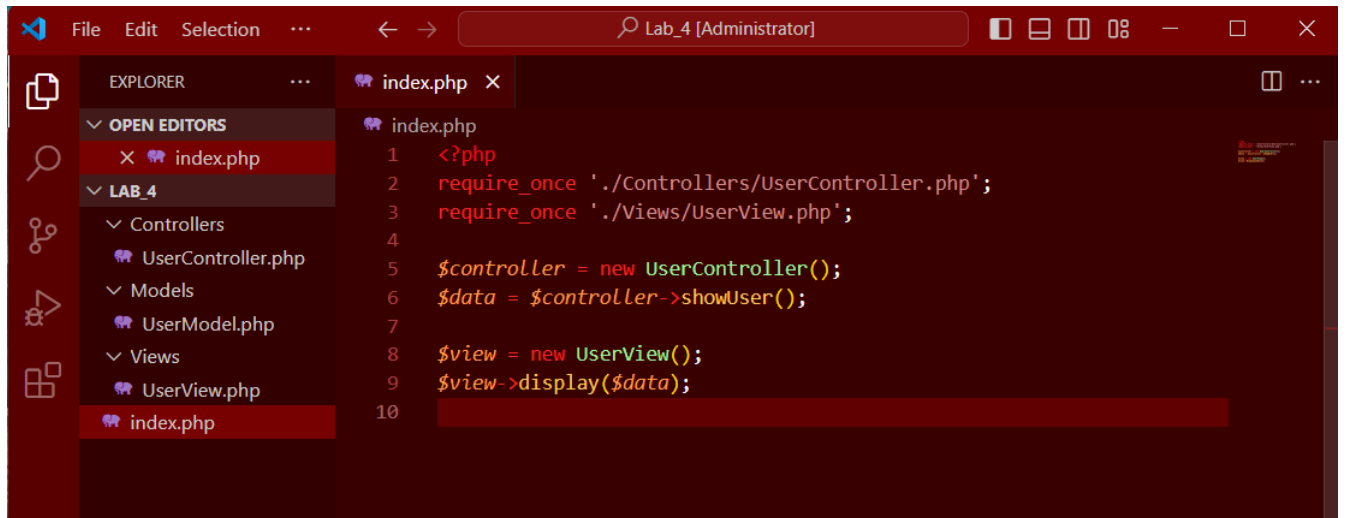


Лабораторна робота №4

Тема: Об'єктно-орієнтоване програмування в PHP

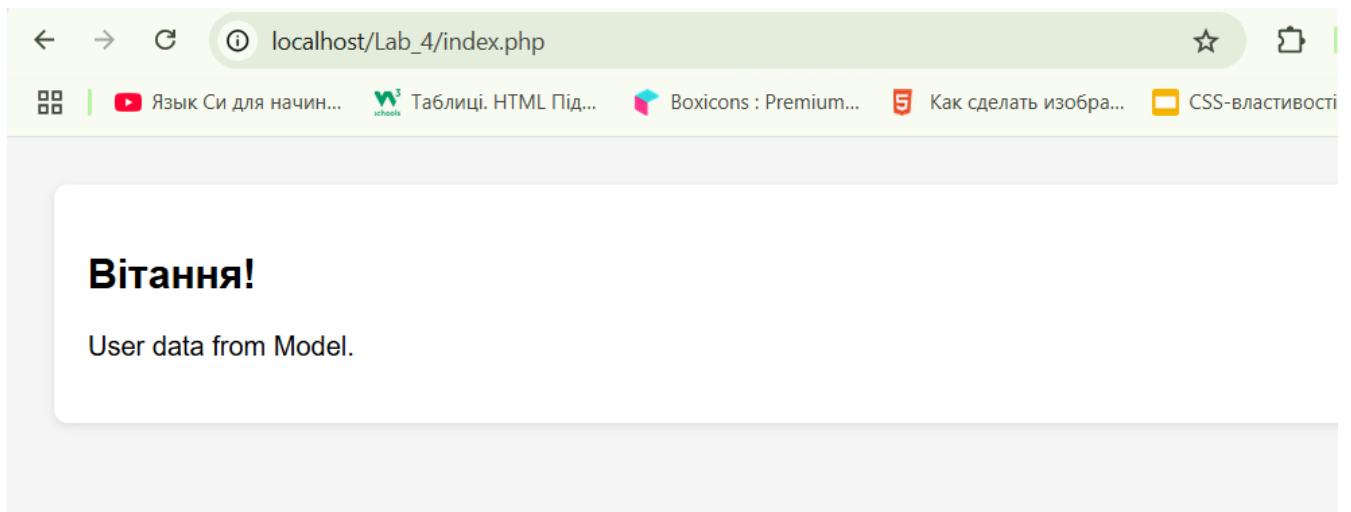
Завдання 1. (Організація класів по каталогах в проєкті)

- 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.



