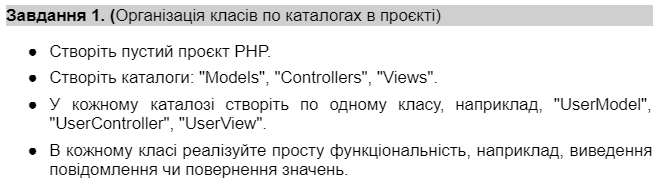
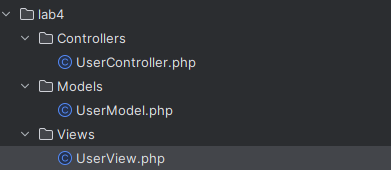
**Лабораторна робота №4**

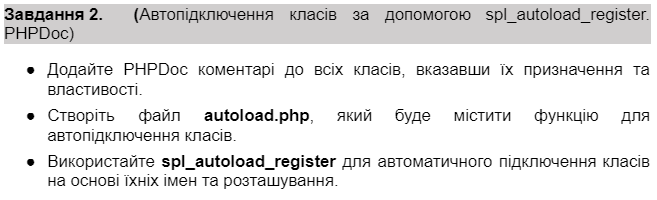
**Тема: Об'єктно-орієнтоване програмування в РНР**

**Мета роботи: навчитися працювати з класами**

**Завдання на лабораторну роботу:**







Лістинг програми(spl\_autoload\_register):  
<?php

/\*\*

\* Файл autoload.php для автоматичного підключення класів на основі їхніх імен та розташування.

\*/

function autoload($className) {

$file = \_\_DIR\_\_ . '/' . str\_replace('\\', '/', $className) . '.php';

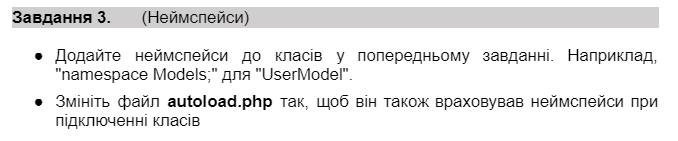
if (file\_exists($file)) {

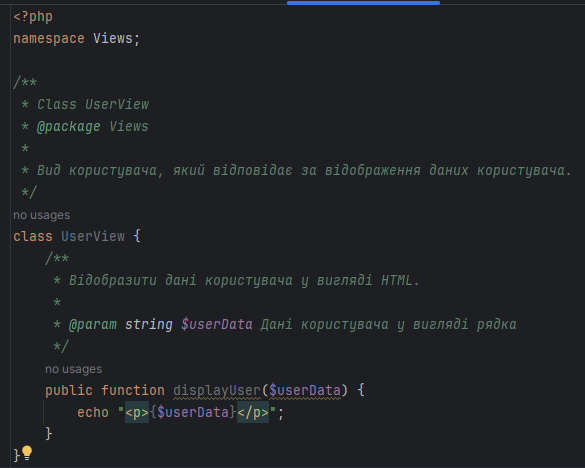
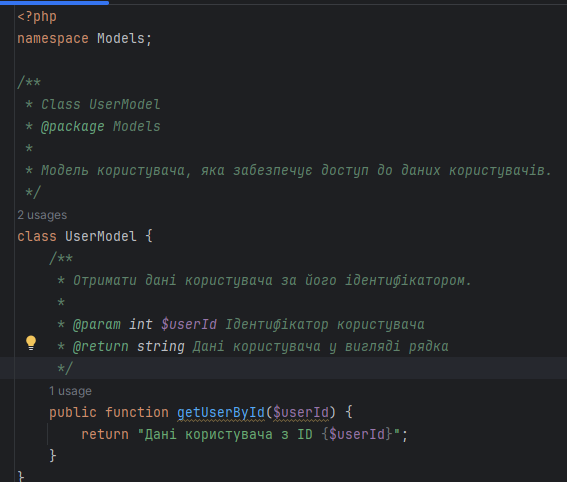
require\_once $file;

