Лабораторна робота №4

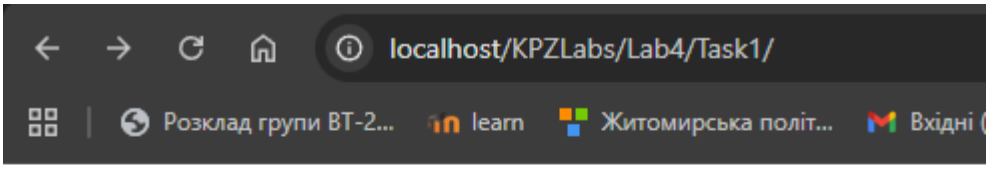
Тема: Поведінкові шаблони

Мета роботи: навчитися реалізовувати структурні шаблони проєктування Ланцюжок відповідальностей, Посередник, Спостерігач, Стратегія, Мементо

Завдання 1: Ланцюжок відповідальностей.

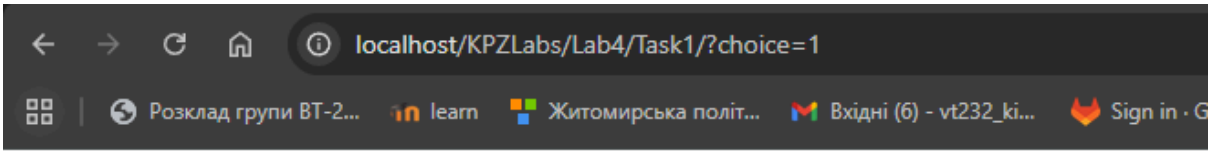
1. Створіть систему підтримки користувачів.
2. Вона має функціонувати так, як покрокове меню (наприклад, коли телефонуєте до оператора мобільного зв'язку). Але замість голосових повідомлень Ви маєте виводити відповідні повідомлення в консоль і чекати на вибір користувача.
3. Система повинна мати мінімум 4 рівні. Всі питання мають на меті обрати правильний рівень підтримки (тобто Handler) для користувача. Тобто вже на першому питанні може бути підібрано правильний рівень підтримки, тоді меню закінчується. А може бути так, що на жодному питанні не буде знайдено правильний рівень (Handler), тоді меню має повторитися.
4. Покажіть правильність роботи свого коду запустивши його в головному методі програми.

Результат виконання:



User Support Menu

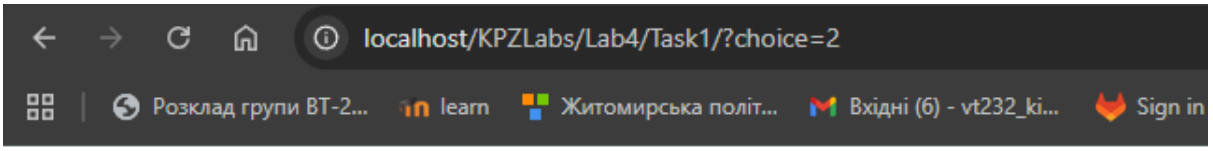
- 1. Basic info
- 2. Technical issues
- 3. Advanced troubleshooting
- 4. Specialist support



User Support Menu

- 1. Basic info
- 2. Technical issues
- 3. Advanced troubleshooting
- 4. Specialist support

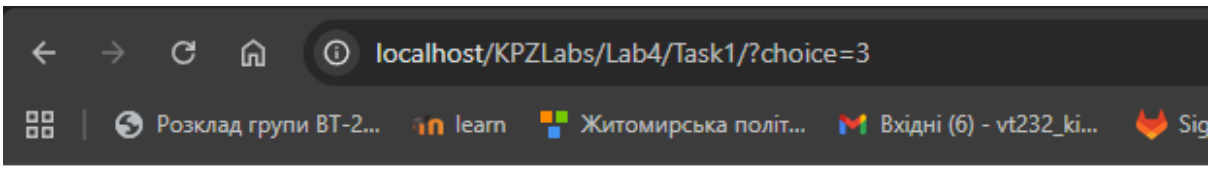
Result: Level 1 Support: Basic information provided.



