 10 - Daten zusammenfassen mit Pandas

Nach einer Modernisierung des Produktionsprozesses werden täglich neben den Alpa-Werten auch die Beta-, Gamma- und Delta-Werte erfasst. Alle sollen innerhalb den Grenzwerte von 95 bis 105 jeweils einschließlich liegen. Die Werte stehen in Form einer CSV-Datei zur Verfügung, wobei jede Zeile die Bauteilnummer, und die vier Werte in alphabetischer Reihenfolge enthält. Die Daten sollen in einer Datenbank (`aufgabe_ls1_09.sql`) gespeichert werden. Die Datenbank liegt schon vor.
Erstellen Sie hierzu ein Python Programm mit folgender Funktionalität:

- Der Benutzer muss zuerst den Namen der Eingabedatei mit den Werten eingeben.
- Das Programm liest anschließend die Werte in einen Pandas-Dataframe ein.
- Das Programm bereinigt die Werte, indem es
  - Doppeleinträge entfernt
  - Zeilen mit fehlenden Einträgen werden entfernt
  - Werte, die größer als 120 sind auf 120 korrigiert und Werte die kleiner als 80 sind auf 80 korrigiert.
- Zuletzt sollen die Werte in der Datenbank gespeichert werden.

Benutzen Sie das Modul `Pandas`. Speichern Sie das Programm unter dem Namen **`aufgabe_ls1_09`** ab. Zum Testen steht Ihnen die Dateien zur Verfügung `alpa_20220910.txt, alpa_20220911.txt` zur Verfügung.

Aufgabe 11 - Reflexion Datenauswertung

a) Wozu dient das Modul `statistics`?
b) Welche Funktion des Moduls `statistics` kann man nutzen, um den arithmetischen Mittelwert zu berechnen?
c) In welcher Form müssen die Daten der Funktion bereitgestellt werden?
d) Was ist der Unterschied zwischen arithmetischen Mittelwert, Median und Modalwert (Modus)? Wie lauten die entsprechenden Funktionen des Moduls `statistics`? Welchen kann man auch bei Daten vom Typ string verwenden?
e) Gegeben sind die neun Zahlen 2, 1, 3, 3, 7, 8, 8, 8, 888. Bestimmen Sie von Hand oder nur mit Hilfe eines Taschenrechners arithmetischen Mittelwert, Median und Modalwert (Modus).
f) Wie würden die Berechnungen aus (e) mit `statistics` realisiert? Geben Sie hierzu beispielhaften Python-Code an.
g) Welche Funktionen des Moduls `statistics` kann man nutzen, um die Standardabweichung zu berechnen? Wozu dient die Standardabweichung?
h) Nennen Sie drei Einsatzmöglichkeiten für das Modul `Pandas`?
i) Am häufigsten arbeiten `Pandas` mit dem Datentyp `Dataframe`. Was muss man sich darunter vorstellen?
j) Nennen Sie einige Dateiformate, aus denen `Pandas` Daten einlesen kann?
k) Welche Methode gibt es bei `Pandas` um Datensätzen mit fehlenden Einträgen zu löschen?
l) Welche Methode gibt es bei `Pandas` um Dupplicate zu entfernen?

Teil D: Zusatzaufgaben

Aufgabe 12 - Sachverhalt modellieren

Gegeben ist der folgende Sachverhalt.

> Jede Person hat einen Namen, eine Telefonnummer und E-Mail.
> Jede Wohnadresse wird von nur einer Person bewohnt. Es kann aber sein, dass einige Wohnadressen nicht bewohnt sind. Den Wohnadressen sind je eine Strasse, eine Stadt, eine PLZ und ein Land zugeteilt. Alle Wohnadressen können bestätigt werden und als Beschriftung (für Postversand) gedruckt werden.
> Es gibt zwei Sorten von Personen: Student, welcher sich für ein Modul einschreiben kann und Professor, welcher einen Lohn hat. Der Student besitzt eine Matrikelnummer und eine Durchschnittsnote.

Modellieren Sie diesen Sachverhalt mit einem UML Klassendiagramm.

