Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Кафедра програмної інженерії

Лабораторна робота №1

з дисципліни «Скриптові мови програмування»

з теми: «ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ НА PHP»

Виконали:

ст. групи ПЗПІ-20-6

Мацак Станіслав Олегович

Масенко Ярослав Володимирович

Падалка Артем Борисович

24 квітня 2022 р.

Перевірила:

к.т.н., доцент кафедри ПІ

Груздо І. В.

Харків 2022

Тема web-застосунку: онлайн-магазин з продажу дизайнерських ножів.

Мета роботи: отримання практичних навичок роботи з ООП в РНР.

Постановка завдання: оглянути об’єктно-орієнтовану парадигму програмування на PHP та створити базовий простий функціонал серверної частини обраного web-застосунку.

Детальна постановка завдання:

Завдання перше:

1) Створити абстрактний клас Page.php із методами, що дозволяють відображати частини сторінки

Завдання друге:

1) Створити світлу та темну модифікацію головної сторінки;

2) Додати sidebar на одну з модифікацій.

Завдання третє:

1) Створити конструктор для абстрактного класу Page.php та для його наслідників HomePage.php, BasketPage.php.

Завдання четверте:

1) Реалізувати в наслідниках абстрактний метод getMain(), що повертає головну частину сторінки;

2) Створити методи getHeader(), getBody(), getFooter().

Завдання п’яте:

1) Успадкувати абстрактний клас Page.php створеними класами-наслідниками;

2) Перевірити реалізацію усіх абстрактних методів базового класу.

Завдання шосте:

1) Додати властивість, що зберігає в собі повний рядок html-розмітки сторінок.

Завдання сьоме:

1) Розробити методи видалення та додавання товарів у сесії.

2) Перевірити правильність додавання та видалення товарів із кошику.

Хід роботи:

*1. Створити клас, що описує web-сторінку відповідно до обраної теми або до теми за варіантом. Клас містить функції, що відображають різні її частини.*

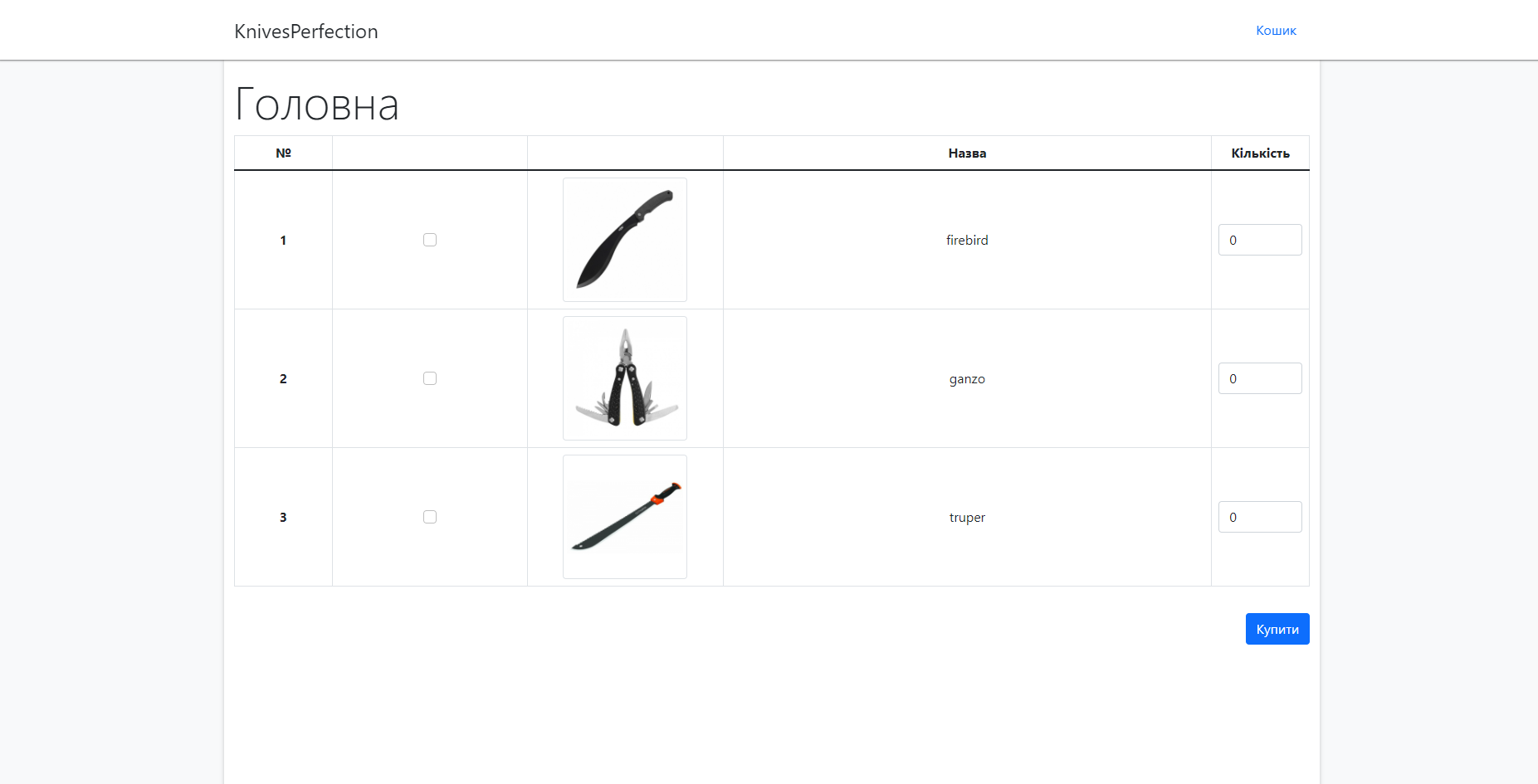
Створено клас Page.php, який є базовим абстрактним класом сторінки. Це означає, що ми не можемо створити об’єкту цього класу. Він має методи для повернення блоків <head>…</head> (getHead()), оскільки більшість сторінок передбачають однакову «голову» сторінки. Натомість «тіло» сторінок завжди відрізняється, тому функція getMain() є абстрактною. Також є функції getHeader() та getFooter(), оскільки ці частини блоку <body>…</body> теж часто повторюються та можуть використовуватись у наслідників. Одночасний виклик функцій getHeader(), getMain(), getFooter() реалізований у функції getBody(), результатом якої є повністю сформоване тіло сторінки.

