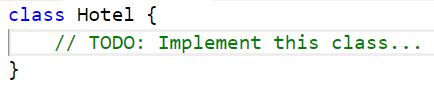
# JS Advanced - Exam

## Problem 2. Hotel



Write a **JavaScript** class Hotelwhich has the following **functionality**:

### Constructor

You should implement at least these properties:

* name - string
* capacity - number
* bookings - array
* currentBookingNumber – number (starts from 1)

At **initialization** of the Hotel **class**, the **constructor** receives only **2** **parameters** (**name**, **capacity**).   
The bookings property needs to be **empty** array by **default.**

* Thenamepropertyrefers to the **hotel's** **name**.
* Thecapacitypropertyrefers for the **count** of the **hotel's** **rooms**.
  + The **single** rooms are **50 %** of the given capacity
  + The **double** rooms are **30 %** of the given capacity
  + The **maisonette** rooms are **20 %** of the given capacity
  + **NOTE**: All room counts should be integers and will always be valid

**Hint:** You can add additional fields to keep the necessary information.

### Accessors

roomsPricing - **Returns** an **object**, containing the following properties:

* **single: 50** (price per night)
* **double: 90** (price per night)
* **maisonette: 135** (price per night)

**Hint: It must be getter!**

### Functions

#### rentARoom( {clientName}, {roomType}, {nights})

Receive **3** parameters:  **clientName** (string), **roomType** (string), **nights** (number)

If the hotel has **available** rooms of the requested roomType, you should create an **object** for the current client’s **booking** which contains properties for (**client name**, **room type**, **nights**, and **rent number**) and **store** it into the bookings property.

* You should also **increase** the currentBookingNumber by 1. The first given currentBookingNumber is 1.
* You should also **decrease** the **capacity** of the requested roomType rooms by 1.

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

"Enjoy your time here Mr./Mrs. {clientName}. Your booking is {currentBookingNumber}."

##### Validations

* If there are **NO** available rooms of the requested roomType, the **function** shouldreturn a **string**, consisting of the message: "No {roomType} rooms available!"

And for **each** of the **other** **available** roomtypes, you should **append** a **string** to the **message** in the following format:

"Available {roomType} rooms: {roomTypeRoomsRemaining}."

Example:

"No single rooms available! Available double rooms: 2. Available maisonette rooms: 1."

Messages must be separated by a **single space**.

#### checkOut({currentBookingNumber})

Receive **one** parameter - currentBookingNumber (number)

You should **calculate** the due amount, as totalMoney, for the **booking** with the given currentBookingNumber, depending on the **nights** spent and the **price** of the roomType.

* You should also **increase** the **capacity** of the given roomType rooms by 1.
* Remove booking from the array.

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

"We hope you enjoyed your time here, Mr./Mrs. {clientName}. The total amount of money you have to pay is {totalMoney} BGN."

* Information about the other rooms is not return, if there are no free ones.

##### Validations

* If the given currentBookingNumber is **invalid** (**non-existent**)return the following string:  
  "The booking {currentBookingNumber} is invalid."

#### **report()**

**This function represents the hotel's "database". Return information about each booking.**

* The hotel’s name must be with capital letters.

**The information should be presented in the following format:**

**"**{hotelName} DATABASE:  
-------------------- **(separate the header and bookings information - 20 dashes)**bookingNumber - {bookingNumber}  
clientName - {clientName}  
roomType - {roomType}  
nights - {nights}  
---------- **(booking separator - 10 dashes**

#### **Validations**

* If there are no bookings, return the following string:
* "{hotelName} DATABASE:

-------------------- (separate the header and bookings information - 20 dashes)

There are currently no bookings."

* There is no new line and delimiter after the last reservation.