User Support Menu

- 1. Basic info
- 2. Technical issues
- 3. Advanced troubleshooting
- 4. Specialist support

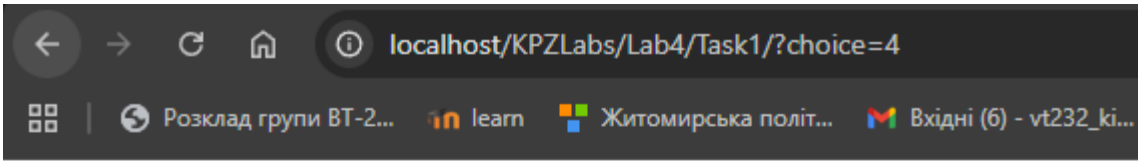
Result: Level 2 Support: Technical issue assistance.



User Support Menu

- 1. Basic info
- 2. Technical issues
- 3. Advanced troubleshooting
- 4. Specialist support

Result: Level 3 Support: Advanced troubleshooting.



User Support Menu

- 1. Basic info
- 2. Technical issues
- 3. Advanced troubleshooting
- 4. Specialist support

Result: Level 4 Support: Specialist connected.

index.php:

```php
<?php
require_once __DIR__ . '/handlers/Level1Support.php';
require_once __DIR__ . '/handlers/Level2Support.php';
require_once __DIR__ . '/handlers/Level3Support.php';
require_once __DIR__ . '/handlers/Level4Support.php';

$level1 = new Level1Support();
$level2 = new Level2Support();
$level3 = new Level3Support();
$level4 = new Level4Support();

$level1->setNext($level2)->setNext($level3)->setNext($level4);

$choice = isset($_GET['choice']) ? intval($_GET['choice']) : 0;

echo "<h2>User Support Menu</h2>";
echo "<ul>
<li><a href='?choice=1'>1. Basic info</a></li>
<li><a href='?choice=2'>2. Technical issues</a></li>
<li><a href='?choice=3'>3. Advanced troubleshooting</a></li>
<li><a href='?choice=4'>4. Specialist support</a></li>
</ul>";

if ($choice > 0) {
   $result = $level1->handle($choice);
   echo "<p><strong>Result:</strong> " . ($result ?? "Invalid choice") .
"</p>";
}
```

інтерфейс Handler:

```php
<?php

interface Handler {
   public function setNext(Handler $handler): Handler;
   public function handle(int $choice): ?string;
}
```

клас AbstractHandler:

```php
<?php

require_once __DIR__ . '/../interfaces/Handler.php';

abstract class AbstractHandler implements Handler {
   private ?Handler $next = null;

   public function setNext(Handler $handler): Handler {
       $this->next = $handler;
       return $handler;
   }
```

```
    public function handle(int $choice): ?string {
        if ($this->next) {
            return $this->next->handle($choice);
        }
        return null;
    }
}
```

клас Level1Support:

```
<?php

require_once __DIR__ . '/../core/AbstractHandler.php';

class Level1Support extends AbstractHandler {
    public function handle(int $choice): ?string {
        if ($choice === 1) {
            return "Level 1 Support: Basic information provided.";
        }
        return parent::handle($choice);
    }
}
```

клас Level2Support:

```
<?php

require_once __DIR__ . '/../core/AbstractHandler.php';

class Level2Support extends AbstractHandler {
    public function handle(int $choice): ?string {
        if ($choice === 2) {
            return "Level 2 Support: Technical issue assistance.";
        }
        return parent::handle($choice);
    }
}
```

клас Level3Support:

```
<?php

require_once __DIR__ . '/../core/AbstractHandler.php';

class Level3Support extends AbstractHandler {
```

```
    public function handle(int $choice): ?string {
        if ($choice === 3) {
            return "Level 3 Support: Advanced troubleshooting.";
        }
        return parent::handle($choice);
    }
}
```