Код цього класу представлений нижче.

**<?php  
abstract class Page  
{  
 private string $pageContent;  
 private string $title;  
   
 function \_\_construct($title){  
 $this->title = $title;  
 $this->pageContent = $this->getHead().$this->getBody();  
 }  
  
 function getHead() : string{  
 return '<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1">  
 <meta http-equiv="X-UA-Compatible" content="ie=edge">  
 <link href="styles/bootstrap.css" rel="stylesheet" type="text/css">  
 <link href="styles/style.css" rel="stylesheet" type="text/css">  
 <title>'.$this->title.'</title>  
 '.$this->getScripts().'  
 </head>';  
 }  
  
 function getScripts() : string{  
 return "";  
 }  
  
 function getBody() : string{  
 return '<body>'.$this->getHeader().$this->getMain().$this->getFooter().'</body>';  
 }  
   
 abstract function getMain() : string;  
  
 function getHeader() : string{  
 return '<div class="header container-fluid sticky-top bg-white">  
 <header class="container d-flex flex-wrap justify-content-center py-3">  
 <a href="/" class="d-flex align-items-center mb-3 mb-md-0 me-md-auto text-dark text-decoration-none">  
 <span class="fs-4">KnivesPerfection</span>  
 </a>  
 <a href="/basket.php" class="nav-link">Кошик</a>  
 </header>  
 </div>';  
 }  
  
 function getFooter() : string{  
 return '<div class="container-fluid bg-white text-center py-3 border-top">  
 2021  
 </div>';  
 }  
  
 function getPage(){  
 echo "<!doctype html><html lang = 'uk'>".$this->pageContent."</html>";  
 }  
}**

*2. Створити похідні від цього класу класи, що описують різні модифікації сторінки в межах теми.*

Головну сторінку сайту було створено у 2 модифікаціях – зі світлою та темною темою. Також, темна тема має додатковий функціонал – sidebar, що допомагає при навігації по сайту. Скриншоти та код різних модифікацій подані нижче.

