V13 Fahrzeug

Json

JSON (JavaScript Object Notation) ist ein leichtgewichtiges Datenformat, das zur Darstellung strukturierter Daten verwendet wird. Es ist leicht lesbar und schreibbar für Menschen und einfach zu parsen und zu generieren für Maschinen, und wird häufig in Webanwendungen zur **Übertragung von Daten** zwischen einem Server und einem Client eingesetzt.

ToJson-Methode

Die ToJson-Methode ist eine normale Methode, die in Datenklassen verwendet wird, um Objekte in gültige JSON-Strings zu konvertieren. Diese JSON-Strings können dann über das Internet übertragen oder in Datenbanken gespeichert werden.

ToJson Bau

Um einen gültigen JSON-String zu erstellen, muss man zunächst verstehen, was ein JSON-String ist. Es handelt sich entweder um eine Zeichenkette, die eine dynamische Map repräsentiert, oder um eine Liste von dynamischen Maps. Die Umwandlung eines Objekts in JSON besteht aus zwei Schritten:

1. **Umwandlung des Objekts in eine Map:** Um ein Objekt in eine Map zu konvertieren, erstellt man eine Map mit dem Typ `Map<String, dynamic>`. Die Schlüssel (Keys) dieser Map sind die Namen der Attribute, und die Werte (Values) sind die entsprechenden Attributwerte. Diese Logik wird in der Methode `toMap` umgesetzt.

```
  Map<String, dynamic> toMap() {
 return <String, dynamic>{
   'insasseZahl': insasseZahl,
   // in Datenbank ist baujahr als int gespeichert
   // fromMillisecondsSinceEpoch diese rechnet den Unterschied in
Millisekunden zwischen dem 01.01.1970 und dem Datum
   // Das Ergebnis ist der int Wert, der in der Datenbank gespeichert ist.
   'baujahr': baujahr.millisecondsSinceEpoch,
   'reifen': reifen,
   'marke': marke,
  };
 }
```

2. **Umwandlung dieser Map in einen JSON-String:** Nachdem die Map erstellt wurde, kann man sie mit der eingebauten Methode `json.encode(map)` in einen JSON-String umwandeln. Diese Methode wandelt eine `Map<String, dynamic>` in eine gültige JSON-String um.

```
// Die String, dynamische Mappe wird als JSON-codiertes Objekt zurückgegeben.
String toJson() => json.encode(toMap());
```

```
    // so Sieht das JSON-Objekt aus
'''
    {
     "reifenZahl":40,
     "insasseZahl":5,
     "baujahr":1577833200000,
     "reifenRadius":30.0,
     "reifenBreite":20.0,
     "marke":"Auto1"
     }
  '''

    // so sieht die JSON-Liste aus
'''
  [
    {
    "reifenZahl": 4,
    "insasseZahl": 5,
    "baujahr": 1577833200000,
    "reifenRadius": 30.0,
    "reifenBreite": 20.0,
    "marke": "Auto1"
    },
    {
    "reifenZahl": 4,
    "insasseZahl": 5,
    "baujahr": 1577833200000,
    "reifenRadius": 30.0,
    "reifenBreite": 20.0,
    "marke": "Auto2"
    },
    {
    "reifenZahl": 4,
    "insasseZahl": 5,
    "baujahr": 1577833200000,
    "reifenRadius": 30.0,
    "reifenBreite": 20.0,
    "marke": "Auto3"
    }
  ]
'''
```

Um die Komplexität der Methode `toJson` zu verringern, ist es für Anfänger sinnvoll, den Prozess auf zwei Methoden zu verteilen: `toMap` und `toJson`.

FromJson Methode

Die `FromJson` Methode ist eine spezielle Factory-Methode in Datenklassen. Sie erhält einen JSON-String als Parameter und erstellt basierend auf diesem String das entsprechende Objekt der Klasse.

Diese Methode nimmt die `<String, dynamic>` Map, die durch das Parsen des JSON-Strings erzeugt wurde, und weist die Werte den entsprechenden Attributen der Klasse zu. Somit wird ein Objekt erstellt, das die im JSON enthaltenen Daten widerspiegelt.

