**Лабораторна робота №4**

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

***Мета:*** навчитися працювати з класами

**Хід роботи:**

Завдання 0. Ретельно вивчити теорію за посиланнями:

https://www.w3schools.com/php/default.asp:

PHP OOP Classes and Objects

PHP OOP Constructor

PHP OOP Destructor

PHP OOP Access Modifiers

PHP OOP Inheritance

PHP OOP Class Constants

PHP OOP Abstract Classes

PHP OOP Interfaces

PHP OOP Static Methods

PHP OOP Static Properties

PHP Namespaces

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.

Завдання 5 (Створення класу. Методи GET i SET)

Створіть клас Circle з полями: координати центру і радіус кола

Створіть конструктор, що приймає значення для 3-х полів

Створіть метод \_\_toString(), що повертає рядок в форматі: «Коло з центром в (х, у) і радіусом radius»

Створіть методи GET і SET для всіх 3-х полів

Створіть об’єкт та перевірте всі його методи

Завдання 6 (Модифікатори доступу)

В класі з попереднього завдання зробіть всі поля private.

Створіть метод, що приймає об’єкт коло, і повертає true, якщо дані кола перетинаються, і false, якщо вони не перетинаються.

Завдання 7 (Статичні властивості і методи)

Створіть директорію text, а в ній 3 текстових файла

Створіть клас зі статичним полем dir=”text”

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

Завдання 8 (Наслідування)

Створіть клас Human з властивостями, що характеризують людину (зріст, маса, вік…). Створіть методи GET i SET для кожної властивості

Створіть клас Student, який успадковуватиметься від класу Human:

Додайте властивості, специфічні тільки для студента (назва ВНЗ, курс…)

Додайте в клас методи GET i SET для всіх нових властивостей.

Реалізуйте метод, який буде переводити студента на новий курс (тобто просто збільшувати значення поля «курс» на 1)

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)

Зробіть клас Human абстрактним.

Напишіть метод «Народження дитини» в класі Human, що викликає метод «Повідомлення при народженні дитини» (не забудьте поставити модифікатор protected), який буде абстрактним

Реалізуйте «Повідомлення при народженні дитини» у класів Student та Programmer

Перевірте роботу методів «народження»

Завдання 10 (Інтерфейси)

Створіть інтерфейс «Прибирання будинку», в якому опишіть 2 методи: «Прибирання кімнати» і «Прибирання кухні»

Додайте створений інтерфейс в клас Human

Реалізуйте у кожному класі-спадкоємці (Student та Programmer) обидва методи

Реалізація повинна бути у вигляді одного з рядків: «Студент прибирає кімнату», «Студент прибирає кухню», «Програміст прибирає кімнату», «Програміст прибирає кухню»,

Перевірте роботу методів прибирання в обох класах

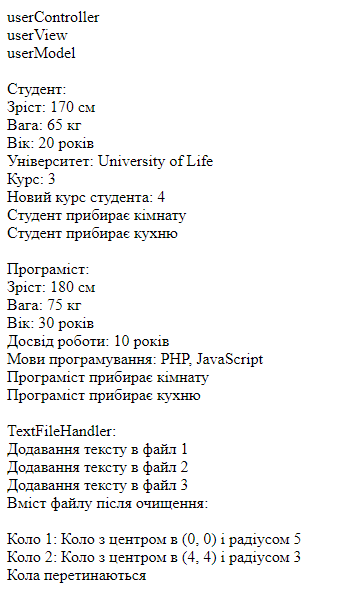


Рис 1. Результат перевірки роботи усіх класів та їх методів

**Controllers/UserConroller.php**

<?php  
  
namespace Controllers;  
  
*/\*\*  
 \* Class UserController  
 \** ***@package*** *Controllers  
 \*/*class UserController  
{  
 */\*\*  
 \* Функція виведення повідомлення  
 \** ***@param*** *string $message Повідомлення для виведення  
 \*/* public function showMessage($message) {  
 echo $message;  
 }  
}