Рисунок 1.1 – Головна сторінка сайту у світлій темі

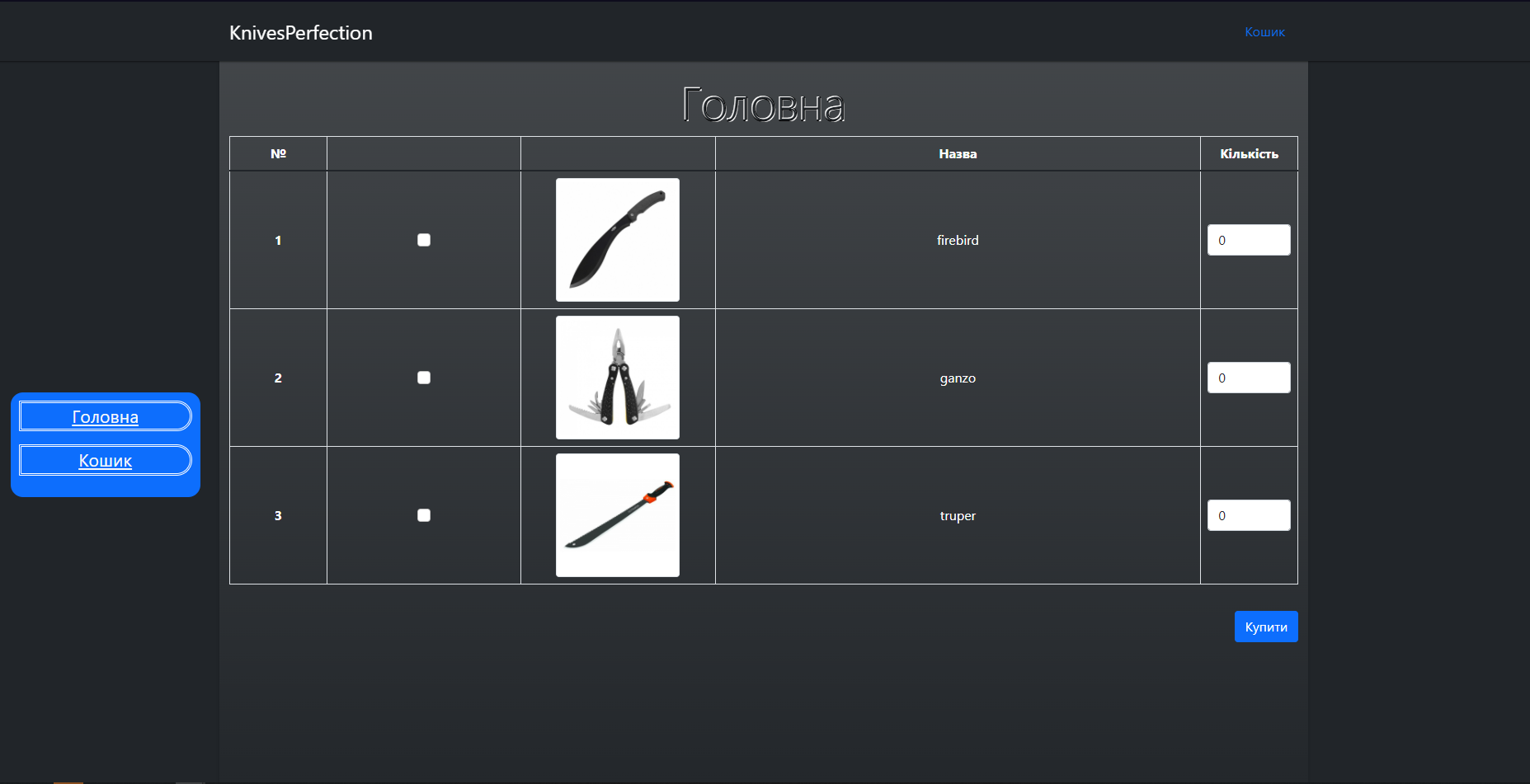


Рисунок 1.2 – Головна сторінка сайту в темній темі та з sidebar

Код світлої модифікації головної сторінки:

**<?php  
class HomePage extends Page{  
 function \_\_construct(){  
 parent::*\_\_construct*("Головна");  
 }  
  
 function getMain() : string{  
  
  
 $products = [["firebird", "/img/firebird.jpg"], ["ganzo", "/img/ganzo.jpg"], ["truper", "/img/truper.jpg"]];  
  
