МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет Магістерської підготовки

Кафедра Автоматизованих систем моніторингу навколишнього середовища

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: Моделі формування класів у методології розробки програмного продукту мовою PHP

Виконав студент 2 курсу групи МІС-18
спеціальності 122 Комп'ютерні науки

Владимиров Денис Дмитрович

Керівник професор каф. АСМНС, к. т. н., доцент Великодний Станіслав Сергійович

Консультант _____

Рецензент к. т. н., доцент
Гнатовська Ганна Арнольдівна

Одеса 2019

АНОТАЦІЯ

Моделі формування класів у методології розробки програмного продукту мовою PHP. Владимиров Денис Дмитрович.

На сьогоднішній день PHP – це потужний кросплатформний набір засобів, який розташовується на сервері і призначається для обробки коду, вбудованого в html-документи. Мета роботи – розглянути методи і функції, що використовуються під час створення класів на мові PHP, тобто аналіз моделей формування класів в методології розробки програмного проекту.

Об’єкт дослідження – процес створення відкритого програмного проекту.

Предмет дослідження – моделі формування класів.

Об'єктно-орієнтоване програмування (ООП) ґрунтується на трьох концепціях: інкапсуляції, успадкування і поліморфізму. Інкапсуляція – це механізм, який дозволяє захистити атрибути й методи об'єкта від некоректного використання. Згідно з принципами інкапсуляції атрибути класу не можуть бути доступними для екземплярів інших класів безпосередньо. Доступ до атрибутів має здійснюватися лише через методи класу.

PHP володіє величезним набором функцій і великою гнучкістю, які можуть бути значно розширені за допомогою додаткових зовнішніх бібліотек. PHP – це об’єктно-орієнтована мова, і можливостей в цьому напрямі дуже багато.

Основою ООП є об’єкт, а фабрикою для створення об’єктів є клас. Клас складається з наступних частин: властивості, конструктор, методи. Класи в PHP мають ряд спеціальних методів. Ці методи є досить гнучким інструментом: перевизначаючи їх, можна добитися істотного зміни поведінки об'єкта.

Магістерська кваліфікаційна робота містить: 84 сторінок, 13 рисунків, 1 таблицю, перелік посилань з 21 найменування, 1 додаток.

КЛЮЧОВІ СЛОВА: КЛАС, МОДЕЛЬ, МЕТОД, ЗМІННА, КОНСТРУКТОР, УСПАДКУВАННЯ, ПОЛІМОРФІЗМ, ФУНКЦІЯ, ШАБЛОН, ІЄРАРХІЯ

SUMMARY

Class formation models in the methodology of software development in the PHP language. Vladymyrov Denys.

To date PHP is a powerful cross-platform toolkit that resides on a server and is designed to handle code embedded in html-documents. PHP has a great set of features and great flexibility, which can be greatly expanded with additional external libraries. The purpose of the master's work – consider the methods and functions used when creating classes in language PHP, i. e. analysis of models of the formation of classes in the methodology of software development project.

The object of study is the process of creating an open-source software project.

Purpose of the study – the model for the formation of classes.

Object-oriented programming (OOP) is based on three concepts: encapsulation, inheritance and polymorphism. Encapsulation - a mechanism that helps protect the attributes and methods of the object from incorrect use. According to the principles of encapsulation class attributes may not be available for instances of other classes directly. Access attributes should be done only through class methods.

PHP has a great set of features and great flexibility, which can be greatly expanded with additional external libraries. PHP – an object-oriented language, and capabilities in this area very much.

The basis of the OOP is an object, and a factory for creating objects of a class. The class consists of the following parts: properties, constructor methods. Classes in PHP have a number of special techniques. These methods are very flexible tool: overriding them, you can achieve a significant change in the behavior of the object.

Master thesis contains: 84 pages, 13 illustrations, 1 table, list of references from 21 names, 1 appendix.

KEY WORDS: CLASS, MODEL, METHODS, VARIABLES, CONSTRUCTORS, INHERITANCE, POLYMORPHISM, FUNCTIONS, TEMPLATES, HIERARCHIES

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАК

ООП – об’єктно-орієнтоване програмування.

СУБД – система управління базами даних.

APC – Alternative PHP Cache.

CSS – Cascading Style Sheets.

CVS – Concurrent Versions System.

FI – Form Interpreter.

FTP – File Transfer Protocol.

HTTP – Hyper Text Transfer Protocol.

PDO – PHP Data Object.

PHP – Personal Home Page.

SOAP – Simple Object Access Protocol.

SPL – Standard PHP Library.

SQL – Structured Query Language.

XHTML – Extensible Hypertext Markup Language.

XML – eXtensible Markup Language.

ВСТУП

PHP – одна з найпопулярніших мов програмування в мережі Інтернет, але на початку розглядалася як проста мова написання сценаріїв. Дослівно абревіатура переводиться як Personal Home Page (PHP). Дана мова існує з 1994 р. Її творцем є Расмус Лердорф. Історія створення досить банальна, так як метою було написання простого «двигуна», але тепер ця мова використовує більше 10,5 млн. сайтів та інших різних проектів в мережі Інтернет по всьому світу. Перші версії скрипт – двигуна використовувалися виключно тільки в особистих цілях автора. PHP почав своє життя як ненав'язлива CGI-оболонка, написана на Perl.

Вже через рік з'явилася перша версія продукту, і вже тоді вона мала велику популярність. Але автор розумів, що ця версія володіла дуже скромними можливостями. На той момент у перелік функцій входили: найпростіший аналізатор коду, кілька команд і утиліт, які підходили тільки для персональних сторінок (гостьова книга, лічильник тощо). Процес створення швидко розвивався і вже до середини 1995 р. PHP був ґрунтовно перероблений. Був доданий Form Interpreter (FI), для спрощення обробки даних, що вводяться за допомогою форм. Також з'явилася підтримка MySQL, що дозволила працювати з базами даних.

У такому вигляді з'явилася друга версія продукту, яка носила назву PHP/FI Version 2. Після цього люди ще більше зацікавилися і стали самі писати бібліотеки, розширюючи функціональність мови. Версія PHP 3.0 піддалася значній переробці, що визначила сучасний вигляд і стиль мови програмування. У 1997 р. два ізраїльські програмісти, Енді Гутманс і Зеєв Сураські, повністю переписали код інтерпретатора. PHP 3.0 був офіційно випущений в червні 1998 р.

Однією з найсильніших сторін PHP 3.0 була можливість розширення ядра додатковими модулями. Згодом інтерфейс написання розширень привернув до PHP безліч сторонніх розробників, що працюють над своїми моду-

лями, що дало PHP можливість працювати з величезною кількістю баз даних, протоколів, підтримувати велике число API. Велика кількість розробників призвело до швидкого розвитку мови і стрімкого зростання його популярності. З цієї версії акронім PHP розшифровується як «PHP: hypertext Preprocessor», замість застарілого «Personal Home Page».

П'ята версія PHP була випущена розробниками 13 липня 2004 р. Вся інфраструктура об'єктної моделі в PHP 5 була повністю переписана. Зміни включають оновлення ядра Zend (Zend Engine 2), що істотно збільшило ефективність інтерпретатора. Введена підтримка мови розмітки XML. Повністю перероблені функції ООП, які стали багато в чому схожі з моделлю, використовуваною в Java. Зокрема, введено деструктор, відкриті, закриті та захищені члени і методи, інтерфейси й клонування об'єктів. Нововведення, проте, були зроблені з розрахунком зберегти найбільшу сумісність з кодом на попередніх версіях мови.

PHP 6 заснована на стабільної версії PHP 5.x. Підтримка Unicode в даний момент може бути встановлена тільки на рівні «per request», тобто для кожного запрошуваного файлу. Це означає, що PHP доведеться зберігати варіанти класів, імен методів і функцій одночасно в таблиці символів Unicode і в non-Unicode, що, звичайно ж, збільшує кількість споживаних ресурсів. Розробники PHP 6.0 вирішили зробити настройку Unicode на рівні всього сервера, а не запиту. Відключення підтримки юнікоду, якщо така не потрібна, може збільшити продуктивність строкових функцій до 300% і додатків в цілому до 25%.

У движок PHP версії 6.0 буде доданий новий тип даних – int64, який використовується за замовчуванням для integer. Однак, команда break розшириться статичною міткою, тому буде можливо написати break foo і це перекине на мітку foo: у коді. В версії PHP 6.0 додано нове ключове слово для доступу до подальшої зв'язці – static :: static2 (), що дозволить управляти static під час виконання.

Типізовані значення при поверненні параметрів з функцій. Однак така можливість додана, але не вирішено питання її синтаксису. У будь-якому випадку, це корисна можливість. Робота Alternative PHP Cache (APC) з байткодом буде включена в основну поставку PHP в якості стандарту, але, ймовірно, не буде активізована по-замовчуванню, але результати її роботи будуть стимулювати хостерів включати цю опцію.

Отже, на сьогоднішній день PHP, незважаючи на свою скромну назву (Personal Home Page – персональна домашня сторінка) – це потужний кросплатформний набір засобів, який розташовується на сервері і призначається для обробки коду, вбудованого в html-документи. Завдяки цьому, з'являється можливість створювати динамічні Web-сторінки. PHP володіє величезним набором функцій і великою гнучкістю, які можуть бути значно розширені за допомогою додаткових зовнішніх бібліотек. Ви можете керувати доступом до ваших сторінок, створювати і обробляти бази даних будь-якої складності, генерувати зображення або PDF-документи і тому подібне.

Мета роботи: розглянути методи і функції, що використовуються під час створення класів на мові PHP, тобто аналіз моделей формування класів в методології розробки програмного проекту.

Об’єкт дослідження – процес створення відкритого програмного проекту.

Предмет дослідження – моделі формування класів.

1АНАЛІТИЧНА ЧАСТИНА

1.1 Принципи формування класів

Класи були додані в код, який став основою версії PHP 3.0. Додані вони були як синтаксична прикраса для організації доступу до наборів даних. PHP вже підтримував поняття асоціативних масивів, і додані нововведення було нічим іншим, як новим незвичним способом доступу до подібних розділів. Тим не менш, як показав час, цей новий синтаксис надав набагато серйозніший вплив на PHP, ніж планувалося спочатку [1] [1)].

Починаючи з 4 версії, в PHP з'явилися об'єктно-орієнтовані властивості. Після появи версії PHP 5 можливості об'єктно-орієнтованого програмування на мові PHP були суттєво розширені [1], а крім того, підвищилася швидкодія розроблюваних програм. Однак більшість із нових властивостей залишаються невидимими. Ці зміни в основному стосуються ядра Zend 2, яке в PHP 5 стало набагато ефективнішим в порівнянні з ядром PHP 4. Крім підвищення швидкості обробки сценаріїв, в PHP 5 були додані нові можливості, які зробили цю мову програмування по-справжньому об'єктно-орієнтованою.

Об'єктно-орієнтований підхід до програмування припускає використання об'єктів і класів [2] [2)]. В даний час об'єктно-орієнтований підхід широко розповсюджений, і багато університетських курсів з програмування починаються саме з його вивчення. При об'єктно-орієнтованому програмуванні (ООП) найчастіше використовуються мови програмування Java і C++.

ООП – це не просто використання іншого синтаксису, а багато в чому й інший підхід до аналізу і вирішення завдань. ООП розробляється шляхом моделювання процесів в реальній предметній області. Наприклад, програміст, який розробляє програму підтримки функціонування відділу продажів деякої компанії, може розглядати її з точки зору взаємодії покупців, продав-

---

[1)] [1] Lecky-Thompson E., Nowicki S. D. Professional PHP6. Boston: Paperback, 2009. 512 p.

[2)] [2] Ловэйн П. Объектно-ориентированное программирование на PHP. Москва: Вильямс, 2017. 225 с.

ців та кредитної політики, тобто в термінах, якими оперують фахівці самого відділу продажів.

В ООП основними елементами програми є об'єкти (object) [3] [1)]. Вони є представленням реальних об'єктів, які задіяні при вирішенні реальної проблеми. Наприклад, якщо додаток відноситься до предметної області продажу уживаних автомобілів, то, можливо, в якості об'єктів будуть обрані автомобілі і покупці. Якщо ж розв'язувана задача пов'язана з космосом, то об'єкти будуть представляти зірки і планети. ООП є концепцією [4] [2)], яка надає нову термінологію для опису аналізованих проблем. Розуміння цієї термінології – ключ до розуміння самого об'єктно-орієнтованого підходу.

1.1.1 Поняття класу в PHP

Клас (class) є шаблоном (каркасом), який можна використовувати для створення об'єктів [5] [3)]. Клас визначає властивості і атрибути об'єкта, а також дії, які він може виконувати. Наприклад, розглянемо клас, який визначає чотириколісний автомобіль з двигуном, який може переміщатися вперед й паркуватися. На його основі можна створити новий об'єкт (новий автомобіль). Однак у процесі використання цього об'єкта можна виявити відсутність деяких важливих деталей, таких як двері, кермо або задня передача. Це може статися через те, що ці компоненти при створенні класу не були враховані.

Розробник, який займається створенням класу, знає все про його внутрішній устрій. Однак для тих, хто буде використовувати цей клас, подібна інформація зовсім необов'язкова. Наприклад, необов'язково знати, як функціонує телефон – подзвонити можна і без цієї інформації. Фахівцеві, сконструйованих телефон, відомо про його внутрішній архітектурі. Тому при появі нових технологій він зможе розкрити телефонний апарат і вдосконалити йо-

---

[1)] [3] Zandstr M. PHP Objects, Patterns and Practice. Moscow: Vilyams, 2016. 480 p.

[2)] [4] Кузнецов М. С., Симдянов И. А. Объектно-ориентированное программирование на PHP. Москва: БХВ, 2018. 608 с.

[3)] [5] Gutrans A., Bakken S., Rethans D. PHP 5 Power Programming. New York: Prentice hall, 2013. 704 p.

го. Разом з тим, до тих пір, поки не зміниться зовнішній інтерфейс (клавішна панель і кнопки) телефону, на його використання ніщо не вплине.

Поява структурного підходу до програмування характеризується в першу чергу застосуванням функцій, коли частину коду можна оформити у вигляді функції і використовувати його кілька разів у різних місцях програми [6] [1)]. Такий прийом дозволяє не тільки повторно використовувати код, але й налагоджувати його невеликими блоками, які відповідають конкретним завданням.

Класи – це конструкції, що моделюють наші поняття. Клас застосовують як розширений тип змінної. Тільки при оголошенні такої «змінної» ми отримуємо не звичайну змінну, а модель об'єкта, яку надалі будемо називати просто об'єктом. При цьому об'єкт розділений на дві частини: закриту, яка доступна тільки функціям всередині об'єкта, і відкриту, яку можна викликати для цього об'єкта з тексту основної програми.

Уточнення класу (поняття) відбуваються за допомогою спеціального механізму, названого спадкуванням. При оголошенні класу можна дізнатися, що новий клас успадковує від вже існуючого. Крім спадкування, існує інший тип відносин, званий агрегацією. Класи можуть в якості членів містити інші об'єкти. Якщо клас «водний транспорт» може бути базовим для підводного човна, крейсера, лайнера, то в свою чергу будь-який з кораблів може містити такі об'єкти, як палуба, двигун, навігаційне обладнання тощо. Причому кількість і види об'єктів, що містяться в класі, не обмежені.

1.1.2 Ініціалізація змінних

Часто деяким атрибутам класу буває необхідно присвоювати значення відразу після створення представника класу. Коли ми створювали клас статей, для присвоювання значень атрибутів (властивостей) класу ми викорис-

[1)] [6] Гамма Э., Хелм Р., Джонсон Р., Влиссидес Д. Приемы объектно-ориентированного проектирования. Санкт-Петербург: Гамма, 2018. 368 с.

товували спеціальну функцію make_article (). Взагалі кажучи, ми вчинили не зовсім вірно, тому що зайнялися винаходом велосипеда. Спеціально для завдання початкових значень атрибутів класу існує два стандартних методу. В PHP5 можна ініціалізувати значення за допомогою оператора var або за допомогою функції конструктора. За допомогою var можна ініціалізувати тільки константні значення. Для завдання не константних значень використовують функцію конструктор, яка викликається автоматично, коли об'єкт конструюється з класу. Функція-конструктор повинна мати ім'я, що збігається з ім'ям всього класу, в якому вона визначена.

1.1.3 Клас як абстрактний тип

Клас це конструкція, що задає зразок, по якому буде побудований об'єкт. Образно кажучи, можна представити його як форму, за допомогою якої відливають деталі на металургійних виробництвах. Одного разу розробивши таку форму з потрібними геометричними параметрами, можна відлив з неї будь-яку кількість деталей, які будуть точними копіями один одного і цієї форми. Точно так само в програмі, один раз написавши клас, можна потім створювати будь-яку кількість об'єктів цього класу, які будуть містити всі змінні і методи, які є у їх класу.

Оголошення класу здійснюється за допомогою ключового слова class, за яким слідують назва класу та компоненти класу у фігурних дужках [7] [1)].

Об'єкт класу оголошується за допомогою ключового слова new, за яким слідує назва класу. Як тільки об'єкт класу оголошений, з'являється можливість звертатися до його компонентів за допомогою уточненого імені, яке включає ім'я об'єкта, наступну за ним стрілку (->) і ім'я методу і члена класу.

Методи і члени класу можуть викликатися не тільки з зовнішньої програми, але і з самого класу. Для того щоб звернутися до змінної або методу класу усередині класу, це звернення необхідно випередити конструкцією $

[1)] [7] Макконелл С. Совершенный код. Москва: БХВ-Петербург, 2018. 510 с.

this->. Змінна $ this, яка неявно присутня в кожному класі, є посиланням на поточний об'єкт класу і повідомляє інтерпретатору PHP, що ви звертаєтеся до змінної даного класу, а не створюєте нову. Членів класу можна присвоювати значення, як звичайним змінним програми.

В PHP 5.x новою важливою особливістю в моделі об'єктного орієнтування є контролі доступу. В PHP 5 об'єктна модель передбачає три рівні доступу до членів класу, що обмежують дані, які можуть вилучатись в сценаріях. Це рівні public, private і protected; їх можна застосовувати і до методів, і до властивостей класу. До членів класу, які оголошені як public (загальнодоступні), доступ може бути здійснений з будь-якого місця в межах сценарію. За допомогою об'єкта їх можна викликати або видозмінювати або зсередини самого об'єкта, або за його межами. Навпаки, доступ до членів класу, які були оголошені як private (закриті), може бути здійснений тільки з примірника цього класу за допомогою змінної $ this.