**Handlers/TextFileHanler.php**

<?php  
  
namespace Handlers;  
*/\*\*  
 \* Клас для роботи з текстовими файлами  
 \*/*class TextFileHandler  
{  
 */\*\** ***@var*** *string $dir Директорія для файлів \*/* public static *$dir* = "text/";  
  
 */\*\*  
 \* Метод для читання вмісту файлу  
 \** ***@param*** *string $filename Назва файлу для читання  
 \** ***@return*** *string|null Вміст файлу або null, якщо файл не знайдено  
 \*/* public static function readFile($filename)  
 {  
 $file\_path = self::*$dir* . $filename;  
 if (file\_exists($file\_path)) {  
 return file\_get\_contents($file\_path);  
 }  
 return null;  
 }  
  
 */\*\*  
 \* Метод для запису в файл  
 \** ***@param*** *string $filename Назва файлу для запису  
 \** ***@param*** *string $content Зміст для запису  
 \** ***@return*** *bool true, якщо запис успішний, інакше false  
 \*/* public static function writeFile($filename, $content)  
 {  
 $file\_path = self::*$dir* . $filename;  
 return file\_put\_contents($file\_path, $content, *FILE\_APPEND*) !== false;  
 }  
  
 */\*\*  
 \* Метод для очищення вмісту файлу  
 \** ***@param*** *string $filename Назва файлу для очищення  
 \** ***@return*** *bool true, якщо очищення успішне, інакше false  
 \*/* public static function clearFile($filename)  
 {  
 $file\_path = self::*$dir* . $filename;  
 return file\_put\_contents($file\_path, '') !== false;  
 }  
}

**Interfaces/HouseCleaning.php**

<?php  
  
namespace Interfaces;  
*/\*\*  
 \* Інтерфейс для прибирання будинку  
 \*/*interface HouseCleaning  
{  
 */\*\*  
 \* Метод для прибирання кімнати  
 \** ***@return*** *string Рядок з результатом прибирання кімнати*

*\*/* public function cleanRoom();  
  
 */\*\*  
 \* Метод для прибирання кухні  
 \** ***@return*** *string Рядок з результатом прибирання кухні  
 \*/* public function cleanKitchen();  
}

**Models/Circle.php**

<?php  
  
namespace Models;  
*/\*\*  
 \* Клас Circle для роботи з колом  
 \*/*class Circle  
{  
 */\*\** ***@var*** *float $x Координата X центра кола \*/* private $x;  
 */\*\** ***@var*** *float $y Координата Y центра кола \*/* private $y;  
 */\*\** ***@var*** *float $radius Радіус кола \*/* private $radius;  
  
 */\*\*  
 \* Конструктор класу Circle  
 \** ***@param*** *float $x Координата X центра кола  
 \** ***@param*** *float $y Координата Y центра кола  
 \** ***@param*** *float $radius Радіус кола  
 \*/* public function \_\_construct($x, $y, $radius)  
 {  
 $this->x = $x;  
 $this->y = $y;  
 $this->radius = $radius;  
 }  
  
 */\*\*  
 \* Повертає рядок з описом кола  
 \** ***@return*** *string Рядок з описом кола  
 \*/* public function \_\_toString()  
 {  
 return "Коло з центром в ($this->x, $this->y) і радіусом $this->radius";  
 }  
  
 */\*\*  
 \* Отримує значення координати X центра кола  
 \** ***@return*** *float Координата X центра кола  
 \*/* public function getX()  
 {  
 return $this->x;  
 }  
  
 */\*\*  
 \* Встановлює значення координати X центра кола  
 \** ***@param*** *float $x Нове значення для координати X центра кола  
 \*/*

public function setX($x)  
 {  
 $this->x = $x;  
 }  
  
 */\*\*  
 \* Отримує значення координати Y центра кола  
 \** ***@return*** *float Координата Y центра кола  
 \*/* public function getY()  
 {  
 return $this->y;  
 }  
  