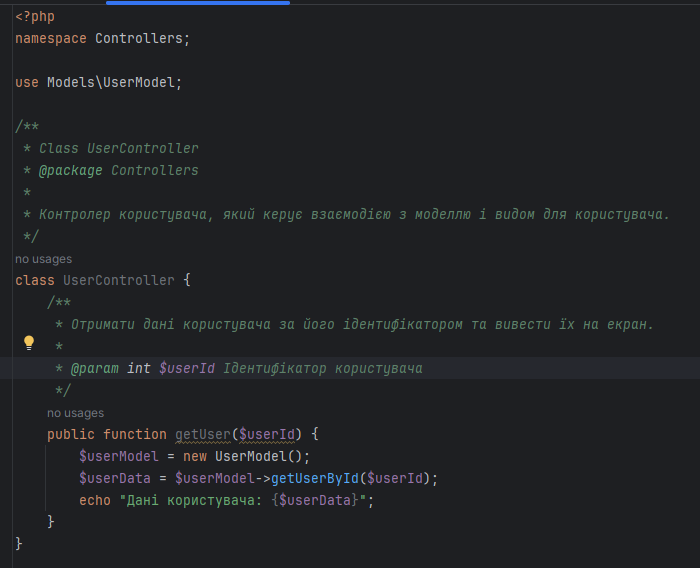
}

}

spl\_autoload\_register('autoload');







Лістинг програми(**autoload.php**):

<?php

/\*\*

\* Файл autoload.php для автоматичного підключення класів на основі їхніх імен та розташування, включаючи неймспейси.

\*/

function autoload($className) {

$className = ltrim($className, '\\');

$fileName = '';

$namespace = '';

if ($lastNsPos = strrpos($className, '\\')) {

$namespace = substr($className, 0, $lastNsPos);

$className = substr($className, $lastNsPos + 1);

$fileName = str\_replace('\\', DIRECTORY\_SEPARATOR, $namespace) . DIRECTORY\_SEPARATOR;

}

$fileName .= str\_replace('\_', DIRECTORY\_SEPARATOR, $className) . '.php';

$fullFileName = \_\_DIR\_\_ . DIRECTORY\_SEPARATOR . $fileName;

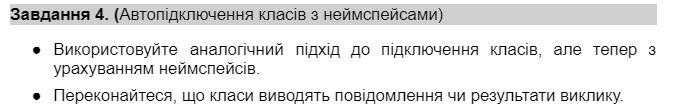
if (file\_exists($fullFileName)) {

require $fullFileName;

}

}

spl\_autoload\_register('autoload');



Лістинг програми(autoload.php):

<?php

/\*\*

\* Файл autoload.php для автоматичного підключення класів на основі їхніх неймспейсів і імен.

\*/

function autoload($className) {

$className = ltrim($className, '\\');

$fileName = '';

$namespace = '';

if ($lastNsPos = strrpos($className, '\\')) {

$namespace = substr($className, 0, $lastNsPos);

$className = substr($className, $lastNsPos + 1);

$fileName = str\_replace('\\', DIRECTORY\_SEPARATOR, $namespace) . DIRECTORY\_SEPARATOR;

}

$fileName .= str\_replace('\_', DIRECTORY\_SEPARATOR, $className) . '.php';

$fullFileName = \_\_DIR\_\_ . DIRECTORY\_SEPARATOR . $fileName;

if (file\_exists($fullFileName)) {

require $fullFileName;

}

}

spl\_autoload\_register('autoload');

Лістинг програми(index.php):

<?php

require\_once('autoload.php');

use Controllers\UserController;

use Views\UserView;

$userController = new UserController();

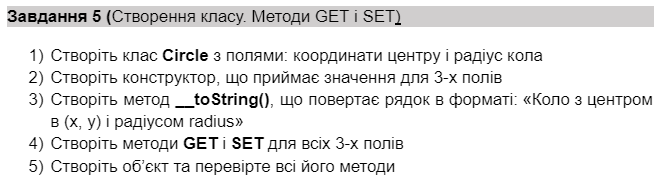
$userView = new UserView();

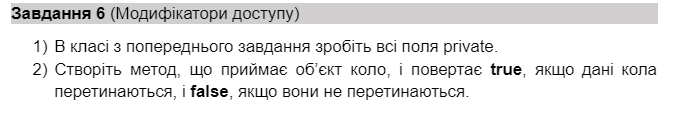
$userData = $userController->getUser(123);

$userView->displayUser($userData);

Результат виконання:







Лістинг програми(Circle.php):

<?php

namespace Models;

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() {

return "Circle with center at ({$this->x}, {$this->y}) and radius {$this->radius}";

}

public function getX() {

return $this->x;

}

public function getY() {

return $this->y;

}

public function getRadius() {

return $this->radius;

}

public function setX($x) {

$this->x = $x;

}

public function setY($y) {

$this->y = $y;

}

public function setRadius($radius) {

$this->radius = $radius;

}

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());

}

}

Лістинг програми(index.php)

<?php

use Models\Circle;

require\_once 'D:\WampServer\www\lab4\autoload.php';

