Lab: 5
Termin: Woche 13

Ein Restaurant benötigt eine Anwendung zur Verwaltung seiner Speisekarte und Bestellungen. Das Restaurantpersonal muss in der Lage sein, eine neue Bestellung zu registrieren, die Informationen über den Kunden (Name und Adresse), die bestellten Speisen, den Gesamtpreis, das voraussichtliche Lieferdatum und die Uhrzeit der Bestellung enthält. Bei der Registrierung einer Bestellung muss das Personal in der Lage sein, die Gerichte aus der Speisekarte auszuwählen und die Kundeninformationen entweder durch Suchen oder durch Anlegung eines neuen Kunden hinzuzufügen. Die Suche nach dem Kunden sollte nach einem Teil des Namens oder der Adresse erfolgen, wobei die Groß- und Kleinschreibung nicht berücksichtigt werden muss. Z.B.: Der Kunde "Max Mustermann" mit der Adresse "Frankfurterstr. 20" kann entweder durch die Suche mit der Zeichenfolge "muster" oder "frankfurt" gefunden werden. Beachten Sie, dass mehrere Kunden denselben Namen oder dieselbe Adresse haben können. Die Speisekarte muss nicht unbedingt eine Suchfunktion haben. Es reicht aus, wenn die Menüpunkte mit einer entsprechenden Nummerierung aufgelistet werden und das Personal aufgefordert wird, die jeweilige Nummer des Menüpunktes zu wählen.
Das Personal muss auch in der Lage sein, die Speisekarte und die Kunden zu verwalten, so dass die Anwendung über CRUD-Operationen für Gerichte und Kunden verfügen muss, d.h.: Hinzufügen, Anzeigen, Aktualisieren und Löschen von bestehenden Gerichten/Kunden.

Die Anwendung soll aus folgenden Komponenten/Packeten bestehen:

- ui
- controller
- repository
- modelle
- tests
- app.py - wird als Einstiegspunkt zum Starten der Anwendung verwendet

Die erwarteten Modelle (Klassen) sind:

- Identifizierbar (nicht zur Instanziierung gedacht)
  - Attribute: id
- Gericht (erbt von Identifiable, soll nicht instanziiert werden)
  - Attribute: Portionsgröße (350g), Preis
- GekochterGericht (erbt Gericht)
  - Zubereitungszeit
- Getränk (erbt Gericht)
  - Attribute: Alkoholgehalt
- Kunde (erbt Identifizierbar)
  - Attribute: Name, Adresse
- Bestellung (erbt Identifizierbar):
  - Attribute: Kunden-ID, Liste der IDs für Gerichte, Liste der IDs für Getränke, Gesamtkosten
  - Methoden:
    - öffentliche Methode zur Berechnung der Kosten - muss mit der Funktion reduce durchgeführt werden.
    - private Methode zur Erzeugung eines Strings, der die Rechnung darstellt. Soll die Standard-Map-Funktion als Teil der Implementierung verwenden
    - öffentliche Methode zur Anzeige des Schecks (sie verwendet die private Methode zur Erzeugung der Rechnung)

Das Repository ist für die Verwaltung der Daten zuständig. Alle Daten, mit denen die Anwendung arbeitet, müssen gespeichert und aus Dateien gelesen werden. Das Repository sollte eines der folgenden Formate unterstützen: eigenes benutzerdefiniertes Format, CSV, JSON, Pickle.
Verwenden Sie die folgende Klassenstruktur, um die Speicherung der verschiedenen Modelle zu implementieren:

Lab: 5
Termin: Woche 13

DataRepo (soll nicht instanziiert werden, es ist als eine abstrakte Klasse gedacht):

- Attribute:
  - datei
- Methoden:
    - save (speichert eine Liste von Objekten in einer Datei)
    - load (liest eine Liste von Objekten aus einer Datei)
    - read file (liest den Inhalt einer Datei und gibt ihn zurück)
    - write to file (schreibt einen String in eine Datei und überschreibt die Datei)
    - convert to string (empfängt als Argument eine Liste von Objekten, die in einen String konvertiert und später in der Datei gespeichert werden müssen. Soll nicht implementiert werden, nur definiert)
    - convert from string (empfängt einen String und konvertiert ihn in eine Liste von Objekten, die zuvor aus einer Datei gelesen wurden. Soll nicht implementiert werden, nur definiert)

Die folgenden Klassen erben von DataRepo und implementieren die Methoden convert from string und convert to string: CookedDishRepo, DrinkRepo, CustomerRepo, OrderRepo.
Die Konvertierungsmethoden müssen die Standard-Map-Funktion als Teil der Implementierung verwenden.

Die folgenden Funktionalitäten müssen getestet werden:

- Hinzufügen eines Gerichts
- Suche nach einem Kunden anhand seines Teilnamens
- Suche nach einem Kunden anhand seiner Teiladresse
- Aktualisieren eines Kundennamens
- Generierung des Strings, der die Rechnung darstellt
- Konvertierung und Speicherung einer Bestellunginstanz in einer Datei
- Einlesen und Konvertieren einer Bestellunginstanz aus einer Datei

Alle Such- und Löschvorgänge müssen mit der Standard filter-Funktion durchgeführt werden.
Alles sollte als Konsolenanwendung implementiert werden.

Lab: 5
Termin: Woche 13

⚠️ Stellen Sie sicher, dass alle Funktionen ein klares Ziel/eine klare Verantwortung, Dokumentation, sinnvolle Namen und Ausnahmebehandlung haben. Stellen Sie außerdem sicher, dass alle Funktionen in Python-Module und Paketen (Packages) aufgeteilt sind.

ℹ️ Falls eine Klasse nicht instanziierbar ist, dass heißt:

```
class A():
    pass

a = A()  # nicht erlaubt
```

ℹ️ Beispiel zum Nutzungs eines Repo:

```
drink_repo = DrinkRepo()
drink1 = Drink('Vodka', 10)
drink2 = Drink('Orangensaft', 0)
drinks = [drink1, drink2]
drink_repo.save(drinks)  # Getränke Objekte speichern
drinks = drink_repo.load()  # Getränke Objekte laden
```