**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");

$userModel = new Models\UserModels();

$userController = new Controllers\UserControllers();

$userView = new Views\UserViews();

echo $userModel->getMessage() . "<br>";

echo $userController->getMessage() . "<br>";

echo $userView->getMessage() . "<br>";

<?php

function autoload($class)

{

    str\_replace('\\', '/', $class);

    $path = "$class.php";

    if (is\_file($path))

        require\_once($path);

}

spl\_autoload\_register("autoload");

<?php

namespace Controllers;

/\*\*

 \* UserController відповідає за обробку запитів користувачів

 \*/

class UserControllers

{

    public function getMessage()

    {

        return "Hello from UserController!";

    }

}

<?php

namespace Models;

/\*\*

 \* UserModel відповідає за роботу з даними користувачів

 \*/

class UserModels

{

    public function getMessage()

    {

        return "Hello from UserModel!";

    }

}

<?php

namespace Views;

/\*\*

 \* UserView відповідає за відображення даних користувачів

 \*/

class UserViews

{

    public function getMessage()

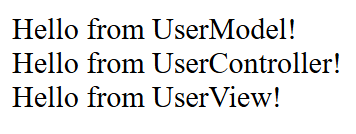
    {

        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(1, 4, 1);

if($Circle1 -> circlesCheck($Circle2)){

    echo 'Кола перетинаються';

}else{

    echo 'Кола не перетинаються';

}

<?php

class Circle{

    private $x;

    private $y;

    private $radius;

    public function \_\_get($key)

    {

        return $this->$key;

    }

    public function \_\_set($key, $value){

        $this->$key = $value;

    }

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

        $this->x = $x;

        $this->y = $y;

        $this->radius = $radius;

    }

    public function \_\_toString() : string

    {

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

    }

    public function circlesCheck($circle)

    {

        $distance = sqrt(pow($circle->x - $this->x, 2) + pow($circle->y - $this->y, 2));

        if ($distance <= ($this->radius + $circle->radius)) {

            return true;

        } else {

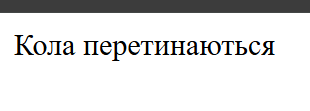
            return false;

        }

    }

}

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");

$Text = new Text();

?>

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <form action="" method="POST">

        <table>

            <th>Читання файлу</th>

            <tr>

                <td>Оберіть файл, який хочете вивести: </td>

                <td>

                    <select name="readFile" id="">

                        <option value="">Оберіть файл</option>

                        <option value="text1.txt" <?php if (isset($\_POST['readFile']) && $\_POST['readFile'] == 'text1.txt') echo 'selected'; ?>>text1</option>

                        <option value="text2.txt" <?php if (isset($\_POST['readFile']) && $\_POST['readFile'] == 'text2.txt') echo 'selected'; ?>>text2</option>

                        <option value="text3.txt" <?php if (isset($\_POST['readFile']) && $\_POST['readFile'] == 'text3.txt') echo 'selected'; ?>>text3</option>

                    </select>

                </td>

                <td>

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

                </td>

            </tr>

            <th>Запис до файлу</th>

            <tr>

                <td>Введіть текст, який бажаєте додати</td>

                <td><input type="text" name="text"></td>

            </tr>

            <tr>

                <td>Оберіть файл, в який хочете додати текст: </td>

                <td>

                    <select name="writeFile" id="">

                        <option value="">Оберіть файл</option>

                        <option value="text1.txt" <?php if (isset($\_POST['writeFile']) && $\_POST['writeFile'] == 'text1.txt') echo 'selected'; ?>>text1</option>

                        <option value="text2.txt" <?php if (isset($\_POST['writeFile']) && $\_POST['writeFile'] == 'text2.txt') echo 'selected'; ?>>text2</option>

                        <option value="text3.txt" <?php if (isset($\_POST['writeFile']) && $\_POST['writeFile'] == 'text3.txt') echo 'selected'; ?>>text3</option>

                    </select>

                </td>

                <td>

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

                </td>

            </tr>

            <th>Очищення файлу</th>

            <tr>

                <td>Оберіть файл, який хочете очистити: </td>

                <td>

                    <select name="clearFile" id="">

                        <option value="">Оберіть файл</option>

                        <option value="text1.txt" <?php if (isset($\_POST['clearFile']) && $\_POST['clearFile'] == 'text1.txt') echo 'selected'; ?>>text1</option>

