

Лабораторна робота №2

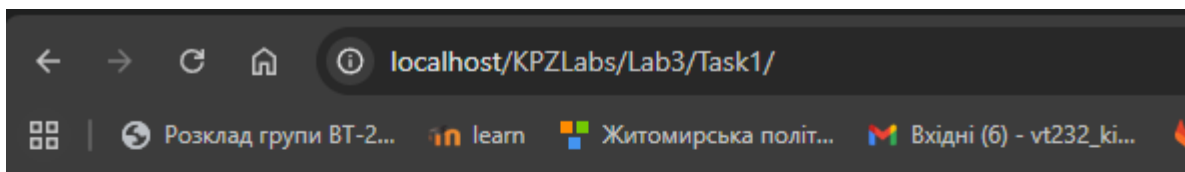
Тема: Структурні шаблони

Мета роботи: навчитися реалізовувати структурні шаблони проектування Адаптер, Декоратор, Міст, Компонувальник, Проксі, Легковаговик

Завдання 1: Адаптер.

1. Створіть клас `Logger`, який буде мати методи `Log()`, `Error()`, `Warn()`, які виводять повідомлення в консоль різними кольорами (зеленим, червоним і оранжевим відповідно).
2. Створіть клас `FileWriter` з методами `Write()`, `WriteLine()`.
3. За допомогою шаблону Адаптер створіть файловий логер.
4. Покажіть правильність роботи свого коду запустивши його в головному методі програми

Результат виконання:

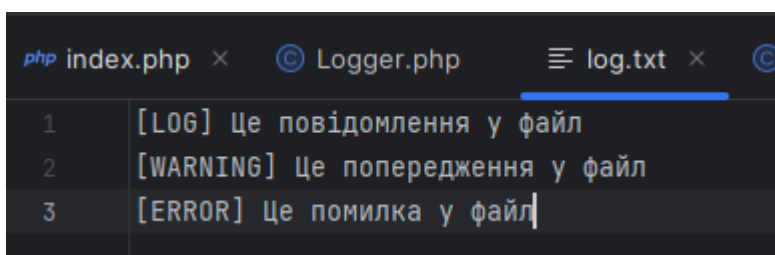


Консольний логер

- [32m[LOG] Це звичайне повідомлення□[0m
- [33m[WARNING] Це попередження□[0m
- [31m[ERROR] Це помилка□[0m

Файловий логер

Перевірте файл `log.txt` для перегляду записів.



index.php:

```
<?php
require_once __DIR__ . '/src/Logger.php';
require_once __DIR__ . '/src/FileWriter.php';
require_once __DIR__ . '/src/FileLoggerAdapter.php';

echo "Консольний логер<br>";
$consoleLogger = new Logger();
$consoleLogger->Log("Це звичайне повідомлення");
$consoleLogger->Warn("Це попередження");
$consoleLogger->Error("Це помилка");

echo "<br>Файловий логер<br>";
$fileWriter = new FileWriter(__DIR__ . "/log.txt");
$fileLogger = new FileLoggerAdapter($fileWriter);
$fileLogger->Log("Це повідомлення у файл");
$fileLogger->Warn("Це попередження у файл");
$fileLogger->Error("Це помилка у файл");

echo "Перевірте файл log.txt для перегляду записів.<br>";
```

клас Logger:

```
<?php

class Logger {
    public function Log($message) {
        echo "\033[32m[LOG] $message\033[0m<br>";
    }

    public function Error($message) {
        echo "\033[31m[ERROR] $message\033[0m<br>";
    }

    public function Warn($message) {
        echo "\033[33m[WARNING] $message\033[0m<br>";
    }
}
```

клас FileWriter:

```
<?php

class FileWriter {
    private $filePath;

    public function __construct($filePath) {
        $this->filePath = $filePath;
    }

    public function Write($text) {
        file_put_contents($this->filePath, $text, FILE_APPEND);
    }

    public function WriteLine($text) {
        file_put_contents($this->filePath, $text . PHP_EOL, FILE_APPEND);
    }
}
```