 $productsBlocks = "";  
 for ($i = 0; $i < count($products); $i ++){  
 $productsBlocks .= '<tr>  
 <th scope="row">'.($i+1).'</th>  
 <td>  
 <input type="checkbox" class="form-check-input" name="'.$i.'\_check">  
 </td>  
 <td>  
 <img src="'.$products[$i][1].'" alt="'.$products[$i][0].' picture" class="img-thumbnail img-small">  
 </td>  
 <td>  
 '.$products[$i][0].'  
 </td>  
 <td>  
 <input type="number" class="form-control" name="'.$i.'\_num" min="0" max="100" value="0">  
 </td>  
 </tr>';  
 }  
  
 return '<main class="bg-light">  
 <div class="container bg-white body\_content">  
 <h1 class="display-4">Головна</h1>  
 <form action="/buy.php" method="POST">  
 <table class="table table-bordered align-middle text-center">  
 <thead>  
 <tr>  
 <th scope="col" class="col-1">№</th>  
 <th scope="col" class="col-2"></th>  
 <th scope="col" class="col-2"></th>  
 <th scope="col" class="col-5">Назва</th>  
 <th scope="col" class="col-1">Кількість</th>  
 </tr>  
 </thead>  
 <tbody>  
 '.$productsBlocks.'  
 </tbody>  
 </table>  
 <div class="text-end py-3"><input type="submit" class="btn btn-primary" value="Купити"></div>  
 </form>  
 </div>  
 </main>';  
 }  
}**

Код темної модифікації головної сторінки з sidebar:

<?php  
class HomePageWithBar extends Page{  
 function \_\_construct(){  
 parent::*\_\_construct*("Головна");  
 }  
  
 function getHeader() : string{  
 return '<div class="header container-fluid sticky-top bg-dark">  
 <header class="container d-flex flex-wrap justify-content-center py-3">  
 <a href="/" class="d-flex align-items-center mb-3 mb-md-0 me-md-auto text-dark text-decoration-none">  
 <span class="fs-4" style="color: white">KnivesPerfection</span>  
 </a>  
 <a href="/basket.php" class="nav-link">Кошик</a>  
 </header>  
 </div>';  
 }  
  
 function getMain() : string{  
  
  
 $products = [["firebird", "/img/firebird.jpg"], ["ganzo", "/img/ganzo.jpg"], ["truper", "/img/truper.jpg"]];  
  
 $productsBlocks = "";  
 for ($i = 0; $i < count($products); $i ++){  
 $productsBlocks .= '<tr style="color: white">  
 <th scope="row">'.($i+1).'</th>  
 <td>  
 <input type="checkbox" class="form-check-input" name="'.$i.'\_check">  
 </td>  
 <td>  
 <img src="'.$products[$i][1].'" alt="'.$products[$i][0].' picture" class="img-thumbnail img-small">  
 </td>  
 <td>  
 '.$products[$i][0].'  
 </td>  
 <td>  
 <input type="number" class="form-control" name="'.$i.'\_num" min="0" max="100" value="0">  
 </td>  
 </tr>';  
 }  
  
 return '<main class="bg-dark">  
 <div class="container bg-gradient body\_content">  
 <h1 class="display-4" style="text-align: center; text-shadow: -1px -1px 0 white, 1px -1px 0 #000, -1px 1px 0 #000, 1px 1px 0 white;">Головна</h1>  
 <form action="/buy.php" method="POST">  
 <table class="table table-bordered align-middle text-center">  
 <thead>  
 <tr style="color: white">  
 <th scope="col" class="col-1">№</th>  
 <th scope="col" class="col-2"></th>  
 <th scope="col" class="col-2"></th>  
 <th scope="col" class="col-5">Назва</th>  
 <th scope="col" class="col-1">Кількість</th>  
 </tr>  
 </thead>  
 <tbody>  
 '.$productsBlocks.'  
 </tbody>  
 </table>  
 <div class="text-end py-3"><input type="submit" class="btn btn-primary" value="Купити"></div>  
 </form>  
 </div>'.$this->getBar().'  
 </main>';  
 }  
  
 function getBar(){  
 return '<aside>  
 <div style="position: fixed; background: #0d6efd; float: left; top: 50%;   
 margin-left: 1%; padding: 10px; align-items: center; width: 230px; text-align: center; font-size: 16pt; border-radius: 15px">  
 <p style="border-style: double; border-color: white; border-radius: 0px 50px 50px 0px;">  
 <a class="text-light" href="../index.php">Головна</a>  
 </p>  
 <p class=".text-light" style="border-style: double; border-color: white; border-radius: 0px 50px 50px 0px;">  
 <a class="text-light" href="../buy.php">Кошик</a>  
 </p>  
 </div>  
 </aside>';  
 }  
}

Також було створено сторінку з кошиком, що описується детально у пункті 4.

