**Лабораторна робота №4**

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

**Мета роботи:** навчитися працювати з класами

Хід роботи:

Завдання 1-4.

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

Лістинг:

<?php

require\_once 'autoload.php';

use Models\PersonModel;

use Controllers\PersonController;

use Views\PersonView;

$model = new PersonModel("Hello from Roman!");

$controller = new PersonController($model);

$view = new PersonView();

$output = $controller->process();

$view->render($output);

<?php

spl\_autoload\_register(function ($class) {

    $prefix = '';

    $base\_dir = \_\_DIR\_\_ . '/';

    $class = str\_replace('\\', '/', $class);

    $file = $base\_dir . $class . '.php';

    $file = str\_replace('Models/', 'Models/', $file);

    $file = str\_replace('Controllers/', 'Controllers/', $file);

    $file = str\_replace('Views/', 'Views/', $file);

    if (file\_exists($file)) {

        require\_once $file;

    }

});

<?php

namespace Controllers;

use Models\PersonModel;

/\*\*

 \* Class PersonController

 \* Контролер для обробки логіки користувача

 \*/

class PersonController {

    private PersonModel $model;

    public function \_\_construct(PersonModel $model) {

        $this->model = $model;

    }

    public function process(): string {

        return "Ім'я з контролера: " . $this->model->getName();

    }

}

<?php

namespace Controllers;

/\*\*

 \* UserController відповідає за обробку запитів користувачів

 \*/

class UserControllers

{

    public function getMessage()

    {

        return "Hello from UserController!";

    }

}

<?php

namespace Models;

/\*\*

 \* Class PersonModel

 \* Модель для роботи з особистими даними користувача

 \*/

class PersonModel {

    private string $name;

    public function \_\_construct(string $name = "Невідомий") {

        $this->name = $name;

    }

    public function getName(): string {

        return $this->name;

    }

}

<?php

namespace Models;

/\*\*

 \* UserModel відповідає за роботу з даними користувачів

 \*/

class UserModels

{

    public function getMessage()

    {

        return "Hello from UserModel!";

    }

}

<?php

namespace Views;

/\*\*

 \* Class PersonView

 \* Представлення для виводу інформації

 \*/

class PersonView {

    public function render(string $content): void {

        echo "<h3>Вивід з View:</h3>";

        echo "<p>$content</p>";

    }

}

<?php

namespace Views;

/\*\*

 \* UserView відповідає за відображення даних користувачів

 \*/

class UserViews

{

    public function getMessage()

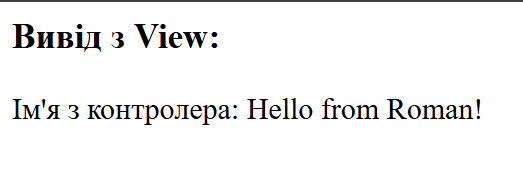
    {

        return "Hello from UserView!";

    }

}

Результат виконання:



Завдання 5-6

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

Лістинг:

<?php

require\_once 'classes/Circle.php';

$circle1 = new Circle(2, 5, 1);

$circle2 = new Circle(3, 5, 1);

if ($circle1->intersects($circle2)) {

    echo "Кола перетинаються";

} else {

    echo "Кола не перетинаються";

}

?>

<?php

/\*\*

 \* Class Circle

 \* Представляє коло з координатами центру і радіусом

 \*/

class Circle {

    private float $x;

    private float $y;

    private float $radius;

    public function \_\_construct(float $x, float $y, float $radius) {

        $this->x = $x;

        $this->y = $y;

        $this->radius = $radius;

    }

    public function getX(): float {

        return $this->x;

    }

    public function getY(): float {

        return $this->y;

    }

    public function getRadius(): float {

        return $this->radius;

    }

    public function setX(float $x): void {

        $this->x = $x;

    }

    public function setY(float $y): void {

        $this->y = $y;

    }

    public function setRadius(float $radius): void {

        $this->radius = $radius;

    }

    public function \_\_toString(): string {

        return "Коло з центром в ({$this->x}, {$this->y}) і радіусом {$this->radius}";

    }

    public function intersects(Circle $other): bool {

        $dx = $this->x - $other->getX();

        $dy = $this->y - $other->getY();

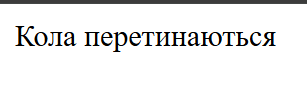
        $distance = sqrt($dx \*\* 2 + $dy \*\* 2);

        return $distance < ($this->radius + $other->getRadius());