клас FileLoggerAdapter:

```
<?php
require_once __DIR__ . '/Logger.php';
require_once __DIR__ . '/FileWriter.php';

class FileLoggerAdapter extends Logger {
    private $fileWriter;

    public function __construct(FileWriter $fileWriter) {
        $this->fileWriter = $fileWriter;
    }

    public function Log($message) {
        $this->fileWriter->WriteLine("[LOG] $message");
    }

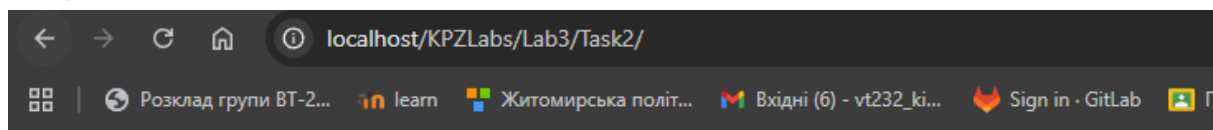
    public function Error($message) {
        $this->fileWriter->WriteLine("[ERROR] $message");
    }

    public function Warn($message) {
        $this->fileWriter->WriteLine("[WARNING] $message");
    }
}
```

Завдання 2: Декоратор.

1. Ви розробляєте РПГ гру. Створіть класи героїв Warrior, Mage, Palladin.
2. Для героїв створіть інвентар (одяг, зброю, артефакти), який може підходити будь-якому типу героїв, у вигляді декораторів.
3. Важливою вимогою є можливість використання декількох екземплярів інвентаря на герої одночасно.
4. Покажіть правильність роботи свого коду запустивши його в головному методі програми.

Результат виконання:



index.php:

```
<?php
require_once __DIR__ . '/src/Warrior.php';
require_once __DIR__ . '/src/Mage.php';
require_once __DIR__ . '/src/Palladin.php';
require_once __DIR__ . '/src/Armor.php';
require_once __DIR__ . '/src/Weapon.php';
require_once __DIR__ . '/src/Artifact.php';

echo "==== RPG Game: Heroes with Inventory ===<br>";

$hero1 = new Warrior();
$hero1 = new Armor($hero1);
$hero1 = new Weapon($hero1);
$hero1 = new Artifact($hero1);

echo $hero1->getName() . " | Power: " . $hero1->getPower() . "<br>";

$hero2 = new Mage();
$hero2 = new Weapon($hero2);
$hero2 = new Armor($hero2);

echo $hero2->getName() . " | Power: " . $hero2->getPower() . "<br>";

$hero3 = new Palladin();
```

```
$hero3 = new Artifact($hero3);

echo $hero3->getName() . " | Power: " . $hero3->getPower() . "<br>";

testFactory($kiFactory);
testFactory($balaxyFactory);
```

інтерфейс Hero:

```
<?php

interface Hero {
    public function getName();
    public function getPower();
}
```

клас Warrior:

```
<?php

require_once __DIR__ . '/Hero.php';

class Warrior implements Hero {
    public function getName() {
        return "Warrior";
    }

    public function getPower() {
        return 50;
    }
}
```

клас Mage:

```
<?php

require_once __DIR__ . '/Hero.php';

class Mage implements Hero {
    public function getName() {
        return "Mage";
    }

    public function getPower() {
        return 40;
    }
}
```

клас Palladin:

```
<?php

require_once __DIR__ . '/Hero.php';

class Palladin implements Hero {
    public function getName() {
        return "Palladin";
    }

    public function getPower() {
        return 60;
    }
}
```

клас HeroDecorator:

```
<?php

require_once __DIR__ . '/Hero.php';

abstract class HeroDecorator implements Hero {
    protected $hero;

    public function __construct(Hero $hero) {
        $this->hero = $hero;
    }

    public function getName() {
        return $this->hero->getName();
    }

    public function getPower() {
        return $this->hero->getPower();
    }
}
```

клас Armor:

```
<?php

require_once __DIR__ . '/HeroDecorator.php';

class Armor extends HeroDecorator {
    public function getName() {
        return $this->hero->getName() . " with Armor";
    }

    public function getPower() {
        return $this->hero->getPower() + 20;
    }
}
```

клас Weapon:

```
<?php

require_once __DIR__ . '/HeroDecorator.php';

class Weapon extends HeroDecorator {
    public function getName() {
        return $this->hero->getName() . " with Weapon";
    }

    public function getPower() {
        return $this->hero->getPower() + 30;
    }
}
```