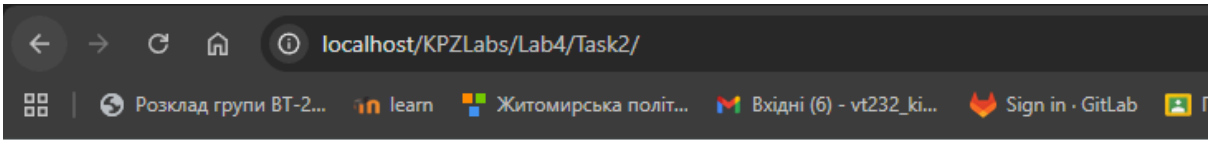
клас Level4Support:

```
<?php

require_once __DIR__ . '/../core/AbstractHandler.php';

class Level4Support extends AbstractHandler {
    public function handle(int $choice): ?string {
        if ($choice === 4) {
            return "Level 4 Support: Specialist connected.";
        }
        return parent::handle($choice);
    }
}
```

Завдання 2: Посередник.

1. Відрефакторте код продемонстрований на лекції за допомогою використання шаблону Посередник.
2. В результаті рефакторингу Aircraft не повинен “знати” про Runway і навпаки. Обидві сутності повинні “знати” лише про CommandCentre.
3. Покажіть правильність роботи свого коду запустивши його в головному методі програми.

Результат виконання:



Aircraft Boeing 737 requests landing...
Aircraft Boeing 737 has landed on Runway_68ad90326b36c.
Runway Runway_68ad90326b36c is busy!
Aircraft Airbus A320 requests landing...
Aircraft Airbus A320 has landed on Runway_68ad90326b371.
Runway Runway_68ad90326b371 is busy!
Aircraft Boeing 737 requests take-off...
Runway Runway_68ad90326b36c is free!
Aircraft Boeing 737 has taken off.
Aircraft Airbus A320 requests take-off...
Runway Runway_68ad90326b371 is free!
Aircraft Airbus A320 has taken off.

index.php:

```php
<?php
require_once __DIR__ . '/mediator/Aircraft.php';
require_once __DIR__ . '/mediator/Runway.php';
require_once __DIR__ . '/mediator/CommandCentre.php';

use mediator\Aircraft;
use mediator\Runway;
use mediator\CommandCentre;

$runway1 = new Runway();
$runway2 = new Runway();

$plane1 = new Aircraft("Boeing 737");
```

```php
$plane2 = new Aircraft("Airbus A320");

$centre = new CommandCentre([$runway1, $runway2]);

$plane1->setCommandCentre($centre);
$plane2->setCommandCentre($centre);

$plane1->requestLanding();
$plane2->requestLanding();
$plane1->requestTakeOff();
$plane2->requestTakeOff();
```

клас CommandCentre:

```php
<?php

namespace mediator;

class CommandCentre
{
   private array $runways = [];

   public function __construct(array $runways)
   {
       $this->runways = $runways;
   }
   public function handleLanding(Aircraft $aircraft): void
   {
       echo "Aircraft {$aircraft->getName()} requests landing...<br>";
       foreach ($this->runways as $runway) {
           if ($runway->isFree()) {
               echo "Aircraft {$aircraft->getName()} has landed on
{$runway->getId()}.<br>";
               $runway->occupy($aircraft);
               return;
           }
       }
       echo "No free runway for {$aircraft->getName()}!<br>";
   }
   public function handleTakeOff(Aircraft $aircraft): void
   {
       echo "Aircraft {$aircraft->getName()} requests take-off...<br>";
       foreach ($this->runways as $runway) {
           if (!$runway->isFree()) {
               $runway->release();
               echo "Aircraft {$aircraft->getName()} has taken off.<br>";
               $aircraft->markTakingOff(true);
               return;
           }
       }
       echo "No runway to release for {$aircraft->getName()}!<br>";
   }
}
```