$circle1 = new Circle(0, 0, 5);

echo $circle1 . "<br>";

echo "X: " . $circle1->getX() . ", Y: " . $circle1->getY() . ", Radius: " . $circle1->getRadius() . "<br>"; // Виведе: X: 0, Y: 0, Radius: 5

$circle1->setX(2);

$circle1->setY(3);

$circle1->setRadius(8);

echo "Updated: X: " . $circle1->getX() . ", Y: " . $circle1->getY() . ", Radius: " . $circle1->getRadius() . "<br>"; // Виведе: Updated: X: 2, Y: 3, Radius: 8

$circle2 = new Circle(4, 4, 5);

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

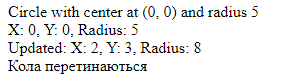
echo "Кола перетинаються";

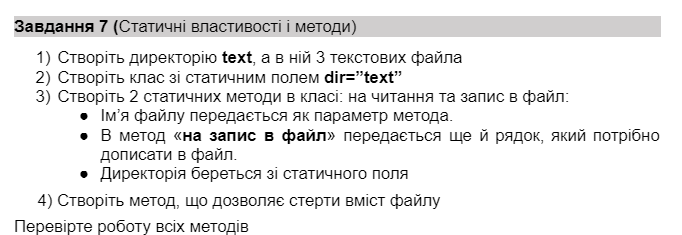
} else {

echo "Кола не перетинаються";

}

Результат виконання програми:





Лістинг програми(index.php)  
<?php

use Models\Circle;

require\_once 'D:\WampServer\www\lab4\autoload.php';

class FileManager {

public static $dir = "text/";

public static function readFromFile($filename) {

$filePath = self::$dir . $filename;

if (file\_exists($filePath)) {

return file\_get\_contents($filePath);

} else {

return "File $filename not found!";

}

}

public static function writeToFile($filename, $contentToAppend) {

$filePath = self::$dir . $filename;

$content = file\_exists($filePath) ? file\_get\_contents($filePath) : '';

$content .= $contentToAppend;

file\_put\_contents($filePath, $content);

}

public static function clearFile($filename) {

$filePath = self::$dir . $filename;

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

}

}

$circle1 = new Circle(0, 0, 5);

echo $circle1 . "<br>";

echo "X: " . $circle1->getX() . ", Y: " . $circle1->getY() . ", Radius: " . $circle1->getRadius() . "<br>";

$circle1->setX(2);

$circle1->setY(3);

$circle1->setRadius(8);

echo "Updated: X: " . $circle1->getX() . ", Y: " . $circle1->getY() . ", Radius: " . $circle1->getRadius() . "<br>";

$circle2 = new Circle(4, 4, 5);

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

echo "Кола перетинаються";

} else {

echo "Кола не перетинаються";

}

$file1 = 'file1.txt';

$file2 = 'file2.txt';

$file3 = 'file3.txt';

FileManager::writeToFile($file1, "First line of file 1.\n");

FileManager::writeToFile($file2, "First line of file 2.\n");

echo FileManager::readFromFile($file1) . "<br>";

echo FileManager::readFromFile($file2) . "<br>";

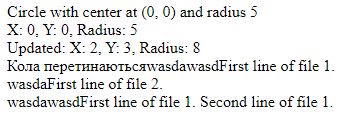
FileManager::writeToFile($file1, "Second line of file 1.\n");

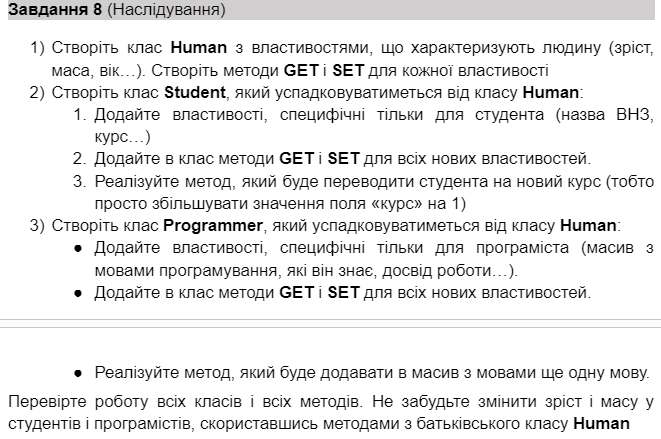
echo FileManager::readFromFile($file1) . "<br>";

FileManager::clearFile($file2);

echo FileManager::readFromFile($file2) . "<br>";

Результат виконання програми:





Лістинг програми(Human):

<?php

namespace Models;

class Human

{

private $height;

private $weight;

private $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;

}

public function \_\_toString()

{

return "Людина з ростом {$this->height} см, вагою {$this->weight} кг і віком {$this->age} років.";

