# **Task 1**

Implement the *Student*class, the constructor of which accepts 2 parameters *fullName*- the name and surname of the student, *direction*- the direction in which he studies.

Create a *showFullName()*method that returns the student's first and last name.

Create a *nameIncludes(data)* method that, using the *showFullName()*method, checks for the text *data*argument in the student’s name and returns true if a match is found or false if not found.

Create a static *studentBuilder()*method that returns a new instance of the class named ‘Ihor Kohut’ and the direction ‘qc’.

Create a getter and setter *direction()*to read and specify the *direction*field.

Create an instance of class *stud1*named 'Ivan Petrenko' and direction 'web'.

Create an instance of class *stud2*named 'Sergiy Koval' and direction 'python'.

Create an instance of the *stud3*class named ‘Ihor Kohut’ and the direction ‘qc’ using the static *studentBuilder()* method.

Usage example:

const stud1 = new Student('Ivan Petrenko', 'web');

stud1.nameIncludes('Ivan');   // true

stud1.nameIncludes('Denysenko'); // false

# **Task 2**

Create a *Person* class that has a constructor that takes *name* and *surname* parameters and contains a *showFullName()* method that displays the person's first and last names.

From the *Person* class, the *Student* class is inherited, the constructor of which, in addition to *name* and *surname*, takes the parameter *year* (the year of entering the university).

In the *Student* class, you need to override the *showFullName(midleName)* method to display not only the first and last name, but also the middle name (*midleName*) of the student.

Also, in the *Student* class, you need to implement the *showCourse()* method, which will display the student's current course (from 1 to 6). The value of the course will be determined as the difference between the current year (to determine independently) and the year of admission to the university year.

Result example:

const stud1 = new Student("Petro", "Petrenko", 2017);

console.log(stud1.showFullName("Petrovych")); // Petrenko Petro Petrovych

console.log("Current course: " + stud1.showCourse()); //Current course: 4

**Task 3**

# Create a *Worker* class that will have a constructor that accepts the following properties: *fullName* (first and last name), *dayRate* (rate per day of work), *workingDays* (number of days worked).

# 1) the class must have a *showSalary()* method that will display the employee's salary. Salary is the product of the *dayRate* by the number of days worked *workingDays*.

# 2) add a personal *experience* field and assign it a value of 1.2 and use it as an additional multiplier when determining the salary - create the *showSalaryWithExperience()* method. Print the salary value with this coefficient.

# 3) add getters and setters for the *experience* field. Set experience = 1.5 and display it.

# 4) Derive salary value with new experience.

# 5) Create multiple instances of the class (workers) with different salaries as shown in the example below. Sort the salaries of the most experienced workers by growth and display the result in the format: worker\_fullName: salary\_value

Example usage:

const worker1 = new Worker("John Johnson", 20, 23);

console.log(worker1.fullName);

worker1.showSalary();

console.log("New experience: " + worker1.showExp);

worker1.showSalaryWithExperience();

worker1.setExp = 1.5;

console.log("New experience: " + worker1.showExp);

worker1.showSalaryWithExperience();

const worker2 = new Worker("Tom Tomson", 48, 22);

. . . . . .

const worker3 = new Worker("Andy Ander", 29, 23);

. . . . . .

Output example:

John Johnson

John Johnson salary: 460

New experience: 1.2

John Johnson salary: 552

New experience: 1.5

John Johnson salary: 690

Tom Tomson

Tom Tomson salary: 1056

. . . . . .

Andy Ander

Andy Ander salary: 667

. . . . . .

Sorted salary:

John Johnson: 690

Andy Ander: 1000.5

Tom Tomson: 1584