The screenshot shows the Visual Studio Code editor interface. The Explorer sidebar on the left displays the project structure: a folder named 'LAB_4' containing three subfolders: 'Controllers', 'Models', and 'Views'. Each folder contains a single PHP file: 'UserController.php', 'UserModel.php', and 'UserView.php' respectively. The main editor window shows the 'index.php' file, which contains the following PHP code:

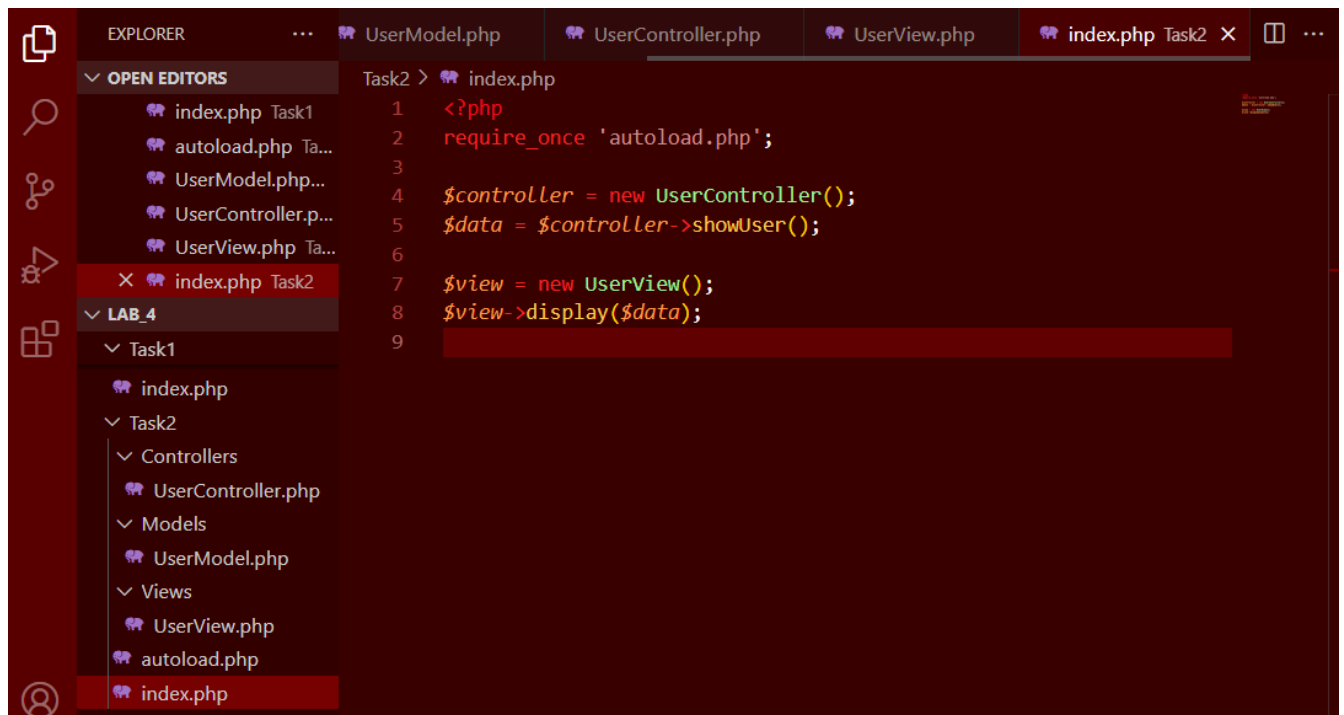
```
1 <?php
2 require_once './Controllers/UserController.php';
3 require_once './Views/UserView.php';
4
5 $controller = new UserController();
6 $data = $controller->showUser();
7
8 $view = new UserView();
9 $view->display($data);
10
```



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.

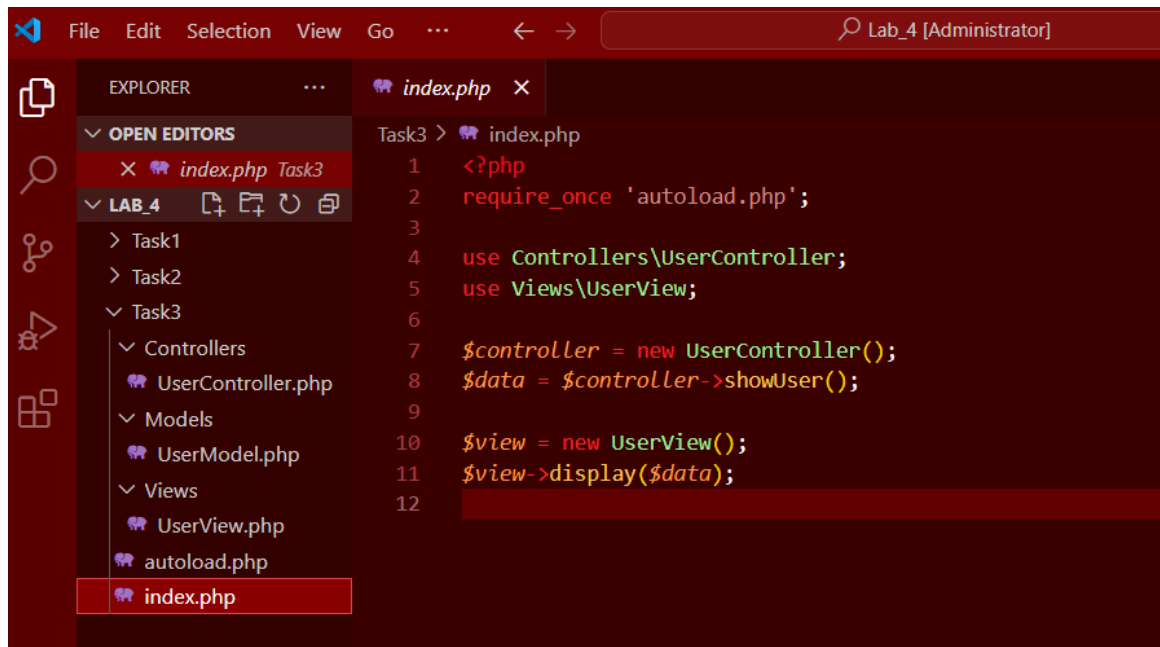


Вітання!