Третім і останнім рівнем доступу в PHP є protected (захищений). Цей рівень подібний до рівня private, оскільки він забороняє зовнішній доступ до члена класу. Однак на відміну від рівня private, що обмежує доступ тільки до того класу, в якому він визначений, рівень protected дозволяє доступ як з нього самого, так і з будь-яких дочірніх класів.

Ще одним нововведенням в об'єктній моделі в PHP 5 є зазначення типів (type hinting). У самій мові PHP використання типів не передбачено. Це означає, що змінні можуть зберігати дані довільного типу. Дійсно, одна і та ж змінна в одному випадку може оброблятися як цілочислова змінна, а в іншому – як строкова змінна. Тим не менш, оскільки методи всередині об'єктів часто приймають параметри, які є екземплярами інших об'єктів, PHP 5 дозволяє обмежувати типи даних для параметрів методів [8] [1)].

При використанні оператора clone за замовчуванням повертається точна копія клонованого об'єкту. Однак клас також може реалізувати спеціаль-

[1)] [8] Кухарчик А. Л. PHP 5 на примерах. Москва: Новое знание, 2009. 240 с.

ний метод _ clone (), що дозволяє управляти елементами, які будуть копіюватися з одного примірника в інший. У цьому спеціальному методі змінна $this посилається на нову копію об'єкта разом з усіма значеннями з початкового об'єкта.

1.1.4 Статичні змінні

Клас може містити оголошення властивостей. Кожен екземпляр класу володіє своєю власною копією властивостей. Однак клас може містити ще один різновид властивостей – статичні властивості [9] [1)]. На відміну від звичайних властивостей, вони мають відношення не до конкретного об'єкта, а до всього класу в цілому. Тому їх часто називають властивостями класу в протилежність властивостей об'єкта. Статичні властивості можна розглядати як свого роду глобальні змінні, оголошені усередині класу, які доступні за межами ерозії, але виключно в контексті класу.

Оголошуються статичні властивості з ключовим словом static. Синтаксис доступу до статичних властивостей класу дещо незвичайний – ім'я властивостей повинно передувати імені класу. В разі, якщо доступ до статичних властивостей здійснюється з методів класу, синтаксис звернення можна дещо спростити – замість імені класу вказати ключове слово self, яке представляє собою спрощений варіант імені поточного класу.

1.1.5 Конструктори і деструктори

Серед методів класу розрізняють два особливих методу: конструктор і деструктор [10] [2]. Конструктор – це спеціальний метод класу, призначений для ініціалізації членів класу. Цей метод виконується раніше всіх інших методів класу під час оголошення об'єкта. У конструкторі зазвичай здійснюють

[1)] [9] Beyross I. PHP 5.1 for Beginners. New York: Sams, 2010. 360 p.
[2)] [10] Basher A. PHP Essentials. Southgempton: Course Technology, 2008. 326 p.

ініціалізацію членів класу і резервування ресурсів, необхідних для роботи об'єкта – виконується відкриття файлів, з'єднання з базою даних тощо. Для того щоб використовувати в класі конструктор, необхідно оголосити у ньому метод з ім'ям _construct ().

Деструкція – це спеціальний метод класу, призначений для звільнення ресурсів, зайнятих об'єктом під час його існування. Цей метод завжди викликається після всіх інших методів під час знищення об'єкта. У ньому можна закрити відкриті файли, від'єднатися від бази даних.

Конструктор і деструктор – це ключові методи класу. Їх наявність є однією з ознак об'єктно-орієнтованої технології. Але це не означає, що в кожному класі обов'язково повинні бути констуктор і деструктор – це необов'язкові елементи класу і їх слід застосовувати тільки при необхідності.

1.1.6 Спадкування класів та константи

Константи класів, які є новою особливістю PHP 5.x, дозволяють визначати постійні значення у визначеннях класу. Визначення константи всередині класу здійснюється за допомогою ключового слова const, за яким слідує ім'я константи і її значення.

Спадкування – це можливість розширювати можливості одного класу функціональністю іншого класу. Коли один клас успадковує інший, то всі методи, властивості і константи батьківського класу стають доступнішими і з класу-спадкоємця. Більш того, класи-спадкоємці можуть також продовжувати реалізацію деяких або всіх методів, властивостей і констант батьківського класу, щоб забезпечити додаткові або відмінні функціональні можливості. Щоб один клас міг успадковувати інший клас, в його визначенні ставиться ключове слово extends.

Глобальні константи в PHP існують з давніх часів. вони можуть бути оголошені за допомогою функції define(). З виходом PHP 5 з'явилася можли-

вість визначати константи всередині класу [11] [1)]. Подібно статичним членам, вони належать усьому класу в цілому, а не окремим його екземплярам. В іменах констант класу розрізняються символи верхнього та нижнього регістрів. Оголошення констант виглядає дуже просто, а доступ до них здійснюється так само, як і до статичних членів класу.

1.1.7 Абстрактні, вбудовані класи та їх інтерфейси

Якщо судити з назви, абстрактні класи використовуються в PHP для визначення абстрактних об'єктів. У ООП абстрактні класи призначені для того, щоб створити суперклас, який буде визначати абстрактні характеристики його класів-спадкоємців. Насправді абстрактні класи можуть містити в собі якийсь код, а можуть бути взагалі без коду; крім цього, на їх основі можна створити екземпляр безпосередньо.

Будь об'єктно-орієнтована мова поставляється з бібліотекою вбудованих класів, і PHP не є винятком. PHP 5 містить стандартну бібліотеку SPL, в яку входить велика кількість готових класів та інтерфейсів. У версії 5.1 в залежності від конфігурації є понад 100 вбудованих класів і інтерфейсів – це помітне збільшення в порівнянні з 5.0 [12] [2)]. Наявність готових класів прискорює розробку, а будучи написані на мові C, вони істотно підвищують продуктивність. Але навіть якщо вбудований клас робить не зовсім те, що вам потрібно, його без зусиль можна розширити під свої завдання.

Оскільки мова PHP призначений для створення динамічних Web-сторінок, наявність підтримки для роботи з базами даних неможливо переоцінити. У PHP 5 включено розширення mysqli (MySQL Improved), орієнтоване на версії MySQL починаючи з 4.1. Тепер можна користуватися наявними в MySQL механізмом підготовки запитів, причому для цієї мети пропонується

[1)] [11] Котеров Д. В., Костарев А.Ф. PHP 5. Санкт-Петербург: БХВ-Петербург, 2005. 1120 с.

[2)] [12] Коггзолл Дж. PHP 5: Полное руководство. Москва: Издательский дом «Вильямс», 2006. 752 с.

об'єктно-орієнтований інтерфейс, який дозволяє робити все те, що раніше виконувалося процедурно. SQLite – це база даних, вбудована безпосередньо в PHP. Вона не настільки універсальна, як MySQL, але є ідеальним рішенням в деяких ситуаціях і найчастіше дозволяє створювати більш швидкі і гнучкі додатки, що вживають мало пам'яті. І для неї теж є повністю об'єктно-орієнтований інтерфейс.

В випадку, якщо потрібно звертатися до декількох різних систем управління базами даних (СУБД), то цей пакет стане ідеальним рішенням. Загальний для різних баз даних інтерфейс PDO став можливий тільки з появою нової об'єктної моделі.

Спадкування визначає взаємовідносини між батьківськими і дочірніми класами. Однак найчастіше виникає потреба додати в клас додатковий «інтерфейс», тобто, по суті, декларувати додаткові угоди, яких буде дотримуватися цей клас. Вся ця проблема вирішується за рахунок множинного спадкування і похідних з двох класів. В PHP обраний інтерфейс як альтернатива множинного спадкоємства, який дозволяє програмісту визначати додаткові угоди та за яким має слідувати клас [13] – [15] [1)].

Оголошення інтерфейсу дуже нагадує оголошення класу з тією лише різницею, що інтерфейс може містити в собі тільки прототипи функцій (без реалізації) та константи. Будь клас, який «заявляє» про підтримку того чи іншого інтерфейсу, автоматично отримує всі константи інтерфейсу і зобов'язаний утримувати реалізацію всіх функцій, оголошених в інтерфейсі. Інтерфейс може успадковувати інший інтерфейс. Синтаксис оголошення похідних інтерфейсів нагадує синтаксис оголошення похідних класів, тільки в разі інтерфейсів допускається множинне спадкування. Подібно до того як класи ре-

---

[1]) [13] Орлов А. А. PHP: Полезные приемы. Москва: Горячая линия-Телеком, 2014. 224 с.

[14] Мазуркевич А. М., Еловой Д. М. PHP: настольная книга программиста. Минск: Новое знание, 2013. 480 с.

[15] Котеров Д. В. Самоучитель PHP 4. Санкт-Петербург: БХВ-Петербург, 2003. 576 с.

алізують інтерфейси, самі інтерфейси можуть розширювати тільки інші інтерфейси за умови відсутності конфліктів імен між ними.

1.2 Огляд редакторів PHP-коду

Найвідомішими серед редакторів PHP кодів є [16] – [20] [1)]: PHP Expert Editor, PHP Coder, PHPNotepad, Top PHP Studio, Davor's PHP Editor, NuSphere PhpED, VS.Php, DzSoft PHP Editor, TsWebEditor, PHPEdit, Tavrida PHP Editor, HAPedit , Scriptomania.

PHP Expert Editor – потужне і зручне в роботі інтегроване середовище розробки PHP для Windows [21] [2)]. Цей редактор спеціально розроблений для PHP-майстрів, має хороші особливості для новачків і професійних програмістів. PHP Expert Editor має внутрішній HTTP-сервер і відладчик, для того, щоб виконувати, перевіряти і налагоджувати сценарії PHP, робити перевірку синтаксису PHP, використовуючи внутрішній браузер, експлорер коду, файловий експлорер, FTP-клієнт, експлорер проектів та шаблони коду.

PHP Coder – інтегроване середовище розробки для PHP програмістів. Головна особливість – можливість тісного інтегрування з PHP інтерпретатором і PHP документацією, PHP Coder дозволяє Вам економити час на розробці PHP скриптів. Особливості: крім щільної інтеграції з PHP інтерпретатором і документацією, вікна попереднього перегляду, повноцінна підсвічування синтаксису для HTML і PHP, система автозавершення для часто використовуваних конструкцій, органайзер проекту, підтримка пошуку і заміни, кнопки для тегів і функцій, що настроюється интрефейс і багато іншого.

[1)] [16] Зольников Д. С PHP 5. Как самостоятельно создать сайт любой сложности. Москва: НТ-Пресс, 2007. 272 с.

[17] Кухарчик А. PHP: обучение на примерах. Минск: Новое знание, 2014. 237 с.

[18] Аргерих Л. И. Профессиональное PHP программирование. Санкт-Петербург: Символ-Плюс, 2013. 1048 с.

[19] Кузнецов М. В., Симдянов И. В., Голышев С. В. PHP 5. Практика разработки Web-сайтов. Санкт-Петербург: БХВ-Петербург, 2009. 960 с.

[20] Хольцнер С. PHP в примерах. Москва: ООО «Бином-Пресс», 2007. 352 с.

[2)] [21] Каждан И. А. PHP Expert Editor. Москва: Компьютерная газета, 2008. 600 с.

PHPNotepad забезпечує багато корисні особливості редагування PHP, в яких Ви потребуєте. Серед основних особливостей – підсвічування синтаксису, нумерація рядків, автозавершення коду, функціональні підказки, табульованого редагування та ін.

Top PHP Studio – інтегроване середовище обробки для PHP, що забезпечує всебічне і зручне в роботі рішення для редагування, тестування і налагодження PHP додатків. Основні особливості: вбудований веб-сервер, внутрішній веб-браузер для того, щоб перевіряти сценарії PHP, зручний редактор коду, що підтримує підсвічування синтаксису з перебудовується конфігурацією для PHP, HTML, XML, SQL, автозавершення функції / параметрів для стандартних функцій PHP, експлорер коду , вбудований клієнт FTP, менеджер проектів, система пошуку в файлах, система перевірки синтаксису PHP коду, утиліта порівняння файлів, фрагменти коду для CSS і HTML-тегів, шаблони, що настроєний інтерфейс і багато іншого.

Davor's PHP Editor – компактна і швидка інтегроване середовище розробки Windows для розробників PHP. Має: віконний інтерфейс вільного стилю з конфігурується середовищем, просунуту систему підсвічування коду з перебудовується конфігурацією, множинні вікна редагування, розширений мультіредактор, підтримує підсвічування синтаксису для багатьох мов програмування – PHP, HTML, CSS, XML, JS, SQL і ASP, містить менеджер проекту, систему перевірки синтаксису PHP, можливість запуску скриптів і попередній перегляд, розширений аналіз коду (PHP, JS і HTML), редактор HTML за типом WYSIWYG, підтримка синхронізації файлів на основі FTP, зараз розробляється інтегрований відладчик PHP.

NuSphere PhpED – це професійна інтегрована середа обробки, призначена головним чином для того, щоб формувати програми з використанням баз даних і PHP, HTML, XML, CSS. Стійкий редактор коду, відмінна система підсвічування коду в мовах PHP, XML, XHTML, HTML, CSS і JavaScript, потужний PHP відладчик, профілювальник і видавець – все в одному. Програма також включає інтегровану базу даних і клієнти CVS, сервіси SOAP, валідатор

HTML і інструментальні засоби форматер Коду, підтримка роботи з Smarty, редактор Unicode, SFTP підтримується для безпечних завантажень і закачувань, Telnet, SSH термінали для віддаленого адміністрування, підтримка функціональних можливостей MySQL, Oracle, MSSQL SQLite, Interbase, підтримка аналізу помилок, графічний інтерфейс користувача повністю настроюється, аналізатор PHP коду.

VS.Php – Php editor for Visual Studio (.Net) – за допомогою даної програми можна розробляти PHP додатки в середовищі Visual Studio.Net 2018. Ця програма дозволяє Вам формувати, редагувати, відлагоджувати і розгортати PHP додатки, використовуючи знайомий інтерфейс найпопулярнішою інтегрованого середовища розробки від Microsoft. Всупереч іншим програмам інтегрованого середовища розробки PHP, VS.Php доповнює знайомий інтерфейс Microsoft Visual Studio.Net 2018, дозволяючи розробляти додатки PHP, не покидаючи знайомого середовища розробки для програмістів. NET.

DzSoft PHP Editor – зручний і потужний інструмент для того, щоб писати і тестувати сценарії PHP і HTML/XML сторінки. Будучи розробленим безумовно для PHP розробки, він має дружній, але потужний інтерфейс, який є зручним і для новачків, і для досвідчених програмістів. Має зручний попередній перегляд у внутрішньому браузері (не потрібний зовнішній веб-сервер), швидкі вставки коду і багато інших необхідних особливості, які допоможуть в зручній і продуктивній розробці PHP скриптів.

TsWebEditor – текстовий редактор для HTML, PHP, Perl, JavaScript, CSS і багатьох інших мов. Він відображає вихідний текст кольоровим (підсвічування синтаксису), відображає підказки коду для функцій, завершення коду (PHP, JavaScript, ASP, HTML), має браузер коду, PHP відладчик, перевіряє правильність синтаксису, має довідкову систему, майстер CSS, редактор html, виробляє перевірку правильності синтаксису HTML, має вбудований зручний проектувальник SQL для MySQL і PostgreSQL, клієнт FTP, шаблони коду, виробляє зручний пошук і заміну, роботу із зовнішніми програмами, закладки, можливість вибору стовпця, рядка, майстра та багато інших особливостей.

PHPEdit – це повноцінна середовище розробки PHP для Windows, один з найпотужніших інструментів для розробки та налагодження PHP сценаріїв. Він має систему тісної інтеграції, систему автозавершення коду, просунуту підсвітці коду для PHP, CSS, Javascript, SQL, XML, XSLT тощо, систему автоматичної перевірці і синтаксису PHP, інтегрований відладчик, браузер коду, шаблони.

Tavrida PHP Editor – додаток в стилі Delphi для розробників PHP з підтримкою баз даних та внутрішньою системою тестування за допомогою вбудованого веб-сервера. Основні особливості: Delphi подібний інтерфейс: основне вікно з палітрою тегів (аналог палітри компонентів), редактор тегів (аналог інспектора об'єктів), веб-сайти управляються як проекти – набори файлів різних типів: HTML, JavaScript, PHP, CSS, XML, текстові, зображення тощо; багатофункціональний редактор коду з HTML, JavaScript, PHP, CSS та XML підсвічуванням синтаксису, система автозавершення коду, можливість вставляти стандартні html-теги з одиночним кліком мишки, зручний принцип встановлення параметрів HTML-тегів і JavaScript обробників подій, можливість перевіряти веб-сторінки з різними методами передачі параметрів (GET і POST), відображення результатів виконання у вбудованому веб-браузері, вбудований клієнт FTP та інше.

HAPedit – акронім для ASP, HTML і PHP редактора. Являє собою win32 редактор, корисний для всіх розробників динамічних веб-сторінок. Його основні особливості: підсвічування синтаксису для HTML і PHP, HTML і ASP, HTML, JavaScript, CSS і SQL; можливість попереднього перегляду сторінки в браузері; органайзер проекту; транслятор PHP коду, вбудований редактор HTML, автозавершення коду, консоль SQL, менеджер FTP, експлорер коду, настроюваний інтерфейс і інші особливості. Також є можливість підкачки мовних та інших модулів.

Scriptomania – редактор PHP коду, що володіє високою функціональністю. Основні особливості: редактор HTML / PHP / JavaScript / MySQL коду з просунутою системою підсвічування синтаксису, підтримка SQL, механізми

автозавершення коду (HTML / PHP), а також механізм корекції коду, можливість локального тестування скриптів, обширні інструментальні засоби.

Технологія PHP дала можливість вбудовувати виконуючий на сервері код в текст сторінки, що дозволило відокремити HTML-розмітку від програмної логіки. Вносити зміни стало набагато простіше, а оскільки PHP – інтерпретована мова, то відпала необхідність в перекомпіляції. Способи застосування PHP численні і різноманітні, але головна причина використання цієї технології – простота роботи. Саме тому PHP завоювала таку популярність. Поява версії 5 ще суттєвіше полегшила життя Web-програмістів. Тепер ви можете додати в свій арсенал засобів розробки потужну і в той же час не дуже складну об’єктно-орієнтовану мову. Використання об'єктно-орієнтованого підходу також помітно спрощує написання і читання програм. Зокрема, часто виявляється, що в програмі група функцій використовує одні й ті ж змінні. Доводиться щоразу передавати ці змінні в якості параметрів (що саме по собі утомливо) або оголошувати їх як глобальні, ризикуючи виникненням конфліктів імен, коли різні функції використовують одну і ту ж змінну в різних цілях і перезаписують в неї свої значення, заважаючи один одному. Крім того, класи об'єктів можуть успадковувати властивості один одного, що позбавляє програмістів від дублювання коду – якщо в грамотно розробленою програмою знаходять помилку, то вона виправляється в одному місці.

