Zusatzaufgaben: Objekte, Klassen

Jede dieser Aufgaben ist soviel Wert wie ein ganzes Hausaufgabenblatt. Die Aufgaben könnt ihr im Laufe des Semesters lösen, indem ihr eine entsprechende Datenstruktur baut. Die Punkte können als Zusatzpunkte fehlende Punkte bei den drei Hausaufgabenblättern ausgleichen.

1. **Eine Klasse für Daten**

   Definieren Sie eine Klasse `Data`, die als Datencontainer dienen soll. Sie soll eine Methode `titles()` haben, die eine Liste der Bezeichnungen der bisher gespeicherten Daten enthält, sowie eine Methode `set_data`, die einen Namen und ein Datenobjekt übergeben bekommt und dieses zu dem Namen in einem Wörterbuch speichert, sowie eine Methode `get_data`, die einen Namen übergeben bekommt und das Datenobjekt zurückgibt. Weiterhin soll es noch eine Methode `set_properties` und eine Methode `get_properties` geben, die zu einem Namen ein Dictionary von Eigenschaften speichert. Außerdem soll die Klasse Methoden `pickle` und `unpickle` haben, die einen Dateinamen übergeben bekommen und das Objekt mit Hilfe des Moduls `pickle` speichern bzw. lesen.

   Wenn Sie eine besondere Art von Daten speichern wollen, können Sie von dieser Klasse eine andere ableiten, z.B. `EEGData`, und dort noch spezielle Methoden zum Lesen von Daten in gewissen Formaten und zum Auswerten, Anzeigen, etc. hinzufügen. Dabei ist es wichtig, sich die Eigenschaften der entsprechenden Daten zu überlegen (z.B. bei einer Tabelle die Namen der Spalten oder der Zeilen, Einheiten, was weiß ich), um diese Eigenschaften in den Auswertungsfunktionen benutzen zu können.

2. **Eine Klasse für Kompositionen**

   Definieren sie ein Klasse `Komposition`, die eine Komposition repräsentiert. Die Aufgabe ist offen gehalten. Die Klasse kann als Attribute etwa `akkordschema` und `stimmen` enthalten. Diese Attributen wären dann Listen von Objekten der Klasse `Akkord`, bzw. `Stimme`. Wie Sie weitermachen, können Sie überlegen. Stimmen wiederum bestünden aus einer Abfolge von Tönen. Die Klasse `Ton` sollte mindestens die Attribute `dauer, notenwert, lautstaerke` haben. Diese Objekte können auch Abspielmethoden haben, die sie hörbar machen.

3. **Eine Klasse für Himmelskörper**

   Definieren Sie eine Klasse `Body`, die als Attribute die Masse, den Radius, den Namen des Himmelskörpers enthält. Der Konstruktor sollte diese Parameter übergeben bekommen. Weitere mögliche Attribute: Bahnparameter, um die Keplerschen Bahnen

zu berechnen, oder die Position und Geschwindigkeit zu einem gewissen Zeitpunkt, was ebenfalls die Berechnung der Bahnen erlaubt.

Definieren Sie eine Klasse `Viewer`, die in ihrem Konstruktor eine Liste von Himmelskörpern bekommt und Methoden `visualize` und `update`, die diese Himmelskörper 3d visualisieren, bzw. die Positionen aktualisieren. Da in galaktischen Maßstäben die Himmelskörper oft furchtbar klein sind, sollte die Methode `visualize` einen optionalen Paramtere `scale` haben, mit dem man die Himmelskörper vergrößern kann. Am besten eignet sich dafür das Moduld `vpython` (es heißt auch 'visual python' oder 'visual'.)

4. **Game of Life**

   Programmieren Sie das Game of Life. Dafür ist es angemessen, eine Klasse `Welt` zu definieren, die den Zustand der Welt in einem Attribut speichert und eine Methode `schritt` hat, die einen Entwicklungssschritt beschreibt. Weiterhin kann die Klasse eine Methode `anzeigen` haben, die mit Hilfe von matplotlib den Zustand anzeigt.