                        <option value="text2.txt" <?php if (isset($\_POST['clearFile']) && $\_POST['clearFile'] == 'text2.txt') echo 'selected'; ?>>text2</option>

                        <option value="text3.txt" <?php if (isset($\_POST['clearFile']) && $\_POST['clearFile'] == 'text3.txt') echo 'selected'; ?>>text3</option>

                    </select>

                </td>

                <td><input type="submit" value="Очистити вміст файлу" name="clear"></td>

            </tr>

        </table>

    </form>

    <?php

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

        $fileForRead = $\_POST['readFile'];

        $textForm = $\_POST['text'];

        $fileForWrite = $\_POST['writeFile'];

        $fileForClear = $\_POST['clearFile'];

        if (isset($\_POST['read'])) {

            echo "<h3>$fileForRead</h3>" . $Text->readText($fileForRead);

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

            echo $Text->writeText($fileForWrite, $textForm);

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

            echo $Text->clearText($fileForClear);

        }

    }

    ?>

</body>

</html>

<?php

class Text

{

    protected static $dir = "text";

    static public function ReadText($filename)

    {

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

        if (is\_file($path)) {

            return file\_get\_contents($path);

        }

    }

    static public function WriteText($filename, $text)

    {

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

        if (is\_file($path)) {

            file\_put\_contents($path, $text . "\n", FILE\_APPEND);

            echo "У файл було успішно додано текст";

        }

    }

    static public function ClearText($filename)

    {

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

        if (is\_file($path)) {

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

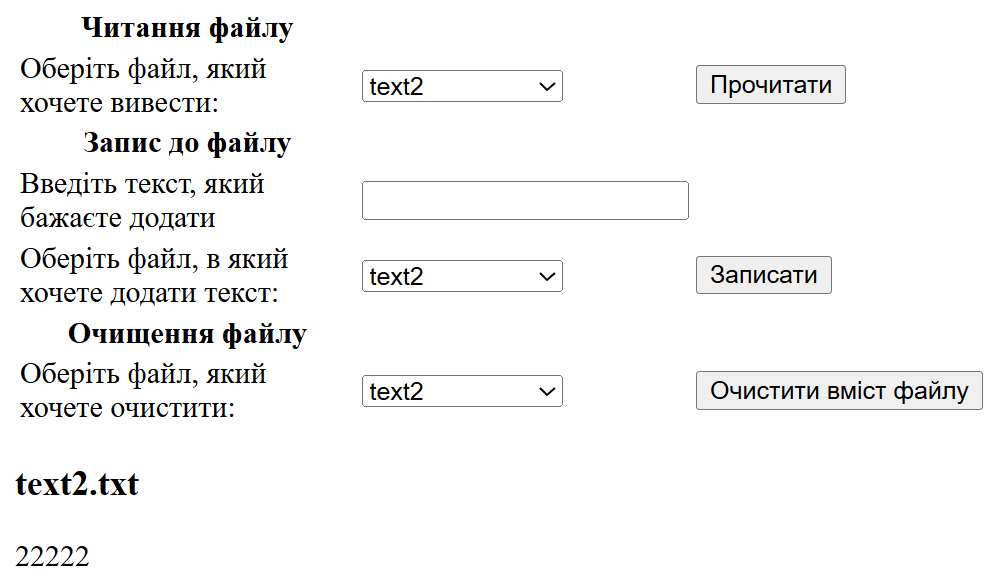
            echo "Файл було успішно очищено";

        }

    }

}

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

function autoload($class)

{

    $path = "classes/$class.php";

    if (file\_exists($path))

        require\_once($path);

}

spl\_autoload\_register("autoload");

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

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

echo $programmer ->getprogrammingLanguages("0"). "<br>";

echo $programmer->getprogrammingLanguages("") . "<br>";

echo $programmer->getProgrammerInfo();

echo $student->getStudentInfo();

echo $student -> setHeight(65);

echo $student->getStudentInfo();

<?php

class Human

{

    public $name;

    public $age;

    public $height;

    public $weight;

    public function getName(){

        return $this->name;

