FL10. Offline. Lesson 15. OOP

**DEADLINE:** 24/03/2019

## FOLDER STRUCTURE

FLX\_homework\_15/

├─ homework

├─ src/

├─ index.js

├─ .eslintrc.js

└─ index.html

## TASK

1. You should create two classes **Employee** and **Company**.
2. Constructor of **Company** class should expect three arguments:
   1. **name** - Name of company.
   2. **owner** - Owner of company.
   3. **maxCount** - Max count of employees who could work in company.
3. Constructor of **Employee** class should expect four arguments:
   1. **name** - Name of employee.
   2. **primarySkill** - Primary skill.
   3. **age** - Age of employee.
   4. **salary** - Salary of employee.
4. **Company** class should have at least following methods:
   1. **addNewEmployee** - add new employee to list of the company( If there no free place for new employee, you should remove employee who has the lowest salary and add new employee).
   2. **removeEmployee** - remove employee by id(it can be index in array).
   3. **getAvarageSalary** - return average salary of employees.
   4. **getEmployees** - return all employees. (employees field should be private).
   5. **getFormattedListOfEmployees** - return list of employees in next format `${name} - works in ${companyName} ${timeInCompany} seconds`
   6. **getAvarageAge** - return average age of employees.
   7. **getHistory** - return history log(you should have private field **\_logs** where save all information about actions).
5. **Employee** class should have at least following methods:
   1. **getSalary** - return salary of employee.
   2. **setSalary** - sets salary of employee(cannot set smaller salary than employee has now).
   3. **getWorkTimeInSeconds** - return work time in seconds (should calculate time which employee works in all company).
   4. **hire** - sets current name of company.
   5. **fire** - remove name of company.
   6. **getHistory** - return history log(All actions fire/hire should be save information message in private field).
6. All functions should contain assertions that prevent non-logical behaviour(example: set to salary only finite number)
7. There should be some script that demonstrates work of all functions.

## RESTRICTIONS

* Do not use ES6 classes
* Do not use any of build systems(e.g. Grunt, Gulp, Webpack).

## BEFORE SUBMIT

* Check files structure;
* Check code structure(code style);
* Make sure your code doesn’t have mistakes

## USEFUL LINKS

* <http://eloquentjavascript.net/1st_edition/chapter8.html>
* <https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/Object-oriented_JS>