клас Aircraft:

```php
<?php

namespace mediator;

class Aircraft
{
    private string $name;
    private bool $isTakingOff = false;
    private ?CommandCentre $centre = null;

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

    public function setCommandCentre(CommandCentre $centre): void
    {
        $this->centre = $centre;
    }

    public function getName(): string
    {
        return $this->name;
    }

    public function isTakingOff(): bool
    {
        return $this->isTakingOff;
    }

    public function requestLanding(): void
    {
        if ($this->centre) {
            $this->centre->handleLanding($this);
        }
    }

    public function requestTakeOff(): void
    {
        if ($this->centre) {
            $this->centre->handleTakeOff($this);
        }
    }

    public function markTakingOff(bool $status): void
    {
        $this->isTakingOff = $status;
    }
}
```

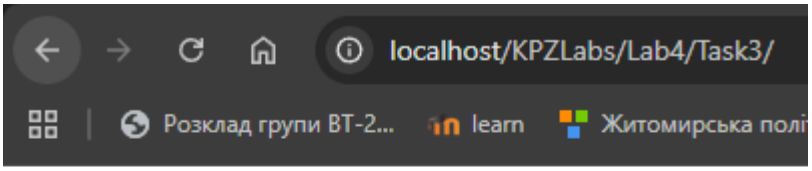
клас Runway:

```php
<?php

namespace mediator;

class Runway
{
    private string $id;
    private ?Aircraft $aircraft = null;

    public function __construct()
    {
        $this->id = uniqid("Runway_");
    }

    public function getId(): string
    {
        return $this->id;
    }

    public function isFree(): bool
    {
        return $this->aircraft === null;
    }

    public function occupy(Aircraft $aircraft): void
    {
        $this->aircraft = $aircraft;
        $this->highlightRed();
    }

    public function release(): void
    {
        $this->aircraft = null;
        $this->highlightGreen();
    }

    private function highlightRed(): void
    {
        echo "Runway {$this->id} is busy!<br>";
    }

    private function highlightGreen(): void
    {
        echo "Runway {$this->id} is free!<br>";
    }
}
```

Завдання 3: Спостерігач.

1. Додайте до Вашого LightHTML з завдання 5 ЛР №3 можливість додавання EventListener до Ваших HTML елементів.
2. Додайте можливість підписки не різні івенти (“click”, “mouseover” тощо).
3. Покажіть правильність роботи свого коду запустивши його в головному методі програми.
4. **Це завдання виконайте в окремому PR.**

Результат виконання:



HTML Output:

Event Simulation:
Clicked on element!
Mouse over element!

index.php:

```php
<?php

require_once __DIR__ . '/LightHTML/Element.php';
require_once __DIR__ . '/LightHTML/EventListener.php';
require_once __DIR__ . '/LightHTML/Document.php';

use LightHTML\Element;
use LightHTML\EventListener;
use LightHTML\Document;

class ClickListener implements EventListener {
    public function handle(Element $element): void {
        echo "Clicked on element!<br>";
    }
}

class MouseOverListener implements EventListener {
    public function handle(Element $element): void {
        echo "Mouse over element!<br>";
    }
```

```
}
$button = new Element("button");
$button->setAttribute("id", "btn1")
   ->setAttribute("class", "btn")
   ->addEventListener("click", new ClickListener())
   ->addEventListener("mouseover", new MouseOverListener());

$doc = new Document();
$doc->addElement($button);

echo "HTML Output:<br>";
echo $doc->render();

echo "<br>Event Simulation:<br>";
$button->triggerEvent("click");
$button->triggerEvent("mouseover");
```

інтерфейс EventListener:

```
<?php

namespace LightHTML;

interface EventListener
{
   public function handle(Element $element): void;
}
```