клас Artifact :

```
<?php

require_once __DIR__ . '/HeroDecorator.php';

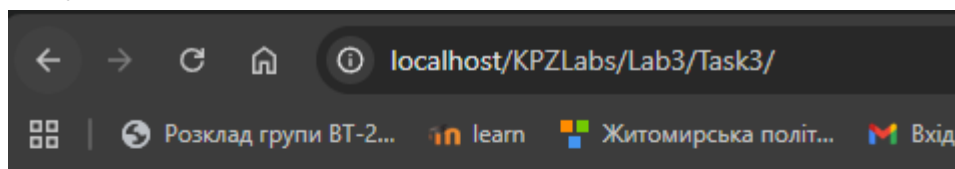
class Artifact extends HeroDecorator {
    public function getName() {
        return $this->hero->getName() . " with Artifact";
    }

    public function getPower() {
        return $this->hero->getPower() + 50;
    }
}
```

Завдання 3: Міст.

1. Ви працюєте над графічним редактором. Створіть базовий клас Shape.
2. Створіть дочірні до Shape класи, Circle, Square, Triangle.
3. За допомогою шаблону Міст додайте можливість рендерингу кожної з фігур як векторної або растрової графіки (вивівши відповідне повідомлення у консоль, наприклад "Drawing Triangle as pixels").
4. Покажіть правильність роботи свого коду запустивши його в головному методі програми.

Результат виконання:



```
==== Bridge Pattern Example ====  
Drawing Circle as vector graphics  
Drawing Circle as pixels  
Drawing Square as vector graphics  
Drawing Square as pixels  
Drawing Triangle as vector graphics  
Drawing Triangle as pixels
```

index.php:

```
<?php  
require_once __DIR__ . '/src/VectorRenderer.php';  
require_once __DIR__ . '/src/RasterRenderer.php';  
require_once __DIR__ . '/src/Circle.php';  
require_once __DIR__ . '/src/Square.php';  
require_once __DIR__ . '/src/Triangle.php';  
  
echo "==== Bridge Pattern Example ====<br>";  
  
$vectorRenderer = new VectorRenderer();  
$rasterRenderer = new RasterRenderer();  
  
$circleVector = new Circle($vectorRenderer);
```



```

$circleRaster = new Circle($rasterRenderer);

$squareVector = new Square($vectorRenderer);
$squareRaster = new Square($rasterRenderer);

$triangleVector = new Triangle($vectorRenderer);
$triangleRaster = new Triangle($rasterRenderer);

$circleVector->draw();
$circleRaster->draw();

$squareVector->draw();
$squareRaster->draw();

$triangleVector->draw();
$triangleRaster->draw();

if ($auth1->authenticate("admin", "1234")) {
    echo "Authentication successful\n";
} else {
    echo "Authentication failed\n";
}

```

інтерфейс Renderer:

```

<?php

interface Renderer {
    public function render($shapeName);
}

```

клас VectorRenderer:

```

<?php

require_once __DIR__ . '/Renderer.php';

class VectorRenderer implements Renderer
{
    public function render($shapeName)
    {
        echo "Drawing $shapeName as vector graphics<br>";
    }
}

```

клас RasterRenderer:

```
<?php

require_once __DIR__ . '/Renderer.php';

class RasterRenderer implements Renderer {
    public function render($shapeName) {
        echo "Drawing $shapeName as pixels<br>";
    }
}
```

клас Circle:

```
<?php

require_once __DIR__ . '/Shape.php';

class Circle extends Shape {
    public function draw() {
        $this->renderer->render("Circle");
    }
}
```

клас Square:

```
<?php

require_once __DIR__ . '/Shape.php';

class Square extends Shape {
    public function draw() {
        $this->renderer->render("Square");
    }
}
```

клас Triangle:

```
<?php

require_once __DIR__ . '/Shape.php';

class Triangle extends Shape {
    public function draw() {
        $this->renderer->render("Triangle");
    }
}
```

