**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ**

**КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

**ЗВІТ З ЛАБОРАТОРНОЇ РОБОТИ №3**

з дисципліни:« Проектування веб-орієнтованих інформаційних систем**»**

Виконав: студент групи ЦТ.м-51н

Кириченко Д.С.

Варіант: 8

Перевірила: Парфененко Ю.В.

Суми 2025

**ТЕМА:** Реалізація абстрактних класів та інтерфейсів в PHP

**Варіант 8**

**Завдання 2.** Створіть абстрактний клас Figure з методом обчислення площі та методом виводу координат центру фігури на екран. Створити похідні класи Rectangle та Circle з власними методами. Продемонструйте виклик методів похідних класів.

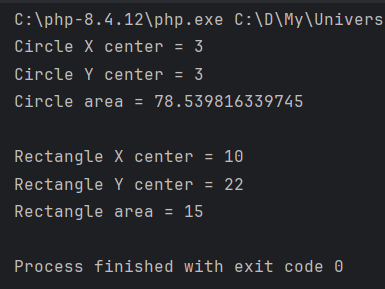


Рисунок 1 – Результат виконання завдання 2

**Програмний код ФАЙЛ index2.php**

<?php  
  
abstract class Figure // Basic class  
{  
 // Center coordinates  
 protected float $x\_center;  
 protected float $y\_center;  
 public function \_\_construct(float $x\_center, float $y\_center) {  
 $this->x\_center = $x\_center;  
 $this->y\_center = $y\_center;  
 }  
 // Abstract method for finding area  
 abstract public function area();  
  
 // Method for printing center coordinates  
 abstract public function printCenterCoordinates();  
}  
  
class Circle extends Figure  
{  
 // Circle radius  
 private float $radius;  
 public function \_\_construct(float $x\_center, float $y\_center, float $radius) {  
 parent::*\_\_construct*($x\_center, $y\_center);  
 $this->radius = $radius;  
 }  
 // Area finding  
 public function area(): float  
 {  
 return *M\_PI* \* ($this->radius \*\* 2);  
 }  
  
 public function printCenterCoordinates() {  
 echo "Circle X center = " . $this->x\_center . *PHP\_EOL* .  
 "Circle Y center = " . $this->y\_center . *PHP\_EOL*;  
 }  
}  
  
class Rectangle extends Figure  
{  
 // Rectangle properties  
 private float $width;  
 private float $height;  
  
 public function \_\_construct(  
 float $x\_center,  
 float $y\_center,  
 float $width,  
 float $height  
 ) {  
 parent::*\_\_construct*($x\_center, $y\_center);  
 $this->width = $width;  
 $this->height = $height;  
 }  
  
 // Area finding  
 public function area(): float  
 {  
 return $this->width \* $this->height;  
 }  
  
 public function printCenterCoordinates() {  
 echo "Rectangle X center = " . $this->x\_center . *PHP\_EOL* .  
 "Rectangle Y center = " . $this->y\_center . *PHP\_EOL*;  
 }  
}  
  
$circle = new Circle(3, 3, 5);  
$circle->printCenterCoordinates();  
echo "Circle area = " . $circle->area() . *PHP\_EOL* . *PHP\_EOL*;  
  
$rectangle = new Rectangle(10, 22, 3, 5);  
$rectangle->printCenterCoordinates();  
echo "Rectangle area = " . $rectangle->area() . *PHP\_EOL*;

**Завдання 3.** Для предметної області із завдання 3 лабораторної роботи №1 створіть батьківський абстрактний клас та ще один клас, для якого цей абстрактний клас також буде батьківським. Реалізуйте щонайменше один абстрактний метод та продемонструйте його реалізацію у дочірніх класах.

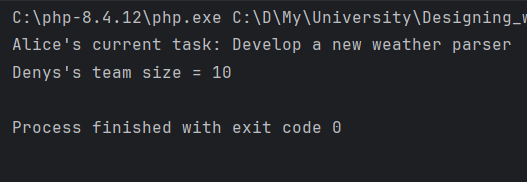


Рисунок 2 – Результат виконання завдання 3

**Програмний код ФАЙЛ index3.php**

<?php  
  
// Variant 8  
  
// Abstract parental class  
abstract class Employee {  
 protected string $name;  
 protected float $salary;  
 public function \_\_construct(  
 string $name,  
 float $salary  
 ) {  
 $this->name = $name;  
 $this->salary = $salary;  
 }  
}  
  
