**Лабораторна робота №4**

**Об'єктно-орієнтоване програмування в РНР**

**Мета:** навчитися працювати з класами

Виконання лабораторної роботи:

**Завдання 1. (**Організація класів по каталогах в проєкті)

* Створіть пустий проєкт PHP.
* Створіть каталоги: "Models", "Controllers", "Views".
* У кожному каталозі створіть по одному класу, наприклад, "UserModel", "UserController", "UserView".
* В кожному класі реалізуйте просту функціональність, наприклад, виведення повідомлення чи повернення значень.

**Завдання 2. (**Автопідключення класів за допомогою spl\_autoload\_register. PHPDoc)

* Додайте PHPDoc коментарі до всіх класів, вказавши їх призначення та властивості.
* Створіть файл **autoload.php**, який буде містити функцію для автопідключення класів.
* Використайте **spl\_autoload\_register** для автоматичного підключення класів на основі їхніх імен та розташування.

**Завдання 3.** (Неймспейси)

* Додайте неймспейси до класів у попередньому завданні. Наприклад, "namespace Models;" для "UserModel".

Змініть файл **autoload.php** так, щоб він також враховував неймспейси при підключенні класів

**Завдання 4. (**Автопідключення класів з неймспейсами)

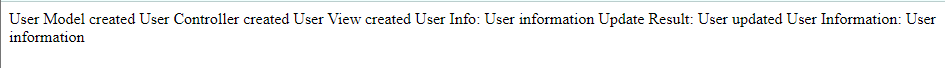
* Використовуйте аналогічний підхід до підключення класів, але тепер з урахуванням неймспейсів.
* Переконайтеся, що класи виводять повідомлення чи результати виклику.

*Виконання завданнь:*

<?php  
require\_once 'autoload.php';  
  
use Models\UserModel;  
use Controllers\UserController;  
use Views\UserView;  
  
// Створення об'єктів класів  
$userModel = new UserModel();  
$userController = new UserController();  
$userView = new UserView();  
  
// Використання функціоналу класів  
$userInfo = $userModel->getUserInfo();  
echo "User Info: $userInfo\n";  
  
$updateResult = $userController->updateUser();  
echo "Update Result: $updateResult\n";  
  
$userView->displayUser($userInfo);  
?>

</html>

Результат виконання завдання:



Зобр 4.1 Результат виконання 1-4 завданнь

**Завдання 5 (**Створення класу. Методи GET i SET)

1. Створіть клас **Circle** з полями: координати центру і радіус кола
2. Створіть конструктор, що приймає значення для 3-х полів
3. Створіть метод **\_\_toString()**, що повертає рядок в форматі: «Коло з центром в (х, у) і радіусом radius»
4. Створіть методи **GET** і **SET** для всіх 3-х полів
5. Створіть об’єкт та перевірте всі його методи

**Завдання 6** (Модифікатори доступу)

1. В класі з попереднього завдання зробіть всі поля private.
2. Створіть метод, що приймає об’єкт коло, і повертає **true**, якщо дані кола перетинаються, і **false**, якщо вони не перетинаються.

Виконання завдання:

<?php  
  
class Circle  
{  
 private $x;  
 private $y;  
 private $radius;  
  
 public function \_\_construct($x, $y, $radius)  
 {  
 $this->x = $x;  
 $this->y = $y;  
 $this->radius = $radius;  
 }  
  
 public function getX()  
 {  
 return $this->x;  
 }  
  
 public function setX($x)  
 {  
 $this->x = $x;  
 }  
  
 public function getY()  
 {  
 return $this->y;  
 }  
  
 public function setY($y)  
 {  
 $this->y = $y;  
 }  
  
 public function getRadius()  
 {  
 return $this->radius;  
 }  
  
 public function setRadius($radius)  
 {  
 $this->radius = $radius;  
 }  
  
 public function \_\_toString()  
 {  
 return "Коло з центром в ({$this->x}, {$this->y}) і радіусом {$this->radius}";  
 }  
  