Отже, робота з класами не представляє великої складності. Всі професійні Web програмісти намагаються максимально розподіляти код в окремі класи. Клас може мати заздалегідь призначені внутрішні змінні зі значенням (дія класу за замовчуванням) або йому можуть призначатися нові значення внутрішніх змінних з середовища поза класу. Клас може мати необмежену кількість внутрішніх функцій, які можуть викликати один одного. Все це в цілому дає можливість створити ефективний і багатофункціональний об'єкт.

2 МОДЕЛЮВАННЯ ОБ'ЄКТА ДОСЛІДЖЕННЯ

2.1 Моделі варіантів використання

В першу чергу PHP використовується для створення скриптів, що працюють на стороні сервера, для цього його, власне, і придумали. PHP здатний вирішувати ті ж завдання, що і будь-які інші CGI-скрипти, у тому числі обробляти дані html-форм, динамічно генерувати html сторінки тощо, але є й інші області, де може використовуватися PHP. Всього виділяють три основні області застосування PHP.

Перша область – це створення додатків (скриптів), які виконуються на стороні сервера. PHP найбільш широко використовується саме для створення такого роду скриптів. Для того щоб працювати таким чином, знадобиться PHP-парсер (тобто обробник php-скриптів) і web-сервер для обробки скрипта, браузер для перегляду результатів роботи скрипта і текстовий редактор для написання самого php-коду. Парсер PHP розповсюджується у вигляді CGI-програми або серверного модуля. Як встановити його і web-сервер на свій комп'ютер, ми розглянемо згодом.

Друга область – це створення скриптів, що виконуються в командному рядку. Тобто за допомогою PHP можна створювати такі скрипти, які будуть виконуватися, незалежно від web-сервера і браузера, на конкретній машині. Для такої роботи потрібно лише парсер PHP (в цьому випадку його називають інтерпретатором командного рядка (cli, command line interpreter)). Цей спосіб роботи підходить, наприклад, для скриптів, які повинні виконуватися регулярно за допомогою різних планувальників задач або для вирішення задач простої обробки тексту.

І остання область – це створення GUI-додатків (графічних інтерфейсів), що виконуються на стороні клієнта. В принципі це не найкращий спосіб використовувати PHP, особливо для початківців, але якщо ви вже досконально вивчили PHP, то такі можливості мови можуть виявитися вельми корисні.

Для застосування PHP в цій області буде потрібно спеціальний інструмент – PHP-GTK, який є розширенням PHP.

Представимо ці області у діаграмі на рис. 1, а математичні функції у наступній діаграмі (рис.2).

2.2 Використання віртуальних серверів на прикладі моделі взаємодії

Віртуальний виділений сервер емулює роботу окремого фізичного сервера. На одній машині може бути запущено безліч віртуальних серверів. Крім деяких очевидних обмежень, кожен віртуальний сервер надає повний і незалежний контроль і управління, як надає його звичайний виділений сервер.

Локальний сервер – спеціальна програма, що дозволяє веб-розробникам розробляти сайт на локальному (домашньому) комп'ютері, без необхідності виходу в Інтернет. Потреба в такій програмі виникає при розробці динамічних сайтів, тобто сайтів, що використовують у своїй роботі php (або perl) – скрипти. Для тестування звичайних htm-сайтів локальний сервер не потрібен.

Для того, щоб браузер зрозумів php код, скрипт потрібно попередньо обробити спеціальним обробником, який поверне результат у вигляді звичайного html. Такий обробник є майже на кожному інтернет-сервер, що надає послуги хостингу, за винятком деяких безкоштовних хостингів. Однак на домашньому комп'ютері його немає, тому й створили такий дистрибутив, як локальний сервер, який, після його встановлення, імітує роботу інтернет-сервера.

У своїй практиці я використовую локальний сервер, під назвою Денвер. Денвер (Denwer) – це набір програм для створення сайту на локальному комп'ютері, без виходу в Інтернет. Проект Денвер був розроблений Дмитром Котеровим і включає в себе:

- Apache, SSI, mod_rewrite, mod_php;
- PHP4 з підтримкою GD і MySQL;

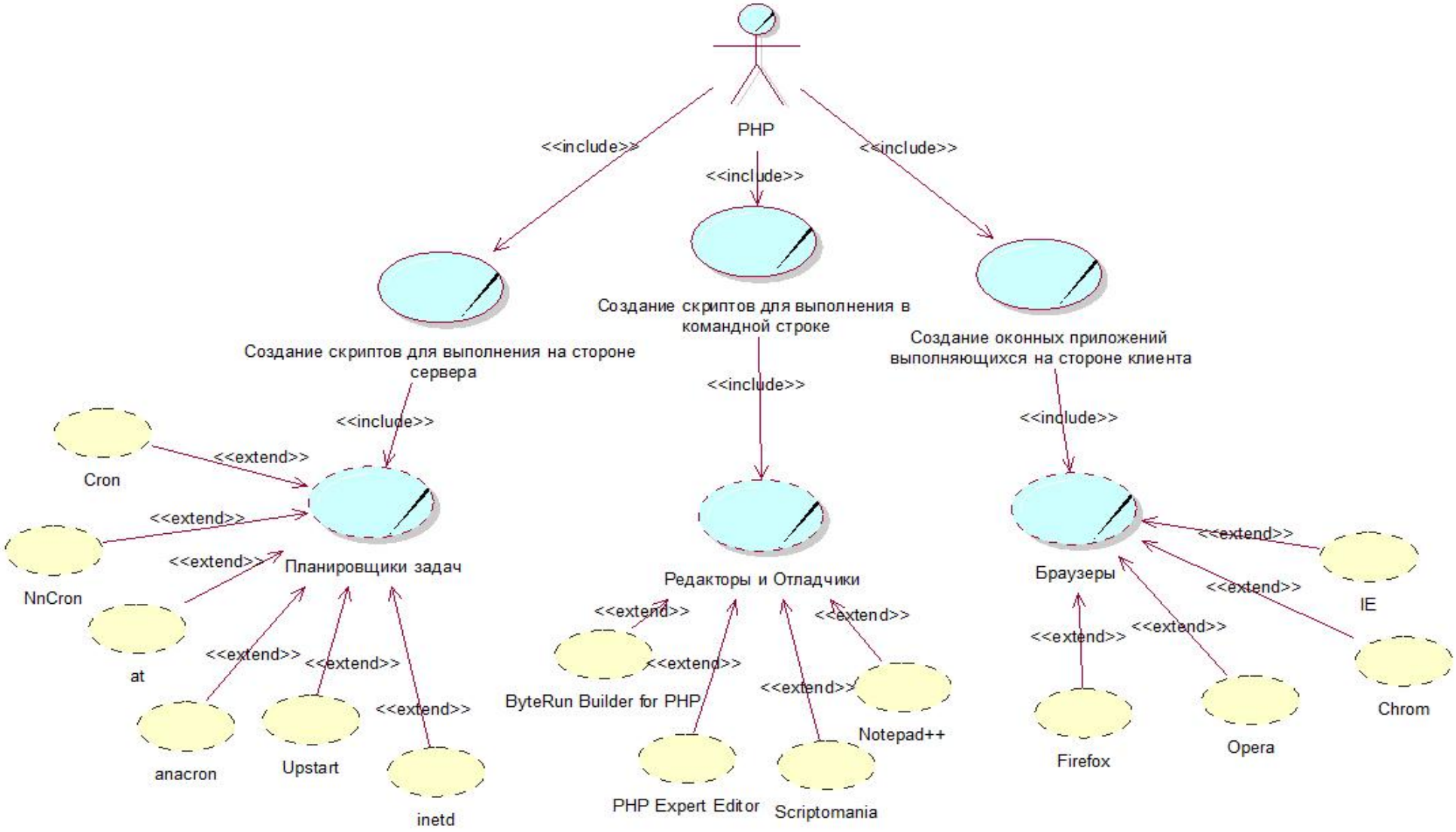

Рисунок 1 – Діаграма варіантів використання для областей застосування PHP

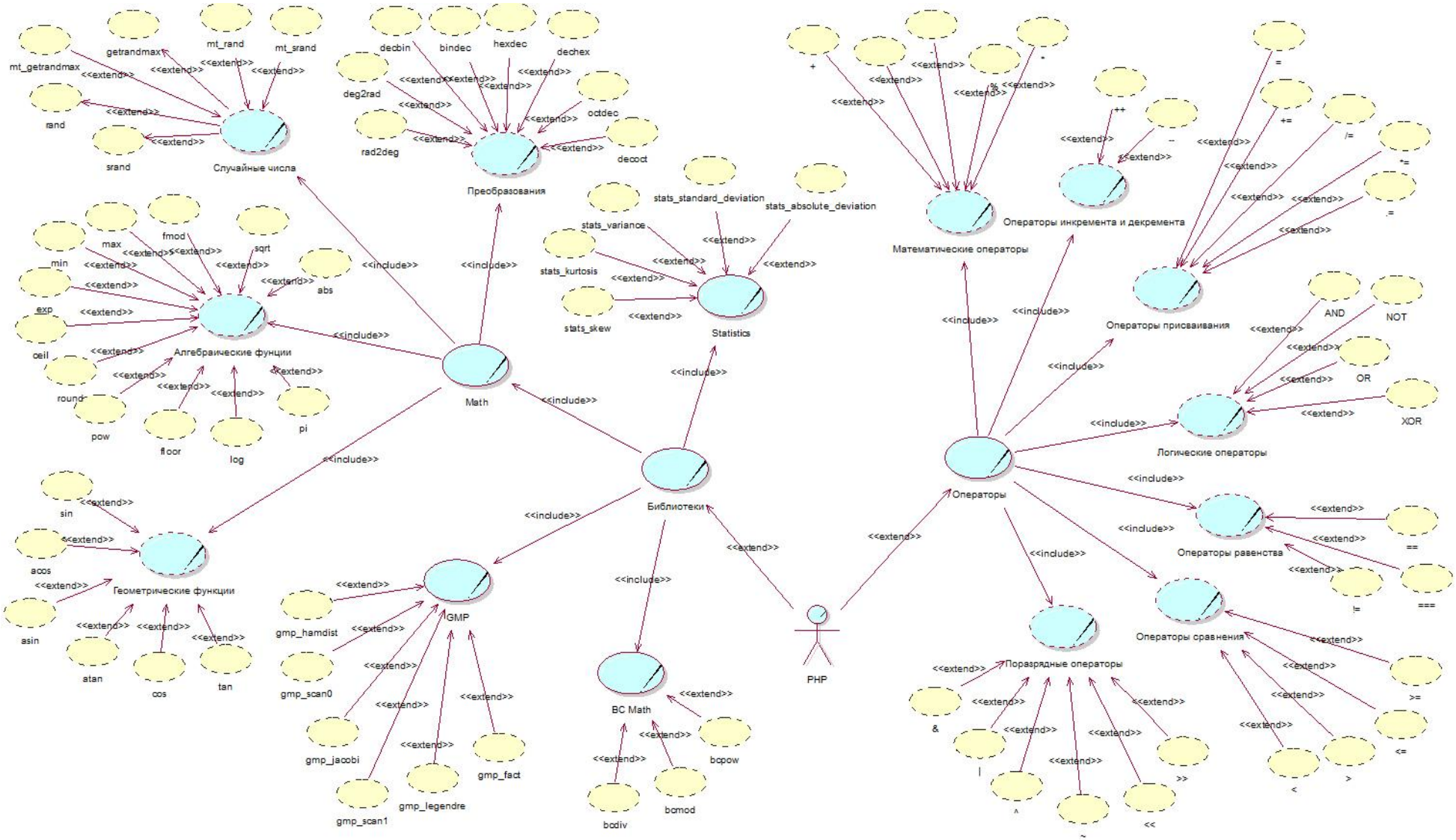

Рисунок 2 – Діаграма варіантів використання математичних функцій

– MySQL з підтримкою транзакцій (mysqld-max);
– систему управління віртуальними хостами, засновану на шаблонах;
– систему управління запуском і завершенням;
– phpMyAdmin – систему управління MySQL через Web-інтерфейс;
– ядро Perl без стандартних бібліотек (вони поставляються окремо);
– емулятор sendmail, який підтримує роботу спільно з PHP і Perl;
– інсталятор.

Розглянемо на діаграмах взаємодії як саме відбувається установка дистрибутиву Денвер. Діаграма послідовності складається чотирьох частин (рис. 3).

На діаграмі послідовності об'єкти зображуються у вигляді прямокутників на вершині пунктирною вертикальної лінії. Ця вертикальна лінія називається лінією життя (lifeline) об'єкта. Вона являє собою фрагмент життєвого циклу об'єкта в процесі взаємодії. Кожне повідомлення представляється у вигляді стрілки між лініями життя двох об'єктів.

Другим видом діаграм взаємодії є кооперативна діаграма. На кооперативної діаграмі екземпляри об'єктів показані у вигляді піктограм. Лінії між ними позначають повідомлення, обмін якими здійснюється в рамках даного варіанту використання.

Кожен вид діаграм взаємодії має свої переваги. На діаграмах послідовності робиться акцент саме на послідовності повідомлень, при цьому легше спостерігати порядок, в якому відбуваються різні події. У разі кооперативних діаграм можна використовувати просторове розташування об'єктів для того, щоб показати їх статичну взаємодію (рис. 4).

Однією з головних властивостей будь якої діаграми взаємодії є її простота. Подивившись на діаграму, можна легко побачити всі повідомлення. Однак при спробі зобразити щось більш складне, ніж єдиний послідовний процес без безлічі умовних переходів або циклів, даний підхід може не спрацювати.