}

}

Лістинг програми(Student):

<?php

namespace Models;

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 advanceCourse()

{

$this->course += 1;

}

public function \_\_toString()

{

return parent::\_\_toString() . ", студент ВНЗ {$this->university}, курс {$this->course}.";

}

}

Лістинг програми(Programmer):

<?php

namespace Models;

class Programmer extends Human

{

private $languages;

private $experience;

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

{

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

$this->languages = $languages;

$this->experience = $experience;

}

public function getLanguages()

{

return $this->languages;

}

public function setLanguages($languages)

{

$this->languages = $languages;

}

public function getExperience()

{

return $this->experience;

}

public function setExperience($experience)

{

$this->experience = $experience;

}

public function addLanguage($language)

{

$this->languages[] = $language;

}

public function \_\_toString()

{

$languages = implode(', ', $this->languages);

return parent::\_\_toString() . ", програміст зі знанням мов: {$languages}, досвід роботи: {$this->experience} років.";

}

}

Лістинг програми(index.php)

<?php

use Models\Human;

use Models\Programmer;

use Models\Student;

require\_once 'D:\WampServer\www\lab4\autoload.php';

$human = new Human(180, 75, 30);

echo "Людина 1: " . $human . "<br>";

$human->setHeight(185);

$human->setWeight(80);

$human->setAge(35);

echo "Оновлені дані: " . $human . "<br>";

$programmer = new Programmer(170, 70, 25, ['PHP', 'JavaScript'], 5);

echo "Програміст: " . $programmer . "<br>";

$programmer->addLanguage('Python');

echo "Оновлені дані програміста: " . $programmer . "<br>";

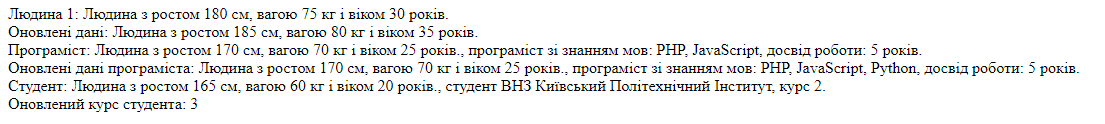
$student = new Student(165, 60, 20, 'Київський Політехнічний Інститут', 2);

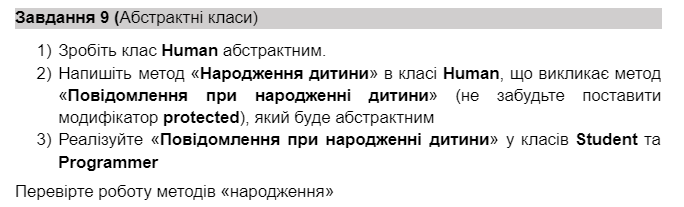
echo "Студент: " . $student . "<br>";

$student->advanceCourse();

echo "Оновлений курс студента: " . $student->getCourse() . "<br>";

Результат виконання програми:





Лістинг програми(Human):

<?php

namespace Models;

abstract class Human

{

private $height;

private $weight;

private $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;

}

abstract protected function birthAnnouncement();

public function giveBirth()

{

$this->birthAnnouncement();

}

public function \_\_toString()

{

return "Людина з зростом {$this->height} см, вагою {$this->weight} кг і віком {$this->age} років";

}

}

Лістинг програми(Student):

<?php

namespace Models;

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 advanceCourse()

{

$this->course += 1;

}

protected function birthAnnouncement()

{

echo "Студент народив дитину!<br>";

}

public function \_\_toString()

{

return parent::\_\_toString() . ", студент ВНЗ {$this->university}, курс {$this->course}.";

}

}

Лістинг програми(Programmer):

<?php

namespace Models;

class Programmer extends Human

{

private $languages;

private $experience;

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

{

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

$this->languages = $languages;

$this->experience = $experience;

}

public function getLanguages()

{

return $this->languages;

}

public function setLanguages($languages)

{

$this->languages = $languages;

}

public function getExperience()

{

return $this->experience;

}

public function setExperience($experience)

{

$this->experience = $experience;

}

public function addLanguage($language)

{

$this->languages[] = $language;

}

protected function birthAnnouncement()

{

echo "Програміст народив дитину!<br>";

}

public function \_\_toString()

{

$languages = implode(', ', $this->languages);

return parent::\_\_toString() . ", програміст зі знанням мов: {$languages}, досвід роботи: {$this->experience} років.";

}

}

Лістинг програми(index.php):

<?php

require\_once 'D:\WampServer\www\lab4\autoload.php';

use Models\Human;

use Models\Student;

use Models\Programmer;

