1. NVS-Test: Synchrone Programmierung

Schreibe ein *NodeJS* Programm, das basierend auf der Input-Datei `database.json` die gestellten Aufgaben implementiert.

Gehe dabei wie in unserer gemeinsam entwickelten Application mit den Notizzetteln bzw. der Übungsaufgabe mit den Personen vor.

Konkret bedeutet das:

- Verwende für deine Lösung 2 Dateien

  app.js der Einsprungspunkt in dein Programm; kümmert sich um die Parameterübergabe, etc.

  worker.js stellt die Business-Logic deines Programms zur Verfügung

- Installiere dir folgende 3rd party Module
  - *yargs*
  - *lodash*
  - *moment*
- Realisiere mit diesem Setup die in *"Aufgabe"* gestellten Aufgaben

Aufgabe

Dein Programm soll folgende Kommandos und die damit verbundenen Parameter verarbeiten können:

- `node app.js print-sorted --input="database.json"`
- `node app.js find-all --input="database.json" --firstname="Wolf"`
- `node app.js pick-adult --input="database.json"`

Beschreibung

Das Kommando `print-sorted` soll mittels der Funktion `getSortedByName` (`worker.js`) alle Personen nach ihrem Nachnamen aufsteigend sortieren und sie mittels der Funktion `printPerson` (`worker.js`) ausgeben.

Das Kommando `find-all` soll mittels der Funktion `findAllWithFirstname` (`worker.js`) alle Personen finden, deren Vornamen dem gegebenen Parameter *firstname* entsprechen oder ihn beinhalten und sie mittels der Funktion `printPerson` oder `printAgeOf` (`worker.js`) ausgeben (nimm die Funktion, die dir lieber ist).

Das Kommando `pick-adult` soll mittels der Funktion `pickRandomAdult` (`worker.js`) eine zufällige Person aus der übergebenen Datei auswählen, die $\geq$ 18 Jahre alt ist und diese mittels der Funktion `printAgeOf` (`worker.js`) ausgeben.

Modul: app.js

Dieses Modul soll das Programm steuern, d.h.

- das Kommandos extrahieren
- die entsprechende Parameter extrahieren
- die jeweilige Funktion von `worker.js` aufrufen
- ggf. Rückgabewerte von `worker.js` weiterverarbeiten/ausgeben

Das Modul soll selbst **keine Operationen** auf Files oder Objekten durchführen. Es soll ausschließlich der Steuerung dienen so, wie wir es in unseren Übungsbeispielen gemacht haben).

Modul: worker.js

Dieses Modul soll die tatsächliche Arbeit verrichten.

Implementiere (und exportiere) dazu die folgenden Funktionen:

getSortedByName(input):

Verwende die Funktion `orderBy` aus dem Modul `lodash`, um alle Personen deiner Datenbank nach ihrem Nachnamen aufsteigend zu sortieren.

- Parameter:

  input der Pfad zu der zu verwendenden Input-Datei

- Rückgabewert:

  Success: *sortiertes Array*

  Failure: `undefined`

printPerson(person):

`printPerson` bekommt ein Objekt der Form

```
{ firstname: "Wolf", lastname: "REITSAMER", dateOfBirth: "18.04.2004", ... }
```

und gibt die Person folgendermaßen aus:

```
REITSAMER, Wolf, 2004-04-18
```

Die Funktion liefert nichts zurück.

Tipp: suche in der Dokumentation von `moment` unter *"String + Format"* nach einem Weg das von uns verwendete Format zu parsen

findAllWithFirstname(input, firstname):

Die Funktion soll alle Personen finden und zurückliefern, deren Vornamen dem gegebenen Vornamen (`firstname`) entsprechen oder diesen beinhalten.

- Parameter:

  input der Pfad zu der zu verwendenden Input-Datei

  firstname der im Vornamen der Person zu suchende String bzw. Teilstring

- Rückgabewert:

  Success: `[{ Person 1 }, { Person 2 }, ... ]`

  Failure: `undefined`

Tipp: verwende die Methode `.includes` der String-Klasse, um zu überprüfen, ob ein String einen anderen inkludiert.

Achtung: Groß- und Kleinschreibung sollen keine Rolle spielen!

printAgeOf(person):

Die Funktion bekommt ein Object der Form

```
{ firstname: "Wolf", lastname: "REITSAMER", dateOfBirth: "18.04.2004", ... }
```

und gibt die Person folgendermaßen aus:

```
REITSAMER, Wolf, 15 years
```

Tipp: verwende die Funktion `.diff` des Moduls *moment*, um die Differenz des Geburtsdatums der Person zum jetzigen Zeitpunkt in Jahren zu berechnen.

pickRandomAdult(input):

Wählt mit Hilfe der *lodash* Funktion `shuffle` eine zufällige Person, die älter (oder gleich) 18 Jahre alt ist aus der Datenbank aus und liefert diese zurück.

- Parameter:

  input der Pfad zu der zu verwendenden Input-Datei

- Rückgabewert:

  Success: {*die zufällig gewählte Person*}

  Failure: `undefined`

Allgemeines

Lies in der Dokumentation der verwendeten Module nach wie die hier zu verwendenden Funktionen funktionieren. Alle Funktionen haben gute Beispiele in der Dokumentation!

Implementiere in `worker.js` die oben aufgelisteten Funktionen.

Jede Funktion, die den Pfad zu einer Datei erhält, soll gleich zu Beginn überprüfen, ob die Datei existiert. Existiert die Datei nicht, soll die Funktion sofort abbrechen und unter *Failure* definierte Object zurückliefern (siehe *Rückgabewert* der Funktionen).

– Gutes Gelingen –