клас Shape:

```
<?php

require_once __DIR__ . '/Renderer.php';

abstract class Shape {
    protected $renderer;

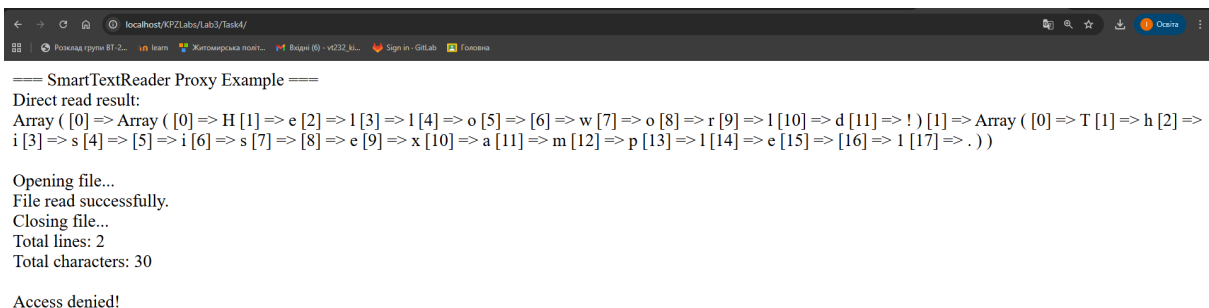
    public function __construct(Renderer $renderer) {
        $this->renderer = $renderer;
    }

    abstract public function draw();
}
```

Завдання 4: Проксі.

1. Створіть клас SmartTextReader, який вміє читати вміст текстового файлу і перетворювати його на двовірний масив якому зовнішній масив відповідає рядкам тексту, а вкладені масиви відповідають символам у відповідному рядку.
2. Створіть проксі для SmartTextReader з логуванням SmartTextChecker, який буде виводити інформацію про успішне відкриття, прочитання і закриття файлу, а також буде виводити загальну кількість рядків і символів у прочитаному тексті.
3. Створіть проксі для SmartTextReader з обмеженням доступу до певних файлів SmartTextReaderLocker. Цей клас в конструкторі приймає регулярний вираз, по якому лімітується доступ до певної групи файлів. Якщо клієнт викликатиме метод для прочитання такого лімітованого файлу, замість прочитання файлу в консоль має виводитися повідомлення "Access denied!".
4. Покажіть правильність роботи свого коду запустивши його в головному методі програми.

Результат виконання:

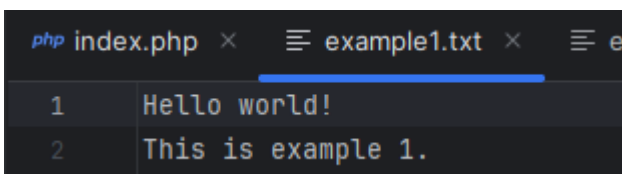


```
==== SmartTextReader Proxy Example ====
Direct read result:
Array ( [0] => Array ( [0] => H [1] => e [2] => l [3] => l [4] => o [5] => [6] => w [7] => o [8] => r [9] => l [10] => d [11] => ! ) [1] => Array ( [0] => T [1] => h [2] => i [3] => s [4] => [5] => i [6] => s [7] => [8] => e [9] => x [10] => a [11] => m [12] => p [13] => l [14] => e [15] => [16] => l [17] => . ) )

Opening file...
File read successfully.
Closing file...
Total lines: 2
Total characters: 30

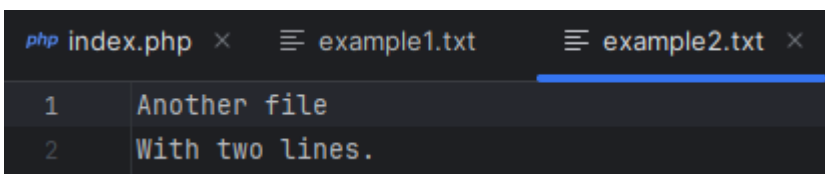
Access denied!
```

example1.txt:



```
php index.php x example1.txt x e
1 Hello world!
2 This is example 1.
```

example1.txt:



```
php index.php x example1.txt example2.txt x
1 Another file
2 With two lines.
```

index.php:

```
<?php
require_once __DIR__ . '/src/SmartTextReader.php';
require_once __DIR__ . '/src/SmartTextChecker.php';
require_once __DIR__ . '/src/SmartTextReaderLocker.php';

echo "=== SmartTextReader Proxy Example ===<br>";

file_put_contents("example1.txt", "Hello world!\nThis is example 1.");
file_put_contents("example2.txt", "Another file\nWith two lines.");

$reader1 = new SmartTextReader("example1.txt");
$data1 = $reader1->read();
echo "Direct read result:<br>";
print_r($data1);
echo "<br><br>";

$checker = new SmartTextChecker(new SmartTextReader("example1.txt"));
$data2 = $checker->read();
echo "<br>";

$lockedReader = new SmartTextReaderLocker(new
SmartTextReader("example2.txt"), "/example2/");
$lockedReader->read();

$allowedReader = new SmartTextReaderLocker(new
SmartTextReader("example1.txt"), "/example2/");
$data3 = $allowedReader->read();
```