*3. Описати конструктор для будь-якого елемента для обраної теми сайту.*

Конструктор описано в базовому абстрактному класі сторінок:

function \_\_construct($title){  
 $this->title = $title;  
 $this->pageContent = $this->getHead().$this->getBody();  
}

Він приймає в якості параметра рядок $title, що змінює відображуваний у вкладці браузера напис. Також конструктор генерує вміст сторінки, що складається із блоків Head та Body та отримується шляхом виклику відповідних функцій. Нагадаємо, що метод getBody() містить виклик абстрактної функції getMain(), що має реалізуватись у кожному наслідуваному класі. Інакше метод та конструктор не спрацюють.

Виклик конструктора з класів-наслідників виглядає наступним чином:

function \_\_construct(){  
 parent::*\_\_construct*("Головна");  
}

(для головної сторінки);

function \_\_construct(){  
 parent::*\_\_construct*("Кошик");  
}

(для кошика).

*4. Продемонструвати можливість роботи з методами в РНР.*

В якості прикладу демонстрації роботи з методами в PHP обрано метод getMain() з класу, що генерує сторінку «Кошик» на сайті.

Код цього методу:

function getMain() : string{  
  
  
 $returnString = '  
 <script>  
 function Delete(i) {  
 if (confirm("Видалити товар з кошика?")) {  
 fetch("/delete.php",{  
 method: "POST",  
 body: JSON.stringify(  
 {  
 id: i  
 }  
 )   
 }).then((response) => {  
 return response.text();  
 }).then((html) => {  
 document.body.innerHTML = html   
 });  
 }  
 }  
 </script>  
 <main class="bg-light">  
  
 <div class="container bg-white body\_content">  
 <!---<div class="bg-white">ВИ В КОШИКУ (СМІТНИКУ, ХАХА)</div>!--->  
 <h1 class="display-4">Кошик</h1>  
 <table class="table table-bordered align-middle text-center">  
 <thead>  
 <tr>  
 <th scope="col" class="col-1">№</th>  
 <th scope="col" class="col-2"></th>  
 <th scope="col" class="col-5">Назва</th>  
 <th scope="col" class="col-1">Кількість</th>  
 <th scope="col" class="col-1"></th>  
 </tr>  
 </thead>  
 <tbody>  
 ';  
  
 $products = [["firebird", "/img/firebird.jpg"], ["ganzo", "/img/ganzo.jpg"], ["truper", "/img/truper.jpg"]];  
  
 if (count($\_SESSION) <= count($products)){  
 return $returnString.'<a href="/index.php">Перейти до покупок</a></tbody></table></div></main>';  
 }  
  
 for ($i = 0; $i < 3; $i++){  
  
 $i\_check = $i."\_check";  
 $i\_num = $i."\_num";  
  
 if (isset($\_SESSION[$i\_check]) && $\_SESSION[$i\_check] == "on"){  
  
 $num = 0;  
 if (isset($\_SESSION[$i\_num])) {  
 $num = $\_SESSION[$i\_num];  
 }  
  
 $returnString .='<tr>  
 <th scope="row">'.($i+1).'</th>  
 <td>  
 <img src="'.$products[$i][1].'" alt="'.$products[$i][0].' picture" class="img-thumbnail img-small">  
 </td>  
 <td>  
 '.$products[$i][0].'  
 </td>  
 <td>  
 '.$num.'  
 </td>  
 <td>  
 <button type="button" class="btn btn-danger" name="'.$i.'-button" onclick="Delete('.$i.'***)***">  
 <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor" viewBox="0 0 16 16">  
 <path d="M2 0a2 2 0 0 0-2 2v12a2 2 0 0 0 2 2h12a2 2 0 0 0 2-2V2a2 2 0 0 0-2-2H2zm3.354 4.646L8 7.293l2.646-2.647a.5.5 0 0 1 .708.708L8.707 8l2.647 2.646a.5.5 0 0 1-.708.708L8 8.707l-2.646 2.647a.5.5 0 0 1-.708-.708L7.293 8 4.646 5.354a.5.5 0 1 1 .708-.708z"/>  
 </svg>  
 </button>  
 </td>  
 </tr>';  
 }  
 }  
 return $returnString.'</tbody></table></div></main>';  
}

Задекларовано, що функція повертає рядкове значення.