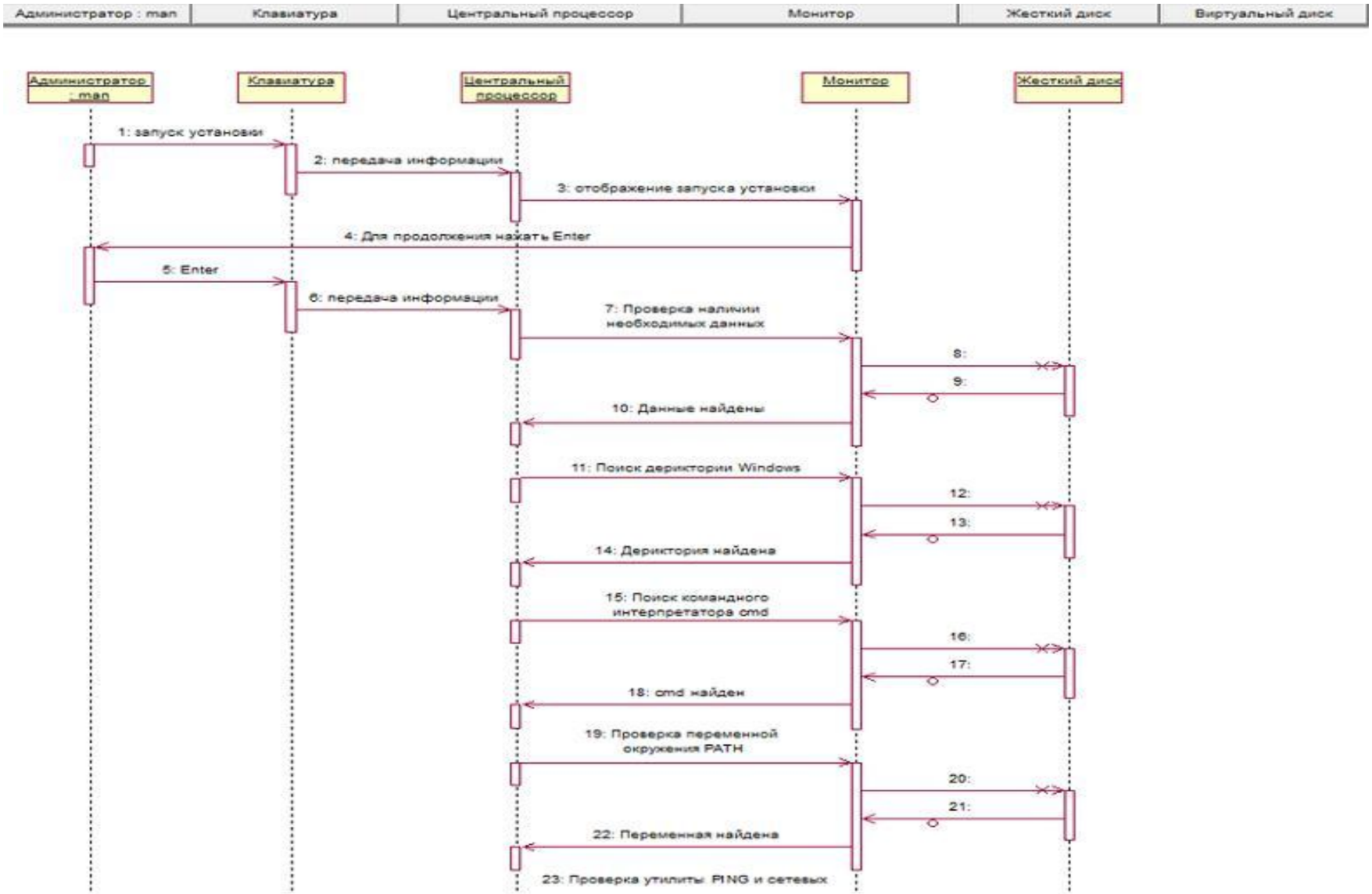

Рисунок 3 – Діаграма послідовності

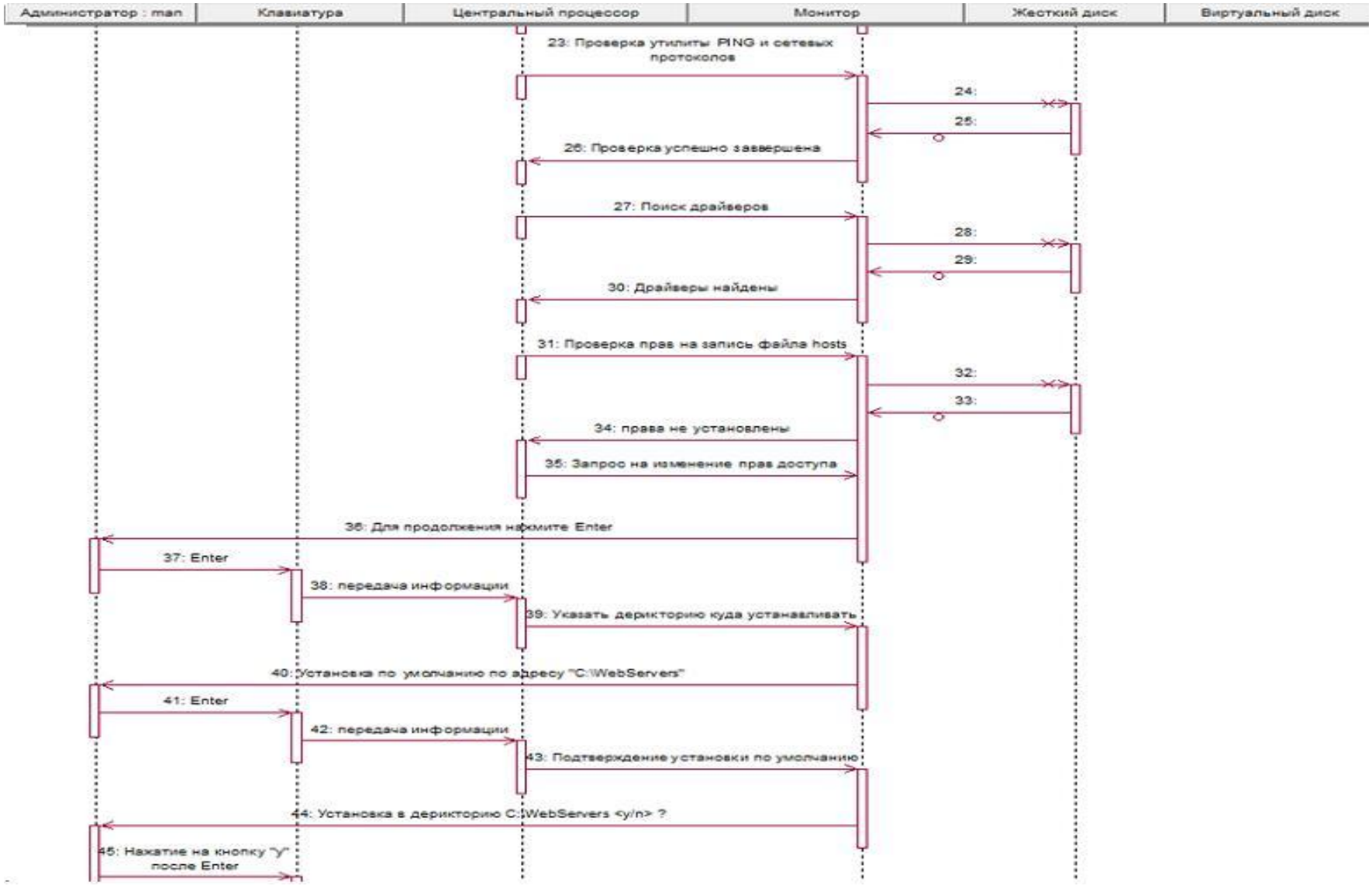

Рисунок 3, аркуш 2

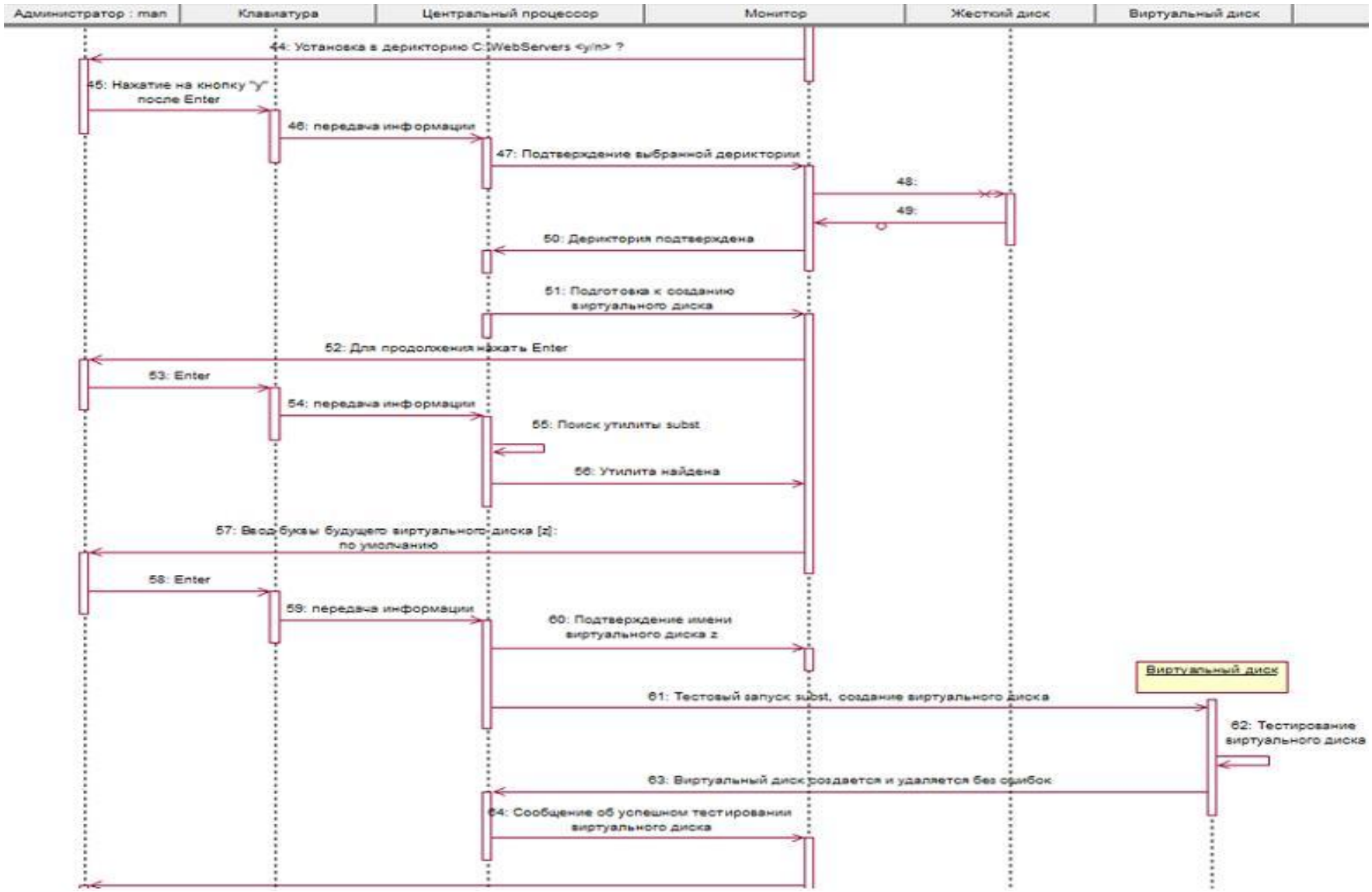

Рисунок 3, аркуш 3

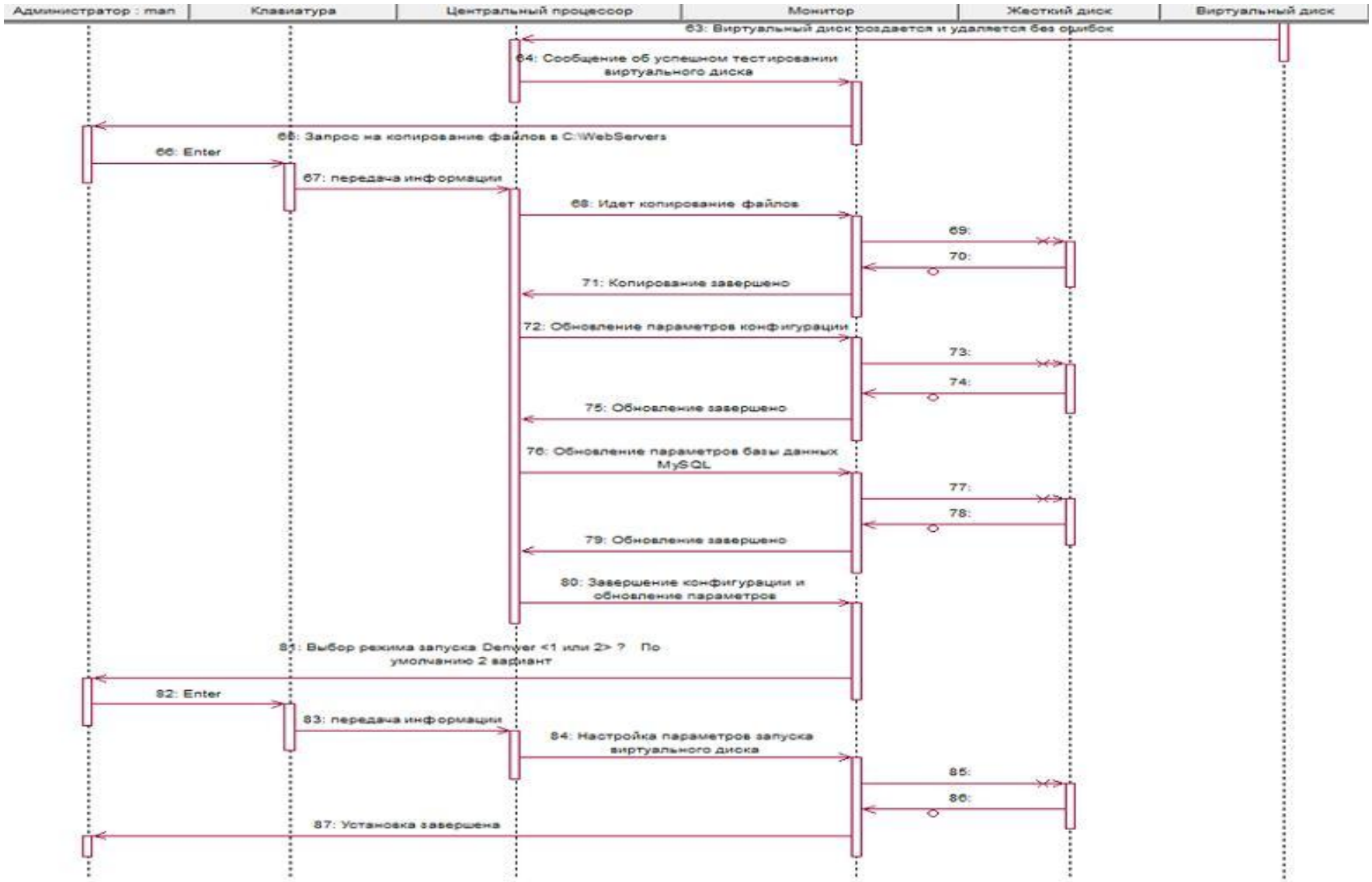

Рисунок 3, аркуш 4

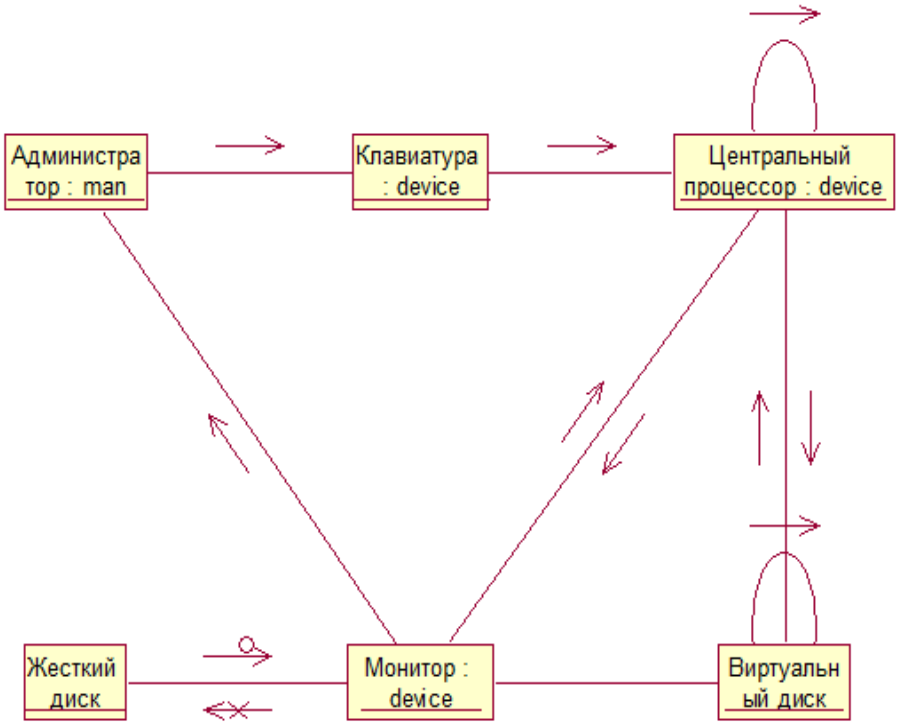

Рисунок 4 – Кооперативна діаграма

Для відображення умовної поведінки на діаграмах взаємодії існує два підходи. Один з них полягає у використанні окремих діаграм для кожного сценарію. Другий полягає в тому, що повідомлення супроводжуються умовами, що показують поведінку об'єктів.

3 РОЗРОБКА ЗАСОБІВ МОДЕЛЮВАННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

Сценарна або скріптова мова – це мова програмування, розроблена для запису «сценаріїв», послідовностей операцій, які користувач може виконувати на комп'ютері. У прикладній програмі, сценарій (скріпт) – це програма, яка автоматизує деяку задачу, яку без сценарію користувач робив би вручну, використовуючи інтерфейс програми.

Для відображення можливостей сценарних додатків розробимо програмний продукт – лічильник відвідувань веб-сайту. Як описано в першому розділі, середа розробки PHP в першу чергу використовується для створення сценарних додатків, які працюють на стороні сервера. Якщо скрипт обробляється сервером, клієнтові посилаються тільки результати роботи скрипта. Щоб досягти коректного відображення інформації про відвідувачів веб-сторінки, сценарний додаток повинен працювати на стороні сервера.

Лічильник відвідувань – це сервіс, призначений для зовнішнього незалежного вимірювання відвідуваності сайтів. Можливості лічильників можуть дуже сильно відрізняться один від одного, але всі вони зазвичай складаються з двох частин:

- код, який розміщується на сторінках веб-сайту, для збору інформації про відвідування;
- движок, який підраховує отриману інформацію та подає її у вигляді статистичного звіту.

Припустимо в нас є якась цілочисельна змінна counter, яка була оголошена в файлі labeling.php і ця змінна повинна буде збільшиться на 1 після відвідання сторінки користувачем. Визначити чи відвідав користувач сторінку з цим кодом досить таки просто. При кожному відвідуванні сторінки, інтерпретатор завантажує весь вміст сторінки. За допомогою відкриваючих і закриваючих тегів (<? php ?> або скорочений варіант <? ?>), інтерпретатор може перемикатися між режимами HTML і PHP.

Коли PHP обробляє файл, він просто передає його текст, поки не зустріне один з перерахованих спеціальних тегів, який повідомляє йому про необхідність почати інтерпретацію тексту як коду PHP. Потім він виконує весь знайдений код до закриваючого тега, що говорить інтерпретатору, що далі знову йде просто текст. Цей механізм дозволяє впроваджувати PHP-код в HTML – все за межами тегів PHP залишається незмінним, тоді як всередині інтерпретується як код.

Движок сценарного додатку – це набір класів і функцій для реалізації тих чи інших вимог. На прикладі нашого лічильника відвідувань, в стандартному движку повинна бути функція запису кількості відвідувань у файл. З цього файлу можна виводити загальну статистку відвідувань як однієї веб-сторінки так і всього веб-сайту.

3.1 Розробка технічного завдання

Технічне завдання – початковий документ на проектування технічного об'єкта. ТЗ встановлює основне призначення розроблюваного об'єкта, його технічні та тактико-технічні характеристики, показники якості і техніко-економічні вимоги, припис щодо виконання необхідних стадій створення документації (конструкторської, технологічної, програмної тощо) та її склад, а також спеціальні вимоги.

Структура і функціонал програмного продукту.

Щоб розробити такий сценарний додаток як лічильник відвідувань потрібно поставити перед собою ряд вимог, що дозволяють реалізувати необхідні можливості цього програмного продукту. На початку можна визначити головну характеристику, що визначає цей сценарний додаток. Програма повинна підраховувати відвідувачів, тоді вона буде називатися лічильником.

Лічильник може відображати кількість відвідувань у вигляді одного числа, яке з часом звичайно буде накопичуватися і тоді практичності від такої програми буде мало. Визначимо із загальної статистики відвідувань

більш детальну. З додаванням ще декількох лічильників, ми можемо отримати додаткову інформацію про кількість відвідувачів в день, тиждень, місяць, і навіть на рік. Вся інформація буде так само записуватися в файл із загальною статистикою. Але на відміну від лічильника загальної статистики, ці будуть оновлюватися раз на день, у тиждень тощо.

Для створення детальної статистики, нам потрібно взяти додаткову інформацію про кожного користувача. Основною відмінністю користувачів один від одного є ір-адреси. Що б визначити кількість унікальних користувачів використовуємо ір-адреси користувачів.

Корисною функцією для такої сценарної програми буде визначення часу і дати візиту останнього користувача. З такою функцією можна добре відсортувати всіх відвідувачів за часом останнього візиту. Якщо ір-адреса останнього відвідувача збігатиметься з однією з попередніх, замість запису нового користувача з такою ж ір-адресою будемо використовувати змінну що відповідає за кількість відвідувань. Завдяки цій змінній можна буде зменшити кількість рядків у самій статистиці користувачів позбувшись повторень. Якщо відвідувач з однією ір-адресою зайде на сторінку 3 рази, то і змінна буде зберігати в собі число «3».

Крім користувачів сторінку можуть відвідати і програми-роботи. До них можна віднести ботів пошукових систем, генераторів спаму, веб-скріперов, шпигунських програм тощо. Щоб відрізнити реального користувача від робота, нам потрібно визначити операційну систему користувача. Створивши масив з відомими операційними системами, ми зможемо визначити яка операційна система встановлена на комп'ютері користувача. Якщо збігів не знайдено порівнюємо з масивом відомих пошукових і спам роботів.