клас SmartTextReader:

```
<?php

class SmartTextReader {
    private $filePath;

    public function __construct($filePath) {
        $this->filePath = $filePath;
    }

    public function read() {
        if (!file_exists($this->filePath)) {
            throw new Exception("File not found: " . $this->filePath);
        }
        $lines = file($this->filePath, FILE_IGNORE_NEW_LINES);
        $result = [];

        foreach ($lines as $line) {
            $result[] = str_split($line);
        }
        return $result;
    }
}
```

клас SmartTextChecker:

```
<?php

require_once __DIR__ . '/SmartTextReader.php';

class SmartTextChecker {
    private $reader;

    public function __construct(SmartTextReader $reader) {
        $this->reader = $reader;
    }

    public function read() {
        echo "Opening file...<br>";
        $data = $this->reader->read();
        echo "File read successfully.<br>";
        echo "Closing file...<br>";

        $lineCount = count($data);
        $charCount = 0;
        foreach ($data as $line) {
            $charCount += count($line);
        }

        echo "Total lines: $lineCount<br>";
        echo "Total characters: $charCount<br>";

        return $data;
    }
}
```

клас SmartTextReaderLocker:

```
<?php

require_once __DIR__ . '/SmartTextReader.php';

class SmartTextReaderLocker {
    private $reader;
    private $pattern;

    public function __construct(SmartTextReader $reader, $pattern) {
        $this->reader = $reader;
        $this->pattern = $pattern;
    }

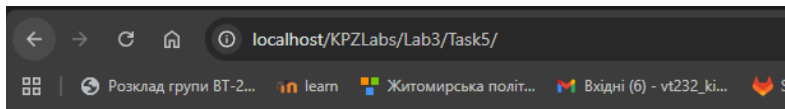
    public function read() {
        if (preg_match($this->pattern, $this->readerFileName())) {
            echo "Access denied!<br>";
            return null;
        }
        return $this->reader->read();
    }
}
```

```
private function readerFileName() {  
    $reflection = new ReflectionClass($this->reader);  
    $prop = $reflection->getProperty('filePath');  
    $prop->setAccessible(true);  
    return $prop->getValue($this->reader);  
}  
}
```

Завдання 5: Компонувальник.

1. Вам потрібно створити власну мову розмітки LightHTML.
2. Кожен елемент розмітки має наслідувати клас LightNode.
3. Створіть два дочірніх класи від LightNode: LightElementNode, LightTextNode.
4. LightTextNode може містити лише текст.
5. LightElementNode може містити будь-які LightNode. LightElementNode повинен мати інформацію про назву тега, його тип відображення (блочний чи рядковий), тип закриття (одиничний тег, як `` чи з закриваючим тегом) список CSS класів, кількість дочірніх елементів, а також має бути можливість виводити на екран його outerHTML і innerHTML.
6. За допомогою своєї мови розмітки виведіть в консоль елемент сторінки на Ваш вибір (наприклад якусь таблицю, список тощо).
7. Покажіть правильність роботи свого коду запустивши його в головному методі програми.

Результат виконання:



==== LightHTML Example ====

OuterHTML:

- Перший елемент
- Другий елемент
- Третій елемент

InnerHTML:

- Перший елемент
- Другий елемент
- Третій елемент

Кількість дочірніх елементів ul: 3

index.php:

```
<?php
require_once __DIR__ . '/src/LightTextNode.php';
require_once __DIR__ . '/src/LightElementNode.php';

echo "==== LightHTML Example ====<br>";

$ul = new LightElementNode('ul', 'block', false, ['my-list']);

$li1 = new LightElementNode('li', 'block', false);
$li1->addChild(new LightTextNode("Перший елемент"));

$li2 = new LightElementNode('li', 'block', false);
$li2->addChild(new LightTextNode("Другий елемент"));

$li3 = new LightElementNode('li', 'block', false);
$li3->addChild(new LightTextNode("Третій елемент"));

$ul->addChild($li1);
$ul->addChild($li2);
$ul->addChild($li3);

echo "OuterHTML:<br>";
echo $ul->outerHTML() . "<br><br>";

echo "InnerHTML:<br>";
echo $ul->innerHTML() . "<br><br>";

echo "Кількість дочірніх елементів ul: " . $ul->childCount() . "<br>";
```