клас Element:

```
<?php

namespace LightHTML;

class Element
{
   private string $tag;
   private array $attributes = [];
   private array $children = [];
   private array $eventListeners = [];

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

   public function setAttribute(string $name, string $value): self
   {
       $this->attributes[$name] = $value;
       return $this;
   }

   public function addChild(Element $child): self
```

```
    {
        $this->children[] = $child;
        return $this;
    }

    public function addEventListener(string $event, EventListener $listener):
self
    {
        if (!isset($this->eventListeners[$event])) {
            $this->eventListeners[$event] = [];
        }
        $this->eventListeners[$event][] = $listener;
        return $this;
    }

    public function triggerEvent(string $event): void
    {
        if (isset($this->eventListeners[$event])) {
            foreach ($this->eventListeners[$event] as $listener) {
                $listener->handle($this);
            }
        }
    }

    public function render(): string
    {
        $attrs = '';
        foreach ($this->attributes as $k => $v) {
            $attrs .= " $k=\"$v\"";
        }

        $html = "<{$this->tag}$attrs>";
        foreach ($this->children as $child) {
            $html .= $child->render();
        }
        $html .= "</{$this->tag}>";

        return $html;
    }
}
```

клас Document:

```
<?php

namespace LightHTML;

class Document
{
    private array $elements = [];

    public function addElement(Element $element): self
    {
```

Завдання 4: Стратегія.

1. Додайте до Вашого LightHTML з завдання 5 ЛР №3 новий елемент Image, який за допомогою стратегії в залежності від переданого href буде завантажувати картинку або з файлової системи або з мережі.
2. Покажіть правильність роботи свого коду запустивши його в головному методі програми.
3. **Це завдання виконайте в окремому PR.**

Результат виконання:

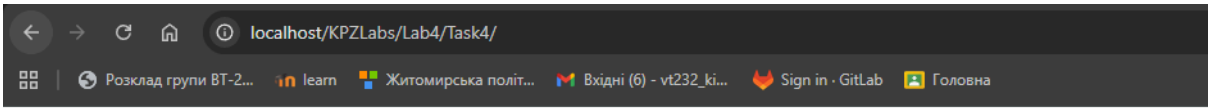


Image element → Loading image from file: images/local_pic.png

Image element → Loading image from network: http://example.com/picture.jpg

index.php:

```php
<?php

require_once __DIR__ . '/LightHTML/HTMLElement.php';
require_once __DIR__ . '/LightHTML/Image.php';
require_once __DIR__ . '/Strategies/ImageLoaderStrategy.php';
require_once __DIR__ . '/Strategies/FileImageLoader.php';
require_once __DIR__ . '/Strategies/NetworkImageLoader.php';

use LightHTML\Image;

$fileImage = new Image("images/local_pic.png");
echo $fileImage->render();

$netImage = new Image("http://example.com/picture.jpg");
echo $netImage->render();
```

клас HTMLElement:

```php
<?php

namespace LightHTML;

abstract class HTMLElement
{
    protected $tag;
    protected $content = '';

    public function __construct($tag, $content = '')
```

```
    {
        $this->tag = $tag;
        $this->content = $content;
    }

    public function render()
    {
        return "<{$this->tag}>{$this->content}</{$this->tag}>";
    }
}
```

клас Image:

```
<?php

namespace LightHTML;

use Strategies\ImageLoaderStrategy;
use Strategies\FileImageLoader;
use Strategies\NetworkImageLoader;

class Image extends HTMLElement
{
    private $href;
    private $loader;

    public function __construct($href)
    {
        parent::__construct("img");
        $this->href = $href;

        if (strpos($href, "http") === 0) {
            $this->loader = new NetworkImageLoader();
        } else {
            $this->loader = new FileImageLoader();
        }
    }
    public function render()
    {
        $result = $this->loader->load($this->href);
        return "<br>Image element → {$result}<br>";
    }
}
```

інтерфейс ImageLoaderStrategy:

```
<?php

namespace Strategies;

interface ImageLoaderStrategy
{
    public function load($href): string;
}
```

