## Problem 2. Summer camp

Write a **class Summer camp**, which supports the described functionality below.

**Functionality**

**Constructor**

Should have these **4** properties:

* **organizer - string**
* **location - string**
* **priceForTheCamp - {"child": 150, "student": 300, "collegian": 500}**
* **listOfParticipants - empty array**

**At the initialization** of the **SummerCamp** class, the **constructor** accepts the **organizer** and **location.** The **priceForTheCamp** is an **object**, the **submitted values** are by **default** and represent the price for the stay in the camp depending on the **condition** of the participant **("child", "student", "collegian")**.

### **registerParticipant (name, condition, money)**

This method register participant to the camping. The methodaccepts 3 arguments:

* **name (string);**
* **condition (string);**
* **money (number);**
* If the given **condition** of participants, is not present in **priceForTheCamp** object with the specified default values **("child", "student", "collegian")**, an error with the following message should be **thrown**:

**"Unsuccessful registration at the camp."**

* If the **name** of the current participant is already present in **listOfParticipants** array, **return** the following message:

**`The {name} is already registered at the camp.`**

* If the submitted **money** is less than the **price** for the stay in the camp (the **price** is determined by the **priceForTheCamp** object, depending on the **condition** of the participant), **return** the following message:

**`The money is not enough to pay the stay at the camp.`**

* Otherwise, should **add** the participant, with properties: **{name, condition, power: default 100,** **wins: default 0}** to the **listOfParticipants array** and **return**:

**`The {name} was successfully registered.`**

### **unregisterParticipant (name)**

This method removes a participant from the camping. The methodaccepts 1 argument:

* **name (string)**;
* If the **name** of the current participant is not present in **listOfParticipants** array, an error with the following message should be **thrown**:

**`The {name} is not registered in the camp.`**

* **Otherwise,** this function should **remove** the participant from the **listOfParticipants** arrayand **return:**

**`The {name} removed successfully.`**

### **timeToPlay (typeOfGame, participant1, participant2)**

Method can take 2 or 3 arguments depending on the type of game:

* **typeOfGame (string);**
* **participant1 - name(string);**
* **participant2 - name(string) - optional;**
* There are two possible types of games:
* **WaterBalloonFights** -> you will get **two** players**.**

Example **-> timeToPlay ("WaterBalloonFights", "Petar", "John")**

**Note:** The **condition** of the participants must match **(**Example: **"Petar" - "child" and "John" - "child")**

* **Battleship** -> you will get **one** player**.**

Example **-> timeToPlay ("Battleship", "Petar")**

* If any of the submitted participants **names** are not presentin **listOfParticipants** array, an error with the following message should be **thrown**:

**`Invalid entered name/s.`**

* If two names are submitted, check that the participants' **condition** matches, if not matched, an error with the following message should be **thrown**:

**`Choose players with equal condition.`**

* If the type of game is **Battleship** increase the **power** property of the **participant** by a **value** of **20**, and **return** the message:

**`The {name}** **successfully completed the game {typeOfGame}.`**

* If the type of game is **WaterBalloonFights,** you must check whether the value of the **power** of one participant **is greater** than the value of the **power** of the **other** participant, and in this case increase the value of the **wins** property **by one** per **winner** (with the **bigger power**), and **return** the following message:

**`The {name} is winner in the game {typeOfGame}.`**

**Note:** The **{name}** is the name of the winner in this game.

* Otherwise, the function **returns** the message:

**`There is no winner.`**

**toString ()**

* At the first line return:

**`{organizer} will take {numberOfParticipants} participants on camping to {location}`**

* On the lines, display information about each **participant, sorted** in **descending** order by their **wins** in the following format:

**`{name} - {condition} - {power} - {wins}`**

### **Examples**

|  |
| --- |
| **Input 1** |
| **const** **summerCamp** = **new** **SummerCamp("Jane Austen", "Pancharevo Sofia 1137, Bulgaria");**  console.log(summerCamp.registerParticipant("Petar Petarson", "student", 200));  console.log(summerCamp.registerParticipant("Petar Petarson", "student", 300));  console.log(summerCamp.registerParticipant("Petar Petarson", "student", 300));  console.log(summerCamp.registerParticipant("Leila Wolfe", "childd", 200)); |

|  |
| --- |
| **Output 1** |
| The money is not enough to pay the stay at the camp.  The Petar Petarson was successfully registered.  The Petar Petarson is already registered at the camp.  **Uncaught Error:** Unsuccessful registration at the camp. |

|  |
| --- |
| **Input 2** |
| **const** **summerCamp** = **new** **SummerCamp("Jane Austen", "Pancharevo Sofia 1137, Bulgaria");**  console.log(summerCamp.registerParticipant("Petar Petarson", "student", 300));  console.log(summerCamp.unregisterParticipant("Petar"));  console.log(summerCamp.unregisterParticipant("Petar Petarson")); |

|  |
| --- |
| **Output 2** |
| The Petar Petarson was successfully registered.  **Uncaught Error:** The Petar is not registered in the camp.  The Petar Petarson removed successfully. |

|  |
| --- |
| **Input 3** |
| **const** **summerCamp** = **new** **SummerCamp("Jane Austen", "Pancharevo Sofia 1137, Bulgaria");**  console.log(summerCamp.registerParticipant("Petar Petarson", "student", 300));  console.log(summerCamp.timeToPlay("Battleship", "Petar Petarson"));  console.log(summerCamp.registerParticipant("Sara Dickinson", "child", 200));  console.log(summerCamp.timeToPlay("WaterBalloonFights", "Petar Petarson", "Sara Dickinson"));  console.log(summerCamp.registerParticipant("Dimitur Kostov", "student", 300));  console.log(summerCamp.timeToPlay("WaterBalloonFights", "Petar Petarson", "Dimitur Kostov")); |

|  |
| --- |
| **Output 3** |
| The Petar Petarson was successfully registered.  The Petar Petarson successfully completed the game Battleship.  The Sara Dickinson was successfully registered.  **Uncaught Error:** Choose players with equal condition.  The Dimitur Kostov was successfully registered.  The Petar Petarson is winner in the game WaterBalloonFights. |

|  |
| --- |
| **Input 4** |
| **const** **summerCamp** = **new** **SummerCamp("Jane Austen", "Pancharevo Sofia 1137, Bulgaria");**  console.log(summerCamp.registerParticipant("Petar Petarson", "student", 300));  console.log(summerCamp.timeToPlay("Battleship", "Petar Petarson"));  console.log(summerCamp.registerParticipant("Sara Dickinson", "child", 200));  console.log(summerCamp.timeToPlay("WaterBalloonFights", "Petar Petarson", "Sara Dickinson"));  console.log(summerCamp.registerParticipant("Dimitur Kostov", "student", 300));  console.log(summerCamp.timeToPlay("WaterBalloonFights", "Petar Petarson", "Dimitur Kostov"));  console.log(summerCamp.toString()); |

|  |
| --- |
| **Output 4** |
| The Petar Petarson was successfully registered.  The Petar Petarson successfully completed the game Battleship.  The Sara Dickinson was successfully registered.  **Uncaught Error:** Choose players with equal condition.  The Dimitur Kostov was successfully registered.  The Petar Petarson is winner in the game WaterBalloonFights.  Jane Austen will take 3 participants on camping to Pancharevo Sofia 1137, Bulgaria  Petar Petarson - student - 120 - 1  Sara Dickinson - child - 100 - 0  Dimitur Kostov - student - 100 - 0 |