клас LightNode:

```
<?php

abstract class LightNode {
    abstract public function outerHTML();
    abstract public function innerHTML();
}
```

клас LightTextNode:

```
<?php

require_once __DIR__ . '/LightNode.php';

class LightTextNode extends LightNode {
    private $text;

    public function __construct($text) {
        $this->text = $text;
    }

    public function outerHTML() {
        return htmlspecialchars($this->text);
    }

    public function innerHTML() {
        return htmlspecialchars($this->text);
    }
}
```

класс LightElementNode:

```
<?php

require_once __DIR__ . '/LightNode.php';

class LightElementNode extends LightNode {
    private $tagName;
    private $displayType;
    private $selfClosing;
    private $classes = [];
    private $children = [];

    public function __construct($tagName, $displayType = 'block', $selfClosing
= false, $classes = []) {
        $this->tagName = $tagName;
        $this->displayType = $displayType;
        $this->selfClosing = $selfClosing;
        $this->classes = $classes;
    }

    public function addChild(LightNode $child) {
        if ($this->selfClosing) {
            throw new Exception("Self-closing tag <{$this->tagName}/> cannot
have children");
        }
        $this->children[] = $child;
    }

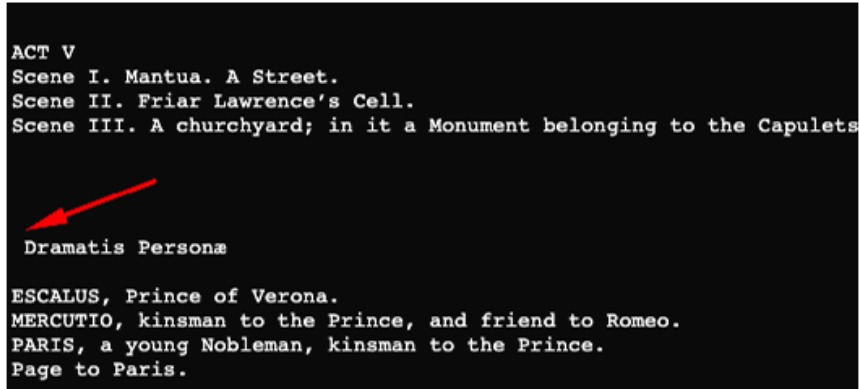
    public function childCount() {
        return count($this->children);
    }

    public function innerHTML() {
        $html = '';
        foreach ($this->children as $child) {
            $html .= $child->outerHTML();
        }
        return $html;
    }

    public function outerHTML() {
        $classAttr = !empty($this->classes) ? ' class="' . implode(' ',
$this->classes) . '"' : '';
        if ($this->selfClosing) {
            return "<{$this->tagName}{$classAttr}/>";
        } else {
            return
"<{$this->tagName}{$classAttr}>{$this->innerHTML()}</{$this->tagName}>";
        }
    }
}
```

Завдання 6: Легковаговик.

1. За допомогою свого LightHTML з завдання 1 перетворіть текст [книги](#) в HTML верстку за такими правилами:
 - a. Перший рядок має бути елементом `<h1>`
 - b. Якщо в рядку менше 20 символів - це має бути елемент `<h2>`
 - c. Якщо рядок починається з пробільного символу - це має бути `<blockquote>`.



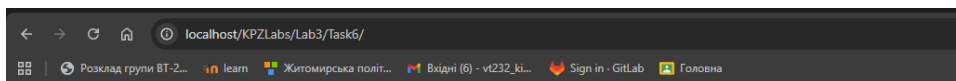
```
ACT V
Scene I. Mantua. A Street.
Scene II. Friar Lawrence's Cell.
Scene III. A churchyard; in it a Monument belonging to the Capulets

Dramatis Personæ

ESCALUS, Prince of Verona.
MERCUTIO, kinsman to the Prince, and friend to Romeo.
PARIS, a young Nobleman, kinsman to the Prince.
Page to Paris.
```

- d. В будь-якому іншому випадку - елемент `<p>`
2. Покажіть скільки займає все дерево Вашої верстки, коли воно повністю утримується в пам'яті процесу.
 3. Використайте Легковаговик на Ваших класах HTML елементів, щоб зменшити споживання пам'яті.
 4. Покажіть правильність роботи свого коду запустивши його в головному методі програми.

