JS Advanced - Retake Exam

Exam problems for the "JavaScript Advanced" course @ SoftUni. Submit your solutions in the SoftUni judge system at https://judge.softuni.bg/Contests/Practice/Index/1624#2

Problem 3. Organization

```
class Organization {
    // TODO: Implement this class...
}
```

You need to implement a structure that will be used by organizations to track their employees. Write a **JavaScript** class **`Organization`** which has the following **functionality**:

Constructor

You need to support the following properties:

- **`name`** - string
- **`employees`** - array
- **`budget`** - number

At **initialization** of the **`Organization` class**, the **constructor** receives only **2 parameters** (**`name`** and **`budget`**).
The **`employees`** property needs to be **empty** by default.

- The **`name`** property refers to the **organization's name**.
- The **`budget`** property refers to the total **budget**.
- For **each department** there is a **portion** of the total budget:
  - The **marketing** department starts with **40%** of the **total budget**.
  - The **finance** department budget starts with **25%** of the **total budget**.
  - The **production** department budget starts with **35%** of **total budget**.

Accessors

departmentsBudget - **Returns** an **object**, containing the **current budget** of each department:

- **marketing: {`marketingBudget`}**
- **finance: {`financeBudget`}**
- **production: {`productionBudget`}**

Ensure **all properties** have the **correct data types** and the **accessor name** is the **same** as **above**.

Functions

add({employeeName}, {department} , {salary})

Receive **3** parameters: **`employeeName`** (string), **`department`** (string) and **`salary`** (number).

If the organization has **enough budget** in the requested **department** to pay the **employee's** desired **`salary`**, you should create an **object** for the current employee and store it in the **`employees`** array, with the following properties:

- **`employeeName`**
- **`department`**
- **`salary`**

You should also **descrease** the corresponding department's **`budget`**.

Then you should return a **string**, in the following format:

"**`Welcome to the {department} team Mr./Mrs. {employeeName}.`**"

Validations

- If the requested **`department`** cannot afford the employee **`salary`**, the **function** should return a **string**: "**`The salary that {departmentName} department can offer to you Mr./Mrs. {employeeName} is ${departmentBudget}.`**"

employeeExists({employeeName})

Receive **1** parameter: **`employeeName`** (string). Checks if an employee with the given name is present in the organization.

- If there is an employee with the given name, the function should return a string in the following format: "**`Mr./Mrs. {employeeName} is part of the {employeeDepartment} department.`**"
- If the employee is **NOT** part of the organization, the function should return a string in the following format: "**`Mr./Mrs. {employeeName} is not working in {organizationName}.`**"

leaveOrganization({employeeName})

Receive **1** parameter: **`employeeName`** (string).

You should **remove** the **employee** from his corresponding **department** and **increase** the department budget with its **salary**. After successfully removing the employee, the function should return a string in the following format:
"**`It was pleasure for {organizationName} to work with Mr./Mrs. {employeeName}.`**"

If the employee is **NOT** part of the organization, the function should return a string in the following format:
"**`Mr./Mrs. {employeeName} is not working in {organizationName}.`**"

status()

This function represents the organization's "database". Prints **information** about **each department**. The information should be presented in the following format:

`"{organizationName.toUpperCase()} DEPARTMENTS:`

`Marketing | Employees: {marketingEmployeesCount}: {employee1Name}, {employee2Name} | Remaining Budget: {marketingRemainingBudget}`

`Finance | Employees: {financeEmployeesCount}: {employee1Name}, {employee2Name} | Remaining Budget: {financeRemainingBudget}`

Production | Employees: {productionEmployeesCount}: {employee1Name}, {employee2Name} | Remaining Budget: {productionRemainingBudget}

The **employees**' **names** in each department, should be **sorted** by their **salary** in **descending** order.

Note that the **new line** ("**\n**") must be in the **beginning of each department**. For example:

"**\nMarketing | Employees: {marketingEmployeesCount}: {employee1Name}, {employee2Name} | Remaining Budget: {marketingRemainingBudget}**"

Submission

Submit only the **Organization class** as **JavaScript code**.

Examples

Input

`let organization = new Organization('SoftUni', 20000);`
 	
`console.log(organization.add('Peter', 'marketing', 1200));`
`console.log(organization.add('Robert', 'production', 2000));`
`console.log(organization.leaveOrganization('Peter'));`

Output

`Welcome to the marketing team Mr./Mrs. Peter.`
`Welcome to the production team Mr./Mrs. Robert.`
`It was pleasure for SoftUni to work with Mr./Mrs. Peter.`

Input

`let organization = new Organization('SBTech', 1000);`
 	
`console.log(organization.add('Peter', 'marketing', 800));`
`console.log(organization.add('Robert', 'production', 2000));`
`console.log(organization.add('Peter', 'production', 2000));`

Output
The salary that marketing department can offer to you Mr./Mrs. Peter is $400. The salary that production department can offer to you Mr./Mrs. Robert is $350. The salary that production department can offer to you Mr./Mrs. Peter is $350.