User data from Model.

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

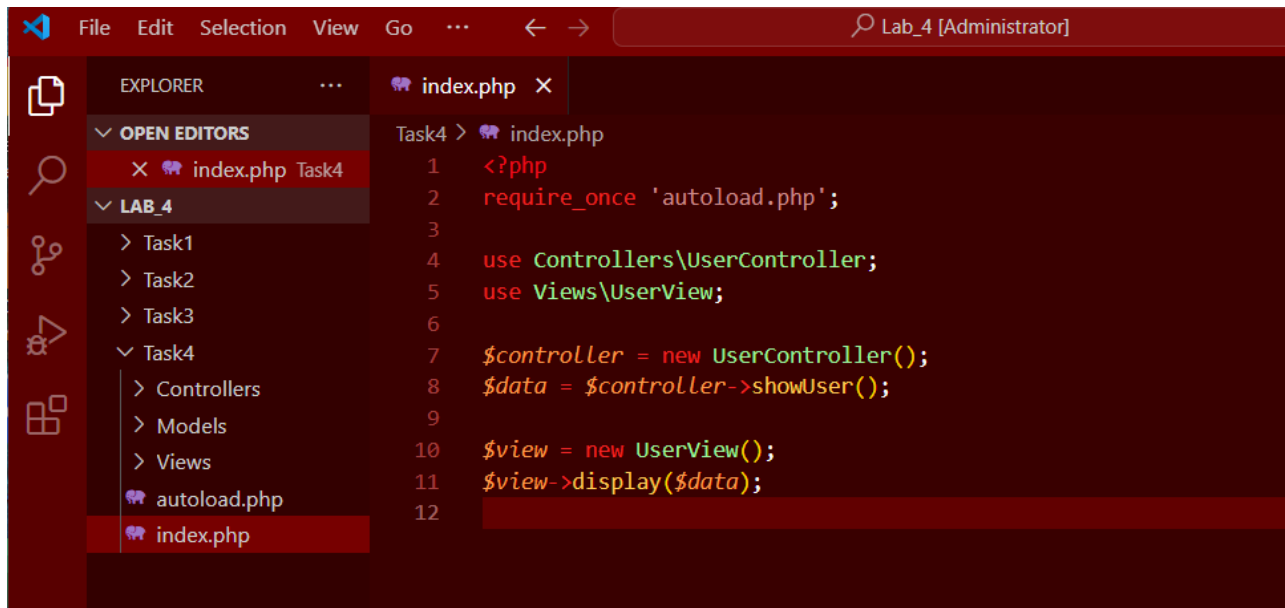


Вітання!

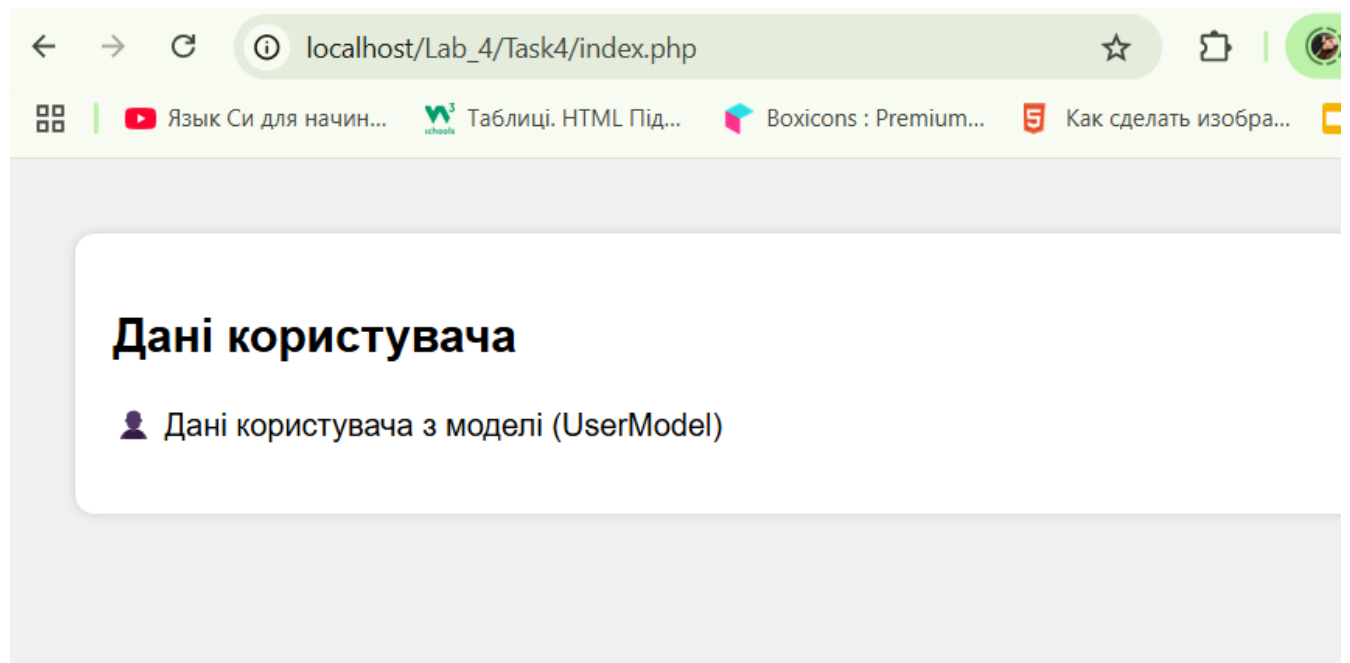
User data from Model.