Aufteilung der FromJson Methode in zwei Schritte

Um den Aufbau der **FromJson** Methode einfacher zu gestalten, teilen wir sie in zwei Teile auf:

1. **fromJson**

Die **fromJson** Methode nimmt den JSON-String (wie er z.B. aus einer Datenbank kommt) und decodiert ihn in eine Mappe vom Typ `<String, dynamic>`. Das bedeutet, sie wandelt den JSON-String in eine Key-Value-Struktur um, bei der die Keys die Namen der Attribute der Klasse darstellen und die Values die entsprechenden Werte sind.

Beispiel:

```
  factory V13Auto.fromJson(String source) {
    // Die Methode fromMap wird aufgerufen, um ein V13Auto-Objekt zu erzeugen
    // Die Methode json.decode() wandelt den JSON-String in eine Map<String,
dynamic> um,
    // somit kann die Methode fromMap() es handeln
    return V13Auto.fromMap(json.decode(source) as Map<String, dynamic>);
  }
```

2.**fromMap**

Die fromMap Methode nimmt die Mappe <String, dynamic>, die von der fromJson Methode bereitgestellt wurde. Da wir bereits wissen, dass diese Mappe den Attributen unserer Klasse entspricht, können wir die einzelnen Werte direkt den Attributen unserer Klasse zuweisen und darauf basierend das entsprechende Objekt erstellen.

```
  factory V13Auto.fromMap(Map<String, dynamic> map) {
    // Die Erzeugung des Objekts erfolgt mit den Werten aus der Map
    return V13Auto(
      // Der Wert von 'reifenZahl' wird aus der Map gelesen und in das Attribut
'reifenZahl' geschrieben
      // weil die Mappe dynamische Values hat, muss die Value mit 'as' gecastet
werden
      reifenZahl: map['reifenZahl'] as int,
      insasseZahl: map['insasseZahl'] as int,
      // DateTime.fromMillisecondsSinceEpoch() erzeugt ein DateTime-Objekt aus
Millisekunden
      // weile die DateTime-Objekte in der Mappe als int gespeichert sind, wird
der Wert mit 'as' gecastet
      baujahr: DateTime.fromMillisecondsSinceEpoch(map['baujahr'] as int),
      reifenRadius: map['reifenRadius'] as double,
      reifenBreite: map['reifenBreite'] as double,
      // marke ist ein optionales Attribut, deshalb wird es mit einem ternären
```

```
  Operator gelesen
      marke: map['marke'] != null ? map['marke'] as String : null,
    );
  }
```

Hinweis: Durch diese Aufteilung wird der Code klarer und besser strukturiert. Die fromJson Methode kümmert sich nur um die Dekodierung des JSON-Strings, während die fromMap Methode für das eigentliche Erstellen des Objekts verantwortlich ist.

JSON-List-Mapping zu einer Liste von Objekten

Um eine JSON-String-Repräsentation einer Liste von Mappen in eine Liste von Objekten umzuwandeln, gehen wir wie folgt vor:

Schritte

1. **Leere Liste erstellen:** Zunächst erstellen wir eine leere Liste vom Typ des zu erzeugenden Objekts.
2. **JSON dekodieren:** Mit der Methode `jsonDecode()` können wir den JSON-String in eine Liste von Mappen (`List<Map<String, dynamic>>`) umwandeln.
3. **Durch die Liste iterieren:** Für jede Map in der Liste rufen wir die `fromJson()`-Methode der jeweiligen Klasse auf, um das Objekt zu erstellen, und fügen es der leeren Liste hinzu.
4. **Endergebnis:** Am Ende haben wir eine Liste der gewünschten Objekte, die aus dem JSON-String generiert wurden. **Beispiel**

```
  final List<V13Auto> autos = [];
  // JSON String in Map-Liste umwandeln
  final jsonList = jsonDecode(v13AutoJsonFile);
  print(jsonList);
  // Map-Liste durchgehen
  for (var element in jsonList) {
  // Map in Objekt umwandeln und in die Liste einfügen
    autos.add(V13Auto.fromMap(element));
    // die eingefügten Objekte printen
    print(autos.last.toString());
  }
```

Aufgabe

Erstellen Sie eine Klasse mit **einfachen** Attributen wie String, bool, int und double. Implementieren Sie darin die Methoden `toMap`, `toJson`, `fromMap` und `fromJson`. Testen Sie anschließend Ihre Klasse.