Problem 2. Vegetable store

```
class VegetableStore{
    //TODO Implement this class
}
```

Write a **class Vegetable store**, which supports the described functionality below.

Functionality

Constructor

Should have these **3** properties:

- `owner - string`
- `location - string`
- `availableProducts - empty array`

At the initialization of the **VegetableStore** class, the **constructor** accepts the **owner** and **location.**

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

loadingVegetables (vegetables)

This method makes loading of the products in the store. The method takes 1 argument: **vegetables (array of strings)**.

- **Every element** into this array is information about vegetable in the format:

  `"{type} {quantity} {price}"`

  - They are separated by a single space. The **quantity** and **price** are per unit kilogram.

    Example: [**`"Okra 2.5 3.5", "Beans 10 2.8", "Celery 5.5 2.2"…`**]
- If the **type** of the current vegetable is already present in **`availableProducts`** array, add the new quantity to the old one and update the old price per kilogram **only if** the current one is **higher**.
- Otherwise, should **add** the vegetable, with properties: **`{type, quantity, price}`** to the **`availableProducts`** **array**.
- In all cases, you must **finally return a string** in the following format:

  `` `Successfully added {type1}, {type2}, …{typeN}` ``

Note: When returning the **string**, keep in mind that the different **types** of **vegetables must** be:

- **Unique** - for instance:
  - **"Successfully added Okra, Beans, Celery"** - is a correctly returned string
  - **"Successfully added Okra, Beans, Okra"** - is not a correctly returned string
- **Separated** by **comma** and **space (`, `)**

buyingVegetables (selectedProducts)

With this method, customers can buy products from the store. The method takes 1 argument: **selectedProducts (array of strings)**.

- **Every element** in this array is information about the selected vegetables in the format:

  `"{type} {quantity}"`

- For each element of the array `selectedProducts`, check:
  - If the **type** of the current vegetable is not present in `availableProducts` array, an error with the following message should be **thrown**:

`` `{type} is not available in the store, your current bill is ${totalPrice}.` ``

    - **totalPrice -** is the total price of all customer's **purchases**, if there are **no** purchases yet the **value** should **be 0.00.**

  - If the **quantity** selected by the customer for a given vegetable **is greater** than the quantity recorded in the array `availableProducts`, an error with the following message should be **thrown**:

`` `The quantity {quantity} for the vegetable {type} is not available in the store, your current bill is ${totalPrice}.` ``

    - **totalPrice -** is the total price of all customer's **purchases**, if there are **no** purchases yet the **value** should **be 0.00.**

  - Otherwise, if the above conditions are not met, you have to **calculate** the **price** for the given vegetable by **multiplying** the price per kilogram for the **given type** by the **quantity** desired by the customer. Then reduce the quantity recorded in the `availableProducts` array.
  - **Note: Add** a **variable** that will calculate the **total price** obtained from the individual prices of **each** vegetable in the array.

- Finally, you need to **return** the string in the following format:

  `` `Great choice! You must pay the following amount ${totalPrice}.` ``

Note: The `totalPrice` must be rounded to the second decimal point and **before** the **price** must have a **dollar sign** (**$**).

rottingVegetable (type, quantity)

With this method, the freshness of the vegetables in the store is preserved, removing the rotting vegetables. The method takes 2 arguments:

- `type (string)`
- `quantity (number)`

- If the submitted **type** is not present in the **`availableProducts`** array, an error with the following message should be **thrown**:

  `` `{type} is not available in the store.` ``

- If the submitted **quantity is greater** than the quantity recorded in the **`availableProducts`** array, then the **value** of the quantity in the array becomes **zero,** and **return** the **following string:**

  `` `The entire quantity of the {type} has been removed.` ``

- Otherwise, reduce the **quantity** recorded in the array **`availableProducts`** with the quantity obtained as a parameter, and **return** the string in the following format:

  `` `Some quantity of the {type} has been removed.` ``

revision ()

- This method **returns all** available **products** in the store in the following format:
  - The first line shows the following message:

    `"Available vegetables:"`

  - On the new line, display information about each vegetable sorted in **ascending** order of **price**:

    `` `{type}-{quantity}-${price}` ``

  - The last line shows the following message:

    `` `The owner of the store is {owner}, and the location is {location}.` ``

Example

Input 1

`let vegStore = new VegetableStore("Jerrie Munro", "1463 Pette Kyosheta, Sofia");`
`console.log(vegStore.loadingVegetables(["Okra 2.5 3.5", "Beans 10 2.8", "Celery 5.5 2.2", "Celery 0.5 2.5"]));`

Output 1

```
Successfully added Okra, Beans, Celery
```

Input 2

```
let vegStore = new VegetableStore("Jerrie Munro", "1463 Pette Kyosheta, Sofia");
console.log(vegStore.loadingVegetables(["Okra 2.5 3.5", "Beans 10 2.8", "Celery 5.5 2.2", "Celery 0.5 2.5"]));
console.log(vegStore.buyingVegetables(["Okra 1"]));
console.log(vegStore.buyingVegetables(["Beans 8", "Okra 1.5"]));
console.log(vegStore.buyingVegetables(["Banana 1", "Beans 2"]));
```

Output 2

```
Successfully added Okra, Beans, Celery

Great choice! You must pay the following amount $3.50.
Great choice! You must pay the following amount $27.65.
Uncaught Error: Banana is not available in the store, your current bill is $0.00.
```

Input 3

```
let vegStore = new VegetableStore("Jerrie Munro", "1463 Pette Kyosheta, Sofia");
console.log(vegStore.loadingVegetables(["Okra 2.5 3.5", "Beans 10 2.8", "Celery 5.5 2.2", "Celery 0.5 2.5"]));
console.log(vegStore.rottingVegetable("Okra", 1));
console.log(vegStore.rottingVegetable("Okra", 2.5));
console.log(vegStore.buyingVegetables(["Beans 8", "Okra 1.5"]));
```

Output 3

```
Successfully added Okra, Beans, Celery
Some quantity of the Okra has been removed.
The entire quantity of the Okra has been removed.
Uncaught Error: The quantity 1.5 for the vegetable Okra is not available in the store, your current bill is $22.40.
```

Input 4

```
let vegStore = new VegetableStore("Jerrie Munro", "1463 Pette Kyosheta,
Sofia");
console.log(vegStore.loadingVegetables(["Okra 2.5 3.5", "Beans 10 2.8",
"Celery 5.5 2.2", "Celery 0.5 2.5"]));
console.log(vegStore.rottingVegetable("Okra", 1));
console.log(vegStore.rottingVegetable("Okra", 2.5));
console.log(vegStore.buyingVegetables(["Beans 8", "Celery 1.5"]));
console.log(vegStore.revision());
```

Output 4

```
Successfully added Okra, Beans, Celery
Some quantity of the Okra has been removed.
The entire quantity of the Okra has been removed.
Great choice! You must pay the following amount $26.15.
Available vegetables:
Celery-4.5-$2.5
Beans-2-$2.8
Okra-0-$3.5
The owner of the store is Jerrie Munro, and the location is 1463 Pette
Kyosheta, Sofia.
```