Незалежно від використовуваної користувачем операційної системи, для перегляду веб-сторінок необхідний веб-оглядач (браузер). Браузер – комплексний додаток для обробки і виведення різних складових веб-сторінки і для надання інтерфейсу між веб-сайтом і його відвідувачем. Для того щоб дізнатися який браузер використовує відвідувач, створимо масив з великою кі-

лькістю браузерів. Порівнюючи браузер користувача з елементами масиву визначимо який використовує відвідувач.

Знаючи ір-адресу користувача ми можемо визначити його ім'я хоста й його країну. Створивши список всіх країн світу, додамо кожну країну у вигляді елемента масиву. Таким чином ми отримаємо масив, елементи якого допоможуть визначити країну відвідувача. Дана інформація добре доповнить статистику і в подальшому дозволить створювати рейтинги відвідувачів по країнах і хостам.

Завдяки реферному заголовку протоколу HTTP, власник веб-сайту отримує можливість дізнатися, за якими пошуковими запитами, як часто й на які саме сторінки потрапляють люди. Використовуючи цей заголовок ми зможемо відобразити в статистиці пошуковий запит користувача, якщо він знайшов сторінку за допомогою пошукової системи. Так само ми отримаємо url посилання ресурсу з якого перейшов користувач.

Під час переміщення відвідувача сторінками веб-сайту, можна також вести облік його активності. Таким чином ми зможемо дізнатися які веб-сторінки відвідував користувач і визначити останню відвідану сторінку.

Після досягнення всіх вищенаведених можливостей можна скласти систему рейтингів. Кожен рейтинг буде складатися з наступних стовпців: найменування, кількість відвідувачів, процентне співвідношення. За допомогою інформації отриманої від кожного користувача, створимо рейтинг програмного забезпечення яке було використане відвідувачем. Другий рейтинг допоможе визначити яка операційна система частіше встановлена на комп'ютерах відвідувачів. Наступний рейтинг покаже, користувачі яких країн найчастіше заходили на веб-сторінки. Якщо веб-сторінки відвідували не тільки користувачі але й роботи, доцільно створити рейтинг цих роботів. Стовпці цього рейтингу будуть такі ж як і в інших: назва робота, кількість відвідувань, відсотки. Отримавши імена хостів відвідувачів можна створити рейтинг хостів. В залежності від структури веб-сайту і її величини з'являється необхідність

перевірки відвідування певних сторінок. За цим створимо рейтинг, який буде показувати які сторінки користувачі відвідували частіше.

Якщо користувач знайшов наш веб-сайт за допомогою пошукової системи можна скласти рейтинг ключових слів, завдяки яким був знайдений адресу веб-сайту. Останній рейтинг дозволить показати з яких джерел користувачі переходили на сайт. Цей рейтинг буде особливо корисним при рекламуванні веб-сайту, тому що можна буде відстежити роботу рекламних компаній.

Тепер у нас є всі необхідні можливості лічильника, які можна відобразити в статистиці. Щоб статистика не була надто завантажена, розділимо її на 3 частини – загальну статистику, детальну статистику і статистику за часом.

У загальну статистику помістимо ту інформацію, яка буде нести загальний характер. А саме рейтинги відвідувань, і інформацію про відвідування цей день, в цьому тижні, цього місяця, цьому році, а так само скільки всього було відвідувачів і скільки всього унікальних відвідувачів.

В детальної статистиці повинна відображатися повна інформація про кожного відвідувача. Для кращого візуального сприймання ця статистика повинна бути у вигляді таблиці і містити в собі наступні дані про користувача: дата і час відвідування, країна відвідувача, адреса відвідувача або ім'я хоста, IP адреса, кількість відвідувань, операційна система користувача, браузер і за якою посиланням перейшов користувач на цей сайт. Роботи повинні відрізнятися від звичайних користувачів, за цим рядок робота буде виділено іншим кольором.

Остання статистика повинна показувати тимчасову шкалу відвідувань. На кожному з чотирьох графіків буде видно коли користувачі проявляли активність на веб-сайті. Перший графік – кількість відвідувань за один день. На цьому графіку повинно бути показано в котрій годині користувач відвідав веб-сайт. Другий графік – кількість відвідувань за тиждень, показує в який день тижня веб-сайт відвідували частіше. Точно так само і на двох інших

графіках, в третьому повинно бути показано якого числа було більше відвідувань, а в четвертому в якому місяці.

Проаналізувавши всі розглянуті можливості сценарного додатка сформуємо чіткий список технічного завдання:

а) створити лічильник який буде підраховувати відвідувачів веб-сторінки та визначати:
   1) кількість відвідувань в день, на тиждень, на місяць;
   2) ір-адресу відвідувача;
   3) дату і час відвідування останнього відвідувача;
   4) унікальних користувачів за допомогою ір-адреси;
   5) операційну систему відвідувача;
   6) веб-браузер відвідувача;
   7) ім’я хоста та країну відвідувача;
   8) реферерне посилання (якщо така є);
   9) пошуковий запит користувача, за допомогою якого він зміг знайти веб-сторінку (якщо така є);
   10) останню відвідану сторінку користувачем.

б) створити наступну систему рейтингів для отримання докладної інформації, а саме рейтинги:
   1) браузерів;
   2) операційних систем;
   3) країн відвідувачів;
   4) роботів;
   5) хостів;
   6) відвіданих сторінок;
   7) ключових слів;
   8) Рейтинг джерел

в) Показати результат лічильників у:
   1) загальній статистиці;
   2) детальній статистиці;

3) статистиці за часом.

3.2 Розробка фізичної моделі програмного забезпечення

Кінцевим результатом дипломної роботи є розроблений сценарний додаток, що дозволяє відстежити статистику відвідуваності на веб-сайті. Структуру файлів цього додатка розберемо на діаграмі компонентів (рис. 5). В основній папці лежать папки конфігурації (conf), таблиць стилів сайту (css), папка з необхідними зображеннями (images), папка з мовами (language), папка з необхідними для роботи даними (lib) і папка для зберігання записаної інформації (var) . Крім папок там є файли формату php:

- constants.php містить константи адреси каталогів, шляху до файлів, кількість файлів лічильника, ім'я файлу лічильника тощо;
- index.php відкривається в першу чергу коли відвідувач зайшов на сайт і типово показує загальну статистику;
- log_processor.php відповідає за процес створення списку відвіданих сторінок, за запис і чистку хостів;
- mark_page.php відзначає відвідану сторінку і записує відвідування в файл зберігання статистики;
- show_config.php дозволяє переглянути сторінку конфігурації програми;
- show_detailed.php дозволяє побачити веб-сторінку детальної статистики;
- show_global.php дозволяє побачити веб-сторінку загальної статистики;
- show_time.php дозволяє побачити веб-сторінку статистики за часом.

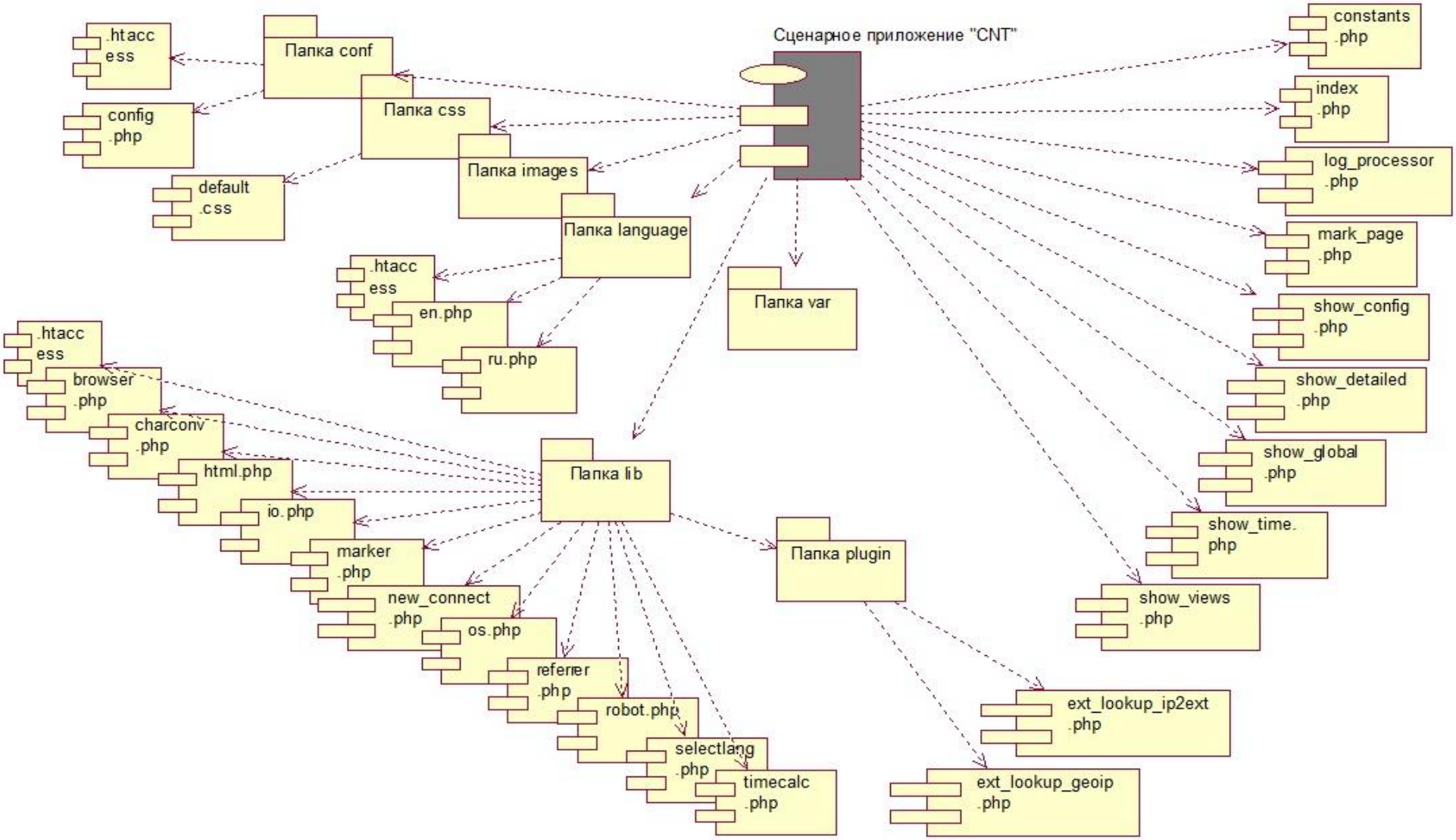

Рисунок 5 – Діаграма компонентів структури файлів сценарного додатку

3.2.1 Розробка можливостей лічильника

Використовуючи ООП для реалізації цього додатку, необхідно оголосити два класи. Розглянемо ці класи на наступній діаграмі (рис. 6).

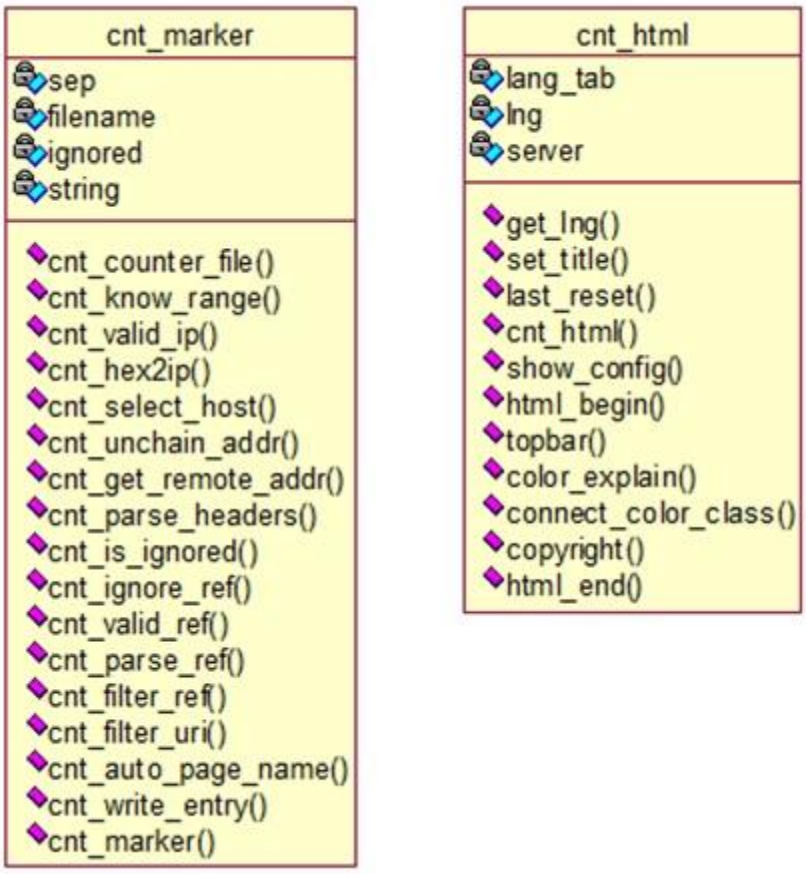

Рисунок 6 – Діаграма класів (оголошення класів)

Перший клас – cnt_marker, розташований за адресою: /lib/marker.php, де він і оголошується. Цей клас відповідає за визначення відвідувача веб-сайта. Об'єктами класу будуть самі користувачі, які відвідали веб-сайт. Функції цього класу будуть виконувати деякі з вимог до можливостей програми, які були вказані в технічному завданні.

Розглянемо на наступній діаграмі (рис. 7) всі елементи масивів які дають можливість визначити браузер, операційну систему та робота.

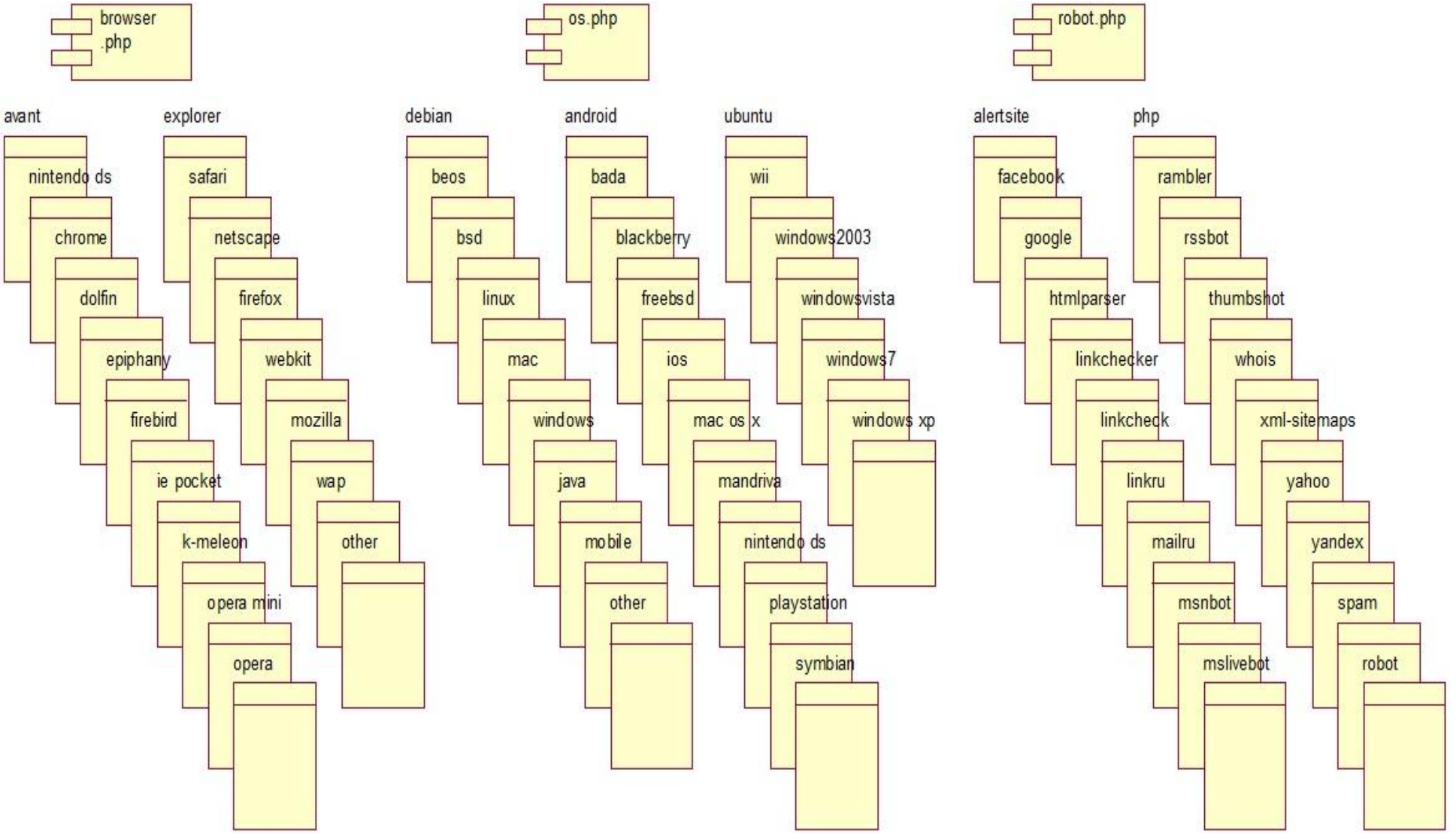

Рисунок 7 – Діаграма масивів елементів, що визначають браузер, операційну систему та робота

3.2.2 Реалізація статистики й створення систем рейтингів

Для реалізації сторінок з загальної, детальної та статистики з часу використовуємо функції з оголошеннго раніше класу cnt_html. Використовуючи функції початку всіх html документів (html_begin) і панелі навігації (topbar) створимо першу, початкову частину для цих трьох веб-сторінок. У другій частині буде основний вміст сторінок, для загальної статистики це система рейтингів і статистика відвідувань, для детальної таблиця з повною інформацією про кожного відвідувача, а для статистики за часом це графіки відвідувань. Завершальною частиною цих сторінок будуть функції: яка містить інформацію про творця програми (copyright), функція панелі навігації (topbar), інформація про час завантаження сторінки, функція кінця html документа (html_end).

Розглянемо детальніше використовувані функції для відображення загальної статистики (show_global.php) в табл. 1.