Тіло методу генерує рядок, що буде повернено методом. Зокрема, створюється шапка таблиці з назвою колонок. Потім кожен продукт з масиву (в майбутньому – бази даних) перебирається та перевіряється, чи був він попередньо в сесії доданий. Якщо хоча б один продукт в сесії додавався, то змінна $basketEmpty стане false, що зламає перевірку після закнічення циклу. Якщо ж ця перевірка пройде, то продуктів у сесії не додавалось (або вони вже видалені), після чого з’являється напис «Перейти до покупок».

if ($basketEmpty){  
 return $returnString.'<a href="/index.php">Перейти до покупок</a></tbody></table></div></main>';  
}

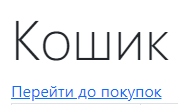
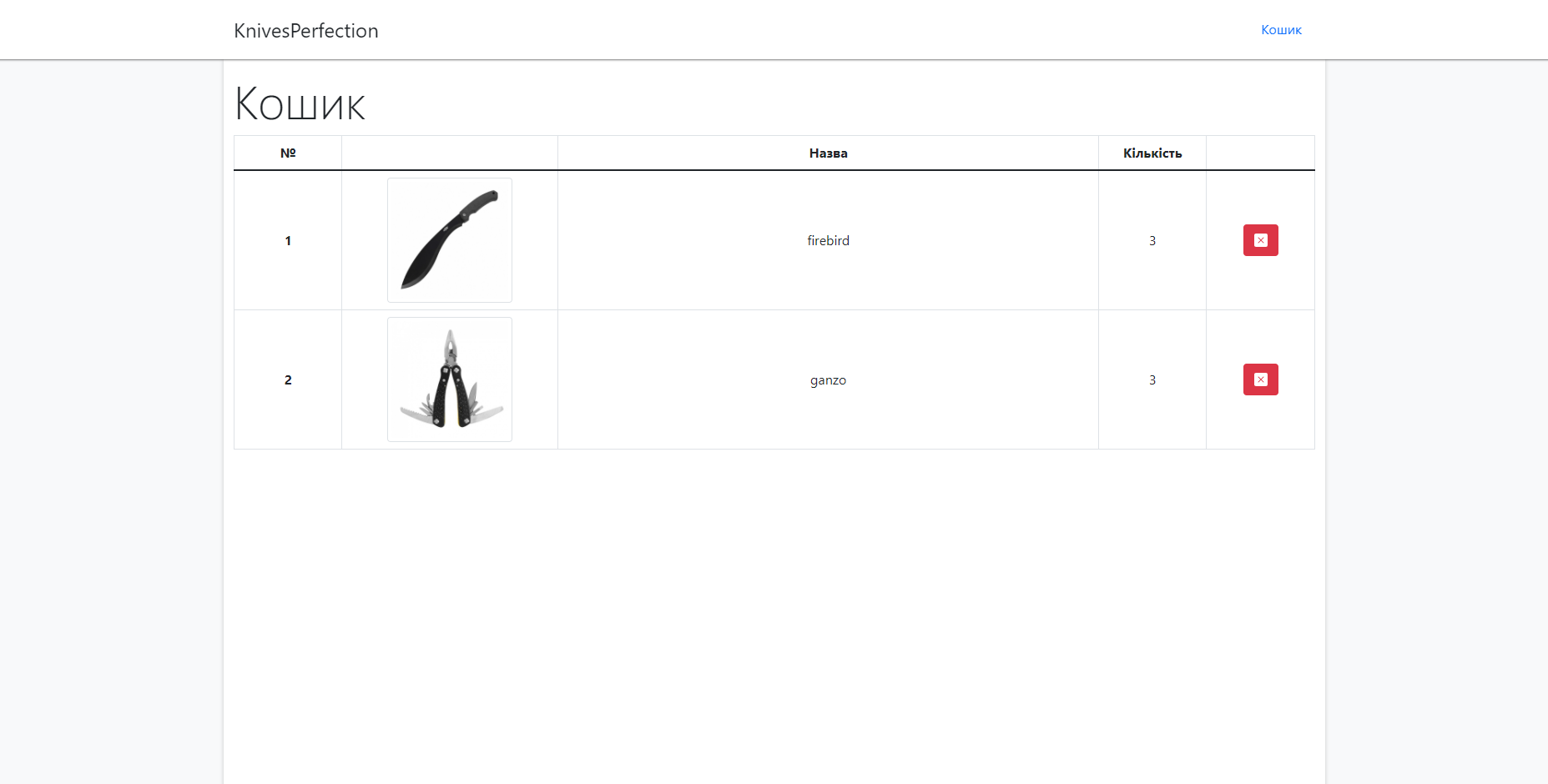


