JS Advanced: Exam 18 March 2018

Problems for exam preparation for the "JavaScript Advanced" course @ SoftUni. Submit your solutions in the SoftUni judge system at https://judge.softuni.bg/Contests/974/.

Problem 2. Payment Package (Unit Testing)

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

PaymentPackage.js

```
class PaymentPackage {
  constructor(name, value) {
    this.name = name;
    this.value = value;
    this.VAT = 20;      // Default value
    this.active = true; // Default value
  }

  get name() {
    return this._name;
  }

  set name(newValue) {
    if (typeof newValue !== 'string') {
      throw new Error('Name must be a non-empty string');
    }
    if (newValue.length === 0) {
      throw new Error('Name must be a non-empty string');
    }
    this._name = newValue;
  }

  get value() {
    return this._value;
  }

  set value(newValue) {
    if (typeof newValue !== 'number') {
      throw new Error('Value must be a non-negative number');
    }
    if (newValue < 0) {
      throw new Error('Value must be a non-negative number');
    }
    this._value = newValue;
  }

  get VAT() {
    return this._VAT;
  }

  set VAT(newValue) {
    if (typeof newValue !== 'number') {
      throw new Error('VAT must be a non-negative number');
```

```
    }
    if (newValue < 0) {
      throw new Error('VAT must be a non-negative number');
    }
    this._VAT = newValue;
  }

  get active() {
    return this._active;
  }

  set active(newValue) {
    if (typeof newValue !== 'boolean') {
      throw new Error('Active status must be a boolean');
    }
    this._active = newValue;
  }

  toString() {
    const output = [
      `Package: ${this.name}` + (this.active === false ? ' (inactive)' : ''),
      `- Value (excl. VAT): ${this.value}`,
      `- Value (VAT ${this.VAT}%): ${this.value * (1 + this.VAT / 100)}`
    ];
    return output.join('\n');
  }
}
```

Functionality

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

- Can be **instantiated** with two parameters – a string name and number value
- Accessor **name** – used to get and set the value of name
- Accessor **value** – used to get and set the value of value
- Accessor **VAT** – used to get and set the value of VAT
- Accessor **active** – used to get and set the value of active
- Function **toString()** – return a string, containing an overview of the instance; if the package is **not active**, append the label "**(inactive)**" to the printed **name**

When creating an instance, or changing any of the property values, the parameters are validated. They must follow these rules:

- **name** – non-empty string
- **value** – non-negative number
- **VAT** – non-negative number
- **active** – Boolean

If any of the requirements aren't met, the operation must throw an error.

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

```
// Should throw an error
try {
    const hrPack = new PaymentPackage('HR Services');
} catch(err) {
    console.log('Error: ' + err.message);
}
const packages = [
    new PaymentPackage('HR Services', 1500),
    new PaymentPackage('Consultation', 800),
    new PaymentPackage('Partnership Fee', 7000),
];
console.log(packages.join('\n'));

const wrongPack = new PaymentPackage('Transfer Fee', 100);
// Should throw an error
try {
    wrongPack.active = null;
} catch(err) {
    console.log('Error: ' + err.message);
}
```

Corresponding output

```
Error: Value must be a non-negative number
Package: HR Services
- Value (excl. VAT): 1500
- Value (VAT 20%): 1800
Package: Consultation
- Value (excl. VAT): 800
- Value (VAT 20%): 960
Package: Partnership Fee
- Value (excl. VAT): 7000
- Value (VAT 20%): 8400
Error: Active status must be a boolean
```

Your Task

Using **Mocha** and **Chai** write **JS unit tests** to test the entire functionality of the **`PaymentPackage`** 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.