 */\*\*  
 \* Встановлює значення координати Y центра кола  
 \** ***@param*** *float $y Нове значення для координати Y центра кола  
 \*/* public function setY($y)  
 {  
 $this->y = $y;  
 }  
  
 */\*\*  
 \* Отримує значення радіуса R кола  
 \** ***@return*** *float Радіус R кола  
 \*/* public function getRadius()  
 {  
 return $this->radius;  
 }  
  
 */\*\*  
 \* Встановлює значення радіуса R кола  
 \** ***@param*** *float $r Нове значення радіуса R кола  
 \*/* public function setRadius($r)  
 {  
 $this->radius = $r;  
 }  
  
 */\*\*  
 \* Перевіряє, чи перетинаються два кола  
 \** ***@param*** *Circle $circle Інше коло для порівняння  
 \** ***@return*** *bool true, якщо кола перетинаються, інакше false  
 \*/* public function intersects(Circle $circle)  
 {  
 $distance = sqrt(pow($this->x - $circle->getX(), 2) + pow($this->y - $circle->getY(), 2));  
 return $distance < ($this->radius + $circle->getRadius());  
 }  
  
}

**Models/Human.php**

<?php  
  
namespace Models;  
*/\*\*  
 \* Абстрактний клас для людини  
 \*/*abstract class Human

{  
 */\*\** ***@var*** *float $height Зріст людини \*/* protected $height;  
 */\*\** ***@var*** *float $weight Вага людини \*/* protected $weight;  
 */\*\** ***@var*** *int $age Вік людини \*/* protected $age;  
  
 */\*\*  
 \* Отримує значення зросту  
 \** ***@return*** *float Зріст людини  
 \*/* public function getHeight()  
 {  
 return $this->height;  
 }  
  
 */\*\*  
 \* Встановлює значення зросту  
 \** ***@param*** *float $height Нове значення для зросту  
 \*/* public function setHeight($height)  
 {  
 $this->height = $height;  
 }  
  
 */\*\*  
 \* Отримує значення ваги  
 \** ***@return*** *float Вага людини  
 \*/* public function getWeight()  
 {  
 return $this->weight;  
 }  
  
 */\*\*  
 \* Встановлює значення ваги  
 \** ***@param*** *float $weight Нове значення для ваги  
 \*/* public function setWeight($weight)  
 {  
 $this->weight = $weight;  
 }  
  
 */\*\*  
 \* Отримує значення віку  
 \** ***@return*** *int Вік людини  
 \*/* public function getAge()  
 {  
 return $this->age;  
 }  
  
 */\*\*  
 \* Встановлює значення віку  
 \** ***@param*** *int $age Нове значення для віку  
 \*/* public function setAge($age)  
 {  
 $this->age = $age;  
 }  
}

**Models/Programmer.php**

<?php  
  
namespace Models;  
  
use Interfaces\HouseCleaning;  
  
*/\*\*  
 \* Клас програміста, який успадковує клас Human і реалізує інтерфейс HouseCleaning  
 \*/*class Programmer extends Human implements HouseCleaning  
{  
 */\*\** ***@var*** *array $programmingLanguages Масив з мовами програмування \*/* private $programmingLanguages;  
 */\*\** ***@var*** *int $experience Досвід роботи \*/* private $experience;  
  
 */\*\*  
 \* Отримує масив з мовами програмування  
 \** ***@return*** *array Масив з мовами програмування  
 \*/* public function getProgrammingLanguages()  
 {  
 return $this->programmingLanguages;  
 }  
  
 */\*\*  
 \* Додає нову мову програмування до масиву  
 \** ***@param*** *string $language Нова мова програмування  
 \*/* public function addProgrammingLanguage($language)  
 {  
 $this->programmingLanguages[] = $language;  
 }  
  
 */\*\*  
 \* Отримує значення досвіду роботи  
 \** ***@return*** *int Досвід роботи  
 \*/* public function getExperience()  
 {  
 return $this->experience;  
 }  
  