Aufgabe 13 - Sachverhalt modellieren

Entwerfen Sie ein vollständiges Klassendiagramm.

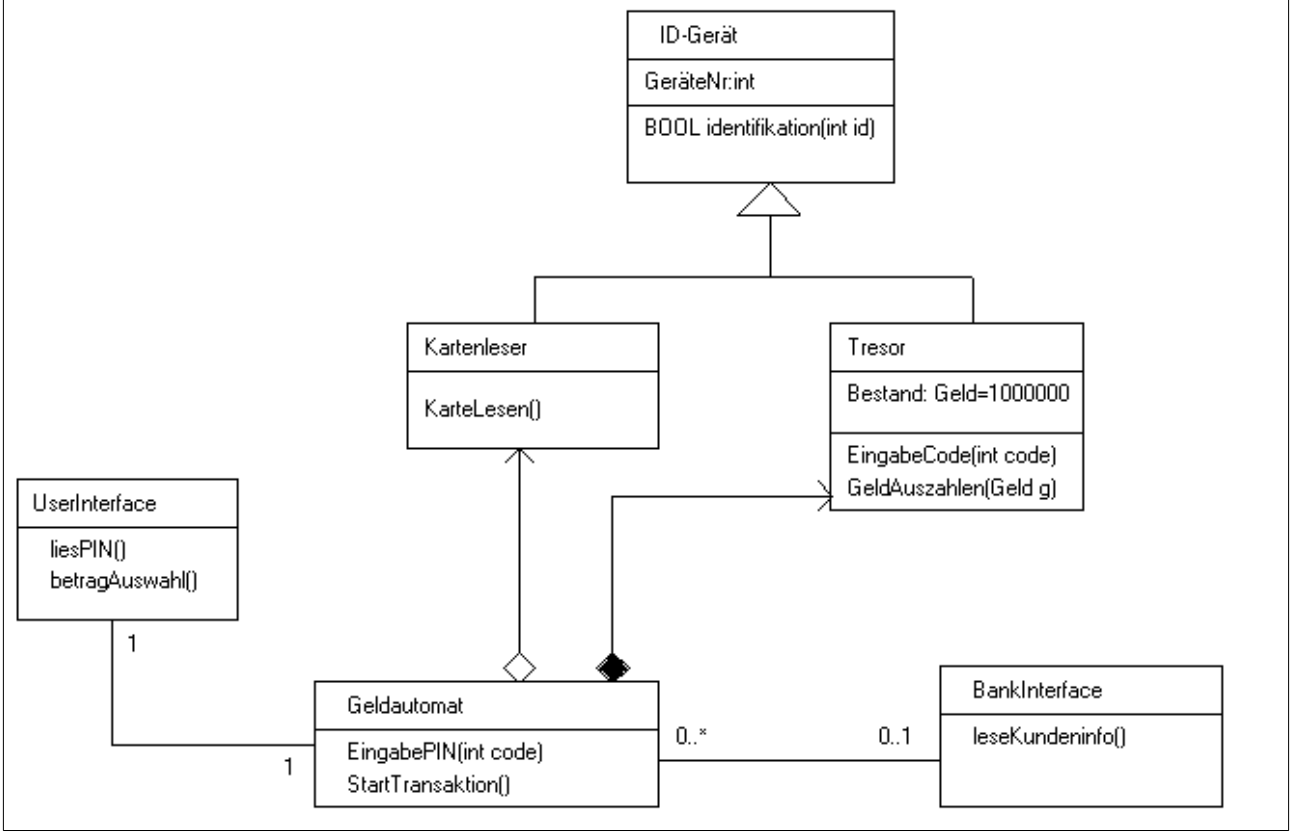
> Allgemeine Infos zum Aufgabenumfeld:
> Sie entwickeln Software für ein Computer-Schachspiel. Ihre Aufgabe ist der Entwurf des Spielbretts mit den Figuren. Zeigen Sie bei Assoziationen die Richtung des Zugriffs durch Pfeile an
> Die folgenden Informationen sollen darstellt werden:
> Ein Schach-Spiel besteht aus einem Schachbrett und 2 Mannschaften. Das Schachbrett besteht aus 64 Feldern, die jeweils eine x- und eine y-Koordinate haben.
> Die Mannschaften unterscheiden sich durch die Farbe. Jede Mannschaft besteht aus insgesamt 16 Figuren. Das sind 8 Bauern, 2 Türme, 2 Läufer, 2 Springer, 1 Dame und 1 König.
> Sorgen Sie dafür, dass die folgenden Informationen im Modell enthalten sind:
> - Jede Figur steht entweder auf einem Feld oder wurde bereits geschlagen
> - Jede Figur weiß, zu welcher Mannschaft sie gehört
> - Umgekehrt kennt auch jede Mannschaft ihre Figuren
> - Stellen Sie im Klassendiagramm dar, dass eine Figur nicht gleichzeitig z. B. Bauer und Läufer sein darf. Eine Figur ist entweder ein Bauer, eine Dame, ein Turm, oder...
> - Jedes Feld weiß, ob es durch eine Figur besetzt ist – und wenn ja mit welcher
> - Jede Figur soll eine Methode *moveTo* bieten, die es erlaubt die Figur auf ein anderes Zielfeld zu bewegen.