    }

    public function getAge()

    {

        return $this->age;

    }

    public function getHeight()

    {

        return $this->height;

    }

    public function getWeight()

    {

        return $this->weight;

    }

    public function setName($value)

    {

        $this->name = $value;

    }

    public function setAge($value)

    {

        $this->age = $value;

    }

    public function setHeight($value)

    {

        $this->height = $value;

    }

    public function setWeight($value)

    {

        $this->weight = $value;

    }

    public function \_\_construct($name, $height, $weight, $age)

    {

        $this->name = $name;

        $this->height = $height;

        $this->weight = $weight;

        $this->age = $age;

    }

    public function getInfo()

    {

        return "Name: {$this->name}; Height: {$this->height}; Weight: {$this->weight}; Age: {$this->age}";

    }

}

<?php

class Programmer extends Human

{

    public $programmingLanguages = [];

    public $experience;

    public function getprogrammingLanguages($value)

    {

        if($value != ""){

            if (array\_key\_exists($value, $this->programmingLanguages)) {

                return $this->programmingLanguages[$value];

            } else {

                return null;

            }

        }else{

            return implode(", ", $this->programmingLanguages);

        }

    }

    public function getExperience(){

        return $this->experience;

    }

    public function setProgrammingLanguages($key, $value){

        $this->programmingLanguages[$key] = $value;

    }

    public function setExperience($value){

        $this->experience = $value;

    }

    public function \_\_construct($name, $height, $weight, $age, $programmingLanguages, $experience)

    {

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

        $this->programmingLanguages = $programmingLanguages;

        $this->experience = $experience;

    }

    public function addLanguage($language){

        $this->programmingLanguages[] = $language;

    }

    public function getProgrammerInfo()

    {

        return parent::getInfo() . "<br> Programming language: ". implode(", ", $this->programmingLanguages)."; Experience: {$this->experience} years<br>";

    }

}

<?php

class Student extends Human{

    public $university;

    public $course;

    public function setUniversity($value){

        $this->university = $value;

    }

    public function setCourse($value){

        $this->course = $value;

    }

    public function getUniversity(){

        return $this->university;

    }

    public function getCourse()

    {

        return $this->course;

    }

    public function \_\_construct($name, $height, $weight, $age, $university, $course)

    {

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

        $this->university = $university;

        $this->course = $course;

    }

    public function nextCourse(){

        return $this->course++;

    }

    public function getStudentInfo()

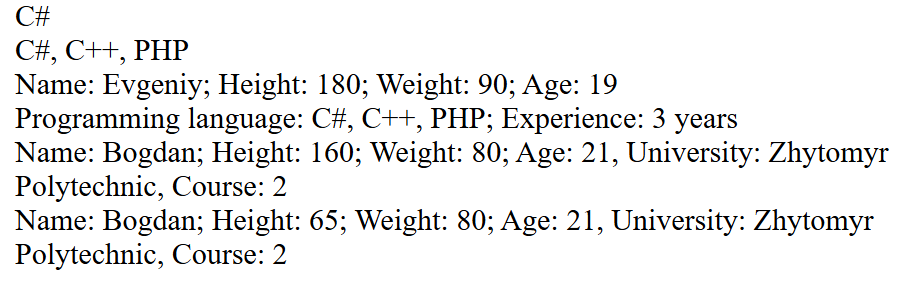
    {

        return parent::getInfo() . ", University: {$this->university}, Course: {$this->course} <br>";

    }

}

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

function autoload($class)

{

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

    $path = "./{$class}.php";

    if (file\_exists($path)) {

        require\_once($path);

    }

}

spl\_autoload\_register("autoload");

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

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

echo $programmer->getProgrammerInfo();

echo $student->getStudentInfo();

echo $student->cleanKitchen();

echo $programmer->cleanRoom();

<?php

namespace classes;

abstract class Human

{

    public $name;

    public $age;

    public $height;

    public $weight;

    public function getName(){

        return $this->name;

    }

    public function getAge()

    {

        return $this->age;

    }

    public function getHeight()

    {

        return $this->height;

    }

    public function getWeight()

    {

        return $this->weight;

    }

    public function setName($value)

    {

        $this->name = $value;

    }

    public function setAge($value)

