Aufgabe Logistikzentrum

In dieser Aufgabe soll ein Logistikzentrum auf einfache Weise dargestellt werden. In diesem Beispiel vertreibt die Firma, die das Logistikzentrum verwaltet, zwei Produkte: CDs und Autos. Es gibt mehrere Lager und die Akteure sind Kunden und Mitarbeiter. Die Aufgabe zielt darauf ab, Objektorientierung in einem komplexen Kontext einzuüben und ein vereinfachtes Logistikzentrum objektorientiert abzubilden.

Stelle dir zunächst wie in Folie 7 der Powerpoint-Präsentation ein Logistikzentrum vor.
Welche Klassen benötigt man, und welche Methoden sind sinnvoll/ könnten sinnvoll werden?
Wie hängen die einzelnen Klassen zusammen?
Du kannst hierfür auch gerne ein Klassendiagramm erstellen.

Dein Programm soll jetzt mindestens die folgenden Klassen und Methoden enthalten. Du kannst auch gerne dein Programm zusätzlich erweitern, in dieser Aufgabenstellung beschränken wir uns jedoch auf die folgenden Funktionalitäten und Klassen:

1. Die Klassen CD und Auto erben beide von der Klasse Artikel.
   Ein Artikel ist abstrakt und kann nicht instanziiert werden. Alle Klassen, die von Artikel erben, müssen die Methode „getPreis“ implementieren. Diese Methode wird erst in den Subklassen implementiert bzw. näher spezifiziert und gibt den Preis des jeweiligen Artikels zurück.
2. Eine CD besitzt die Attribute: bands, verkäufe, gründung. Jede CD hat außerdem einen Preis von 10 (Wie nennt man so eine Variable?).
3. Ein Artikel besitzt die Attribute: modell, baujahr, verkäufe. Jedes Auto hat außerdem einen Preis von 450.
4. Erstelle die Klasse Artikelliste:
   Dem Konstruktor wird nichts (bis auf „self“ natürlich) übergeben. Es wird eine leere Liste erstellt, die im Folgenden alle Artikelobjekte enthält.
   Weitere Methoden:
   - Mit der Methode „artikel_hinzufügen(self,artikel)“ sollen Artikel in die Artikelliste hinzugefügt werden.
   - Mit der Methode „summe(self)“ sollen die Summe aller Artikelpreise aus der Artikelliste zurückgegeben werden
   - Die Methode „getEintrag(self, index)“ soll den Artikel an der Stelle index der Artikelliste zurückgeben.
5. Erstelle die Klasse Lager. Beim Erstellen eines Lagers (also Implementierung des Konstruktors) wird eine Artikelliste erstellt, dieser Artikelliste werden dann alle Artikel aus den Datensätzen „Musik1“ und „mtcars2“ hinzugefügt und die Artikelliste wird als Attribut „lagerliste“ gespeichert. Beachte hierfür den Hinweis am Ende der gesamten Aufgabe.
   Jedes Lager soll folgende Funktionalitäten erfüllen:
   - Man kann mit der Methode „artikel_verfügbar(self,artikel)“ überprüfen, ob ein Artikel verfügbar ist.
   - Mit der Funktion „artikel_löschen(self,artikel)“ kann man Artikel aus der Liste löschen. Ist der übergebene Artikel nicht im Lager vorhanden, soll eine Fehlermeldung zurückgegeben werden.
   - Man kann überprüfen, wie viele Artikel im Lager vorhanden sind.
   - Man kann sich angeben lassen, wie viel die gesamten Artikel im Lager wert sind *(Hinweis: siehe Klasse Artikelliste)*.

6. Die Klasse Person besitzt die Attribute name, geburtsdatum, land.
7. Erstelle die Klasse Mitarbeiter. Jeder Mitarbeiter besitzt die Attribute name, geburtsdatum, land, größe und erbt von der Klasse Person.
   Die Klasse hat folgende Methoden:
   - Die Methode „bearbeite_auftrag(artikelliste,lager)“ soll alle Artikel aus der übergebenen Artikelliste aus dem Lager löschen.
   - Die Funktion „getName(self)“ gibt den Namen des Mitarbeiters zurück.

