

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ
УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Кафедра «Програмна інженерія»

ЗВІТ

до практичного заняття №1 з дисципліни

«Аналіз та рефакторинг коду»

На тему: «Правила оформлення програмного коду»

Виконав:

ст. гр. ПЗП-22-7

Колесник Олександр Андрійович

Прийняв:

ст. викладач кафедри ПІ

Сокорчук Ігор Петрович

Харків 2024

1 МЕТА

Навчитися рефакторингу програмного коду, закріпити основні правила оформлення коду.

2 ЗАВДАННЯ

Обрати мову програмування для прикладів коду. Створити презентацію на тему «Правила оформлення програмного коду».

3 ХІД РОБОТИ

Було обрано мову програмування РНР. У презентації (Додаток А) наведено основні рекомендації щодо оформлення програмного коду з описами, а також приклад коду з та без використання цих рекомендацій.

ВИСНОВКИ

Отже, у ході роботи було набуто навичок рефакторингу програмного коду, детально розглянуто основні правила оформлення коду.

ДОДАТОК А

Презентація на тему «Правила оформлення програмного коду».

ПРЕЗЕНТАЦІЯ НА ТЕМУ: “ОСНОВНІ РЕКОМЕНДАЦІЇ НАПИСАННЯ КОДУ ДЛЯ РНР”

Презентацію підготував:
Колесник Олександр Андрійович
з групи ПЗПІ-22-7

ВСТУП

PHP є однією з найпопулярніших мов програмування для веброзробки, яка забезпечує динамічність та інтерактивність вебсайтів. Однак, для забезпечення спільної роботи над проєктами та зручності підтримки коду, важливо дотримуватися стандартів написання коду. Основними такими стандартами є PSR-1 та PSR-12, які забезпечують уніфікованість та читабельність коду.

```
10 2 references | 0 implementations
11 class ChatPage extends Page
12 {
13     1 reference
14     private $host = "127.0.0.1";
15     1 reference
16     private $port = 20205;
17
18 2 references
19 private function getStr(): string
20 {
21     $dbPath = 'lw1.db';
22     $conn = new SQLite3(filename: $dbPath);
23
24     $query = "SELECT creation_date, sender, text FROM message";
25     $result = $conn->query(query: $query);
```


ОСНОВИ СТИЛЮ

PSR-1 визначає базові правила написання коду на PHP. Одним із ключових моментів є використання англійської мови для іменування змінних та класів, що підвищує зрозумілість коду. Наприклад, змінні повинні мати логічні імена, англійською мовою. Крім того, рекомендується відокремлювати логічні блоки коду новими рядками для покращення читабельності.

```
//Поганий приклад
$ИмяПользователя = "John"; // Імена змінних не англійською
$validationdata = true; // Не дотримано camelCase

//Гарний приклад
$username = "John"; // Імена англійською, читабельні
$validationData = true; // Дотримано camelCase
```


ВІДСТУПИ

Згідно з PSR-12, для відступів слід використовувати чотири пробіли, а не табуляцію. Це дозволяє забезпечити уніфікований стиль кодування та уникнути проблем з відображенням коду в різних редакторах. Відступи повинні відповідати рівню вкладеності конструкцій.

```
//Поганий приклад
if($x > 10){
echo "X більше 10";
}

//Гарний приклад
if ($x > 10) {
    echo "X більше 10"; // Використано відступи з чотирьох пробілів
}
```


СКОРОЧЕННЯ ВКЛАДЕНОСТІ

```
//Поганий приклад
if ($a > 0) {
    if ($b > 0) {
        if ($c > 0) {
            echo "Всі змінні більше нуля";
        }
    }
}

//Гарний приклад
if ($a > 0 && $b > 0 && $c > 0) {
    echo "Всі змінні більше нуля"; // Скорочено вкладеність
}
```

Щоб код був зрозумілим і легким для підтримки, важливо зменшувати вкладеність логічних блоків та уникати дублювання коду. Глибока вкладеність ускладнює розуміння логіки програми, тому краще розділяти великі блоки на менші функції або методи, що підвищує модульність та читабельність.

