**JS OOP Fundamentals: ES6 Class Syntax**

**Problem 1. Computer**

**Your Task**

Write a **Computer** **class** which supports the described functionality below.

|  |
| --- |
| **class** **Computer** {  // ***TODO: implement this class…***  } |

**Functionality**

**constructor()**

Receives **3** parameters at **instantiation** of the class (**ramMemory, cpuGhz, hddMemory**), where each of them is a number. The class should contain the following **5 properties** (create more if needed):

* **ramMemory – number** (should be the same as the received **ramMemory**)
* **cpuGhz – number** (should be the same as the received **cpuGhz**)
* **hddMemory – number** (should be the same as the received **hddMemory**)
* **taskManager – empty array**
* **installedPrograms** **– empty array**

**installAProgram({name}, {requiredHddMemory})**

This **method** should **install a new program** on the computer and **save it** in the **installedPrograms property.**

* if the computer has enough available **hddMemory** to install the given program, the computer class should create a **new object** with the given **name** and **requiredHddMemory** and store it in the **installedPrograms array property.**

keep in mind that you should also decrease the total **hddMemory** on the computer with the current **requiredHddMemory** value for the currently installed program.

finally, on success, this **method** should return the **newly created object**

* if the total **hddMemory** is **exceeded** while trying to install a new program, **new error** should be **thrown**, with the following message:

**"There is not enough space on the hard drive!"**

* If the program is already installed, **new error** should be **thrown**, with the following message:

**"The program is already installed!"**

**uninstallAProgram({name})**

This **method** should **uninstall a program** from the computer and **save it** in the **installedPrograms property.**

* if the **installedPrograms array property** contains an object with the given name, that object should be removed from the array.

Also, logically reversed move is to free up some space off the computer, increase the **hddMemory** with the capacity of the current uninstalled program.

finally, on success, this **method** should return the **installedPrograms array,** where the given program name is excluded.

* If there is no installed program with such name, **new error** should be **thrown**, with the following message:

**"Control panel is not responding. "**

**openAProgram({name})**

This **method** should **open a program** , which is already installed on the computer. The **method** receives a string (name of the program).

* if the **name** is not present in the **installedPrograms** property, **new error** should be **thrown**, with the following message:

**"The {name} is not recognized "**

* if the given **name** is an installed program and it is already open, **new error** should be **thrown**, with the following message:

**"The {name} is already open "**

To open an installed program, you must calculate how much **RAM memory** and **CPU usage** the program will need. To find out how much:

* **RAM Memory** the current program will need, use the following formula:

**(programRequiredHddMemory / ramMemory) \* 1.5**

* **CPU Usage** the current program will need, use the following formula:

**(programRequiredHddMemory** **/ cpuGhz) \* 1.5**

**Keep in mind that both formulas calculate a number in percent (%) for the current ram and cpu usage.**

* If the **total ram usage reaches or exceeds 100%** (the ram usage for all opened programs), **new error** should be **thrown**, with the following message:

**"The {programName} caused out of memory exception."**

* If the **total cpu usage reaches or exceeds 100%** (the cpu usage for all opened programs), **new error** should be **thrown**, with the following message:

**"The {programName} caused out of cpu exception."**

* If both **ram usage and cpu usage** reaches or exceeds 100%, **new error** should be **thrown**, with the following message:

**"The {programName} caused both out of memory and out of cpu exception."**

When ram and cpu usages are calculated, **create a new object with the following properties** (the properties must be exactly as they are mentioned! Also, you must not round the numbers!):

* name (name of the program)
* **ramUsage** (current ram usage that the program uses in %)
* **cpuUsage** (current cpu usage that the program uses in %)

When the object is created, push it in the **taskManager array propery.**

**The method must return the newly created object.**

**closeAProgram({name})**

This **method** should **close a program** , which is already opened in the **task manager** on the computer. The **method** receives a string (name of the program).

* if the **taskManager array property** contains an object with the given name, that object should be removed from the array.

Also, logically reversed move is to free up some ram and cpu usage off the computer, you need to figure out a properties in which to store those values.

finally, on success, this **method** should return the **taskManager array property,** where the given program name is excluded.

* if the **name** is not present in the **installedPrograms** property, **new error** should be **thrown**, with the following message:

**"The {name} is not recognized. "**

* if the given **name** is not currently opened in the **taskManager array property**, **new error** should be **thrown**, with the following message:

**"The {name} is not currently opened. "**

**taskManagerView()**

This **method** prints all opened programs (the objects in the taskManager array property). Keep in mind that the percentages for (**cpu** and **ram usage**) must be shown, **truncated**, without the decimal part.

* if there is at least one opened program, visualize it in the following format:

**"Program – {programName} | Usage – CPU: {cpuUsage}%, RAM: {ramUsage}%"**

* If there is no opened program, render the following string:

**"All running smooth so far"**

If there is more than one opened program, each of them must be in a **new line**.

This method **returns** a **string** in the format mentioned above.

**Examples**

|  |
| --- |
| **Input** |
| let computer = new Computer(4096, 7.5, 250000);  computer.installAProgram('Word', 7300);  computer.installAProgram('Excel', 10240);  computer.installAProgram('PowerPoint', 12288);  computer.installAProgram('Solitare', 1500);  computer.openAProgram('Word');  computer.openAProgram('Excel');  computer.openAProgram('PowerPoint');  computer.openAProgram('Solitare');  console.log(computer.taskManagerView()); |
| **Output** |
| Program - Word | Usage - CPU: 3%, RAM: 3%  Program - Excel | Usage - CPU: 4%, RAM: 4%  Program - PowerPoint | Usage - CPU: 5%, RAM: 5%  **Program** **- Solitare | Usage - CPU: 1%, RAM: 1%** |

*GOOD LUCK!😊*