# JS Advanced: Retake Exam 26 April 2018

Problems for retake exam [“JavaScript Advanced” course @ SoftUni](https://softuni.bg/courses/javascript-advanced). Submit your solutions in the SoftUni judge system at <https://judge.softuni.bg/Contests/1004/>.

# Problem 2. Subscription Card (Unit Testing)

You are given the following **JavaScript class**:

|  |
| --- |
| SubscriptionCard.js |
| **class** SubscriptionCard {  constructor(firstName, lastName, SSN) {  **this**.**\_firstName** = firstName;  **this**.**\_lastName** = lastName;  **this**.**\_SSN** = SSN;  **this**.**\_subscriptions** = [];  **this**.**\_blocked** = **false**;  }   **get** firstName() {  **return this**.**\_firstName**;  }   **get** lastName() {  **return this**.**\_lastName**;  }   **get** SSN() {  **return this**.**\_SSN**;  }   **get** isBlocked() {  **return this**.**\_blocked**;  }   addSubscription(line, startDate, endDate) {  **this**.**\_subscriptions**.push({  line,  startDate,  endDate  });  }   isValid(line, date) {  **if** (**this**.isBlocked) **return false**;  **return this**.**\_subscriptions**.filter(s => s.line === line || s.line === **'\*'**)  .filter(s => {  **return** s.**startDate** <= date &&  s.**endDate** >= date;  }).length > 0;  }   block() {  **this**.**\_blocked** = **true**;  }   unblock() {  **this**.**\_blocked** = **false**;  } } module.exports = SubscriptionCard; |

### Functionality

The above code defines a **class** that contains information about a **subscription card**. An **instance** of the class should support the following operations:

* Can be **instantiated** with three parameters – first and last name and a Social Security number (SSN)
* Accessor firstName – used to get the value of **firstName**.
* Accessor lastName – used to get the value of **lastName**.
* Accessor SSN – used to get the value of **SSN**.
* Accessor isBlocked – used to find out if the card is blocked.
* Function addSubscription() – adds an entry in the subscriptions with start and end date.
* Function isValid() – return true if the card is valid for the given date; read further for validity checks
* Function block() – set block to true.
* Function unblock() – set block to false.

Once created, the values of **firstName**, **lastName** and **SSN** must not be changed, so these properties do not have defined setters and should not be directly accessible. The values are **not validated**.

A subscription will have the following format:

{

line: String,

startDate: Date,

endDate: Date

}

The value of **line** can be set to "**\***" (asterisk), which means it covers all lines. The function isValid() must iterate through all available **subscriptions** and see if there is one for the **given line**, or one for **all lines**, with starting date before or on the **given date** and end date after or on the **given date**. If the card is blocked, the function must always return false.

***Scroll down for examples and details about submitting to Judge.***

### Examples

This is an example how this code is **intended to be used**:

|  |
| --- |
| Sample code usage |
| const card = new **SubscriptionCard**('Pesho', 'Petrov', '00000000');  card.**addSubscription**('120', new **Date**('2018-04-22'), new **Date**('2018-05-21'));  card.**addSubscription**('\*', new **Date**('2018-05-25'), new **Date**('2018-06-24'));  card.**block**();  card.**unblock**();  **console**.**log**(card.**isValid**('120', new **Date**('2018-04-22')));  card.firstName = 'Gosho';  **console**.**log**(card.firstName); |
| Corresponding output |
| True  pesho |

### Your Task

Using **Mocha** and **Chai** write **JS unit tests** to test the entire functionality of the SubscriptionCard class. Make sure instances of it have all the required functionality and validation. You may use the following code as a template:

|  |
| --- |
| describe(**"*TODO* …"**, **function**() {  ***it***(**"*TODO …*"**, **function**() {  *//* ***TODO:*** …  });  *//* ***TODO:*** …  }); |

### Submission

Submit your tests inside a describe() statement, as shown above.