ДОКУМЕНТУВАННЯ ТА КОМЕНТУВАННЯ КОДУ

Кожен програміст повинен залишати коментарі у коді для пояснення складних або неочевидних частин. Це важливо як для інших розробників, що працюватимуть з кодом, так і для самого автора коду, який через деякий час може не пам'ятати всі деталі. Коментарі мають бути короткими, інформативними та зрозумілими.

```
//Поганий приклад
// ця функція робить щось
0 references
function doSomething(): void
{
    // код функції
}

//Гарний приклад
/**
 * Виконує зведення у квадраті вхідного значення
 *
 * @param int $input Вхідне значення
 * @return void
 */
0 references
function doSomething($input): void
{
    // код функції
}
```


ДОВЖИНА РЯДКІВ

У PSR-12 зазначається, що м'яке обмеження для довжини рядка становить 120 символів. Якщо рядок довший, краще використовувати конкатенацію або перенос рядка. Такий підхід дозволяє зберегти чистоту коду та зручність для читання, особливо в малих редакторах коду.

```
//Поганий приклад
$veryLongString = "Це дуже довгий рядок, який перевищує обмеження в 120 символів, [i] його важко читати та відслідковувати пі

//Гарний приклад
$veryLongString = "...Це дуже довгий рядок, який перевищує обмеження в 120 символів, " .
"[i] його важко читати та відслідковувати під час редагування.";
```


ОПЕРАТОРИ ТА КЛЮЧОВІ СЛОВА

Для підвищення читабельності коду між операторами та операндами, як і між ключовими словами та їх умовами, слід залишати пробіли.

```
//Поганий приклад
if($x>5){echo"X більше 5";}

//Гарний приклад
if ($x > 5) {
    echo "X більше 5";
}
```


ІМЕНУВАННЯ КЛАСІВ ТА МЕТОДІВ

Імена класів повинні використовувати стиль CamelCase, де кожне слово починається з великої літери, наприклад, UserProfile. Для методів і функцій застосовується стиль camelCase, де перше слово починається з малої літери, а наступні – з великої, як у прикладі getUserData().

```
//Поганий приклад
0 references | 0 implementations
class userProfile {
    0 references | 0 overrides
    function getUserdata(): void {
    }
}

//Гарний приклад
0 references | 0 implementations
class UserProfile {
    0 references | 0 overrides
    function getUserData(): void {
    }
}
```


ВІДКРИВАЮЧІ ТА ЗАКРИВАЮЧІ ДУЖКИ

Згідно з рекомендаціями, відкриваючі фігурні дужки у визначенні класів або методів мають розташовуватися на новому рядку, а закриваючі – на наступному рядку після тіла конструкції. Це правило стосується також керуючих конструкцій, таких як `if`, `else` або `switch`.

```
//Поганий приклад
if ($x > 5)
{
    echo "X більше 5"; }

//Гарний приклад
if ($x > 5) {
    echo "X більше 5";
}
```


ФОРМАТУВАННЯ МАСИВІВ

Для підвищення читабельності великі масиви слід формувати таким чином, щоб кожен елемент розташовувався на новому рядку. Це полегшує розуміння структури масиву та робить його зручнішим для змін.

```
//Поганий приклад
$array = array("ID" => 123, "Name" => "John", "Age" => 25);

//Гарний приклад
$array = array(
    'ID' => 123,
    'Name' => 'John',
    'Age' => 25
);
```


ДЯКУЮ ЗА УВАГУ

СПИСОК ДЖЕРЕЛ:
СТАНДАРТИ, СТИЛІ ТА ПРАВИЛА
ОФОРМЛЕННЯ КОДУ PHP
PSR-1: BASIC CODING STANDARD
PSR-12: EXTENDED CODING STYLE