$programmer = new Programmer(180, 75, 30, ['PHP', 'JavaScript'], 8);

echo "Програміст:<br>";

echo $programmer . "<br>";

$programmer->giveBirth();

echo "<br>";

$student = new Student(170, 60, 20, 'КНУ', 3);

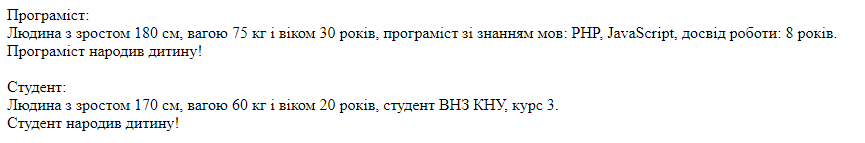
echo "Студент:<br>";

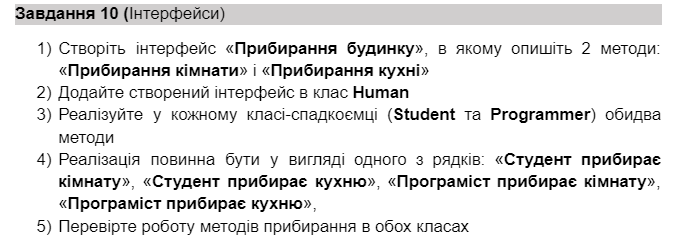
echo $student . "<br>";

$student->giveBirth();

echo "<br>";

Результат виконання програми:





Лістинг програми(HouseCleaning)

<?php

namespace Interfaces;

interface HouseCleaning

{

public function cleanRoom();

public function cleanKitchen();

}

Лістинг програми(Human)

<?php

namespace Models;

use Interfaces\HouseCleaning;

abstract class Human implements HouseCleaning

{

private $height;

private $weight;

private $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;

}

abstract protected function birthAnnouncement();

public function giveBirth()

{

$this->birthAnnouncement();

}

}

Лістинг програми(Student)

<?php

namespace Models;

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 nextCourse()

{

$this->course += 1;

}

protected function birthAnnouncement()

{

echo "Студент народив дитину!<br>";

}

public function cleanRoom()

{

echo "Студент прибирає кімнату.<br>";

}

public function cleanKitchen()

{

echo "Студент прибирає кухню.<br>";

}

public function \_\_toString()

{

return "Student: " . $this->getHeight() . " см, " . $this->getWeight() . " кг, " . $this->getAge() . " років, Університет: " . $this->university . ", Курс: " . $this->course;

}

}

Лістинг програми(Programmer)

<?php

namespace Models;

class Programmer extends Human

{

private $languages;

private $experience;

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

{

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

$this->languages = $languages;

$this->experience = $experience;

}

public function getLanguages()

{

return $this->languages;

}

public function setLanguages($languages)

{

$this->languages = $languages;

}

public function getExperience()

{

return $this->experience;

}

public function setExperience($experience)

{

$this->experience = $experience;

}

public function addLanguage($language)

{

$this->languages[] = $language;

}

protected function birthAnnouncement()

{

echo "Програміст народив дитину!<br>";

}

public function cleanRoom()

{

echo "Програміст прибирає кімнату.<br>";

}

public function cleanKitchen()

{

echo "Програміст прибирає кухню.<br>";

}

public function \_\_toString()

{

return "Programmer: " . $this->getHeight() . " см, " . $this->getWeight() . " кг, " . $this->getAge() . " років, Мови програмування: " . implode(', ', $this->languages) . ", Досвід: " . $this->experience . " років";

}

}

Лістинг програми(index.php)

<?php

require\_once 'D:\WampServer\www\lab4\autoload.php';

use Models\Human;

use Models\Student;

use Models\Programmer;

$student = new Student(180, 75, 25, 'Київський національний університет', 2);

echo $student;

echo "<br>";

$student->cleanRoom();

$student->cleanKitchen();

echo "<br>";

$programmer = new Programmer(175, 70, 30, ['PHP', 'JavaScript'], 5);

echo $programmer;

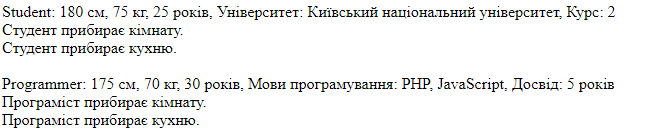
echo "<br>";

$programmer->cleanRoom();

$programmer->cleanKitchen();

echo "<br>";

Результат виконання роботи:



Посилання на github: https://github.com/loorkky/Backend.git