Aufgabe 14 - Sachverhalt modellieren

Sie sollen für einen Pizza-Service eine Auftragsverwaltung programmieren. Erstellen Sie hierzu ein UML Klassendiagramm und beschreiben Sie es textuell.

Aufgabe 15 - Diagramm-Interpretation

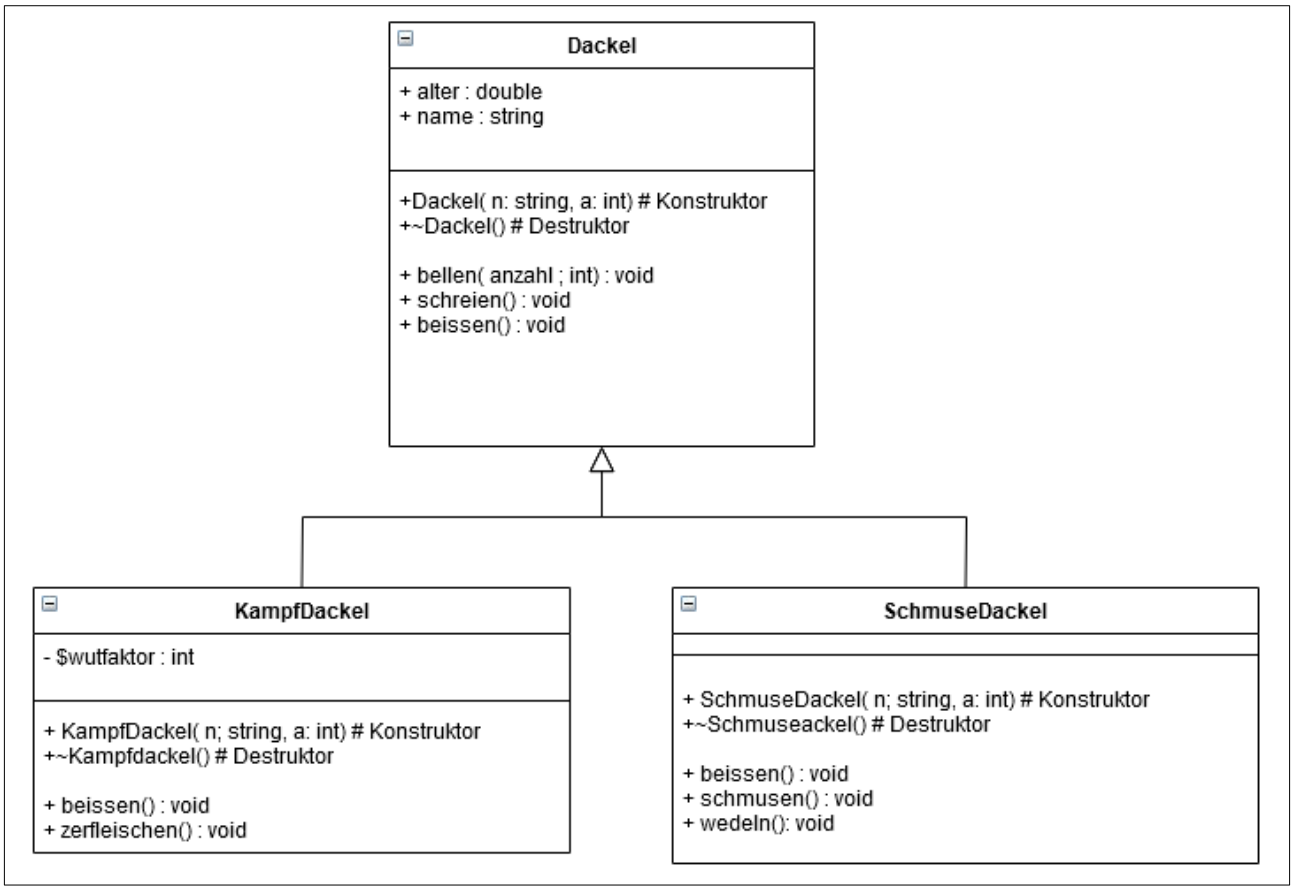
Abb 3: Klassendiagramm "automatische Geldausgabe"