 */\*\*  
 \* Встановлює значення досвіду роботи  
 \** ***@param*** *int $experience Нове значення для досвіду роботи  
 \*/* public function setExperience($experience)  
 {  
 $this->experience = $experience;  
 }  
  
 */\*\*  
 \* Реалізація методу прибирання кімнати з інтерфейсу HouseCleaning  
 \** ***@return*** *string Рядок з результатом прибирання кімнати  
 \*/* public function cleanRoom()  
 {  
 return "Програміст прибирає кімнату";  
 }

*/\*\*  
 \* Реалізація методу прибирання кухні з інтерфейсу HouseCleaning  
 \** ***@return*** *string Рядок з результатом прибирання кухні  
 \*/* public function cleanKitchen()  
 {  
 return "Програміст прибирає кухню";  
 }  
}

**Models/Student.php**

<?php  
  
namespace Models;  
  
use Interfaces\HouseCleaning;  
  
*/\*\*  
 \* Клас студента, який успадковує клас Human і реалізує інтерфейс HouseCleaning  
 \*/*class Student extends Human implements HouseCleaning  
{  
 */\*\** ***@var*** *string $university Назва ВНЗ \*/* private $university;  
 */\*\** ***@var*** *int $course Курс студента \*/* private $course;  
  
 */\*\*  
 \* Отримує значення назви ВНЗ  
 \** ***@return*** *string Назва ВНЗ  
 \*/* public function getUniversity()  
 {  
 return $this->university;  
 }  
  
 */\*\*  
 \* Встановлює значення назви ВНЗ  
 \** ***@param*** *string $university Нове значення для назви ВНЗ  
 \*/* public function setUniversity($university)  
 {  
 $this->university = $university;  
 }  
  
 */\*\*  
 \* Отримує значення курсу  
 \** ***@return*** *int Курс студента  
 \*/* public function getCourse()  
 {  
 return $this->course;  
 }  
  
 */\*\*  
 \* Встановлює значення курсу  
 \** ***@param*** *int $course Нове значення для курсу  
 \*/* public function setCourse($course)  
 {  
 $this->course = $course;  
 }  
  
 */\*\*  
 \* Метод для переведення студента на новий курс  
 \*/* public function moveToNextCourse()  
 {  
 $this->course++;  
 }  
  
 */\*\*  
 \* Реалізація методу прибирання кімнати з інтерфейсу HouseCleaning  
 \** ***@return*** *string Рядок з результатом прибирання кімнати  
 \*/* public function cleanRoom()  
 {  
 return "Студент прибирає кімнату";  
 }  
  
 */\*\*  
 \* Реалізація методу прибирання кухні з інтерфейсу HouseCleaning  
 \** ***@return*** *string Рядок з результатом прибирання кухні  
 \*/* public function cleanKitchen()  
 {  
 return "Студент прибирає кухню";  
 }  
}

**Models/UserModel.php**

<?php  
  
namespace Models;  
  
*/\*\*  
 \* Class UserModel  
 \** ***@package*** *Models  
 \*/*class UserModel  
{  
 */\*\*  
 \* Функція виведення повідомлення  
 \** ***@param*** *string $message Повідомлення для виведення  
 \*/* public function showMessage($message) {  
 echo $message;  
 }  
}

**Views/UserView.php**

<?php  
  
namespace Views;  
  
*/\*\*  
 \* Class UserView  
 \** ***@package*** *Views  
 \*/*class UserView  
{  
 */\*\*  
 \* Функція виведення повідомлення  
 \** ***@param*** *string $message Повідомлення для виведення  
 \*/* public function showMessage($message) {  
 echo $message;  
 }  
}