клас FileImageLoader:

```php
<?php

namespace Strategies;

class FileImageLoader implements ImageLoaderStrategy
{
   public function load($href): string
   {
       return "Loading image from file: {$href}";
   }
}
```

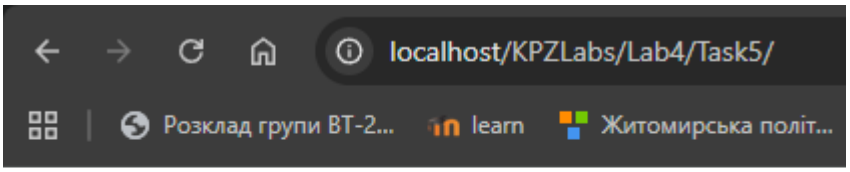
клас NetworkImageLoader:

```php
<?php

namespace Strategies;

class NetworkImageLoader implements ImageLoaderStrategy
{
   public function load($href): string
   {
       return "Loading image from network: {$href}";
   }
}
```

Завдання 5: Мементо.

1. Уявіть, що ви створюєте програму текстового редактора.
2. Для цього завдання буде достатньо POC-версії: клас `TextEditor`, який містить в собі поточну версію текстового документу, який представлений спеціальним класом `TextDocument`. Назви класам можете давати власні.
3. Реалізуйте функцію збереження і скасування, яка дозволить користувачам відмінити зміни, зроблені в документі. Завдання полягає в тому, щоб зберегти стан документа в різні моменти часу та відновити його, коли це необхідно, не розкриваючи внутрішню реалізацію документа чи текстового редактору.

Результат виконання:



Current: Hello World!!!
Undo 1: Hello World
Undo 2: Hello

index.php:

```php
<?php

require_once __DIR__ . '/Models/TextDocument.php';
require_once __DIR__ . '/Models/DocumentMemento.php';
require_once __DIR__ . '/Models/TextEditor.php';
require_once __DIR__ . '/Services/History.php';

use Models\TextDocument;
use Models\TextEditor;
use Services\History;

$document = new TextDocument();
$editor = new TextEditor($document);
$history = new History();

$editor->write("Hello");
$history->push($editor->save());

$editor->write(" World");
$history->push($editor->save());
```

```php
$editor->write("!!!");

echo "Current: " . $editor->getContent() . "<br>";

$editor->restore($history->pop());
echo "Undo 1: " . $editor->getContent() . "<br>";

$editor->restore($history->pop());
echo "Undo 2: " . $editor->getContent() . "<br>";
```

клас TextDocument:

```php
<?php

namespace Models;

class TextDocument
{
    private $content;
    public function __construct($content = "")
    {
        $this->content = $content;
    }
    public function setContent($content)
    {
        $this->content = $content;
    }
    public function getContent()
    {
        return $this->content;
    }
}
```

клас DocumentMemento:

```php
<?php

namespace Models;
class DocumentMemento
{
    private $content;
    public function __construct($content)
    {
        $this->content = $content;
    }
    public function getSavedContent()
    {
        return $this->content;
    }
}
```

клас TextEditor:

```php
<?php

namespace Models;

class TextEditor
{
    private $document;
    public function __construct(TextDocument $document)
    {
        $this->document = $document;
    }
    public function write($text)
    {
        $this->document->setContent($this->document->getContent() . $text);
    }
    public function getContent()
    {
        return $this->document->getContent();
    }
    public function save(): DocumentMemento
    {
        return new DocumentMemento($this->document->getContent());
    }
    public function restore(DocumentMemento $memento)
    {
        $this->document->setContent($memento->getSavedContent());
    }
}
```

клас History:

```php
<?php

namespace Services;

use Models\DocumentMemento;

class History
{
    private $mementos = [];
    public function push(DocumentMemento $memento)
    {
        $this->mementos[] = $memento;
    }
    public function pop(): ?DocumentMemento
    {
        return array_pop($this->mementos);
    }
```

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