Рисунок 1.3 – поява напису «Перейти до покупок», коли корзина пуста

Якщо ж продукти в сесії додавались, то буде згенеровано рядок з таблицею обраних товарів.

Якщо користувач натискає на кнопку видалення товару, і підтверджує свою дію, запускається скрипт, що описаний на початку методу. Він передає у форматі JSON дані про товар, які необхідно видалити (в даному випадку, id). Після цього відбувається POST-запит на сторону сервера, де в сесії товар прибирається із списку обраних для покупки, а його кількість стає рівною нулю. За це відповідає скрипт delete.php. Його код наведено нижче.

<?php  
function autoload($class){  
 include "classes/" . $class . ".php";  
}  
  
spl\_autoload\_register("autoload");  
  
session\_start();  
  
$deletedItem = json\_decode(file\_get\_contents('php://input'), true);  
  
$\_SESSION[$deletedItem["id"]."\_check"] = "off";  
$\_SESSION[$deletedItem["id"]."\_num"] = 0;  
  
$page = new BasketPage();  
$page->getPage();

Рисунок 1.4 – Заповнена корзина товарів

Сервер повертає заново згенеровану сторінку, що вже не містить видаленого товару. Новий код вставляється в DOM-об’єкт document.body, тому вся описана процедура відбувається навіть без перезавантаження сторінки.

*5. Реалізувати успадкування.*

Успадкування реалізоване в наведених вище класах HomePage.php та BasketPage.php. Завдяки цьому суттєво зменшується обсяг повторюваного коду, а виклик конструктора будує велику частину сторінки автоматично.

Частина абстрактного класу Page:

<?php  
abstract class Page  
{  
 private string $pageContent;  
 private string $title;  
   
 function \_\_construct($title){  
 $this->title = $title;  
 $this->pageContent = $this->getHead().$this->getBody();  
 }

…

}

Частина класу-наслідника HomePage:

<?php  
class HomePage extends Page{  
 function \_\_construct(){  
 parent::*\_\_construct*("Головна");  
 }

…

}

Частина класу-наслідника BasketPage:

<?php  
class BasketPage extends Page{  
 function \_\_construct(){  
 parent::*\_\_construct*("Кошик");  
 }

…

}

*6. Написати скрипт, що виводить сторінки за допомогою властивостей і*

*методів цих класів.*

Клас index.php, що запускається при відкритті веб-сторінки першочергово, має вигляд, поданий нижче.

<?php  
session\_start();  
function autoload($class){  
 include "classes/" . $class . ".php";  
}  
  
spl\_autoload\_register("autoload");  
  
  
$page = new HomePage();  
$page->getPage();

Зміна $page є об’єктом класу HomePage. Завдяки методу getPage() повертається згенерована сторінка. Однак цей метод теж викликає інші методи всередині класу.

function getPage(){  
 echo "<!doctype html><html lang = 'uk'>".$this->pageContent."</html>";  
}

Тіло методу складається із виведення обмежувальних html-тегів <!doctype html><html lang = 'uk'> та </html>. Між ними додається змінна $pageContent об’єкту $this (того, в якому викликається метод getPage).

Змінна $pageContent ініціалізується в конструкторі.

function \_\_construct($title){  
 $this->title = $title;  
 $this->pageContent = $this->getHead().$this->getBody();  
}

Вона генерується шляхом поєднання результатів викликів функцій getHead() та getBody(), детально описаних у пункті 1 цієї роботи.