**autoload.php**

<?php  
*/\*\*  
 \* Функція для автопідключення класів за їх іменем та розташуванням  
 \** ***@param*** *string $class\_name Ім'я класу для підключення  
 \*/*function autoloadClasses($class\_name) {  
 $class\_file = *\_\_DIR\_\_* . '/' . str\_replace('\\', '/', $class\_name) . '.php';  
 if (file\_exists($class\_file)) {  
 include $class\_file;  
 }  
}  
  
spl\_autoload\_register('autoloadClasses');

**index.php**

<?php  
  
require\_once 'autoload.php';  
  
  
use Models\Human;  
use Models\Student;  
use Models\Programmer;  
use Models\Circle;  
use Models\UserModel;  
use Controllers\UserController;  
use Views\UserView;  
  
$userController = new UserController();  
$userView = new UserView();  
$userModel = new UserModel();  
  
echo $userController->showMessage("userController");  
echo "<br>";  
echo $userView->showMessage("userView");  
echo "<br>";  
echo $userModel->showMessage("userModel");  
  
$student = new Student();  
$student->setHeight(170);  
$student->setWeight(65);  
$student->setAge(20);  
$student->setUniversity("University of Life");  
$student->setCourse(3);  
  
echo "<br><br>Студент:<br>";  
echo "Зріст: " . $student->getHeight() . " см<br>";  
echo "Вага: " . $student->getWeight() . " кг<br>";  
echo "Вік: " . $student->getAge() . " років<br>";  
echo "Університет: " . $student->getUniversity() . "<br>";  
echo "Курс: " . $student->getCourse() . "<br>";  
$student->moveToNextCourse();  
echo "Новий курс студента: " . $student->getCourse() . "<br>";  
echo $student->cleanRoom() . "<br>";  
echo $student->cleanKitchen() . "<br>";  
  
$programmer = new Programmer();  
$programmer->setHeight(180);  
$programmer->setWeight(75);  
$programmer->setAge(30);  
$programmer->setExperience(10);  
$programmer->addProgrammingLanguage("PHP");  
$programmer->addProgrammingLanguage("JavaScript");  
  
echo "<br>Програміст:<br>";  
echo "Зріст: " . $programmer->getHeight() . " см<br>";  
echo "Вага: " . $programmer->getWeight() . " кг<br>";  
echo "Вік: " . $programmer->getAge() . " років<br>";  
echo "Досвід роботи: " . $programmer->getExperience() . " років<br>";  
echo "Мови програмування: " . implode(", ", $programmer->getProgrammingLanguages()) . "<br>";  
echo $programmer->cleanRoom() . "<br>";  
echo $programmer->cleanKitchen() . "<br>";  
  
require\_once 'Handlers/TextFileHandler.php';  
  
use Handlers\TextFileHandler;  
  
echo "<br>TextFileHandler:<br>";  
TextFileHandler::*writeFile*('file1.txt', 'Додавання тексту в файл 1');  
TextFileHandler::*writeFile*('file2.txt', 'Додавання тексту в файл 2');  
TextFileHandler::*writeFile*('file3.txt', 'Додавання тексту в файл 3');  
  
echo TextFileHandler::*readFile*('file1.txt') . "<br>";  
echo TextFileHandler::*readFile*('file2.txt') . "<br>";  
echo TextFileHandler::*readFile*('file3.txt') . "<br>";  
  
TextFileHandler::*clearFile*('file1.txt');  
TextFileHandler::*clearFile*('file2.txt');  
TextFileHandler::*clearFile*('file3.txt');  
echo "Вміст файлу після очищення: " . TextFileHandler::*readFile*('file1.txt') . "<br>";  
  
$circle1 = new Circle(0, 0, 5);  
$circle2 = new Circle(4, 4, 3);  
  
echo "<br>Коло 1: " . $circle1 . "<br>";  
echo "Коло 2: " . $circle2 . "<br>";  
  
if ($circle1->intersects($circle2)) {  
 echo "Кола перетинаються<br>";  
} else {  
 echo "Кола не перетинаються<br>";  
}

***Висновки:*** ми навчилися працювати з класами