1 Practice 5.1. Classes

> This practice is the next in the series of practices consisting of a Contacts application. This practice is based on the previous one. Before start it, finish the previous one, make the corrections you consider, if needed, and start coding.
>
> The inclusion of comments in the scripts will be valued.
>
> Before start, create a new git branch named `contacts-u5` and change to it:
>
> ```
> git branch contacts-u5
> git checkout contacts-u5
> ```
>
> Do all the practice in the contacts-u5 branch. Once finished, do the submission via GitHub Classroom as detailed in the end of the practice.

1.1 Exercise 1

Make a class, named **`Contact`**, for your contacts. The class must have the next **private** properties:

- $id (int)
- $title (string)
- $name (string)
- $surname (string)
- $birthdate (string)
- $phone (string)
- $email (string)
- $favourite (bool)
- $important (bool)
- $archived (bool)

Each property must have its **set** and **get** methods. These methods and the class must be **strictly typed**.

Add a **constructor** that gets an optional array as parameter and assigns the values of the array to the object properties. If no parameters are passed, assign them default values.

For example, the constructor definition could be:

```
public function __construct(array $contactArray = array("id"=> 0, "title"=>
↪   "Mr.", "name"=> "", "surname"=> "", "birthdate"=>"", "phone"=>"",
↪   "email"=>"", "favourite"=>false, "important"=>false,
↪   "archived"=>false))
```

Override the method **`__toString()`**, to return a string with all the data of the object, formatted properly.

Modify your `contact_form`.php and `contact_list` scripts to work with this class. You must use objects of the class `Contact` instead of arrays and variables(ex. `$contact->getName()` instead of `$contact['name']`).

1.2 Exercise 2

Move the functions `checkContactDate` and `checkBirthday` from the file `functions.php` from the Unit 3 to the class. Also move the function `validateContact` from the `functions.php` file of the practice 4.1 to the class. They will become class methods. Their signatures should be:

- **`checkContactDate() : bool`**
- **`checkBirthday() : ?int`**
- **`validate() : array`**

In addition to the previous validations and functionality, do the next validations with these methods:

- Use the `validate()` method to validate the contact data and return an array with validation errors.
- Use the `checkContactDate()` method to check the validity of the date in the `contact_form` script.
- Use the `checkBirthday()` method to calculate the numbers of days until the contact's birthday in the `contact_form` script. If the number of days returned is 0, show the message:
  - *Today is CONTACTNAME's birthday.* (replace CONTACTNAME with the contact's name).

  If the number of days is between 1 and 7, show the message:
  - *There are X days left for CONTACTNAME's birthday.*

1.3 How to submit to GitHub Classroom

1. This task must be submitted to the same repository of the previous one (Practice 4.1). Remember to make a new branch before start the practice.
2. Once you finish the task, make a commit with the comment "PRACTICE 5.1 SUBMISSION COMMIT", merge the branch into the main branch, and push it to GitHub. For example:

```
git commit -m "PRACTICE 5.1 SUBMISSION COMMIT"
git checkout main
git merge contacts-u5
git push
```

3. Before that, you can do the commits and pushes you want. If you change your code after your submission commit, make another commit and push with the same text in the message adding the corrections you've done.

If you have any doubt in your task, you can push your code and ask me by email what's your problem.