Таблиця 1 – Функції show_global.php

Назва функції	Опис
cnt_histcalc	Підрахунок статистики
cnt_rank_head	Створення заголовка рейтингу
cnt_list_item	Створення одного елемента рейтингу
cnt_rank_sum	Загальна сума рейтингу
cnt_refgen	Визначення імені реферера
cnt_sort_page_count	Сортування сторінок лічильника
cnt_show_browser	Створення рейтингу браузерів
cnt_show_os	Створення рейтингу операційних систем
cnt_show_extension	Створення рейтингу країн
cnt_show_robot	Створення рейтингу роботів
cnt_show_top_hosts	Створення рейтингу хостів

Кінець таблиці 1

Назва функції	Опис
cnt_show_top_pages	Створення рейтингу відвідуваних сторінок
cnt_show_top_origins	Створення рейтингу джерел
cnt_show_top_keys	Створення рейтингу ключових слів
cnt_show_access	Загальна інформація про відвідини

У кожній функції створення рейтингу так само використовуються й інші функції, що дозволяють значно зменшити розміри програмного коду. До таких функцій можна віднести cnt_rank_head, cnt_list_item, cnt_rank_sum, для рейтингу джерел використовується додаткова функція cnt_refgen, для рейтингу відвідуваних сторінок функція cnt_sort_page_count, а для підрахунку загальної інформації про відвідувачів використовується функція cnt_histcalc.

4 ЕКСПЕРИМЕНТАЛЬНА ПРОГРАМНА РЕАЛІЗАЦІЯ

4.1 Об'єктно-орієнтовані можливості PHP

Стратегію ООП найкраще описати як зміщення пріоритетів в процесі програмування від функціональності додатку до структур даних. Це дозволяє програмісту моделювати у створюваних додатках реальні об'єкти і ситуації. Технологія ООП володіє трьома головними перевагами:

– вона легка для розуміння – ООП дозволяє мислити категоріями щоденних об’єктів;

– надійна і проста для супроводу – правильне проектування забезпечує простоту розширення і модифікації об'єктно-орієнтованих програм. Модульна структура дозволяє вносити незалежні зміни в різні частини програми, зводячи до мінімуму ризик помилок програмування;

– прискорює цикл розробки – модульність і тут відіграє важливу роль, оскільки різні компоненти ОО – програм можна легко використовувати в інших програмах, що зменшує надмірність коду і знижує ризик внесення помилок при копіюванні.

Специфіка ООП помітно підвищує ефективність праці програмістів і дозволяє їм створювати більш потужні, масштабовані та ефективні додатки. Більшість переваг ООП обумовлені одним з його фундаментальних принципів - інкапсуляцією. Інкапсуляцією називається включення різних дрібних елементів в більш великий об'єкт, в результаті чого програміст працює безпосередньо з цим об'єктом. Це призводить до спрощення програми, оскільки з неї виключаються другорядні деталі.

Інкапсуляцію можна порівняти з роботою автомобіля з точки зору типового водія. Багато водіїв не розбираються в подробицях внутрішнього пристрою машини, але при цьому управляють нею саме так, як було задумано. Хай вони не знають, за якими принципами побудований двигун, гальмо або рульове керування – існує спеціальний інтерфейс, який автоматизує і спро-

щує ці складні операції. Сказане також відноситься до інкапсуляції і ОВП – багато подробиць «внутрішнього устрою» ховаються від користувача, що дозволяє йому зосередитися на вирішенні конкретних завдань. У ООП ця можливість забезпечується класами, об'єктами і різними засобами вираження ієрархічних зв'язків між ними (класи і об'єкти розглядаються нижче).

4.2 PHP і ООП

Хоча PHP володіє загальними об'єктно-орієнтованими можливостями, він не є повноцінною об’єктно-орієнтованою мовою (наприклад, такою, як C++ або Java). Зокрема, в PHP не підтримуються наступні об'єктно-орієнтовані можливості:

– множинне спадкування;

– автоматичний виклик конструкторів (якщо ви хочете, щоб при конструюванні об'єкта похідного класу викликався конструктор базового класу, вам доведеться викликати його явно);

– абстрактні класи;

– перевантаження методів;

– перевантаження операторів (це пов'язано з тим, що PHP є мовою з вільною типізацією (за додатковою інформацією звертайтеся до глави 2);

– закритий і відкритий доступ, віртуальні функції;

– деструктори;

– поліморфізм.

Але і без усього перерахованого все одно можна отримати користь з об'єктно-орієнтованих можливостей, підтримуваних PHP. Реалізація ООП в PHP надає колосальну допомогу в модульному оформленні функціональності вашої програми.

4.2.1 Створення класів у PHP

PHP підтримує широкі об'єктно-орієнтовані можливості, повна підтримка яких була введена в п'ятій версії мови.

Клас в PHP оголошується за допомогою ключового слова class. Методи і поля класу можуть бути загальнодоступними (public, за замовчуванням), захищеними (protected) і прихованими (private). PHP підтримує всі три основних механізми ООП - інкапсуляцію, поліморфізм і успадкування (батьківський клас вказується за допомогою ключового слова extends після імені класу). Підтримуються інтерфейси (ставляться у відповідність з допомогою implements). Дозволяється оголошення фінальних, абстрактних методів і класів. Множинне спадкування класів не підтримується, проте клас може реалізовувати декілька інтерфейсів. Для звернення до методів батьківського класу використовується ключове слово parent.

Починаючи з версії 5.4.0 множинне спадкування може бути реалізовано за допомогою механізму особливостей (англ. trait). Особливості схожі на домішки (англ. mixins), за винятком того що для них не можна безпосередньо створити екземпляр. Повторне використання коду укладено у використанні коду особливості в декількох класах. Допускається використовувати в одному класі кілька особливостей. Механізм особливостей має засоби розв'язання конфліктів імен. При запуску програми код особливості буде міститися в коді класу.

PHP – це об’єктно-орієнтована мова, і можливостей в цьому напрямі дуже багато. Основою ООП є об’єкт, а фабрикою для створення об’єктів є клас.

Клас складається з наступних частин:

– властивості;
– конструктор;
– методи.

Класи в PHP мають ряд спеціальних методів (англ. magic methods), що починаються з двох символів підкреслення. Особливо варто відзначити конструктор (construct (), у версіях до 5.0 конструктором служив метод, однойменний з класом) і деструктор (destruct ()), а також методи читання (get ()) і запису (set ()), згортання (sleep ()) і розгортання (wake ()), клонування (clone ()) та ін Ці методи є досить гнучким інструментом: перевизначаючи їх, можна добитися істотного зміни поведінки об'єкта.

4.2.2 Створення опису класів

Клас – це базове поняття в ООП. Якщо сказати простіше, то клас – це своєрідний тип змінної. Екземпляр класу – це об'єкт. Об'єкт - це сукупність даних (властивостей) і функцій (методів) для їх обробки. Дані та методи називаються членами класу. Взагалі, об'єктом є все те, що підтримує інкапсуляцію.

Усередині об'єкту дані і код (члени класу) можуть бути або відкриті, або ні. Відкриті дані і члени класу є доступними для інших частин програми, які не є частиною об'єкта. А ось закриті дані і члени класу доступні тільки усередині цього об'єкту.

Опис класів в PHP починається службовою командою class:

class Ім’я_класу { // Опис членів класу – даних і методів для їх обробки }

Для оголошення об'єкту необхідно використовувати оператор new:

Об'єкт = newІм*я_класу;

Дані в PHP5 описуються за допомогою службових слів public, protected, або private. Метод описується так само, як і звичайна функція. Методу також

можна передавати параметри. Був створений пустий клас з іменем «MyClass»:

```
<?php
    class MyClass
 {
}
 ?>
```

4.2.3 Класи, об’єкти і об’явлення методів

Класи утворюють синтаксичну базу об’єктно-орієнтованого програмування. Їх можна розглядати як свого роду «контейнери» для логічно зв’язаних даних і функцій (називаються методами). Клас представляє собою шаблон, по якому створюються конкретні екземпляри, які використовуються в програмі. Екземпляри класів називаються об’єктами (по аналогії з тим, як змінна

Клас також можна розглядати як тип даних, а об’єкт – як змінну (по аналогією з тим, як змінна $counter відноситься до цілого, а змінна $last_name – до строкового типу). Програма може одночасно працювати з декількома об’єктами одного класу як з декількома змінними цілого типу. Загальний формат класів PHP, який приведений в лістингу коду 4.1.

Лістинг 4.1 – Об’явлення класів в PHP

```
class Class_name
{
var $attribute_1;
...
var $attribute_N;
function function1()
{
...
}
...
```

```
function functionN()
{
...
}
// end Class_name
```

Тобто, об’явлення класу повинне починатися з ключового слова class (подібне до того, як об’явлення функції починається з ключового слова function). Кожному об’явленню атрибута, який міститься в класі, повинне передкувати ключове слово var. Атрибути можуть відноситися до будь-якого типу даних, що підтримуються в PHP; їх можна розглядати як змінні невеликими відмінностями. Після об’явлення атрибутів слідує об’явлення методів, які дуже схожі на типові об’явлення функцій.

По загальноприйнятим правилам об’явлення класів ООП починаються з прописної літери, а всі слова в іменах методів, окрім першого, починаються з прописних літер (перше слово починається зі строкової літери). Використовувати будь-які позначення, головне – вибрати стандарт та дотримуватися його.

Методи часто використовуються для роботи з атрибутами класів. При посиланнях на атрибути всередині методу використовується спеціальна змінна $this. Синтаксис методів продемонстрований в наступному прикладі:

```
<?
class Webpage
{
var $bgcolor;
function set BgColor($color)
{
$this->bgcolor = $color;
}
function get BgColor()
{
return $this->bgcolor;
}
```

```
}
?>
```

Змінна $this посилається на екземпляр об’єкту, для якого був викликаний метод. Оскільки в будь-якому класі може існувати декілька екземплярів об’єктів, уточнення $this необхідне для посилань на атрибути, які належать поточному об’єкту. При використанні цього синтаксису необхідно звернути увагу на дві обставини:

- атрибут, на який вказує посилання в методі, не потрібно передавати в вигляді параметра функції;
- знак долара ($) ставиться перед змінною $this, але не перед ім’ям атрибуту (як у звичайній змінній).

4.2.4 Створення об’єктів і робота з ними

Об’єкти створюються оператором new. Наприклад, об’єкт створеного класу Web page створюється наступною командою:

```
$home_page = newWebpage;
```

Новий об’єкт з ім’ям $some_pageмає власний набір атрибутів і методів, які перелічені в класі Webpage. Для зміни значень атрибуту $bgcolor, що належить до конкретного об’єкту, можна скористатися певним в класі методом setBgColor( ):

```
$some_page->setBgColor("black");
```

Слідує розуміти, що PHP також дозволяє отримати в явній мірі значення атрибуту з указанням імен об’єкту і атрибуту:

```
$some_page->bgcolor;
```

Однак, другий спосіб суперечить принципу інкапсуляції, і при роботі з ООП дотримуватися такого принципу не слідує.

4.2.5 Конструктори

Дуже часто при створенні об’єкту потрібно задати значення деяких атрибутів. Розробники технології врахували цю обставину і реалізували її в концепції конструктору. Конструктор представляє собою метод, який задає значення деяких атрибутів (а також може викликати інші методи). Конструктори викликаються автоматично при створенні нових об’єктів. Щоб це стало можливим, ім’я методу-конструктору повинне співпадати з ім’ям класу, в якому він міститься. Приклад конструктору наведений в лістингу 4.2.

Лістинг 4.2 – Використання конструктору

```
<?
class Webpage
{
var $bgcolor;
function Webpage($color)
{
$this->bgcolor = $color;
} }
// Викликати конструктор класу Webpage
$page = newWebpage("brown");
?>
```

В ранніх версіях створення об’єкту і ініціалізації атрибутів виконувалося роздільно. Конструктори дозволяють виконувати ці дії за один етап.

В залежності від кількості параметрів, що передаються, можуть бути викликані різні конструктори. Наприклад, в лістингу 4.2 показано, що в залежності від кількості об’єктних параметрів, які передаються, можуть викликатися різні конструктори. Наприклад, в лістингу 6.2 об’єкти класу Webpage можуть створюватися двома способами. По-перше, можна викликати конструктор, який просто створює об’єкт, але не ініціалізує його атрибути:

```
$page = newWebpage;
```

По-друге, об’єкт можна створити за допомогою конструктора, визначеного в класі, – в цьому випадку створюється об’єкт класу Webpage і його значення присвоюється атрибуту bgcolor:

```
$page = newWebpage("brown")
```

4.2.6 Деструктори

В PHP відсутня безпосередня підтримка деструкторів. Тим не менш, можна легко імітувати роботу деструктора, викликаючи функція PHP unset( ). Ця функція знищує вміст змінної і повертає системі ресурси, які вона займає. З об’єктами unset()працює так само як і з змінними. Наприклад, в випадку роботи з об’єктом $Webpage, після завершення роботи з цим конкретним об’єктом викликається функція unset ($Webpage), яка видаляє з пам’яті весь зміст $Webpage. Діючи в дусі інкапсуляції, можна помістити виклик unset( ) в метод з ім’ям destroy( ), а потім викликати його:

```
$Website->destroy();
```

Необхідність в виклику деструкторів виникає лише при роботі з об’єктами, які використовують великий об’єм ресурсів, оскільки всі змінні і об’єкти автоматично знищуються при завершенні сценарію.

4.2.7 Статичні поля класу

Як звичайна локальна змінна, статична змінна доступна тільки в межах функції. Тим не менш, на відміну від звичайних локальних, статичні змінні зберігають значення між викликами функції.

Статичні поля класу працюють за таким же принципом. Статичне поле класу пов'язано зі своїм класом, проте воно зберігає своє значення протягом всієї роботи скрипта. Порівняйте це з звичайними полями: вони пов'язані з певним об'єктом, і вони губляться при видаленні цього об'єкта.

Статичні поля корисні у випадках, коли вам потрібно зберігати певне значення, що відноситься до всього класу, а не до окремого об'єкту. Вони схожі на глобальні змінні класу.

Щоб створити статичну змінну, додайте ключове слово static в її завданні:

```
class MyClass
{
public static$myProperty;
}
```

Наведемо приклад того, як працюють статичні змінні:

```
class Member
{
private $username;
public static $numMembers = 0;

public function__construct( $username )
{
   $this->username = $username;
self::$numMembers ++;
 }
}

echo Member::$numMembers . "<br>";  // відобразить "0"
$a Member = newMember( "fred" );
echo Member::$numMembers . "<br>";  // відобразить "1"
$another Member = new Member( "mary" );
echo Member::$numMembers . "<br>";  // відобразить "2"
```

У класі Member два поля: приватне поле $ username і статичну $ num Members, яке спочатку отримує значення 0.

Конструктор отримує як параметр аргумент $ username і встановлює полю щойно створеного об'єкта значення цього параметра. У той же час, він збільшує значення поля $ num Members, тим самим даючи зрозуміти, що число об'єктів нашого класу збільшилася на 1.

Конструктор звертається до статичного полю так: self :: $ num Members. Ключове слово self схоже на $this, яке ми розглянули в минулому уроці. Тоді як $ this посилається на поточний об'єкт, self – на поточний клас. Також тоді як для отримання доступу до полів і методів об'єкту ви використовуєте ->, то в цьому випадку використовуйте :: для отримання доступу до полів і методів класу.

По завершенню скрипт створює кілька об'єктів класу Member і відображає на сторінці їх кількість, тобто значення статичної змінної $ numMembers. Відзначте, що дана змінна зберігає своє значення протягом всієї роботи скрипта, незважаючи на об'єкти класу.

Отже, щоб отримати доступ до статичного поля класу, використовується оператор «::». Тут ми не можемо скористатися ключовим словом self, так як код знаходиться за межами класу, тому ми пишемо ім'я класу, потім ::, а потім ім'я поля (Member :: $ numMembers). У межах конструктора теж потрібно використовувати саме таку структуру, а не self.Немає необхідності створювати об'єкти класу для того, щоб користуватися його статичними полями.

4.2.8 Просте і ієрархічне наслідування

Клас є шаблоном, по якому створюються реальні об’єкти з певними характеристиками і функціями. Неважко уявити ситуацію, при якій такий об’єкт є частиною другого об’єкту. Наприклад, автомобіль можна вважати приватним випадком категорії «транспортний засіб», до якого відносяться і літаки. Хоча різні типи транспортних засобів дуже відрізняються один від

одного, всі вони характеризуються атрибутами з загального набору (кількість коліс, потужність, максимальна швидкість, модель тощо). Нехай конкретні значення цих атрибутів дуже відрізняються – всі атрибути одно властиві всім транспортним засобам. Таким чином, субкласи «автомобіль» і «літак» наслідують загальний набір базових характеристик від суперкласу «транспортний засіб» Концепція отриманих класом характеристик від іншого, більш загального класу називається наслідуванням.

Наслідування є виключно корисним засобом програмування, оскільки його застосування запобігає копіюванню коду, сумісно використаних структурами даних, – наприклад, загальних характеристик різних типів транспортних засобів, які були вище перелічені. В загальному випадку синтаксис наслідування характеристик іншого класу в PHP виглядає наступним чином:

```
class Class_name2 extends Class_name1
{
оголошення атрибутів;
оголошення методів;
}
```

Ключове слово extends вказує на те, що клас Class_name2 наслідує всі характеристики класу Class_name1.

Окрім можливості багаторазового використання коду, наслідування володіє ще однією важливою перевагою – знижується вірогідність помилок при модифікації програми. Наприклад, в ієрархії, яка зображена на рис. 4.1, зміни в класі «автомобіль» ніяк не позначаться на коді (і даних) класу «літак», і навпаки. Виклик конструктору похідного класу не призводить до автоматичного виклику конструктору базового класу.

В лістингу коду (додаток А.1) наведені класи, що моделюють ієрархію, що зображена на рис. 8. Об’єкти цих класів створюються наступним чином:

```
$tractor = new Vehicle;
$gulfstream = new Airplane;
```

Рисунок 8 – Ієрархія транспортних засобів

Наведені команди створюють два об’єкти. Перший об’єкт, $tractor, відноситься до класу Vehicle. Другий об’єкт, $gulfstream, відноситься до класу Airplane і тому володіє загальними характеристиками класу Vehicle, так і уточненими характеристиками класу Airplane.

