SPENGERGASSE

2AHWII-Y PLF2

10.12.2024

Themen: Programmieren, Datenbanken

Datenbank: Personen importieren (12p)

In der Angabe befindet sich eine `persons.csv` Datei. Importiere diese - wie im Unterricht - in eine sqlite3 Datenbank `persons.db`!

Die Felder heißen:

- Vorname
- Nachname
- Geschlecht
- Gewicht
- Groesse

Beachte, dass Gewicht und Groesse ein Zahlen-Datentyp sind und achte auf die im CSV verwendeten Trennzeichen.

Committe und pushe Deine Änderungen (das erzeugte Datenbankfile `persons.db`) jetzt.

Programmieren: FizzBuzz (18p)

Editiere dazu die noch leere Datei "fizzbuzz.ts". Ausführung mit `deno fizzbuzz.ts`.

Schreibe ein Programm, das die Zahlen von 1 bis 100 ausgibt. Für Vielfache von drei gib "Fizz" anstelle der Zahl aus. Für Vielfache von fünf gib "Buzz" anstelle der Zahl aus. Für Zahlen, die Vielfache sowohl von drei als auch von fünf sind, gib "FizzBuzz" anstelle der Zahl aus.

git add fizzbuzz.ts - commit - push

Programmieren: CSV einlesen und auf console ausgeben (27p)

Gib nun die einzelnen Zeilen (Records) der `persons.csv` Datei anders formatiert aus. Editiere dazu die Datei `persons.ts`.

Folge exakt den folgenden Beispielen:

```
Kathrin Bussmann ist weiblich. Sie ist 66kg schwer und 175cm groß.
Simeon Schwirkschlies ist männlich. Er ist 75kg schwer und 188cm groß.
```

add persons.ts - commit - push

Programmieren: `bmi()` sowie Testmethode (23p)

Führe nun im Angabeordner `deno init` aus und stelle fest, welche Dateien dazugekommen sind.

Für diese Aufgabe wirst Du die Datei `main_test.ts` sowie `main.ts` editieren.

`main.ts`: `bmi(kg: number, m: number) : number)`

Benenne die Funktion "add" in beiden Dateien auf "bmi" um. Sie soll aus den beiden Paratern Gewicht und Größe den BMI errechnen und zurückgeben. Der BMI (Body-Mass-Index) wird folgendermaßen errechnet: $kg/(m^2)$, also Körpergewicht (in kg) dividiert durch das Quadrat der Körpergröße (in m). Dieser Wert gilt im Gesundheitswesen als Einheit für die Feststellung von Unter- Normal- oder Übergewicht.

`main_test.ts`

Dies wird aufgerufen, wenn `deno test` aufgerufen wird. Ändere dazu die bereits vorhandene Testmethode, um die Funktion `bmi()` zu testen:

Folgende BMI Werte sind (mit Rundungsfehler) korrekt:

```
["kg", "m", "bmi"],
[42, 1.68, 14.9],
[56, 1.63, 21.1],
[127, 2.2, 26.2],
[105, 1.65, 38.6],
[131, 1.83, 39.1],
```

Die zu verwendende Funktion bei den Tests heißt `assertAlmostEquals()` und wird so aufgerufen:

```
assertAlmostEquals(actual, expected, tolerance);
```

Der tolerance Wert ist `0.1` -- die Testwerte sind auf eine Nachkommastelle gerundet.

add - Commit - Push

Erreichbare Punkte: 80 (`0-40 / 41-50 / 51-60 / 61-70 / 71-80`)