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.

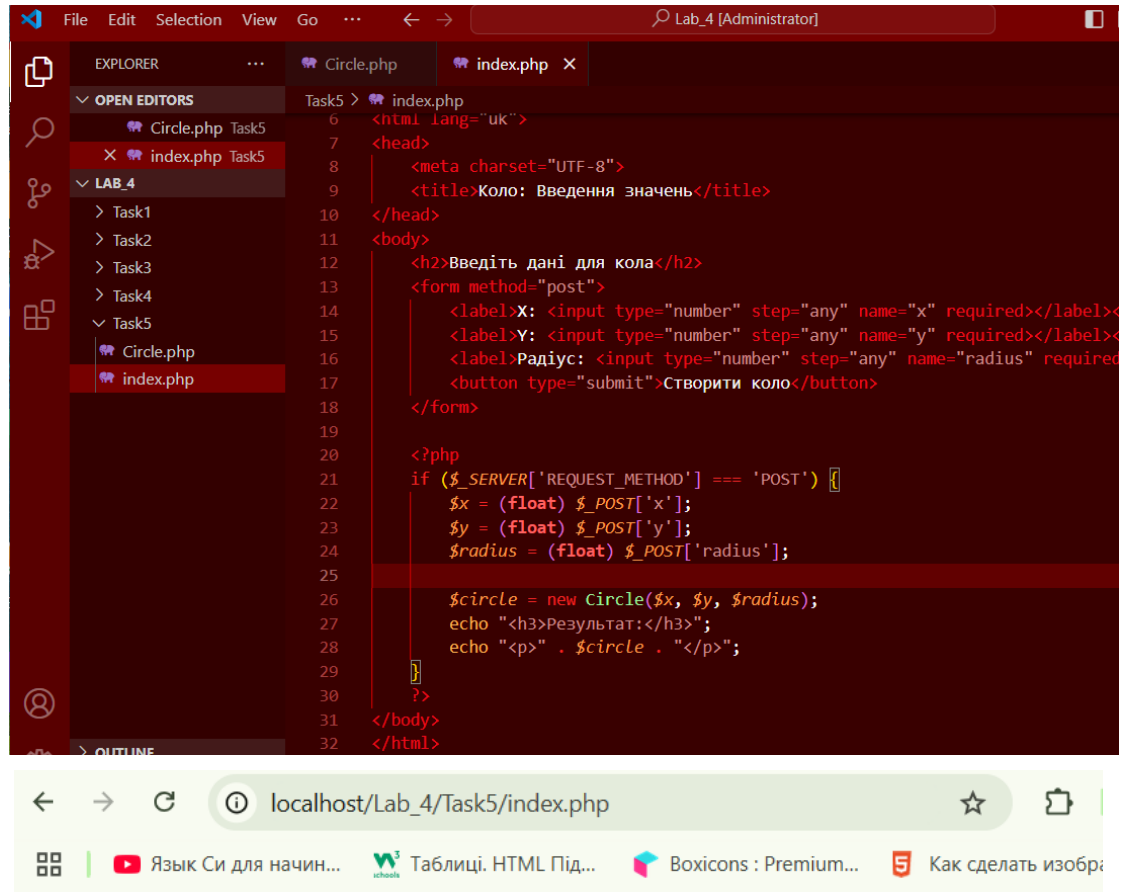


```
1 <?php
2 require_once 'autoload.php';
3
4 use Controllers\UserController;
5 use Views\UserView;
6
7 $controller = new UserController();
8 $data = $controller->showUser();
9
10 $view = new UserView();
11 $view->display($data);
12
```



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



Введіть дані для кола

X:

Y:

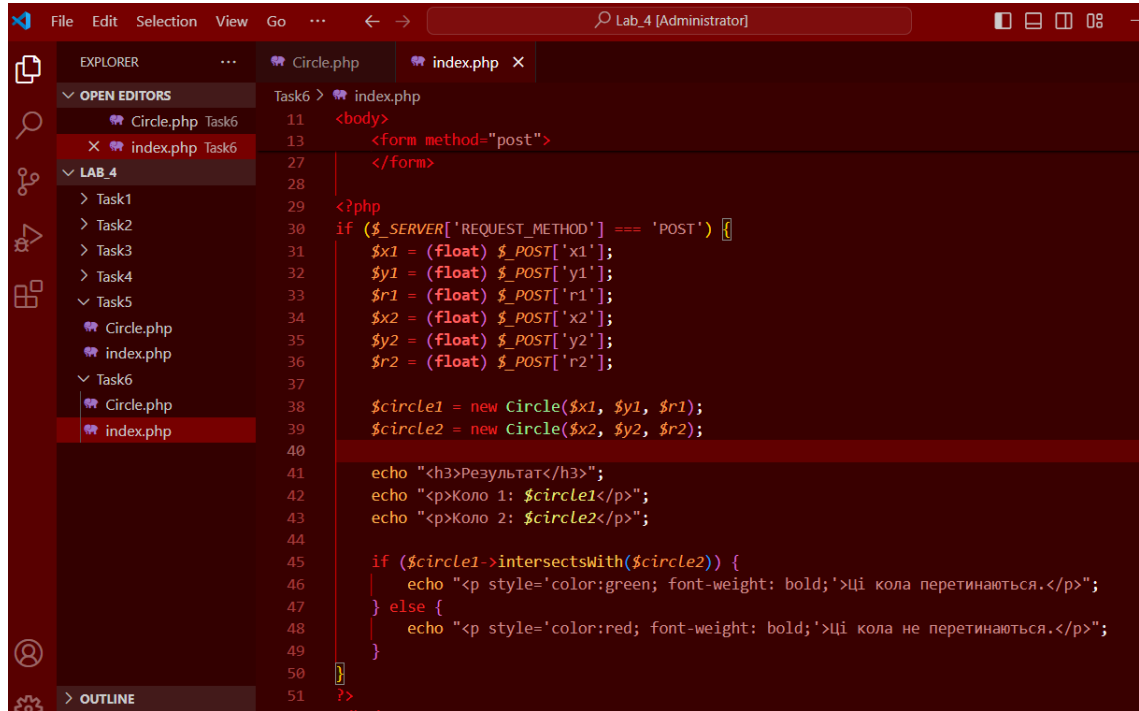
Радіус:

Результат:

Коло з центром в (2, 6) і радіусом 10

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.



```
File Edit Selection View Go ... Lab_4 [Administrator]
EXPLORER
  OPEN EDITORS
    Circle.php Task6
    index.php Task6
  LAB_4
    Task1
    Task2
    Task3
    Task4
    Task5
    Circle.php
    index.php
    Task6
      Circle.php
      index.php
  OUTLINE
Task6 > index.php
11 <body>
12 <form method="post">
13 </form>
27
28
29 <?php
30 if ($_SERVER['REQUEST_METHOD'] === 'POST') {
31     $x1 = (float) $_POST['x1'];
32     $y1 = (float) $_POST['y1'];
33     $r1 = (float) $_POST['r1'];
34     $x2 = (float) $_POST['x2'];
35     $y2 = (float) $_POST['y2'];
36     $r2 = (float) $_POST['r2'];
37
38     $circle1 = new Circle($x1, $y1, $r1);
39     $circle2 = new Circle($x2, $y2, $r2);
40
41     echo "<h3>Результат</h3>";
42     echo "<p>Коло 1: $circle1</p>";
43     echo "<p>Коло 2: $circle2</p>";
44
45     if ($circle1->intersects($circle2)) {
46         echo "<p style='color:green; font-weight: bold;'>Ці кола перетинаються.</p>";
47     } else {
48         echo "<p style='color:red; font-weight: bold;'>Ці кола не перетинаються.</p>";
49     }
50 }
51 ?>
```

Введіть дані для двох кіл

Коло 1

X:

Y:

Радіус:

Коло 2

X:

Y:

Радіус:

Результат

Коло 1: Коло з центром в (5, 8) і радіусом 10

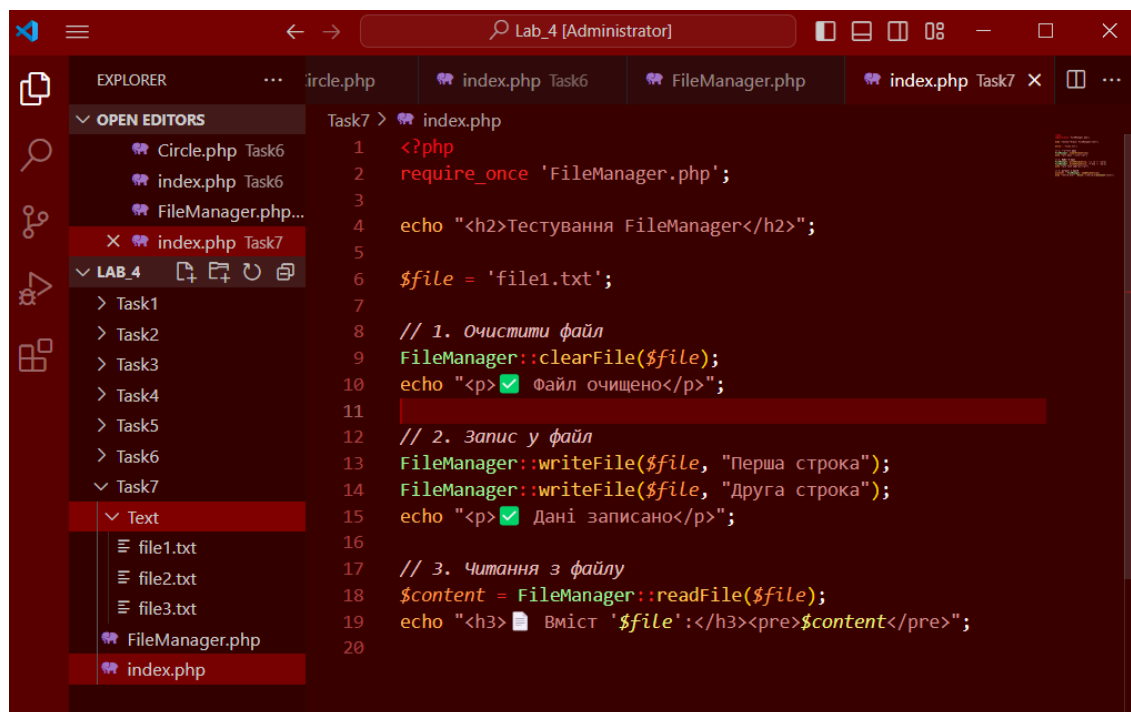
Коло 2: Коло з центром в (6, 5) і радіусом 8

Ці кола перетинаються.

