Лабораторна робота №4

Тема: Об'єктно-орієнтоване програмування в PHP

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.

Лістинг (UserController):

```php
<?php
/**
 * Class UserController
 * Контролер, що обробляє логіку користувача.
 * @author Oleg
 */
namespace Controllers;

class UserController {
    public function showMessage() {
        echo "user controller";
    }
}
```

Лістинг (UserModel):

```php
<?php
/**
 * Class UserModel
 * Модель користувача, що містить дані користувача.
 */
namespace Models;

class UserModel {
    public function getUserName() {
        return "user model";
    }
}
```

Лістинг (UserView):

```php
<?php
/**
 * Class UserView
 * Відображає інформацію користувача.
 */
namespace Views;

class UserView {
    public function render() {
        echo "user view";
    }
```

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.

PHPDoc коментарі до класів можна побачити в попередньому коді

Лістинг (autoload):

```php
<?php
spl_autoload_register(function ($class) { //урахування неймспейсу
    $class = str_replace('\\', DIRECTORY_SEPARATOR, $class);
    $path = __DIR__ . '/' . $class . '.php';
    if (file_exists($path)) {
        require_once $path;
    }
});
//автоматичне підключення класу без ручного пропису require
```

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.

Неймспейси можна побачити в лістингу першого завдання, модифікований autoload також знаходиться в попередньому лістингу

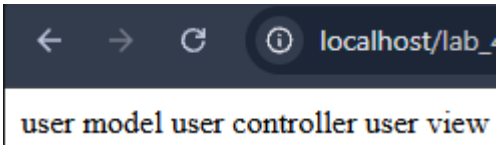

Рис. 1. Результат виконання завдання

Завдання 5 (Створення класу. Методи GET і 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 (x, y) 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.

Лістинг (circle):

```php
<?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 __toString() { //цей метод викликається, коли об'єкт намагаються вивести як рядок:
        return "Коло з центром в ($this->x, $this->y) і радіусом $this->radius";
```

```
 }

    public function getX() { return $this->x; }//отримання
    public function getY() { return $this->y; }
    public function getRadius() { return $this->radius; }

    public function setX($x) { $this->x = $x; }// заміна
    public function setY($y) { $this->y = $y; }
    public function setRadius($radius) { $this->radius = $radius; }

    public function intersects(Circle $circle) {//перетин кіл
        $distance = sqrt(pow($this->x - $circle->getX(), 2) + pow($this->y - $circle-
>getY(), 2));
        return $distance < ($this->radius + $circle->getRadius());
    }
}
```

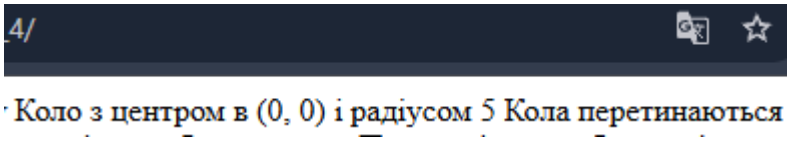

Рис. 2. Результат виконання програми

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

Лістинг (FileManager):

```
<?php
class FileManager {
    public static $dir = 'text';
    public static function readFile($filename) {
        $path = self::$dir . '/' . $filename;
        if (file_exists($path)) {
            return file_get_contents($path);
        } else {
            return "Файл не знайдено.\n";
}
    }
```

```php
public static function writeFile($filename, $content) {
        $path = self::$dir . '/' . $filename;
        file_put_contents($path, $content, FILE_APPEND);
    }
    public static function clearFile($filename) {
        $path = self::$dir . '/' . $filename;
        file_put_contents($path, '');
    }
}
```

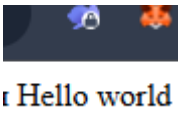

Рис. 3. Результат виконання завдання

Завдання 8 (Наслідування)

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

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

Лістинг (human):

```php
<?php
abstract class Human {
    protected $height, $weight, $age;

    public function __construct($h, $w, $a) {
        $this->height = $h;
        $this->weight = $w;
        $this->age = $a;
    }

    public function getHeight() { return $this->height; }
    public function setHeight($h) { $this->height = $h; }

    public function getWeight() { return $this->weight; }
    public function setWeight($w) { $this->weight = $w; }

    public function getAge() { return $this->age; }
    public function setAge($a) { $this->age = $a; }

    abstract protected function onChildBorn();
    public function birthChild() {
        $this->onChildBorn();
    }
}
```