8. Erstelle nun die Klasse Kunde. Diese Klasse erweitert die Klasse Person um das Attribut Kontostand. Der Kontostand beträgt für jeden Kunden beim Initialisieren 1000.
   Die Klasse Kunde soll folgende Methoden besitzen:
   - Die Methode „def kaufen(self,*args)“ ist angelehnt an das Universitätsbeispiel und realisiert Polymorphismus:
     a) Werden der Funktion drei Elemente übergeben, wird die Funktion „kaufen1(self,lager,artikelliste,mitarbeiter)“ aufgerufen. Hier bearbeitet ein Mitarbeiter den Auftrag und der Kontostand des Kunden wird um den Kaufbetrag reduziert.
     b) Werden der Funktion vier Elemente übergeben, wird die Funktion „kaufen2(self,lager,artikelliste,mitarbeiter,gutscheincode)“ aufgerufen. Wird als Gutscheincode der String "GutscheinA" aufgerufen, werden dem Kunden 100 (Euro) auf sein Konto zugeschrieben, danach wird der Kauf wie in a) getätigt.
     Wird als Gutscheincode der String "GutscheinB" aufgerufen, wird der gesamte Kaufpreis um 50% reduziert, der Rest ist gleich wie in a).
     Wird ein anderer Gutscheincode eingegeben, soll eine aussagekräftige Fehlermeldung zurückgeben werden.
   - Die Methode „getName(self)“ gibt den Namen des Kunden zurück.

9. Erstelle die Klasse Personenliste:
   Dem Konstruktor wird nichts (bis auf „self“ natürlich) übergeben. Es wird eine leere Liste erstellt, die im Folgenden alle Personen enthält.
   Zusätzliche Methoden:
   - Mit der Methode „person_hinzufügen(self,person)“ sollen Personen in die Personenliste hinzugefügt werden.
   - Die Methode „getEintrag(self, index)“ soll die Person an der Stelle index der Personenliste zurückgeben.

Teste dein Programm auf die folgenden Funktionalitäten in der main-Methode:
- Erstelle ein Lager:
  - Wie viele Artikel sind im neu initialisierten Lager enthalten? (Ergebnis: 68)
  - Was ist der Lager-Gesamtwert? (Ergebnis: 28840)
- Erstelle einen Kunden.
- Füge ein von dir erstelltes Auto und eine CD in eine Einkaufsliste des Kunden hinzu.
- Ändere den Preis des 5. Produktes auf 120 (Euro).
- Füge außerdem das Produkt, das sich an der 10. Stelle des Lagers befindet, in die Einkaufsliste ein.
- Der Kunde möchte nun die Artikel auf der Einkaufsliste bezahlen. Wickeln sie diesen Einkauf ab (Funktion „kaufen (…)“). Teste hierbei die verschiedenen Eingabemöglichkeiten sowie Gutscheincodes.

- Teste erneut folgende Kennzahlen:
    - Wie viele Artikel sind nun im Lager enthalten? (Ergebnis: 67)
    - Was ist der Lager-Gesamtwert? (Ergebnis: 28060)
- Erstelle eine Mitarbeiter- und eine Kundenliste. Lese hierfür die Datei „Personen2“ bzw. „Kunden“ ein.
- Gebe jeweils den Namen des 1. Eintrags der Liste aus. (Ergebnis: Moritz, Kunde 2)
- Füge den Listen jeweils eine Person hinzu.

Hinweis:

Für diese Aufgabe müssen verschiedene Datensätze eingelesen werden. Diese Datensätze finden sich in dem Repository Data.
Beispielsweise die Datei „Musik1“ kann wie folgt eingelesen werden:

```
df = pd.read_csv("C:\\Users\\luis.kaiser\\Documents\\Data\\"+"Musik1"+".csv", delimiter=",",
                 encoding='latin-1')
```

Es bietet sich an, in den Klassen Artikelliste und Personenliste jeweils Methoden zu implementieren, die die gewünschten Daten einliest und als CD’s und Auto’s bzw. Mitarbeiter und Kunden in eine Liste einfügt. Das erleichtert die Implementierung der Klasse Lager und vereinfacht viele Aufrufe in der main-Methode.
Die i-te Zeile und j-te Spalte eines Dataframes (df) kannst Du wie folgt auslesen:
df.iat[i,j]