### Submission

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

### Constraints

**The given capacity will greater or equal to 10.**

### Examples

|  |
| --- |
| **Input 1** |
| let hotel = new Hotel('HotUni', 10);  console.log(hotel.rentARoom('Peter', 'single', 4));  console.log(hotel.rentARoom('Robert', 'double', 4));  console.log(hotel.rentARoom('Geroge', 'maisonette', 6)); |

|  |
| --- |
| **Output 1** |
| Enjoy your time here Mr./Mrs. Peter. Your booking is 1.  Enjoy your time here Mr./Mrs. Robert. Your booking is 2.  Enjoy your time here Mr./Mrs. Geroge. Your booking is 3. |

|  |
| --- |
| **Input 2** |
| let hotel = new Hotel('HotUni', 10);  hotel.rentARoom('Peter', 'single', 4);  hotel.rentARoom('Robert', 'double', 4);  hotel.rentARoom('Geroge', 'maisonette', 6);  console.log(hotel.report()); |
| **Output 2** |
| HOTUNI DATABASE:  --------------------  bookingNumber - 1  clientName - Peter  roomType - single  nights - 4  ----------  bookingNumber - 2  clientName - Robert  roomType - double  nights - 4  ----------  bookingNumber - 3  clientName - Geroge  roomType - maisonette  nights - 6 |

class Hotel {

    constructor(name, capacity) {

        this.name = name;

        this.capacity = capacity;

        this.bookings = [];

        this.currentBookingNumber = 1;

        this.roomCapacity = {

            single: this.capacity \* 0.5,

            double: this.capacity \* 0.3,

            maisonette: this.capacity \* 0.2

        };

    }

    get roomsPricing() {

        return {

            single: 50,

            double: 90,

            maisonette: 135

        }

    }

    rentARoom(clientName, roomType, nights) {

        if (this.roomCapacity[roomType] >= 1) {

            this.bookings.push({ clientName, roomType, nights, currentBooking: this.currentBookingNumber })

            this.currentBookingNumber++;

            this.roomCapacity[roomType] -= 1;

            return `Enjoy your time here Mr./Mrs. ${clientName}. Your booking is ${currentBooking}.`

        } else {

            let result = [];

            result.push(`No ${roomType} rooms available.`);

            for (const key in this.roomCapacity) {

                //singe !== double

                if (key !== roomType) {

                    result.push(`Available ${key} rooms: ${this.roomCapacity[key]}.`)

                }

            }

            return result.join(' ');

        }

    }

    checkOut(currentBookingNumber) {

        let currentCheckout = this.bookings.find(b => b.currentBooking == currentBookingNumber);

        if (!currentCheckout) {

            return `The booking ${currentBookingNumber} is invalid.`

        } else {

            let pricePerNight = this.roomsPricing[currentCheckout.roomType];

            let totalMoney = pricePerNight \* currentCheckout.nights;

            this.roomCapacity[currentCheckout.roomType]++;

            let index = this.bookings.indexOf(currentCheckout);

            this.bookings.splice(index, 1);

            return `We hope you enjoyed your time here, Mr./Mrs. ${currentCheckout.clientName}. The total amount of money you have to pay is ${totalMoney} BGN."`

        }

    }

    report() {

        if (this.bookings.length == 0) {

            let result = [];

            result.push(`${this.name} DATABASE:`);

            result.push('-------------------- ');

            result.push(`There are currently no bookings.`)

            return result.join('\n')

        }

        let result = [];

        let count = this.bookings.length;

        result.push(`${this.name} DATABASE:`);

        result.push('-------------------- ');

        this.bookings.forEach(c => {

            if (count > 1) {

                result.push(`bookingNumber - ${c.currentBooking}`)

                result.push(`clientName - ${c.clientName}`)

                result.push(`roomType - ${c.roomType}`)

                result.push(`nights - ${c.nights}`)

                result.push(`----------`)

                count--;

            } else {

                result.push(`bookingNumber - ${c.currentBooking}`)

                result.push(`clientName - ${c.clientName}`)

                result.push(`roomType - ${c.roomType}`)

                result.push(`nights - ${c.nights}`)

            }

        });

        return result.join('\n')

    }

}

let hotel = new Hotel('HotUni', 10);

hotel.rentARoom('Peter', 'single', 4);

hotel.rentARoom('Robert', 'double', 4);

hotel.rentARoom('Geroge', 'maisonette', 6);