Лістинг (student):

```php
<?php
require_once 'HouseCleaning.php';
require_once 'Human.php';

class Student extends Human implements HouseCleaning {
    private $university, $course;
```

```
    public function __construct($h, $w, $a, $univ, $course) {
        parent::__construct($h, $w, $a);
        $this->university = $univ;
        $this->course = $course;
    }

    public function increaseCourse() {
        $this->course++;
    }

    public function cleanRoom() {
        echo "Студент прибирає кімнату\n";
    }

    public function cleanKitchen() {
        echo "Студент прибирає кухню\n";
    }

    protected function onChildBorn() {
        echo "Студент став батьком!\n";
    }
}
```

Лістинг (programmer):

```
<?php
require_once 'HouseCleaning.php';
require_once 'Human.php';

class Programmer extends Human implements HouseCleaning {
    private $languages = [], $experience;

    public function __construct($h, $w, $a, $langs, $exp) {
        parent::__construct($h, $w, $a);
        $this->languages = $langs;
        $this->experience = $exp;
    }

    public function addLanguage($lang) {
        $this->languages[] = $lang;
    }

    public function cleanRoom() {
        echo "Програміст прибирає кімнату\n";
    }

    public function cleanKitchen() {
        echo "Програміст прибирає кухню\n";
```

```
    }

    protected function onChildBorn() {
        echo "Програміст став батьком!\n";
    }
}
```

кімнату Студент став батьком! Програміст прибирає кухню Програміст став батьком!

Рис. 4. Результат виконання програми

Лістинг (index.php, де викликаються та тестуються всі методи):

```
<?php
require_once 'autoload.php';
require_once 'Circle.php';
require_once 'FileManager.php';
require_once 'Student.php';
require_once 'Programmer.php';

use Models\UserModel;
use Controllers\UserController;
use Views\UserView;

$model = new UserModel();
$controller = new UserController();
$view = new UserView();

echo $model->getUserName() . "\n";
$controller->showMessage();
echo "\n";
$view->render();
echo "\n";

// Перевірка Circle
$circle1 = new Circle(0, 0, 5);
$circle2 = new Circle(3, 4, 5);
echo $circle1 . "\n";
echo $circle1->intersects($circle2) ? "Кола перетинаються\n" : "Не перетинаються\n";

// Статичні методи
if (!is_dir("text")) mkdir("text");
FileManager::writeFile("test.txt", "Hello world\n");
echo FileManager::readFile("test.txt");
//FileManager::clearFile("test.txt");

// Люди, студенти, програмісти
$student = new Student(180, 70, 20, "КНУ", 2);
$student->increaseCourse();
```

```php
$student->cleanRoom();
$student->birthChild();

$programmer = new Programmer(175, 80, 25, ["PHP"], 5);
$programmer->addLanguage("Python");
$programmer->cleanKitchen();
$programmer->birthChild();
```

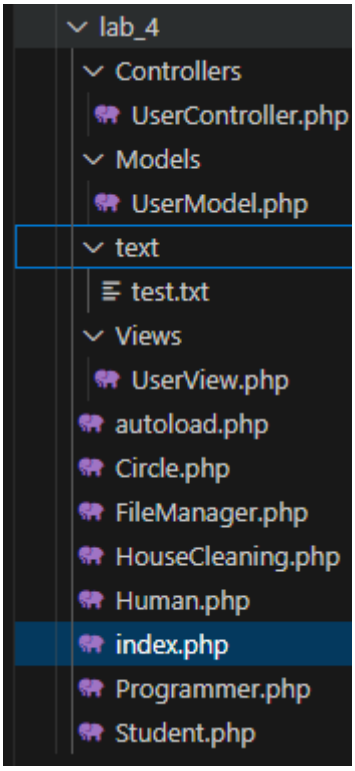

Рис. 5. Структура проекту