Ситуація, за якої клас наслідує властивості деяких батьківських класів, яке називається множинним спадкуванням. На жаль, в PHP множинне спадкування не підтримується. Для прикладу, виконання наступної конструкції неможливе в PHP:

```
classAirplaneextendsVehicleextendsBuilding.
```

4.2.9 Множинне наслідування

Зі збільшенням розмірів складності програм може виникнути необхідність в множинному наслідуванні. Клас буде наслідувати свої властивості від інших класів, які в свою чергу, будуть наслідувати властивості від третіх класів тощо. Множинне наслідування розвиває модульну структуру програми, забезпечуючи простоту супроводу і більш чітку логічну структуру. При використанні прикладу з транспортними засобами в великій програмі може виникнути необхідність в додатковому розбитті на субкласи суперкласу Vehicle, що продовжить логічний розвиток ієрархії. Наприклад, транспортні засоби можна поділити на наземні, морські та повітряні, щоб суперклас спе-

ціалізованих субкласів вибирався в залежності від середовища, в яке переміщується даний транспортний засіб. Оновлений варіант ієрархії представлений на рис. 9.

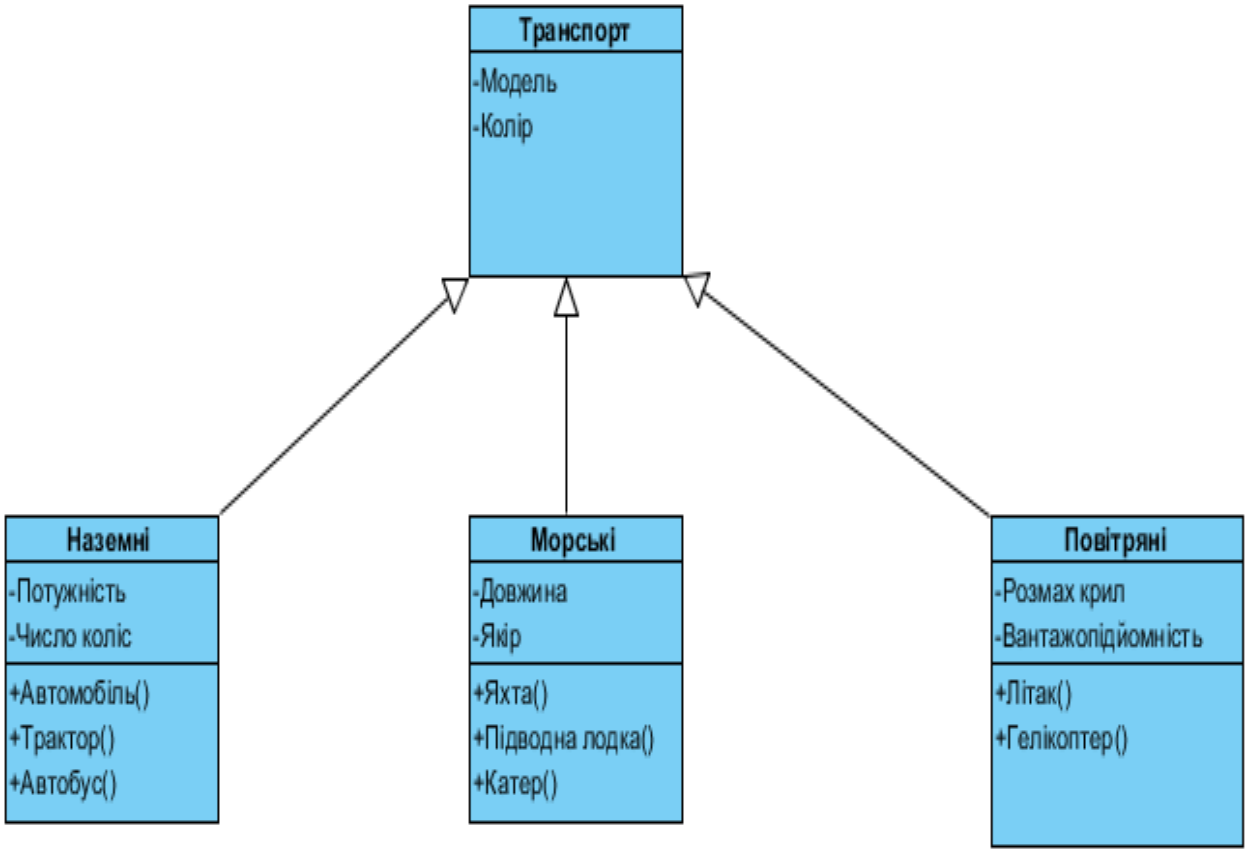

Рисунок 9 – Багаторівневе наслідування в ієрархії Vehicle

Приклад, приведений в лістингу 4.3 підкреслює деякі важливі аспекти багаторівневого наслідування в PHP.

Лістинг 4.3 – Багаторівневе наслідування

```
<?
class Vehicle
 {
//Об’явлення атрибутів...
//Об’явлення методів...
}
class Land extends Vehicle
```

```
{
// Об’явлення атрибутів...
// Об’явлення методів...
}
class Car extends Land
{
// Об’явлення атрибутів...
// Об’явлення методів...
}
$nissan = newCar;
?>
```

Об’єкт $nissan містить всі атрибути і методи класів Car, LandiVehicle. Виходячи з цього, програма є виключно модульною. Припустимо, що в майбутньому до програми в клас Land необхідно буде додати новий атрибут. Для цього, необхідно буде внести відповідні зміни в клас Land, і цей атрибут негайно стане доступним для класів Land і Car, не впливаючи на функціональність інших класів. Таким чином, модульність коду і гнучкість відносяться до числа основних переваг ООП.

Хоча при багаторівневому наслідуванні відбувається наслідування своїх характеристик від ланцюга батьків, конструктори батьківських класів не викликаються автоматично при створенні об’єктів класу-наслідника. Ці конструктори можуть викликатися класом – наслідником в вигляді методів.

4.2.10 Абстрактні класи та робота з ними

В деяких ситуаціях буває зручно створювати клас, об’єкти якого ніколи не створюються (даний клас потрібен лише як базовий клас для створення похідних класів). Такі класи називаються абстрактними, які зазвичай використовуються в випадках, коли розробник програми бажає забезпечити обов’язкову підтримку деяких функціональних можливостей всіма класами, похідними від абстрактного базового класу.

В PHP відсутня пряма підтримка абстрактних класів, проте існує просте обхідне рішення – достатньо визначити в «абстрактному» класі конструктор і включити в нього виклик команди die( ). Повернемося до класу з лістингу 4.3. Для представлення реальних об’єктів (наприклад, автомобілів) необхідно створювати клас, похідний від цих класів. Отже, щоб запобігти можливому створенню об’єктів класів LandiVehicle, необхідно включити в їх конструктори виклики die(), як це показано в лістингу коду (додаток А.2).

Спроба створення екземпляра цих абстрактних класів призведе до видачі повідомлення про помилку і завершення програми.

4.2.11 Перевантаження методів

Перевантаженням методів називається визначення деяких методів з однаковими іменами, але різною кількістю або типами параметрів. Як і в випадку з абстрактними класами, в PHP ця можливість не підтримується, але існує просте обхідне рішення, приведене в лістингу (додаток А.3).

Результат роботи скрипта наведено на рис. 10.

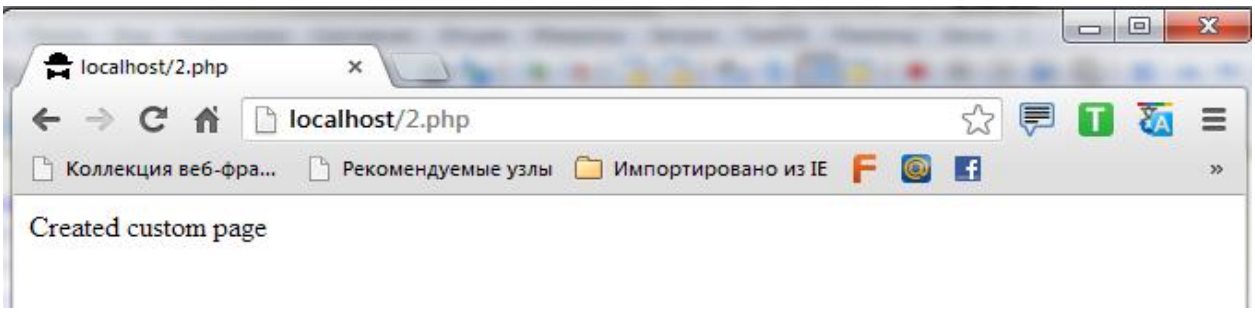

Рисунок 10 – Результат роботи коду

В приведеному прикладі при створенні нового об’єкту з ім’ям $html_page передається один аргумент. Оскільки в класі був визначений конструктор за умовчуванням (Page( )), викликається саме він. Однак, конструктор за умовчанням всього лише вибирає, якому з конструкторів (Page0( ) або Page1( ))слідує передати управління. При виборі конструктора використ-

товуються функції func_num_args( )ifunc_get_arg( ), які, відповідно, визначають кількість аргументів і читають ці аргументи.

4.2.12 Функції для роботи з класами та об’єктами

В PHP існує декілька стандартних функцій для роботи з класами та об’єктами. Всі вони часто використовуються, особливо в процесі розробки інтерфейсу, адміністрування і діагностування помилок.

```
get_class_methods()
```

Функція get_class_methods( ) повертає масив імен методів класу із заданим ім'ям. Синтаксис функціїget_class_methods():

```
arrayget_class_methods (stringім’я_класу)
```

Простий приклад використання get_class_methods() приведений в лістингу 4.4.

Лістинг 4.4 – Отримання списку методів класу

```
<?
...
class Airplane extends Vehicle
{
var $wingspan;
function set Wing Span($wingspan)
{
$this->wingspan = $wingspan;
}
function get WingSpan()
{
return $this->wingspan;
}
}
```

```
$cls_methods = get_class_methods(Airplane);
// Масив $cls_methods містить імена всіх методів,
// об’явлення в класах "Airplane" і "Vehicle"
?>
```

Як видно з лістингу 4.4, функція get_class_methods( ) дозволяє легко отримати інформацію про всі методи, які підтримуються класом.

```
get_class_vars()
```

Функція get_class_vars( ) повертає масив імен методів класу із заданим ім'ям. Синтаксис функції get_class_vars():

```
arrayget_class_vars (string ім’я_класу)
```

Приклад використання get_class_vars() наведений в лістингу коду 4.5.

Лістинг 4.5 – Отримання списку атрибутів класу функцією get_class_vars()

```
<?
classVehicle
{
var $model;
var $current_speed;
}
classAirplaneextendsVehicle
{
var $wingspan; } $a_class = "Airplane";
$attribs = get_class_vars($a_class);
// $attribs = array ( "wingspan", "model", "current_speed")
?>
```

Масив $attribs заповнюється іменами всіх атрибутів класу Airplane.

```
get_object_vars()
```

Функція get_object_vars() повертає асоціативний масив з інформацією про всі атрибути об’єкту з заданим ім’ям. Синтаксис функції get_object_vars():

```
arrayget_object_vars (object ім’я_об’єкту)
```

Приклад використання функції get_object_vars() наведений в лістингу (додаток А.4). Результат роботи скрипта наведено на рис. 11.

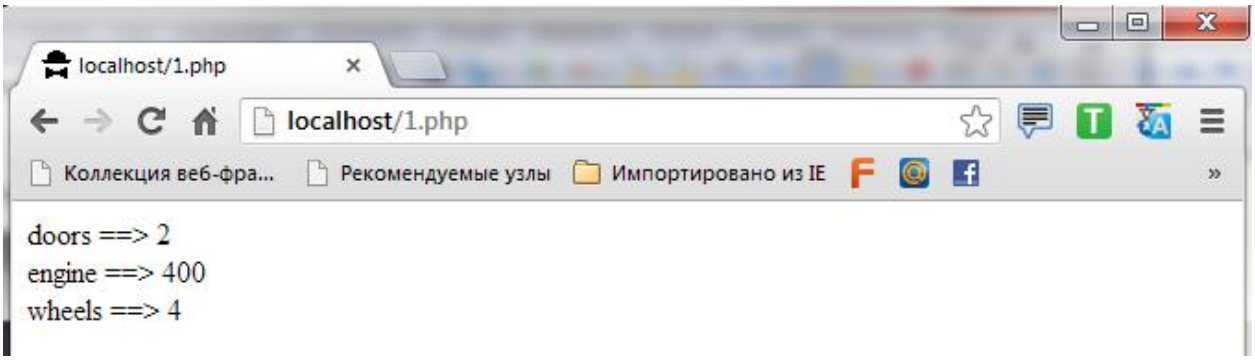

Рисунок 11 – Результат роботи скріпта

Функція get_object_vars() дозволяє швидко отримати інформацію про атрибути конкретного об’єкту і їх значення в вигляді асоціативного масиву.

```
method_exists()
```

Функція method_exists() перевіряє, чи підтримується об’єктом метод з заданим ім’ям. Якщо метод підтримується, функція повертає TRUE, в іншому випадку повертається FALSE. Синтаксис функції method_exists():

```
bool method_exists (object ім’я_об’єкту, string
ім’я_методу)
```

Приклад використання методу method_exists() наведений в лістингу (додаток А.5). Результат роботи скрипта наведено на рис. 12.

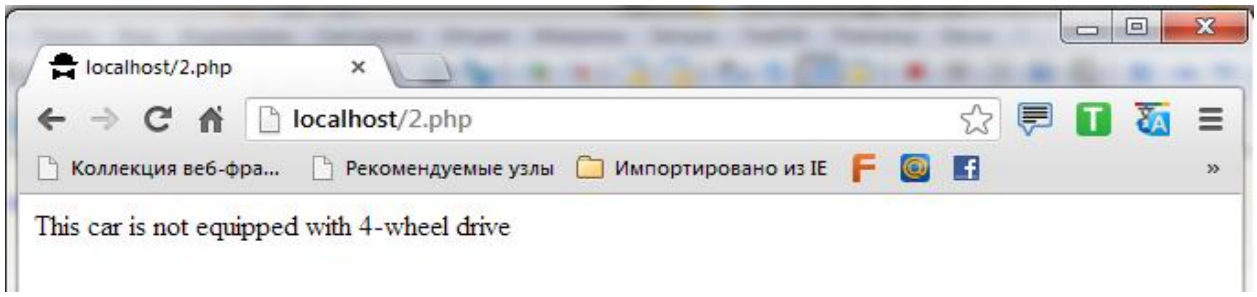

Рисунок 12 – Результат роботи скрипта

В лістингу з додатка А.5 функція method_exists () перевіряє, чи підтримується об’єктом $car метод з ім’ям set Four Wheel Drive(). Якщо метод підтримується, функція повертає логічну істину і фрагмент виводить відповідне повідомлення. В іншому випадку повертається FALSE і виводиться інше повідомлення.

```
get_class( )
```

Функція get_class( ) повертає ім’я класу, до якого відноситься об’єкт з заданим ім’ям. Синтаксис функції get_class( ):

```
stringget_class (object ім’я_об’єкту);
```

Приклад використання get_class( ) наведений в лістингу 4.6.

Лістинг 4.6 – Отримання ім’я класу функцією get_class()

```
<?
class Vehicle
 {
...
```

```
class Land extends Vehicle
 {
...
}
// Створити об’єкт з ім’ям $car $car = newLand;
// Змінній $class_a присвоюється рядок "Land"
$class_a = get_class($car);
?>
```

В результаті змінній $class_a присвоюється ім’я класу, на основі якого був створений об’єкт $car:

```
get_parent_class()
```

Функція get_parent_class( ) повертає ім’я батьківського класу (якщо він існує) для об’єкту з заданим ім’ям. Синтаксис функції get_parent_dass():

```
stringget_parent_class (object имя_обьекта);
```

Лістинг 4.7 демонструє використання get_parent_class().

Лістинг 4.7 – Отримання ім’я батьківського класу за допомогою функції get_parent_class ()

```
<?
class Vehicle {
...
}
class Land extends Vehicle {
...
}
// Створити об’єкт з ім’ям $car $car = new Land;
// Змінній $parent присвоюється рядок "Vehicle"
$parent = get_parent_dass($car);
?>
```

При виклику get_parent_class () змінній $parent буде присвоєний рядок "Vehicle":

```
is_subclass_of( )
```

Функція is_subclass_of () перевіряє, чи був об’єкт створений на базі класу, який має батьківський клас з заданим ім’ям. Функція повертає TRUE, якщо перевірка дає позитивний результат, і FALSE в іншому випадку. Синтаксис функції is_subclass_of ():

```
bool is_subclass_of (object объект, string ім’я_класу)
```

Використання is_subclass_of( ) продемонстроване в лістингу 4.8.

Лістинг 4.8 – Використання функції is_subdass_of( )

```
<?
class Vehicle
{
...
}
class Land extends Vehicle
{
...
}
$auto = new Land;
// Змінній  $is_subclass присвоюється  TRUE
$is_subclass = is_subclass_of($auto, "Vehicle");
?>
```

В лістингу 4.8 змінній $is_subclass () присвоюється признак того, чи належить об’єкт $auto до субкласу батьківського класу Vehicle. В наведеному фрагменті $auto відноситься до класу Vehicle; отже, змінній $is_subclass () буде присвоєно значення TRUE.

Функція get_declared_classes ( ) повертає масив з іменами всіх визначених класів (лістинг 4.9). Синтаксис функції get_declared_classes ( ):

```
arrayget_declared_classes ( )
```

Лістинг 4.9 – Отримання списку класів функцією get_declared_classes ()

```
<?
class Vehicle
{
...
}
Class Land extends Vehicle
 {
...
}
$declared_classes = get_declared_classes();
// $declared_classes = array("Vehicle", "Land")
?>
```

4.3 Інтерфейси в PHP

Інтерфейс – набір методів без реалізації. Тобто, в інтерфейс входять методи з ім’ям і вхідними параметрами. Будь-який клас, який реалізує даний інтерфейс, зобов’язаний реалізувати кожний метод. Для прикладу розглянемо аналог з життя. Кожна людина виконую певні функції. Наприклад, навчається, працює, прибирає помешкання, готує їжу та займається іншими очевидними речами. В даному прикладі, інтерфейс – це область занять, наприклад, навчання, робота, прибирання квартири тощо. А методи інтерфейсу – це вже конкретна задача в даній області. Наприклад, в прибиранні квартири можуть бути такі методи: миття посуду чи підлоги, винесення сміття і інші. Для приготування їжі можуть бути використані такі методи: приготування борщу, чищення картоплі та інше.

Інтерфейси – це можливість визначити рід завдань для об'єкта, які він повинен реалізовувати.

Для прикладу був створений наступний інтерфейс:

```
<? php
        interface FileInterface
   {
        public function readFromFile ($path);
        public function writeToFile ($path,$some);
        }
      ?>
```

Даний інтерфейс просто описує роботу з файлом. Відповідно, ті об’єкти, які повинні читати з файлу і записувати різноманітні дані, зобов’язані реалізувати інтерфейс «File Interface».

