1. Gegeben ist folgende MariaDB-Tabelle:

```
CREATE TABLE `weather` (
  `measured` datetime NOT NULL,
  `temperature` decimal(4,1) DEFAULT NULL,
  `humidity` tinyint(4) DEFAULT NULL,
  `pressure` decimal(5,1) DEFAULT NULL,
  PRIMARY KEY (`measured`)
);
```

In dieser stehen Wetterdaten zu verschiedenen Zeitpunkten. Mit folgenden INSERT-Befehl können Daten eingefügt werden:

```
INSERT INTO `weather` (`measured`, `temperature`, `humidity`, `pressure`) VALUES
('2022-07-24 14:36:31', '33.9', 34, '998.0'),
('2022-07-24 10:33:21', '28.6', 42, '998.1'),
('2022-07-24 00:48:15', '23.1', 52, '998.5');
```

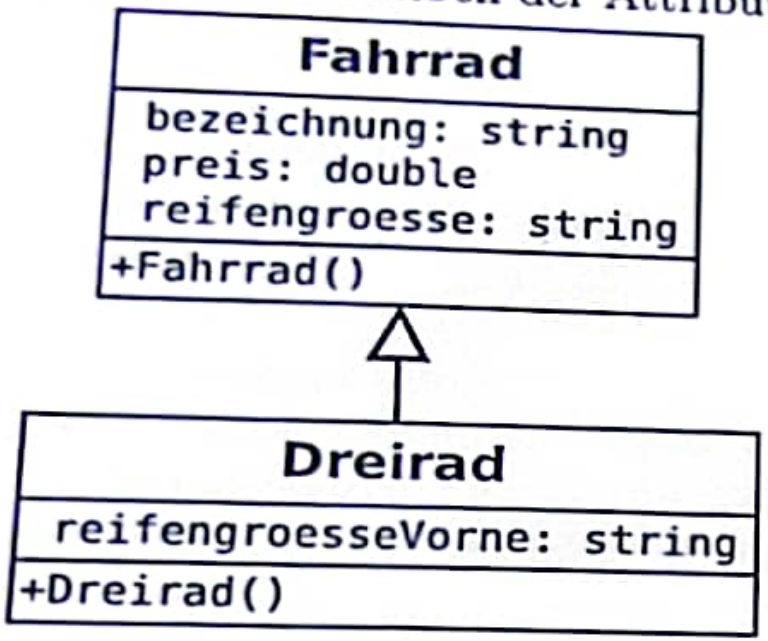
Entwickeln Sie einen Node.js Service, der nach GET-Anfrage der URL /data den jüngsten Eintrag aus der Tabelle in JSON-Notation zurück liefert. (15 Punkte)

2. Gegeben ist folgende HTML-Seite, die nicht modifiziert werden darf:

```
<!DOCTYPE html>
<html lang="de">
<head>
  <meta charset="utf-8" />
  <title>Weather</title>
</head>
<body>
  <h1>Weather</h1>
  <div>Time <span id="time"></span>\\
    Temperature <span id="temperature"></span>\\
    Humidity <span id="humidity"></span>\\
    Pressure <span id="pressure"></span>
  </div>
<script defer src="jquery.js"></script>
<script defer src="weather.js"></script>
</body>
</html>
```

Mit *jQuery* ist die Funktionalität für die Datei `weather.js` zu programmieren. Diese Web-Seite soll alle 5 Minuten den in der vorigen Aufgabe entwickelten Service per GET anfragen. Die Wetterdaten sind darauf in den entsprechenden -Tags darzustellen. *(15 Punkte)*

3. Implementieren Sie die aufgeführte Klassenstruktur für einen Fahrradhändler mittels JavaScript in Klassensyntax. Die Attribute sind mit geeigneten Standardwerten zu initialisieren. Für das Lesen und Schreiben der Attribute implementieren Sie getter uns setter. *(13 Punkte)*



4. Erläutern Sie stichpunktartig, was unter „Insecure Direct Object References“ verstanden wird. Geben Sie ein **eigenes Beispiel** dazu an. Nennen Sie eine Maßnahme zur Abwehr und erläutern Sie diese anhand Ihres Beispiels. *(6 Punkte)*

5. *Erläutern Sie stichpunktartig, was* **Parameter-Injection** in Abgrenzung zu **SQL-Injection** *und* **Cross-Site-Scripting** ist. Geben Sie ein konkretes Quelltext-Beispiel an, wie Parameter-*Injection verhindert* werden kann. *(5 Punkte)*

6. Skizzieren Sie die Grundarchitektur für ein einfaches Web-Angebot nach Empfehlung des BSI-Standards zur Internet-Sicherheit. *(5 Punkte)*

7. Die Funktion `factorial()` ist wie folgt definiert:

```
function factorial(n) {
    let val = 1;
    for(let i = 1; i <= n; i++) {
        val *= i;
    }
    return val;
}
```

Ihre Aufgabe ist es **White-Box-Tests mit QUnit** zu schreiben.

7. a) Welches Verfahren wenden Sie zum Finden von Testfällen für die Funktion `factorial(n)` an? Beschreiben Sie einen Test nach dem genannten Verfahren. *(3 Punkte)*

7. b) Implementieren Sie den zuvor beschriebenen Testfall für die Funktion `factorial(n)`. Dabei ist nur der JavaScript-Teil anzugeben. *(4 Punkte)*