# Lab: Classes

Problems for in-class lab for the ["JavaScript Advanced" course @ SoftUni](https://softuni.bg/trainings/3588/js-advanced-january-2022). Submit your solutions in the SoftUni judge system at <https://judge.softuni.bg/Contests/2768/Classes-Lab>.

## Person

Write a **class** that represents a personal record. It has the following properties, all set from the constructor:

* firstName
* lastName
* age
* email

And a method toString(), which prints a summary of the information. See the example for formatting details.

### Input

The constructor function will receive valid parameters.

### Output

The toString()method should **return** a string in the following format:

**`{firstName} {lastName} (age: {age}, email: {email})`**

Submit the class definition as is, **without** wrapping it in any function.

### Example

|  |
| --- |
| Sample Input |
| let person = new Person('Anna', 'Simpson', 22, 'anna@yahoo.com');  console.log(person.toString()); |
| Output |
| Anna Simpson (age: 22, email: anna@yahoo.com) |

class Person{

    constructor(firstName, lastName, age, email) {

        this.firstName = firstName,

        this.lastName = lastName,

        this.age = age,

        this.email = email

    }

    toString() {

        return `${this.firstName} ${this.lastName} (age: ${this.age}, email: ${this.email})`

    };

}

    const myPerson = new Person('John', 'Smith', 27, 'john@mail.com');

    console.log(myPerson.toString())

## Get Persons

Write a function that returns an array of **Person** objects. Use the class from the previous task, create the following instances, and return them in an array:

|  |  |  |  |
| --- | --- | --- | --- |
| First Name | Last Name | Age | Email |
| Anna | Simpson | 22 | anna@yahoo.com |
| SoftUni |  |  |  |
| Stephan | Johnson | 25 |  |
| Gabriel | Peterson | 24 | g.p@gmail.com |

For any empty cells, do not supply a parameter (call the constructor with fewer parameters).

### Input / Output

There will be **no input**, the data is static and matches the table above. As **output**, **return an array** with **Person** **instances**.

Submit a function that returns the required output.

|  |
| --- |
| function solve (){ |
|  | class Person { |
|  | constructor(firstName, lastName, age, email){ |
|  | this.firstName=firstName; |
|  | this.lastName=lastName; |
|  | this.age=age; |
|  | this.email=email; |
|  | } |
|  | toString(){ |
|  | return `${this.firstName} ${this.lastName} (age: ${this.age}, email: ${this.email})` |
|  | } |
|  | } |
|  |  |
|  | let people = []; |
|  |  |
|  | let person1 = new Person('Anna', 'Simpson', 22, 'anna@yahoo.com'); |
|  | let person2 = new Person('SoftUni'); |
|  | let person3 = new Person('Stephan', 'Johnson', 25); |
|  | let person4 = new Person('Gabriel', 'Peterson', 24, "g.p@gmail.com"); |
|  |  |
|  | people.push(person1);people.push(person2);people.push(person3);people.push(person4); |
|  |  |
|  | return people; |
|  | } |

## Circle

Write a **class** that represents a **Circle**. It has only one data property - its **radius**, and it is set through the **constructor**. The class needs to have **getter** and **setter** methods for its **diameter** - the setter needs to calculate the radius and change it and the getter needs to use the radius to calculate the diameter and return it.

The circle also has a getter area(), which calculates and **returns** its area.

### Input

The constructor function and diameter setter will receive valid parameters.

### Output

The diameter() and area() getters should **return** numbers.

Submit the class definition as is, **without** wrapping it in any function.

### Examples

|  |  |
| --- | --- |
| Sample Input | Output |
| let c = new Circle(2);  console.log(`Radius: ${c.radius}`);  console.log(`Diameter: ${c.diameter}`);  console.log(`Area: ${c.area}`);  c.diameter = 1.6;  console.log(`Radius: ${c.radius}`);  console.log(`Diameter: ${c.diameter}`);  console.log(`Area: ${c.area}`); | Radius: 2  Diameter: 4  Area: 12.566370614359172  Radius: 0.8  Diameter: 1.6  Area: 2.0106192982974678 |

class Circle {

    constructor(radius){

        this.radius=radius;

    }

    get diameter() {

        return this.radius \* 2;

    }

    set diameter(diameter) {

        this.radius = diameter / 2;

    }

    get area() {

        return this.radius\*this.radius\*Math.PI;

      }

}

## Point Distance

Write a JS **class** that represents a **Point**. It has **x** and **y** coordinates as properties, that are set through the constructor, and a **static method** for finding the distance between two points, called distance().

### Input

The distance() method should receive two **Point** objects as parameters.

### Output

The distance() method should **return** a number, the distance between the two-point parameters.

Submit the class definition as is, **without** wrapping it in any function.

### Example

|  |  |
| --- | --- |
| Sample Input | Output |
| let p1 = new Point(5, 5);  let p2 = new Point(9, 8);  console.log(Point.distance(p1, p2)); | 5 |

class Point {

    constructor(x,y) {

        this.x = x;

        this.y = y;

    }

    static distance(p1,p2) {

        return Math.sqrt((p1.x - p2.x) \*\* 2 + (p1.y - p2.y) \*\* 2);

    }

}

const point1 = new Point(1,1);

const point2 = new Point(4,5);

console.log(point1, point2);

console.log(Point.distance(point1, point2));