Lab 7 – Programming Arrays, Objects

# Purpose

* Programming using PHP Arrays
* Programming using PHP Objects

# Due Date

* This lab must be handed in:

**Sunday Mar 09, 2025 – before midnight**

# Assessment

* This Lab is worth 2% of your total course mark.

# Assigned Readings

* **Lecture Slides** posted on Brightspace
* Module 3 -> Part 3
* The following chapters of **Fundamentals of Web Development** will be useful in completing this exercise:
* Chapter 12

# Lab Supplies

To complete this lab you will require the following lab supplies:

* Lecture Slides (**Module 3 -> Part 3**) posted on Brightspace
* Textbook: **Fundamentals of Web Development** by Randy Connolly and Ricardo Hoar
* EasyPHP, or other WAMP server
* Eclipse, Notepad++ (or other text editor, or IDE)

# Summary of Tasks

1. Develop the logic to solve and display the output for Arrays.php.
2. Develop the logic to solve and display the output for Objects.php.
3. View your webpage using a web browser
4. Submit source code of all PHP files on Brightspace

# Task 1

Implement the following Design Pattern to create a ‘Common Look and Feel’ to be used on every page of your website.

|  |
| --- |
|  |

Your web site will include the following PHP scripts:

* Header.php
* Footer.php
* Menu.php
* Arrays.php
* Employee.php, Supervisor.php, Objects.php

**Header.php**

Header.php must contain a script to display a Common Header that will appear on every page. The header must display Program Name and Course Name

**Footer.php**

Footer.php must contain a script to display a Common Footer that will appear on every page. The footer must contain Student Number, First Name, Last Name, and Email Address

**Menu.php**

Menu.php must contain a script to display a Common Menu to be shown on every page. The menu must contain links to Arrays.php and Objects.php.

**Arrays.php**

In a PHP main program, create an array using the following values: 9, 3, 1, 0, 99, 2, 5, 6, 1, 1, 55.

Write a function that passes the array as parameter and computes the average entry of the values in the array. **This function must return the computed average value to the main program.** In the main program, you will display: “The average is: <insert result>”.

Write another function that passes the array as parameter and finds the entry with the highest value. **This function must return the maximum value and the index of the corresponding value to the main program.** In the main program, you will display: The value at index <insert the index> is maximum which is: <insert the maximum value>”.

Write another function that passes the array as parameter and finds the mode of the array. **This function must return the mode value to the main program.** In the main program, you will display: "The mode of the array is <insert the mode>”

\*\* The "mode" is the value that occurs most often in an array.

In the main program, create another array with the following values: 9, 4, 1, 0, 23, 22, 4, 6, 4, 32, 55.

Write a new function that passes both arrays as parameters and tests the values of both arrays. If the values at the same index match, **this function must return the matching indices to the main program.** In the main program, you will display: “The values at <insert the matching index> match.”

The sample output for Arrays.php is below:

|  |
| --- |
|  |

**Employee.php/Supervisor.php/ Objects.php**

Implement the following PHP scripts:

***Employee.php***:

Define a class ***Employee*** which contains protected properties: *employeeId, firstName, lastName****.*** Create a ***constructor*** method that takes in *employeeId, firstName, lastName****.*** You need to write the getters and setters of the corresponding properties of the class.

The prototype of the Employee class is below:

|  |
| --- |
| **class** Employee{  **protected** $employeeId;  **protected** $firstName;  **protected** $lastName;    **function** \_\_construct($employeeId, $firstName, $lastName){  }  **public function** getEmployeeId(){  }  **public function** setEmployeeId($employeeId){  }  **public function** getFirstName(){  }  **public function** setFirstName($firstName){  }  **public function** getLastName(){  }  **public function** setLastName($lastName){  }  } |

***Supervisor.php***:

Define a derived class ***Supervisor*** that inherits from the ***Employee*** class and contains a private property: *employees*. Create a ***constructor*** method that takes in *employeeId, firstName, lastName, employees****.*** You may need to call the parent ***constructor*** for *employeeId, firstName, lastName****.*** You need to write the getters and setters of the corresponding property of the class.

The prototype of the Supervisor class is below:

|  |
| --- |
| **class** Supervisor **extends** Employee{  **private** $employees;  **function** \_\_construct($employeeId, $firstName, $lastName, $employees){  }  **public function** getEmployees(){  }  **public function** setEmployees($employees){  }  } |

***Objects.php:***

Include Employee.php and Supervisor.php.

Instantiate (Create) six objects (employee1, employee2, employee3, employee4, employee5, employee6) of the ***Employee*** class.

Create an array of objects named ***employees1*** which will consist of employee1, employee2, and employee3. Create another array of objects named ***employees2*** which will consist of employee4, and employee5, and employee6.

Instantiate two objects (**supervisor1**, **supervisor2**) of the ***Supervisor*** class such that the first three employees (employee1, employee2, employee3) will be supervised by **supervisor1** and rest of the employees (employee4, employee5, employee6) will be supervised by **supervisor2**.

Display the employee ID, first and last name of the employees supervised by each of the Supervisors. You also need to display the name of the Supervisor of each employee.

**Note:** In ***Objects.php,*** you must invoke ***getEmployees()*** function of the ***Supervisor*** class to retrieve the employees of the corresponding Supervisor and then invoke the required functions (i.e., getters) of the Employee class to display the properties of each employee.

Sample output for ***Objects.php*** is as follows:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | |  |  |  |  |  | |

# Task 2

Create Lab 7 submission folder ‘**Lab7**’ and copy **Arrays.php**, **Objects.php, Employee.php, Supervisor.php** and any other required files **(**e.g**. css file)** into this folder**.**Create a **Lab7.zip** file by compressing the '**Lab7'** folder.   
To hand in your lab go to Brightspace and navigate to Course Content 🡪 Labs.

Then click on ‘Lab 7 – Programming Arrays and Objects’ link.

Upload **Lab7.zip** on Brightspace.

**IMPORTANT NOTE**:

**You MUST demo the lab and explain the code to the Lab Professor to get marks.**