// Subclass worker  
class Worker extends Employee  
{  
 private string $task;  
 // Constructor with parameter by default for each property  
 public function \_\_construct(  
 string $name = "Nick",  
 float $salary = 0,  
 string $task = "Working"  
 ) {  
 parent::*\_\_construct*($name, $salary);  
 $this->task = $task;  
 }  
  
 // Universal get function for every property  
 public function \_\_get(string $property): mixed {  
 // Check whether the property exists  
 if(property\_exists($this, $property)) {  
 return $this->$property;  
 }  
 return null;  
 }  
  
 // Universal setter for each property  
 public function \_\_set(string $property, mixed $value): void {  
 // Check whether the property exists  
 if(property\_exists($this, $property)) {  
 $this->$property = $value;  
 }  
 }  
  
 // Function to show worker current task  
 public function showTasks(): void {  
 echo $this->name . "'s current task: " . $this->task . *PHP\_EOL*;  
 }  
}  
  
// Subclass manager  
class Manger extends Employee {  
 private int $teamSize;  
  
 public function \_\_construct(  
 string $name,  
 int $salary,  
 int $teamSize  
 ) {  
 parent::*\_\_construct*($name, $salary);  
 $this->teamSize = $teamSize;  
 }  
  
 // Universal get function for every property  
 public function \_\_get(string $property): mixed {  
 // Check whether the property exists  
 if(property\_exists($this, $property)) {  
 return $this->$property;  
 }  
 return null;  
 }  
  
 // Universal setter for each property  
 public function \_\_set(string $property, mixed $value): void {  
 // Check whether the property exists  
 if(property\_exists($this, $property)) {  
 $this->$property = $value;  
 }  
 }  
 // Function to show team size  
 public function showTeamSize(): void {  
 echo $this->name . "'s team size = " . $this->teamSize . *PHP\_EOL*;  
 }  
}  
  
  
// Create a new Object  
$worker = new Worker(  
 "Alice",  
 20000,  
 "Develop a new weather parser"  
);  
$worker->showTasks();  
  
$manger = new Manger(  
 "Denys",  
 100000,  
 10  
);  
$manger->showTeamSize();

**Завдання 4.** Реалізуйте інтерфейс ILoger та клас FileLoger, який записує дату, час відправлення повідомлення та його зміст у текстовий файл. Продемонструйте роботу з методом log() класу FileLoger.

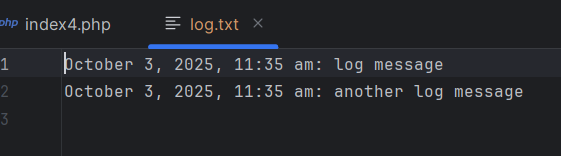


Рисунок 3 – Результат виконання завдання 4

**Програмний код ФАЙЛ index4.php**

<?php  
  
interface ILoger  
{  
 // Definition of a function which must be redefined in the next class  
 public function log($message);  
}  
  
class FileLoger implements ILoger  
{  
 private $file;  
 private $logFile;  
  
 public function \_\_construct($filename, $mode = 'a')  
 {  
 $this->logFile = $filename;  
 $this->file = fopen($filename, $mode) or die('Could not open the log file'); // Open file or print an error  
 }  
  
 // Redefinition of "log" function  
 public function log($message)  
 {  
 $message = date("F j, Y, g:i a") . ': ' . $message . "\n";  
 fwrite($this->file, $message);  
 }  
  
 public function \_\_destruct()  
 {  
 if ($this->file) {  
 fclose($this->file);  
 }  
 }  
}  
  
// Usage  
$FLog = new FileLoger('./log.txt', 'w');  
$FLog->log('log message');  
$FLog->log('another log message');

**Завдання 6.** Реалізуйте ієрархію класів. Визначте «Figure» як інтерфейс.

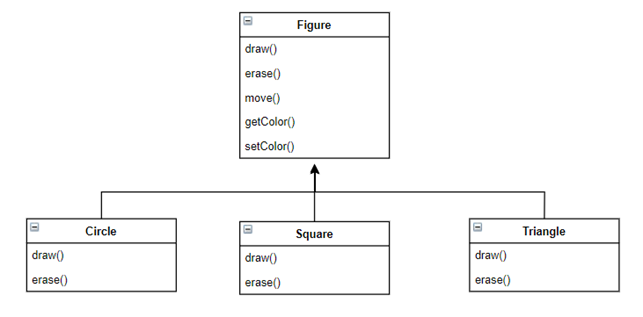


Рисунок 4 – Ієрархія класів

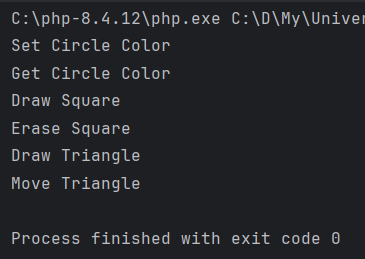


Рисунок 5 – Результат виконання завдання 6

**Програмний код ФАЙЛ index6.php**

<?php  
  
// Figure interface  
interface Figure  
{  
 // Methods of interface to implement  
 public function draw();  
 public function move();  
 public function erase();  
 public function getColor();  
 public function setColor();  
}  
  