    }

}

Результат виконання:



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
* Створіть метод, що дозволяє стерти вміст файлу
* Перевірте роботу всіх методів

Лістинг:

<?php

require\_once 'classes/Text.php';

?>

<html lang="en">

<body>

<form action="" method="POST">

    <table border="1" cellpadding="10">

        <tr>

            <td>Оберіть файл для читання:

                <select name="readFile">

                    <option value="text1.txt">text1.txt</option>

                    <option value="text2.txt">text2.txt</option>

                    <option value="text3.txt">text3.txt</option>

                </select>

                <input type="submit" name="read" value="Прочитати">

            </td>

        </tr>

        <tr>

            <td>Запис у файл:

                <select name="writeFile">

                    <option value="text1.txt">text1.txt</option>

                    <option value="text2.txt">text2.txt</option>

                    <option value="text3.txt">text3.txt</option>

                </select>

                <input type="text" name="text" placeholder="Введіть текст">

                <input type="submit" name="write" value="Записати">

            </td>

        </tr>

        <tr>

            <td>Очистити файл:

                <select name="clearFile">

                    <option value="text1.txt">text1.txt</option>

                    <option value="text2.txt">text2.txt</option>

                    <option value="text3.txt">text3.txt</option>

                </select>

                <input type="submit" name="clear" value="Очистити">

            </td>

        </tr>

    </table>

</form>

<?php

if ($\_SERVER['REQUEST\_METHOD'] == "POST") {

    if (isset($\_POST['read'])) {

        $file = $\_POST['readFile'];

        echo "<h3>$file</h3><pre>" . Text::readFromFile($file) . "</pre>";

    } elseif (isset($\_POST['write'])) {

        $file = $\_POST['writeFile'];

        $text = $\_POST['text'];

        Text::writeToFile($file, $text);

        echo "<p>Текст записано в $file</p>";

    } elseif (isset($\_POST['clear'])) {

        $file = $\_POST['clearFile'];

        Text::clearFile($file);

        echo "<p>Файл $file очищено</p>";

    }

}

?>

</body>

</html>

<?php

/\*\*

 \* Class Text

 \* Статичний клас для роботи з текстовими файлами у папці "text"

 \*/

class Text {

    public static string $dir = "text";

    public static function writeToFile(string $filename, string $text): void {

        $path = self::$dir . "/" . $filename;

        file\_put\_contents($path, $text . PHP\_EOL, FILE\_APPEND);

    }

    public static function readFromFile(string $filename): string {

        $path = self::$dir . "/" . $filename;

        return file\_exists($path) ? file\_get\_contents($path) : "Файл не знайдено.";

    }

    public static function clearFile(string $filename): void {

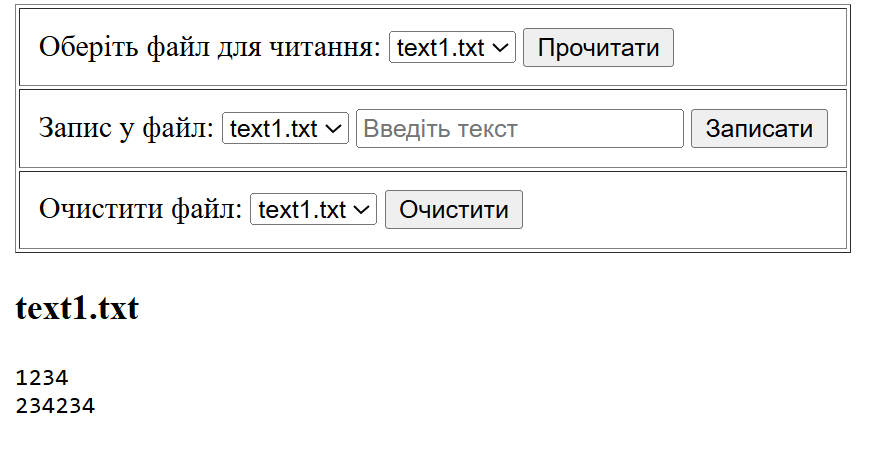
        $path = self::$dir . "/" . $filename;

        file\_put\_contents($path, "");

    }

}

Результат виконання:



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

Лістинг:

<?php

spl\_autoload\_register(function ($class) {

    $path = "classes/" . $class . ".php";

    if (file\_exists($path)) {

        require\_once $path;