Beantworten Sie die folgenden Fragen:

a) Was kann alles ein ID-Gerät sein?
b) Wie viele UserInterface besitzt ein Geldautomat?
c) Wie bezeichnet man die Beziehung zwischen Tresor und Geldautomat?
d) Gibt es in dem Diagramm eine Vererbungsbeziehung, was ist die Vaterklasse, was ist die Kindklasse?
e) Ist die Beziehung zwischen BankInterface und Geldautomat eine Aggregation?
f) Hat der Kartenleser ein Attribut namens Kartelesen()?
g) Kann ein BankInterface mit mehreren Geldautomaten in Beziehung stehen?

Aufgabe 16 - Klassendiagramm in Programmcode umsetzen



Programmieren Sie die obenstehenden Klassen mit Python. Beachten Sie folgende Anforderungen:

- Soweit nichts anderes gesagt ist, sollen die Methoden sollen jeweils nur einen Kontrolltext ausgeben, der über die aufgerufene Methode und den Namen des Objekts informiert, z.B. „Hasso: Ich belle“.
- Wird in der Klasse Dackel die Methode bellen aufgerufen, so soll der Dackel so oft Wau sagen, wie der Parameter angibt. Wird die Mthode bellen ohne Parameter aufgerufen, so erfolgt nur der Kontrolltext.
- Der Konstruktor soll es erlauben, beim anlegen eines Dackels wahlweise den Namen und das Alter mit anzugeben.
- In der Klasse KampfDackel soll die Überlagerung der Methode beissen verbal etwas kräftiger zubeissen.
- In der Klasse SchmuseDackel soll die Überlagerung der Methode beissen schmusen statt beissen.

Erzeugen Sie einige Objekte und testen Sie Ihre Klassen.

Speichern Sie das Skript unter dem Namen **`aufgabe_ls1_05.py`** ab.

Aufgabe 17 - Batch-Version des Nutzungsdatensystems

Erstellen Sie eine Batch-Version des Nutzungsdatensystems. Diese Version prüft in regelmäßigen Abständen, ob neue Dateien mit Bewegungsdaten in einem Verzeichnis vorliegen. Wenn ja, werden diese importiert. Eine Benutzerinteraktion erfolgt nicht. In einer Log-Datei wird protokolliert, wann Importe vorgenommen wurden. Die importierten Dateien werden anschließend in einem separaten Verzeichnis archiviert.

Speichern Sie das Programm unter dem Namen **`aufgabe_ls1_06`** ab.

Aufgabe 18 - Objekt-orientierter Datenbankzugriff

Für das Nutzungsdatensystem wird noch eine prozedurale Version des Moduls dblib genutzt. Ersetzen Sie diese durch eine objekt-orientierte Version.

Speichern Sie das Programm unter dem Namen **`aufgabe_ls1_07`** ab.