    {

        $this->age = $value;

    }

    public function setHeight($value)

    {

        $this->height = $value;

    }

    public function setWeight($value)

    {

        $this->weight = $value;

    }

    public function \_\_construct($name, $height, $weight, $age)

    {

        $this->name = $name;

        $this->height = $height;

        $this->weight = $weight;

        $this->age = $age;

    }

    public function getInfo()

    {

        return "Name: {$this->name}; Height: {$this->height}; Weight: {$this->weight}; Age: {$this->age}";

    }

    protected abstract function birthMessage();

    public function getBirth(){

        $message = $this->birthMessage();

        echo "Дитина народилась: $message<br>";

    }

    abstract protected function cleanMessage();

    public function cleanRoom(): string

    {

        $message = $this->cleanMessage();

        return "$message прибирає кімнату<br>";

    }

    public function cleanKitchen(): string

    {

        $message = $this->cleanMessage();

        return "$message прибирає кухню<br>";

    }

}

<?php

namespace classes;

interface ICleanHouse {

    public function cleanRoom():string;

    public function cleanKitchen(): string;

}

<?php

namespace classes;

class Programmer extends Human

{

    public $programmingLanguages = [];

    public $experience;

    public function getprogrammingLanguages($value)

    {

        if($value != ""){

            if (array\_key\_exists($value, $this->programmingLanguages)) {

                return $this->programmingLanguages[$value];

            } else {

                return null;

            }

        }else{

            return implode(", ", $this->programmingLanguages);

        }

    }

    public function getExperience(){

        return $this->experience;

    }

    public function setProgrammingLanguages($key, $value){

        $this->programmingLanguages[$key] = $value;

    }

    public function setExperience($value){

        $this->experience = $value;

    }

    public function \_\_construct($name, $height, $weight, $age, $programmingLanguages, $experience)

    {

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

        $this->programmingLanguages = $programmingLanguages;

        $this->experience = $experience;

    }

    public function addLanguage($language){

        $this->programmingLanguages[] = $language;

    }

    public function getProgrammerInfo()

    {

        return parent::getInfo() . "<br> Programming language: ". implode(", ", $this->programmingLanguages)."; Experience: {$this->experience} years<br>";

    }

    protected function birthMessage(){

        return "Привіт, я програміст";

    }

    protected function cleanMessage()

    {

        return "Програміст";

    }

}

<?php

namespace classes;

class Student extends Human{

    public $university;

    public $course;

    public function setUniversity($value){

        $this->university = $value;

    }

    public function setCourse($value){

        $this->course = $value;

    }

    public function getUniversity(){

        return $this->university;

    }

    public function getCourse()

    {

        return $this->course;

    }

    public function \_\_construct($name, $height, $weight, $age, $university, $course)

    {

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

        $this->university = $university;

        $this->course = $course;

    }

    public function nextCourse(){

        return $this->course++;

    }

    public function getStudentInfo()

    {

        return parent::getInfo() . ", University: {$this->university}, Course: {$this->course} <br>";

    }

    protected function birthMessage()

    {

        return "Привіт, я студент";

    }

    protected function cleanMessage()

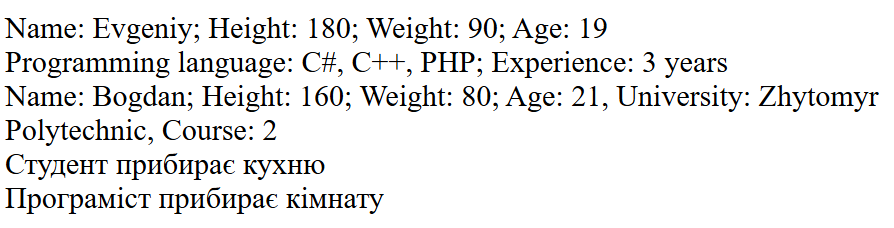
    {

        return "Студент";

    }

}

Виконання програми:



**Висновок :** Під час виконання лабораторної роботи було розглянуто основи об'єктно-орієнтованого програмування (ООП) в PHP. Це дозволило навчитися створювати та використовувати класи і об'єкти, що є основними концепціями ООП, і дають змогу ефективно структурувати код для полегшення розробки великих і складних програмних систем.