    }

});

$programmer = new Programmer("Oleksander", 180, 90, 19, ["C#", "C++", "PHP"], 3);

$student = new Student("Roman", 160, 80, 21, "Zhytomyr Polytechnic", 2);

echo "<h3>Програміст:</h3>";

echo $programmer->getInfo();

$programmer->addLanguage("Python");

echo $programmer->getInfo();

echo "<h3>Студент:</h3>";

echo $student->getInfo();

$student->setHeight(165);

echo $student->getInfo();

<?php

/\*\*

 \* Class Human

 \* Базовий клас людини

 \*/

class Human {

    protected string $name;

    protected float $height;

    protected float $weight;

    protected int $age;

    public function \_\_construct(string $name, float $height, float $weight, int $age) {

        $this->name = $name;

        $this->height = $height;

        $this->weight = $weight;

        $this->age = $age;

    }

    public function getHeight(): float { return $this->height; }

    public function getWeight(): float { return $this->weight; }

    public function getAge(): int { return $this->age; }

    public function setHeight(float $h): void { $this->height = $h; }

    public function setWeight(float $w): void { $this->weight = $w; }

    public function setAge(int $a): void { $this->age = $a; }

    public function getName(): string { return $this->name; }

}

<?php

class Programmer extends Human {

    private array $languages;

    private int $experience;

    public function \_\_construct(string $name, float $height, float $weight, int $age, array $languages, int $experience) {

       parent::\_\_construct($name, $height, $weight, $age);

        $this->languages = $languages;

        $this->experience = $experience;

    }

    public function addLanguage(string $lang): void {

        if (!in\_array($lang, $this->languages)) {

            $this->languages[] = $lang;

        }

    }

    public function getInfo(): string {

        return "<p>Ім'я: {$this->name} | Вік: {$this->age} | Мови: " . implode(", ", $this->languages) . " | Досвід: {$this->experience} років</p>";

    }

}

<?php

class Student extends Human {

    private string $university;

    private int $course;

    public function \_\_construct(string $name, float $height, float $weight, int $age, string $university, int $course) {

        parent::\_\_construct($name, $height, $weight, $age);

        $this->name = $name;

        $this->university = $university;

        $this->course = $course;

    }

    public function promote(): void {

        $this->course++;

    }

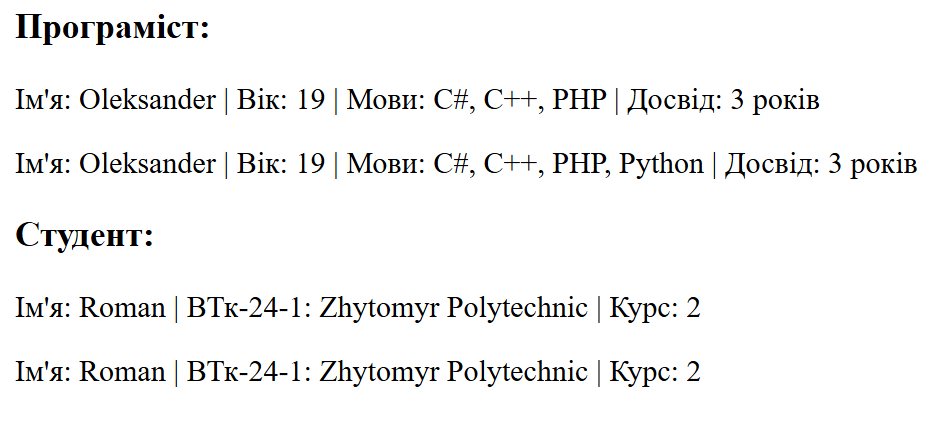
    public function getInfo(): string {

        return "<p>Ім'я: {$this->name} | ВТк-24-1: {$this->university} | Курс: {$this->course}</p>";

    }

}

Результат виконання:



Завдання 9-10

* 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»
* 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

Лістинг:

<?php

spl\_autoload\_register(function ($class) {

    $class = str\_replace('\\', '/', $class);

    $path = \_\_DIR\_\_ . '/classes/' . basename($class) . '.php';

    if (file\_exists($path)) {

        require\_once $path;

    }

});

use Classes\Programmer;

use Classes\Student;

$programmer = new Programmer("Roman", 180, 90, 19, ["C#", "C++", "PHP"], 3);