Результат виконання:



Act V

Scene I. Mantua. A Street.

Scene II. Friar Lawrence's Cell.

Scene III. A churchyard; in it a Monument belonging to the Capulets

Dramatis Personæ

ESCALUS, Prince of Verona.

MERCUTIO, kinsman to the Prince, and friend to Romeo.

PARIS, a young Nobleman, kinsman to the Prince.

Page to Paris.

index.php:

```
<?php
require_once __DIR__ . '/src/LightTextNode.php';
require_once __DIR__ . '/src/LightElementNode.php';
require_once __DIR__ . '/src/LightFlyweightFactory.php';

echo "=== LightHTML Book Renderer ===<br>";

$bookText = <<<EOT
Act V
Scene I. Mantua. A Street.
Scene II. Friar Lawrence's Cell.
Scene III. A churchyard; in it a Monument belonging to the Capulets

Dramatis Personæ

ESCALUS, Prince of Verona.
MERCUTIO, kinsman to the Prince, and friend to Romeo.
PARIS, a young Nobleman, kinsman to the Prince.
Page to Paris.
EOT;

$lines = explode("\n", $bookText);

$root = new LightElementNode('div', 'block', false, ['book']);

foreach ($lines as $i => $line) {
    $line = rtrim($line);
    $node = null;

    if ($i === 0) {
        $tag = LightFlyweightFactory::getTag('h1');
        $node = new LightElementNode($tag, 'block');
    } elseif (strlen($line) < 20) {
        $tag = LightFlyweightFactory::getTag('h2');
        $node = new LightElementNode($tag, 'block');
    } elseif (preg_match('/^\s/', $line)) {
        $tag = LightFlyweightFactory::getTag('blockquote');
        $node = new LightElementNode($tag, 'block');
    } else {
        $tag = LightFlyweightFactory::getTag('p');
        $node = new LightElementNode($tag, 'block');
    }

    $node->addChild(new LightTextNode(trim($line)));
    $root->addChild($node);
}

echo $root->outerHTML() . "<br><br>";

echo "Total children in root: " . $root->childCount() . "<br>";

echo "Memory usage: " . memory_get_usage() . " bytes<br>";
```

клас LightFlyweightFactory:

```
<?php

class LightFlyweightFactory {
    private static $tags = [];

    public static function getTag($tagName) {
        if (!isset(self::$tags[$tagName])) {
            self::$tags[$tagName] = $tagName;
        }
        return self::$tags[$tagName];
    }
}
```

клас LightTextNode:

```
<?php

require_once __DIR__ . '/LightNode.php';

class LightTextNode extends LightNode {
    private $text;

    public function __construct($text) {
        $this->text = $text;
    }

    public function outerHTML() {
        return htmlspecialchars($this->text);
    }

    public function innerHTML() {
        return htmlspecialchars($this->text);
    }
}
```

клас LightElementNode:

```
<?php

require_once __DIR__ . '/LightNode.php';

class LightElementNode extends LightNode {
    private $tagName;
    private $displayType;
    private $selfClosing;
    private $classes = [];
    private $children = [];
```

```

    public function __construct($tagName, $displayType = 'block', $selfClosing
= false, $classes = []) {
        $this->tagName = $tagName;
        $this->displayType = $displayType;
        $this->selfClosing = $selfClosing;
        $this->classes = $classes;
    }

    public function addChild(LightNode $child) {
        if ($this->selfClosing) {
            throw new Exception("Self-closing tag <{$this->tagName}/> cannot
have children");
        }
        $this->children[] = $child;
    }

    public function childCount() {
        return count($this->children);
    }

    public function innerHTML() {
        $html = '';
        foreach ($this->children as $child) {
            $html .= $child->outerHTML();
        }
        return $html;
    }

    public function outerHTML() {
        $classAttr = !empty($this->classes) ? ' class="' . implode(' ',
$this->classes) . '"' : '';
        if ($this->selfClosing) {
            return "<{$this->tagName}{$classAttr}/>";
        } else {
            return
"<{$this->tagName}{$classAttr}>{$this->innerHTML()}</{$this->tagName}>";
        }
    }
}

```

Посилання на репозиторій: <https://github.com/KuzminIvan232/KPZLabs>