 public function intersectsWith(Circle $circle)  
 {  
 $distance = sqrt(pow($this->x - $circle->getX(), 2) + pow($this->y - $circle->getY(), 2));  
 $sumOfRadius = $this->radius + $circle->getRadius();  
  
 return $distance <= $sumOfRadius;  
 }  
}  
  
// Перевірка роботи класу  
$circle1 = new Circle(0, 0, 5);  
$circle2 = new Circle(8, 8, 5);  
  
echo $circle1 . "\n";  
echo $circle2 . "\n";  
  
echo "Перетинаються? " . ($circle1->intersectsWith($circle2) ? "Так" : "Ні") . "\n";  
?>

Результат виконання завдання:



Зобр 4.2 Результат виконання 5-6 завданнь

**Завдання 7 (**Статичні властивості і методи)

1. Створіть директорію **text**, а в ній 3 текстових файла
2. Створіть клас зі статичним полем **dir=”text”**
3. Створіть 2 статичних методи в класі: на читання та запис в файл:

* Ім’я файлу передається як параметр метода.
* В метод «**на запис в файл**» передається ще й рядок, який потрібно дописати в файл.
* Директорія береться зі статичного поля

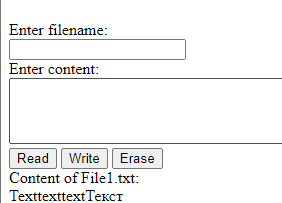
     4) Створіть метод, що дозволяє стерти вміст файлу

Перевірте роботу всіх методів

Виконання завдання:

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>File Manager</title>  
</head>  
<body>  
<h2>File Manager</h2>  
<form action="" method="post">  
 <label for="filename">Enter filename:</label><br>  
 <input type="text" id="filename" name="filename"><br>  
 <label for="content">Enter content:</label><br>  
 <textarea id="content" name="content" rows="4" cols="50"></textarea><br>  
 <input type="submit" value="Read" name="read">  
 <input type="submit" value="Write" name="write">  
 <input type="submit" value="Erase" name="erase">  
</form>  
  
<?php  
class FileManager  
{  
 public static *$dir* = "text";  
  
 public static function readFromFile($filename)  
 {  
 $filepath = self::*$dir* . '/' . $filename;  
 if (file\_exists($filepath)) {  
 $content = file\_get\_contents($filepath);  
 echo "Content of $filename: <br>$content <br>";  
 } else {  
 echo "File $filename does not exist <br>";  
 }  
 }  
  
 public static function writeToFile($filename, $content)  
 {  
 $filepath = self::$dir . '/' . $filename;  
 $file = fopen($filepath, "a"); // 'a' mode appends to the file  
 fwrite($file, $content);  
 fclose($file);  
 echo "Content written to $filename <br>";  
 }  
  
 public static function eraseFileContent($filename)  
 {  
 $filepath = self::$dir . '/' . $filename;  
 $file = fopen($filepath, "w"); // 'w' mode truncates the file  
 fclose($file);  
 echo "Content of $filename erased <br>";  
 }  
}  
  
// Create text directory if it doesn't exist  
if (!is\_dir(FileManager::$dir)) {  
 mkdir(FileManager::$dir);  
}  
  
if (isset($\_POST['read'])) {  
 $filename = $\_POST['filename'];  
 FileManager::readFromFile($filename);  
}  
  
if (isset($\_POST['write'])) {  
 $filename = $\_POST['filename'];  
 $content = $\_POST['content'];  
 FileManager::writeToFile($filename, $content);  
}  
  
if (isset($\_POST['erase'])) {  
 $filename = $\_POST['filename'];  
 FileManager::eraseFileContent($filename);  
}  
?>  
</body>  
</html>

Результат виконання завдання:



Зобр 4.3 Результат виконання сьомого завдання

**Завдання 8** (Наслідування)

1. Створіть клас **Human** з властивостями, що характеризують людину (зріст, маса, вік…). Створіть методи **GET** i **SET** для кожної властивості
2. Створіть клас **Student**, який успадковуватиметься від класу **Human**:
3. Додайте властивості, специфічні тільки для студента (назва ВНЗ, курс…)
4. Додайте в клас методи **GET** i **SET** для всіх нових властивостей.
5. Реалізуйте метод, який буде переводити студента на новий курс (тобто просто збільшувати значення поля «курс» на 1)
6. Створіть клас **Programmer**, який успадковуватиметься від класу **Human**:

* Додайте властивості, специфічні тільки для програміста (масив з мовами програмування, які він знає, досвід роботи…).
* Додайте в клас методи **GET** i **SET** для всіх нових властивостей.
* Реалізуйте метод, який буде додавати в масив з мовами ще одну мову.

Перевірте роботу всіх класів і всіх методів. Не забудьте змінити зріст і масу у студентів і програмістів, скориставшись методами з батьківського класу **Human**

**Завдання 9 (**Абстрактні класи)

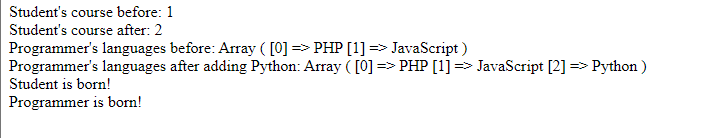
1. Зробіть клас **Human** абстрактним.
2. Напишіть метод «**Народження дитини**» в класі **Human**, що викликає метод «**Повідомлення при народженні дитини**» (не забудьте поставити модифікатор **protected**), який буде абстрактним
3. Реалізуйте «**Повідомлення при народженні дитини**» у класів **Student** та **Programmer**

Перевірте роботу методів «народження»

Виконання завдання:

<?php  
  
// Завдання 8 (Наслідування)  
  
class Human {  
 protected $height;  
 protected $weight;  
 protected $age;  
  
 public function \_\_construct($height, $weight, $age) {  
 $this->height = $height;  
 $this->weight = $weight;  
 $this->age = $age;  
 }  
  
  
 public function getHeight() {  
 return $this->height;  
 }  
  
 public function setHeight($height) {  
 $this->height = $height;  
 }  
  
 public function getWeight() {  
 return $this->weight;  
 }  
  
 public function setWeight($weight) {  
 $this->weight = $weight;  
 }  
  
 public function getAge() {  
 return $this->age;  
 }  
  
 public function setAge($age) {  
 $this->age = $age;  
 }  
}  
  
class Student extends Human {  
 private $university;  
 private $course;  
  
 public function \_\_construct($height, $weight, $age, $university, $course) {  
 parent::*\_\_construct*($height, $weight, $age);  
 $this->university = $university;  
 $this->course = $course;  
 }  
  
  
 public function getUniversity() {  
 return $this->university;  
 }  
  
 public function setUniversity($university) {  
 $this->university = $university;  
 }  
  
 public function getCourse() {  
 return $this->course;  
 }  
  
 public function setCourse($course) {  
 $this->course = $course;  
 }  
  
 public function moveToNextCourse() {  
 $this->course++;  
 }  
}  
  
class Programmer extends Human {  
 private $programmingLanguages = [];  
 private $experience;  
  
 public function \_\_construct($height, $weight, $age, $programmingLanguages, $experience) {  
 parent::*\_\_construct*($height, $weight, $age);  
 $this->programmingLanguages = $programmingLanguages;  
 $this->experience = $experience;  
 }  
  
  
 public function getProgrammingLanguages() {  
 return $this->programmingLanguages;  
 }  
  
 public function setProgrammingLanguages($programmingLanguages) {  
 $this->programmingLanguages = $programmingLanguages;  
 }  
  
 public function getExperience() {  
 return $this->experience;  
 }  
  
 public function setExperience($experience) {  
 $this->experience = $experience;  
 }  
  
 public function addProgrammingLanguage($language) {  
 $this->programmingLanguages[] = $language;  
 }  
}  
  