$student = new Student("Oleksander", 160, 80, 21, "Zhytomyr Polytechnic", 2);

echo "<h3>Програміст:</h3>";

echo $programmer->getInfo();

echo $programmer->cleanKitchen();

echo $programmer->cleanRoom();

echo "<h3>Студент:</h3>";

echo $student->getInfo();

echo $student->cleanKitchen();

echo $student->cleanRoom();

<?php

namespace Classes;

require\_once 'ICleanHouse.php';

abstract class Human implements ICleanHouse {

    protected string $name;

    protected float $height;

    protected float $weight;

    protected int $age;

    public function \_\_construct(string $name, float $height, float $weight, int $age) {

        $this->name = $name;

        $this->height = $height;

        $this->weight = $weight;

        $this->age = $age;

    }

    public function giveBirth(): string {

        return $this->onBirthMessage();

    }

    abstract protected function onBirthMessage(): string;

}

<?php

namespace Classes;

interface ICleanHouse {

    public function cleanRoom(): string;

    public function cleanKitchen(): string;

}

<?php

namespace Classes;

require\_once 'Human.php';

class Programmer extends Human {

    protected array $languages;

    protected int $experience;

    public function \_\_construct(string $name, float $height, float $weight, int $age, array $languages, int $experience) {

        parent::\_\_construct($name, $height, $weight, $age);

        $this->languages = $languages;

        $this->experience = $experience;

    }

    protected function onBirthMessage(): string {

        return "Народився програміст: {$this->name}";

    }

    public function cleanRoom(): string {

        return "<p>Програміст прибирає кімнату</p>";

    }

    public function cleanKitchen(): string {

        return "<p>Програміст прибирає кухню</p>";

    }

    public function getInfo(): string {

        $langs = implode(", ", $this->languages);

        return "<p>Ім'я: {$this->name}, Мови: $langs, Досвід: {$this->experience} років</p>";

    }

}

<?php

namespace Classes;

require\_once 'Human.php';

class Student extends Human {

    protected string $university;

    protected int $course;

    public function \_\_construct(string $name, float $height, float $weight, int $age, string $university, int $course) {

        parent::\_\_construct($name, $height, $weight, $age);

        $this->university = $university;

        $this->course = $course;

    }

    protected function onBirthMessage(): string {

        return "Народився студент: {$this->name}";

    }

    public function cleanRoom(): string {

        return "<p>Студент прибирає кімнату</p>";

    }

    public function cleanKitchen(): string {

        return "<p>Студент прибирає кухню</p>";

    }

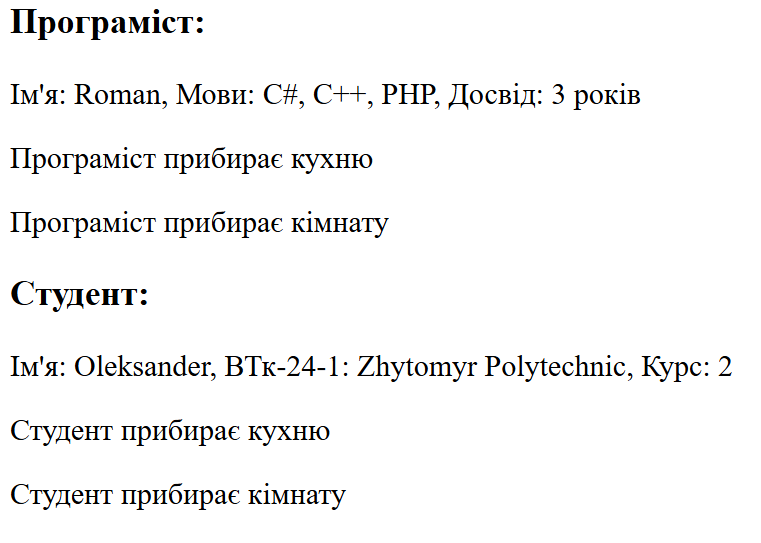
    public function getInfo(): string {

        return "<p>Ім'я: {$this->name}, ВТк-24-1: {$this->university}, Курс: {$this->course}</p>";

    }

}

Виконання програми:



**Висновок :** В результаті роботи вдалося освоїти принципи інкапсуляції, наслідування та поліморфізму, які є основними складовими ООП. Робота з класами допомогла зрозуміти, як правильно організовувати код для створення більш гнучких, масштабованих і підтримуваних програм, що значно спрощує процес розробки та супроводу програмних проектів.