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



```
<?php
require_once 'FileManager.php';

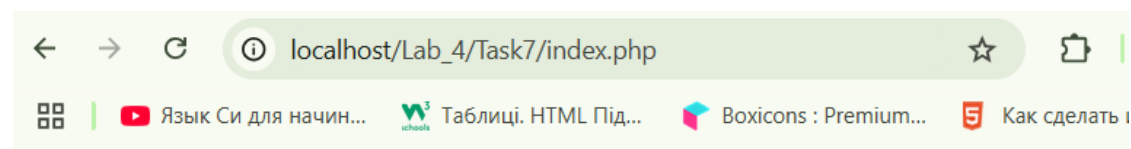
echo "<h2>Тестування FileManager</h2>";

$file = 'file1.txt';

// 1. Очистити файл
FileManager::clearFile($file);
echo "<p>✅ Файл очищено</p>";

// 2. Запис у файл
FileManager::writeFile($file, "Перша строка");
FileManager::writeFile($file, "Друга строка");
echo "<p>✅ Дані записано</p>";

// 3. Читання з файлу
$content = FileManager::readFile($file);
echo "<h3> 📄 Вміст '$file':</h3><pre>$content</pre>";
```



Тестування FileManager

✅ Файл очищено

✅ Дані записано

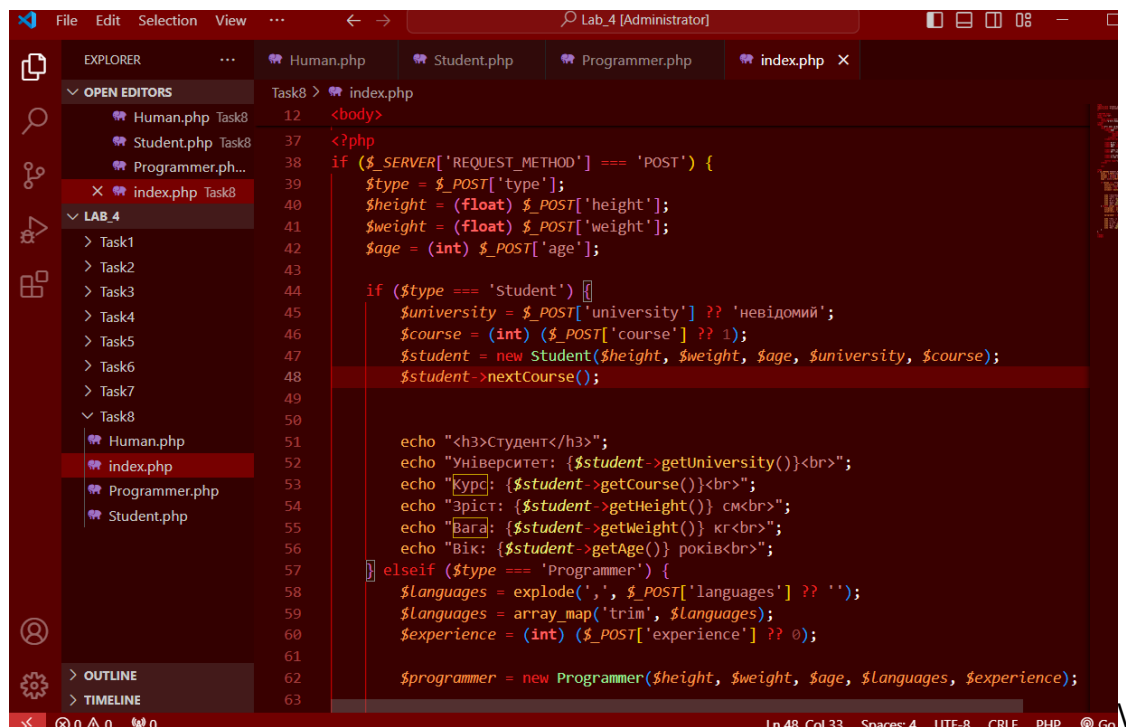
📄 Вміст 'file1.txt':

Перша строка

Друга строка

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.



```
File Edit Selection View ... Lab_4 [Administrator]
EXPLORER
  Task8 > index.php
  Human.php Task8
  Student.php Task8
  Programmer.ph...
  X index.php Task8
  LAB_4
    Task1
    Task2
    Task3
    Task4
    Task5
    Task6
    Task7
    Task8
      Human.php
      index.php
      Programmer.php
      Student.php
  OUTLINE
  TIMELINE
  Task8 > index.php
  12 <body>
  37 <?php
  38 if ($_SERVER['REQUEST_METHOD'] === 'POST') {
  39     $type = $_POST['type'];
  40     $height = (float) $_POST['height'];
  41     $weight = (float) $_POST['weight'];
  42     $age = (int) $_POST['age'];
  43
  44     if ($type === 'Student') {
  45         $university = $_POST['university'] ?? 'невідомий';
  46         $course = (int) ($_POST['course'] ?? 1);
  47         $student = new Student($height, $weight, $age, $university, $course);
  48         $student->nextCourse();
  49
  50
  51         echo "<h3>Студент</h3>";
  52         echo "Університет: {$student->getUniversity()}<br>";
  53         echo "Курс: {$student->getCourse()}<br>";
  54         echo "Зріст: {$student->getHeight()} см<br>";
  55         echo "Вага: {$student->getWeight()} кг<br>";
  56         echo "Вік: {$student->getAge()} років<br>";
  57     } elseif ($type === 'Programmer') {
  58         $languages = explode(',', $_POST['languages'] ?? '');
  59         $languages = array_map('trim', $languages);
  60         $experience = (int) ($_POST['experience'] ?? 0);
  61
  62         $programmer = new Programmer($height, $weight, $age, $languages, $experience);
  63     }
```