// Завдання 9 (Абстрактні класи)  
  
abstract class HumanAbstract {  
 abstract protected function birthMessage();  
  
 public function giveBirth() {  
 $this->birthMessage();  
 }  
}  
  
class StudentChild extends HumanAbstract {  
 protected function birthMessage() {  
 echo "Student is born!";  
 }  
}  
  
class ProgrammerChild extends HumanAbstract {  
 protected function birthMessage() {  
 echo "Programmer is born!";  
 }  
}  
  
// Перевірка роботи класів та методів  
  
$student = new Student(170, 60, 20, "Harvard", 1);  
$programmer = new Programmer(175, 70, 25, ["PHP", "JavaScript"], "5 years");  
  
echo "Student's course before: " . $student->getCourse() . "<br>";  
$student->moveToNextCourse();  
echo "Student's course after: " . $student->getCourse() . "<br>";  
  
echo "Programmer's languages before: ";  
print\_r($programmer->getProgrammingLanguages());  
echo "<br>";  
$programmer->addProgrammingLanguage("Python");  
echo "Programmer's languages after adding Python: ";  
print\_r($programmer->getProgrammingLanguages());  
echo "<br>";  
  
// Перевірка роботи методу "народження"  
$studentChild = new StudentChild();  
$programmerChild = new ProgrammerChild();  
  
$studentChild->giveBirth();  
echo "<br>";  
$programmerChild->giveBirth();  
  
?>

Результат виконання завдання:



Зобр 4.4 Результат виконання 8-9 завданнь

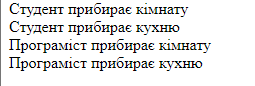
**Завдання 10 (**Інтерфейси)

1. Створіть інтерфейс «**Прибирання будинку**», в якому опишіть 2 методи: «**Прибирання кімнати**» і «**Прибирання кухні**»
2. Додайте створений інтерфейс в клас **Human**
3. Реалізуйте у кожному класі-спадкоємці (**Student** та **Programmer**) обидва методи
4. Реалізація повинна бути у вигляді одного з рядків: «**Студент прибирає кімнату**», «**Студент прибирає кухню**», «**Програміст прибирає кімнату**», «**Програміст прибирає кухню**»,
5. Перевірте роботу методів прибирання в обох класах

Виконання завдання:

<?php  
  
// Завдання 10 (Інтерфейси)  
  
// Інтерфейс "Прибирання будинку"  
interface HouseCleaning {  
 public function cleanRoom();  
 public function cleanKitchen();  
}  
  
// Клас Human  
class Human {  
 // Код класу Human може бути порожнім або містити загальні властивості та методи для всіх людей  
}  
  
// Клас Student успадковує Human та реалізує інтерфейс HouseCleaning  
class Student extends Human implements HouseCleaning {  
 // Реалізація методів інтерфейсу HouseCleaning для студента  
 public function cleanRoom() {  
 echo "Студент прибирає кімнату<br>";  
 }  
  
 public function cleanKitchen() {  
 echo "Студент прибирає кухню<br>";  
 }  
}  
  
// Клас Programmer також успадковує Human та реалізує інтерфейс HouseCleaning  
class Programmer extends Human implements HouseCleaning {  
 // Реалізація методів інтерфейсу HouseCleaning для програміста  
 public function cleanRoom() {  
 echo "Програміст прибирає кімнату<br>";  
 }  
  
 public function cleanKitchen() {  
 echo "Програміст прибирає кухню<br>";  
 }  
}  
  
// Перевірка роботи методів прибирання в обох класах  
$student = new Student();  
$programmer = new Programmer();  
  
$student->cleanRoom(); // Студент прибирає кімнату  
$student->cleanKitchen(); // Студент прибирає кухню  
  
$programmer->cleanRoom(); // Програміст прибирає кімнату  
$programmer->cleanKitchen(); // Програміст прибирає кухню  
  
?>

Результат виконання завдання:



Зобр 4.5 Результат виконання десятого завдання

Висновок: В ході виконання цієї лабораторної роботи я вдосконалив свої навички роботи з середовищем програмування php за допомогою практики над cookie? Session та файлами.