// Class Circle with implemented interface Figure  
class Circle implements Figure  
{  
 // Implemented interface methods  
 public function draw() {  
 echo "Draw Circle" . *PHP\_EOL*;  
 }  
  
 public function move() {  
 echo "Move Circle" . *PHP\_EOL*;  
 }  
  
 public function erase() {  
 echo "Erase Circle" . *PHP\_EOL*;  
 }  
  
 public function getColor() {  
 echo "Get Circle Color" . *PHP\_EOL*;  
 }  
  
 public function setColor() {  
 echo "Set Circle Color" . *PHP\_EOL*;  
 }  
}  
  
// Class Square with implemented interface Figure  
class Square implements Figure {  
 // Implemented interface methods  
 public function draw() {  
 echo "Draw Square" . *PHP\_EOL*;  
 }  
  
 public function move() {  
 echo "Move Square" . *PHP\_EOL*;  
 }  
  
 public function erase() {  
 echo "Erase Square" . *PHP\_EOL*;  
 }  
  
 public function getColor() {  
 echo "Get Square Color" . *PHP\_EOL*;  
 }  
  
 public function setColor() {  
 echo "Set Square Color" . *PHP\_EOL*;  
 }  
}  
  
// Class Triangle with implemented interface Figure  
class Triangle implements Figure {  
 // Implemented interface methods  
 public function draw() {  
 echo "Draw Triangle" . *PHP\_EOL*;  
 }  
  
 public function move() {  
 echo "Move Triangle" . *PHP\_EOL*;  
 }  
  
 public function erase() {  
 echo "Erase Triangle" . *PHP\_EOL*;  
 }  
  
 public function getColor() {  
 echo "Get Triangle Color" . *PHP\_EOL*;  
 }  
  
 public function setColor() {  
 echo "Set Triangle Color" . *PHP\_EOL*;  
 }  
}  
  
// Create new objects  
$circle = new Circle();  
$square = new Square();  
$triangle = new Triangle();  
  
// Call methods of created objects  
$circle->setColor();  
$circle->getColor();  
$square->draw();  
$square->erase();  
$triangle->draw();  
$triangle->move();

**Завдання 8.** Для предметної області із завдання 3 лабораторної роботи №2 створіть клас, який реалізує декілька інтерфейсів.

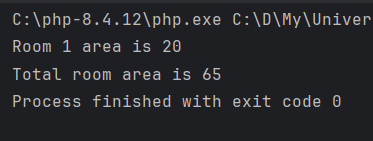


Рисунок 6 – Результат виконання завдання 8

**Програмний код ФАЙЛ index8.php**

<?php  
  
// First interface  
interface RoomMeasurements {  
 public function getArea(): float;  
}  
  
// Second interface  
interface totalMeasurements {  
 public static function getTotalArea(): float;  
}  
  
class Room implements RoomMeasurements, totalMeasurements  
{  
 // Properties  
 protected float $area;  
 // Static property for total area in all apartments  
 private static float *$total\_area* = 0;  
  
 // Constructor  
 public function \_\_construct(float $area = 0) {  
 $this->area = $area;  
 self::*$total\_area* += $area;  
 }  
  
 // Getter that uses properties like 'area' to check its existence and return value or null  
 public function \_\_get(string $property): mixed {  
 if (property\_exists($this, $property)) {  
 return $this->$property;  
 }  
 return null;  
 }  
  
 // Function to get room area  
 public function getArea(): float {  
 return $this->area;  
 }  
  
 // Static function to get total area of all apartments  
 public static function getTotalArea(): float {  
 return self::*$total\_area*;  
 }  
}  
  
// Creating objects of room class  
$room1 = new Room(20);  
$room2 = new Room(30);  
$room3 = new Room(15);  
  
// Calling class method to get room area  
$room\_area = $room1->getArea();  
echo "Room 1 area is " . $room\_area . *PHP\_EOL*;  
  
// Calling static method to get total area  
$total\_room\_area = Room::*getTotalArea*();  
echo "Total room area is " . $total\_room\_area;