Створення об'єкта (Student або Programmer)

Тип:

Зріст:

Вага:

Вік:

Університет (для студента):

Курс (для студента):

Мови (для програміста):

Досвід (років):

Студент

Університет: KBNU

Курс: 5

Зріст: 193 см

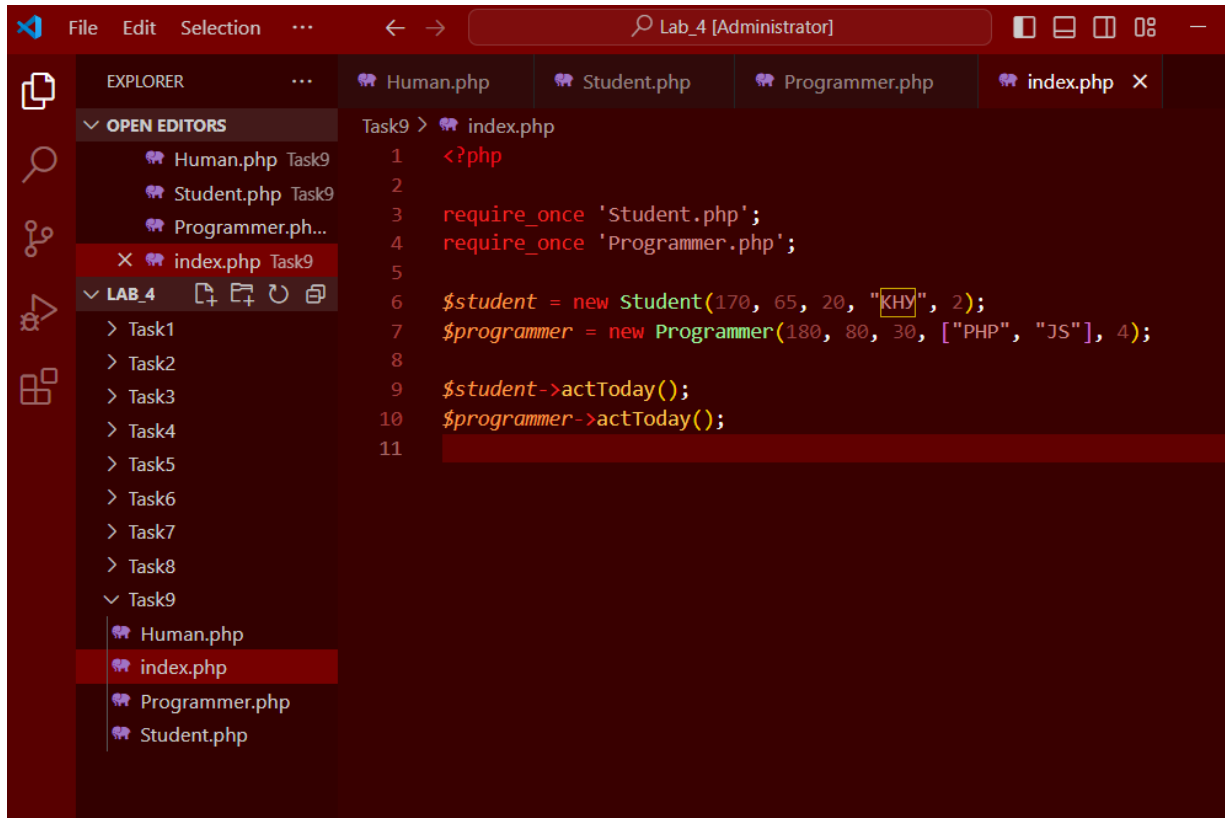
Вага: 85 кг

Вік: 17 років

Завдання 9 (Абстрактні класи)

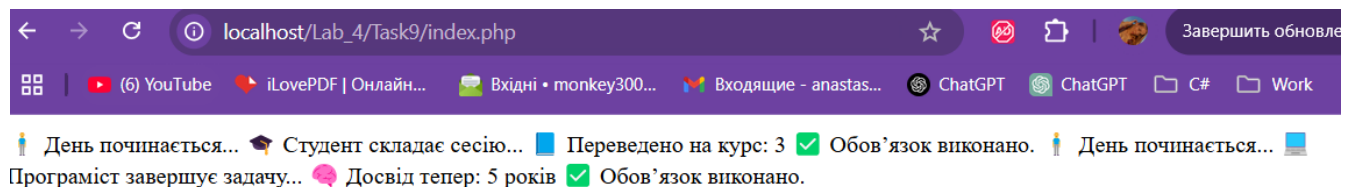
1. Зробіть клас **Human** абстрактним.
2. Напишіть метод «**Народження дитини**» в класі **Human**, що викликає метод «**Повідомлення при народженні дитини**» (не забудьте поставити модифікатор **protected**), який буде абстрактним
3. Реалізуйте «**Повідомлення при народженні дитини**» у класів **Student** та **Programmer**

Перевірте роботу методів «народження»