Отже, ініціалізована змінна повертається у методі getPage(), користувач бачить відображення сторінки.

*7. Розробити реалізацію сторінок сайту з інтернет-магазином (для обраної теми).*

В ході попередніх пунктів описано роботу сторінки товарів (HomePage.php, пункт 2), сторінки «Кошик» (BasketPage.php, пункт 4). Також описано, як працюють методи getHeader(), abstract getMain(), getFooter(), що повертають відповідно «шапку», «тіло», та «підвал» сторінки та задають змінну $pageContent у методі getPage() (пункти 1 та 6 роботи). Резульати виконання програми та її складових також проілюстровано у попередніх пунктах роботи за допомогою скріншотів.

Висновок: в ході роботи було створено каркас майбутнього сайту для реалізації власного товару. За допомогою ООП створено абстрактний клас сторінки, від якої наслідуються інші сторінки, створено методи для повернення сервером html-контенту, утворено зв’язок між клієнтською та серверною частиною веб-застосунку: дані передаються POST-запитами у вигляді параметрів запиту або JSON-об’єктів. Сторінка працює без збоїв, виявлені помилки було усунуто.

Відповіді на контрольні питання:

*2. Створення об’єктів і робота з ними*

Об’єкти класу створюються за допомогою операції new. Змінні, що можуть виступати об’єктами певного класу, позначаються за допомогою знаку $.

Отже, створення об’єкту може виглядати так:

$object = new SomeClass();

Змінні та методи класу певного об’єкту викликаються за допомогою наступного синтаксису:

$page->count – для змінних чи

$page->count() – для методів.

*4. Конструктори.*

При створенні об’єкту класу може викликатись описаний користувачем конструктор. Конструктор описується за допомогою зарезервованого слова \_\_construct:

function \_\_*construct*($param1, $param2){…};

Якщо конструктор має певні параметри, то при створенні об’єкту цього класу необхідно передати аргументи в конструктор. Наприклад, виклик конструктора може виглядати так:

$car = new Car(“blue”, “Lamborghini”, ‘250mph”);

Звідси видно, що конструктор викликається неявно при створенні об’єкту.

*6. Просте та ієрархічне спадкування*

Просте спадування в PHP реалізоване за допомогою ключового слова extends. Таким чином, один клас може спадкувати методи та змінні свого базового класу, якщо вони не задекларовані як private. Тобто, public та protected елементи батьківсього класу наслідуються його нащадками.

Також PHP підтримує ієрархічне спадкування. Ієрархічне успадкування - це тип успадкування, в якому програма складається з одного з батьків та кількох дочірніх класів. Отже, кілька класів можуть бути нащадками одного батьківського класу. Натомість один клас може бути нащадком лише одного класу. Тобто, один клас не може бути нащадком відразу кількох батьківських класів. Це дозволяють робити інтерфейси.

*8. Абстрактні класи*

Абстрактний клас містить один або більше абстрактних методів – методів, що не мають тіла та реалізовуються у класах-нащадках. Об’єкт абстрактного класу створити неможливо. Якщо будь-який клас має хоча б один абстрактний метод, він повинен описуватися як абстрактний.

Абстрактні класи можна наслідувати від абстрактних класів, однак кілька абстрактних класів не можуть бути батьками одного нащадка. Натомість інтерфейси – можуть. Також абстрактний клас може мати певну базову реалізацію, що використовується нащадками, в той час як інтерфейси просто оголошують перелік методів, які клас повинен реалізовувати.

*10. Функції для роботи з класами та об’єктами*

Є кілька вбудованих функцій, що допомагають працювати з PHP-класами та об’єктами. Для прикладу, функція get\_class() повертає назву класу, до якого належить об’єкт.

Функція \_\_autoload дозволяє автоматично завантажувати класи, які не імпортовано. В якості параметра вона приймає назву класу, який варто завантажити. Функція автоматично спрацьовує в разі, якщо не знайдено класу в уже імпортованих файлах. Функція class\_alias() створює псевдонім для вказаного класу.

Функція class\_exists() перевіряє, чи був оголошений клас. Аналогічно, функція enum\_exists() перевіряє наявність оголошення перерахування.

Функція is\_a() перевіряє, чи належить об’єкт даному класу, або одному з його предків. А функція get\_parent\_class() повертає ім’я батьківського класу для об’єкту чи класу.