Створимо ще один інтерфейс:

```
<? php
          interface Client
    {
            public function buy ($id);
            public function repayment($id);
             }
        ?>
```

Представлений вище інтерфейс, реалізує функцію клієнта, тобто є можливість щось купити (function buy ($id)), а також повернути (function repayment($id)) .

Далі, необхідним кроком, було створення класу, який реалізує ці інтерфейси, тобто клас, який містить дві області завдань – бути клієнтом і працювати з файлом (додаток А.6).

Реалізація повинна здійснюватися для кожного методу кожного інтерфейсу. Вона навіть може бути порожньою.

Лістинг 4.10 – Використання класу Shop

```
<? php
          require_once "shop.php";
          $shop = new Shop();
          $shop->buy(5);
          $shop->repayment(5);
    ?>
```

4.4 Перевірка роботи коду за допомогою редактору PHP

PHP – скриптова мова програмування загального призначення, інтенсивно застосовується для розробки веб-додатків. Мова і його інтерпретатор розробляються групою ентузіастів у рамках проекту з відкритим кодом. PHP-скрипти зазвичай обробляються інтерпретатором в порядку, що забезпечує кросплатформеність розробленого додатка:

– лексичний аналіз вихідного коду і генерація лексем,

– синтаксичний аналіз отриманих лексем,

– генерація байт-коду,

– виконання байт-коду інтерпретатором.

PHP Expert Editor – редактор для PHP, Perl, Python, HTML, JavaScript і інших файлів з підтримкою UTF-8. Програма розроблена спеціально для PHP-розробників, в неї інтегрована клієнтська частина відладчика PHP DBG (англ. DBG). Програма має вбудований HTTP-сервер і дозволяє запускати на стороні сервера скрипти на PHP, Perl, Python.

Перевірка синтаксису PHP, вбудований браузер, FTP-клієнт з підтримкою SFTP, оглядач коду і файлів, підтримка проектів, що настроюються шаблони коду, що настроюється підсвічування коду, і багато інших функцій для підвищення зручності розробки.

Розкриємо додаткові функціональні властивості PHP Expert Editor:

– підтримка UTF-8;

- настроювання підсвічування коду;
- згортання коду;
- вбудований браузер;
- вбудований FTP-клієнт з підтримкою SFTP;
- File Explorer з Обраними папками;
- Project Explorer;
- Library Explorer;
- гарячі клавіші і клавіші роботи в редакторі;
- клавіатурні макроси;
- PHP макроси;
- автозбереження;
- перевірка синтаксису PHP;
- запуск скриптів і перегляд результату у вбудованому браузері;
- відладчик;
- для запуску та налагодження PHP-скріптів можна використовувати вбудований або будь-який зовнішній HTTP-сервер;
- підтримка всіх відомих Content-Type. Можливість налагоджувати скрипти, які генерують різний контент, наприклад, картинки;
- швидка вставка всіх функцій PHP з підказкою параметрів;
- швидка навігація в коді за допомогою гарячих клавіш і миші;
- підсвічування парних дужок;
- настроювані шаблони коду для швидкої вставки часто вживаних фрагментів;
- експорт вихідного тексту в HTML та RTF з підсвічуванням;
- закладки;
- підтримка довідки PHP з можливістю пошуку по ключовому слову в поточній позиції;
- keymaping (Default, Classic, Brief, Epsilon, Visual Studio);
- підтримка форматів файлів Windows, Unix, Mac;

– підтримка Perl, Python, Ruby, Tcl. Є можливість використовувати трохи інтерпретаторів, не тільки PHP.

Для відображення процесу перевірки написаного коду за допомогою редактору PHP Expert Editor, був обраний скрипт лістингу (додаток А.4), а саме: «Отримання інформації змінних об’єкту», який представлений на рис. 13.

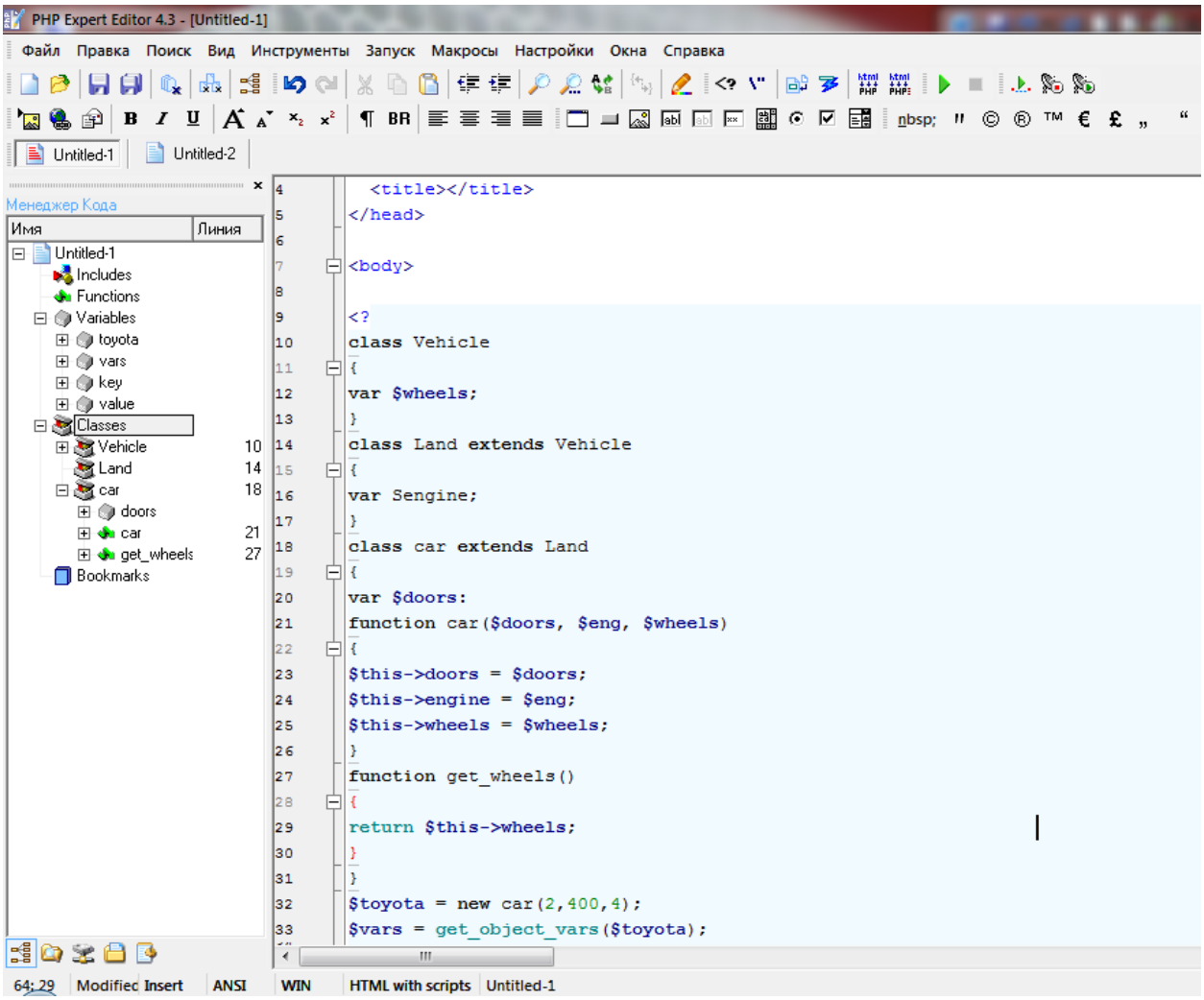

Рисунок 13 – Перевірка правильності написання коду за допомогою редактору PHP Expert Editor

В разі відсутності помилок в процесі проведення перевірки скріпту, що могли міститися у коді, з’являється відповідне повідомлення «Помилок немає». Після чого користувачу представляється можливість запустити скріпт у браузері і отримати результат його виконання.

ВИСНОВКИ

Клас – це базове поняття в ООП. Якщо сказати простіше, то клас – це своєрідний тип змінної. Екземпляр класу – це об'єкт. Об'єкт – це сукупність даних (властивостей) і функцій (методів) для їх обробки. Дані та методи називаються членами класу. Взагалі, об'єктом є все те, що підтримує інкапсуляцію. Кожний реальний предмет – це об'єкт. Прикладами реальних об'єктів є автомобіль, літак, завод, людина, матриця, вектор тощо. Ті самі слова: «автомобіль», «літак», «людина» тощо позначають не об'єкти, а класи, коли йдеться не про конкретний автомобіль, літак або людину, а, наприклад, про автомобіль або літак як різновид транспортного засобу і про людину як біологічний вид.

Клас складається з наступних частин: властивості, конструктор, методи. Класи в PHP мають ряд спеціальних методів. Ці методи є досить гнучким інструментом: перевизначаючи їх, можна добитися істотного зміни поведінки об'єкта.

Отже, клас є абстракцією множини об'єктів, що мають спільні властивості і поведінку. Транспортний засіб і біологічний вид – це теж класи. Відношення між літаком і транспортним засобом або між видом «homosapiens» і біологічним видом взагалі є відношенням успадкування, або відношенням «є» (англ. «is a»): літак є різновидом транспортного засобу, вид «homosapiens» є різновидом біологічного виду. Коли клас B є різновидом класу A, то A називається класом-предком, B – класом-нащадком.

З погляду програмування об'єкт складається з атрибутів і методів. Атрибути описують властивості об'єкта у певний момент часу, методи – властиву для об'єкта поведінку. Всі об'єкти, що є екземплярами одного класу, мають однаковий набір атрибутів і методів. Значення атрибутів зберігаються в змінних, а дії методів описуються в процедурах або функціях Тому клас може бути визначений як набір оголошень змінних-атрибутів і підпрограм-методів. Визначені всередині класу елементи даних називаються змінними-членами

класу, процедури і функції – функціями-членами, або методами класу. Оголошення класу називається його інтерфейсом, а опис його методів – реалізацією. Як правило, інтерфейс класу відокремлюється від його реалізації.

ООП ґрунтується на трьох концепціях: інкапсуляції, успадкування і поліморфізму. Інкапсуляція – це механізм, який дозволяє захистити атрибути й методи об'єкта від некоректного використання. Згідно з принципами інкапсуляції атрибути класу не можуть бути доступними для екземплярів інших класів безпосередньо. Доступ до атрибутів має здійснюватися лише через методи класу. Наприклад, доступ до двигуна автомобіля можна здійснити лише за допомогою методів «завести», «вимкнути», «перемкнути швидкість» тощо.

Саме завдяки інкапсуляції можна отримати зиск у разі відокремлення інтерфейсу класу від його реалізації. Адже інкапсуляція дає можливість зробити програми, що використовують об'єкти певного класу, незалежними від способу реалізації цього класу.

У процесі виконання магістерської роботи – було досягнуто мету роботи – розглянути методи і функції, що використовуються під час створення класів на мові PHP, тобто аналіз моделей формування класів в методології розробки програмного проекту.

При виконанні роботи, можна зробити наступний підсумок, що на сьогоднішній день PHP, незважаючи на свою скромну назва (Personal Home Page – персональна домашня сторінка), – це потужний кросплатформний набір засобів, який розташовується на сервері і призначається для обробки коду, вбудованого в html-документи. Завдяки цьому, з'являється можливість створювати динамічні Web-сторінки.

PHP володіє величезним набором функцій і великою гнучкістю, які можуть бути значно розширені за допомогою додаткових зовнішніх бібліотек. PHP – це об’єктно-орієнтована мова, і можливостей в цьому напрямі дуже багато. Основою ООП є об’єкт, а фабрикою для створення об’єктів є клас.

ПЕРЕЛІК ПОСИЛАНЬ

1. Lecky-Thompson E., Nowicki S. D. Professional PHP6. Boston: Paperback, 2009. 512 p.
2. Ловэйн П. Объектно-ориентированное программирование на PHP. Москва: Вильямс, 2017. 225 с.
3. Zandstr M. PHP Objects, Patterns and Practice. Moscow: Vilyams, 2016. 480 p.
4. Кузнецов М. С., Симдянов И. А. Объектно-ориентированное программирование на PHP. Москва: БХВ, 2018. 608 с.
5. Guttans A., Bakken S., Rethans D. PHP 5 Power Programming. New York: Prentice hall, 2013. 704 p.
6. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Д. Приемы объектно-ориентированного проектирования. Санкт-Петербург: Гамма, 2018. 368 с.
7. Макконелл С. Совершенный код. Москва: БХВ-Петербург, 2018. 510 с.
8. Кухарчик А. Л. PHP 5 на примерах. Москва: Новое знание, 2009. 240 с.
9. Beyross I. PHP 5.1 for Beginners. New York: Sams, 2010. 360 p.
10. Basher A. PHP Essentials. Southgempton: Course Technology, 2008. 326 p.
11. Котеров Д. В., Костарев А.Ф. PHP 5. Санкт-Петербург: БХВ-Петербург, 2005. 1120 с.
12. Коггзолл Дж. PHP 5: Полное руководство. Москва: Издательский дом «Вильямс», 2006. 752 с.
13. Орлов А. А. PHP: Полезные приемы. Москва: Горячая линия-Телеком, 2014. 224 с.
14. Мазуркевич А. М., Еловой Д. М. PHP: настольная книга программиста. Минск: Новое знание, 2013. 480 с.
15. Котеров Д. В. Самоучитель PHP 4. Санкт-Петербург: БХВ-Петербург, 2003. 576 с.

16. Зольников Д. С PHP 5. Как самостоятельно создать сайт любой сложности. Москва: НТ-Пресс, 2007. 272 с.
17. Кухарчик А. PHP: обучение на примерах. Минск: Новое знание, 2014. 237 с.
18. Аргерих Л. И. Профессиональное PHP программирование. Санкт-Петербург: Символ-Плюс, 2013. 1048 с.
19. Кузнецов М. В., Симдянов И. В., Голышев С. В. PHP 5. Практика разработки Web-сайтов. Санкт-Петербург: БХВ-Петербург, 2009. 960 с.
20. Хольцнер С. PHP в примерах. Москва: ООО «Бином-Пресс», 2007. 352 с.
21. Каждан И. А. PHP Expert Editor. Москва: Компьютерная газета, 2008. 600 с.

ДОДАТОК А

Фрагменти лістінгів PHP-коду експериментальної програмної реалізації

А.1 Представлення різних типів транспортних засобів за допомогою наслідування

```
<?
// Транспортний засіб
class Vehicle
{
var $model;
var $current_speed;
function set Speed ($mph)
{
$this->current_speed = $mph;
}
function get Speed()
{
return $this->current_speed;
}
}
// Автомобіль
class Auto extends Vehicle
{
var $fue1_type;
function set Fuel Type ($fuel)
{
$this->fuel_type = $fuel;
}
function get Fuel Type()
{
return $this->fuel_type;
}
}
// Літак
class Airplane extends Vehicle
{
var $wingspan;
function set WingSpan ($wingspan)
{
```

```
$this->wingspan = $wingspan;
}
function get WingSpan()
 {
return $this->wingspan;
}
}
?>
```

А.2 Створення абстрактних класів

```
<?
class Vehicle
{
//Об’явлення атрибутів...
function Vehicle()
 }
die ("Cannot create Abstract Vehicle class!");
}
//Об’явлення інших  методів...
}
class Land extends Vehicle
{
//Об’явлення атрибутів...
function Land()
 }
die ("Cannot create Abstract Land class!");
}
//Об’явлення інших  методів..
 }
class Car extends Land
 {
//Об’явлення атрибутів...
//Об’явлення методів...
}
?>
```

А.3 Перевантаження методів

```
<?
class Page
{
var $bgcolor;
var $textcolor;
function Page()
{
// Визначити кількість переданих аргументів і викликати метод з
потрібним ім’ям
$name = "Page".func_num_args();
// Call $name with correct number of arguments passed in
if ( func_num_args() == 0 ) :
$this -> $name();
else :
$this->$name (func_get_arg(0));
end if;
}
function Page0() {
$this->bgcolor = "white";
$this->textcolor = "black";
print "Created default page";
}
function Page1($bgcolor)
{
$this -> bgcolor = $bgcolor;
$this -> textcolor = "black";
print "Created custom page";
}
}
$html_page – new Page("red");
?>
```

А.4 Отримання інформації змінних об’єкту

```
<?
class Vehicle
{
```

```
var $wheels;
}
class Land extends Vehicle
{
var Sengine;
}
class car extends Land
{
var $doors:
function car ($doors, $eng, $wheels)
{
$this->doors = $doors;
$this->engine = $eng;
$this->wheels = $wheels;
}
Function get_wheels()
{
return $this->wheels;
}
}
$toyota = new car(2,400,4);
$vars = get_object_vars($toyota);
while (list($key, $value) = each($vars)) :
print "$key ==> $value<br>";
end while;
// Вихідні данні:
// doors ==> 2
// engine ==> 400
// wheels ==> 2
?>
```

А.5 Перевірка підтримки методу об’єктом

```
<?
classVehicle
 {
...
}
class Land extends Vehicle
```

```
{
var $four Wheel;
function set Four Wheel Drive ()
 {
$this->four Weel = 1;
}
}
// Створити об’єкт з ім’ям $car
$car = new Land;
// Якщо метод "four Wheel Drive" підтримується класом
"Land"
// або "Vehicle", виклик method_exists повертає TRUE;
// в іншому випадку повертає FALSE.
// В даному прикладі method_exists() повертає TRUE.
if (method_exists($car, "set four Wheel Drive")) :
print "This car is equipped with 4-wheel drive";
else :
print "This car is not equipped with 4-wheel drive";
end if;
?>
```

А.6 Створення класу, який реалізує інтерфейси та містить області завдань

```
<? php
     require_once "fileinterface.php";
     require_once "client.php";
     class Shop implements FileInterface,Client\
 {
       public function readFromFile($path)
  {
       echo "Зчитуємо з файлу і повертаємо рядок <br />";
       }
       public function writeToFile($path,$some)
  {
       echo "Записуємо в файл дані $some<br />";
       }
       public function buy($id)
```

```
       {
            echo "Дякую за покупку <br />";
            $this->writeToFile("data.db", "Був куплений товар
$id");
            }
            public function repayment($id)
      {
           $this->readFromFile("data.db");
           //Перевірка того, чи була насправді здійснена покупка
товару $id
           $this->writeToFile("data.db", "Було виконане повер-
нення товару $id");
           }
           }
?>
```