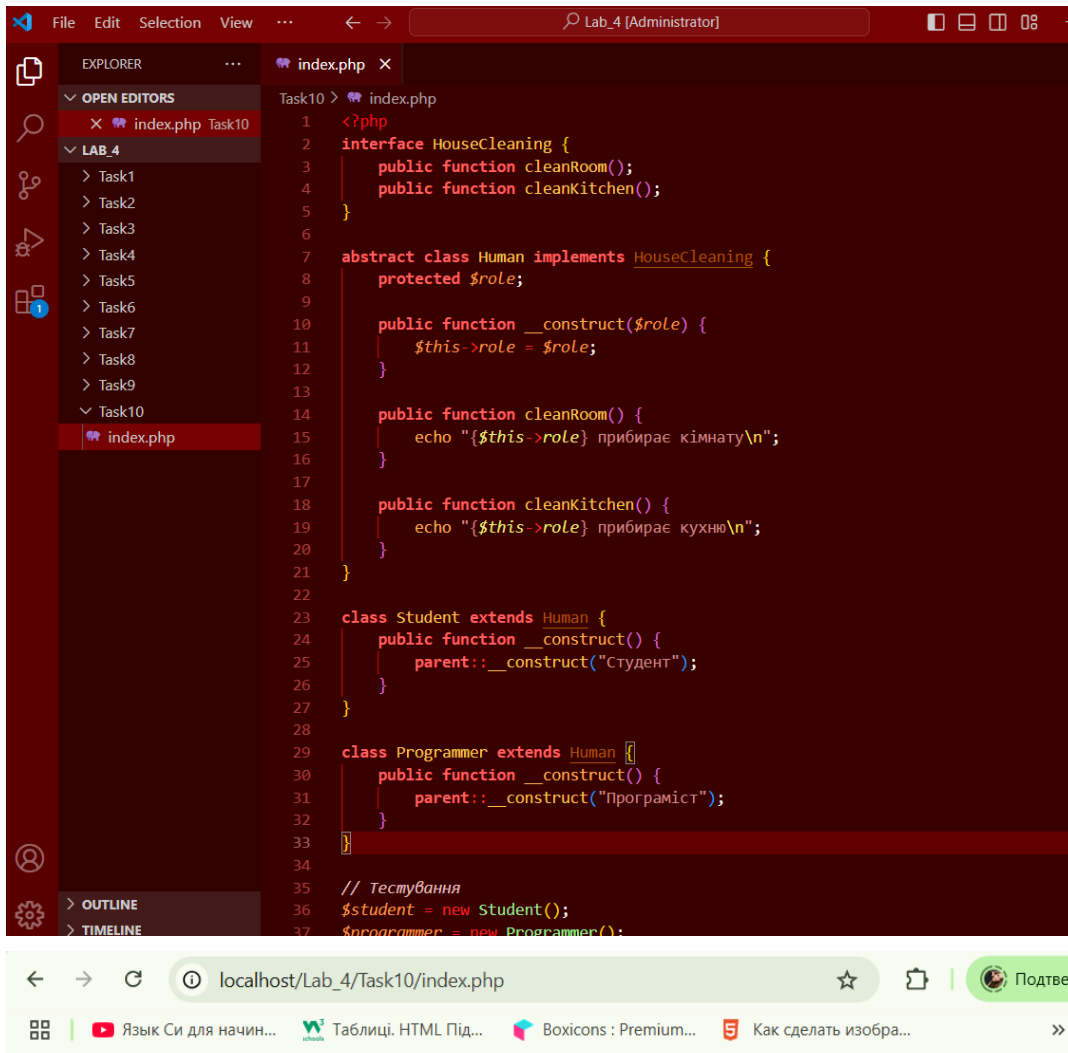
The screenshot shows the Visual Studio Code editor with the following files open: Human.php, Student.php, Programmer.php, and index.php. The Explorer sidebar shows a project structure with a 'LAB_4' folder containing 'Task1' through 'Task9'. The 'index.php' file is selected and its content is displayed in the editor. The code in 'index.php' is as follows:

```
1 <?php
2
3 require_once 'Student.php';
4 require_once 'Programmer.php';
5
6 $student = new Student(170, 65, 20, "КНУ", 2);
7 $programmer = new Programmer(180, 80, 30, ["PHP", "JS"], 4);
8
9 $student->actToday();
10 $programmer->actToday();
11
```



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



```
1 <?php
2 interface HouseCleaning {
3     public function cleanRoom();
4     public function cleanKitchen();
5 }
6
7 abstract class Human implements HouseCleaning {
8     protected $role;
9
10    public function __construct($role) {
11        $this->role = $role;
12    }
13
14    public function cleanRoom() {
15        echo "{$this->role} прибирає кімнату\n";
16    }
17
18    public function cleanKitchen() {
19        echo "{$this->role} прибирає кухню\n";
20    }
21 }
22
23 class Student extends Human {
24     public function __construct() {
25         parent::__construct("Студент");
26     }
27 }
28
29 class Programmer extends Human {
30     public function __construct() {
31         parent::__construct("Програміст");
32     }
33 }
34
35 // Тестування
36 $student = new Student();
37 $programmer = new Programmer();
```

localhost/Lab_4/Task10/index.php

Язык Си для начин... Таблиці. HTML Під... Boxicons : Premium... Как сделать изобра...

Студент прибирає кімнату Студент прибирає кухню Програміст прибирає кімнату Програміст прибирає кухню