JS Advanced Exam – 13 March 2022

Problem 2. Car Dealership

```
class CarDealership {

}
```

Write a class **`CarDealership`**, which implements the following functionality:

Functionality

Constructor

Should have these **4** properties:

- **`name`** – String
- **`availableCars`** – Array
- **`soldCars`** – Array
- **`totalIncome`** – **default: 0**

At the initialization of the `CarDealership` class, the **constructor** accepts the **`name`.** The **`totalIncome`** has a **default value of 0!** The rest of the properties must be **`empty`!**

Hint: You can add more properties to help you finish the task.

`addCar (model, horsepower, price, mileage)` - This method should **add a new car** to the dealership. The method accepts **4 arguments:**

- If any of the following requirements is **NOT fulfilled**, an **error** with the following message should be **thrown**: **`"Invalid input!"`**
  - **`Model`** – non-empty string;
  - **`Horsepower`** – positive integer number;
  - **`Price`** – positive number;
  - **`Mileage`** – positive number.

Hint: Zero is also a positive number.

- Otherwise, you should **add the car**, with properties: **`{model, horsepower, price, mileage}`** to the **`availableCars`** array and **return**:

"New car added: {model} - {horsepower} HP - {mileage} km - {price}$"

- When **returning** the result, the **Mileage and Price** must be **rounded to the second decimal point!**

sellCar (model, desiredMileage) – This method should **search for a car** with the given **model** in the **availableCars** array, and then **sell** it. Accepts **2 arguments**.

- If a car with the given **model** cannot be found, an error with the following message should be **thrown**:

"{model} was not found!"

- If you **find the car with the given model**, you should look up its **mileage**. The person who wants to buy it has a simple request. He is looking for a car with a **mileage** that is **less or equal** to his **desired mileage**. To ensure the sale of the car you must make a bargain:
  - If the **found** car's mileage is **less than or equal to** the **desiredMileage** – the price stays the same!
  - If the **difference** between the **car's mileage** and the **desiredMileage** is less or equal to **40.000 km** – the price gets **deducted by 5%**!
  - If the **difference** between the **car's mileage** and the **desiredMileage** is more than **40.000 km** – the price gets **deducted by 10%**!
- You should **remove** the car from the **availableCars** array and **add** it to the **soldCars** array in the following format: **{model, horsepower, soldPrice}**
- Finally, you must add the **soldPrice** to the **totalIncome** and return:

"{model} was sold for {soldPrice}$"

Note: soldPrice must be **rounded** to the second decimal point!

currentCar () - This method should just return all available cars separated by a new line in format:

```
"-Available cars:
---{model} - {horsepower} HP - {mileage} km - {price}$
---{model} - {horsepower} HP - {mileage} km - {price}$"
```

Note: mileage and **price** must be **rounded** to the second decimal point!

- If there are **no available** cars, just return:

"There are no available cars"

`salesReport (criteria)` – This method accepts 1 argument. It should **sort** the sold cars, **based on a given criterion**. The two possible criteria are – **"horsepower"** or **"model"**

- If the given criteria **do not match** either of the possible criteria, an **error** with the following message should be **thrown**:

  `"Invalid criteria!"`

- If the given criteria is **"horsepower"** – the sold **cars** must be **sorted** by their **horsepower** in **descending order**;
- If the given criteria is **"model"** – the sold cars must be **sorted alphabetically** by their **model**;
- Finally, **return all sorted** sold cars **separated** by **a new line** in format:

```
"-{dealershipName} has a total income of {totalIncome}$
-{soldCarsCount} cars sold:
---{model} - {horsepower} HP - {price}$
---{model} - {horsepower} HP - {price}$"
…
```

Note: `totalIncome` and `price` must be rounded to the second decimal point!

Example

Input 1

`let dealership = new CarDealership('SoftAuto');`
`console.log(dealership.addCar('Toyota Corolla', 100, 3500,`
`190000));`
`console.log(dealership.addCar('Mercedes C63', 300, 29000,`
`187000));`
`console.log(dealership.addCar('', 120, 4900, 240000));`

Output 1

`New car added: Toyota Corolla - 100 HP - 190000.00 km - 3500.00$`
`New car added: Mercedes C63 - 300 HP - 187000.00 km - 29000.00$`
`Uncaught Error Error: Invalid input!`

Input 2
`let dealership = new CarDealership('SoftAuto');`

```
dealership.addCar('Toyota Corolla', 100, 3500, 190000);

dealership.addCar('Mercedes C63', 300, 29000, 187000);

dealership.addCar('Audi A3', 120, 4900, 240000);

console.log(dealership.sellCar('Toyota Corolla', 230000));

console.log(dealership.sellCar('Mercedes C63', 110000));
```

Output 2

```
Toyota Corolla was sold for 3500.00$

Mercedes C63 was sold for 26100.00$
```

Input 3

```
let dealership = new CarDealership('SoftAuto');

dealership.addCar('Toyota Corolla', 100, 3500, 190000);

dealership.addCar('Mercedes C63', 300, 29000, 187000);

dealership.addCar('Audi A3', 120, 4900, 240000);

console.log(dealership.currentCar());
```

Output 3

```
-Available cars:

---Toyota Corolla - 100 HP - 190000.00 km - 3500.00$

---Mercedes C63 - 300 HP - 187000.00 km - 29000.00$

---Audi A3 - 120 HP - 240000.00 km - 4900.00$
```

Input 4

```
let dealership = new CarDealership('SoftAuto');

dealership.addCar('Toyota Corolla', 100, 3500, 190000);

dealership.addCar('Mercedes C63', 300, 29000, 187000);

dealership.addCar('Audi A3', 120, 4900, 240000);
```

```
dealership.sellCar('Toyota Corolla', 230000);

dealership.sellCar('Mercedes C63', 110000);

console.log(dealership.salesReport('horsepower'));
```

Output 4
-SoftAuto has a total income of 29600.00$ -2 cars sold: ---Mercedes C63 - 300 HP - 26100.00$ ---Toyota Corolla - 100 HP - 3500.00$