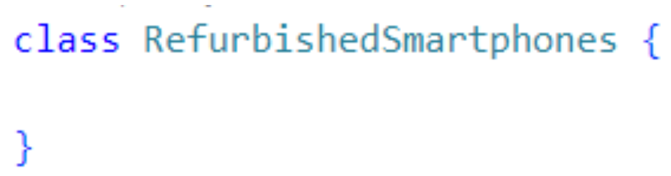
JS Advanced Exam – Retake

Problem 2. Refurbished Smartphones



```
class RefurbishedSmartphones {

}
```

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

Functionality

Constructor

Should have these **4** properties:

- **`retailer`** **–** String
- **`availableSmartphones`** **–** Array
- **`soldSmartphones`** **–** Array
- **`revenue`** **– default: 0**

At the initialization of the `RefurbishedSmartphones` class, the **constructor** accepts the **`retailer`.** The **`revenue`** 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.

`addSmartphone (model, storage, price, condition) -` This method should **add a new smartphone** to the retailer. The method accepts **4 arguments:**

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

Hint: Zero is also a positive number.

- Otherwise, you should **add the smartphone**, with properties: **`{model, storage, price, condition}`** to the **`availableSmartphones`** array and **return**:

  `"New smartphone added: {model} / {storage} GB / {condition} condition - {price}$"`
- When **returning** the result, the **Price** must be **rounded to the second decimal point!**

`sellSmartphone (model, desiredStorage) –` This method should **search for a smartphone** with the given **model** in the **`availableSmartphones`** array, and then **sell** it. Accepts **2 arguments**.

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

  `"{model} was not found!"`
- If you **find the smartphone with the given model**, you should look up its **storage**. The person who wants to buy it has a simple request. He is looking for a smartphone with a **storage** that is **more or equal** to his **desired storage**. To ensure the sale of the smartphone you must make a bargain:
  - If the **found** smartphone's storage is **more than or equal to** the **`desiredStorage`** – the price stays the same!
  - If the **difference** between the **smartphone's storage** and the **`desiredStorage`** is less or equal to **128 GB** – the price gets **deducted by 10%**!
  - If the **difference** between the **smartphone's storage** and the **`desiredStorage`** is more than **128 GB** – the price gets **deducted by 20%**!
- You should **remove** the smartphone from the **`availableSmartphones`** array and **add** it to the **`soldSmartphones`** array in the following format: **`{model, storage, soldPrice}`**
- Finally, you must add the **`soldPrice`** to the **`revenue`** and return:

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

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

upgradePhones () - This method should find the **storage** for every available smartphone and **double** it, then **return** them separated by a new line in format:

```
" Upgraded Smartphones:
{model} / {storage} GB / {condition} condition / {price}$
{model} / {storage} GB / {condition} condition / {price}$"
```

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

Note: storage must be **updated** to **`availableSmartphones`** array!

- If there are **no available** smartphones, **throw**:

```
"There are no available smartphones!"
```

salesJournal (criteria) – This method accepts 1 argument. It should **sort** the sold smartphones, **based on a given criteria**. The two possible criteria are – **"storage"** 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 **"storage"** – the sold **smartphones** must be **sorted** by their **storage** in **descending order**;
- If the given criteria is **"model"** – the sold smartphones must be **sorted alphabetically** by their **model**;
- Finally, **return all sorted** sold smartphones **separated** by **a new line** in format:

```
"{ RetailerName} has a total income of { revenue }$
{soldSmartphonesCount} smartphones sold:
{model} / {storage} GB / {price}$
{model} / {storage} GB / {price}$"
…
```

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

Example

Input 1

`let retailer = new RefurbishedSmartphones('SecondLife Devices');`
`console.log(retailer.addSmartphone('Samsung S20 Ultra', 256, 1000, 'good'));`
`console.log(retailer.addSmartphone('Iphone 12 mini', 128, 800, 'perfect'));`
`console.log(retailer.addSmartphone('', 512, 1900, 'good'));`

Output 1

```
New smartphone added: Samsung S20 Ultra / 256 GB / good condition -
1000.00$
New smartphone added: Iphone 12 mini / 128 GB / perfect condition -
800.00$
Uncaught Error Error: Invalid smartphone!
```

Input 2

```
let retailer = new RefurbishedSmartphones('SecondLife Devices');
retailer.addSmartphone('Samsung S20 Ultra', 256, 1000, 'good');
retailer.addSmartphone('Iphone 12 mini', 128, 800, 'perfect');
retailer.addSmartphone('Xiaomi Redmi Note 10 Pro', 128, 330, 'perfect');
console.log(retailer.sellSmartphone('Samsung S20 Ultra', 256));
console.log(retailer.sellSmartphone('Xiaomi Redmi Note 10 Pro', 256));
console.log(retailer.sellSmartphone('Samsung Galaxy A13', 64));
```

Output 2

```
Samsung S20 Ultra was sold for 1000.00$
Xiaomi Redmi Note 10 Pro was sold for 297.00$
Uncaught Error Error: Samsung Galaxy A13 was not found!
```

Input 3

```
let retailer = new RefurbishedSmartphones('SecondLife Devices');
retailer.addSmartphone('Samsung S20 Ultra', 256, 1000, 'good');
retailer.addSmartphone('Iphone 12 mini', 128, 800, 'perfect');
retailer.addSmartphone('Xiaomi Redmi Note 10 Pro', 128, 330, 'perfect');
console.log(retailer.upgradePhones());
```

Output 3

```
Upgraded Smartphones:
Samsung S20 Ultra / 512 GB / good condition / 1000.00$
```

```
Iphone 12 mini / 256 GB / perfect condition / 800.00$

Xiaomi Redmi Note 10 Pro / 256 GB / perfect condition / 330.00$
```

Input 4

```
let retailer = new RefurbishedSmartphones('SecondLife Devices');

retailer.addSmartphone('Samsung S20 Ultra', 256, 1000, 'good');

retailer.addSmartphone('Iphone 12 mini', 128, 800, 'perfect');

retailer.addSmartphone('Xiaomi Redmi Note 10 Pro', 128, 330, 'perfect');

retailer.sellSmartphone('Samsung S20 Ultra', 256);

retailer.sellSmartphone('Xiaomi Redmi Note 10 Pro', 256);

console.log(retailer.salesJournal('model'));
```

Output 4

```
SecondLife Devices has a total income of 1297.00$

2 smartphones sold:

Samsung S20 Ultra / 256 GB / 1000.00$

Xiaomi Redmi Note 10 Pro / 128 GB / 297.00$
```